

Query Management Facility™



Installing and Managing QMF on VM/ESA

Version 7

Query Management Facility™



Installing and Managing QMF on VM/ESA

Version 7

Note!

Before using this information and the product it supports, be sure to read the general information under "Appendix E. Notices" on page 327.

Second Edition (September 2000)

This edition applies to Query Management Facility, a feature of Version 7 Release 1 of DATABASE 2 Server for VM and VSE< (DB2for VM and VSE), 5697-F42, (VM environment only), and to any subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces GC26-9573-00.

© **Copyright International Business Machines Corporation 1983, 2000. All rights reserved.**

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

Contents

The QMF Library	ix	Planning for Installing QMF into a Workstation Database Server	20
About This Book	xi	Chapter 3. Installing QMF 7.1 into the DB2 for VM Database	21
How to Use This book.	xi	QMF Installation Flow Diagram	21
What You Should Know Before You Begin	xii	The Installation Steps	24
Locating Prerequisite Documentation	xii	Preliminary: Read the Program Directory and Complete the QMF 7.1 Worksheet	24
Part 1. Installing QMF for VM/ESA	1	Step 1—Create QMF Installation Control File: DSQ2ECTL.	25
Chapter 1. Introduction	3	Step 2—Create DB2 for VM DBSPACES: DSQ2DBSC	26
Overview of QMF	3	Step 3—Run QMF Installation EXEC: DSQ2EINS	28
QMF Objects	3	Step 4—Start QMF: DSQ2EINV	29
Overview of QMF with Remote Unit of Work.	4	Step 5—Run the IVP for QMF Interactive Mode : DSQ2EIVP	36
Some Terminology	5	Step 6—Installing the QMF Sample Objects and Application Objects: DSQ2ESQD and DSQ2ESQI	37
Overview of the Installation Process	5	Step 7—Running the Batch-Mode IVP (Optional): DSQ2EBAT	38
Where the Objects Reside	5	Step 8—Deleting Previous Versions of QMF (Optional): DSQ2BDEL	40
Local and Remote Installation	5	Step 9—Post-Installation Cleanup	40
Connecting to a Remote Database from VM	5	Step 10—Load QMF Database Packages to a Remote Server (Optional): DSQ2BPKB.	41
Connecting to Workstation Database Servers from VM	6	Chapter 4. Installing a QMF 7.1 National Language Feature (NLF)	43
Chapter 2. Planning for Installation	7	NLF Installation EXECs	43
Hardware Requirements	7	Installing a National Language Feature	43
Prerequisite Software	7	Hardware and Program Product Requirements	44
Products Required to Support Remote Unit of Work	11	The Installation Steps	44
Virtual Storage Requirements	12	Preliminary: Read the NLF Program Directory and Complete the Worksheet	44
Discontiguous Shared Segments (DCSS) Storage Requirements	12	Step 1—Create the QMF NLF Installation Control File: DSQ2nCTL	45
Disk Storage Requirements	12	Step 2—Run QMF NLF Installation EXEC: DSQ2nINS	46
Required DB2 for VM Knowledge.	12	Step 3—Start QMF NLF: DSQ2nINV	47
DB2 for VM Requirements for QMF	13	Step 4—Run the IVP for QMF NLF Interactive Mode: DSQ2nIVP	48
A PUBLIC DBSPACE is Required for Saving Data	13		
Database CONNECT ID “Q” and “SQLDBA”	14		
QMF SQL Install Packages	14		
Further Requirements	14		
Before You Begin	18		
Previous Releases of QMF	18		
Migration and Fallback	18		
QMF National Language Feature (NLF) Considerations	19		

Step 5—Install QMF NLF Sample Objects and Application Objects: DSQ2nSQD and DSQ2nSQI	49
Step 6—Run the IVP for QMF NLF Batch Mode (Optional): DSQ2nBAT	50
Step 7—Post-Installation Cleanup	50

Part 2. Managing QMF for VM/ESA 51

Chapter 5. Starting QMF 57

Before you Start QMF	57
Establishing a Database Connection	57
Initializing the QMF Session	58
Quick Start	58
Setting up QMF to Run under ISPF	59
Before you start QMF	59
Starting QMF from a Menu Option	59
Starting QMF with the ISPSTART Command	61
Starting QMF in Batch Mode in ISPF	62
Examples of Starting QMF under ISPF	63
Setting up QMF to Run under CMS	64
Starting QMF Directly with the DSQQMFE Module	64
Starting QMF in a Batch CMS Environment	64
Examples of Starting QMF under CMS	65
Creating a CMS EXEC	65
Verify Program Modules	65
Verify QMF Data Files	65
GDDM Considerations	66
DB2 for VM Considerations	66

Chapter 6. Customizing Your Start Procedure 67

Quick Start	67
Setting Default Start Values Using the REXX Program DSQSCMDn	68
Naming the Program Segment	72
dcssname	72
DSQSDCSS	73
Customizing Report Storage and Report Performance	73
Adjusting Storage for Report Data (DSQSBSTG)	73
Adjusting Reserved Storage Used for Report Data (DSQSRSTG)	74
Acquiring Extra Storage (DSQSPILL)	75

Controlling the Number of Report Rows Retrieved for Display (DSQSIROW)	79
Setting the Level of Trace Detail (DSQSDEBUG)	81
Controlling Initial Activities During a Session	82
Specifying the Location to Connect to When Starting QMF (DSQSDBNM)	82
Specifying an Interactive or Noninteractive QMF Session (DSQSMODE)	83
Naming a Procedure to Run When QMF Starts (DSQSRUN)	84
Setting Printing for Double-Byte Character Set Data (DSQSDBCS)	90

Chapter 7. The QMF Session Control Facility 91

Installing or Removing Q.SYSTEM_INI	91
Importing the Default System Initialization Procedure	91
When Does the Q.SYSTEM_INI Procedure Run?	91
Using Q.SYSTEM_INI	92
Example Shipped with QMF	92
User Session Procedure Example	92
Procedure that Displays an Object list	93
Security and Sharing Session Procedure	94
Diagnosis Considerations	94

Chapter 8. Establishing QMF Support for End Users 95

The role of the Q.AUTHID	95
Quick Start	95
Ensuring That Users Have Access to CMS	96
Creating User Profiles to Enable User Access to QMF	97
Using the Q User Profile, a Special QMF Profile	97
Establishing a Profile Structure for Your Installation	98
Adding a New User Profile to the Q.PROFILES Table	98
Preventing Users Without Unique Profiles from Using QMF	99
Reading the Q.PROFILES Table	100
Providing the Correct Profile for the User's Operating Environment	104
Updating User Profiles	105
Deleting Profiles from the Q.PROFILES Table	106

Controlling Access to QMF and Database Objects	107	Allocating and Reallocating Resources	
SQL Privileges Required to Access Objects	107	Using EXECs	130
Granting and Revoking SQL Privileges	109	Preparing the Allocation EXEC	130
Sharing QMF Objects with Other Users	111	Preparing the Reallocation EXEC.	135
Allowing Uncommitted Read	111	Other Allocation Methods	138
Setting Standards for Creating Objects	112	Customizing the Document Editing Interface for Users.	139
Customizing a User's Database Object List	112	Changing the Application	139
Using the Default Object Lists.	113	Renaming the Document Interface Macros and EXEC	139
Changing the Default List	114	Placing the Q.DSQAED2S Procedure in the Database	139
Object List Storage Requirement	115	Transferring Ownership to Q	140
Enabling Users to Create Tables in the Database.	116	Changing the Data Components	140
Choosing and Acquiring a dbspace for the User	118	Changing the EXECs and Macros	142
Granting a User DB2 for VM RESOURCE Authority	118	Customizing the QMF Edit Command	143
Enabling Users to Confirm Table Changes Before They are Made	119	Enabling English Support in an NLF Environment	144
Enabling Users to Support a Chart	120	Using Global Variables to Define the Currency Symbol	145
Maintaining QMF Objects Using QMF Control Tables	120	Chapter 9. Enabling Users to Print Objects	147
Reading the Q.OBJECT_DIRECTORY Table	121	Quick start	147
Reading the Q.OBJECT_DATA Table	122	Printing Objects	148
Reading the Q.OBJECT_REMARKS Table	123	Deciding Whether to Use QMF or GDDM Services for Printing	149
Listing QMF Queries, Forms, and Procedures	123	Using GDDM Services to Handle Printing	149
Displaying QMF Queries, Forms, and Procedures	124	Choosing a GDDM Nickname for Your Printer	150
Transferring Ownership of Queries, Forms, and Procedures	124	Creating the Nickname Specification	151
Deleting Obsolete Queries, Forms, and Procedures	125	Testing the Nickname Definitions in External Default Files	154
Enlarging the dbspace for the QMF Object Control Tables	126	How QMF Interfaces with Your GDDM Nickname	154
Maintaining Tables and Views Using DB2 for VM System Tables	127	Using QMF's DSQPRINT to Handle Printing	155
Listing Tables and Views	128	Defining a Synonym for the Print Function Key	156
Transferring Ownership of a Table or View	128	Updating User Profiles to Enable GDDM Printing	156
Deleting a Table or View from the Database.	128	Chapter 10. Customizing QMF Commands	159
Supporting Locally Defined Date/Time Formats	128	Quick Start	159
Accessing the DXT End User Dialogs (ISPF Only)	129	Using the Default Synonyms Provided with QMF	159
Supporting the EXTRACT Command	129	Displaying Printed Reports (DPRE)	160
Allocating Resources.	129	Creating a Command Synonym Table	162
		Entering Command Synonym Definitions into a Command Synonym Table.	163
		Choosing a Verb	163

Choosing an Object Name	165	Generating Your Program	203
Choosing the Synonym Definition	165	Writing an Edit Routine in PL/I without	
Activating the Synonyms	168	Language Environment (LE)	203
Minimizing Maintenance of Command		How a PL/I Edit Routine Interacts with	
Synonym Tables	170	QMF	204
Assigning One Synonym Table to all		Compiling Your Program	209
Users	170	Creating Your DSQUEDIT Module File in	
Assigning Views of a Synonym Table to		PL/I	210
Individual Users	170	Writing an Edit Routine in PL/I with	
Chapter 11. Customizing QMF Function		Language Environment (LE)	211
Keys	173	Generating Your PL/I Program for LE	212
Quick Start	173	Writing an Edit Routine in COBOL without	
Choosing the Keys You Want to Customize	173	Language Environment (LE)	213
Default Keys on Full-screen Panels	174	How a COBOL Edit Routine Interacts	
Default Keys on Window Panels	175	with QMF	214
Creating the Function Key Table	176	Compiling Your Program	219
Entering Your Function Key Definitions into		Assembling the Run Time Options Macro	
the Table	177	(COBOL II)	220
Linking a Command with a Function Key	177	Generating Your Program	220
Labeling the Function Key and		Writing an Edit Routine in COBOL with	
Positioning it on the Screen	179	Language Environment (LE)	221
Examples of Key Definitions	179	Generating Your COBOL Program for LE	222
Identifying the Panel You Want to Customize	181	Handling Double-Byte Character Set Data	223
Full-screen Panel Identifiers	181	Edit Codes for DBCS Data	223
Window Panel Identifiers	181	What the Edit Routine Receives	223
Activating New Function Key Definitions	184	Ensuring the Edit Routine Returns the	
		Right Results	224
Chapter 12. Creating Your Own Edit		Chapter 13. Controlling QMF Resources	
Codes for QMF Forms	187	Using a Governor Exit Routine	227
Quick Start	187	Quick start	227
Choosing an Edit Code	188	Using the IBM-Supplied Governor Exit	
Handling DATE, TIME, and TIMESTAMP		Routine	228
Data Types	189	Activating the Default Limits	229
Calling Your Exit Routine to Format the		How a Governor Exit Routine Controls	
Data	191	Resources	230
Passing Information to and from the Exit		Defining Your Own Resource Limits	233
Routine	193	Creating your own Resource Control	
Fields of the Interface Control Block	193	Table	236
Fields That Characterize the Input Area	195	Modifying the IBM-supplied Governor Exit	
Fields That Characterize the Output Area	196	Routine or Writing Your Own.	238
Passing Control to the Exit Routine When		Program Components of the Governor	
QMF Terminates	197	Exit Routine	239
Writing an Edit Routine in High-Level		How CMS Interacts with the Governor	
Assembler (HLASM) or Assembler	197	Exit Routine	240
How an Assembler Edit Routine Interacts		How and When QMF Calls the Governor	
with CMS	198	Exit Routine	241
How an Assembler Edit Routine Interacts		Passing Resource Control Information to	
with QMF	199	the Governor Exit.	244
Assembling Your Program	202		

Storing Resource Control Information for the Duration of a QMF Session	258
Canceling User Activity	259
Providing Messages for Canceled Activities	260
Assembling and Generating Your Governor Exit Routine	261
Assembling Your Governor Exit	261
Building a Module File or Creating a Load Library Member	262
Chapter 14. Customizing a Remote Database Connection	263
Quick Start	263
Determining the Remote Database Connection Needed	264
Connecting with Remote Unit of Work	265
Connecting with DB2-to-DB2 Distributed Unit of Work	265
Verifying the Connections Necessary for Remote Unit of Work	266
Checking DB2 for VM Connections	266
Checking DB2 for VM Connections	266
Preparing a Non-DB2 for VM Location for Access by QMF VM Users	267
Creating Command Synonym Tables	267
Preparing QMF to Support the DPRE Command	269
Preparing QMF to Support Other Commands	269
Creating Function Key Tables	269
Updating QMF Governor Control Tables	270
Installing the National Language Feature in the QMF Server	270
Code Page Support	270
Enabling Your Users to Access a Remote Database	271
Updating a User's Profile	271
Specifying Access for Current SQL Authorization ID	271
Connecting to the Local Database	271
Connecting to the Remote Database	271
Specifying a Location Name	272
Where Data Must be Located for User Access	273
Preventing SQL Errors	274
Translating User IDs	275
Deleting QMF Users from Each Remote QMF Location	275

Enabling Administrator Access to Your Location	275
--	-----

Chapter 15. Customizing the Batch Processing Program	277
Quick Start	277
Enabling Your Users to Use Batch Mode	278
Sending a Job to the CMS Batch Machine	279
Running Batch Jobs on Your Machine	281
Debugging a Procedure	282
Using the QMF Batch Query/Procedure Application (BATCH)	282
MACLIBs Required	283
Using the Application	283
Filling in the Prompt Panel	283
Modifying the Batch Application	286

Chapter 16. Troubleshooting and Problem Diagnosis	289
Quick Start	289
Troubleshooting Common Problems	290
Handling Initialization Errors	290
Handling Warning Messages	291
Handling GDDM Errors During Printing	292
Handling QMF Errors During Printing	292
Handling CMS Command Errors	294
Handling Display Errors	295
Solving Slow Performance Problems	296
Determining the Problem Using Diagnosis Aids	298
Choosing the Right Diagnosis Aid for the Symptoms	298
Diagnosing Your Problem Using QMF Message Support	298
Using the QMF Trace Facility	300
Abend Handling	305
Using the QMF Interrupt Facility	306
Using Error Log Reports from the Q.ERROR_LOG Table	308
Reporting a Problem to IBM	309
Using ServiceLink to Search for Previously Reported Problems	310
Working with Your IBM Support Center	312

Part 3. Appendixes 313

Appendix A. Installation Checklists	315
QMF Installation Checklist	315
QMF NLF Installation Checklist	316

Appendix B. QMF Objects Residing in DB2 for VM	317
Input to DSQ2EINS or DSQ2nINS	317
QMF User ID	317
QMF Control Tables	317
Default List Views	318
QMF Packages	318
NLF Parts	318

Appendix C. Migration and Fallback Considerations	319
Migrating from a Previous QMF Release to QMF 7.1	319
Global Variables and the Governor	319
Use of the Invocation Procedure	319
Q.VPROFILE	319
Multiple Releases of QMF	320
Migrating to a new DB2 for VM level	320
Migration and 31-Digit Decimal Support	321
Fallback	321
Re-establishing the Earlier Profiles	322
Using Version 7 Objects Under Earlier QMF Releases	322
Using Version 7 QMF Commands with Earlier Releases	323
31-Digit Decimal Support	324

Appendix D. QMF Control Tables and dbspaces Used by QMF	325
--	------------

Appendix E. Notices	327
Trademarks	330

Glossary of Terms and Acronyms	331
---------------------------------------	------------

Bibliography	345
APPC Publications	345
CICS Publications	345
COBOL Publications	346
DATABASE 2 Publications	346
DCF Publications	347
DRDA Publications	347
DXT Publications	347
Graphical Data Display Manager (GDDM) Publications	347
HLASM Publications	347
ISPF/PDF Publications	347
OS/390 Publications	348
PL/I Publications	348
REXX Publications	348
ServiceLink Publications	348
VM Publications	349
VSE Publications	349

Index	351
--------------	------------

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.

Evaluating

Introducing
QMF

GC27-0714

Installing, planning for, administering, and diagnosing

Installing
and
Managing
QMF on
OS/390

GC27-0719

Installing
and
Managing
QMF on
VM/ESA

GC27-0720

Installing
and
Managing
QMF on
VSE/ESA

GC27-0721

Installing
and
Managing
QMF for
Windows

GC27-0722

QMF
Messages
and Codes

GC27-0717

QMF High
Performance
Option User's
Guide for
OS/390

SC27-0724

Using

Using
QMF

SC27-0716

QMF
Reference

SC27-0715

Getting
Started
With QMF
for Windows

SC27-0723

Application programming

Developing
QMF
Applications

SC27-0718

Online libraries



SK2T-0730
OS/390, VM,
& VSE



SK2T-6700
OS/390 only



SK2T-2067
VM only



SK2T-0060
VSE only

About This Book

Installing and Managing QMF on VM/ESA helps you install and maintain the Query Management Facility (QMF) product under the Virtual Machine/Enterprise System Architecture (VM/ESA[®]) operating system.

This book is written for VM/ESA system programmers responsible for installing and maintaining QMF with the DB2[®] for VM relational database. It is also designed for network administrators responsible for installing and maintaining network applications. References to "Workstation Database Servers" in this book apply to:

- DB2 Common Server V2
- DB2 Parallel Edition for AIX[®] V1.2
- DataJoiner[®] V1.2.1 and V2
- DB2 Universal Database V5

How to Use This book

The administration and customization tasks in this book assume QMF was installed according to procedures in Part 1, "Part 1. Installing QMF for VM/ESA" on page 1.

Most of the administration and customization tasks shown in this book are done using the QMF product itself. Therefore, before you begin the tasks in this book, check with the system installer to see if the installation verification procedure (IVP) has been run. If not, run the IVP yourself 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 presented in Part 1, "Part 1. Installing QMF for VM/ESA" on page 1.

Most of these tasks require that you have DB2 for VM database administrator (DBA) authority. If the program installer followed the default procedure in Part 1, "Part 1. Installing QMF for VM/ESA" on page 1, the user ID Q was defined for you during QMF installation. This user ID has DBA authority.

Each chapter in Part 2, "Part 2. Managing QMF for VM/ESA" on page 51, includes a section called "Quick start". Use these sections to get an overview of how to accomplish a certain task. After you read the quick start section to understand all the steps involved in the task, see the page indicated if you need more information on how to perform each step.

About This Book

What You Should Know Before You Begin

The tasks explained in this book assume you have a working knowledge of the following products:

- VM/ESA, an operating system under which QMF runs.
- Conversational Monitor System (CMS), a environment in which QMF runs. It manages the communication with your terminal.
- Interactive System Productivity (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.
- DATABASE2 for VM/ESA (DB2 for VM), a database manager for QMF.
- Data Extract (DXT™), a facility that can supply the DB2 load utility with data.
- Assembler programming language, which you need if you plan to modify the IBM-supplied governor exit routine or write one of your own. You might also use HLASM or assembler if you plan to create your own edit codes in assembler for QMF forms.
- PL/I, which you might use if you plan to create your own edit codes in PL/I for QMF forms.
- VS COBOL II or COBOL/370™, which you might use if you plan to create your own edit codes in COBOL for QMF forms.
- Restructured Extended Executor (REXX) language, and
- A general knowledge of the structure and function of QMF

Publications that discuss these products are listed in “Bibliography” on page 345.

Additionally, you might want to become familiar with some of the end-user functions provided by QMF. The QMF end-user functions are explained in *Using QMF*. Order numbers for this and other QMF publications are listed on page “The QMF Library” on page ix.

Locating Prerequisite Documentation

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

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

The *QMF Program Directory* documents how to install QMF from tape to disk. It also documents changes to the install process after this book is published. You'll find it packed in the shipping carton with your installation tape.

“Appendix C. Migration and Fallback Considerations” on page 319, explains how to migrate objects from earlier versions and releases of QMF.

For a list of QMF publications, see “The QMF Library” on page ix. Publications from other IBM product families are found in the “Bibliography” on page 345.

Part 1. Installing QMF for VM/ESA

Chapter 1. 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 7.1, 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

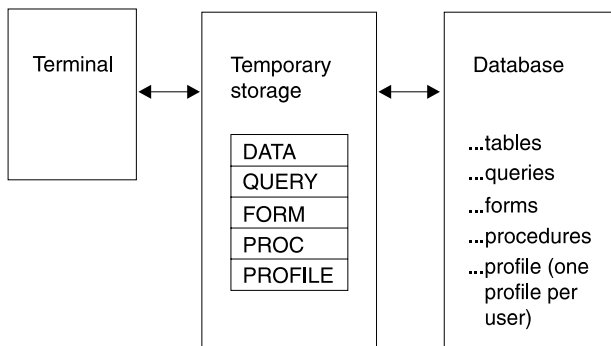
Contains one or more QMF commands that can be run as a group.

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:

Introduction



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 AS/400[®] 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.

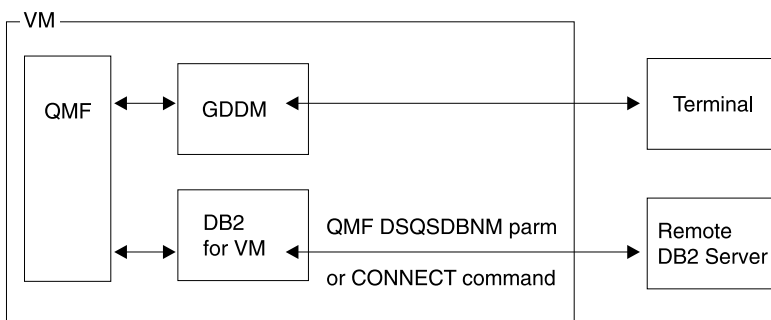


Figure 1. QMF Relationship to VM, DB2 for VM, and GDDM

Note: Before you can connect to a location you must have QMF installed in the database at that location.

Some Terminology

You are installing QMF Version 7 Release 1 (for brevity referred to as QMF 7.1). We also use “VnRn” to point out earlier releases of QMF: For example, a “QMF V2R4 form” is a QMF form that was created under QMF Version 2 Release 4. Where “QMF” appears without a qualifier (For example, “QMF will run on ...”) we mean QMF 7.1.

Overview of the Installation Process

QMF installation involves three object groups:

1. QMF load modules
2. QMF control tables, catalog views, and sample tables
3. QMF SQL packages

Where the Objects Reside

The load modules are saved into a discontinuous 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 DB2 database from VM (with the DSQSDBNM startup parameter or the CONNECT command) perform the following task:

- From the OS/390 system, install the QMF control tables, catalog views, sample tables, and packages/plan in the DB2 database you want to connect to.

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 you plan to connect to DB2 for VM databases from OS/390 (via the DSQSDBNM startup parameter or the CONNECT command) perform the following tasks:

- From OS/390, use the requester-database option to install the QMF load modules in OS/390.

Introduction

- From VM, install the QMF control tables, catalog views, sample tables, and packages/plan in the DB2 for VM database you want to connect to. You can do this with a database-only installation.

Connecting to Workstation Database Servers from VM

After installing QMF for VM, you can connect to a workstation database server from VM. To do so, install the QMF control tables, catalog views, sample tables, and packages in the workstation database server you want to connect to. You can do this with a database-only installation.

Chapter 2. Planning for Installation

This chapter describes the hardware, program products, and storage required to install and run QMF. It presents an installation planning overview. For installation details see “Chapter 3. Installing QMF 7.1 into the DB2 for VM Database” on page 21.

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 the Graphical Data Display Manager (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 7.1. Later releases that are not available at the QMF Version 7.1 announcement time are not supported unless specifically stated otherwise.

Table 1. Prerequisite Software For QMF For VM/ESA Version 7.1

Required product	Version and release	Number
IBM VM/ESA	Version 2 Release 2.0	5654-030
SQL/DS for VM	Version 3 Release 5	5688-103
GDDM/VMXA or	Version 2 Release 3	5684-007
GDDM/VM	Version 3 Release 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. Later releases that are not available at the QMF Version 7.1 announcement time are not supported unless specifically stated otherwise.

Planning for Installation

Table 2. Prerequisite software for optional functions for QMF for VM Version 7 Release 1

Product	Version and release	Number
ISPF	Version 3 Release 2	5684-043
CHARTS (Interactive Chart Utility):		
GDDM — PGF (for GDDM/VMXA Version 2 Release 3) or	Version 2 Release 1.1	5668-812
GDDM — PGF (for GDDM/VM Version 3 Release 1.1)	Version 2 Release 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 Release 2	5684-123
QMF Document Interface:		
VM/SP System Product Editor (XEDIT)		
IBM OfficeVision/VM	Version 1 Release 2	5684-084
ISPF/Program Development Facility for VM	Version 3 Release 2	5684-123
Callable Interface Programs using the callable interface can be written in:		
IBM C/370 Compiler and C/370 Library	Version 2	5688-187
IBM HLASM	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	Version 2 Release 5	5688-806
(REXX and the SAA callable interface for FORTRAN are not supported in the QMF/CICS environment.)		

Table 2. Prerequisite software for optional functions for QMF for VM Version 7 Release 1 (continued)

Product	Version and release	Number
OS PL/I	Version 2 Release 2.3	5668-909
IBM PL/I for MVS and VM	Version 1 Release 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 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
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
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
IBM COBOL for MVS and VM	Version 1 Release 2	5688-197
OS PL/I	Version 2 Release 3	5668-909
IBM PL/I for MVS and VM	Version 1 Release 1.1	5688-235
Assembler H or standard assembler	Version 2 Release 1	5668-962
Governor Exit Routine		
IBM HLASM	Version 1	5696-234
QMF for Windows:		
Microsoft Windows** or	Version 3 Release 1	

Planning for Installation

Table 2. Prerequisite software for optional functions for QMF for VM Version 7 Release 1 (continued)

Product	Version and release	Number
Microsoft Windows** for Workgroups or	Version 3 Release 1 or Release 1.1	
Microsoft Windows 95 or		
Microsoft Windows NT		
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 Release 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 Release 5	5688-103
QMF for VM	Version 7	5697-F42
At the remote DB2 for VM database:		
SQL/DS for VM	Version 3 Release 5	5688-103
QMF for VM	Version 7	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 Release 5	5688-103
QMF for VM	Version 7	5697-F42
At the remote DB2 for MVS/ESA location:		
DB2 for MVS	Version 3 Release 1	5685-DB2
QMF for OS/390	Version 7	5675-DB2
Connection to remote DB2 for VSE DRDA Application Server:		
At the local DB2 for VM location:		
SQL/DS for VM	Version 3 Release 5	5688-103
QMF for VM	Version 7	5697-F42

Table 2. Prerequisite software for optional functions for QMF for VM Version 7 Release 1 (continued)

Product	Version and release	Number
At the remote DB2 for VSE/ESA location:		
SQL/DS for VSE	Version 3 Release 5	5688-103
QMF for VSE	Version 7	5697-F42
Connection to DB2 PE, DataJoiner, Common Server, AS/400:		
At the local DB2 for VM location:		
SQL/DS for VM	Version 3 Release 5	5697-F42
QMF for VM	Version 7	5697-F42
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 or	Version 2 Release 1	53H7474
DB2 for OS/2 or	Version 2 Release 1	41H2114
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	10H2421
DB2 for SCO OpenServer or	Version 2 Release 1	79H5359
DB2 for SINIX or	Version 2 Release 1	79H4133
DB2 for AS/400	Version 4 Release 4	5769-ST1

Products Required to Support Remote Unit of Work

Remote unit of work (RUW) support is not available in all environments in which QMF operates. For example, when running QMF in VSE/ESA, you cannot connect to another location. However, the QMF objects stored in a VSE DB2 database can be accessed by other QMF requesters in a Distributed Relational Database Architecture (DRDA) network. To see if RUW is supported in your operating environment, see the documentation for the database you are using.

Planning for Installation

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.

Discontiguous Shared Segments (DCSS) Storage Requirements

Reference note

Refer to the Program Directory on the ISD tape for information on this topic.

Disk Storage Requirements

Reference note

Refer to the Program Directory on the ISD tape for information on this topic.

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 14. 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*
- 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 for QMF

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.

For information on creating and assigning PUBLIC DBSPACES, see “Choosing and Acquiring a dbspace for the User” on page 118. If you have a DBSPACE available from installing a previous version of QMF, you can use that DBSPACE for QMF V7R1.

Planning for Installation

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 userid 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 these programs (DSQCBINS and DSQCBSQL) 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 4096-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 use of the DB2 ACQUIRE DBSPACE command. Because QMF issues the ACQUIRE DBSPACE command, you must be sure you have first created the appropriate DBSPACES.

The DBSPACES required by QMF, as well as their contents and default sizes, are shown in Table 3.

Table 3. 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
DSQTSYN	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.

Planning for Installation

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, instead, 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 V7R1 into a database that does not contain any version of QMF, you need to create all ten DBSPACES shown in Table 3 on page 15.
 - If you have QMF V2R4 or an earlier release of QMF installed in the same database in which you are installing QMF V7R1, you should already have nine DBSPACES. You need to create one additional DBSPACE for the Q.DSQ_RESERVED control table.
 - If you have QMF V3R1 or a later release installed in the same database in which you are installing QMF V7R1, 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=''
```

Notes:

1. If you plan to create DBSPACES while installing QMF, see the discussion in “Step 2—Create DB2 for VM DBSPACES: DSQ2DBSC” on page 26.
2. 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 4. 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

Planning for Installation

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 V7R1 access modules contain the DSQC 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 V7R1, 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 V7R1 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 certain kinds of help before they can operate the new release of QMF. Supplying this help is what “migration” means.

If you decide to go back to your earlier release of QMF, your V7R1 users might need help. Supplying this help is what “fallback” means.

Migration and fallback are post-installation operations. You'll find them described in “Appendix C. Migration and Fallback Considerations” on page 319. For planning purposes, you should read about them before you begin the V7R1 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 a language of their choice. NLFs enable users to enter QMF commands, view help and other information, 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're using an NLF**.

A QMF NLF is installed *after* you have installed QMF. For a description of NLF, see “Chapter 4. Installing a QMF 7.1 National Language Feature (NLF)” on page 43.

Some names of programs and phases shown in this book have an *n* symbol in them, indicating that the name can vary. If you're using an NLF, replace all *n* symbols you see in this book with the one-character national language identifier (NLID) from Table 5 that matches the NLF you installed. The table also shows the names by which QMF recognizes each language.

Table 5. NLIDs representing QMF base (English) and National Language Features (NLFs)

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
Simplified Chinese	R	S-CHINESE
Spanish	S	ESPANOL

Planning for Installation

Table 5. NLIDs representing QMF base (English) and National Language Features (NLFs) (continued)

NLF	NLID	Name QMF uses for this NLF
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.

Planning for Installing QMF into a Workstation Database Server

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 3. Installing QMF 7.1 into the DB2 for VM Database

This chapter explains the steps for performing a database-only installation of QMF 7.1. If you have already installed QMF 7.1 and want to install it into another database, follow the directions in this chapter.

If you are installing QMF 7.1 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 the DB2 for VM database. For information on how to use REXX, see *VM System Product Interpreter Reference*

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 2 on page 22 is a flow diagram of QMF installation to help acquaint you with the installation process before starting. You might also find the optional “QMF Installation Checklist” on page 315 helpful in monitoring your installation process.

Installing QMF 7.1

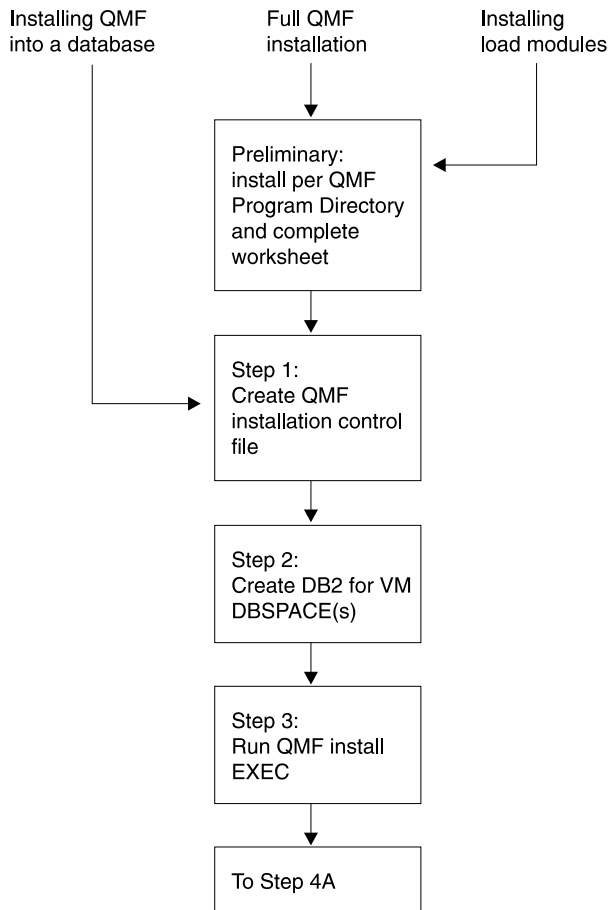


Figure 2. Installation steps for QMF 7.1 (Part 1 of 3)

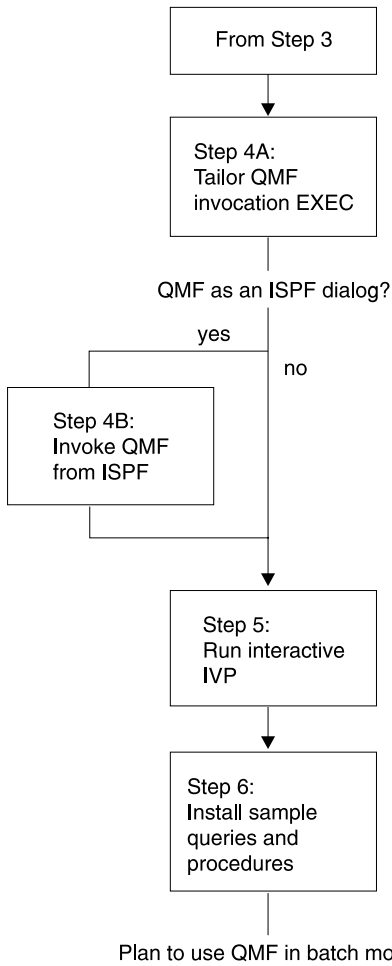


Figure 2. Installation steps for QMF 7.1 (Part 2 of 3)

Installing QMF 7.1

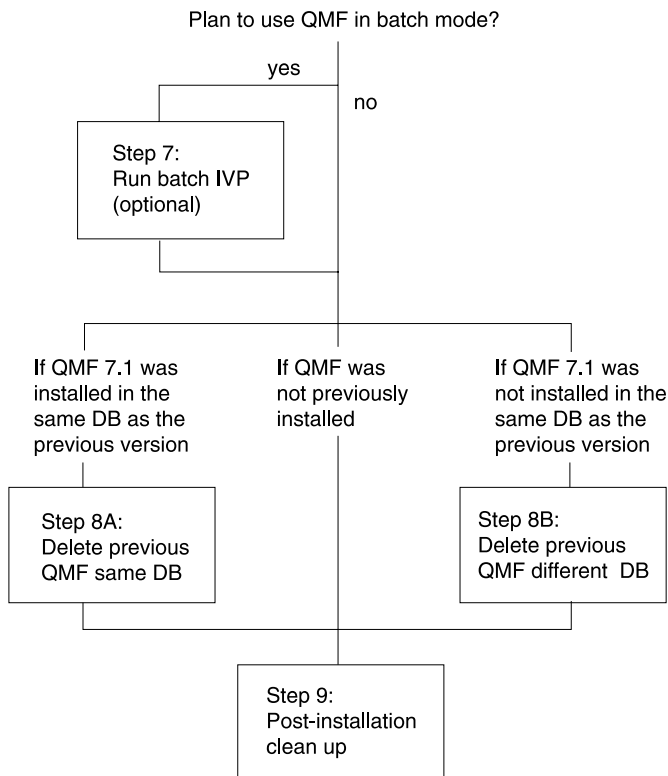


Figure 2. Installation steps for QMF 7.1 (Part 3 of 3)

The Installation Steps

The installation steps are outlined on the following pages.

If you are performing a QMF 7.1 migration installation, that is, if you are installing QMF 7.1 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 7.1 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 into DB2 for VM is described in this book.

The following worksheet lists the information you provide during QMF installation.

Table 6. Information Required during QMF Installation (QMF 7.1 Worksheet)

Information required for:	Supply data fields containing _____			
	No prior QMF	QMF Migration	QMF in DB2 Workstation Server	QMF in AS/400 Server
• Database/location name	_____	_____	_____	_____
• Database type (DB2 VM, DB2 Workstation Server, or DB2 for AS/400)	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_TABLE table (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 AS/400 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.

Installing QMF 7.1

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 6 on page 25).

Anytime during this process, you can enter:

- HELP on the command line for information
- CANCEL to terminate the process before completion

A file, QMFV710E 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—Create 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 7.1 into has QMF 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 6 on page 25). 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 QMFV710E 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 QMFV710E 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 V7R1) 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=''
```

Installing QMF 7.1

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 QMFV710E 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
```

5. Ensure that you have an A-disk to generate temporary files.

If you are performing a QMF 7.1 migration installation from QMF V2R4 or earlier, the Q.OBJECT_DATA table is unloaded during this step. Make sure that you have enough space on your A-disk for the data file.

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 “Planning for Installing QMF into a Workstation Database Server” on page 20.

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 7.1 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).

Installing QMF 7.1

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 3 on page 31. You need to 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 command are described clearly in simpler QMF invocation EXECs. 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 1 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 = 'QMF710'||lchar    /* QMF DCSS name for ISPSTART */
sql_link = 'SQLMACH 195 195'  /* DB2 VM minidisk link information*/
qmf_link = 'P697F42A 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 3. Sample QMF Invocation EXEC (DSQ2EINV)

Notes on Figure 3:

1.

Installing QMF 7.1

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.

For further information on the ISPSTART command, see “Starting QMF with the ISPSTART Command” on page 61.

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.

See “Chapter 5. Starting QMF” on page 57 for further information on CMS and non-CMS subset forms.

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 megabytes of storage without ISPF or 6.0 megabytes 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 discontinuous 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. For information on GDDM nicknames, see GDDM support in “Chapter 9. Enabling Users to Print Objects” on page 147.

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 4096.

Note: The larger the user's spill file, the less often the user encounters the “incomplete data” condition. For more information, see “Customizing Report Storage and Report Performance” on page 73.

- 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.

Installing QMF 7.1

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 TXTLIBs 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*

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 5—Run the IVP for QMF Interactive Mode : DSQ2EIVP

Note: Be sure that you have installed QMF 7.1 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.

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.

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 25. If the DBSPACE value is not correct, update your profile to contain the correct value before you continue.

Step 5D—Test the QMF IVP Procedure

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 the 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 7.1 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—Run the IVP for QMF Interactive Mode : DSQ2EIVP” on page 36.
2. If not done in “Step 5—Run the IVP for QMF Interactive Mode : DSQ2EIVP” on page 36, and you are installing on a DB2 for VM server, issue the command:

```
CONNECT Q (PASSWORD=xxx
```

where xxx is the password of Q.

Installing QMF 7.1

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 6 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. Then rerun procedure DSQ2ESQI.

Step 7—Running the Batch-Mode IVP (Optional): DSQ2EBAT

If you plan to run QMF procedures in batch mode, you should 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 minidisk.

Your CMS PROFILE EXEC should define the following files:

- A print file (DSQPRINT) for printing items
- A message file (DSQDEBUG) for commands run, error messages (if any), and trace output

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, it is possible to get QMF error messages if there is an error. 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. For information on how to use the message utility, see “Diagnosing Your Problem Using QMF Message Support” on page 298.

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 wasn’t 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. For information on how to use the message utility, see “Diagnosing Your Problem Using QMF Message Support” on page 298.

Restarting the Batch IVP

This IVP starting the DSQ2EINV EXEC with the appropriate parameters.

Expected Results from Executing the Batch IVP

The output looks something like the following. (The 'hyphened' lines indicate the beginning and ending of trace records.)

Installing QMF 7.1

```
-----  
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 7.1 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 7.1, 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 7.1, 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 QMFV710E 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.

Congratulations! You have now completed installing the QMF product.

For information on customizing your system, see “Part 2. Managing QMF for VM/ESA” on page 51.

Step 10—Load QMF Database Packages to a Remote Server (Optional): DSQ2BPKB

In order for a QMF Version 7 Release 1 requester installation to be able to communicate to a server, QMF 7.1 packages must be present at the server. If a complete QMF 7.1 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 3.3 or above where migration is not an option, you can run the new install package job, DSQ2BPKB, to install QMF V7 packages at the remote server. Then access from QMF for VM 7.1 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 V3.1
- DB2 for VM/VSE V3.5
- DB2 Universal Database V5
- DataJoiner V2
- DB2 Common Server V2.1
- DB2 Parallel Edition V1.2
- DataJoiner V1.2
- DB2 for AS/400 V4.4

Following is a list of the considerations for running the job (DSQ2BPKB) to load QMF database packages to a remote server.

1. The application server must contain at least QMF 3.3. 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 userid at the server must have administrator authority.
5. This job can be rerun.

Installing QMF 7.1

Chapter 4. Installing a QMF 7.1 National Language Feature (NLF)

This chapter parallels the installation steps for QMF 7.1. 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 3. Installing QMF 7.1 into the DB2 for VM Database” on page 21.

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 19 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 userid 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 Chinese, Japanese, and Korean NLFs use DBCS characters; they require the hardware and program products shown in “Chapter 2. Planning for Installation” on page 7.

The Installation Steps

The installation steps are outlined on the following pages.

Note: You must first install the QMF 7.1 base product before you can install a QMF National Language Feature.

The QMF 7.1 distribution and production minidisks are required for the NLF installation.

Figure 2 on page 22 is an overview of the installation process. The optional “QMF NLF Installation Checklist” on page 316 may be helpful in monitoring your 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 PTF's and APAR's, 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 7. Information Required during QMF NLF Installation (QMF 7.1 Worksheet)

Information required for:	Your data:	QMF in DB2 workstation server	QMF in AS/400 server
Database/location	_____	_____	_____
Database type (DB2 VM, DB2 workstation server, or DB2 AS/400 server)	DB2VM	DB2WS	DB2400
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, DSQ2nCTL, 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. Ensure that your A-disk is not full (so QMF has room to generate temporary files).
3. Fill in the worksheet shown in Table 7, if you have not already done so.
4. Run the EXEC: DSQ2nCTL.

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 25.)

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 QMFV710n 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 minidisk in WRITE mode.
2. The QMF NLF installation control file QMFV710n INSTALL must exist on the QMF NLF distribution minidisk.
3. You must be linked to the DB2 for VM production minidisk 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 “Planning for Installing QMF into a Workstation Database Server” on page 20.

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 45.) 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 29, noting the differences listed here. Recall that you can either tailor the QMF invocation EXEC (Step 4A) or invoke QMF from the ISPF environment (Step 4B).

Step 3A—Tailor the QMF Invocation EXEC: DSQ2nINV

Follow “Step 4A—Tailor the QMF Invocation EXEC: DSQ2EINV (Optional)” on page 30.

Modify the QMF NLF invocation EXEC, DSQ2nINV, to meet the requirements of your installation. The alterable parameters are the same as in Figure 3 on page 31. Note that DSQ2nINV is only a sample NLF invocation EXEC. The necessary links to minidisks, filedefs, SQLINIT, and the ISPSTART command are described clearly in simpler QMF invocation EXECs. These EXECs, DSQ2nINV1 (with ISPF) and DSQ2nINV2 (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.

Step 3B—Invoking QMF from an ISPF Environment (Optional)

Follow “Step 4B—Invoke QMF from an ISPF Environment (Optional)” on page 35 and make changes to the ISPF Master Application Menu as shown in Figure 5 on page 48.

Test the QMF Command Interface

Test the command interface by issuing the following command:

```
CMS DSQ2nC11
```

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 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”.

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 6 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.

Step 6—Run the IVP for QMF NLF Batch Mode (Optional): DSQ2nBAT

Follow the directions for “Step 7—Running the Batch-Mode IVP (Optional): DSQ2EBAT” on page 38.

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, QMFV710n 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 45. 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 2. Managing QMF for VM/ESA

Chapter 5. Starting QMF	57
Before you Start QMF.	57
Establishing a Database Connection	57
Initializing the QMF Session	58
Quick Start	58
Setting up QMF to Run under ISPF	59
Before you start QMF	59
Starting QMF from a Menu Option	59
Starting QMF with the ISPSTART Command	61
PGM Form	61
Program Segment Form	62
Starting QMF in Batch Mode in ISPF.	62
Examples of Starting QMF under ISPF	63
Setting up QMF to Run under CMS	64
Starting QMF Directly with the DSQQMFE Module	64
Starting QMF in a Batch CMS Environment	64
Examples of Starting QMF under CMS	65
Creating a CMS EXEC	65
Verify Program Modules	65
Verify QMF Data Files	65
GDDM Considerations	66
DB2 for VM Considerations.	66
Chapter 6. Customizing Your Start Procedure	67
Quick Start	67
Setting Default Start Values Using the REXX Program DSQSCMDn	68
Naming the Program Segment	72
dcssname	72
DSQSDCSS	73
Customizing Report Storage and Report Performance	73
Adjusting Storage for Report Data (DSQSBSTG)	73
Choosing the Right Amount of Virtual Storage for Each User	73
Performance Tradeoffs	74
Adjusting Reserved Storage Used for Report Data (DSQSRSTG)	74
DSQSBSTG and DSQSRSTG Value of 0	74
Small Value for DSQSBSTG or Large Value for DSQSRSTG	75
Acquiring Extra Storage (DSQSPILL).	75
Allocating a Spill File for CMS Users.	76
Estimating the Space Required for a Spill File	76
Using a Spill File in a Noninteractive QMF Session.	78
Solving Some Spill File Problems	78
Controlling the Number of Report Rows Retrieved for Display (DSQSIROW)	79
Performance with Small DSQSIROW Values	80
Performance with Large DSQSIROW Values	81
Setting the Level of Trace Detail (DSQSDBUG)	81
Controlling Initial Activities During a Session	82
Specifying the Location to Connect to When Starting QMF (DSQSDBNM)	82
Specifying an Interactive or Noninteractive QMF Session (DSQSMODE).	83
Naming a Procedure to Run When QMF Starts (DSQSRUN)	84
Running an Initial Procedure Noninteractively	85
Performing Interactive QMF Work with an Initial Procedure	85
Passing Variable Values to an Initial Procedure.	86
Setting Printing for Double-Byte Character Set Data (DSQSDBCS)	90
Chapter 7. The QMF Session Control Facility	91
Installing or Removing Q.SYSTEM_INI	91
Importing the Default System Initialization Procedure.	91
When Does the Q.SYSTEM_INI Procedure Run?	91
Using Q.SYSTEM_INI.	92
Example Shipped with QMF	92
User Session Procedure Example	92
Procedure that Displays an Object list	93
Security and Sharing Session Procedure.	94

Diagnosis Considerations	94	Sizing a dbspace	118
Chapter 8. Establishing QMF Support for End Users	95	Granting a User DB2 for VM RESOURCE Authority	118
The role of the Q AUTHID	95	Enabling Users to Confirm Table Changes Before They are Made	119
Quick Start	95	Enabling Users to Support a Chart	120
Ensuring That Users Have Access to CMS	96	Maintaining QMF Objects Using QMF Control Tables	120
Creating User Profiles to Enable User Access to QMF	97	Reading the Q.OBJECT_DIRECTORY Table	121
Using the Q User Profile, a Special QMF Profile	97	Reading the Q.OBJECT_DATA Table	122
Establishing a Profile Structure for Your Installation	98	Reading the Q.OBJECT_REMARKS Table	123
Adding a New User Profile to the Q.PROFILES Table	98	Listing QMF Queries, Forms, and Procedures	123
Preventing Users Without Unique Profiles from Using QMF	99	Displaying QMF Queries, Forms, and Procedures	124
Reading the Q.PROFILES Table	100	Transferring Ownership of Queries, Forms, and Procedures	124
Providing the Correct Profile for the User's Operating Environment	104	Deleting Obsolete Queries, Forms, and Procedures	125
Updating User Profiles	105	Enlarging the dbspace for the QMF Object Control Tables	126
Using the SET PROFILE Command	105	Maintaining Tables and Views Using DB2 for VM System Tables	127
Using SQL UPDATE Statements	105	Listing Tables and Views	128
Updating the SYSTEM Profile.	106	Transferring Ownership of a Table or View	128
Deleting Profiles from the Q.PROFILES Table	106	Deleting a Table or View from the Database.	128
Controlling Access to QMF and Database Objects	107	Supporting Locally Defined Date/Time Formats	128
SQL Privileges Required to Access Objects	107	Accessing the DXT End User Dialogs (ISPF Only)	129
SQL Privileges Required for QMF Commands	108	Supporting the EXTRACT Command	129
SQL Privileges Required for Prompted and QBE Queries	109	Allocating Resources.	129
SQL Privileges Required for the Table Editor.	109	Allocating and Reallocating Resources Using EXECs	130
Granting and Revoking SQL Privileges	109	Preparing the Allocation EXEC	130
Using the SQL GRANT Statement	110	Preparing the Reallocation EXEC.	135
Using the SQL REVOKE Statement	110	Other Allocation Methods	138
Sharing QMF Objects with Other Users	111	Customizing the Document Editing Interface for Users.	139
Allowing Uncommitted Read	111	Changing the Application	139
Setting Standards for Creating Objects	112	Renaming the Document Interface Macros and EXEC	139
Customizing a User's Database Object List	112	Placing the Q.DSQAED2S Procedure in the Database	139
Using the Default Object Lists.	113	Transferring Ownership to Q	140
Changing the Default List	114	Changing the Data Components	140
Object List Storage Requirement	115	The Message Component	140
Enabling Users to Create Tables in the Database	116		
Choosing and Acquiring a dbspace for the User	118		
Using the SQL ACQUIRE Statement	118		

The DCF Components	141	Customizing DPRE	161
Changing the EXECs and Macros	142	Creating a Command Synonym Table	162
Changing DSQABD2Q	142	Entering Command Synonym Definitions	
Changing DSQABD2I	142	into a Command Synonym Table.	163
Changing DSQABD2C	142	Choosing a Verb	163
Customizing the QMF Edit Command	143	Rules for the VERB Column	164
Enabling English Support in an NLF		Using Base QMF Verbs as Command	
Environment	144	Synonym Verbs	164
Using Global Variables to Define the		Choosing an Object Name	165
Currency Symbol	145	Choosing the Synonym Definition	165
Chapter 9. Enabling Users to Print		Using a Procedure in the Synonym	
Objects	147	Definition	165
Quick start	147	Using Variables in the Synonym	
Printing Objects	148	Definition	166
Deciding Whether to Use QMF or GDDM		Keying Information Into the	
Services for Printing	149	SYNONYM_DEFINITION Column	168
Using GDDM Services to Handle Printing	149	Activating the Synonyms	168
Choosing a GDDM Nickname for Your		Minimizing Maintenance of Command	
Printer	150	Synonym Tables	170
Choosing the Right Type of GDDM		Assigning One Synonym Table to all	
Device	150	Users	170
Creating the Nickname Specification	151	Assigning Views of a Synonym Table to	
Example Nickname for a Family 2		Individual Users	170
GDDM Printer.	151	Synonyms for Public or Private Use	170
Example Nickname for a Family 3		Synonyms for Public or Group Use	171
GDDM Printer.	152	Synonyms Paired with an	
Example Nickname for a Family 4		Authorization Table	171
GDDM Printer.	152		
Defining Multiple Nicknames with		Chapter 11. Customizing QMF Function	
One Definition.	152	Keys	173
Examples of Nickname Definitions	153	Quick Start	173
Updating the GDDM Defaults Module		Choosing the Keys You Want to Customize	173
with the Nickname	154	Default Keys on Full-screen Panels	174
Testing the Nickname Definitions in		Default Keys on Window Panels.	175
External Default Files	154	Creating the Function Key Table	176
How QMF Interfaces with Your GDDM		Entering Your Function Key Definitions into	
Nickname	154	the Table.	177
Using QMF's DSQPRINT to Handle Printing	155	Linking a Command with a Function Key	177
Defining a Synonym for the Print Function		Labeling the Function Key and	
Key	156	Positioning it on the Screen	179
Updating User Profiles to Enable GDDM		Examples of Key Definitions	179
Printing	156	Entering a Definition for a Key on a	
		Full-screen Panel	179
		Entering a Definition for a Key on a	
		Window Panel.	180
		Entering a Key Definition for a Help	
		or Prompt Panel	181
Chapter 10. Customizing QMF Commands	159	Identifying the Panel You Want to Customize	181
Quick Start	159	Full-screen Panel Identifiers	181
Using the Default Synonyms Provided with		Window Panel Identifiers	181
QMF	159		
Displaying Printed Reports (DPRE).	160		
Using DPRE	160		

Command Windows	182
Forms Windows	182
Global Variable Windows	182
Help and Prompt Windows	182
Location Windows	182
Object List Windows.	182
Prompted Query Windows.	183
Activating New Function Key Definitions	184

Chapter 12. Creating Your Own Edit

Codes for QMF Forms	187
Quick Start	187
Choosing an Edit Code	188
Handling DATE, TIME, and TIMESTAMP Data Types	189
Calling Your Exit Routine to Format the Data	191
Passing Information to and from the Exit Routine	193
Fields of the Interface Control Block	193
Fields That Characterize the Input Area	195
How U-Type Edit Codes are Represented in the Input Area	196
How V-Type Edit Codes are Represented in the Input Area	196
Fields That Characterize the Output Area	196
Passing Control to the Exit Routine When QMF Terminates	197
Writing an Edit Routine in High-Level Assembler (HLASM) or Assembler	197
How an Assembler Edit Routine Interacts with CMS	198
How an Assembler Edit Routine Interacts with QMF	199
Assembling Your Program	202
Generating Your Program	203
Writing an Edit Routine in PL/I without Language Environment (LE)	203
How a PL/I Edit Routine Interacts with QMF	204
Compiling Your Program	209
Creating Your DSQUEDIT Module File in PL/I	210
Writing an Edit Routine in PL/I with Language Environment (LE)	211
Generating Your PL/I Program for LE	212
Writing an Edit Routine in COBOL without Language Environment (LE)	213
How a COBOL Edit Routine Interacts with QMF	214

Compiling Your Program	219
Assembling the Run Time Options Macro (COBOL II)	220
Generating Your Program	220
Writing an Edit Routine in COBOL with Language Environment (LE)	221
Generating Your COBOL Program for LE	222
Handling Double-Byte Character Set Data	223
Edit Codes for DBCS Data	223
What the Edit Routine Receives	223
Data from Graphic Columns	223
Data from Character Columns	224
Ensuring the Edit Routine Returns the Right Results	224
Overflowing the ECSRSLT Field	224
Printing the Report Column	224

Chapter 13. Controlling QMF Resources

Using a Governor Exit Routine	227
Quick start	227
Using the IBM-Supplied Governor Exit Routine	228
Activating the Default Limits	229
How a Governor Exit Routine Controls Resources	230
How the Governor Knows What the Resource Limits Are	231
How the Governor Knows When You Reach a Resource Limit	232
What Happens When You Reach a Resource Limit.	233
Defining Your Own Resource Limits	233
Creating your own Resource Control Table	236
VM Timer Considerations	236
Modifying the IBM-supplied Governor Exit Routine or Writing Your Own.	238
Program Components of the Governor Exit Routine	239
How CMS Interacts with the Governor Exit Routine	240
How and When QMF Calls the Governor Exit Routine	241
Points at Which QMF Calls the Governor	241
What Happens Upon Entry to the Governor Exit Routine	243
Establishing Addressability for Function Calls	243

Passing Resource Control Information to the Governor Exit.	244	Restricting Use of the APPLDATA Column	270
Structure of the DXEGOVA Control Block	245	Avoiding Use of Some Special Characters	270
Addressing the Resource Control Table Structure of the DXEXCBA Control Block	249	Enabling Your Users to Access a Remote Database.	271
Storing Resource Control Information for the Duration of a QMF Session	258	Updating a User's Profile	271
Canceling User Activity.	259	Specifying Access for Current SQL Authorization ID	271
Providing Messages for Canceled Activities	260	Connecting to the Local Database	271
Assembling and Generating Your Governor Exit Routine	261	Connecting to the Remote Database.	271
Assembling Your Governor Exit	261	Specifying a Location Name	272
Building a Module File or Creating a Load Library Member	262	In DB2	272
Chapter 14. Customizing a Remote Database Connection	263	In DB2 for VM.	273
Quick Start	263	Where Data Must be Located for User Access	273
Determining the Remote Database Connection Needed	264	Working with QMF Objects	273
Connecting with Remote Unit of Work	265	Working With Tables.	274
Connecting with DB2-to-DB2 Distributed Unit of Work	265	Preventing SQL Errors	274
Specifying a Table or View with a Three-part Name in DB2	265	Translating User IDs	275
Directing a Query Using Three-part Names	265	Translating Names	275
Verifying the Connections Necessary for Remote Unit of Work	266	Deleting QMF Users from Each Remote QMF Location	275
Checking DB2 for VM Connections	266	Enabling Administrator Access to Your Location	275
Checking DB2 for VM Connections	266	Chapter 15. Customizing the Batch Processing Program	277
Preparing a Non-DB2 for VM Location for Access by QMF VM Users	267	Quick Start	277
Creating Command Synonym Tables	267	Enabling Your Users to Use Batch Mode	278
Sample Remote Server Command Synonym Table for the CMS Environment	268	Sending a Job to the CMS Batch Machine	279
Preparing QMF to Support the DPRE Command	269	Running Batch Jobs on Your Machine	281
Preparing QMF to Support Other Commands	269	Debugging a Procedure.	282
Creating Function Key Tables.	269	Using the QMF Batch Query/Procedure Application (BATCH)	282
Updating QMF Governor Control Tables	270	MACLIBs Required	283
Installing the National Language Feature in the QMF Server	270	Using the Application	283
Code Page Support	270	Filling in the Prompt Panel	283
		Required Entry Fields	284
		Optional Entry Fields	285
		Modifying the Batch Application.	286
		Chapter 16. Troubleshooting and Problem Diagnosis	289
		Quick Start	289
		Troubleshooting Common Problems.	290
		Handling Initialization Errors.	290
		Handling Warning Messages	291
		Handling GDDM Errors During Printing	292
		Handling QMF Errors During Printing	292

Handling CMS Command Errors	294
Using the CMS Command to Run an EXEC	294
Issuing the CMS Command if QMF is Started Using ISPF	294
Using the DB2 for VM CONNECT Command	294
Using the DB2 for VM COMMIT Command	295
Handling Display Errors	295
Using the HEX Function	295
Using QMF-provided Hex and Bit Edit Codes.	295
Handling Binary Data with User-Written Edit Routines.	295
Solving Slow Performance Problems	296
Resetting the Data Object to Improve Performance	296
Increasing the User's Report Storage	297
Using REXX Function Packages	297
Determining the Problem Using Diagnosis Aids	298
Choosing the Right Diagnosis Aid for the Symptoms	298
Diagnosing Your Problem Using QMF Message Support	298
Determining which QMF Function Issued an Error Message	299
Handling System Error Messages	300
Handling SQL Return Codes	300
Using the QMF Trace Facility	300
Allocating the Trace File	301
Starting the Trace Facility	301
Getting the Right Level of Detail in Your Trace Output	302
Tracing at the Module Level	304
Viewing QMF Trace Data	304
Determining the QMF Service Level	305
Turning Off the Trace Facility	305
Abend Handling	305
Using the QMF Interrupt Facility	306
Creating an Interrupt	306
Displaying Trace Information After Creating an Interrupt	307
Error Handling	308
Using Error Log Reports from the Q.ERROR_LOG Table	308
Reporting a Problem to IBM	309
Using ServiceLink to Search for Previously Reported Problems	310

Chapter 5. Starting QMF

This chapter describes the various methods you can use to start QMF. You can start QMF running under ISPF or CMS, or from the QMF server.

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

Before you Start QMF

Before you start QMF, you need to decide which environment you want QMF to run in. The method used to start QMF depends upon the environment from which QMF is started and whether the user wants to run QMF under ISPF.

Establishing a Database Connection

Before you start QMF, you need to establish the CMS and DB2 for VM environment.

The DB2 for VM database program usually operates in its own virtual machine associated with a VM logon ID. So does each QMF user. 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 those for DB2 for VM.

Any combination of the following situations can exist:

Case 1: The DB2 for VM directory has IUCV ALLOW. In that case, any other virtual machine can communicate with the DB2 for VM machine, and nothing else need be done to allow communication.

Case 2: The QMF user's entry has IUCV ANY. In that case, the QMF user can communicate with any other virtual machine, including the DB2 for VM machine.

Case 3: The QMF user's entry has IUCV *sqldsid*, where *sqldsid* is the user ID of the DB2 for VM virtual machine. The QMF user can have this directory entry in any case, and must have it if neither case 1 nor 2 holds.

Case 4: The DB2 for VM directory has 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.

Starting QMF

Setting up a user's directory entry is a normal part of the task of providing a VM logon ID. For instructions about it, see *VM/SP Planning and System Generation Guide* and *DB2 Server for VM System Administration*

If your installation requires the use of QMF in different databases, you must install QMF into each unique DB2 for VM database. Each database contains the following:

- QMF control tables
- QMF DB2 for VM packages
- QMF sample tables and queries
- QMF views (system tables)

To install QMF into multiple databases, see “Part 1. Installing QMF for VM/ESA” on page 1.

Initializing the QMF Session

When initializing the QMF session:

- DB2 for VM user ID = VM logon ID
- QMF does an implicit connect

Quick Start

Table 8 outlines ways you can set up QMF to start.

The *n* symbol in each example represents the national language identifier (NLID). Substitute the NLID from Table 5 on page 19 that corresponds to the national language in which you want to start QMF. For example, to start an English QMF session, enter QMFE.

For more information on any of the tasks listed, see the page shown at the right of the table.

Table 8. Options for starting QMF

To do this task:	See:
To set up QMF to start with the PGM form of the ISPSTART command , enter: ISPSTART PGM(DSQMF) NEWAPPL(DSQE) PARM(...) You can also use the program segment form.	Page 61
To set up QMF to start in batch mode in ISPF , enter: ISPSTART PGM(DSQMF) NEWAPPL(DSQE) PARM(...DSQSMODE=B,DSQSRUN=aaa.bbb) You can also start QMF in batch mode using a EXEC.	Page 62
To set up QMF to start directly with the DSQMF module , enter: DSQMF DSQSBSTG=123456,DSQSIROW=0,DSQSRUN=SAM.PROG1	Page 64
To set up QMF to start in a batch CMS environment , enter: DSQMF ...DSQSMODE=B,DSQSRUN=aaa.bbb	Page 64

Table 8. Options for starting QMF (continued)

To do this task:	See:
To create a new CMS EXEC to start QMF, you need to ensure that the program modules and data files are available to QMF, and that GDDM and DB2 for VM considerations have been met.	Page 65

Setting up QMF to Run under ISPF

You can let users start QMF using ISPF services. You can do this in three ways:

- ISPF has an initial dialog to which you can add QMF.
- 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 methods to start the others. For example, you can run an initial dialog from an EXEC.

If you are going to run QMF under ISPF, you must start the QMF program dialog using the ISPF SELECT service. When a CMS command is used, results are unpredictable.

Restrictions:

1. You cannot run QMF as a command dialog.
2. You cannot enter QMF from a split screen or create a split screen during a QMF session if QMF was started as an initial dialog.

For information on ISPF, see *Interactive System Productivity Facility for VM Dialog Management Services and Examples*

Before you start QMF

A FILEDEF ISPLLIB statement for DSQDLIB LOADLIB must be in place before QMF is started; it can be done either before or after ISPF is started, for example:

```
FILEDEF ISPLLIB DISK DSQDLIB LOADLIB * (PERM)
```

Starting QMF from a Menu Option

If you choose to set up a menu option to start QMF, the menu must point to QMF, but can also point to QMF resources. Figure 6 on page 60, which shows a sample definition for the ISPF master application menu, illustrates how to add an option to the menu. In this definition, Option 2 was added for reaching QMF through an EXEC.

figure, this function transforms options 2, 3, and 4 into the operand portions of an ISPF command that is executed like ISPSTART (see the previous section). The command executed invokes the appropriate QMF module (DSQQMFE, DSQQMFU, or DSQQMFK), and passes it the value 150 for the DSQSIROW parameter. The DSQSIROW parameter is discussed in “Controlling the Number of Report Rows Retrieved for Display (DSQSIROW)” on page 79.

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 you create for the menu option has no resources to allocate. This leaves it with a single function, starting QMF, and you can do this without an EXEC.

Starting QMF with the ISPSTART Command

Use the ISPF command ISPSTART to develop an EXEC to enable users to start QMF as an ISPF dialog. For example, you can enter the following statement from the command line:

```
ISPSTART PGM(DSQQMFE) NEWAPPL(DSQE) PARM(...)
```

ISPSTART starts QMF as the new ISPF application DSQE. The QMF program DSQQMFE must be run with an application ID of DSQE. (The national language ID for each must be the same.)

The optional PARM operand, which passes parameter values to the QMF program DSQQMFE, might look like this:

```
PARM(DSQSBSTG=256000,DSQSIROW=50,DSQSRUN=SAM.PROG1)
```

Parameters are discussed more fully later in this chapter.

You can start QMF with either the PGM form or program segment form of the ISPSTART command.

PGM Form

PGM is the object of the ISPSTART command; with PGM, you specify the QMF program DSQQMFE. To do this, enter the following statement from the command line in CMS, or include it as a statement in an EXEC:

```
ISPSTART PGM(DSQQMFE)
NEWAPPL(DSQE)           PARM(dcsname(DSQSBSTG=n1,...))
```

When the PGM form is used, the QMF program segment is started indirectly through the IBM-supplied program, DSQQMFE, and QMF runs in CMS subset mode.

Starting QMF

Environmental considerations: When you start QMF using the PGM form of the ISPSTART command, QMF runs in CMS subset mode. All subsequent QMF processes also run in CMS subset mode.

Program Segment Form

DCSS is the ISPF keyword for the QMF program segment used by QMF when QMF is started with ISPSTART; *dcssname* is the name of the program segment being used. To start QMF using the program segment form of ISPSTART, enter the following statement from the command line in CMS or include it as a statement in an EXEC:

```
ISPSTART DCSS(dcssname) NEWAPPL(DSQE)
          PARM(DSQSBSTG=n1,...)
```

When the program segment form is used, the QMF program segment is started directly. QMF can run in CMS SUBSET or CMS non-SUBSET mode when started using a program segment (as explained in Environmental Considerations).

When QMF is executed as an ISPF dialog, the QMF program DSQQMFE must be run with an application ID of DSQE. You can find more information on the parameters and the *dcssname* under “Chapter 6. Customizing Your Start Procedure” on page 67.

Environmental Considerations: When you start QMF using the program segment form of the ISPSTART command, QMF runs in *either* CMS non-SUBSET mode *or* CMS SUBSET mode. The mode depends on the calling environment. For example, if you are running in CMS non-SUBSET mode and start QMF using a program segment, QMF runs in non-SUBSET mode.

When running QMF in CMS non-SUBSET mode, you must be sure that all programs called from within QMF, including QMF exit routines, are relocatable. Running programs or tools that are not relocatable or that run at specific locations in storage can cause unpredictable results when run from within QMF.

When QMF uses the command interface, it always runs in CMS subset mode regardless of how it is started. This means that all processes running under QMF within the command interface are also running in CMS subset mode, until the command interface returns control to the initial QMF program segment, at which time QMF runs in the mode in which it was started.

Starting QMF in Batch Mode in ISPF

You can start QMF running in batch mode. You might want to start QMF in batch mode to save resources and time.

You can start QMF using ISPF with an EXEC. To start QMF from an EXEC, place the following statement in the startup file:

```
ISPSTART CMD(exec_name) NEWAPPL
```

where *exec_name* is the name of the EXEC that starts QMF. The example startup EXEC distributed with QMF is DSQ2EINV.

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

For additional samples and information about running QMF in batch mode, see “Chapter 15. Customizing the Batch Processing Program” on page 277.

Examples of Starting QMF under ISPF

The following are some examples of starting and passing parameters to QMF under ISPF.

- Starting from an EXEC and specifying QMF as the initial dialog:

```
ISPSTART PGM(DSQMFE) 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 DSQSRSTG (amount of reserved storage).

- Starting from an EXEC operating within ISPF:

```
ISPEXEC SELECT PGM(DSQMFE) 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(DSQMFE) NEWAPPL(DSQE) PARM(DSQSPILL=NO) '
              .
              .
              .
```

This code passes NO for DSQSPILL whenever a user selects option 1.

- Starting from an EXEC and specifying an initial procedure:

```
ISPSTART PGM(DSQMFE) 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 the preceding example results in the following QMF command:

Starting QMF

```
RUN Q.IPROC(&TABLE=Q.STAFF
```

Setting up QMF to Run under CMS

In CMS, a user can start QMF in the following ways:

- By using the DSQQMFE command either with the NUCXLOAD command or in an EXEC
- In a batch CMS environment

Note that, if you use the CONCAT option on the ISPLLIB FILEDEF statement, you must also issue a GLOBAL LOADLIB DSQDLIB.

Starting QMF Directly with the DSQQMFE Module

Before you enter DSQQMFE, it must be loaded as a nucleus extension with the NUCXLOAD command:

```
GLOBAL LOADLIB DSQDLIB  
NUCXLOAD DSQQMFE
```

To run QMF independently of ISPF, use either of the following commands:

```
DSQQMFEdcssname (DSQSBSTG=n1,...)
```

```
DSQQMFE DSQSBSTG=n1,...
```

Because *dcssname* is optional, the second statement is also correct.

The parameters you can specify are the same as those available under ISPF. The parameters and the *dcssname* are discussed later in this chapter.

When QMF is started in CMS independently of ISPF, the following return codes are valid:

0	Execution successful
4	Warning condition occurred
8	Error condition occurred
16	Severe error occurred

Starting QMF in a Batch CMS Environment

To start QMF without using ISPF services, place the following statement in a start job:

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

where 'DSQSMODE=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 CMS

The following are some examples of starting and passing parameters to QMF operating independently of ISPF:

- Starting from CMS:

```
DSQQMFE dcssname(DSQSBSTG=50000,DSQSDEBUG=NONE,DSQSMODE=B)
```

This statement turns on L2 tracing (DSQSDEBUG=NONE), passes a value of 50 000 for DSQSBSTG (maximum storage for reports), and passes a value of B (batch) for DSQSMODE (mode of operation).

- 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

The DSQSRUN parameter as specified in the preceding example results in the following QMF command:

```
RUN Q.IPROC(&TABLE=Q.STAFF)
```

Creating a CMS EXEC

To create a new CMS EXEC to start QMF, you need to ensure that the program modules and data files are available to QMF, and that GDDM and DB2 for VM considerations have been met.

Verify Program Modules

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. For more information about making these modules available, see "Part 1. Installing QMF for VM/ESA" on page 1.

Verify QMF Data Files

The following list of data files is 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

DSQDEBUG cannot be allocated to a disk by using the shared file system (SFS).

DSQPRINT

Print data output

DSQSPILL

Spill data file

Starting QMF

DSQSPILL cannot be allocated to a disk by using the shared file system (SFS). You may, instead, choose to use a temporary disk.

DSQEDIT

Edit transfer file

DSQEDIT cannot be allocated to a disk by using the shared file system (SFS). Instead, use a temporary disk.

DSQPNLE

QMF panel file

ISPLLIB

Filedef for QMF library; contains the QMF programs (for example, DSQDLIB).

DSQDLIB

QMF load library

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 can remove the lines in the invocation EXEC that refer to GDDM.

If you do not have GDDM in a DCSS, you must access the GDDM TXTLIBs and perform the necessary FILEDEFS. If you want to change the release of GDDM being used by QMF, you must rebuild the QMF shared segment using the DSQ2ESEG EXEC.

DB2 for VM Considerations

QMF supports DATE, TIME, and TIMESTAMP data types, so users can make use of local date/time exit routines. 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 is started.

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 rebuilt using the DSQ2ESEG EXEC.

Chapter 6. Customizing Your Start Procedure

This chapter describes the various methods you can use to pass parameters to the program to help you customize a user's QMF session.

Quick Start

Table 9 shows how to use the program parameters to customize aspects of the QMF session. The command syntax in the examples applies to starting QMF with the DSQQMFE module. If you start QMF differently, see the command syntax in “Chapter 5. Starting QMF” on page 57.

The *n* symbol in each example represents the national language identifier (NLID). Substitute the NLID from Table 5 on page 19 that corresponds to the national language in which you want to start QMF. For example, to start an English QMF session, enter QMFE.

For more information on any of the tasks listed, see the page shown at the right of the table.

Table 9. Passing parameters

To do this task:	See:
To name the program segment something other than the default QMF710E use the dcssname or dsqsdcss parameters. For example to change the name to QMFNEW when using the ISPSTART command, enter: ISPSTART DCSS(QMFNEW)	Page 72
To set limits on the amount of storage used for QMF queries and reports, use the DSQSBSTG parameter if you want any limit other than 0. For example, to specify a limit of 1 000 000 bytes (1MB): DSQQMF _n B=1000000	Page 73
To set limits on the amount of CMS storage used for QMF queries and reports, use the DSQSRSTG parameter if you want any limit other than 0. For example, to specify a limit of 1 000 000 bytes (1MB): DSQQMF _n R=1000000	Page 74
To use temporary storage (a spill file) as extra storage for report data, use the DSQSPILL parameter. For example, enter: DSQQMF _n L=YES	Page 75
To allow QMF to retrieve any number of rows other than 100 before QMF displays the first screen of the report, use the DSQSIROW parameter. For example, to allow QMF to retrieve 200 rows before displaying the first screen, enter: DSQQMF _n F=200	Page 79

Customizing Your Start Procedure

Table 9. Passing parameters (continued)

To do this task:	See:
To log QMF activity in the trace data , including activity before the user's profile is established, use the DSQSDBG parameter. For example, enter: DSQQMFn T=ALL	Page 81
To specify a database location to connect to when starting QMF other than the default location , use the DSQSDBNM parameter. DSQQMFn D=DBNAME DSQQMFn D=DBNAME	Page 82
To run QMF without user interaction (either with or without a terminal) , use the DSQSMODE parameter and specify an initial procedure using the DSQSRUN parameter. You might also choose to use the DSQSDBNM parameter to ensure you connect to the database location you want. For example, to do some noninteractive QMF work using the Q user ID and an example procedure named STARTPROC, enter: DSQQMFn M=B,D=DBNAME,I=STARTPROC	Page 83
To run an initial procedure when QMF starts , use the DSQSRUN parameter. For example, to run a procedure called STARTPROC, enter: DSQQMFn I=STARTPROC	Page 84
To use an initialization program to specify values for program parameters other than the default values set by QMF , use the DSQSCMDn parameter. For example, enter: DSQQMFn DSQSCMDE=NULL	Page 68
To print DBCS data from non-DBCS terminals , use the DSQSDBCS parameter. For example, enter: DSQQMFn K=YES	Page 90

Setting Default Start Values Using the REXX Program DSQSCMDn

Parameter name

DSQSCMDn

Short form

(no short form)

Valid values

NULL or

Default

DSQSCMDE

You can specify default values for the program parameters using an initialization program. IBM supplies the REXX program DSQSCMDn for this purpose. DSQSCMDn 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 DSQSCMDn. The parameter values you specify when a workstation session is started override the values set in DSQSCMDn.

DSQSCMDn is valid only as a start function keyword on the START command when QMF is started from an application program using the callable interface.

The REXX program method must be used by programs using the callable interface that want to run in all the SAA environments without changing their programs.

For more information on the START command and the SAA callable interface, see *QMF Reference* and *Developing QMF Applications*.

For CMS, QMF calls the REXX program DSQSCMDE (see Figure 7 on page 70) 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.

If you do not want to provide a parameter value in DSQSCMDE, you can use 'NULL'.

┌ **General-Use Programming Interface** ───────────────────────────────────

Customizing Your Start Procedure

```
/*REXX -----*/
/* DSQSCMDE: */
/*
/* COPYRIGHT: LICENSED MATERIALS - PROPERTY OF IBM */
/*          5675-DB2, 5697-F42 (C) COPYRIGHT IBM CORP. */
/*          1989, 2000. (PUBLISHED) */
/*          ALL RIGHTS RESERVED. */
/*          US GOVERNMENT USERS RESTRICTED RIGHTS - */
/*          USE, DUPLICATION OR DISCLOSURE RESTRICTED */
/*          BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP. */
/*
/* STATUS: VERSION 7 RELEASE 1 LEVEL 0 */
/*
/* This REXX program returns default QMF program parameters. */
/* Values returned by this program can be substituted with */
/* values specified on the QMF START command. */
/*-----*/
          Trace Off
/* Signal ON ERROR          Immediate exit upon any error condition */

PARSE UPPER SOURCE CSYS .
```

Figure 7. Example REXX program DSQSCMDE (Part 1 of 3)

Customizing Your Start Procedure

```
/*-----*/
/* Customer should tailor the QMF environment by adjusting any */
/* of the following variable values. Each variable value is */
/* commented indicating the environment(s) in which it is */
/* effective. */
/* */
/* IMPORTANT: */
/* A value must be specified for each one of the following */
/* variables. Also each variable can only contain a single */
/* value and must NOT contain a blank. Use the term NULL */
/* instead of a blank value. */
/*-----*/
DSQADPAN = "1" /* CMS and TSO */
DSQALANG = "E" /* CMS and TSO */
DSQSBSTG = "NULL" /* CMS and TSO */
DSQSDBCS = "NO" /* CMS and TSO */
DSQSDBNM = "NULL" /* CMS and TSO */
DSQSDBG = "NONE" /* CMS and TSO */
DSQSIROW = "100" /* CMS and TSO */
DSQSMODE = "BATCH" /* CMS and TSO */
DSQSPILL = "NULL" /* CMS and TSO */
DSQSRSTG = "0" /* CMS and TSO */
DSQSRUN = "NULL" /* CMS and TSO */

DSQSDCSS = "QMF710E" /* CMS only */

DSQSPLAN = "QMF710" /* TSO only */
DSQSPRID = "PRIMEID" /* TSO only */
DSQSSUBS = "DSN" /* TSO only */
```

Figure 7. Example REXX program DSQSCMDE (Part 2 of 3)

Customizing Your Start Procedure

```
/*-----*/
/* Return variables to the QMF start function.          */
/*                                                     */
/* IMPORTANT: Sequence of variables in RETURN statement must NOT */
/*           be altered.                                */
/*-----*/

IF CSYS = CMS THEN DO
    RETURN DSQSMODE DSQSRUN DSQALANG DSQSIROW DSQSRSTG ,
           DSQSDBCS DSQSDBG DSQSDCSS DSQSBSTG DSQSPILL ,
           DSQSDBNM DSQADPAN
END
ELSE DO
    RETURN DSQSMODE DSQSRUN DSQALANG DSQSIROW DSQSRSTG ,
           DSQSDBCS DSQSDBG DSQSPLAN DSQSSUBS DSQSBSTG ,
           DSQSPRID DSQSPILL DSQSDBNM DSQADPAN
END

ERROR:          /* Immediate exit upon any error condition */
EXIT 12
```

Figure 7. Example REXX program DSQSCMDE (Part 3 of 3)

End of General-Use Programming Interface

Naming the Program Segment

You can use *dsqsdcss* or *dcssname* to name the program segment.

The name of the QMF program segment. The suggested program segment name and default value is: **QMF710E**

dcssname

The syntax of *dcssname* is still supported in QMF.

Notes:

1. In the PGM form of the ISPSTART command ISPSTART PGM(DSQMFE)... (see “Starting QMF with the ISPSTART Command” on page 61), this parameter is optional provided the default DCSS name is used.
2. In the DCSS form of the command ISPSTART DCSS(*dcssname*)... (see “Program Segment Form” on page 62), a DCSS name must be specified.
3. If QMF is not running as an ISPF dialog, and DSQMFE *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.

Customizing Report Storage and Report Performance

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
- Spill storage used when virtual storage for reports is full
- How many rows of data are retrieved before QMF displays the first screen of the report

Adjusting Storage for Report Data (DSQSBSTG)

Parameter name

DSQSBSTG

Short form

B

Valid values

From 0 to 9 999 999 bytes

Default

0

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 9 999 999. If DSQSBSTG is specified with a nonzero value less than a QMF-determined minimum (15 to 32 kilobytes, 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. For information on DSQSRSTG, see the discussion in “Adjusting Reserved Storage Used for Report Data (DSQSRSTG)” on page 74.

Choosing the Right Amount of Virtual Storage for Each User

Each QMF CMS region requires at least 4.5 megabytes of virtual storage. Additional storage generally provides improved performance since QMF is able to keep more data records in virtual storage.

Customizing Your Start Procedure

Performance Tradeoffs

You can 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. Also, the user doesn't receive any notification if there is insufficient storage and QMF can still complete report processing. Thus, if you do not provide enough space, performance might be poor even using a spill file, because QMF must return to the database many times to retrieve all the requested data. For this reason, IBM recommends that you ensure your users have enough virtual storage for the QMF work they need to do.

You might also 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 13. Controlling QMF Resources Using a Governor Exit Routine” on page 227.

Adjusting Reserved Storage Used for Report Data (DSQSRSTG)

Parameter name

DSQSRSTG

Short form

R

Valid values

From 0 to 9 999 999 bytes

Default

0

The value of this parameter is a positive whole number ranging in value from 0 through 9 999 999 with a default of 0. The value can affect the running of 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 user's 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 to determine the amount to allow for the use of QMF reports. The remaining storage is available for other programs, including OS/390 system services, 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.

DSQSBSTG and 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. But a user who does use a VM system service or a CMS command and has DSQSRSTG=0 and DSQSBSTG=0, runs the risk of failing and possibly 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.

Acquiring Extra Storage (DSQSPILL)

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 allows you to allocate a spill file, which is extra storage used when a user's storage is full.

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

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 (all rows requested have been retrieved) and the data object is complete.
- Storage you defined for the spill file is full.

Customizing Your Start Procedure

Allocating a Spill File for CMS Users

You can allocate a spill file through a FILEDEF statement. The statement looks like this:

```
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. The records must *always* be unblocked. (A block is the size of a VM page: 4096 bytes.)

For information on calculating the appropriate spill file size, see “Estimating the Space Required for a Spill File”.

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. Then enlarge the virtual storage according to how many individual spill files you'll 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 10 on page 77). See Table 11 on page 77 for sample calculations.
 - All the 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 4096 or less**, calculate the number of rows per page (R) using $R = 4096/W$, and round the result down to the next lowest integer.

When W is 4096 or less, QMF fits as many rows as it can into a page, without spanning pages.
3. **If W is greater than 4096**, calculate the number of pages per row (P), using $P = W/4096$, and round up to the next highest integer.

When W is greater than 4096, QMF uses the minimum number of pages to hold a row, spanning pages regardless of column boundaries. Each row begins at the start of a page.
4. **Calculate the number of pages required for the spill file**, according to the value of W:

- If W is 4096 or less, calculate the number of pages required for the spill file by *dividing* the number of rows in the table by R.
- If W is greater than 4096, calculate the number of pages required for the spill file by *multiplying* the number of rows in the table by P.

Table 10. 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	
LONG VARGRAPHIC	
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 11 shows a sample calculation for a spill file.

Table 11. 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

Customizing Your Start Procedure

Table 11. Sample row width calculation for a spill file (continued)

Content of Row	Calculation	Contribution to Width
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:

$$\begin{array}{r} 600\ 000\ \text{rows} \quad 1\ \text{track} \quad 1\ \text{cylinder} \\ \hline \text{-----} * \text{-----} * \text{-----} = 571\ \text{cylinders} \\ 7 \qquad 10\ \text{blocks} \quad 15\ \text{tracks} \end{array}$$

When $R=6000$, 2 buffers/row:

$$\begin{array}{r} 6000\ \text{rows} * 2\ \text{blocks/row} * 1\ \text{track} \quad 1\ \text{cylinder} \\ \hline \text{-----} * \text{-----} = 800\ \text{cylinders} \\ 10\ \text{blocks} \quad 15\ \text{tracks} \end{array}$$

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. However, 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.

For more information on noninteractive QMF sessions, see “Specifying an Interactive or Noninteractive QMF Session (DSQSMODE)” on page 83.

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 sufficient storage available to QMF after your data is retrieved a first time, QMF does not need to reaccess the database to obtain rows a second time.

If you have memory constraints and defined a DSQSPILL file, part of the processing time is writing the data to DSQSPILL so it can be fetched later.

The performance is affected by several things:

- The value of DSQSIROW (initial number of rows to fetch). This primarily affects the initial display of the report only.
- Whether or not you do something 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. The effect of this is usually small.
- Whether, when multiple passes are required, the data is fetched from the database the second time (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 backward you want to scroll, the complexity of the report, and other factors.

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

You get the best performance when there is sufficient memory to hold all data and DSQSPILL is not used.

Although it might not reduce the total amount of resource consumed to process your data, if you are able to get the complete answer set into virtual memory before the first display (DSQSIROW is large), the database locks are released and scrolling around the displayed report performs fastest. This slows the display of the first report screen. Releasing the locks might have the effect of improving performance for other users.

Controlling the Number of Report Rows Retrieved for Display (DSQSIROW)

Parameter name

DSQSIROW

Short form

F

Customizing Your Start Procedure

Valid values

Any number from 0 through 9 999 999

Default

A minimum of 100 rows retrieved before first screen of report is displayed

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 76 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 4096-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 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 *QMF Reference* for more information about these formatting options.

Performance with Small DSQSIROW Values

If you use too small a value for DSQSIROW, QMF might not be able to complete the data object before the first screen of data is displayed. An incomplete data object causes share locks on the data, which can prevent other users' attempts to update 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 zero for DSQSIROW unless you want to minimize the number of open read locks while QMF is retrieving or formatting data. See “Specifying an Interactive or Noninteractive QMF Session (DSQSMODE)” on page 83 for more information about noninteractive QMF sessions.

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, for example, QMF might display a message indicating that there is insufficient storage available to satisfy the user’s request.

When storage for the partition is full, QMF stops retrieving rows or terminates. When you plan your values for DSQSBSTG and DSQSIROW, remember that QMF might time out waiting for storage to become available.

Setting the Level of Trace Detail (DSQSDEBUG)

Parameter name

DSQSDEBUG

Short form

T

Valid values

ALL or NONE

Default

NONE (no trace data)

Use DSQSDEBUG 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 value and remains at ALL.

The tracing you set using 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. For more information on valid trace values, see “Getting the Right Level of Detail in Your Trace Output” on page 302.

Set DSQSDEBUG 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
```

Customizing Your Start Procedure

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.

"Specifying an Interactive or Noninteractive QMF Session (DSQSMODE)" on page 83 explains interactive and noninteractive sessions in more detail.

After QMF starts, you can turn tracing off by using the following 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 300 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

Using the parameters explained in this section, you can customize a QMF session to do work without user interaction, so that fewer resources are used. For example, you might start a noninteractive session, specify a CONNECT ID and password for the connection to the database, and run a QMF procedure that queries an inventory table and prints a report to a file for later analysis.

Although these parameters are most useful for noninteractive QMF sessions, they can also be used interactively.

Specifying the Location to Connect to When Starting QMF (DSQSDBNM)

Parameter name

DSQSDBNM

Short form

D

Valid values

Any valid database name

Default

(no default)

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 and release level of the application requestor that initiates the remote unit of work connections. The lengths for each release level are shown in Table 12.

Table 12. Application requester type and release level for DSQSDBNM length and location value length

Requester type	Release	Maximum Length
MVS™ DB2	2.3 and 3.1	16
VM SQL/DS™	3.5 only	18

Specifying an Interactive or Noninteractive QMF Session (DSQSMODE)

Parameter name

DSQSMODE

Short form

M

Valid values

B (noninteractive) or I (interactive)

Default

I

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.

Customizing Your Start Procedure

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. “Naming a Procedure to Run When QMF Starts (DSQSRUN)” explains this parameter in more detail.

Additionally, 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.

Naming a Procedure to Run When QMF Starts (DSQSRUN)

Parameter name

DSQSRUN

Short form

I

Valid values

Any valid procedure name (see *QMF Reference*)

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.

Important: QMF does not allow blanks in the user ID and procedure syntax. For example, QMF doesn't 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 so you can conserve resources normally used when running interactively.
- Allow users to perform interactive QMF work within the confines of a predefined procedure, then exit when they are finished with the work specified in the procedure.

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 something like the following query. For this example, we'll call this query INVENTORY_QUERY:

```
SELECT * FROM INVENTORY
WHERE STOCK < 20
```

The procedure you use to run this query and print the status report might look something like this one. For this example, we'll call this QMF procedure INVENTORY_PROC:

```
RUN QUERY INVENTORY_QUERY
PRINT REPORT
EXIT
```

The procedure includes an EXIT command because, when QMF is running 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 might use the DSQSRUN parameter to query the INVENTORY table off-shift 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, making it easy for 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 might not know exactly how to produce the status report. In this case, you might put a variable in the query so that the user needs only to enter the value that indicates low stock. We will call this query INVENTORY_QUERY.

```
SELECT * FROM INVENTORY
WHERE STOCK < &LOWSTOCK
```

Customizing Your Start Procedure

Because the user might want to view the data before printing it, your INVENTORY_PROC procedure might not include the EXIT command:

```
RUN QUERY INVENTORY_QUERY
```

You might then use the DSQSRUN parameter without specifying the DSQSMODE parameter, so that you start an interactive session for the user:

```
DSQQMFn I=INVENTORY_PROC
```

The INVENTORY_PROC procedure prompts the user for the &LOWSTOCK variable value. For additional examples of how to use variables with an initial procedure, see “Passing Variable Values to an Initial Procedure”. *QMF Reference* 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. Thus, pressing the End function key from the report panel reruns the initial procedure; it does not display the QMF Home panel.

Additionally, when you use the DSQSRUN parameter, ensure that the DSQEC_RERUN_IPROC global variable is set to 0 and that the current object is not the QMF Home panel. *Developing QMF Applications* 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 ensure 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.

Environment	Number of additional ampersands	Example
CMS with ISPF	1	&&variable=value
CMS without ISPF using EXEC	2	&&&variable=value
CMS with ISPF using EXEC	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, 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 *QMF Reference*.

For example, suppose you frequently need two pieces of information about employees in your organization. One piece of information is the name of the employee, and the other varies. You might define a query that includes NAME and uses a variable for the other column. Figure 8 shows how to pass a column name using DSQSRUN. 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

```
DSQQMFn I=JONES.PROC2 (&COL=YEARS)
```

Resulting RUN command

```
RUN PROC JONES.PROC2 (&COL=YEARS)
```

Figure 8. Passing a QMF column name using DSQSRUN

Figure 9 on page 88 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.

Customizing Your Start Procedure

Query (named JONES.QUERY3)

```
SELECT &COLS  
FROM Q.STAFF
```

Procedure (named JONES.PROC3)

```
RUN QUERY JONES.QUERY3 (&&COLS=&COLS
```

DSQSRUN parameter

```
DSQQMFn I=JONES.PROC3(&COLS=((DEPT,NAME, SALARY))
```

Resulting RUN command

```
RUN PROC JONES.PROC3(&COLS=((DEPT,NAME,SALARY)))
```

Figure 9. 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 *QMF Reference* for more information about the WHERE keyword.)

These examples contain character strings, for which special syntax is required because of how 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 might run a query like the one in Figure 10. When you pass the character string MGR on the DSQSRUN parameter, be sure to enclose the value in single quotes.

Figure 11 on page 89 shows how to pass variable values that contain commas.

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

```
DSQQMFn I=JONES.PROC4(&JOB='MGR')
```

Resulting RUN command

```
RUN PROC JONES.PROC4 (&JOB='MGR')
```

Figure 10. Passing a string within single quotes using DSQSRUN

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

```
DSQQMFn I=JONES.PROC5(&CITY='SAN JOSE,CA')
```

Resulting RUN command

```
RUN PROC JONES.PROC5 (&CITY='SAN JOSE,CA')
```

Figure 11. Passing a comma within a string using DSQSRUN

Figure 12 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.

Figure 13 shows how to pass values for variables in two different parts of the

Query (named JONES.QUERY6)

```
SELECT *
FROM Q.STAFF
WHERE NAME=&NAME
```

Procedure (named JONES.PROC6)

```
RUN QUERY JONES.QUERY6 (&&NAME=&NAME
```

DSQSRUN parameter

```
DSQQMFn I=JONES.PROC6(&NAME='O''BRIEN')
```

Resulting RUN command

```
RUN PROC JONES.PROC6 (&NAME='O''BRIEN')
```

Figure 12. Passing an apostrophe as part of a string using DSQSRUN

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

```
DSQQMFn I=JONES.PROC7(&V1=(((10,38))) &V2='MGR')
```

Resulting RUN command

```
RUN PROC JONES.PROC7(&V1=(((10,38))) &V2='MGR')
```

Figure 13. Passing multiple variable parameters and values using DSQSRUN

Setting Printing for Double-Byte Character Set Data (DSQSDBCS)

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 you support uses an IBM 3279 display terminal and needs to print a table (DBCSTABLE) whose nonnumeric columns contain DBCS data. The following statement starts the Uppercase NLF from a 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 9. Enabling Users to Print Objects” on page 147.

Chapter 7. The QMF Session Control Facility

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

Installing or Removing 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's 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')
```

Importing the Default System Initialization Procedure

QMF provides you with a procedure to import the default QMF system initialization procedure correctly under authorization ID of "Q" and shared between all QMF users.

Before running this procedure, ensure that the QMF command language is set to English. To do so, set the QMF global variable DSQEC_NLFCMD_LANG to 1. When the procedure is done you can return to the presiding language by setting QMF global variable DSQEC_NLFCMD_LANG to 0.

Issue the following commands:

```
IMPORT PROC FROM DSQ0BINI PROC *  
SAVE PROC AS Q.SYSTEM_INI (COM='DEFAULT QMF SYSTEM PROCEDURE' SHARE=YES)
```

The sample PROC is saved in the database as Q.SYSTEM_INI PROC with SHARE=YES.

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.

The QMF Session Control Facility

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 a 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 proc provided with QMF makes SHARE=YES the default for all users.

```
--
-- QUERY                D S Q 0 B I N I
-- MANAGEMENT          -----
-- FACILITY
--
-- Q M F   S Y S T E M   I N I T I A L I Z A T I O N   P R O C
-- ----   -
--
-- FUNCTION: PROVIDE AN EXAMPLE QMF SYSTEM INITIALIZATION PROCEDURE
--           THAT CAN BE ADDED AFTER QMF INSTALLATION. YOU MAY MOD-
--           IFY OR REPLACE THIS PROCEDURE WITH YOUR OWN VERSION.
--
--           THE PROCEDURE MUST BE STORED IN THE DATABASE UNDER THE
--           NAME OF Q.SYSTEM_INI BEFORE IT WILL RUN AUTOMATICALLY.
--           -----
--
-- THE COMMAND BELOW IS AN EXAMPLE OF ESTABLISHING A NEW DEFAULT
-- FOR THE SHARE OPTION OF THE SAVE COMMAND THAT WILL APPLY TO ALL
-- QMF USERS. (REMOVE THE LEADING COMMENT SYMBOLS "--" TO ACTIVATE
-- IT.)
--
-- SET GLOBAL ( DSQEC_SHARE=1 -- MAKE SHARE=YES THE DEFAULT FOR ALL
```

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

Figure 14. The Q.SYSTEM_INI shipped with QMF

Q.SYSTEM_INI is located in the QMF product as DSQ00BINI.

User Session Procedure Example

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

procedure. In the following example, the user session procedure is called `USER_INI`.

```

PROC                Q.SYSTEM_INI                LINE    1

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

```

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

```

PROC                WILLIAMS.USER_INI           LINE
1

-- This QMF procedure example shows how to setup QMF session defaults for
-- A QMF user. The following settings replace any settings set by the
-- SYSTEM_INI proc.
--
QMF SET GLOBAL (DSQEC_NLFCMD_LANG=1) -- Process English Commands
QMF RESET PROC                       -- Hide Contents of this PROC
QMF SET PROFILE (SPACE=MYSPEACE)     -- Store data in MYSPACE.
QMF SET PROFILE (PRINTER=MYROOM)     -- Print reports at My Printer
QMF SET GLOBAL (DSQDC_LIST_ORDER=3A) -- Object List Sorted by Object Name
QMF SET GLOBAL (DSQEC_RESET_RPT=2)   -- Always ResetReports
QMF SET GLOBAL (DSQEC_SHARE = 1)     -- Always Share My QMF Objects
QMF SET GLOBAL (DSQEC_NLFCMD_LANG=0) -- Return to Presiding Language

```

Figure 16. User session procedure example: `user.USER_INI`

Procedure that Displays an Object list

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

The QMF Session Control Facility

```
PROC                Q.SYSTEM_INI                LINE    1

-- This QMF procedure example shows how to set up QMF session defaults for
-- every QMF user to display a list of objects instead of the QMF Home
-- screen.
--
QMF SET GLOBAL (DSQEC_NLFCMD_LANG=1)  -- Process English Commands
QMF RESET PROC                        -- Hide Contents of this procedure
QMF SET GLOBAL (DSQDC_LIST_ORDER=3A)  -- Object List sorted by object name
QMF SET GLOBAL (DSQEC_NLFCMD_LANG=0)  -- Return to Presiding Language
QMF LIST ALL                          -- LIST OBJECTS FOR ENGLISH
```

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

Security and Sharing Session Procedure

The QMF session procedure Q.SYSTEM_INI and other objects used or called by this procedure take on the same security as any other QMF object or database object does during a QMF session. The Q.SYSTEM_INI procedure is not special, other than QMF tries to execute it each time a QMF session is started. If the procedure doesn't exist, then QMF doesn't try 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 300 for more information on QMF trace options.

Chapter 8. Establishing QMF Support for End Users

After you start QMF and the Home panel is displayed, you can use QMF facilities to help you customize support for end users. This chapter discusses how to set up QMF so that your end users are able to access QMF and work with data in the database.

The role of the 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 *QMF Reference*.)
- Default views for the database object list, explained in “Customizing a User’s Database Object List” on page 112.

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

Quick Start

Use the steps in Table 13 to guide you in setting up and maintaining the QMF environment for users. If you need more information, see the page shown at the right of the table.

Table 13. Establishing QMF support for end users

To do this task:	See:
Ensure users are recognized by VM by assigning them an appropriate user ID. You also need to assign an authorization ID to establish QMF and DB2 for VM authorities.	Page 96
Ensure users have a QMF profile either by allowing them to use the SYSTEM row of the Q.PROFILES table or by inserting a unique row into Q.PROFILES based on the user’s SQL authorization ID.	Page 97
Provide access to database and QMF objects your users need to work with , using SQL GRANT statements for tables and views, and the SHARE parameter of the QMF SAVE command for QMF queries, forms, and procedures.	Page 107
Customize a user’s database object list , using the DSQEC_TABS_SQL and DSQEC_COLS_SQL global variables.	Page 112
Customize the document editing interface in ISPF, using the IBM-supplied macro.	Page 139

Establishing QMF Support for End Users

Table 13. Establishing QMF support for end users (continued)

To do this task:	See:
Enable users to create tables (if necessary) by assigning a private dbspace or by granting DB2 for VM RESOURCE authority and assigning a public dbspace.	Page 116
Enable users to support a chart using the Interactive Chart Utility (ICU) of GDDM.	Page 120
Maintain your users' queries, forms, and procedures by updating and reorganizing the QMF object control tables (Q.OBJECT_DIRECTORY, Q.OBJECT_DATA, and Q.OBJECT_REMARKS).	Page 120
When necessary, enlarge the dbspace for the QMF object control tables using the DB2 for VM DBS utility UNLOAD and RELOAD commands. Recreate indexes and any views you defined on the tables.	Page 126
Maintain your users' database tables and views by updating and reorganizing DB2 for VM system tables.	Page 127

Ensuring That Users Have Access to CMS

Provide a new user with a VM logon ID. Set up the new users as you would an 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 give 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 need not be reissued. You should plan to log on with the new user ID and give the SQLINIT 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. For information about how the CONNECT command is used in running batch jobs, see "Chapter 15. Customizing the Batch Processing Program" on page 277. For information about connecting to a different database during a QMF session, see "Chapter 14. Customizing a Remote Database Connection" on page 263.

After a user has received CONNECT authority (has been assigned an 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 8 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 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 need not 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 that 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 *DB2 Server for VM Database Administration* See also “Required Entry Fields” on page 284.

Creating User Profiles to Enable User Access to QMF

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 100 shows the Q.PROFILES table in detail and discusses possible profile values.

Using the Q User Profile, a Special QMF Profile

QMF installation automatically grants DBA authority to the user ID Q. The user Q owns and manages these QMF resources:

- All QMF control tables, shown in “Appendix D. QMF Control Tables and dbspaces Used by QMF” on page 325.
- The sample tables shipped with QMF. (For descriptions of the sample tables, see *QMF Reference*.)

Establishing QMF Support for End Users

- Default views for the database object list, explained in “Customizing a User’s Database Object List” on page 112.

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

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 100. 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”. Set the CREATOR column of Q.PROFILES to the SQL 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 don’t have unique profiles from using QMF.

Adding a New User Profile to the Q.PROFILES Table

You can use SQL INSERT queries or the QMF Table Editor (described in *Using QMF*) to add new user profiles to the Q.PROFILES table. Figure 18 on page 99 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, as shown, to distinguish between an English and an NLF environment.

The values shown in the figure are examples of profile values you can use. See Table 14 on page 101 for other valid profile values.

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', 'STUTZBER'
'DEUTSCH', 'DEUTASTEN'
'COMMAND_SYNONYM_D', 'SCHICHT'
'CICSVSE')
```

Figure 18. Creating a user profile

Important: Always specify a TRANSLATION value when inserting a row into Q.PROFILES, or the TRANSLATION value defaults to a null value and the profile row is automatically ignored. Figure 18 shows only a subset of all possible profile values. Use “Reading the Q.PROFILES Table” on page 100 for guidance in specifying additional values.

To enroll many users, set up a template query that describes a standard profile and 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 *QMF Reference*.

If you’re 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 row of Q.PROFILES. Figure 19 on page 100 shows SQL statements that delete this row. You can also use the Table Editor, as explained in *Using QMF*.

Establishing QMF Support for End Users

**Base QMF (English)
German NLF**

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

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

Important: 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 won't be able to use QMF. An example of creating a unique profile is shown in Figure 18 on page 99.

Reading the Q.PROFILES Table

Table 14 on page 101 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're 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. Browse the DSQ2\$UPO EXEC to see the default values for each NLF. (Replace the n symbol with an NLID from Table 5 on page 19.)

The Q.PROFILES table has the index Q.PROFILEX, with the attribute UNIQUE. The keyed columns are CREATOR, TRANSLATION, and ENVIRONMENT. No three rows can have identical values for these three columns.

Table 14. Structure of the Q.PROFILES table

Column Name	Data Type and Length	Nulls Allowed	Function and Possible Values
CREATOR	CHAR (8)	No	<p>Function: Specifies the SQL authorization ID (the user) who owns the profile.</p> <p>Values: SQL authorization ID or SYSTEM (default). The SYSTEM row is shipped with Q.PROFILES for English and each NLF; users who don't have unique profile rows can use the SYSTEM row.</p>
CASE	CHAR (18)	Yes	<p>Function: Specifies whether terminal input is converted to uppercase.</p> <p>Values: UPPER (default), STRING, or MIXED. See <i>QMF Reference</i> for descriptions of these values. CASE might have a different default for NLF users.</p>
DECOPT	CHAR (18)	Yes	<p>Function: Specifies what separators QMF puts in numeric report columns.</p> <p>Values: PERIOD (default), COMMA, and FRENCH. See <i>QMF Reference</i> for more information. DECOPT is translated and might have a different default for NLF users.</p>
CONFIRM	CHAR (18)	Yes	<p>Function: Controls display of confirmation panels.</p> <p>Values: YES (default) if you want confirmation panels displayed before database changes; NO if you don't. See page 119 for information on confirming table changes. CONFIRM might have a different default for NLF users.</p>
WIDTH	CHAR (18)	Yes	<p>Function: Controls number of printed columns per page.</p> <p>Values: 22 to 999. Default = 132.</p>
LENGTH	CHAR (18)	Yes	<p>Function: Controls number of printed lines per page.</p> <p>Values: 1 to 999, or CONT if you want no page breaks. Default = 60.</p>

Establishing QMF Support for End Users

Table 14. Structure of the Q.PROFILES table (continued)

Column Name	Data Type and Length	Nulls Allowed	Function and Possible Values
LANGUAGE	CHAR (18)	Yes	<p>Function: Controls which query language QMF uses when creating a new query after a RESET QUERY command is issued.</p> <p>Values: SQL (default), QBE (for Query-by-Example), or PROMPTED (for Prompted Query). LANGUAGE might have a different default for NLF users.</p>
SPACE	CHAR (50)	Yes	<p>Function: Specifies a dbspace that holds tables created using SAVE DATA and IMPORT commands. In DB2 Parallel Edition, this value refers to a NODEGROUP name. However, QMF 7.1 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.</p> <p>Values: Any valid dbspace name. See “Choosing and Acquiring a dbspace for the User” on page 118 for more information on using dbspaces.</p>
TRACE	CHAR (18)	Yes	<p>Function: Controls the level of detail in trace output.</p> <p>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. NONE inhibits normal levels of tracing. The default varies depending on the value for DSQSMODE. For example, when DSQSMODE is B, the trace level is L2, otherwise it is NONE. See “Using the QMF Trace Facility” on page 300 for more information on the QMF trace facility. See “Setting the Level of Trace Detail (DSQSDEBUG)” on page 81 to specify a trace value when QMF starts. Only the values ALL and NONE are translated in NLFs.</p>

Table 14. Structure of the Q.PROFILES table (continued)

Column Name	Data Type and Length	Nulls Allowed	Function and Possible Values
PRINTER	CHAR (8)	Yes	<p>Function: Controls where printer output is routed.</p> <p>Values: Use a null (default) or blank value to route print output to a file or a printer that is associated with the DSQPRINT FILEDEF. Use a GDDM nickname to direct output to a GDDM-defined printer. See “Chapter 9. Enabling Users to Print Objects” on page 147 for information on choosing and specifying values.</p>
TRANSLATION	CHAR (18)	No	<p>Function: Indicates English or NLF environment</p> <p>Values: English (default) or the name of an NLF. The right-hand column of Table 5 on page 19 shows the translated names you need to use in this column.</p>
PFKEYS	VARCHAR (31)	Yes	<p>Function: Indicates the table or view (if any) where user’s customized function key definitions are stored.</p> <p>Values: Any valid DB2 VM table or view name. If blank or null (default), QMF’s default keys are used. “Chapter 11. Customizing QMF Function Keys” on page 173 describes how to create this table.</p>
SYNONYMS	VARCHAR (31)	Yes	<p>Function: Indicates the table or view (if any) where user’s customized command definitions are stored.</p> <p>Values: Any valid DB2 for VM table or view name. If blank or null (default), no customized definitions are used. “Chapter 10. Customizing QMF Commands” on page 159 describes how to create this table. For NLF users, the IBM-supplied table is named Q.COMMAND_SYNONYM_n, where n is the National Language ID.</p>

Establishing QMF Support for End Users

Table 14. Structure of the Q.PROFILES table (continued)

Column Name	Data Type and Length	Nulls Allowed	Function and Possible Values
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). See "Chapter 13. Controlling QMF Resources Using a Governor Exit Routine" on page 227 for more information.
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: CMS or null. If profiles are stored in DB2 for VM but are being accessed from a DB2 application requester, the value can be any one of the following: TSO, CICS [®] , MVS, or CICS.

Providing the Correct Profile for the User's Operating Environment

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=SQL ID, ENVIRONMENT=current operating environment
2. CREATOR=SQL ID, 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, CICS, CICSMVS, or TSO, when QMF is being started from CMS, CICS, or TSO, respectively.

The value for current operating environment can also be CICSMVS, CICS, or TSO if the profiles are stored in VM DB2 for VM but are being accessed from a DB2 application requester.

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 *Using QMF*.

Using SQL UPDATE Statements

SQL UPDATE statements can be used to update all fields of the Q.PROFILES table, including SYNONYMS, PFKEYS, and RESOURCE_GROUP. See Table 14 on page 101 for descriptions of these columns, including consequences of not specifying their values.

For more information about how to choose values for these columns, see:

- “Chapter 10. Customizing QMF Commands” on page 159
- “Chapter 11. Customizing QMF Function Keys” on page 173
- “Chapter 13. Controlling QMF Resources Using a Governor Exit Routine” on page 227

Use an SQL UPDATE query similar to the one in Figure 20 on page 106 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.

Establishing QMF Support for End Users

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

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

Important: 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 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 as shown in the example in Figure 18 on page 99.

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 SQL authorization ID in that profile have been either deleted or safely transferred to other users:

- For how to perform these tasks for QMF queries, forms, and procedures, see “Maintaining QMF Objects Using QMF Control Tables” on page 120.
- For instructions for database tables and views, see “Maintaining Tables and Views Using DB2 for VM System Tables” on page 127.

When you delete a user profile, all SQL privileges the user had on objects are deleted, as well as all privileges that user granted to other users. To ensure other users won't 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 for VM authorities have been granted. For sample queries you can use, see “Transferring Ownership of Queries, Forms, and Procedures” on page 124.

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

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

Figure 21. Deleting a QMF user profile

If you're 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 don't specify a value for TRANSLATION, QMF deletes the profile in *all* NLF environments.

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. *DB2 Server for VSE & VM SQL Reference* explains the DROP statement. *QMF Reference* explains the ERASE command.

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 might 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 109. "SQL Privileges Required to Access Objects" 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 111.

SQL Privileges Required to Access Objects

Before users can use certain SQL statements with tables or views, you need to grant them the SQL privileges they need. For example, if user JONES enters

Establishing QMF Support for End Users

DISPLAY TABLE SALES_TOTALS but does not have the SQL SELECT privilege for the SALES_TOTALS table, QMF displays the following message:

You lack the authorization needed for this DISPLAY command.

To prevent JONES from getting this kind of error message, grant him the SQL SELECT privilege on the SALES_TOTALS table.

Different SQL privileges are required, depending on whether the user is executing a QMF command, running a prompted or QBE query, or using the Table Editor.

SQL Privileges Required for QMF Commands

Using Table 15, 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 109 for examples of SQL GRANT statements.

Table 15. 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. See "SQL Privileges Required for the Table Editor" on page 109 for this information.
EXPORT TABLE table/view	SELECT
IMPORT TABLE table/view	If the table exists, SELECT, DELETE, and INSERT. If the table does not exist, INSERT. Authority is also required to use the CREATE TABLE statement for the dbspace specified in the SPACE field of the 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. If the table does not exist, CREATE TABLE.
LIST table/view	SELECT

Not all users can use the SAVE command to create a new table. For more information, see "Enabling Users to Create Tables in the Database" on page 116.

For more information on SQL privileges, such as SELECT, INSERT, UPDATE, or DELETE, see *DB2 Server for VSE & VM SQL Reference*

SQL Privileges Required for Prompted and QBE Queries

Using Table 16, 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 16. 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 17, 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 17. 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 for VM DBA authority can grant or revoke SQL privileges for any object in the database. The user Q has this authority, and is predefined to DB2 for VM 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:

Establishing QMF Support for End Users

```
JONES.ORDER_BACKLOG
```

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 22 to grant JONES the SQL UPDATE privilege he needs to edit the ORDER_BACKLOG table in change mode:

```
WITH GRANT OPTION indicates that JONES can grant to other users any of  
GRANT UPDATE ON ORDER_BACKLOG TO JONES WITH GRANT OPTION
```

Figure 22. Granting SQL privileges to a single QMF user

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 *QMF Reference*. You might also consider using a QMF procedure to do the task if there is more than one query. *Using QMF* explains how to create procedures.

Use the keyword PUBLIC to grant SQL privileges to *all* QMF users. For example, use the statement in Figure 23 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 23. Granting an SQL privilege to all QMF users

For more information of the GRANT statement, see *DB2 Server for VSE & VM SQL Reference*

Important: 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. Users editing tables required during QMF initialization can hold locks on the table that prevent QMF from starting for other users.

Using the SQL REVOKE Statement

Use the SQL REVOKE statement to take back privileges granted:

```
REVOKE UPDATE ON ORDER_BACKLOG FROM JONES
```

Figure 24. Revoking an SQL privilege from a QMF user

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

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

For more information of the REVOKE statement, see *DB2 Server for VSE & VM SQL Reference*

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 in Figure 25 saves the current query as ORDER_QUERY and allows any other user to display and run it: The default is defined by the global variable DSQEC_SHARE. See *QMF SAVE QUERY AS ORDER_QUERY (SHARE=YES*

Figure 25. Sharing a QMF object

Reference for details.

The owner of an object can change its shared status at any time, using a DISPLAY command followed by a SAVE command, as shown in Figure 26:

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

Figure 26. Changing the shared status of a QMF object

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

Allowing Uncommitted Read

If you want your QMF session to allow uncommitted read, you can specify a value for the global variable DSQEC_ISOLATION in the Q.SYSTEM_INI procedure.

Establishing QMF Support for End Users

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 7.1 the use of the value '0' is only effective with the following database servers (those supporting the SQL with-clause):

- DB2 for MVS V4 or higher
- DB2 for VM/VSE V4 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 for VM system tables to obtain information about the table. The name of this view is stored in the global variable DSQEC_TABS_SQL. When users issue these commands for a column within a table, QMF uses the global variable 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

QMF provides default views and automatically assigns these views to the user Q during QMF installation:

```
Q.DSQEC_TABS_SQL
Q.DSQEC_COLS_SQL
```

QMF supplies a variation of the following views when QMF is installed into DB2 Common Servers:

```
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 27 shows the type of information the view provides. 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, TNAME, 'TABLE', TABLETYPE, ' ', ' ', 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 for VM TABLES/VIEWS LIST';
GRANT SELECT ON Q.DSQEC_TABS_SQL TO PUBLIC;
```

Figure 27. Default view that provides a list of tables for the LIST command

one:

```
SET GLOBAL (DSQEC_TABS_SQL = userid.your_local_sql_table
```

The view Q.DSQEC_COLS_SQL selects only the column information a user is authorized to see. Figure 28 on page 114 shows the type of information the view provides.

Establishing QMF Support for End Users

```
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 28. 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.your_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 for VM gathers authorization information from the SYSTEM.SYSCATALOG and SYSTEM.SYSCOLUMNS tables. If you don't 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 29 to create your own view. This query eliminates duplicate rows in the view and, although DB2 for VM 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 *QMF Reference* for information on QMF naming conventions.

To override the view you created, you can issue a command similar to the

```
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 29. Customizing your object lists using global variables

following:

```
SET GLOBAL (DSQEC_COLS_SQL = userid.your_local_sql_objects
```

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 DBAs, 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 DBAs can name this view in the DSQEC_TABS_SQL or DSQEC_COLS_SQL 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_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 *Developing QMF Applications*.

Object List Storage Requirement

For the LIST command, there are two sets of storage requirements for each row of the object list.

- The QMF internal RPT record collection requires:
 - Object OWNER key information, 50 bytes
 - REMARKS, up to 254 bytes
 - TABLE with a LABEL, up to 30 bytes
 - ALIAS, 42 bytes
 - Object information for QUERY, PROC, and FORM, 63 bytes

Establishing QMF Support for End Users

- 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.

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 table attributes don't match. For more information on the SAVE DATA command, see *QMF Reference* or 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 table attributes don't match. For more information on the IMPORT command, see *QMF Reference* or 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. Both methods are shown in Table 18 on page 117.

Table 18. Creating tables in the database

If you're creating tables for your users:	If users are creating tables themselves:
<p>Step 1 Acquire a dbspace as shown in Figure 30 on page 118 and define it to DB2 for VM before its first use. Use <i>DB2 Server for VM Database Administration</i> to help you decide on a private or public dbspace.</p>	<p>Step 1 Acquire a dbspace as shown in Figure 30 on page 118 and define it to DB2 for VM before its first use. Use <i>DB2 Server for VM Database Administration</i> to help you decide on a private or public dbspace.</p>
<p>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 <i>Using QMF</i> for examples of creating tables.</p>	<p>Step 2 Assign the dbspace in the user's QMF profile, using an SQL UPDATE statement for the SPACE field. Updating profiles is explained in "Updating User Profiles" on page 105. You can update the SYSTEM profile if you need to change its default values.</p>
<p>Step 3 Create one or more indexes on the tables you create, to improve DB2 for VM performance. See <i>DB2 Server for VSE & VM SQL Reference</i> for information on the CREATE INDEX statement and details on logical design of tables.</p>	<p>Step 3 Grant DB2 for VM 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.</p>
<p>Step 4 Fill the tables with data. Use the Db2 for VM DBS Utility, QMF IMPORT commands (for transferring small tables), or other methods. <i>DB2 Server for VSE & VM Database Services Utility</i> explains how to use the DBS Utility. <i>Using QMF</i> explains exporting and importing objects in QMF.</p>	<p>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 <i>QMF Reference</i> for more information on these commands.</p>
<p>Step 5 Grant SQL privileges for the tables to users who need them, as discussed in "SQL Privileges Required to Access Objects" on page 107.</p>	<p>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 can't read the data to create a new table.</p> <p>SQL privileges for QMF functions and commands are discussed starting in "SQL Privileges Required to Access Objects" on page 107.</p>

Establishing QMF Support for End Users

For more information on the CREATE TABLE, CREATE INDEX, and other SQL statements related to creating tables, see *DB2 Server for VSE & VM SQL Reference*

Choosing and Acquiring a dbspace for the User

A dbspace can be either private or public. Any QMF user with DB2 for VM 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 dspsaces, see *DB2 Server for VM Database Administration*

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 Figure 30. You can enter this statement from the QMF SQL query panel, then press the Run function key to run the query.

Substitute PRIVATE for PUBLIC in the statement if you're acquiring a private

```
ACQUIRE PUBLIC DBSPACE NAMED dbspacename  
(PAGES = 1024)
```

Figure 30. Acquiring a dbspace

dbspace, and be sure to qualify dbspacename with the SQL authorization ID of the user for whom you're acquiring the dbspace.

Sizing a dbspace

The size of the dbspace in an acquire statement is given in *pages*, where one page is 4096 bytes. If you don't 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're 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 for VM 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 a dbspace. Instructions for adding dspsaces are provided in *DB2 Server for VM System Administration*

Granting a User DB2 for VM RESOURCE Authority

You need to grant DB2 for VM 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 in Figure 31 on page 119, where *userid1*, *userid2*, and *userid3*, represent SQL authorization IDs.

```
GRANT RESOURCE TO userid1, userid2, userid3, ...
```

Figure 31. SQL statements to grant RESOURCE authority to more than one user

A user with RESOURCE authority can:

- Acquire a private dbspace for his or her own use
- 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 resource is 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 resource used.

See *DB2 Server for VM Database Administration* for more information on acquiring a dbspace and a discussion of DB2 for VM 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 for VM tables stored in nonrecoverable dbspaces cannot be rolled back, or canceled, answering NO on the Table Editor confirmation prompt panel for database changes doesn't prevent the changes to the table from taking place.

As end users become more comfortable changing data in the database, they might 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
- DSQCP_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.

Establishing QMF Support for End Users

- 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 *Using QMF*.

Enabling Users to Support a Chart

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, if they have libraries in which to store them.

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.

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 dbspace for the tables when it is no longer large enough to hold existing QMF objects.

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 121
- The Q.OBJECT_DATA table, which is described in “Reading the Q.OBJECT_DATA Table” on page 122
- The Q.OBJECT_REMARKS table, which is described in “Reading the Q.OBJECT_REMARKS Table” on page 123

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.

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 the UNIQUE attribute. 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 19:

Table 19. Structure of the Q.OBJECT_DIRECTORY table

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
OWNER	CHAR	8	No	Shows the SQL 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.
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 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.

Establishing QMF Support for End Users

Table 19. Structure of the Q.OBJECT_DIRECTORY table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
LAST_USED	DATE		Yes	Shows the date value for when an object was last used. The value is updated only once each day the object is accessed. Note that the LAST_USED value may not be updated, for performance reasons, when using a QMF object while the current QMF report is not yet complete.

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_OBJDATA, with the UNIQUE attribute. Keyed columns are OWNER, NAME, and SEQ.

The table has the structure shown in Table 20:

Table 20. Structure of the Q.OBJECT_DATA table

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
OWNER	CHAR	8	No	Shows the SQL 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.
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.

Table 20. Structure of the Q.OBJECT_DATA table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
APPLDATA	LONG VARCHAR (see note)	3600 (see note)	Yes	Contains all or a portion of text that defines the object. Text appears in an internal QMF format. The OBJECTLEVEL column in Q.OBJECT_DIRECTORY defines this format.

Note: With DataJoiner V1.2.1 and DB2 for AIX, Parallel Edition V1.2, the data type and length for APPLDATA are VARCHAR(3600). This is a permanent restriction for V1 SQL databases.

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 UNIQUE attribute. Keyed columns are OWNER and NAME.

The table has the structure shown in Table 21:

Table 21. Structure of the Q.OBJECT_REMARKS table

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
OWNER	CHAR	8	No	Shows the SQL authorization ID of the user who created the object
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 DBA authority you can list QMF objects you do not own using the query in Figure 32 on page 124.

Establishing QMF Support for End Users

```
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 32. 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 doesn't provide enough information in the `REMARKS` column, try displaying the object by one of the following methods:

- Connecting to the database using the user's SQL authorization ID. For example, to connect as user JONES who has a password of MYPW:

```
CONNECT JONES (PA=MYPW)
```

Then issue the QMF `DISPLAY` command for each object you want to display.

- Running the following query to share the user's objects, then displaying them from your own ID:

Enclose the value you supply for `userid` in single quotes.

```
UPDATE Q.OBJECT_DIRECTORY
SET RESTRICTED = 'N'
WHERE OWNER = 'userid'
```

Figure 33. Sharing another user's objects with all users

Important: Run this query only if you don't need to track which of the user's objects are restricted and which are not. After you run this query, you can set `RESTRICTED` back to `Y`, but you won't know which objects were originally restricted.

Transferring Ownership of Queries, Forms, and Procedures

Use the queries shown in Figure 34 on page 125 to transfer QMF objects from one user to another. Ensure you run all three queries.

Important: First make sure that the new owner has no objects saved with the name of the object you're transferring, or QMF replaces the existing object with the object 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 34. Transferring QMF objects to another user

In the queries shown in Figure 34, `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 33 on page 124 to change the `RESTRICTED` column value to `Y` if you decide you want to share the object after transferring it.

The user might also have CMS files containing queries, forms, and procedures. You use the QMF `IMPORT` command to save those of interest to other users. Using this command saves them under the current DB2 for VM user ID, and you can then transfer their ownership to another user ID.

Deleting Obsolete Queries, Forms, and Procedures

Use the SQL in Figure 35 to delete *all* of a particular user's QMF queries, forms, and procedures. Ensure 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 35. Deleting unnecessary objects from the QMF control tables

Establishing QMF Support for End Users

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 data last used was before 06/01/95 and delete all the appropriate rows from the three control tables.

Enlarging the dbspace for the QMF Object Control Tables

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 you need it.
2. Unload the dbspace to a CMS sequential file using the UNLOAD dbspace command of the DBS utility.

Table 22 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 4096-byte block.

Table 22. 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 using as the input file the file you specified when you unloaded the tables. 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_OBJDATA 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.DSQC_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')
```

Figure 36. Recreating a view after dropping dbspaces

to PUBLIC:

```
GRANT SELECT ON Q.DSQC_QMFOBJS TO PUBLIC
```

8. Alter the dspace to allow the free space on occupied pages to be used.  
For example:
- ```
ALTER DBSPACE PUBLIC.DSQT SCT1 (PCTFREE=5)
```
9. If you change the QMF control tables, reload the QMF SQL packages with the install exec DSQ2PREP.

For more information on enlarging dbspaces, see *DB2 Server for VM Database Administration* For instructions and syntax of the DBS utility and ISQL commands, see *DB2 Server for VSE & VM Database Services Utility* and *DB2 Server for VSE & VM SQL Reference*

Maintaining Tables and Views Using DB2 for VM System Tables

Anyone with DBA authority can access the DB2 for VM tables to list, display, transfer, or delete tables and views. For complete information on using these DB2 for VM system tables, see *DB2 Server for VSE & VM SQL Reference*

Establishing QMF Support for End Users

Listing Tables and Views

The query in Figure 37 returns a list of tables with columns TABLETYPE (R indicates a table, V indicates a view), TNAME (tablename), DBSPACENAME, and REMARKS.

```
SELECT TABLETYPE, TNAME, DBSPACENAME, REMARKS
FROM SYSTEM.SYSCATALOG
WHERE CREATOR = 'userid'
ORDER BY TABLETYPE, TNAME
```

Figure 37. Listing DB2 for VM tables and views owned by a particular user

Transferring Ownership of a Table or View

Transferring ownership of a table or view is not recommended.

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).

For more information on erasing tables, see *DB2 Server for VM Database Administration*

Supporting Locally Defined Date/Time Formats

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 these are DB2 for VM exits, *not* QMF exits. For details about how these exits are created refer to *DB2 Server for VM System Administration*

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 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
```

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 for VM tables. Possible data sources include IMS™, VSAM, physical sequential files, and tables from other DB2 for VM 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

Supporting the EXTRACT Command

To support the EXTRACT command you must:

- Make files available to the users of that command
- 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 files are described in the *Data Extract: Planning and Administration Guide for Dialogs*. If you are enrolling DXT dialog users, you need to read that document. If you are not, all you need to know about the process is included in the following discussion.

Allocating Resources

QMF can support English, Kanji, and Uppercase (UCF) DXT dialogs. The different DXT files that are required by these dialogs are allocated with ISPF LIBDEF statements (more about using LIBDEF shortly). The files needed for Version 2 Release 2 dialogs are the same for Version 2 Release 3.

Table 23 on page 130 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).

Establishing QMF Support for End Users

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 23. 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. QMF calls one of these just before the execution of an EXTRACT command, and the other just after the execution ends. With no modifications, these EXECs do nothing. But with suitable changes, the first can allocate the added resources, and the second can reallocate them.

Figure 38 on page 132 shows an 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.

Figure 38 shows 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.

Establishing QMF Support for End Users

```

/*****
/* REMOVE THE FOLLOWING STATEMENT TO ACTIVATE EXEC */
/*****
EXIT 0
TRACE OFF
/*****
/* EXEC NAME: DSQABX2L */
/*
/* DESCRIPTIVE NAME: DXT/END USER DIALOGS LIBRARY ALLOCATIONS */
/* EXEC FOR THE QMF-DXT BRIDGE */
/*
/* COPYRIGHT: 5645-DB2, 5648-A70 (C) COPYRIGHT IBM CORP. */
/* 1982, 1998 */
/* (Published) */
/* LICENSED MATERIAL - PROGRAM PROPERTY OF IBM */
/* ALL RIGHTS RESERVED */
/* U.S. GOVERNMENT USERS RESTRICTED RIGHTS */
/* - USE, DUPLICATION OR DISCLOSURE RESTRICTED BY */
/* GSA ADP SCHEDULE CONTRACT WITH IBM CORP. */
/*
/* STATUS: VERSION 7 RELEASE 1 LEVEL 0 */
/*
/* FUNCTION: */
/* THIS EXEC IS CALLED PRIOR TO CALLING THE DXT PRODUCT. THIS */
/* EXEC ALLOWS THE USER TO ALLOCATE DXT LIBRARIES PRIOR TO */
/* STARTING THE DXT PRODUCT. IF YOU ALLOCATED DXT LIBRARIES */
/* PRIOR TO STARTING QMF YOU SHOULD NOT HAVE TO MODIFY THIS */
/* EXEC. IN WHICH CASE THE EXEC SIMPLY EXITS WITH A ZERO */
/* RETURN CODE. */
/*
/* IF YOU DID NOT ALLOCATE DXT LIBRARIES PRIOR TO STARTING */
/* THE QMF PRODUCT YOU WILL NEED TO ALLOCATE THEM USING THIS */
/* EXEC. IF YOU ALLOCATE DXT LIBRARIES USING THIS EXEC YOU */
/* WILL NEED TO CHANGE EXEC "DSQABX2F" WHICH IS EXECUTED */
/* UPON COMPLETION OF THE DXT PRODUCT. */
/*
/* IF YOUR DXT PRODUCT IS NOT THE ENGLISH VERSION, YOU MUST */
/* SET THE LANGUAGE KEY TO THE PROPER VALUE. SEE VARIABLE */
/* "LKEY" IN THIS EXEC FOR CURRENT VALUE. */
/*
/* IF YOU HAVE TAKEN ADVANTAGE OF THE DXT DIALOGS OBJECT */
/* SHARING CAPABILITY, YOU WILL NEED TO SET VARIABLE "OBJSHR" */
/* TO A VALUE OF "YES". BY DOING THIS YOU WILL ALLOCATE THE */
/* SHARED DVRTLIB LOCATED ON THE DXT PRODUCTION DISK. IF YOU */
/* ARE NOT USING OBJECT SHARING, SET THE VARIABLE "OBJSHR" */
/* TO A VALUE OF "NO". */

```

Figure 38. EXEC to allocate DXT CMS files (DSQABX2L) (Part 1 of 4)

Establishing QMF Support for End Users

```

/*                                                    */
/* INPUT: NONE                                        */
/*                                                    */
/* OUTPUT: NONE                                       */
/*                                                    */
/* EXIT CONDITIONS: NONE                             */
/*                                                    */
/* ABEND CODE: VALUE - NONE                          */
/*                                                    */
/* EXTERNAL REFERENCES:                              */
/* ROUTINES: NONE                                    */
/* DATA AREAS: NONE                                 */
/*                                                    */
/* CHANGE ACTIVITY: NONE                             */
/*                                                    */
/*****/
/*-END-OF-SPECIFICATION-*****/

/*****/
/* SET DXT PRODUCT LANGUAGE CODE AND OBJECT SHARING */
/*****/
LKEY = 'E'                                           /* SET ENGLISH LANGUAGE KEY */
OBJSHR = 'NO'                                       /* SET OBJECT SHARING VARIABLE */

/*****/
/* LINK TO DXT PRODUCTION DISK                       */
/*****/
OWNERID      = 'DXT'                                /* DXT PRODUCTION DISK OWER ID */
OWNER_ADDRESS = '191'                               /* DXT PRODUCTION DISK ADDRESS */
PW           = 'DXTREAD'                           /* DXT PRODUCTION DISK READ PW */
USER_ADDRESS = '291'                               /* ACCESS ADDRESS OF USER     */
UMODE       = 'P'                                  /* ACCESS FILE MODE OF USER   */

ADDRESS CMS 'SET CMSTYPE HT'
ADDRESS COMMAND 'CP LINK' OWNERID OWNER_ADDRESS USER_ADDRESS 'RR' PW
IF RC ^= 0 THEN EXIT RC
ADDRESS CMS 'ACCESS' USER_ADDRESS UMODE
IF RC ^= 0 THEN EXIT RC
ADDRESS CMS 'SET CMSTYPE RT'

/*****/
/* SET ISPF ERRORS TO RETURN TO THIS EXEC          */
/*****/
ADDRESS ISPEXEC 'CONTROL ERRORS RETURN'
IF RC ^= 0 THEN EXIT RC

```

Figure 38. EXEC to allocate DXT CMS files (DSQABX2L) (Part 2 of 4)

Establishing QMF Support for End Users

```
/* ***** */
/* ISPF LIBDEF STATEMENTS FOR DXT FOLLOW: */
/* ***** */

LIBS = 'DVRLOAD TXTLIB' UMODE
ADDRESS ISPEXEC 'LIBDEF ISPXLIB FILE ID('LIBS')'
      IF RC = 0 THEN EXIT RC

LIBS = 'DVRPLIB'LKEY' MACLIB' UMODE
ADDRESS ISPEXEC 'LIBDEF ISPPLIB FILE ID('LIBS')'
      IF RC = 0 THEN EXIT RC

LIBS = 'DVRMLIB'LKEY' MACLIB' UMODE
ADDRESS ISPEXEC 'LIBDEF ISPMLIB FILE ID('LIBS')'
      IF RC = 0 THEN EXIT RC

LIBS = 'DVRJEDI'LKEY' MACLIB A , DVRSLIB'LKEY' MACLIB' UMODE
ADDRESS ISPEXEC 'LIBDEF ISPSLIB FILE ID('LIBS')'
      IF RC = 0 THEN EXIT RC

LIBS = 'DVRTLIB'LKEY' MACLIB A , DVRTADM'LKEY' MACLIB' UMODE
ADDRESS ISPEXEC 'LIBDEF ISPTLIB FILE ID('LIBS')'
      IF RC = 0 THEN EXIT RC

ADDRESS ISPEXEC 'LIBDEF ISPTABL FILE ID(DVRTLIB'LKEY' MACLIB A) '
      IF RC = 0 THEN EXIT RC

IF OBJSHR = 'NO' THEN
  DO
    ADDRESS ISPEXEC 'LIBDEF DVRDJEDI FILE ID(DVRJEDI'LKEY' MACLIB A)'
      IF RC = 0 THEN EXIT RC
  END
ELSE
  DO
    ADDRESS ISPEXEC 'LIBDEF DVRDJEDI FILE ID(DVRJEDI'LKEY' MACLIB 'UMODE' )'
      IF RC = 0 THEN EXIT RC
  END
END
```

Figure 38. EXEC to allocate DXT CMS files (DSQABX2L) (Part 3 of 4)

Establishing QMF Support for End Users

```
ADDRESS ISPEXEC 'LIBDEF DVRDJEDO FILE ID(DVRJEDI'LKEY' MACLIB A)'  
  IF RC ^= 0 THEN EXIT RC  
  
ADDRESS ISPEXEC 'LIBDEF DVRDIMEX FILE ID(DVRIMEX'LKEY' MACLIB A)'  
  IF RC ^= 0 THEN EXIT RC  
  
ADDRESS ISPEXEC 'LIBDEF DVREUADD FILE ID(DVRTADM'LKEY' MACLIB' UMODE ')'  
  IF RC ^= 0 THEN EXIT RC  
  
IF OBJSHR = 'YES' THEN  
  DO  
    ADDRESS ISPEXEC 'LIBDEF DVRSTABL FILE ID(DVRTLIB'LKEY' MACLIB' UMODE ')'  
    IF RC ^= 0 THEN EXIT RC  
  END  
  
EXIT 0
```

Figure 38. EXEC to allocate DXT CMS files (DSQABX2L) (Part 4 of 4)

Preparing the Reallocation EXEC

The EXEC is shown in Figure 39 on page 136. It is named DSQABX2F and is located on the QMF production disk. QMF calls this EXEC 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 aren't 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 (see “Preparing the Allocation EXEC” on page 130).

Establishing QMF Support for End Users

```
/******  
/* REMOVE THE FOLLOWING STATEMENT TO ACTIVATE EXEC */  
/******  
EXIT 0  
TRACE OFF  
/******  
/* EXEC NAME: DSQABX2F */  
/* */  
/* DESCRIPTIVE NAME: DXT/END USER DIALOGS LIBRARY FREE */  
/* EXEC FOR THE QMF-DXT BRIDGE */  
/* */  
/* COPYRIGHT: 5645-DB2, 5648-A70 (C) COPYRIGHT IBM CORP. */  
/* 1982, 1998 */  
/* (Published) */  
/* LICENSED MATERIAL - PROGRAM PROPERTY OF IBM */  
/* ALL RIGHTS RESERVED */  
/* U.S. GOVERNMENT USERS RESTRICTED RIGHTS */  
/* - USE, DUPLICATION OR DISCLOSURE RESTRICTED BY */  
/* GSA ADP SCHEDULE CONTRACT WITH IBM CORP. */  
/* */  
/* STATUS: VERSION 7 RELEASE 1 LEVEL 0 */
```

Figure 39. EXEC to reallocate DXT CMS files (DSQABX2F) (Part 1 of 3)

Establishing QMF Support for End Users

```

/*      FUNCTION:                                     */
/*      THIS EXEC IS CALLED WHEN THE DXT PRODUCT HAS ENDED. THIS */
/*      EXEC IS USED TO FREE ANY ALLOCATIONS MADE BY THE EXEC   */
/*      "DSQABX2L" AND REALLOCATE QMF LIBRARIES IF THE "LIBDEF"  */
/*      FUNCTION WAS USED TO ALLOCATE DXT PRODUCT LIBRARIES.    */
/*      IF YOU ALLOCATED ALL OF YOUR DXT LIBRARIES BEFORE YOU   */
/*      STARTED QMF OR ISPF, YOU SHOULD NOT MODIFY THIS EXEC.  */
/*      THE EXEC THAT IS DISTRIBUTED BY THE QMF PRODUCT EXITS  */
/*      AND PERFORMS NO LIBRARY ALLOCATION.                      */
/*      IF YOU ALLOCATED QMF LIBRARIES USING "LIBDEF", YOU MUST */
/*      USE THIS EXEC TO RE-ALLOCATE THE QMF LIBRARIES BECAUSE  */
/*      THEY WERE REPLACED BY DXT LIBRARY DEFINITIONS WHEN EXEC */
/*      "DSQABX2L" WAS EXECUTED.                               */
/*      IF YOUR QMF PRODUCT IS NOT THE ENGLISH VERSION, YOU MUST */
/*      SET THE LANGUAGE KEY TO THE PROPER VALUE. SEE VARIABLE  */
/*      "LKEY" IN THIS EXEC FOR CURRENT VALUE.                  */
/*      INPUT: NONE                                           */
/*      OUTPUT: NONE                                          */
/*      EXIT CONDITIONS: NONE                                */
/*      ABEND CODE: VALUE - NONE                             */
/*      EXTERNAL REFERENCES:                                */
/*      ROUTINES: NONE                                       */
/*      DATA AREAS: NONE                                    */
/*      CHANGE ACTIVITY: NONE                                */
/*      *********************************************************/
/*-END-OF-SPECIFICATION-***********************************/

/*      *********************************************************/
/*      SET QMF PRODUCT LANGUAGE CODE                         */
/*      *********************************************************/
LKEY = 'E'                                     /* SET ENGLISH LANGUAGE KEY */

```

Figure 39. EXEC to reallocate DXT CMS files (DSQABX2F) (Part 2 of 3)

Establishing QMF Support for End Users

```

/*****
/* RELEASE DXT PRODUCTION DISK
/*****
USER_ADDRESS = '291' /* SET ADDRESS OF DXT DISK */

ADDRESS CMS 'SET CMSTYPE HT'
ADDRESS CMS 'RELEASE' USER_ADDRESS '(DET)'
ADDRESS CMS 'SET CMSTYPE RT'

/*****
/* SET ISPF ERRORS TO RETURN TO THIS EXEC
/*****
ADDRESS ISPEXEC 'CONTROL ERRORS RETURN'
IF RC = 0 THEN EXIT RC

/*****
/* RE-ALLOCATE QMF LIBRARIES USING ISPF LIBDEF FUNCTION
/*****

LIBS = 'ADMRLIB TXTLIB * , ADMPLIB TXTLIB * , ADMGLIB TXTLIB * '
ADDRESS ISPEXEC 'LIBDEF ISPLIB FILE ID('LIBS')'
IF RC = 0 THEN EXIT RC

LIBS = 'DSQPLIB'LKEY' MACLIB * '
ADDRESS ISPEXEC 'LIBDEF ISPLIB FILE ID('LIBS')'
IF RC = 0 THEN EXIT RC

LIBS = 'DSQMLIB'LKEY' MACLIB * '
ADDRESS ISPEXEC 'LIBDEF ISPLIB FILE ID('LIBS')'
IF RC = 0 THEN EXIT RC

LIBS = 'DSQSLIBE MACLIB * '
ADDRESS ISPEXEC 'LIBDEF ISPLIB FILE ID('LIBS')'
IF RC = 0 THEN EXIT RC

EXIT 0

```

Figure 39. EXEC to reallocate DXT CMS files (DSQABX2F) (Part 3 of 3)

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, don't modify the EXEC. The unmodified EXEC won't interfere with your alternative method of allocation.

Customizing the Document Editing Interface for Users

General-Use Programming Interface

The document interface is an IBM-supplied macro for the ISPF/PDF and XEDIT editors. Using this macro, you 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, you can create the report at the time the macro is issued, in a QMF session that the macro starts.

End of General-Use Programming Interface

Before you use this macro, you can change it in various ways. Some of these changes are required, while others are optional. This section discusses the changes, both required and optional. To use the document interface, you should also see *Using QMF*.

If you're using an NLF: You also want to 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 located as follows:

- The EXECs and macros are on the QMF production disk.
- The other components are on the QMF distribution disk.

Renaming the Document Interface Macros and EXEC

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 effect on the other components.) IBM recommends the name GETQMF ISREDIT. This is the name used for the macro in this publication and in *Using QMF*. 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.

Establishing QMF Support for End Users

If you're 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
```

Changing the Data Components

There are five data components, all in the QMF distribution disk. 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. You can modify these components in either of the following ways:

- You can retain the changed components on the distribution disk.
If you do, change the names of the original components, and give the changed components the original names.
- You can place the changed components on a new minidisk.
If you do, you must ensure that in the search order the new minidisk is accessed before the old one.

The Message Component

One of the five data components is named DSQAED0L. This component contains:

- The messages that can appear on a user's screen while the user is operating the document interface
- Keywords for certain QMF commands

Do *not* change this component.

If you're using an NLF: The DSQAnD0L component is on the NLF distribution list 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 program. It is a text processing system that supports the use of computers in preparing documents for printing. If your installation uses this program, you might want to change the remaining four data components. These components, known as the DCF components, contain DCF control statements. For more on DCF, see *Document Composition Facility: General Information*

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 a 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
```

Establishing QMF Support for End Users

Changing the EXECs and Macros

As mentioned earlier, these components are all on the QMF production disk. If you change a component, change a copy, not the original, and place the copy on a minidisk that is accessed before the production disk.

If the document interface is issued from a current ISPF session then that session needs to have built the QMF and ISPF definitions for the ISPF libraries (the ones beginning with ISP). This is illustrated in DSQABD2I and *Installing and Managing QMF on VM/ESA* .

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're 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're 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 is the ISPF/PDF editor (provided QMF is started as an ISPF dialog). Other editors can also be used if supported at your site.

The following procedure assumes that you use an editor that can be invoked 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. This file name, which is the only parameter passed to the EXEC, is passed as a positional parameter.

QMF calls the EXEC, XYZEDIT, with the following command (USERA FILE A1 is the file name, file type, and file mode of the file to be edited):

```
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 look like this:

```
EDIT yyyyy (EDITOR=xxxxxxx)
```

where yyyyy is either PROC or QUERY. (Only the current procedure or query can be edited.) xxxxxxxx is the name of the EXEC created to invoke the editor. For more on the EDIT command, see *QMF Reference* .

The file you use can also be used for the ISPF/PDF editor. It's possible it might also be used for another editor that you want to support for the EDIT command.

Establishing QMF Support for End Users

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 FILEDEF DSQEDIT. (Remember that the edit transfer file described by the DSQEDIT filedef cannot be allocated to a disk that is used in the CMS shared file system (SFS).) To bring the object into QMF, the user needs to issue a RESET DATA command. This information, including the file name of the saved object, is provided in the message help for the error message associated with this condition.

Enabling English Support in an NLF Environment

Every NLF has a complete set of translated verbs, keywords, messages, and panels for QMF. The global variable DSQEC_NLFCMD_LANG allows you to change the language in which the user enters commands.

Set DSQEC_NLFCMD_LANG to 1 to allow users to enter commands only in English.

The default value, 0, allows users to enter commands and keywords only in the national language of the current session, except for the following commands:

```
SET
GET
INTERACT
MESSAGE
START
```

QMF allows you to enter these commands in either English or the NLF, regardless of how you set DSQEC_NLFCMD_LANG.

Use the DSQEC_FORM_LANG variable to enable users working in an NLF environment to store their form objects in the English language. The LANGUAGE option on the SAVE, EXPORT, and IMPORT commands allows users to specify the national language of the saved form. The values for this option are ENGLISH and SESSION, and are controlled by the global variable DSQEC_FORM_LANG.

Set DSQEC_FORM_LANG to 0 to use the language of the current session as the national language of the saved form.

The default value is 1, which specifies English as the language of the saved form.

If the user specifies the LANGUAGE keyword on the IMPORT or EXPORT command, that value overrides the current value of the DSQEC_FORM_LANG variable.

To change the national language displayed during a QMF session, the QMF user must end the current QMF session and begin another. You cannot change the language from within the QMF session.

Using Global Variables to Define the Currency Symbol

If you require a currency symbol that is not represented on the keyboard, you can specify the currency symbol by using the HEX value in a Procedure with Logic. For example, the following PROC will set the currency symbol to HEX '9F':

```
/* */  
"SET GLOBAL (DSQDC_CURRENCY =" '9F'X
```

If trailing blanks are needed for the currency symbol, you can put the currency symbol in single quotes as follows:

```
SET GLOBAL (DSQDC_CURRENCY = 'FR '
```

You can use the command in either the command line or in a linear PROC.

Chapter 9. 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's most 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. See "Setting Printing for Double-Byte Character Set Data (DSQSDBCS)" on page 90 for more information.

Quick start

Use Table 24 to guide you in printing QMF objects to a print or display device. If you need more information on any of the steps, see the page listed at the right of the table.

If you receive errors during printing, see "Troubleshooting Common Problems" on page 290 to help you solve the problem.

Table 24. Printing QMF objects

To do this task:	See:
Use the QMF PRINT command or a command synonym to print a QMF object. How QMF prints the object depends on what type of object you're trying to print.	Pages 148 and 156
Choose either QMF services or GDDM services to handle printing, or combine the two to suit your needs. GDDM can print to any device that supports the display of graphics. QMF prints using DSQPRINT.	Page 149
To print using GDDM services: Define a GDDM nickname for your printer and update the GDDM defaults module ADMADFV with the nickname.	Page 149
To print using QMF's DSQPRINT: Allocate DSQPRINT using a FILEDEF that points to the file or output class QMF uses for printing.	Page 155
Update the LENGTH and WIDTH values in the user's profile to specify a page size. To activate GDDM services for printing, provide a valid nickname for the PRINTER field in Q.PROFILES.	Page 156

Enabling Users to Print Objects

Printing Objects

The rules for printing QMF and database objects vary, depending on the type of object. Table 25 summarizes the requirements for each object.

Table 25. Summary of print requirements for QMF and database objects

Object type	Nickname required	GDDM gets control when...	Where output is routed
Chart	Yes	GDDM ICU always gets control when the PRINT command is issued.	Output is controlled by GDDM. For more information, see <i>GDDM Installation and System Management for VM</i> for GDDM 2.3 or <i>GDDM System Customization and Administration</i> for GDDM 3.1.
Form	Yes	GDDM always gets control when the PRINT command is issued.	Output is controlled by GDDM. For more information, see <i>GDDM Installation and System Management for VM</i> for GDDM 2.3 or <i>GDDM System Customization and Administration</i> for GDDM 3.1.
QBE query	No	Only if the nickname is supplied on the PRINT command or in profile.	Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT.
Procedure	No	Only if the nickname is supplied on the PRINT command or in profile.	Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT.
Profile	No	Only if the nickname is supplied on the PRINT command or in profile.	Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT.
Prompted query	Yes	GDDM always gets control when the PRINT command is issued.	Output is controlled by GDDM. For more information, see <i>GDDM Installation and System Management for VM</i> for GDDM 2.3 or <i>GDDM System Customization and Administration</i> for GDDM 3.1.
Report	No	Only if the nickname is supplied on PRINT command or in the profile.	Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT.
SQL query	No	Only if the nickname is supplied on the PRINT command or in the profile.	Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT.
Table	No	Only if the nickname is supplied on the PRINT command or in the profile.	Output goes to the device associated with the GDDM nickname or the ddname DSQPRINT.

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 don't 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 allocating a FILEDEF for DSQPRINT.

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.

Using GDDM Services to Handle Printing

Important: The explanations in this section apply only if you're using the GDDM default values shipped with the GDDM product. For more information on changing these values, see one of the following:

- *GDDM Installation and System Management for VM* (for GDDM 2.3)
- *GDDM System Customization and Administration* (for GDDM 3.1)

To use GDDM services for printing QMF objects, you need to:

1. Choose a GDDM nickname for the print device, as explained in “Choosing a GDDM Nickname for Your Printer” on page 150. 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 direct your printed output to a file or printer. Nicknames 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. This is explained in “Updating the GDDM Defaults Module with the Nickname” on page 154.
3. Update the PRINTER field of the user's row in the Q.PROFILES table, as explained in “Updating User Profiles to Enable GDDM Printing” on page 156.

Enabling Users to Print Objects

Choosing a GDDM Nickname for Your Printer

When a user enters a printer name on the PRINTER keyword of the QMF PRINT command, GDDM first searches the ADMDEFS PROFILE file and then the defaults module, ADMADRV, for a matching nickname that defines how and where to direct the output. GDDM uses nicknames to recognize all the devices with which it can communicate (including terminals).

When printing with GDDM, you don't need a matching FILEDEF for your printer nicknames. GDDM places the output from your PRINT command in a file called xxxxxxxx ADMLIST or xxxxxxxx ADMPRINT, where xxxxxxxx is the printer nickname you used.

You can enter a nonexistent printer nickname, and GDDM simply places the output in a zzzz ADMPRINT file on your A-disk, where zzzz is the nonexistent printer nickname you used. However, the formatting of the output in that situation is unpredictable, so *do not* use nonexistent nicknames.

GDDM printing determines whether an ADMLIST or ADMPRINT file is created, depending on the device token specified in the nickname. System printer output is placed in ADMLIST; queued printer output is placed in ADMPRINT.

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 or GDDM-OS/2[®] Link. A Family 1 device can also include display devices, such as 3270 data-stream terminals. A printer directly attached to a user ID can be accessed as a Family 1 printer from that user ID.
- **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 ADMOPUV utility to print output. This utility is 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 ADMOPUV utility, see *GDDM System Customization and Administration* for GDDM 3.1 or *GDDM Installation and System Management for VM* for GDDM 2.3. These publications also provide more information on each type of GDDM device.

Creating the Nickname Specification

To create a nickname, 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.

Use the format shown in Figure 40 for your ADMMNICK specification.

```
ADMMNICK NAME=nickname,TOFAM=family_type,DEVTOK=device_token
```

Figure 40. Using the ADMMNICK specification to define a nickname

- Use NAME to indicate a 1-character to 8-character printer nickname to use with the QMF PRINT command. For example, if MYPRTR is the nickname, users can enter the command: PRINT REPORT (PRINTER=MYPRTR. NAME can be a single name, a list of names separated by commas, or a name with a leading or trailing ? character used as a wild card to send output to multiple printers that have similar names.
- Use TOFAM to indicate the type of device you're using. GDDM recognizes four families of devices, and handles each differently.
- Use DEVTOK to indicate a valid GDDM device token, which uniquely identifies a device and its print configuration (for example, a 3820 printer that prints 60 rows by 85 columns, 6 lines per inch). For a list of valid device tokens, see:

GDDM System Customization and Administration for GDDM 3.1

GDDM Installation and System Management for VM for GDDM 2.3

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=ADMKSYP
```

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 in Figure 41. 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 41. Using the ADMMNICK specification to define a nickname for a Family 2 printer

Enabling Users to Print Objects

After you create your nickname, 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.

For more information about ADMOPUV, see *GDDM System Customization and Administration*

Example Nickname for a Family 3 GDDM Printer

To define the nickname 370PRINT for a Family 3 GDDM printer, you might use an ADMMNICK specification similar to the one in Figure 42.

```
ADMMNICK NAME=370PRINT,TOFAM=3,DEVTOK=R87S
```

Figure 42. Using the ADMMNICK specification to define a nickname for a Family 3 printer

After you create your nickname, 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 in Figure 43.

```
ADMMNICK NAME=3900PRNT,TOFAM=4,DEVTOK=R87S
```

Figure 43. Using the ADMMNICK specification to define a nickname for a Family 4 printer

After you create your nickname, a file with type ADMIMAGE is created. You can spool the file to PSF/VM automatically if you have the CPSPPOOL processing option set. For more information about Family 4 printing, see *GDDM System Customization and Administration*

Defining Multiple Nicknames with One Definition

You can use a single nickname to define multiple printer addresses by including the wild card ? 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 direct the output from the PRINT command to the printer named MYPRINT2.

Examples of Nickname Definitions

This section shows examples of nicknames you might use for Family 1, 2, or 3 devices. For an example of defining nicknames for Family 4 devices, see “Example Nickname for a Family 4 GDDM Printer” on page 152 or the following manuals:

GDDM System Customization and Administration for GDDM 3.1

GDDM Installation and System Management for VM for GDDM 2.3

- **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=((PRINTCTL,0)),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 file has RECFM=F, GDDM printing changes the DCB of the file from RECFM=F to RECFM=V.

For a list of print control options and how to use them, see *GDDM System Customization and Administration for GDDM 3.1* or *GDDM Installation and System Management for VM for GDDM 2.3*.

- **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 PC printer connected to DOS, GDDM-PCLK must be installed on your workstation.

Enabling Users to Print Objects

- **A PC printer using GDDM-OS/2 Link (for OS/2 users):** Use the following definition to define the nickname GDDMOS2P for a Family 1 printer:
GDDMPRT7 ADMMNICK TOFAM=1,FAM=0,NAME=PMPRINT,TONAME=(*,ADMPMOP)

where * indicates the user's current device or the default value.

To print to a PC printer connected to OS/2, ensure GDDM-OS/2 Link is 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 defaults modules, see:

- *GDDM System Customization and Administration for GDDM 3.1*
- *GDDM Installation and System Management for VM for GDDM 2.3*

Testing the Nickname Definitions in External Default Files

Test your nickname definitions by placing them in an *external default file* named ADMDEFS PROFILE and printing with them until you are satisfied they are working correctly. Then you can assemble them into the *external default module* named 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 files to find a given nickname.

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.

Using QMF's DSQPRINT to Handle Printing

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 don't 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 using a FILEDEF that points to the file or output class QMF uses for printing. 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 route your output to a printer, use this syntax:

```
"FILEDEF DSQPRINT PRINTER (LRECL 121 BLKSIZE 121 RECFM VBA PERM"
```

If you're 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 have specified without having to print the report. This is applicable only when using DSQPRINT. For more information on DPRE, see "Displaying Printed Reports (DPRE)" on page 160 and *QMF Reference*.

Enabling Users to Print Objects

Defining a Synonym for the Print Function Key

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 DSQCIX "PRINT PROC (PRINTER=MYPRINT1"  
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.)

Updating User Profiles to Enable GDDM Printing

When a user enters a QMF PRINT command, QMF references the LENGTH, WIDTH, and PRINTER fields of the user's row in the Q.PROFILES table. Use these fields of the profile to specify the size and destination for the user's output.

To activate GDDM services for printing, specify a default GDDM printer nickname in the PRINTER column of the profile. Ensure the values you supply for LENGTH and WIDTH are the same as the width and length

specified by the device token in the ADMMNICK specification. Also ensure the printer name you use matches one of the entries in the ADMADFV defaults module. If a nickname isn't found, the file nickname ADMPRINT is created.

If you don't specify a printer name in the profile and the user tries to print a chart, form, or prompted query without specifying a printer name, QMF displays the message Please supply a nickname for your printer. Pressing Enter displays a prompt for a printer name. Instruct users to enter a printer name that matches one of the entries in the nickname file.

If the PRINTER field in the user's profile does not contain a GDDM nickname, QMF services are used for printing. You can specify defaults for LENGTH and WIDTH even if PRINTER is blank.

If you specified a default GDDM printer name in your profile but you want to use QMF services for printing, supply a blank value for the PRINTER keyword to override the GDDM printer nickname in the user's profile:

```
PRINT REPORT (PRINTER=' ')
```

Enabling Users to Print Objects

Chapter 10. Customizing QMF Commands

QMF command synonyms help you customize the base set of QMF commands by allowing you to define your own terms and link them to QMF or CMS commands. A synonym might simply be another word for a QMF or CMS command, or it can 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 normally enter a QMF command.

Quick Start

Follow the steps in Table 26 to create a command synonym. If you need more information on any step, see the page listed at the right.

Table 26. Creating synonyms for QMF commands

To do this task:	See:
Use the default synonyms provided with QMF to display a printed report, run a batch query or procedure, customize a report layout, or leave QMF in interactive mode and bridge to ISPF.	Page 159
Create a command synonym table that has the columns VERB, OBJECT, and SYNONYM_DEFINITION. The table links the synonyms you choose with the commands or procedures they represent.	Page 162
Enter synonyms and their definitions into the table. VERB and OBJECT store your synonym; SYNONYM_DEFINITION is the command or procedure that runs when you enter the synonym. Follow the guidelines for valid verbs, object names, and synonym definitions.	Page 163
Activate the synonyms for users. Update the SYNONYMS field of the user's row in Q.PROFILES with the name of the synonym table. Then instruct users to reconnect to the database to initialize the new synonyms.	Page 168
Minimize maintenance of your site's command synonym tables by creating a single synonym table for all users or by creating different types of views on the synonym table.	Page 170

Using the Default Synonyms Provided with QMF

QMF provides four applications that can be used as installation-defined commands. After installation, the application synonyms appear in the Q.COMMAND_SYNONYMS table. Users with access to this table can invoke these applications by entering the appropriate synonym as if it were a QMF command.

Customizing QMF Commands

Display Printed Report

Synonym is DPRE. Displays the user's current report just as it would be printed. For information on customizing DPRE, see "Displaying Printed Reports (DPRE)".

Batch Query/Procedure

Synonym is BATCH. Lets the user run a query or procedure in batch mode rather than running it interactively. For more information on this application, see Using the QMF Batch Query/Procedure Application (BATCH).

Layout Form

Synonym is LAYOUT. Lets the user tailor reports without having to run a query. For an example of how to use this application, see *Using QMF*. For information on the command's syntax, see *QMF Reference*.

Bridge to ISPF

Synonym is ISPF. Lets the user temporarily leave QMF in interactive-mode and "bridge" to ISPF/PDF. The user then conducts an ISPF/PDF session independently of QMF. After the session is ended, the user is returned to QMF, at the point where the ISPF command was issued. For more on the ISPF application, see *Using QMF* and *Developing QMF Applications*.

ISPF considerations:

1. For the first three applications, QMF must be started under ISPF.
2. The synonym ISPF is valid only if QMF is started as an ISPF dialog. If QMF is not started as an ISPF dialog and the user wants to use ISPF, the user can issue the command CMS ISPSTART.

Displaying Printed Reports (DPRE)

You might notice that a printed report doesn't look exactly as it did on a screen. For example, the displayed report is treated as a single page, even with one or more page breaks in the printed report.

The differences between the printed report and its displayed version are largely cosmetic: the facts and figures on the screen and those on the printed page are the same. However, the differences can be important. (For more detailed information about the differences, see *Using QMF*.) Because of this, IBM supplies the QMF application called DPRE to display the report as it would 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.

Using DPRE

To use DPRE, you 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 finish browsing, the printer output disappears.

If you're 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 *QMF Reference*.

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 now alternate changing the FORM panel and browsing the tentative report with DPRE. Users should be aware, however, that the second method of development is expensive relative to the first, and should be used sparingly when resources are at a premium.

When printing large tables, 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 holding the material to be browsed. If an error stops the execution, 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.

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
```

Customizing QMF Commands

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.

Use the following procedure to create a command synonym table. Then see "Entering Command Synonym Definitions into a Command Synonym Table" on page 163 for instructions on entering your synonyms and their definitions.

1. If necessary, acquire or add a dbspace to hold the command synonyms table. Figure 30 on page 118 shows how to acquire a dbspace. If you need to add a dbspace, see *DB2 Server for VM System Administration*
2. From the QMF SQL query panel, run an SQL CREATE TABLE statement similar to the one in Figure 44 to create the table. Substitute your own table name in place of *COMMAND_SYNONYMS* and your own dbspace name for *DBSPACE1*. 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)
IN DBSPACE1
```

Figure 44. 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 SYSTEM.SYSCATALOG table that describe the table's purpose. The following is an example for the COMMAND_SYNONYMS table created with the query in Figure 44.
COMMENT ON TABLE COMMAND_SYNONYMS IS 'SYNONYMS FOR RESEARCH DEPT'

The phrase 'SYNONYMS FOR RESEARCH DEPT' appears in the REMARKS column of the SYSTEM.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 a Command Synonym Table

After you create a command synonym table, use an SQL INSERT statement similar to the one in Figure 45 to enter your synonyms into the table. You can also use the Table Editor to update the table, as explained in *Using QMF*.

After it is activated according to the procedure described in “Activating the INSERT INTO COMMAND_SYNONYMS (VERB,OBJECT,SYNONYM_DEFINITION) VALUES('COMPUTE', 'MONTHLY_SALES', 'RUN PROC JONES.SALES_FIGURES')

Figure 45. Creating a command synonym definition

Synonyms” on page 168, the synonym COMPUTE MONTHLY_SALES runs a QMF linear procedure called SALES_FIGURES, owned by user JONES.

The query in Figure 46 shows an example of a synonym that has no entry in the object column:

After it is activated, the synonym EXECUTE runs the query currently in the INSERT INTO COMMAND_SYNONYMS (VERB,SYNONYM_DEFINITION) VALUES('EXECUTE', 'RUN QUERY')

Figure 46. Creating a command synonym definition

QMF temporary storage area.

The synonyms in Figures 45 and 46 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.

Customizing QMF Commands

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.

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.

Some examples that demonstrate these rules are shown in the following list. 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 *QMF Reference* to find out if the command requires a parameter.

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 can't 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 rules for naming DB2 for VM 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.)

Some examples of valid and invalid objects are shown in the following list.

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)

Choosing the Synonym Definition

The synonym definition is the QMF command or procedure that runs when the user enters the command synonym. 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 starts 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 *Developing QMF Applications*

Using a Procedure in the Synonym Definition

Your synonym definition can include a procedure that does the work of several QMF commands. For example, the linear procedure in Figure 47 on page 166 performs the following tasks:

Customizing QMF Commands

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

```
-- 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 47. Sample procedure to run using a command synonym

the one in Figure 48.

VERB	OBJECT	SYNONYM DEFINITION
SHOW	SALES	RUN PROC SALES_PROC

Figure 48. Using a command synonym to run a procedure

If you're 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're 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 144.

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 49 on page 167 shows a definition that passes the value Q.STAFF for the table name, which is evaluated when MYQUERY runs.

VERB	OBJECT	SYNONYM DEFINITION
EXECUTE	-	RUN QUERY MYQUERY (&&TABLENAME=Q.STAFF

Figure 49. Using variables in command synonym definitions

MYQUERY might look something like:

```
SELECT * FROM &TABLENAME
```

Amperands 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 variable &ALL to show where the information is located within the synonym definition.

The synonym definition in Figure 50 shows an example of a synonym defined using &ALL.

The query named STAFFQUERY might look something like the following:

VERB	OBJECT	SYNONYM DEFINITION
SHOW_INFO	-	RUN QUERY STAFFQUERY (&ALL)

Figure 50. Using the variable &ALL in a command synonym definition

```
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 *QMF Reference*.

Customizing QMF Commands

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 50 on page 167, 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 guidelines 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 can't 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 *QMF Reference*.

Activating the Synonyms

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 51 :

```
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 51. Activating a user's QMF command synonyms

Important: Always specify a value for TRANSLATION when you're updating Q.PROFILES, or you might change more rows than you intend.

The query in Figure 51 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 similar to the one shown in Figure 18 on page 99.

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 on 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 170.

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.

See Table 13 on page 95 for how to grant a user authority to connect to the database. Users who do not have DB2 for VM CONNECT authority can end the current QMF session and start another to activate the synonyms.

Customizing QMF Commands

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 *QMF Reference* for the proper abbreviations for QMF commands.

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 *DB2 Server for VM Database Administration*

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 52 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='CMS'
```

Figure 52. 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 53 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 53. 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 doesn't have one.
2. Use a query similar to the one in Figure 54 to create the view on the synonym table. All users in the `DEPTD02` group can use all public synonyms in the table and any synonyms assigned specifically to the group.

```
CREATE VIEW GROUVIEW (VERB,OBJECT,SYNONYM_DEFINITION)
AS SELECT VERB, OBJECT, SYNONYM_DEFINITION
FROM COMMAND_SYNONYMS
WHERE AUTHID='DEPTD02' OR AUTHID IS NULL
```

Figure 54. 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 55 on page 172 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 55. Creating a view that uses an extra table to control use of synonyms

Chapter 11. 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. Because every site's needs are unique, however, 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.

Quick Start

Follow the steps in Table 27 to customize a default QMF function key. If you need more information on any step, see the page listed at the right of the table.

Table 27. Customizing QMF function keys

To do this task:	See:
Choose the panels and function keys you want to customize. You can change function key settings on all panels except table editor panels and database status panels. Your flexibility in customizing the keys depends on what type of panel you choose.	Page 173
Create a table to hold the customized definitions of your function keys. Include at least four columns: PANEL, ENTRY_TYPE, NUMBER, and PF_SETTING. These columns have information about the command QMF issues when the key is pressed and the label text that is displayed beside the key number on the screen.	Page 176
Insert your customized key definitions into the function key table. To insert a definition, reference the panel ID of the panel you're customizing; the QMF command issued when the key is pressed; the text displayed on the screen next to the number of the key; and where the key is positioned on the screen.	Page 177
Activate the new function key definitions by updating the PFKEYS field of the user's row in Q.PROFILES with the name of the function key table you created.	Page 184

Choosing the Keys 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" on page 174 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.

Customizing QMF Function Keys

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
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

Key	Executed command
Run	RUN QUERY or RUN PROC
Save	SAVE PROFILE
Show	SHOW
Show Field	SHOW FIELD
Show SQL	SHOW SQL
Sort	SORT
Specify	SPECIFY
Specify View	SPECIFY VIEW

Default Keys on Window Panels

Key	Executed command
Attribute	SPECIFY ATTRIBUTES
Backward	BACKWARD
Cancel	CANCEL
Clear	CLEAR
Command	SHOW COMMAND
Comments	SWITCH COMMENTS
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

Customizing QMF Function Keys

Key	Executed command
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 key is pressed.

For more information on the commands associated with these function keys, see *QMF Reference*.

Creating the Function Key Table

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 56 to create the table. Substitute your own name for MY_PFKKEYS and your own dbspace for DBSPACE1.

```
CREATE TABLE MY_PFKKEYS
(PANEL          CHAR(18)      NOT NULL,
 ENTRY_TYPE     CHAR(1)      NOT NULL,
 NUMBER         SMALLINT     NOT NULL,
 PF_SETTING     VARCHAR(254))
IN DBSPACE1
```

Figure 56. Creating a function key table

For information on acquiring a dbspace to hold the table, see “Choosing and Acquiring a dbspace for the User” on page 118. For information on creating a new dbspace, see *DB2 Server for VM Database Administration*

2. Add comments to the SYSTEM.SYSCATALOG table using an SQL statement similar to the following:
COMMENT ON TABLE MY_PFKKEYS 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. For more information on adding comments to the system catalog, see *DB2 Server for VM Database Administration*

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 28 on page 178.

Customizing QMF Function Keys

Table 28. Values to customize your function key table

Column	Value	Information
PANEL	ID of the QMF panel you're customizing	<p>"Full-screen Panel Identifiers" on page 181 shows the IDs you need to use for full-screen panels. "Window Panel Identifiers" on page 181 shows the IDs you need to use for specific window panels.</p> <p>If you want to define the same set of keys to appear on every panel in a class of window panels, use the <i>class ID</i> 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.</p> <p>Changes you make using a class ID apply to <i>all</i> panels customized by that class ID. Help and prompt windows don't have a set of unique IDs; they can be customized using only class IDs.</p>
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	For example, if you're changing the definition for PF5, enter a 5 in this column.
PF_SETTING	Text of the command that runs when the key is pressed	<p>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.</p> <p>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.</p> <p>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.</p>

If you're using an NLF: Ensure the underlying command has the correct national language translation; additionally, it's helpful if the label text for each key is written in the language of the NLF you're 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 values shown in Table 29.

Table 29. 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 177.
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 177). For example, on the Home panel, PF5 appears in row 1 and PF12 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're 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 57 on page 180 for an example. For window panels, you need to type only the label of the key in this column. See Figure 58 on page 180 and Figure 59 on page 181 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 You Want to Customize" on page 181. Use these tables to get the proper values for the PANEL column of the function key table.

Entering a Definition for a Key on a Full-screen Panel

Use the SQL queries shown in Figure 57 on page 180 to change PF2 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 PF2.

Customizing QMF Function Keys

```
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 57. Changing a function key for a QMF command on the Home panel

The QMF Home panel now displays Import for PF2:

```
Type command on command line or use function 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 haven't 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 function 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 58 add a PF3 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 58. 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 59 add a PF13 key to all help panels. The function key executes the CANCEL command, and is labeled CancelMe.

All help and prompt panels are customized using a single class ID. Because

```
INSERT INTO MY_PFKEYS (PANEL,ENTRY_TYPE,NUMBER,PF_SETTING)
VALUES('HEXXX', 'K', 13, 'CANCEL')

INSERT INTO MY_PFKEYS (PANEL,ENTRY_TYPE,NUMBER,PF_SETTING)
VALUES('HEXXX', 'L', 13, 'CancelMe')
```

Figure 59. Changing a function key on a help panel or prompt panel

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 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 are listed in Figure 60. Enter the identifiers in the PANEL column of the function key table exactly as they are shown here.

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 60. Full-screen panel identifiers

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

Customizing QMF Function Keys

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

Panel Identifier	Title or 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
QPCOFU	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)
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

Customizing QMF Function Keys

Panel Identifier	Title or Description
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

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 61 to assign to English QMF user JONES the table MY_PFKEYS, and to German NLF user SCHMIDT the table MEIN_PFKY. 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'
```

Figure 61. 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 170. 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
```

Each time you make a change to the table, instruct users to reconnect to the database to activate the changes you made.

See Table 13 on page 95 for how to grant a user authority to connect to the database. Users who do not have DB2 for VM CONNECT authority can end the current QMF session and start another to activate the new function keys.

Customizing QMF Function Keys

Chapter 12. Creating Your Own Edit Codes for QMF Forms

Note: This chapter contains General Use Programming Interface and Associated Guidance Information.

QMF *forms* help users 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 by 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. *QMF Reference* shows the edit codes supplied with QMF.

This chapter also shows you how to write an edit exit routine in either 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 some supported languages do not support 31-bit addressing.

Before you begin the tasks in this chapter, consider reviewing the sections of *QMF Reference* that describe QMF's functions for report formatting and edit codes.

Quick Start

Use the steps in Table 30 to guide you in creating a user edit exit routine. If you need more information on any step, see the page listed at the right of the table.

Table 30. Creating a user edit exit routine

To do this task:	See:
Decide what you want your routine to do and choose an edit code that identifies the routine. Use either Uxxxx or Vxxxx for your edit code, where xxxx is zero to four letters with no embedded blanks or null values.	Page 188

Creating Your Own Edit Codes for QMF Forms

Table 30. Creating a user edit exit routine (continued)

To do this task:	See:
Request that your exit routine format the data by using fields of the IBM-supplied interface control block.	Page 191
Accept parameters from and return formatted results to the exit routine using the standard input and output fields provided in the interface control block.	Page 193
Request that control pass to your edit exit routine when QMF terminates by setting a termination switch in a field of the interface control block. You might pass control to the edit exit routine if the routine needs to perform cleanup activities, such as releasing storage.	Page 197
To write your edit exit routine in assembler, start with the sample assembler program provided by IBM. After you write your program, assemble and generate the program.	Page 197
To write your edit exit routine in PL/I, start with the sample PL/I program provided by IBM. After you write your program, compile and generate the program and define it to CMS.	Page 203
To write your edit exit routine in COBOL, start with the sample COBOL program provided by IBM. After you write your program, compile and generate the program and define it to CMS. COBOL refers to VS COBOL II, COBOL/370, and IBM COBOL for OS/390 and VM unless otherwise stated.	Page 213

Choosing an Edit Code

Create either a Uxxxx or a 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 in 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 doesn't handle extended floating point data. In this case, IBM recommends using 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 31.

Table 31. 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 need not 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 Data Types

If your installation supports date/time data types, you can format columns with data types of DATE, TIME, and TIMESTAMP. This enables your users to use local date/time exit routines. For more information about these data types, see *Using QMF*.

You need to remember that these are DB2 for VM exits, *not* QMF exits. For details about how these exits are created refer to *DB2 Server for VM System Administration*

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 in program segment mode, you must create two relocatable module files 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
```

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

Creating Your Own Edit Codes for QMF Forms

value to be formatted, which appears in the control block field ECSINP, 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 32.

Table 32. Formatting DATE, TIME, and TIMESTAMP data

Data type	Form of the string
DATE data	<p>yyyy-mm-dd where:</p> <p>yyyy Specifies the year. It is always a four-digit number.</p> <p>mm Specifies the month (01 for January, ... 12 for December). It is always a two-digit number that can contain a leading zero.</p> <p>dd Specifies the day of the month. It is always a two-digit number that can contain a leading zero.</p> <p>The dashes (-) represent true dashes.</p> <p>For example, 1990-12-12 is the date December 12, 1990.</p>
TIME data	<p>hh.mm.ss where:</p> <p>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.</p> <p>mm Specifies the minute. It is always a two-digit number that can contain a leading zero.</p> <p>ss Specifies the second. It is always a two-digit number that can contain a leading zero.</p> <p>The periods represent true periods.</p> <p>For example, 13.08.36 is 1:08 P.M. and 36 seconds in the notation commonly used in the United States.</p>
TIMESTAMP data	<p>yyyy-mm-dd-hh.mm.ss.nnnnnn where:</p> <p>yyyy-mm-dd Specifies the date in the same way it does for DATE data.</p> <p>hh.mm.ss Specifies the time of day in the same way it does for TIME data.</p> <p>nnnnnn Specifies a six-digit number that extends the count of seconds (ss) down to the nearest microsecond.</p> <p>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.</p>

For the data types available, see the ECSINTYP field in Table 33 on page 193.

Calling Your Exit Routine to Format the Data

Figure 62 shows how QMF and your edit exit routine work together to format data using the edit codes you define.

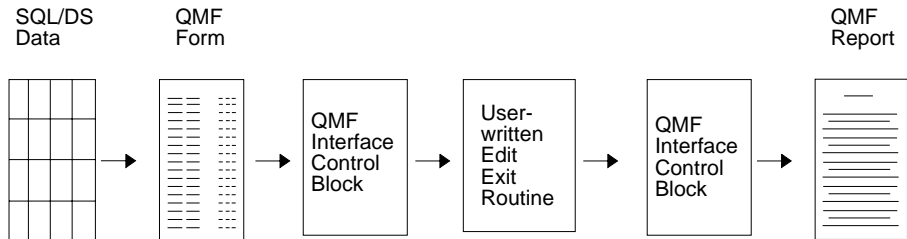


Figure 62. How a user edit routine works with QMF

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 shown in Figure 62. These characteristics reside in specific fields of the control block, which are discussed in “Fields of the Interface Control Block” on page 193. 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 (named DSQUXDTA), one is for PL/I (named DSQUXDTP), and the other is for COBOL (named 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.

QMF supplies a user edit routine in the form of a relocatable module file and a text file (both called DSQUEDIT), which are located on the QMF production disk. Delete or rename the QMF-supplied module and text file DSQUEDIT when you are ready to use your edit routine.

If the programming language you are using supports creation of a relocatable module file, create a module file for the edit routine.

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.

Creating Your Own Edit Codes for QMF Forms

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 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 assembler, 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 63 shows this general structure.

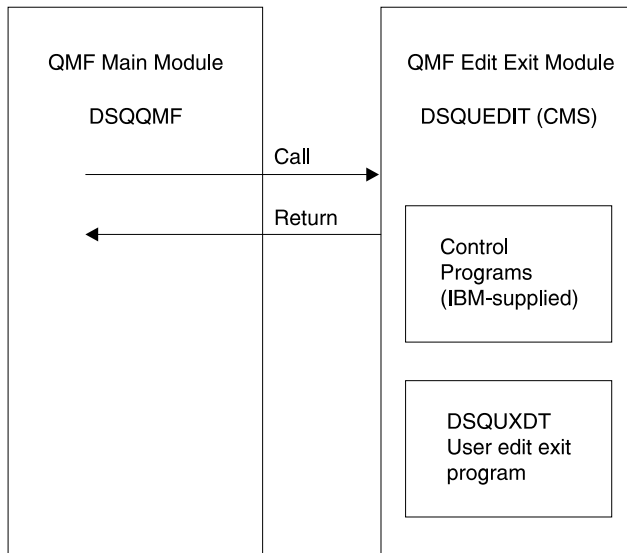


Figure 63. General program structure for edit routines

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 results might be unpredictable.

ECSINPT, ECSRSLT, and ECSRSLEN are fields of the interface control block, explained in Table 33.

Fields of the Interface Control Block

Use the fields of the interface control block to pass information to and from your exit routine. Although there are separate interface control blocks that work with assembler, PL/I, or COBOL, the fields of the interface control block are standard regardless of the programming language your edit exit routine is written in. These fields are shown in Table 33. 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 33. 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.

Creating Your Own Edit Codes for QMF Forms

Table 33. Fields of the QMF interface control block (continued)

Name	Contents												
ECSERRET	<p>Contains a zero at the point of call. Set this to a nonzero return code to record an error. Use one of the values in the following list for an error of the indicated type:</p> <p>Number</p> <table><thead><tr><th></th><th>Error</th></tr></thead><tbody><tr><td>99101</td><td>Unrecognized edit code</td></tr><tr><td>99102</td><td>Improper input data type for edit code</td></tr><tr><td>99103</td><td>Invalid input value for item to be formatted</td></tr><tr><td>99104</td><td>Item to be formatted is too short</td></tr><tr><td>99105</td><td>Not enough room for result in ECSRSLT (result is too wide for the space allotted)</td></tr></tbody></table> <p>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.</p>		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)
	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)												
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 33. Fields of the QMF interface control block (continued)

Name	Contents
ECSINTYP	<p>Indicates, in database terms, how the value to be formatted is represented. Applies to edit codes of every type. Values can be:</p> <p>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</p> <p>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.</p>
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.
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 Data Types” on page 189.

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.

Creating Your Own Edit Codes for QMF Forms

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 two factors:

- Whether the value to be formatted is numeric or character, as determined by the ECSINTYP field.
- Whether the edit code is a U code or a 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 in the decimal format are in the ECSINSCL and ECSINPRC fields. Length (in bytes) is in ECSINLEN.

Numeric data from defined columns, calculations, and summary values is returned as extended floating point values, a data type not explicitly supported by DB2 for VM. 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 ECSRSLLEN. QMF blanks out ECSRSLT before calling the edit routine.

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 High-Level Assembler (HLASM) or Assembler

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 64 shows the program structure of an assembler edit exit routine for CMS.

The IBM-supplied sample edit program for assembler, DSQUXDTA, is located

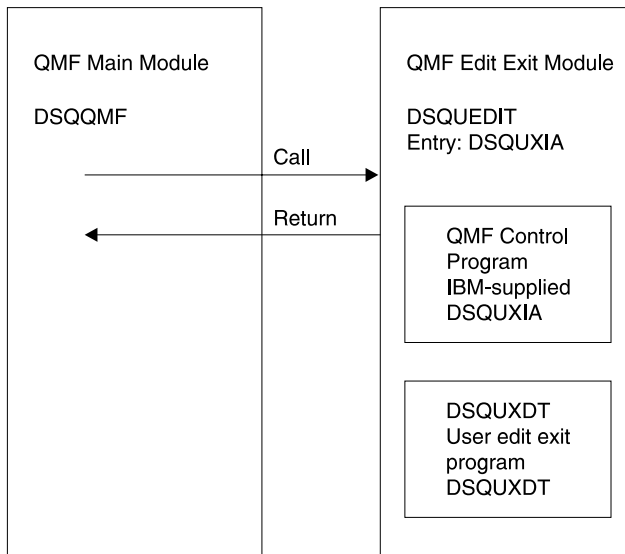


Figure 64. Program structure of an assembler edit exit routine

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. It is DXEECSA that defines the input fields, giving them the names we are using in this chapter.

Creating Your Own Edit Codes for QMF Forms

How an Assembler Edit Routine Interacts with CMS

Linkage obeys the standard IBM calling conventions:

- The address of a parameter list is passed in register 1, as shown in Figure 65.

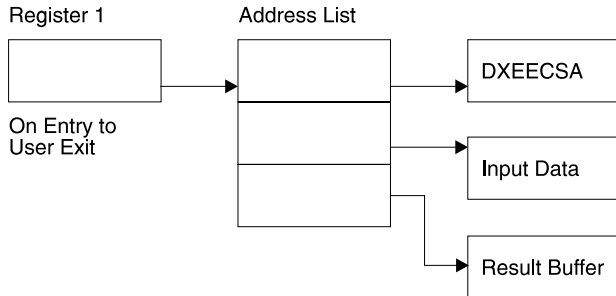


Figure 65. Registers of the program interface in assembler

- The parameter list contains three 4-byte addresses. The addresses point to:
 - The control block
 - The value to be formatted
 - The storage reserved for the formatted results
- Registers 13 and 14 point to the caller's save area and the return point.

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 (see Figure 66 on page 200).

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 edit routine.

Figure 66 on page 200 shows the DXEECS control block for assembler.

Creating Your Own Edit Codes for QMF Forms

```

***** 00001000
*
* CONTROL BLOCK NAME: DXEECS (ASSEMBLER VERSION) * 00002000
*
* FUNCTION: * 00003000
*
* THIS IS THE INTERFACE CONTROL BLOCK BETWEEN QMF AND * 00004000
* THE USER EDITING INTERFACE, DSQUEDIT (TSO/CMS), OR * 00005000
* DSQUEJIC (CICS). * 00006000
*
* IT CONTAINS THE USER'S EDIT CODE, IDENTIFIES THE SOURCE * 00007000
* DATA AND THE TARGET LOCATION FOR THE EDITED RESULT * 00008000
* AND PROVIDES A SCRATCHPAD AREA FOR THE USER EDIT * 00009000
* ROUTINE'S USE. * 00010000
*
* THIS CONTROL BLOCK IS PERSISTENT BETWEEN CALLS TO THE * 00011000
* USER EDIT ROUTINE. * 00012000
*
* THE SCRATCHPAD AREA WILL NOT BE MODIFIED BY QMF AFTER * 00013000
* THE INITIAL INVOCATION OF THE EXIT ROUTINE. * 00014000
*
* STATUS: VERSION 7 RELEASE 1 LEVEL 0 * 00015000
*
* INNER CONTROL BLOCKS: NONE * 00016000
*
* CHANGE ACTIVITY: * 00017000
*
* CHANGE DATE: * 00018000
*
***** 00019000
*
* DXEECS DSECT * 00020000
ECSNAME DS CL8 -- CONTROL BLOCK IDENTIFICATION * 00021000
SPACE * 00022000
ECSSECTL DS XL40 -- EDIT CONTROL * 00023000
ORG ECSECTL * 00024000
ECSREQ DS CL1 ----- FUNCTION REQUEST * 00025000
ECSFREQ EQU C'E' ----- EDIT FUNCTION * 00026000
ECSFREQ EQU C'T' ----- TERMINATE FUNCTION * 00027000
* (TO FREE RESOURCES... QMF * 00028000
* WILL CALL THE USER EDIT * 00029000
* ROUTINE FOR THIS FUNCTION * 00030000
* ONLY IF THE USER EDIT ROUTINE * 00031000
* HAS PREVIOUSLY REQUESTED IT. * 00032000
* SEE ECSRQMF BELOW.) * 00033000
ECSPAD10 DS CL3 ----- RESERVED FIELD * 00034000
ECSECODE DS CL5 ----- EDIT CODE FROM FORM OBJECT * 00035000

```

Figure 66. User edit routine field definitions for assembler DXEECS control block (Part 1 of 3)

Creating Your Own Edit Codes for QMF Forms

ECSPAD20	DS	CL3	----- RESERVED FIELD	00049000
ECSDEPT	DS	CL1	----- SYMBOL FOR DECIMAL POINT	00050000
*			(AS DEFINED BY DECIMAL OPTION IN	00051000
*			CURRENT PROFILE OBJECT	00052000
ECSTHSEP	DS	CL1	----- SYMBOL FOR THOUSANDS SEPARATOR	00053000
*			(AS DEFINED BY DECIMAL OPTION IN	00054000
*			CURRENT PROFILE OBJECT	00055000
ECSPAD30	DS	CL6	----- RESERVED FIELD	00056000
ECSQMF	DS	CL20	----- AREA RESERVED FOR QMF'S USE	00057000
		SPACE		00058000
ECSINDTA	DS	XL16	-- DESCRIPTION OF THE INPUT DATA	00059000
	ORG	ECSINDTA		00060000
ECSINTYP	DS	F	----- DATA TYPE OF THE INPUT AS IT	00061000
*			EXISTS IN THE DATA BASE.	00062000
ECSFLT	EQU	480	----- FLOATING POINT DATA TYPE CODE	00063000
ECSDEC	EQU	484	----- DECIMAL DATA TYPE CODE	00064000
ECSINT	EQU	496	----- INTEGER DATA TYPE CODE	00065000
ECSSINT	EQU	500	----- SMALL INTEGER DATA TYPE CODE	00066000
ECSVCHR	EQU	448	----- VARCHAR DATA TYPE CODE	00067000
ECSFCHR	EQU	452	----- (FIXED) CHARACTER DATA TYPE CODE	00068000
ECSLCHR	EQU	456	----- LONG VARCHAR DATA TYPE CODE	00069000
ECSVG	EQU	464	----- VARGRAPHIC DATA TYPE CODE	00070000
ECSFG	EQU	468	----- (FIXED) GRAPHIC DATA TYPE CODE	00071000
ECSLG	EQU	472	----- LONG VARGRAPHIC DATA TYPE CODE	00072000
ECSDATE	EQU	384	----- DATE DATA TYPE CODE	00073000
ECSTIME	EQU	388	----- TIME DATA TYPE CODE	00074000
ECSTS	EQU	392	----- TIMESTAMP DATA TYPE CODE	00075000
ECSFLT	EQU	940	----- EXTENDED FLOATING PT CODE	00076000
*				00077000
ECSINLEN	DS	F	----- LENGTH OF INPUT DATA	00078000
ECSINPRC	DS	H	----- PRECISION OF INPUT DATA IF IT IS	00079000
*			DECIMAL DATA TYPE (U-TYPE EDIT CODE)	00080000
*			OR IF IT WAS ANY NUMERIC DATA TYPE	00081000
*			(V-TYPE EDIT CODE)...	00082000
*			ZERO OTHERWISE	00083000
ECSINSCL	DS	H	----- SCALE OF INPUT DATA IF IT IS	00084000
*			DECIMAL DATA TYPE (U-TYPE EDIT CODE)	00085000
*			OR IF IT WAS ANY NUMERIC DATA TYPE	00086000
*			(V-TYPE EDIT CODE)...	00087000
*			ZERO OTHERWISE	00088000
ECSINSGN	DS	CL1	----- SIGN OF CONVERTED NUMERIC DATA	00089000
*			(V-TYPE EDIT CODE ONLY)...	00090000
ECSPLUS	EQU	C' '	----- POSITIVE SIGN	00091000
ECSMINUS	EQU	C'-'	----- NEGATIVE SIGN	00092000
*				00093000
ECSINNUL	DS	CL1	----- NULL INPUT DATA INDICATOR	00094000
ECSNULL	EQU	C'N'	----- INPUT DATA IS NULL	00095000
*				00096000

Figure 66. User edit routine field definitions for assembler DXEECS control block (Part 2 of 3)

Creating Your Own Edit Codes for QMF Forms

ECSPAD40	DS	CL10	-----	RESERVED FIELD	00097000
		SPACE			00098000
ECSRSDTA	DS	XL16	--	DESCRIPTION OF THE RESULT BUFFER	00099000
	ORG	ECSRSDTA			00100000
ECSRLEN	DS	F	-----	LENGTH OF RESULT AREA	00101000
*				(EQUIVALENT TO COLUMN WIDTH IN THE	00102000
*				FORM OBJECT	00103000
ECSPAD50	DS	CL12	-----	RESERVED FIELD	00104000
		SPACE			00105000
ECSUCTL	DS	XL16	--	USER CONTROL AREA	00106000
	ORG	ECSUCTL			00107000
ECSERRET	DS	F	-----	EDIT ROUTINE ERROR RETURN CODE	00108000
ECSERR01	EQU	99101	-----	UNRECOGNIZED EDIT CODE	00109000
ECSERR02	EQU	99102	-----	IMPROPER INPUT DATA TYPE	00110000
ECSERR03	EQU	99103	-----	INVALID INPUT DATA VALUE	00111000
ECSERR04	EQU	99104	-----	INPUT DATA LENGTH IS TOO SHORT	00112000
ECSERR05	EQU	99105	-----	RESULT BUFF LENGTH IS TOO SHORT	00113000
*					00114000
ECSRQMF	DS	CL1	-----	REQUEST FOR QMF	00115000
ECSRTERM	EQU	C'T'	-----	REQUEST INVOCATION FOR	00116000
*				TERMINATION FUNCTION	00117000
*					00118000
ECSPAD60	DS	CL11	-----	RESERVED FIELD	00119000
		SPACE			00120000
ECSUSERS	DS	CL256	--	USER SCRATCH PAD AREA	00121000
		SPACE 2			00122000
ECSINPT	DSECT		--	EDIT ROUTINE INPUT DATA	00123000
ECSINPTC	DS	CL32767	-----	CHARACTER STRING	00124000
	ORG	ECSINPTC			00125000
ECSINSIN	DS	H	-----	SMALL INTEGER	00126000
	ORG	ECSINPTC			00127000
ECSININT	DS	F	-----	INTEGER	00128000
	ORG	ECSINPTC			00129000
ECSINFLT	DS	D	-----	FLOATING POINT	00130000
		SPACE 2			00131000
ECSRSLT	DSECT		--	EDIT ROUTINE RESULT BUFFER	00132000
ECSRSLTC	DS	CL32767	-----	CHARACTER STRING	00133000

Figure 66. User edit routine field definitions for assembler DXEECS control block (Part 3 of 3)

Assembling Your Program

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 assembler supplied with CMS.

Generating Your Program

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 PL/I without Language Environment (LE)

The QMF edit exit interface for PL/I consists of these parts:

- A control program, which is shipped with QMF as DSQUXIP
- A control module, which is shipped with QMF as DSQUPLI
- A control block, which is shipped with QMF as DXEECSP
- Your edit exit program, which is named DSQUXDT

Figure 67 on page 204 shows the program structure of a PL/I edit exit routine for CMS.

Creating Your Own Edit Codes for QMF Forms

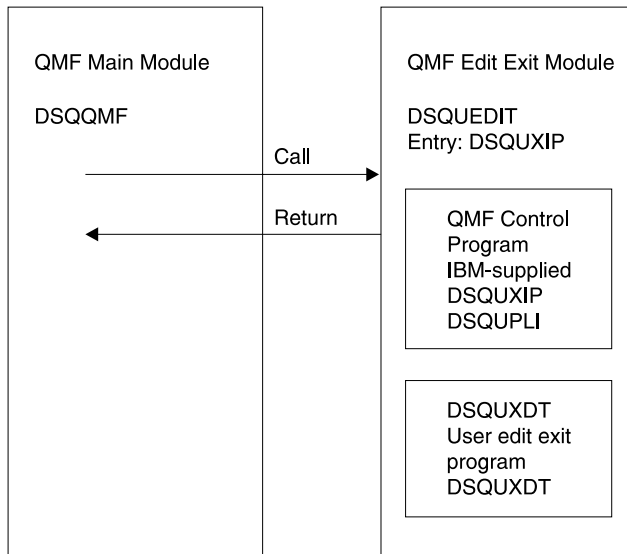


Figure 67. Program structure of a PL/I edit exit routine

The IBM-supplied sample edit program for PL/I, DSQUXDTP, is located on the QMF production disk. The sample program is commented so that you can modify it to suit your needs. If you plan to use the sample 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 DXEECSF, which is a member of DSQUSERE MACLIB. It is DXEECSF 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

Creating Your Own Edit Codes for QMF Forms

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 assembler.

In the sample program, the actual parameter descriptions appear in the previously mentioned block of definitions comprising DXEECS (Figure 68). The declaration for the control block begins with:

```
DECLARE
  1 DXEECS BASED(ECSPTR)
  :
```

The statements defining the other two parameters are:

```
DECLARE
  ECSINPT CHARACTER(32767)
  BASED(ECSINPT), ... 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);
ECSINPT  = ADDR(ECSINPTF);
ECSRSLTP = ADDR(ECSRSLTF);
```

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. 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.

Figure 68 on page 206 shows the DXEECS control block for PL/I.

Creating Your Own Edit Codes for QMF Forms

```

/*****/ 00001000
/*      */ 00002000
/* CONTROL BLOCK NAME: DXEECS (PLI VERSION) */ 00003000
/*      */ 00004000
/* FUNCTION: */ 00005000
/*      */ 00006000
/* THIS IS THE INTERFACE CONTROL BLOCK BETWEEN QMF AND */ 00007000
/* THE USER EDITING ROUTINE INTERFACE, DSQUEDIT (TSO/CMS). */ 00008000
/* OR DSQUECIC (CICS). */ 00009000
/*      */ 00010000
/* IT CONTAINS THE USER'S EDIT CODE, IDENTIFIES THE SOURCE */ 00011000
/* DATA AND THE TARGET LOCATION FOR THE EDITED RESULT */ 00012000
/* AND PROVIDES A SCRATCHPAD AREA FOR THE USER EDIT */ 00013000
/* ROUTINE'S USE. */ 00014000
/*      */ 00015000
/* THIS CONTROL BLOCK IS PERSISTENT BETWEEN CALLS TO THE */ 00016000
/* USER EDIT ROUTINE. */ 00017000
/*      */ 00018000
/* THE SCRATCHPAD AREA WILL NOT BE MODIFIED BY QMF AFTER */ 00019000
/* THE INITIAL INVOCATION OF THE EXIT ROUTINE. */ 00020000
/*      */ 00021000
/*      */ 00022000
/*      */ 00023000
/* STATUS: VERSION 7 RELEASE 1 LEVEL 0 */ 00024000
/*      */ 00025000
/* INNER CONTROL BLOCKS: NONE */ 00026000
/*      */ 00027000
/* CHANGE ACTIVITY: */ 00028000
/*      */ 00029000
/* CHANGE DATE: */ 00030000
/*      */ 00031000
/*****/ 00032000
DECLARE 00033000
  1 DXEECS BASED(ECSPTR), /* EDIT ROUTINE INFORMATION */ 00034000
    3 ECSNAME CHARACTER(8), /* CONTROL BLOCK IDENTIFICATION */ 00035000
                                00036000
  3 ECSEDCTL, /* EDIT CONTROL */ 00037000
    5 ECSFREQ CHARACTER(1), /* FUNCTION REQUEST */ 00038000
                                (CODES ARE DEFINED BELOW) */ 00039000
    5 ECSPAD10 CHARACTER(3), /* RESERVED FIELD */ 00040000
    5 ECSECODE CHARACTER(5), /* EDIT CODE FROM FORM OBJECT */ 00041000
    5 ECSPAD20 CHARACTER(3), /* RESERVED FIELD */ 00042000
    5 ECSDECP CHARACTER(1), /* SYMBOL FOR DECIMAL POINT */ 00043000
                                (AS DEFINED BY DECIMAL OPTION */ 00044000
                                IN CURRENT PROFILE OBJECT) */ 00045000
  5 ECSTHSEP CHARACTER(1), /* SYMBOL FOR THOUSANDS SEPARATOR */ 00046000
                                (AS DEFINED BY DECIMAL OPTION */ 00047000

```

Figure 68. User edit routine field definitions for PL/I DXEECS control block (Part 1 of 4)

Creating Your Own Edit Codes for QMF Forms

```

                                IN CURRENT PROFILE OBJECT) */ 00048000
5 ECSPAD30 CHARACTER(6),      /* RESERVED FIELD */ 00049000
5 ECSQMF CHARACTER(20),      /* AREA RESERVED FOR QMF'S USE */ 00050000
                                00051000
3 ECSINDTA,                  /* DESCRIPTION OF THE INPUT DATA*/ 00052000
5 ECSINTYP FIXED BINARY(31), /* DATA TYPE OF THE INPUT AS 00053000
                                IT EXISTS IN THE DATA BASE 00054000
                                (SEE CODES DEFINED BELOW) */ 00055000
5 ECSINLEN FIXED BINARY(31), /* LENGTH OF INPUT DATA */ 00056000
5 ECSINPRC FIXED BINARY(15), /* PRECISION OF INPUT DATA IF 00057000
                                IS IT DECIMAL DATA TYPE 00058000
                                (U-TYPE EDIT CODE) OR 00059000
                                IF IT WAS ANY NUMERIC 00060000
                                DATA TYPE (V-TYPE EDIT 00061000
                                CODE)... 00062000
                                ZERO OTHERWISE */ 00063000
5 ECSINSCL FIXED BINARY(15), /* SCALE OF INPUT DATA IF 00064000
                                IS IT DECIMAL DATA TYPE 00065000
                                (U-TYPE EDIT CODE) OR 00066000
                                IF IT WAS ANY NUMERIC 00067000
                                DATA TYPE (V-TYPE EDIT 00068000
                                CODE)... 00069000
                                ZERO OTHERWISE */ 00070000
5 ECSINSGN CHARACTER(1),    /* SIGN (V-TYPE EDIT ONLY) 00071000
                                SEE VALUES DEFINED 00072000
                                BELOW */ 00073000
5 ECSINNULL CHARACTER(1),   /* NULL INPUT DATA INDICATOR 00074000
                                SEE VALUE DEFINED 00075000
                                BELOW */ 00076000
5 ECSPAD40 CHARACTER(10),   /* RESERVED FIELD */ 00077000
3 ECSRSDTA,                  /* DESCRIPTION OF THE RESULT 00078000
                                BUFFER */ 00079000
5 ECSRSLEN FIXED BINARY(31), /* LENGTH (EQUIVALENT TO 00080000
                                COLUMN WIDTH IN THE 00081000
                                FORM OBJECT) */ 00082000
5 ECSPAD50 CHARACTER(12),   /* RESERVED FIELD */ 00083000
                                00084000
3 ECSUCTL,                  /* USER CONTROL AREA */ 00085000
5 ECSERRET FIXED BINARY(31), /* EDIT ROUTINE ERROR RETURN CODE 00086000
                                (SEE CODES DEFINED BELOW) */ 00087000
5 ECSRQMF CHARACTER(1),     /* REQUEST FOR QMF 00088000
                                (SEE CODE(S) DEFINED BELOW) */ 00089000
5 ECSPAD60 CHARACTER(11),   /* RESERVED FIELD */ 00090000
                                00091000
3 ECSUSERS CHARACTER(256);  /* USER SCRATCH PAD AREA */ 00092000

```

Figure 68. User edit routine field definitions for PL/I DXEECS control block (Part 2 of 4)

Creating Your Own Edit Codes for QMF Forms

```

                                00093000
DECLARE                               /* INPUT DATA PARAMETER... */ 00094000
  ECSINPT CHARACTER(32767)           /* CHARACTER INPUT DATA */ 00095000
    BASED(ECSINPT),                 00096000
  ECSINSIN FIXED BINARY(15)         /* SMALL INTEGER INPUT DATA */ 00097000
    BASED(ECSINPT),                 00098000
  ECSININT FIXED BINARY(31)         /* INTEGER INPUT DATA */ 00099000
    BASED(ECSINPT),                 00100000
  ECSINFLT FLOAT BINARY(53)         /* FLOATING POINT INPUT DATA */ 00101000
    BASED(ECSINPT);                 00102000
                                      00103000
DECLARE                               /* RESULT BUFFER PARAMETER... */ 00104000
  ECSRSLT CHARACTER(32767)          /* EDIT ROUTINE RESULT BUFFER */ 00105000
    BASED(ECSRSLTP);                 00106000
                                      00107000
DECLARE                               00108000
  (ECSPTR,                           /* MUST CONTAIN DXECS ADDRESS */ 00109000
   ECSINPT,                           /* MUST CONTAIN ECSINPT ADDRESS */ 00110000
   ECSRSLTP                             /* MUST CONTAIN ECSRSLT ADDRESS */ 00111000
  ) POINTER;                           00112000
                                      00113000
                                      00114000
DECLARE (                               /* DATA TYPE CONSTANTS: */ 00115000
                                           (SEE ECSINTYP ABOVE) */ 00116000
  ECSINT INITIAL(496),               /* INTEGER */ */ 00117000
  ECSSINT INITIAL(500),              /* SMALL INTEGER */ */ 00118000
  ECSFLT INITIAL(480),               /* FLOATING POINT */ */ 00119000
  ECSVCHR INITIAL(448),              /* VARYING CHARACTER */ */ 00120000
  ECSFCHR INITIAL(452),              /* FIXED CHARACTER */ */ 00121000
  ECSLCHR INITIAL(456),              /* VERY LONG CHARACTER */ */ 00122000
  ECSVG INITIAL(464),                /* VARYING GRAPHIC */ */ 00123000
  ECSFG INITIAL(468),                /* FIXED GRAPHIC */ */ 00124000
  ECSLG INITIAL(472),                /* VERY LONG GRAPHIC */ */ 00125000
  ECSDEC INITIAL(484),               /* DECIMAL */ */ 00126000
  ECSDATE INITIAL(384),              /* DATE */ */ 00127000
  ECSTIME INITIAL(388),              /* TIME */ */ 00128000
  ECSTS INITIAL(392),                /* TIMESTAMP */ */ 00129000
  ECSFLTIX INITIAL(940)              /* EXTENDED FLOATING POINT */ */ 00130000
  ) FIXED BINARY(31) STATIC;          00131000
                                      00132000
                                      00133000
DECLARE (                               /* FUNCTION REQUEST CONSTANTS */ 00134000

```

Figure 68. User edit routine field definitions for PL/I DXECS control block (Part 3 of 4)

Creating Your Own Edit Codes for QMF Forms

```

                                (SEE ECSFREQ ABOVE)          */ 00135000
ECSFEDIT INITIAL('E'),          /* EDIT                      */ 00136000
ECSFTERM INITIAL('T')          /* TERMINATE                  */ 00137000
                                (TO FREE RESOURCES...      00138000
                                QMF WILL CALL THE USER     00139000
                                EDIT ROUTINE FOR THIS      00140000
                                FUNCTION ONLY IF THE       00141000
                                USER EDIT ROUTINE HAS     00142000
                                PREVIOUSLY REQUESTED     00143000
                                IT.)                      */ 00144000
                                ) CHARACTER(1) STATIC;      00145000
                                                                00146000
                                                                00147000
                                                                00148000
DECLARE (                      /* PLUS/MINUS SIGN CONSTANTS 00149000
                                (SEE ECSINSGN ABOVE)      */ 00149000
                                ECSPLUS INITIAL(' '),      /* INPUT DATA IS POSITIVE   */ 00150000
                                ECSMINUS INITIAL('-'),     /* INPUT DATA IS NEGATIVE   */ 00151000
                                ) CHARACTER(1) STATIC;      00152000
                                                                00153000
                                                                00154000
                                                                00155000
DECLARE (                      /* NULL INDICATION CONSTANT  00155000
                                (SEE ECSINNUL ABOVE)      */ 00156000
                                ECSNULL INITIAL('N')      /* INPUT DATA IS NULL       */ 00157000
                                ) CHARACTER(1) STATIC;      00158000
                                                                00159000
                                                                00160000
                                                                00161000
DECLARE (                      /* REQUEST-FOR-QMF CONSTANTS 00161000
                                (SEE ECSRQMF ABOVE)        */ 00162000
                                ECSRTERM INITIAL('T')     /* REQUEST QMF TO INVOKE    00163000
                                USER EDIT ROUTINE FOR     00164000
                                TERMINATION FUNCTION      */ 00165000
                                ) CHARACTER(1) STATIC;      00166000
                                                                00167000
                                                                00168000
                                                                00169000
DECLARE (                      /* QMF-DEFINED ERROR RETURN  00169000
                                CONSTANTS                  00170000
                                (SEE ECSERRET ABOVE)      */ 00171000
                                ECSERR01 INITIAL(99101),  /* UNRECOGNIZED EDIT CODE   */ 00172000
                                ECSERR02 INITIAL(99102),  /* IMPROPER INPUT DATA TYPE */ 00173000
                                                                /* REQUESTED EDIT EDIT CODE */ 00174000
                                ECSERR03 INITIAL(99103),  /* INVALID INPUT DATA VALUE */ 00175000
                                                                /* RECEIVED                  */ 00176000
                                ECSERR04 INITIAL(99104),  /* LENGTH OF INPUT DATA IS  */ 00177000
                                                                /* SHORT                     */ 00178000
                                ECSERR05 INITIAL(99105)   /* LENGTH OF RESULT BUFFER IS */ 00179000
                                                                /* TOO SHORT                  */ 00180000
                                ) FIXED BINARY(31) STATIC; 00181000

```

Figure 68. User edit routine field definitions for PL/I DXEECS control block (Part 4 of 4)

Compiling Your Program

Before compiling your program, ensure that you can access the IBM-supplied control block DXEECS. DXEECS is located in the QMF library DSQUSERE

Creating Your Own Edit Codes for QMF Forms

MACLIB on the QMF production disk. You need to access the QMF and PL/I production disks. You also 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
```

To compile your edit program, DSQUXDT, your edit program must not use the procedure option MAIN.

Compile without including the 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;
```

If you're using PL/I Version 2: Use the PL/I SYSTEM(MVS) compile-time option. QMF expects the MVS style of parameter list when running in CMS.

Compile the IBM-supplied program, DSQUPLI, using the same options as used for DSQUXDT, except that DSQUPLI specifies the procedure option 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.

Creating Your Own Edit Codes for QMF Forms

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)

The QMF edit exit interface for PL/I in VM for LE consists of these parts:

- Interface control block, which is shipped with QMF as DXEECS
- 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 69 on page 212 shows the program structure of a PL/I edit exit routine for CMS.

Creating Your Own Edit Codes for QMF Forms

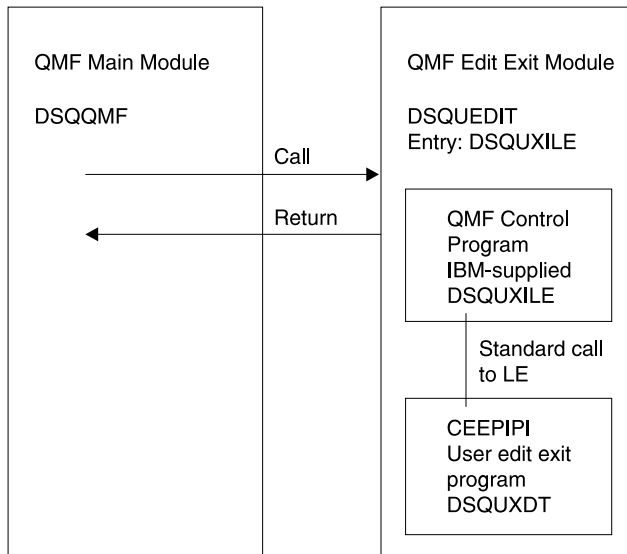


Figure 69. Program structure of a PL/I edit exit routine with LE

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). 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.

Creating Your Own Edit Codes for QMF Forms

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 without Language Environment (LE)

COBOL refers to VS COBOL II, COBOL/370, and IBM COBOL for OS/390 and VM unless otherwise stated.

The QMF edit exit interface for COBOL consists of these parts:

- Interface control block, which is supplied by IBM as DXEEESC
- Control program, which is supplied by IBM 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 70 shows the structure of a COBOL edit exit routine in CMS.

The IBM-supplied sample edit exit program, DSQUXDTC, is commented so

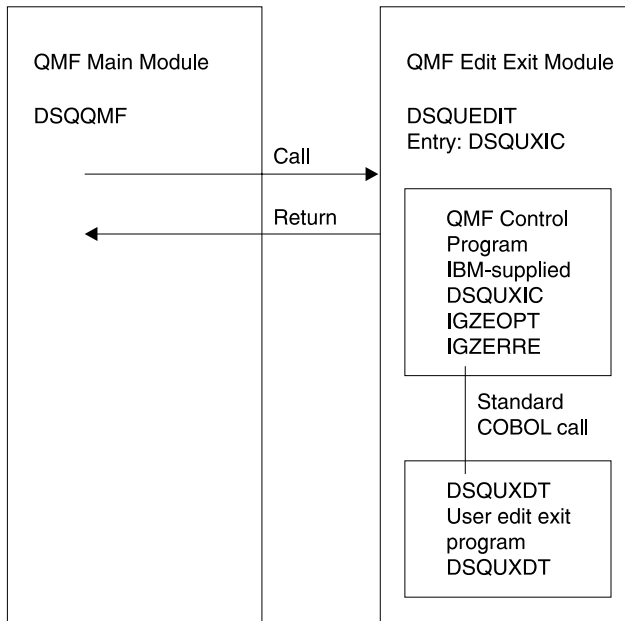


Figure 70. Program structure of a COBOL edit exit routine

that you can browse it online, print it, or modify it 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 DXEEESC, which is a member of DSQUSERE MACLIB. It is DXEEESC that defines the input fields, giving

Creating Your Own Edit Codes for QMF Forms

them the names we are using in this chapter. You can include DXEECS in your own user edit routine. A listing of the module appears in Figure 71 on page 215.

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 DXEECS.

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.

Figure 71 on page 215 shows the DXEECS control block for COBOL.

Creating Your Own Edit Codes for QMF Forms

```

***** 00001000
*      * 00002000
* CONTROL BLOCK NAME: DXEECS (COBOL VERSION) * 00003000
*      * 00004000
* FUNCTION: * 00005000
*      * 00006000
* THIS IS THE INTERFACE CONTROL BLOCK BETWEEN QMF AND * 00007000
* THE USER EDITING INTERFACE, DSQUEDIT (TSO/CMS), OR * 00008000
* DSQUECIC (CICS). * 00009000
*      * 00010000
* IT CONTAINS THE USER'S EDIT CODE, IDENTIFIES THE SOURCE * 00011000
* DATA AND THE TARGET LOCATION FOR THE EDITED RESULT * 00012000
* AND PROVIDES A SCRATCHPAD AREA FOR THE USER EDIT * 00013000
* ROUTINE'S USE. * 00014000
*      * 00015000
* THIS CONTROL BLOCK IS PERSISTENT BETWEEN CALLS TO THE * 00016000
* USER EDIT ROUTINE. * 00017000
*      * 00018000
* THE SCRATCHPAD AREA WILL NOT BE MODIFIED BY QMF AFTER * 00019000
* THE INITIAL INVOCATION OF THE EXIT ROUTINE. * 00020000
*      * 00021000
*      * 00022000
* NOTE: THIS FILE IS DESIGNED TO BE COPIED INTO THE LINKAGE * 00023000
* SECTION OF THE USER EDIT ROUTINE. * 00024000
*      * 00025000
*      * 00026000
* STATUS: VERSION 7 RELEASE 1 LEVEL 0 * 00027000
*      * 00028000
* INNER CONTROL BLOCKS: NONE * 00029000
*      * 00030000
* CHANGE ACTIVITY: NEW CONTROL BLOCK * 00031000
*      * 00032000
* CHANGE DATE: * 00033000
*      * 00034000
***** 00035000
00036000
01 DXEECS. * 00037000
  02 ECSNAME PICTURE X(8). * 00038000
* -- CONTROL BLOCK IDENTIFICATION * 00039000
* * 00040000
  02 ECSEDCTL. * 00041000
* -- EDIT CONTROL * 00042000
* * 00043000
  03 ECSFREQ PICTURE X(1). * 00044000
* -- FUNCTION REQUEST * 00045000

```

Figure 71. User edit routine field definitions for COBOL version of DXEECS control block (Part 1 of 5)

Creating Your Own Edit Codes for QMF Forms

```

      88 ECS-EDIT-FUNCTION      VALUE "E".          00046000
      88 ECS-TERMINATE-FUNCTION VALUE "T".          00047000
*
*      ---- TERMINATE FUNCTION TO FREE RESOURCES. 00048000
*      QMF WILL CALL THE USER EDIT ROUTINE      00049000
*      FOR THIS FUNCTION ONLY IF THE USER       00050000
*      EDIT ROUTINE HAS PREVIOUSLY REQUESTED    00051000
*      IT. (SEE ECSRQMF BELOW.)                 00052000
03 ECSPAD10  PICTURE X(3).                       00053000
*
*      -- RESERVED FIELD                        00054000
03 ECSECODE  PICTURE X(5).                       00055000
*
*      -- EDIT CODE FROM FORM OBJECT           00056000
03 ECSPAD20  PICTURE X(3).                       00057000
*
*      -- RESERVED FIELD                        00058000
03 ECSDECPT  PICTURE X(1).                       00059000
*
*      -- SYMBOL FOR DECIMAL POINT             00060000
*      -- (AS DEFINED BY DECIMAL OPTION IN     00061000
*      -- CURRENT PROFILE OBJECT               00062000
03 ECSTHSEP  PICTURE X(1).                       00063000
*
*      -- SYMBOL FOR THOUSANDS SEPARATOR       00064000
*      -- (AS DEFINED BY DECIMAL OPTION IN     00065000
*      -- CURRENT PROFILE OBJECT               00066000
03 ECSPAD30  PICTURE X(6).                       00067000
*
*      -- RESERVED FIELD                        00068000
03 ECSQMF    PICTURE X(20).                      00069000
*
*      -- AREA RESERVED FOR QMF'S USE         00070000
*
*      00071000
02 ECSINDTA.                                     00072000
*
*      -- DESCRIPTION OF THE INPUT DATA       00073000
*
*      00074000
03 ECSINTYP  PICTURE S9(9) COMPUTATIONAL.        00075000
*
*      -- DATA TYPE OF THE INPUT AS IT       00076000
*      -- EXISTS IN THE DATA BASE.           00077000
      88 ECS-FLOATING-POINT  VALUE IS +480.      00078000
      88 ECS-DECIMAL         VALUE IS +484.      00079000
      88 ECS-INTEGERS        VALUE IS +496.      00080000
      88 ECS-SMALL-INTEGERS  VALUE IS +500.      00081000
      88 ECS-VARCHAR         VALUE IS +448.      00082000
      88 ECS-FIXED-CHAR      VALUE IS +452.      00083000
      88 ECS-LONG-VARCHAR    VALUE IS +456.      00084000
      88 ECS-VARG            VALUE IS +464.      00085000
      88 ECS-FIXED-G         VALUE IS +468.      00086000
      88 ECS-LONG-VARG       VALUE IS +472.      00087000
      88 ECS-DATE            VALUE IS +384.      00088000
      88 ECS-TIME            VALUE IS +388.      00089000
      88 ECS-TIMESTAMP       VALUE IS +392.      00090000
      88 ECS-EXT-FLOATING-POINT VALUE IS +940.    00091000
03 ECSINLEN  PICTURE S9(5) USAGE IS COMPUTATIONAL. 00092000
*
*      -- LENGTH OF INPUT DATA                00093000
03 ECSINPRC  PICTURE S9(2) USAGE IS COMPUTATIONAL. 00094000

```

Figure 71. User edit routine field definitions for COBOL version of DXEECS control block (Part 2 of 5)

Creating Your Own Edit Codes for QMF Forms

```

*           -- PRECISION OF INPUT DATA IF IT IS           00095000
*           -- DECIMAL DATA TYPE (U-TYPE EDIT CODE)       00096000
*           -- OR IF IT WAS ANY NUMERIC DATA TYPE         00097000
*           -- (V-TYPE EDIT CODE)...                       00098000
*           -- ZERO OTHERWISE.                             00099000
03 ECSINSCL PICTURE S9(2) USAGE IS COMPUTATIONAL.         00100000
*           -- SCALE OF INPUT DATA IF IT IS               00101000
*           -- DECIMAL DATA TYPE (U-TYPE EDIT CODE)       00102000
*           -- OR IF IT WAS ANY NUMERIC DATA TYPE         00103000
*           -- (V-TYPE EDIT CODE)...                       00104000
*           -- ZERO OTHERWISE.                             00105000
03 ECSINSGN PICTURE X(1).                                 00106000
*           -- SIGN OF CONVERTED NUMERIC DATA             00107000
*           -- (V-TYPE EDIT CODE ONLY)...                  00108000
      88 ECS-POSITIVE VALUE " ".                          00109000
      88 ECS-NEGATIVE VALUE "-".                          00110000
*
03 ECSINNUL PICTURE X(1).                                 00111000
*           -- NULL INPUT DATA INDICATOR                 00112000
      88 ECS-NULL-DATA VALUE "N".                         00113000
*
03 ECSPAD40 PICTURE X(10).                                00114000
*           -- RESERVED FIELD                              00115000
*
03 ECSPAD40 PICTURE X(10).                                00116000
*           -- RESERVED FIELD                              00117000
*
03 ECSPAD40 PICTURE X(10).                                00118000
*           -- RESERVED FIELD                              00119000
02 ECSRSDTA.                                             00120000
*           -- DESCRIPTION OF THE RESULT BUFFER           00121000
*
03 ECSRSLEN PICTURE S9(5) USAGE IS COMPUTATIONAL.         00122000
*           -- LENGTH OF RESULT AREA                      00123000
*           -- (EQUIVALENT TO COLUMN WIDTH IN THE        00124000
*           -- FORM OBJECT                                 00125000
03 ECSPAD50 PICTURE X(12).                                00126000
*           -- RESERVED FIELD                              00127000
*
03 ECSPAD50 PICTURE X(12).                                00128000
*           -- RESERVED FIELD                              00129000
02 ECSUCTL.                                              00130000
*           -- USER CONTROL AREA                          00131000
*
03 ECSERRET PICTURE S9(9) USAGE IS COMPUTATIONAL.         00132000
*           -- EDIT ROUTINE ERROR RETURN CODE             00133000
*           -- (SEE QMF-DEFINED ERROR CODES BELOW).       00134000
03 ECSRQMF  PICTURE X(1).                                 00135000
*           -- REQUEST FOR QMF                             00136000
*           -- (SEE CODE(S) DEFINED BELOW.)               00137000
03 ECSPAD60 PICTURE X(11).                                00138000
*           -- RESERVED FIELD                              00139000
*
03 ECSPAD60 PICTURE X(11).                                00140000
*           -- RESERVED FIELD                              00141000
02 ECSUSERS.                                             00141000

```

Figure 71. User edit routine field definitions for COBOL version of DXECS control block (Part 3 of 5)

Creating Your Own Edit Codes for QMF Forms

```
*                                -- USER SCRATCH PAD AREA                                00142000
                                                                00143000
03 ECSUSERS-ARRAY                                00144000
    PICTURE X(1)                                00145000
    OCCURS 256 TIMES.                            00146000
                                                                00147000
                                                                00148000
                                                                00149000
*****                                -- EDIT ROUTINE INPUT DATA
01 ECSINPT.                                    00150000
02 ECSINPTC    PICTURE X(32767).                00151000
02 ECSINPT-ARRAY REDEFINES ECSINPTC            00152000
    PICTURE X(1)                                00153000
    OCCURS 32767 TIMES.                        00154000
02 ECSINPT-INTEG-OVL                            00155000
    REDEFINES ECSINPTC.                        00156000
03 ECSINPT-INTEG                                00157000
    PICTURE S9(9)                               00158000
    USAGE IS COMPUTATIONAL.                    00159000
03 FILLER    PICTURE X(1)                       00160000
    OCCURS 32763 TIMES.                        00161000
02 ECSINPT-SMALL-INTEG-OVL                      00162000
    REDEFINES ECSINPTC.                        00163000
03 ECSINPT-SMALL-INTEG                          00164000
    PICTURE S9(4)                               00165000
    USAGE IS COMPUTATIONAL.                    00166000
03 FILLER    PICTURE X(1)                       00167000
    OCCURS 32765 TIMES.                        00168000
02 ECSINPT-FLOATING-POINT-OVL                   00169000
    REDEFINES ECSINPTC.                        00170000
03 ECSINPT-FLOATING-POINT                       00171000
    USAGE IS COMPUTATIONAL-2.                  00172000
03 FILLER    PICTURE X(1)                       00173000
    OCCURS 32759 TIMES.                        00174000
```

Figure 71. User edit routine field definitions for COBOL version of DXEECS control block (Part 4 of 5)

Creating Your Own Edit Codes for QMF Forms

```

                                                                                                                                 00175000
                                                                                                                                 00176000
*****          -- EDIT ROUTINE RESULT BUFFER          00177000
01  ECSRSLT.                                           00178000
    02 ECSRSLT-ARRAY  PICTURE X(1)                    00179000
                        OCCURS 1 TO 32767 TIMES        00180000
                        DEPENDING ON ECSRSLLEN.        00181000
                                                                                                                                 00182000
*****          *****                                00183000
*                                                                                                                                 * 00184000
* THE DATA DEFINITIONS BELOW ARE FOR DOCUMENTATION   * 00185000
* PURPOSES ONLY SINCE COBOL DOES NOT ALLOW LINKAGE   * 00186000
* SECTION DATA DEFINITIONS TO HAVE VALUE CLAUSES   * 00187000
*                                                                                                                                 * 00188000
*****          *****                                00189000
                                                                                                                                 00190000
*****          -- QMF-DEFINED VALUES FOR ECSERRET   00191000
*                                                                                                                                 *
*              (SEE ABOVE).                            00192000
*77  ECS-UNKNOWN-EDIT-CODE                             00193000
*              PICTURE S9(9) VALUE IS +99101         00194000
*              USAGE IS COMPUTATIONAL.                00195000
*77  ECS-IMPROPER-DATA-TYPE                             00196000
*              PICTURE S9(9) VALUE IS +99102         00197000
*              USAGE IS COMPUTATIONAL.                00198000
*77  ECS-INVALID-DATA-VALUE                             00199000
*              PICTURE S9(9) VALUE IS +99103         00200000
*              USAGE IS COMPUTATIONAL.                00201000
*77  ECS-INPUT-DATA-TOO-SHORT                           00202000
*              PICTURE S9(9) VALUE IS +99104         00203000
*              USAGE IS COMPUTATIONAL.                00204000
*77  ECS-RESULT-BUFFER-TOO-SHORT                        00205000
*              PICTURE S9(9) VALUE IS +99105         00206000
*              USAGE IS COMPUTATIONAL.                00207000
*                                                                                                                                 00208000
*                                                                                                                                 00209000
*****          -- POSSIBLE REQUEST-FOR-QMF CODES     00210000
*                                                                                                                                 *
*              (SEE ECSRQMF ABOVE).                    00211000
*77  ECS-CALL-FOR-TERMINATE                             00212000
*              PICTURE X(1) VALUE IS "T".            00213000

```

Figure 71. User edit routine field definitions for COBOL version of DXEECS control block (Part 5 of 5)

Compiling Your Program

To create a CMS module file from COBOL source code, ensure that you can access the IBM-supplied control block DXEECS. DXEECS is located in the QMF library DSQUSERE MACLIB on the QMF production disk. 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:

Creating Your Own Edit Codes for QMF Forms

```
GLOBAL MACLIB DSQUSERE VSC2MAC
```

DXEEESC, as distributed by IBM, uses quotation marks (“”) to delimit character literals. If your program uses apostrophes (’), you must either change DXEEESC as distributed by IBM or copy the structure to your program, changing quotes to apostrophes.

You can compile your program using the options:

COBOL II:

```
LIB, NODYNAM, OBJECT, RENT, RES
```

COBOL/370 and IBM COBOL for OS/390 and VM:

```
LIB, NODYNAM, OBJECT, RENT
```

Assembling the Run Time Options Macro (COBOL II)

Use the C2CUSTL EXEC provided by IBM to assemble IGZOPT. Follow the prompts and add option STAE=N0 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

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 *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 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)

The QMF edit exit interface for COBOL in VM for LE consists of these parts:

- Interface control block, which is supplied with QMF as DXEECS
- 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 72 on page 222 shows the structure of a COBOL edit exit routine in CMS.

Creating Your Own Edit Codes for QMF Forms

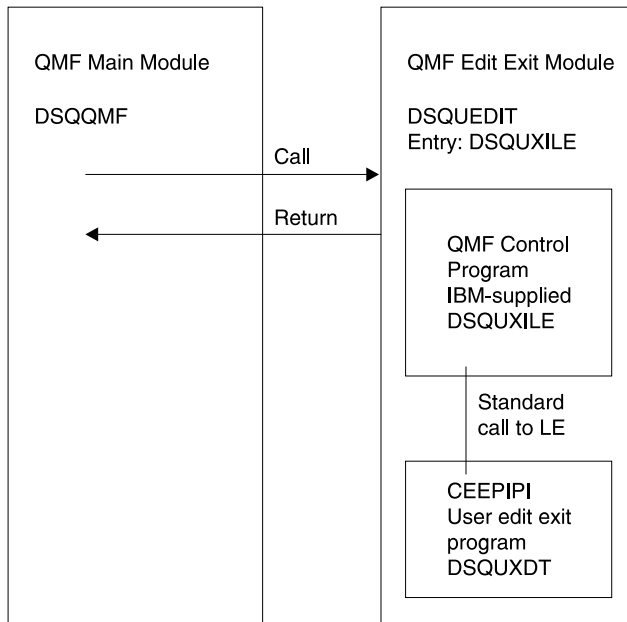


Figure 72. Program structure of a COBOL edit exit routine with LE

Generating Your COBOL Program for LE

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
```

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

Creating Your Own Edit Codes for QMF Forms

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:

```
ccccSodededededededededeSi ccSodededededeSi
```

Here, c, d, and e stand for any possible byte, while 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 ECSRLEN field. If your results do not fit, truncate them on the right. If the last character represented in the truncated results is a DBCS character, be certain to retain its 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 don't 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:

Creating Your Own Edit Codes for QMF Forms

- 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 13. 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 with QMF, with default limits for the amount of time spent executing a QMF command or 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.

Quick start

Use the steps in Table 34 to guide you in setting up and using a governor exit routine. If you need more information on any step, see the page listed at the right of the table.

Table 34. Using a governor exit routine

To do this task:	See:
To prompt users when the number of database rows retrieved reaches 25 000, and cancel data retrieval when the number reaches 100 000 , turn the governor on by setting the INTVAL field of Q.RESOURCE_VIEW to 0 (where RESOURCE_GROUP=SYSTEM and RESOURCE_OPTION=SCOPE). Then update the RESOURCE_GROUP field of the user's profile to SYSTEM, and reconnect to the database.	Page 229
To prompt users when 15 minutes of real time has elapsed and cancel data when 60 minutes of real time has elapsed , turn the governor on by setting the INTVAL field of Q.RESOURCE_VIEW to 0 (where RESOURCE_GROUP=SYSTEM and RESOURCE_OPTION=SCOPE). Then update the RESOURCE_GROUP field of the user's profile to SYSTEM, and reconnect to the database.	Page 229
To set up the governor exit routine to use database row limits other than the defaults of 25 000 and 100 000, add new rows to Q.RESOURCE_TABLE that define the points at which you want to warn the user (optional) and cancel data retrieval. Turn the governor on and update the user's profile as explained in step 227.	Page 233
To limit activities other than the number of rows retrieved from the database , use assembler to modify the IBM-supplied governor exit routine or write a routine of your own.	Page 238
If you modify the IBM-supplied governor exit routine or write your own routine , assemble and generate the routine.	Page 261

Controlling QMF Resources Using a Governor Exit Routine

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 or how much time is spent 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 25000, at which time the user sees the message shown in Figure 73:

```
DSQUn00 QMF governor prompt:  
Command has fetched 25000 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 73. Message displayed when a resource limit is approaching. The n symbol in the figure represents an NLID from Table 5 on page 19

- 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 73. When the IBM-supplied governor cancels data retrieval, the user sees the message shown in Figure 74:

```
Row limit exceeded! Your command canceled by QMF governor.
```

Figure 74. 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 74. For example, if your procedure contains a command that requires the report to complete (such as ERASE), you receive the message shown in Figure 75:

```
Procedure canceled.
```

Figure 75. Message displayed when a procedure is canceled

Users using the SYSTEM profile, discussed in “Establishing a Profile Structure for Your Installation” on page 98, are already set up to use these default values of 25 000 and 100 000. To activate the default values for users with unique profiles, see “Activating the Default Limits” on page 229.

Controlling QMF Resources Using a Governor Exit Routine

The governor exit routine also has predefined values 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 76:

```
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 76. Message displayed when a resource limit is approaching. The n symbol in the figure represents an NLID from

- A time prompt value cancels the command when 60 minutes of real time has been used during the cycle.

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 233.

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 77 from the SQL query panel:

```
UPDATE Q.RESOURCE_VIEW  
SET INTVAL=0  
WHERE RESOURCE_OPTION='SCOPE' AND  
RESOURCE_GROUP='SYSTEM'
```

Figure 77. 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 78 on page 230 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.

Controlling QMF Resources Using a Governor Exit Routine

```
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 78. Updating a user's resource group

For more information on how to create a new user profile in the Q.PROFILES table, see “Creating User Profiles to Enable User Access to QMF” on page 97.

3. Instruct the user to reconnect to the database to activate the new values. For example, user JONES, who has the password MYPW, enters the following command:

```
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.

See Table 13 on page 95 for how to grant a user authority to connect to the database. Users who do not have DB2 for VM CONNECT authority can end the current QMF session and begin another to activate the new resource group.

“How a Governor Exit Routine Controls Resources” 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”. Then see the procedure in “Defining Your Own Resource Limits” on page 233.

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.

Controlling QMF Resources Using a Governor Exit Routine

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 35 on page 237 shows the structure of the Q.RESOURCE_TABLE as it is shipped by IBM. Q.RESOURCE_TABLE has the index Q.RESOURCE_INDEX, with the UNIQUE attribute. 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 six predefined resource options, as shown in Figure 79. Use the CHARVAL column to indicate the limits defined in each row, as shown.

RESOURCE GROUP	RESOURCE OPTION	INTVAL	FLOATVAL	CHARVAL
SYSTEM	SCOPE	0	-	INDICATE WHETHER GOVERNOR IS ACTIVE
SYSTEM	ROWLIMIT	100000	-	CANCEL AFTER FETCHING 100000 ROWS
SYSTEM	ROWPROMPT	25000	-	PROMPT USER AFTER FETCHING 25000 ROWS
SYSTEM	TIMELIMIT	3600	-	CANCEL AFTER 60 MINUTES
SYSTEM	TIMEPROMPT	900	-	PROMPT USER AFTER 15 MINUTES
SYSTEM	TIMECHECK	900	-	15 MINUTE INTERVAL BETWEEN TIME CHECK

Figure 79. Default resource group and options for the IBM-supplied governor exit

SCOPE = 0

Activates governing for a particular resource group.

ROWLIMIT = 100000

If the user decides to continue when warned, the governor exit routine cancels data retrieval activities after 100 000 rows are retrieved. (Retrieval is for FETCH only.) ROWLIMIT is dependent on the buffer size; therefore, more than 100 000 rows can be retrieved if the buffer holds a number of rows not divisible by 100 000.

ROWPROMPT = 25000

Warns the user when 25 000 database rows have been retrieved.

TIMELIMIT = 3600

If the user decides to continue when warned, the governor exit routine cancels the command after 60 minutes have elapsed.

Controlling QMF Resources Using a Governor Exit Routine

TIMELIMIT is checked at TIMECHECK intervals; therefore, more than 60 minutes can elapse if the TIMECHECK interval is set at an interval not divisible by 60.

TIMEPROMPT = 900

Warns the user when 15 minutes have elapsed.

TIMECHECK = 900

Specifies 15 minutes of real time between time checks for 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 80 on page 233 shows how the governor limits use of resources.

Controlling QMF Resources Using a Governor Exit Routine

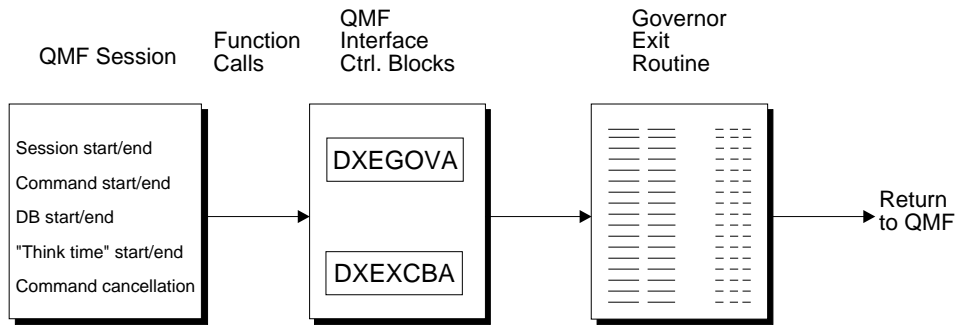


Figure 80. How a governor exit routine works with QMF

QMF calls the governor exit routine at a number of different points within the QMF session, as shown in Figure 80. These calls are called *function calls*. For more information about function calls, see “Points at Which QMF Calls the Governor” on page 241.

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, and the user sees the message in Figure 74 on page 228.

If you use the default limits for number of rows as discussed in Activating the Default Limits, the IBM-supplied governor exit routine also displays a warning before canceling the activity, as shown in Figure 73 on page 228. 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 and the time processing a command. If you want to define resource limits other than the number of rows or real time allowed, 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 238 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

Controlling QMF Resources Using a Governor Exit Routine

governor prompts a user in GROUP1 when the number of rows reaches 10000, and cancels the user's activity when the number of rows reaches 15000. The governor also prompts a user 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 in Figure 81 to set the number of rows at which the user is warned of the approaching resource limit.

If you don't 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 81. Activating prompting for row limit

2. Run the query in Figure 82 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 82. Activating cancellation of activities when user reaches row limit

3. Run the query in Figure 83 to set the real time that elapses before the user is warned of the approaching resource limit.

If you don't 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 83. Activating prompting for time limit

4. Run the query in Figure 84 to set the 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 84. Activating cancelation of activities when user reaches time limit

5. Run the query in Figure 85 on page 235 to set the time between intervals when the governor checks the user's activity.

Controlling QMF Resources Using a Governor Exit Routine

```
INSERT INTO Q.RESOURCE_VIEW (RESOURCE_GROUP,RESOURCE_OPTION,INTVAL)
VALUES('GROUP1','TIMECHECK',600)
```

Figure 85. Activating time interval check

6. Run the query shown in Figure 86 to turn on governing for the GROUP1 resource group.

```
INSERT INTO Q.RESOURCE_VIEW (RESOURCE_GROUP,RESOURCE_OPTION,INTVAL)
VALUES('GROUP1','SCOPE',0)
```

Figure 86. Turning on the governor for a particular 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.

7. Run a query similar to the one in Figure 87 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 87. Updating a user's resource group

If you're using an NLF: Use a similar query to update a user's profile in an NLF environment, but use a TRANSLATION value from Table 5 on page 19.

8. Instruct the user whose profile you updated to reconnect to the database to initialize the resource values. For example, user JONES, who has the password MYPW, can 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.

See Table 13 on page 95 for how to grant a user authority to connect to the database. Users who do not have DB2 for VM CONNECT authority can end the current QMF session and begin another to activate the new resource group.

Controlling QMF Resources Using a Governor Exit Routine

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 35 on page 237.

Figure 88 shows an example of SQL statements you might use to create a table called MY_RESOURCES. Substitute your own table, column, and dbspace 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 88 and Figure 89.

```
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 88. 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 89 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 89. Redefining the Q.RESOURCE_VIEW

VM Timer Considerations

If you plan to use the governor timer options (TIMEPROMPT, TIMELIMIT, or TIMECHECK), you should be aware that the TIME option in VM is implemented in QMF by using the STIMER macro as simulated by VM. How

Controlling QMF Resources Using a Governor Exit Routine

the QMF timer operates depends on the release of VM you are using and how your system environment options are set. The basic design of the QMF governor in VM requires the use of elapsed time. Elapsed time is derived from the amount of virtual CPU run time and virtual wait time.

The time routine is executed when a timer interrupt occurs. The time duration is specified by the TIMECHECK value. When a TIMEPROMPT value is specified and that value has expired, the time routine issues a TPUT to send instructions to the user followed by a TGET to receive the user's response. Because the timer exit can gain control at any point during QMF execution or, in some cases, during processes called by QMF, the state of the environment is unknown.

If you are unsure of the operation or behavior of your environment, use the ROWPROMPT option instead of the TIMEPROMPT option. The following is a list of known restrictions which affect the use and operation of the TIMEPROMPT option:

- Do not use the QMF attention handler in conjunction with the TIMEPROMPT option.

If you interrupt QMF using the attention key, the timer routine might acquire control of the system while the attention handler is waiting for instructions from the user. If this happens, the existing attention TGET is replaced by the TGET of the timer routine. This in turn usually generates a CMS ABEND.

- When using the TIMEPROMPT option, specify a value of at least five minutes (300 seconds).

QMF might return status information to your terminal using ISPF or GDDM. The results of using the TIMEPROMPT option during ISPF or GDDM execution are unpredictable. Specifying a value of at least 300 seconds for the TIMEPROMPT option reduces the risk of these incompatibility problems.

Table 35. 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.

Controlling QMF Resources Using a Governor Exit Routine

Table 35. Structure of the Q.RESOURCE_TABLE table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
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.

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, 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 241 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 244 explains this step.
3. Establish addressability to the QMF cancellation service to cancel activities. “Canceling User Activity” on page 259 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 260 explains this step.

Controlling QMF Resources Using a Governor Exit Routine

5. Assemble and generate your governor exit routine, whether you modified the IBM-supplied governor exit routine or wrote your own. “Assembling and Generating Your Governor Exit Routine” on page 261 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 36 shows these components, whose names vary according to which language you installed (English or an NLF). Replace the *n* symbol in the component names in Table 36 with the NLID (from Table 5 on page 19) that matches the NLF you’re using.

Table 36. English (base product) and NLF components for the IBM-supplied governor exit routine

Member Name	Library	Function
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 control block.
DXEUnGV2	DSQUSERE MACLIB	Contains text and related definitions for the governor exit routine prompts and cancellation message.

You can find these members in the libraries as shown in the table.

If you’re using an NLF: You can govern resources in an NLF session as well as an English QMF session, by using different versions of the member DSQUnGV2 for each language environment. For example, if you have both English and German QMF installed, use the phase DSQUEGV2 for English and the phase DSQUDGV2 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.

Controlling QMF Resources Using a Governor Exit Routine

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 231.

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 241.

Figure 90 shows the program structure of a governor exit routine:

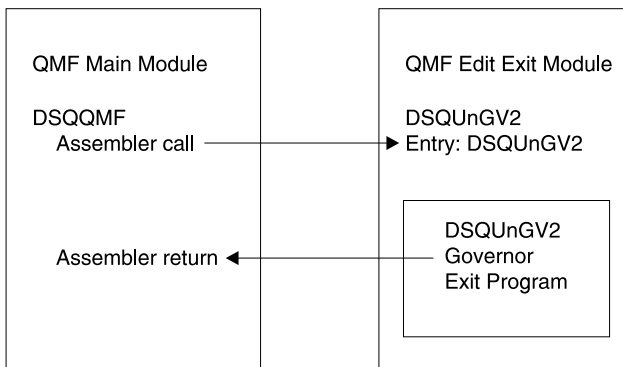


Figure 90. CMS processing that interacts QMF with the governor exit

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.

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.

- 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.

The governor exit routine is also called just before the session ends, when it can perform whatever is needed to discontinue its activities for the user's session. For example, it can release virtual storage.

- 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.

The following QMF commands always force database activity:

Controlling QMF Resources Using a Governor Exit Routine

- 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 QUERY command (for all types of queries)
- The SAVE DATA command (which forces an implicit CREATE TABLE query)
- Scrolling commands that result in retrieving 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 leads 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 then perform the cleanup necessary before theabend 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. The fields of this control block are explained in Table 37 on page 245.

Controlling QMF Resources Using a Governor Exit Routine

Each type of function call has a specific value for the GOVFUNCT field. These values are shown in Figure 92 on page 244.

What Happens Upon Entry to the Governor Exit Routine

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 for the parameter list. Figure 91 shows the contents of Register 1 on a call to the governor.

The parameter list contains two addresses: The address of the DXEXCBA control block and the address of the DXEGOVA control block.

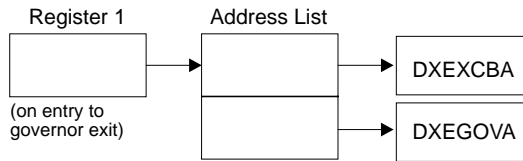


Figure 91. Contents of Register 1 on a call to the governor exit routine

- Register 13 contains the address of the QMF save area
- Register 14 contains the return address
- Register 15 contains the address of the entry point (DSQUnGV2)

Establishing Addressability for Function Calls

Because QMF always branches to an entry point named DSQUnGV2 when it calls the governor, you can't use this entry point 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 92 on page 244. (If you need more information about the activity that occurs at each function call, see "Points at Which QMF Calls the Governor" on page 241.)

Controlling QMF Resources Using a Governor Exit Routine

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 92. Character and numeric values for the GOVFNCT 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 GOVFNCT with binary numbers by using a branch table. QMF uses the branch table to find the addresses to branch to for each type of function call.

Figure 93 shows an example of some code that identifies branch addresses for the IBM-supplied governor.

XR	R07,R07	ZERO REGISTER 7
IC	R07,GOVFNCT	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
	. . .	
	. . .	
	. . .	

Figure 93. Identifying the type of function call and branching to the appropriate address

Because the governor program runs on the same program 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.

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:

Controlling QMF Resources Using a Governor Exit Routine

- “How a Governor Exit Routine Controls Resources” on page 230
- “Defining Your Own Resource Limits” on page 233

QMF passes resource control information using two control blocks named DXEGOVA and DXEXCBA. These are shown in Figure 94 on page 248 and Figure 96 on page 256. 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 DSQUSERE 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 231 for more information on how this view is used.

Table 37 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 VM/ESA environment because it is used in only OS/390 or VSE/ESA operating environments.

Table 37. 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. The code to use this field appears in “Canceling User Activity” on page 259.

Controlling QMF Resources Using a Governor Exit Routine

Table 37. Fields of the DXEGOVA interface control block to the governor (continued)

Field	Data Type	Purpose
GOVFNCT	XL1	<p>Indicates the type of function call. Possible values are:</p> <ul style="list-style-type: none">• 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 (exit for abnormal ending) <p>Code to use this field appears in “Establishing Addressability for Function Calls” on page 243.</p>
GOVGROUP	CL16	<p>Contains the name of the user’s resource group. This value can change after a CONNECT command, when QMF initializes the Q.RESOURCE_TABLE and the Q.PROFILES table. For more on resource groups, read “How the Governor Knows What the Resource Limits Are” on page 231.</p>
GOVNAME	CL8	<p>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.</p>
GOVOROWS	F	<p>Contains the number of rows for the user’s resource group in the resource control table. This value can change after a CONNECT command.</p>
GOVRESCT	10XL128	<p>Contains information from the resource control table. This information is divided into 10 contiguous blocks of storage that are structured like DSECT GOVRESCT. A block contains information about one of the rows for the user’s resource group in the QMF resource control table.</p> <ul style="list-style-type: none">• 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. <p>All blocks are part of a chain, as described in “Addressing the Resource Control Table” on page 249. The value of this field does not change during a session.</p>

Controlling QMF Resources Using a Governor Exit Routine

Table 37. Fields of the DXEGOVA interface control block to the governor (continued)

Field	Data Type	Purpose
GOVRESCT	DSECT	<p>Describes the block of storage containing information on one of the user's rows of the resource control table. All such blocks are linked together in a chain discussed in "Addressing the Resource Control Table" on page 249. The following fields are within the block:</p> <p>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.</p> <p>GOVNULLI(H) Null indicator for INTVAL column.</p> <p>GOVINTVL(F) Value of INTVAL column.</p> <p>GOVNULLF(H) Null indicator for FLOATVAL column.</p> <p>GOVFLOAT(D) Value of FLOATVAL column.</p> <p>GOVNULLC(H) Null indicator for CHARVAL column.</p> <p>GOVCHLEN(H) Length of data in CHARVAL column.</p> <p>GOVCHAR(CL80) Value in CHARVAL column.</p> <p>GOVNEXTR(A) Points to the block of data for the next resource table row. Contains zero if this is the last row.</p> <p>Any null indicator in the structure is zero when its corresponding column value isn't null. If the column value is null, the indicator is not zero.</p>
GOVSQLCA	A	Address of the SQL communications area (SQLCA), which holds information about the SQL SELECT query on the resource control view (Q.RESOURCE_VIEW).
GOVSQLRC	F	Return code from the SQL SELECT query on the resource control view (Q.RESOURCE_VIEW). If it is nonzero, the query failed and no rows are passed to the governor.
GOVUSERS	CL2048	Scratchpad area, retained between session calls. QMF does not change this value.

Figure 94 on page 248 shows the structure of the DXEGOVA control block.

Controlling QMF Resources Using a Governor Exit Routine

```

***** 00001000
*      * 00002000
*      * 00003000
*      * 00004000
*      * 00005000
*      * 00006000
*      * 00007000
*      * 00008000
*      * 00009000
*      * 00010000
*      * 00011000
*      * 00012000
*      * 00013000
*      * 00014000
*      * 00015000
*      * 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 94. The DXEGOVA control block (Part 1 of 2)

Controlling QMF Resources Using a Governor Exit Routine

	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
GOVRESC	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 94. 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 GOVRESC. This DSECT is shown in the last part of Figure 94. 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 GOVRESC 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 GOVRESC DSECT in Figure 94 contains the address of the next block in the chain of resource control information. Each block in the chain has a

Controlling QMF Resources Using a Governor Exit Routine

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 95 shows a part of the code for the IBM-supplied governor that processes the blocks of resource control information. In this code, GOVRESCT 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,GOVRESCT      GET ADDRESS OF 1ST RESOURCE ROW
                USING  GOVRESCT,R05     BASE RESOURCE RECORD ENTRY
LOOK4RES DS     0H                     MAIN LOOP THRU RESOURCE ROWS
                LTR    R05,R05          ANY MORE RESOURCE TABLE ROWS?
                BZ     ENDRESST          NO, END RESOURCE INITIALIZATION
                :
                :
                L      R05,GOVNEXTR      GET ADDRESS ON NEXT RESOURCE ROW
                B      LOOK4RES          BEGIN NEXT ITERATION
ENDRESST DS     0H                     -- BRANCH HERE WHEN FINISHED READING ALL ROWS

                . . .
                . . .
                . . .
                . . .

DXEGOVA DSECT

                . . .
                . . .
                . . .

GOVRESCT DS     10XL128                -- RESOURCE CONTROL TABLE
                ORG   GOVRESCT
GOVRESCT DSECT
                . . .
                . . .
                . . .
GOVNEXTR DS     A                     -- POINTER TO NEXT RESOURCE ROW
                . . .
                . . .
                . . .

```

Figure 95. 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.

Controlling QMF Resources Using a Governor Exit Routine

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 cancelation service to cancel the command.

Table 38 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 in the VM/ESA environment because it is used in only OS/390 or VSE/ESA operating environments.

Table 38. 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.
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. This field can change on a CONNECT command.
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. This field is reset to zero before the function call for the command's termination.

Controlling QMF Resources Using a Governor Exit Routine

Table 38. Fields of the DXEXCBA interface control block to the governor (continued)

Field	Data Type	Purpose
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. You might need to take this into account for asynchronous processing. 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.
XCBDDBMG	CL1	Identifies the database manager. This value is set to 1 for DB2 for VM, and to 2 for DB2.
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. See “Specifying an Interactive or Noninteractive QMF Session (DSQSMODE)” on page 83 for more information on starting a noninteractive session.
XCBERRET	F	Contains the return code to be used in the default cancelation message. For more information on this message, see “Providing Messages for Canceled Activities” on page 260.
XCBINCI (ISPF only)	CL1	Indicates if the current command is being run through the command interface. The field is set to 1 when it is; zero when it isn't. For more information about the command interface, see <i>Developing QMF Applications</i> .
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. See “Setting Printing for Double-Byte Character Set Data (DSQSDBCS)” on page 90 for more information about this parameter.

Controlling QMF Resources Using a Governor Exit Routine

Table 38. Fields of the DXEXCBA interface control block to the governor (continued)

Field	Data Type	Purpose
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. Message logging is described in “Providing Messages for Canceled Activities” on page 260. Using the QMF trace facility is described in “Using the QMF Trace Facility” on page 300.
XCBMGTXT	CL78	Contains the text for a message. The message can be logged in the QMF trace data, displayed on the screen, or both. For more information on how this field is used, see “Providing Messages for Canceled Activities” on page 260.
XCBMSGNO (ISPF only)	CL8	Contains the message ID for an ISPF message definition. The field can be used for a message to be logged in the DSQDEBUG file, displayed on a user’s screen, or both. For more information about XCBMSGNO use, see “Providing Messages for Canceled Activities” on page 260.
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 5 on page 19.) Value does not change during a session.
XCBPANEL (ISPF only)	CL8	Contains the panel ID for the message help panel for a cancelation message. For more information about XCBPANEL use, see “Providing Messages for Canceled Activities” on page 260.
XCBPLAN	CL8	Contains the application plan ID for QMF. This value does not change during a session.
XCBQCE	F	Contains the decimal equivalent of the value of the SQLDERRD(4) field in the SQLCA returned from DB2 for VM. 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 SQL V1, 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 V7R1.0, and does not change during a session.

Controlling QMF Resources Using a Governor Exit Routine

Table 38. Fields of the DXEXCBA interface control block to the governor (continued)

Field	Data Type	Purpose
XCBQRYP	A	<p>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.</p> <p>This field contains information only when data retrieval is requested through one of the commands in the following list; no information is provided for queries on DB2 for VM system tables or QMF control tables.</p> <p>DISPLAY TABLE EDIT TABLE ERASE TABLE EXPORT TABLE IMPORT TABLE PRINT TABLE RUN QUERY SAVE DATA</p>
XCBREFR	CL1	<p>Indicates whether QMF refreshes the screen after returning from the governor; 1 indicates a refresh; 0 indicates no refresh.</p> <p>If your governor displays any screen information, set this field to 1.</p>
XCBRELN	CL2	<p>Identifies the QMF release level. For QMF VM/ESA V7R1, this is 12. The value does not change during a session.</p>
XCBRGRP	CL16	<p>Contains the name of the user's resource group. This value can change after a CONNECT command.</p>
XCBROWSF	F	<p>Reflects the number of rows retrieved into the data object. Initially zero, this field changes value whenever more rows are retrieved. All data retrieval is counted whether data is retrieved from the database or imported from CMS files.</p> <p>QMF does not reset this field, but the governor can. For example, if your governor exit routine monitors the number of database rows retrieved, you can set this field to zero on the function call for the end of the command that began the data retrieval.</p>

Controlling QMF Resources Using a Governor Exit Routine

Table 38. Fields of the DXEXCBA interface control block to the governor (continued)

Field	Data Type	Purpose
XCBSYST	CL1	Identifies the current operating system. The value does not change during a session, and is usually set to 4, indicating CMS. Possible values are: 1 for CMS (VM/SP) 3 for TSO (MVS/XA™ or MVS/ESA™) 4 for CMS (VM/XA or VM/ESA) 5 for CICS (VSE/ESA™, MVS/ESA, or MVS/XA) For information on why the other values here can be valid for QMF VM/ESA V7, see “Providing the Correct Profile for the User’s Operating Environment” on page 104.
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). Using this value in a governor is discussed in “Providing Messages for Canceled Activities” on page 260. 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 300.
XCBUSER	CL8	Contains the users VM logon ID.
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.

Figure 96 on page 256 shows the structure of the DXEXCBA control block. Table 38 on page 251 provides more information on each field in the control block.

Controlling QMF Resources Using a Governor Exit Routine

```

***** 00001000
*      * 00002000
*      * 00003000
*      CONTROL BLOCK NAME: DXEXCBA      * 00004000
*      * 00005000
*      FUNCTION:                          * 00006000
*      * 00007000
*      THIS IS THE INTERFACE CONTROL BLOCK BETWEEN QMF AND * 00008000
*      EXIT ROUTINES.                    * 00009000
*      * 00010000
*      STATUS: VERSION 7 RELEASE 1 LEVEL 0 * 00011000
*      * 00012000
*      INNER CONTROL BLOCKS: NONE        * 00013000
*      * 00014000
*      CHANGE ACTIVITY:                  * 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 (MVS 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
XCBKPARY EQU      C'1'      ----- DBCS                   00049000
      SPACE                               00050000

```

Figure 96. The DXEXCBA control block (Part 1 of 3)

Controlling QMF Resources Using a Governor Exit Routine

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
XCBIAIAC	DS	CL1	----- CURRENT INTERACT MODE	00060000
XCBIAIACY	EQU	C'1'	----- INTERACTIVE EXECUTION	00061000
XCBIAIACN	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			00090000
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
	SPACE			00097000
XCBROWSF	DS	F	----- DATA BASE ROWS FETCHED FROM SOURCE	00098000
*			----- SET BY QMF; EXIT MAY RESET	00099000
	SPACE			00100000

Figure 96. The DXEXCBA control block (Part 2 of 3)

Controlling QMF Resources Using a Governor Exit Routine

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
XCBMGTX	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 96. The DXEXCBA control block (Part 3 of 3)

&rb|;

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 2048-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 97 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.

```
LA    WORKPTR,GOVUSERS
USING WORK,WORKPTR
```

Figure 97. Establishing addressability to the governor scratchpad area

Controlling QMF Resources Using a Governor Exit Routine

In Figure 97 on page 258, 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.

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
- Asynchronous commands when a timer is active

The code for canceling either of the first two activities is contained in the source program DSQUnGV2. To have your governor call the QMF cancellation service to cancel an activity, branch to the address that appears in the DXEGOVA control block field named GOVCADDR. Figure 98 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 98. 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, 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. The tests can also be based on the user's response to a cancellation prompt. Your VM system should be running with the value for the CP TIMER set to REAL.

The timer routine is the CSECT named TIMEX in the source code for the IBM-supplied governor. The source code is the member DSQUnGV2 on the production disk.

Controlling QMF Resources Using a Governor Exit Routine

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.

Providing Messages for Canceled Activities

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.

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 messages discussed 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 file 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 don't want messages to be logged (although initialization errors are logged unless you don't 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 300.

Controlling QMF Resources Using a Governor Exit Routine

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 XCBMGTX 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 XCBMGTX. 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 XCBMGTX, and the HELP panel ID, if any, in XCBPANEL. Leaving XCBPANEL blank, in this case, leaves the user without message help.

The governor can log messages in the ISPF log file. It can do this through the ISPF LOG service discussed in *ISPF for VM Dialog Management Services and Examples*.

Messages do not appear on screen if the command is run in batch or noninteractively from a QMF application.

The IBM-supplied governor does not log messages for termination function calls.

Assembling and Generating Your Governor Exit Routine

Whether you're modifying the IBM-supplied governor exit routine or writing a routine of your own, you need to create a CMS module.

If you're 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 the H or HLASM assembler. 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

Controlling QMF Resources Using a Governor Exit Routine

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)
```

Chapter 14. Customizing a Remote Database Connection

You can customize a remote database connection to allow your users to query relational data on remote systems and build reports or charts to present the data on their local system. You can establish connections to any of the DB2 for OS/390 or DB2 for VM databases within a distributed network. (QMF users cannot connect to OS/400[®] locations.) You can establish this connection during QMF initialization or from within a QMF session. You can establish connections between *like* (for example, DB2 to DB2) and *unlike* (for example, DB2 for OS/390 to DB2 for VM) locations.

QMF enables you to access remote data through the distributed database solutions as implemented by both DB2 for OS/390 and DB2 for VM. Those solutions are based on Distributed Relational Database Architecture (DRDA)[™]. DRDA is an open set of protocols and formats enabling transparent access to local and remote data belonging to like or unlike relational database management systems (RDBMSs).

When your users are connected to a remote location, all the SQL statements within their applications are directed to that database for processing. They can access the data and QMF objects at that location's database in much the same way they access data and objects without a remote unit of work connection. QMF continues to use programs that reside at the same system in which QMF is executing. This type of distributed connection is called *remote unit of work*. (For a schematic diagram, see Figure 99 on page 264.)

Quick Start

Table 39 outlines how you can customize a remote database connection for your users.

For more information on any of the tasks listed, see the page shown at the right of the table.

Table 39. Customizing a remote database connection

To do this task:	See:
Determine the type of database connections your users need depending on which database they need to access.	Page 264
Verify the connections required for remote unit of work.	Page 266
Prepare the remote location for access by your users.	Page 267

Customizing a Remote Database Connection

Table 39. Customizing a remote database connection (continued)

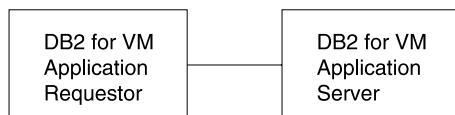
To do this task:	See:
Enable your users to access a remote database by preparing the user profiles and the database location involved.	Page 271
Enable access to your location for other DBAs to set up remote unit of work from their location to yours.	Page 275

Determining the Remote Database Connection Needed

In DB2 for VM, you can use remote unit of work connections to another DB2 for VM database or to a remote DB2 database application server.

Remote unit of work and distributed unit of work can be used together. (For a schematic diagram, see Figure 99.)

Remote unit of work



Remote unit of work and distributed unit of work

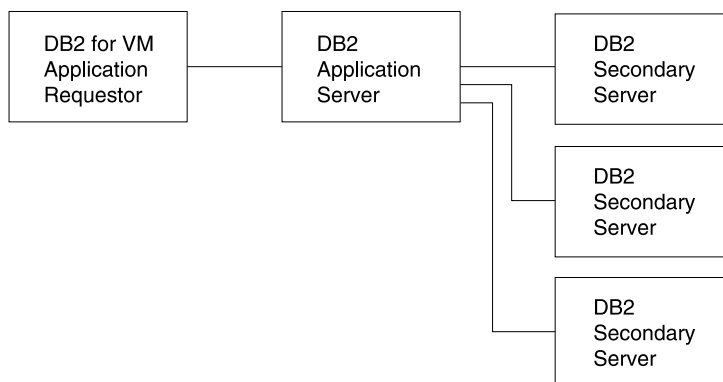


Figure 99. Distributed connection types using remote unit of work and distributed unit of work from QMF/VM

Notes:

1. The application requester is the database management system (DBMS) code that handles the QMF end of the distributed connection.

2. The application server is the DBMS code that supports requests from application requesters and in QMF is called the current location.
3. The secondary server supports requests from the application server and does not use Distributed Relational Database (DRDA) database support protocols.

See *DRDA Connectivity Guide* for more information.

Connecting with Remote Unit of Work

Using remote unit of work, you can connect to and access relational data at a remote DB2 location or a remote DB2 for VM location. The remote location is called the *server*. When connected to the server, you can access the data and QMF objects at that location as you would access the data and objects without a remote unit of work connection.

Connecting with DB2-to-DB2 Distributed Unit of Work

From a DB2 for VM database, you can connect to a DB2 database. From there, you can connect to another DB2 database and set up a DB2-to-DB2 connection using *distributed unit of work*. With distributed unit of work, the application program need not connect to a different database when it accesses data from another location. Instead, the application specifies the other location within a three-part name in a query or QMF command.

Specifying a Table or View with a Three-part Name in DB2

If you are connected to a DB2 subsystem that has distributed data support, you can specify a table or view with a three-part name. QMF remains connected to a single DB2 location, and this location sends all SQL statements that use three-part names (or aliases for them) to the DB2 location referred to in the three-part name. That location then processes the SQL statement.

Restrictions: The following restrictions apply when referring to a table or view using a three-part name:

- From a remote DB2 server, you cannot reference a local object using a three-part name.
- When DB2 for VM is the current location:
 - The location in the three-part name must match the *current location* name (the name of the application server to which the QMF session is currently connected).
 - QMF commands, prompted query, and QBE do not support three-part names.

Directing a Query Using Three-part Names

Establishing a connection to a specified location constitutes most of remote-unit-of-work support. If that connection is made, QMF functions largely as it did before remote-unit-of-work support. Consequently, three-part

Customizing a Remote Database Connection

name support is still provided. If the current location is a DB2 location that supports three-part names, SQL statements using three-part names can direct a query to yet another DB2 location.

Verifying the Connections Necessary for Remote Unit of Work

To ensure that the users can access a remote system, connections between the local and remote systems must be in place.

Checking DB2 for VM Connections

To connect from DB2 for VM to a remote system, you need to ensure that the remote systems have been defined (their LU names are registered).

When an DB2 for VM application requests data from a remote system, DB2 for VM searches the VM Communications Directory to find information about the remote system. You need to check that the following items are available to DB2 for VM:

- Gateway name—local logical unit (LU) name
- Remote LU name
- Remote transaction program name (TPN)
- Conversation security level required by the application server
- User ID identifying application requester at the application server
- Password authorizing application requester at the application server
- Mode name describing session characteristics to use to communicate with the application server
- RDB_NAME

For the DB2 application server to process distributed database requests, check for the following information:

- The application server is defined to the local communication subsystem.
- The necessary security is in place.

For more information about the DB2 for VM connections necessary for remote unit of work, see *DRDA Connectivity Guide*

Checking DB2 for VM Connections

To connect from DB2 to a remote system, you need to ensure that the remote systems have been defined.

When a DB2 application requests data from a remote system, DB2 searches the communications database tables to find information about the remote system. You need to check that the following items are available to DB2:

- Logical unit (LU) name and transaction program name (TPN)

VTAM® must contain the LU name for each server

- Network security information required by the remote site
- Session limits and mode names used to communicate with the remote site

For the DB2 application server to process distributed database requests, check for the following information:

- The application server is defined to the local communication subsystem
- Each potential secondary server destination is defined
- The necessary security is in place

For more information about the DB2 connections necessary for remote unit of work, see *DRDA Connectivity Guide*

Preparing a Non-DB2 for VM Location for Access by QMF VM Users

For users to access the remote location, you must:

- Create command synonym tables
- Prepare QMF to support the DPRE command
- Prepare QMF to support other commands
- Create function key tables, if necessary
- Update QMF governor control tables, if necessary
- Install the National Language feature in the QMF server, if necessary

Tip on naming conventions: Develop consistent naming conventions for the objects stored in a location. Your users then know where any particular object is located. If you establish and use a naming convention as described, you and your users can easily:

- List all objects of the same application at a given location
- List all objects of the same application in all locations

Creating Command Synonym Tables

You need to create command synonym tables in the remote locations to provide your users with commands that work remotely. The commands operate in the environment at which the users are logged on; therefore, users logged on in VM and connected to a remote DB2 database cannot use QMF commands that are defined as CMS commands.

You can create a Q.COMMAND_SYN_CMS table in the non-OS/390 location, to be used when your CMS users are connected to that location.

You create these tables as you do any other command synonym table. Be sure to include the name of the synonym table in the Q.PROFILES table. For more information about using the Q.PROFILES table, see “Reading the Q.PROFILES Table” on page 100.

Customizing a Remote Database Connection

Sample Remote Server Command Synonym Table for the CMS Environment

If you have QMF installed in a workstation database server, this synonym table is provided for you. If QMF is installed in a DB2 for VM server, the synonym table is not provided.

The statements shown in Figure 100 are examples of how to define a command synonym table in a non-OS/390 remote data base server for the CMS environment.

```
CREATE TABLE Q.COMMAND_SYN_CMS
  ("VERB"          CHAR(18)          NOT NULL,
   "OBJECT"        VARCHAR(31),
   "SYNONYM_DEFINITION" VARCHAR(254) NOT NULL,
   "REMARKS"       VARCHAR(254))
  IN DSQDBCTL.DSQTSSYN;
COMMENT ON TABLE Q.COMMAND_SYN_CMS IS
  'QMF CMS COMMAND SYNONYM TABLE';
COMMENT ON COLUMN Q.COMMAND_SYN_CMS.VERB IS
  'NAME OF THE VERB';
COMMENT ON COLUMN Q.COMMAND_SYN_CMS.OBJECT IS
  'NAME OF THE OBJECT';
COMMENT ON COLUMN Q.COMMAND_SYN_CMS.SYNONYM_DEFINITION IS
  'DEFINITION OF SYNONYM';
COMMENT ON COLUMN Q.COMMAND_SYN_CMS.REMARKS IS
  'OPTIONAL COMMENTS ABOUT SYNONYM';

CREATE UNIQUE INDEX Q.COMMAND_SYN_CMSX ON Q.COMMAND_SYN_CMS
  ("VERB" ASC , "OBJECT" ASC)
  USING VCAT QMFCAT
  SUBPAGES 8
  CLOSE NO;

GRANT SELECT ON Q.COMMAND_SYN_CMS TO PUBLIC;

INSERT INTO Q.COMMAND_SYN_CMS
  VALUES('DPRE',NULL,'RUÑ Q.DSQAER2P',
         'QMF DISPLAY PRINTED REPORT APPLICATION');
INSERT INTO Q.COMMAND_SYN_CMS
  VALUES('ISPF',NULL,'CMS DSQAEZ2P',
         'QMF ISPF BRIDGE APPLICATION');
INSERT INTO Q.COMMAND_SYN_CMS
  VALUES('BATCH',NULL,'CMS DSQABB21 DXYEABVP',
         'QMF BATCH APPLICATION');
INSERT INTO Q.COMMAND_SYN_CMS
  VALUES('IRM',NULL,'CMS DSQAEI1P P('&ALL')',
         'QMF IRM BRIDGE APPLICATION');
INSERT INTO Q.COMMAND_SYN_CMS
  VALUES('LAYOUT',NULL,'CMS DSQAELOA',
         'QMF LAYOUT APPLICATION');
```

Figure 100. Sample command synonym table

Preparing QMF to Support the DPRE Command

To allow remote users to issue the DPRE command at a remote location, you must copy the OS/390 procedure to the VM system or the VM procedure to the OS/390 system. The DPRE command procedure for CMS VM is Q.DSQAER2P and for TSO OS/390 is DSQAER1P.

To copy the procedures:

1. Connect to one location.
2. Display the procedure.
3. Connect to the other location.
4. Save the procedure.

Preparing QMF to Support Other Commands

You must have a command synonym table at your current location to define QMF Command Synonyms for the commands that use objects there. The following types of commands use objects at your current location:

- Commands that refer to QMF objects (CONVERT QUERY)
- Commands that read information from tables (DRAW)
- Commands that modify tables (EDIT TABLE)
- Commands that list tables you are authorized to use at that location (LIST TABLES)
- Commands that display a list (LIST TABLES)

Some commands use programs or files from the location in which QMF is executing, requiring a command synonym table at a remote location. The type of commands that run at the remote location are:

- System-specific
- IMPORT/EXPORT commands

Because of this, your users need a command synonym table for each location, and you must add all the commands (and the objects referenced by them) from the Q.COMMAND_SYNONYMS table to the database where the commands run.

Creating Function Key Tables

If you have customized function keys for your local users, you must copy the function key tables to the remote location.

To provide the tables:

1. Display the table.
2. Connect to the other database.
3. Type the SAVE DATA command.
4. Create any indexes.

Customizing a Remote Database Connection

5. Grant SELECT authority.

Updating QMF Governor Control Tables

If necessary, update QMF governor control tables at the remote location. The governor limitations at the remote location are the limits used during remote work, because those are the resources being accessed.

Installing the National Language Feature in the QMF Server

If you're using an NLF, you also add the National Language feature to the QMF remote location, because it must be at both the requester and the server. For the users to access the National Language feature, you must:

1. Prepare Q.PROFILES for NLF.
2. Create a command synonym table for NLF.

Code Page Support

Both DB2 and DB2 for VM can handle character translation between application requesters and servers that are on different systems and use different code pages.

To process character strings coming from an unlike database, you must set up the CCSID conversion rules properly for both the application requester and the application server. Define the proper CCSID translation pair at the server for the application server to recognize the character string sent from the application requester.

The decision to translate depends on whether an entire string (for example, data from a CHAR or VARCHAR column) must be translated.

For CHAR or VARCHAR columns:

- If the column is defined FOR BIT DATA, then its contents are not translated.
- If the column is defined FOR SBCS DATA or FOR MIXED DATA, then its contents are translated.

For more information on FOR BIT DATA, FOR SBCS DATA, and FOR MIXED DATA parameters, see *DB2 REXX SQL for VM/ESA Reference*

Restricting Use of the APPLDATA Column

QMF uses the VARCHAR column APPLDATA (of Q.OBJECT_DATA) to hold the definitions of its procedures, queries, and forms. The definitions of QMF objects contain some data that must not be translated. Therefore, the APPLDATA column is not classified as containing translatable character data, and is defined as FOR BIT DATA.

Avoiding Use of Some Special Characters

Between systems on which QMF can execute and access data, the number of characters that require translation depends on the code pages used at the

application requester and server locations. The list of characters that require translation includes the not sign (~), and the vertical bar (|), as well as any other characters that require translation between the code pages of your requester and server locations.

Because the characters in QMF objects do not get translated, if code pages in the application requester and server are different, avoid using ~ and |.

Enabling Your Users to Access a Remote Database

QMF V7R1 supports remote unit of work access between different databases. You can go between different DB2 for VM databases, between different DB2 databases, or between DB2 for VM and DB2 for OS/390 databases.

Updating a User's Profile

You need to update users' Q.PROFILES tables if they need access to a remote workstation. Update the Q.PROFILES table as explained in "Creating User Profiles to Enable User Access to QMF" on page 97.

Some profile values are attributes of your QMF session (query type and the LANGUAGE parameter, for example), others (SPACE parameter, for example) are related to the current location.

You can set up different rows in a single profile table for a specific user (for access from CMS, TSO, or CICS). You can do this with the ENVIRONMENT column to give values that apply to the QMF operating environment, with values that must be unique for the location on which the profile is stored.

Specifying Access for Current SQL Authorization ID

Your users' CURRENT SQLID is not in effect after a connection to a different location. So, if they need to use the same CURRENT SQLID with multiple DB2 application servers from a single QMF session, they might have to reset the CURRENT SQLID after they connect to each server. For more information, see *QMF Reference*.

Connecting to the Local Database

In CMS, QMF connects to DB2 for VM when you run the SQLINIT EXEC to specify which database to access when QMF issues SQL statements. The EXEC runs before you invoke QMF, and 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 EXEC need not be rerun.

Connecting to the Remote Database

You can provide your users with different methods of connecting to a remote location from a QMF session. You can set up one or more of the following methods:

- Using the program parameter DSQSDBNM

Customizing a Remote Database Connection

Use this parameter to connect to a remote location when you initialize a QMF session. For more information on the DSQSDBNM parameter, “Specifying the Location to Connect to When Starting QMF (DSQSDBNM)” on page 82.

- Using the QMF CONNECT command

Use this command to connect to a remote location during the QMF session. This command lets you connect to a different location within your distributed network during a QMF session.

You can issue the command from:

- The callable or command interface
- The command line
- Within a procedure (linear or with logic)

For more information on the command, the command parameters, and a list of considerations for using the command, see *QMF Reference*. For more information on procedures and the callable or command interface, see *Developing QMF Applications*.

- You can also create a procedure to establish the connection and add a command to the Q.SYNONYMS table to run the procedure. Your users can then enter the command to connect to a remote database.

Specifying a Location Name

QMF uses SQL to access a relational database. In remote unit of work, the application requester takes a CONNECT request and establishes a connection with the remote database management system. In distributed unit of work, the application requester “receives” the SQL request and routes it to the appropriate distributed unit of work server.

QMF uses the term *location name* to denote the DBMS to which it is connected. You can use the location name to connect to a database system or to qualify a table name. For example, an SQL table named SAN_JOSE.JONES.TABLE5 is managed by the database management system (DBMS), whose location name is *SAN_JOSE*.

In DB2

The *location name* refers to an entire subsystem. Servers that are accessible to a DB2 subsystem are defined in the communication database.

If you’re using three-part names: When you are using both remote unit of work and distributed unit of work, the locations you can access with three-part names are accessible to the current application server, which must be a DB2 location.

In DB2 for VM

The location name refers to an entire DB2 for VM database machine and is cataloged in the CMS communications directory.

Where Data Must be Located for User Access

Commands and queries that access data, such as `DISPLAY TABLE tablename`, are directed to the current location, unless the current location is DB2 and *tablename* is a three-part name (or an alias for that name) that refers to a DB2 subsystem other than the current location.

Working with QMF Objects

QMF objects (queries, procedures, and forms) that are retrieved from or stored into the database must reside at the current location; that is, the location you are connected to. This is identical to QMF without remote-unit-of-work support: objects reside at the same location as tables that are accessed without three-part names.

To ensure that, when you save data, procedures, queries, and forms at the current location, you have sufficient database resource, do the following at that location: Routinely monitor your default tablespace (if you connect to a DB2 location) or default dbspace (if you connect to an DB2 for VM location), and the QMF object tables (Q.OBJECT_DIRECTORY, Q.OBJECT_DATA, and Q.OBJECT_REMARKS).

Some QMF objects stored in the database can refer to programs that QMF must invoke. These programs include:

- User edit routines
- REXX programs in support of procedures with logic and report calculations
- Local date and time routines invoked in support of local date and time edit codes
- QMF procedures that contain CMS, CICS, or TSO commands
These procedures must be written to ensure that no attempt is made to execute CMS commands in an TSO environment or TSO commands in a CMS environment.
- An EXEC or CLIST that causes a program to execute QMF commands through the callable or command interface

These programs must reside at the same system in which QMF is executing (the system that you log on to), because these programs can contain operating system commands that cannot be run successfully or with the expected results by that system. Consequently, that system can be different from the system in which the database (and hence the QMF objects) resides. For an example of this, see *Using QMF* and *Developing QMF Applications*.

Customizing a Remote Database Connection

You might want to use QMF objects when the application requester and application server are on systems with different code pages. For important restrictions, see “Code Page Support” on page 270.

Working With Tables

You cannot use a three-part name in data definition statements. However, if you first connect to a location with remote unit of work, you can issue data definition statements such as CREATE and GRANT at that location. You can grant privileges on a table that resides at the current server to users at other locations by using the GRANT clause PUBLIC AT ALL LOCATIONS.

Preventing SQL Errors

SQL errors generally involve the difference between the SQL supported by the like DBMS for your operating environment and the SQL you need to use when connected to an unlike DBMS. To avoid such errors, remember to use the SQL supported by the application server.

For example, if QMF is executing in OS/390 and your current location is DB2 for VM, the syntax you use in an SQL query must be the syntax supported by DB2 for VM. For example, using the 'IN' clause for the CREATE TABLE command, the following (DB2 for VM) syntax is acceptable:

```
CREATE TABLE ... IN DSP3
```

The corresponding syntax in DB2 for VM is unacceptable:

```
CREATE TABLE ... IN DATABASE DSQ3
```

The SQL statement completion information (the SQLCODE) contains the information returned by the current location. If you are using QMF on OS/390, because you might be accustomed to the SQL syntax and information returned by DB2.

If you want QMF query portability, use SQL syntax supported by SAA. This allows the greatest degree of portability between DRDA application implementations.

The following database management issues can affect QMF:

- Some application requester environments can limit the SQL statements available to the application. For example, a DB2 application requester running under CICS is not allowed to update resources at a remote application server.
- An application program must be sensitive to where tables, views, and QMF objects reside within the network, because remote unit of work is limited to a single database management system within a single unit of work.

Restriction: You cannot issue a join for two tables stored at different locations.

Customizing a Remote Database Connection

- The user edit routines and governor exit modules residing at the location where QMF is operating are used for your QMF sessions.
- After connecting to a location, the profile, resource control table, synonyms, and function keys are initialized to the values at that location.

Translating User IDs

Because the user ID of a database user might be translated to another user ID when the user connects to another location, the database administrator might need to define a translated user ID.

Translating Names

When you connect to a location (DB2 for OS/390 or VM), your primary user ID might be translated. To understand how translation is set up, refer to your DBMS manuals or contact your database administrator.

Why Name Translation Might be Necessary: Your one-to-eight character user ID must be unique within a particular operating system, but might not be unique throughout the Systems Network Architecture (SNA) network.

To eliminate naming conflicts, distributed database systems support the following *name translation* schemes:

Outbound name translation

Allows the application requester to translate the end user's name before sending it to the destination in the SNA network.

Inbound name translation

Allows the application server to translate the end user's name it receives from its SNA partner.

Deleting QMF Users from Each Remote QMF Location

After deleting a user's objects from the local QMF, you need to delete the user's objects from each remote location accessed by that user. If you are the QMF administrator for the other remote locations, you can delete the remote objects in the same way you deleted the local objects. If you are not the QMF administrator for the remote locations, you need to request the other administrator to delete the objects for the user ID.

Enabling Administrator Access to Your Location

If there are different QMF administrators for each QMF location, you must enable some access to your location for other administrators, so they can set up remote unit of work from their location.

You must:

- Grant them select authority on QMF requester control tables

Customizing a Remote Database Connection

- Install the governor routine into the QMF requester to prevent remote users from using all your resources

Chapter 15. Customizing the Batch Processing Program

As a QMF administrator, you might need to advise and assist batch mode users. You might also want to run your own procedures in batch mode. This chapter describes how you can use the QMF batch mode.

To enable your users to use batch mode, you need to 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 might 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 most other QMF commands. For more information about writing batch procedures, see *Using QMF*

QMF also supplies the QMF BATCH application to simplify running batch jobs. For more information about this application, see "Using the QMF Batch Query/Procedure Application (BATCH)" on page 282.

Quick Start

Table 40 outlines ways to customize the batch processing program.

For more information on any of the tasks listed, see the page shown at the right of the table.

Table 40. Customizing the batch processing program

To do this task:	See:
To enable your users to run in batch mode , provide them with CONNECT authority: GRANT CONNECT TO userid IDENTIFIED BY password	Page 278
To send a job to the CMS batch machine , you must use a spool and punch the batch job to the specified batch machine.	Page 279
To run a batch job on your own machine , you need to have DSQSMODE set to B when QMF starts. QMF starts, sends the job to batch, and then stops.	Page 281
To debug a batch procedure , use the L1 and L2 trace codes discussed in "Chapter 16. Troubleshooting and Problem Diagnosis" on page 289.	Page 282
To use the QMF BATCH application , you must ensure that the correct MACLIBs are available, call the application, and fill out the prompt panel.	Page 282

Customizing the Batch Processing Program

Enabling Your Users to Use Batch Mode

Assume that you have a procedure running in batch mode. In the course of its execution, it tries to run various queries and procedures and to execute QMF commands. To do this, the CMS batch machine needs the same authority the user would need to do these things interactively. The CMS batch machine can get this authority either implicitly or explicitly. It has the authority of its own user ID unless the QMF CONNECT command is being used to run the batch machine with the authority of another person's user ID.

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
```

Users who can use QMF interactively can also use it in batch mode, while those who cannot use it interactively cannot use it 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 has to 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.

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

Customizing the Batch Processing Program

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're 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 101. 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 102 on page 280 is a sample job to start QMF in batch mode. Lowercase words in it are parts of commands filled in by you.

Customizing the Batch Processing Program

```
/*
/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(DSQMF) NEWAPPL(DSQE)
*   PARM(dcssname(DSQSMODE=B,DSQSRUN=userid.myproc))
* Other forms of QMF invocation which can be used are as follows:
* DSQMF 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 6. Customizing Your Start Procedure for more
* information on these forms of QMF invocation.
```

Figure 102. Sample job to send to CMS batch machine (Part 1 of 2)

Customizing the Batch Processing Program

```
*-----*
* 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 102. Sample job to send to CMS batch machine (Part 2 of 2)

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: QMF710E. 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 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 6. Customizing Your Start Procedure.

Before doing this however, check the ISPF profile and ensure a value is entered for CONSOLE PROCESS OPTION. If you don't specify a value, then ISPF comes up with a console disposition panel prompting you for a CONSOLE PROCESS OPTION value.

Customizing the Batch Processing Program

Debugging a Procedure

You can examine any message regarding an error in the execution of a procedure by arranging to have the message printed, sent to a debug file, or sent to your reader.

“Chapter 16. Troubleshooting and Problem Diagnosis” on page 289 discusses problem diagnosis for interactive sessions. Among the topics discussed are:

- Use of the L1 and L2 trace codes to provide a record of user activities during a QMF session
- Use of the HELP command to reconstruct the message HELP panels for error messages

You can use both the trace codes and the tool to diagnose problems with a batch mode procedure. In fact, L2 tracing is the default for procedures run in batch mode. To change this requires 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 DSQDEBUG, just as it is for an interactive session. 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 contains a record for each QMF command run by the procedure (and its subordinates).

If the procedure terminates prematurely, an error message is logged to the DSQDEBUG data set. You can then use the HELP command to reconstruct the corresponding message HELP panel.

Using the QMF Batch Query/Procedure Application (BATCH)

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. This section explains what the application can do and how to use it.

If you're 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 10. Customizing QMF Commands” on page 159 for the procedure on how to assign synonyms.

MACLIBs Required

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 application skeleton library. It should be concatenated with ISPSLIB. DSQABB2P, DSQABB2J, and DSQABB2S are the members of this library for this application.

Using the Application

The application must be invoked while the 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 the 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, issue:

```
BATCH
```

which results in the display of the prompt panel in Figure 103 on page 284.

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.

Customizing the Batch Processing Program

```

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 for QMF batch execution proc
PROC arguments ==> ARGS
CONNECT PASSWORD ==>      Database password
DISK PASSWORD ==>          User 'A' disk read password
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 nodes to send report.
  USERID      ==>          NODE    ==>
               ==>          ==>

PRINT OUTPUT:  Printer ID and node for printed output.
  ID          ==>          NODE    ==>

1=Help 3=End  Enter=Process batch request

```

Figure 103. 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 41 describes the required fields.

Table 41. 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 the current query or procedure, it is saved in the database using this name. 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.
PROC arguments	Through this field, you can pass arguments to the REXX procedure specified in the OBJECT NAME field.

Table 41. 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. 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.
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

The following is a description of 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. See Table 42 for the required fields and their descriptions.

Table 42. BATCH application optional entry fields

Field	Description
Current OBJECT	If the batch query or procedure is the current object, the user enters YES in this field. The query or procedure is then saved to be run later in batch. The default value for this field is NO. If this field has no value, the user is prompted to supply a value.
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 in batch is the current form, the form is saved in the database using this name. This form is saved using CONFIRM=NO as a profile setting.
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. The default value for this field is NO.

Customizing the Batch Processing Program

Table 42. BATCH application optional entry fields (continued)

Field	Description
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, NONE 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 resulting from running the batch query or procedure, and, optionally, formatted by the specified form, YES should be specified as the value for this field. YES is the default value. 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.
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 in the future won't replace it.

Three modifiable model files are shipped with the product. They provide input to ISPF file tailoring, which in turn produces 3 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

Customizing the Batch Processing Program

- The name of the discontinuous 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.

You can also modify the batch prompt panel (DXYEABVP) so that values for fields automatically appear. To do this, prevent variables in the INIT section of the panel logic from being initialized to blanks. Then, add the variable names (the ones you prevented from being initialized to blanks) to the VGET and VPUT statements in the DSQABB22 EXEC so they look like this:

```
&SUBCOMMAND ISPEXEC VGET (VARIABLE1, VARIABLE2)
&SUBCOMMAND ISPEXEC VPUT (VARIABLE1, VARIABLE2)
```

Add a model or panel that has been modified to the appropriate MACLIB.

Chapter 16. Troubleshooting and Problem Diagnosis

Use this chapter to help solve problems your users might have while using QMF. “Troubleshooting Common Problems” on page 290 provides possible solutions to common problems, while “Determining the Problem Using Diagnosis Aids” on page 298 provides explanations of diagnosis aids that help you solve more complex problems.

Quick Start

Use the steps in Table 43 to guide you in troubleshooting common errors and diagnosing more complex problems. If you need more information on any step, see the page listed at the right.

Table 43. Troubleshooting common errors and diagnosing problems

For information on this problem:	See:
If you encounter GDDM or QMF errors while printing , it's likely you did not supply a printer name, defined the name or allocated the device incorrectly, or encountered an I/O error.	Page 292
If you see warning messages on the QMF Home panel , QMF probably encountered errors during initialization trying to read or load tables or routines.	Page 291
If the report display seems incoherent , you might need to convert raw binary data (from the table that generates the report) to character data before displaying the report.	Page 295
If you're getting slow response , you likely need to either reset the data object or increase your storage.	Page 296
If the problem is none of the above, determine which QMF or CMS diagnostic aids can help you further diagnose the problem.	Page 298
To determine the problem using QMF message support , use the message number on the help panel to determine more information about the error, such as the QMF function that issued it.	Page 298
To determine the problem using the QMF trace facility , turn the trace on by setting the DSQSDEBUG program parameter, displaying the user's profile and changing the value of the TRACE option, or using the command SET PROFILE (TRACE=value. The level of detail for the DSQSDEBUG parameter is either ALL or NONE. You can also specify a selected trace level just prior to running your trace. For example, you can specify SET T=C2D2L2.	Page 300
To determine the problem using the QMF interrupt facility , create and read an interrupt.	Page 306

Troubleshooting and Problem Diagnosis

Table 43. Troubleshooting common errors and diagnosing problems (continued)

For information on this problem:	See:
To determine the problem using reports from the Q.ERROR_LOG table, run a SELECT query on the table, specifying the SQL authorization ID that experienced the error and the approximate date and time of the error.	Page 308
To report a problem to IBM, use IBM's ServiceLink facility (if you have it) or call your IBM Support Center.	Page 309

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 whether all QMF users at your shop cannot get into QMF, and it is not just one user.

Check whether there are any messages on the terminal screen, and look up the explanation for the DSQDEBUB file message in *QMF Messages and Codes*.

If nothing appears on the screen and nothing is in DSQDEBUB, 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 for VM and GDDM during QMF initialization. If any ARI (DB2 for VM) and ADM (GDDM) error messages appear, look them up in the messages and codes book for the appropriate product.

Check that the DB2 for VM database is initialized and working properly. If all users are getting a type of ADMxxxx message upon startup, 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 for all types of failures:

```
INITIAL PGM RC = 0 | 4 - THE INITIALLY INVOKED MODULE ENDED  
WITH A RETURN CODE = 16 .
```

Users should still look on the screen for more messages, and in DSQDEBUB 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 start QMF again.

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 according to instructions in “SQL Privileges Required to Access Objects” on page 107.
- The table conforms to the proper structure:
 - The structure for command synonym tables is shown in “Chapter 10. Customizing QMF Commands” on page 159
 - The structure for function key tables is shown in Figure 56 on page 176
 - The structure for the resource control table is shown in Table 35 on page 237
- All rows of the table contain valid data. If this might be the problem, see:
 - “Entering Command Synonym Definitions into a Command Synonym Table” on page 163 for information on valid command synonym definitions
 - “Entering Your Function Key Definitions into the Table” on page 177 for information on valid function key definitions
 - Table 35 on page 237 for information on valid resource control information
- All rows in the tables are unique.

More information about the error is logged in the user’s trace data. The trace data is stored in DSQDEBUG.

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 according to the instructions in “Viewing QMF Trace Data” on page 304. Search the trace data for the numeric portion of the message number to see information about the error.

Troubleshooting and Problem Diagnosis

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. 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. See “Choosing a GDDM Nickname for Your Printer” on page 150 for information on how to create a nickname for a GDDM printer. For a list of valid device tokens for each family of GDDM printers, see *Graphical Data Display Manager Base: Programming Reference*, Volume 2.

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 *Graphical Data Display Manager Base: Programming Reference*, Volume 2.

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. See “Choosing a GDDM Nickname for Your Printer” on page 150 for examples of syntax for the ADMMNICK specification. After you fix the syntax error, reload the ADMADFC GDDM defaults module.

Handling QMF Errors During Printing

The following information helps you to solve errors that can occur during printing.

Troubleshooting and Problem Diagnosis

What happens:	What it means:	What to do:
<p>You issue the PRINT command from the command line or a function key and see the message:</p> <p>GDDM printer nickname is required for PRINTER.</p>	<p>The object you are trying to print needs a printer nickname, and no printer nickname default exists on your profile.</p>	<p>Press the Enter key again to display a prompt panel on which you can enter a printer nickname and other print parameters. You can set a printer nickname default on your profile to avoid being prompted.</p>
<p>After using the CONNECT command, your PRINT commands result in the message described, or your printed output goes to a different printer.</p>	<p>The CONNECT command replaces your own profile values with those of the user you connected to.</p>	<p>After connecting, remember to enter:</p> <pre>SET PROFILE (PRINTER=prtname</pre> <p>from the command line to establish your printer name as the default.</p>
<p>You issue several PRINT commands but find that only the last object is printed.</p>	<p>Your output file is not defined as DISP MOD, so each PRINT operation reopens the file and overwrites the previous contents.</p>	<p>Change the disposition of your output file to DISP MOD. For example:</p> <pre>FILEDEF DSQPRINT DISK DSQP FILE A (LRECL 133 FECFM F PERM DISP MOD</pre>
<p>You print a QMF object and see unexpected control characters in the printed output or file.</p>	<p>The device token or PROCOPT that you are using do not match the device on which you are actually printing.</p>	<p>Supply the correct device token, or reduce control characters to a minimum by one of these techniques:</p> <ul style="list-style-type: none"> • For a report, table, SQL or QBE query, procedure, or the profile, type <code>PRINTER=' '</code> to bypass GDDM printing. • For other objects, use the following command with no device token: <pre>PROCOPT=((PRINTCTL,0))</pre>
<p>When printing a report, table, SQL or QBE query, procedure, or profile, you see the message:</p> <p>File DSQPRINT did not open.</p>	<p>No printer name default exists on your profile, and no DSQPRINT file or system output is currently allocated.</p>	<p>DSQPRINT must be allocated before issuing a print command.</p>

Troubleshooting and Problem Diagnosis

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 don’t 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. See “Chapter 6. Customizing Your Start Procedure” on page 67 for more information about invoking QMF with the two forms of ISPSTART.

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.

You should 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 for VM 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, you should 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

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) for character data 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 *QMF Reference*.

Handling Binary Data with User-Written Edit Routines

Using the HEX function or the hex and bit edit codes can be a good way to handle binary data. For example, assume that each bit represents a data item and displays in natural language form of the value. If the fifth bit represents gender rather than hex values, a user edit code routine can cause a value of Male or Female to be displayed.

You can create your own edit code and write an edit exit routine in COBOL, PL/I, or assembler to convert the binary data to the character string you want. You might consider predefining some QMF forms that use the new edit

Troubleshooting and Problem Diagnosis

codes you create. You can then make the forms available to your users. See “Chapter 12. Creating Your Own Edit Codes for QMF Forms” on page 187 for more information.

Solving Slow 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’s 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.

Resetting the Data Object to Improve Performance

Suppose that, after viewing parts of a report, a user attempts to run an UPDATE query and waits an unusually long time for the query to return results. Because QMF finishes one database task before starting another, QMF might be attempting to complete the report (retrieve the rest of the rows into the DATA object) before running the UPDATE query. These commands cause QMF to complete the report before the command can run:

CONNECT	REFRESH (of a database object list)
DISPLAY <i>tablename</i> or QMF object (from the database)	RESET QUERY (with LANGUAGE=PROMPTED and modifying query)
DRAW <i>tablename</i>	RUN (an object in the database)
EDIT TABLE	RUN QUERY
ERASE	SAVE (data, form, procedure, or profile)
EXPORT (from the database)	
IMPORT (to the database)	
LIST	
PRINT (from the database)	

Any query run by the user that drops, creates, or modifies an DB2 for VM object forces the completion of the DATA object.

Depending on the setting of the global variable DSQEC_RESET_RPT, you might need to instruct users to issue a RESET DATA command as soon as they finish viewing the necessary parts of the report to prevent performance problems caused by trying to run these commands before QMF completes the report. See the *QMF Reference* for more information.

A user who attempts to execute certain commands during the insufficient storage condition receives an “incomplete DATA” prompt. This is caused by any command that forces QMF to “complete” the current DATA object. (A DATA object is complete when all its rows have been fetched by QMF and none have been discarded without copying to the DSQSPILL file.) To resolve this problem, the prompt offers the user two choices: Either reset the DATA object, or withdraw the command.

If QMF encounters a system error while the insufficient storage condition is in effect, it might reset the user's current DATA object.

Under some unusual processing conditions (such as running out of storage during CLOSE processing of the spill file), insufficient storage can result in abend code 001. You can also get 001 abends if you detach the disk that contains the spill file, erase the spill file, or do a FILEDEF CLEAR that affects the spill file.

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 DSQSRSTG or 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, ensure you specify an adequate amount of virtual storage for the user, using the DSQSBSTG or DSQSRSTG parameter. The parameters are discussed in "Adjusting Storage for Report Data (DSQSBSTG)" on page 73 and "Adjusting Reserved Storage Used for Report Data (DSQSRSTG)" on page 74. 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, as discussed in "Acquiring Extra Storage (DSQSPILL)" on page 75. However, using primarily virtual storage for QMF operations provides better performance. Users who rely on a spill file and have little virtual storage might not notice slow performance for large reports, but other users on the system might.

QMF performance might also slow down if QMF needs a data row (as a result of a SCROLL BACKWARD command) and that data is not in the spill file or in virtual storage, the data cursor is reopened and the row is again retrieved from the database.

Using REXX Function Packages

When using REXX in reports, your response time might slow while the report writer switches between QMF and REXX multiple times. To reduce the input and output time, you can put your REXX procedures in a REXX function package.

For more information about REXX performance, see *Procedures Language VM/REXX User's Guide* or *Procedures Language VM/REXX Reference For* additional information, you can also see the REXX Compiler.

Troubleshooting and Problem Diagnosis

Determining the Problem Using Diagnosis Aids

If you weren't able to solve your problem using the troubleshooting techniques discussed in "Troubleshooting Common Problems" on page 290, use this section to find out which QMF and CMS diagnosis aids can help you determine the problem.

Choosing the Right Diagnosis Aid for the Symptoms

Use Table 44 to help you determine which diagnosis aids you need for the symptoms you're 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, CMS status information, and QMF messages and help to determine the problem.

Table 44. Types of problems and the best diagnosis aids to use for them

	QMF Msg. No.	QMF Trace	CMS dump	CMS Status Info.	Help Message	Non- QMF Msg. No.	Error Log Output
Abend	x	x				x	x
Batch session	x	x		x		x	x
Callable interface	x	x	x	x		x	
Display panel	x	x			x	x	x
Document interface	x	x					
Error messages	x	x			x	x	x
Governor exit routine	x	x	x	x	x	x	
Incorrect output	x	x			x	x	x
Initialization	x	x		x	x	x	x
Installation	x				x	x	x
Interrupt facility	x	x					
Loop		x				x	x
Performance	x	x		x		x	x
Printing	x	x		x	x	x	x
QMF command	x	x			x	x	x
SQL error codes	x	x			x	x	x
Termination	x	x		x	x	x	x

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 5-digit number. These numbers are listed in *QMF Messages and Codes*, which provides more information about 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 45. The trace IDs are the same IDs you use to trace QMF activity for each function, as discussed in “Getting the Right Level of Detail in Your Trace Output” on page 302.

Table 45. 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 preceding messages, the following ranges of message numbers might be generated during QMF initialization:

```
DSQI0001 - DSQI0100
DSQ90000 - DSQ99999
```

Troubleshooting and Problem Diagnosis

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.
```

You can press the Help key to display more information about the message, or see *QMF Messages and Codes*

All uncommitted changes to the database are rolled back when a system problem stops QMF. Error information about the system problem is written to the trace data, which is the only source of information for a system problem that stops QMF. See “Viewing QMF Trace Data” on page 304 for instructions on viewing the trace data. 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 DSQ12002. This message maps to the SQL return code -702, which has the text:

```
NO AVAILABLE SPACE IN THE DBSPACE NUMBER 'dbspace_number' FOR INDEXES.
```

dbspace_number in the message is a placeholder, called a token, for a real database value. The token is located in the SQL communications area (SQLCA) that QMF receives from DB2 for VM.

To find the value of the token:

1. Run a QMF I2 or ALL trace using the procedures described in “Using the QMF Trace Facility”. Keep the trace data online so you can search it easily.
2. Convert the SQL return code to a hexadecimal number. For example, the SQL return code -702 is FFFFFFFD42 in hexadecimal.
3. Locate the hexadecimal number in the trace data. It is in a trace block called SQLCA.
4. Browse the right side of the trace (the eye catcher field) to gather the tokens. See *DB2 Server for VSE & VM SQL Reference* for SQLCA mappings of the tokens.
5. When you find the right token, see *DB2 Server for VM Message and Codes* to solve the problem that caused the SQL return code.

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.

Allocating the Trace File

When you are using procedures involving trace information, ensure that the trace file is allocated *before* you begin the QMF session. This is automatically 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

Using the trace facility is a 4-step process. Each of the steps in this process is discussed in the section referenced in that step.

1. Allocate a file with a file name of DSQDEBUG.

QMF Trace writes its results into this file, which can be printed or displayed. The file 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 302.

Specify a value of ALL on the DSQSDEBUG program parameter when you start QMF, as explained in “Setting the Level of Trace Detail (DSQSDEBUG)” on page 81. 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. The levels of detail are explained in “Getting the Right Level of Detail in Your Trace Output” on page 302.

Troubleshooting and Problem Diagnosis

- 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 for VM CONNECT authority can end the current QMF session and begin another to initialize the values.
- 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. For more information on how to access the DSQDEBUG data set, see "Viewing QMF Trace Data" on page 304.

4. Interpret the trace output.

You can display or print the DSQDEBUG file for analysis. For more information on interpreting trace output, see "Viewing QMF Trace Data" on page 304.

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 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 of 1 traces a function at a medium level of detail.

The value of 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 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 104 shows an example of a RUN QUERY command that failed because the user named columns that were not in the table.

```
-----
***** 94/12/15  20:39 *****
USERID: KRIS
AUTHORIZATION-ID: KRIS
COMMAND TEXT:
RUN QUERY
-----
***** 94/12/15  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 104. Using the L2 trace code to trace a user's commands and messages

Within the DSQDEBUG file, the messages appear chronologically. When commands are included, they also appear chronologically and are intermixed with the messages. The command that a message is concerned with is the command that precedes it in the 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 104 appears in *QMF Messages and Codesas*:

Column &01 is not in table &02.

The bottom half of Figure 104 shows that the value for &01 in the message is DATE and that the value for &02 is STAFF. Substitute these values into the message to help a user solve the problem.

Troubleshooting and Problem Diagnosis

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 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 DSQUTRACE module. Then assemble and link-edit the module. After the module is created, you must make it available as a phase. 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 in a CMS environment. 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 really want to display it).

To allocate (or reallocate) for printing, issue the following statements, which define DSQDEBUG as a PRINT file:

```
FILEDEF DSQDEBUG PRINTER (LRECL 121 FA PERM)
```

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.

You can also issue the following statements to allocate (or reallocate) DSQDEBUG as a sequential file that can be displayed using an online editor. The file 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 APAR number if the module has a PTF applied, as in the following trace example:

```
** DSQFQWRM: ENTERED FROM DSQFMCTL ***  
V7R1.00 01/02/30 12:00 PNxxxxx
```

APAR PNxxxxx represents 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 disk or spool area used 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 DSQSDEBUG, explained in “Setting the Level of Trace Detail (DSQSDEBUG)” on page 81, controls this tracing when QMF is started.

Abend Handling

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.

Troubleshooting and Problem Diagnosis

For the governor, DXEXCBA and DXEGOV are added to the output.

Using the QMF Interrupt Facility

To use the QMF interrupt handler, ensure 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.

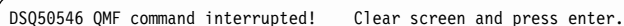
Use the QMF interrupt facility to gather information about a problem. Using the interrupt facility, you can produce an abend dump, or cause trace information to be displayed or written into the DSQDEBUG data set.

You use the interrupt facility under the user ID of the user whose problem you are diagnosing. First, however, you must recreate the problem, 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 105. 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 message shown in Figure 106 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 106. 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 in Figure 107:

```
-- 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 107. 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 107 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.

Troubleshooting and Problem Diagnosis

- 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 let you enter CP commands, for example, to initiate a VM dump.
- 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.

Since the STAX is removed, it causes QMF to generate an error during a termination when it tries to remove the STAX exit that no longer exists.

- 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 46.

Table 46. 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 yyyyymmdd.
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.

Table 46. Structure of the Q.ERROR_LOG table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
USERID	CHAR	8	no	The user ID of the VM logon 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 108 shows the format of the query.

Be sure to use valid formats for the date and times you supply. These formats

```
SELECT TIMESTAMP, MSG_NO, MSGTEXT
FROM Q.ERROR_LOG
WHERE USERID = 'terminal_id'
AND DATESTAMP = 'date'__yyymmdd
AND TIMESTAMP BETWEEN 'time1' AND 'time2'
ORDER BY TIMESTAMP, MSG_NO, MSGTEXT
```

Figure 108. Querying the error log for problem information

are shown in Table 46 on page 308.

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 many reported 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" on page 310

Troubleshooting and Problem Diagnosis

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 312.

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 VM/ESA 7 begins with the component ID 566872101 and a release number (shown in Table 47) that matches the QMF national language environment in which you experienced the problem.

Table 47. Release numbers for QMF base product and NLFs

NLF	ID
Brazilian Portuguese	79B
Danish	795
English	790
French	796
German	797
Italian	798
Japanese	799
Korean	79A
Simplified Chinese	793
Spanish	79C
Swedish	79D
Swiss French	79E
Swiss German	79F
Canadian French	79G
Uppercase English	791

The flowchart in Figure 109 on page 311 shows how to develop your search words as you determine each characteristic of the problem.

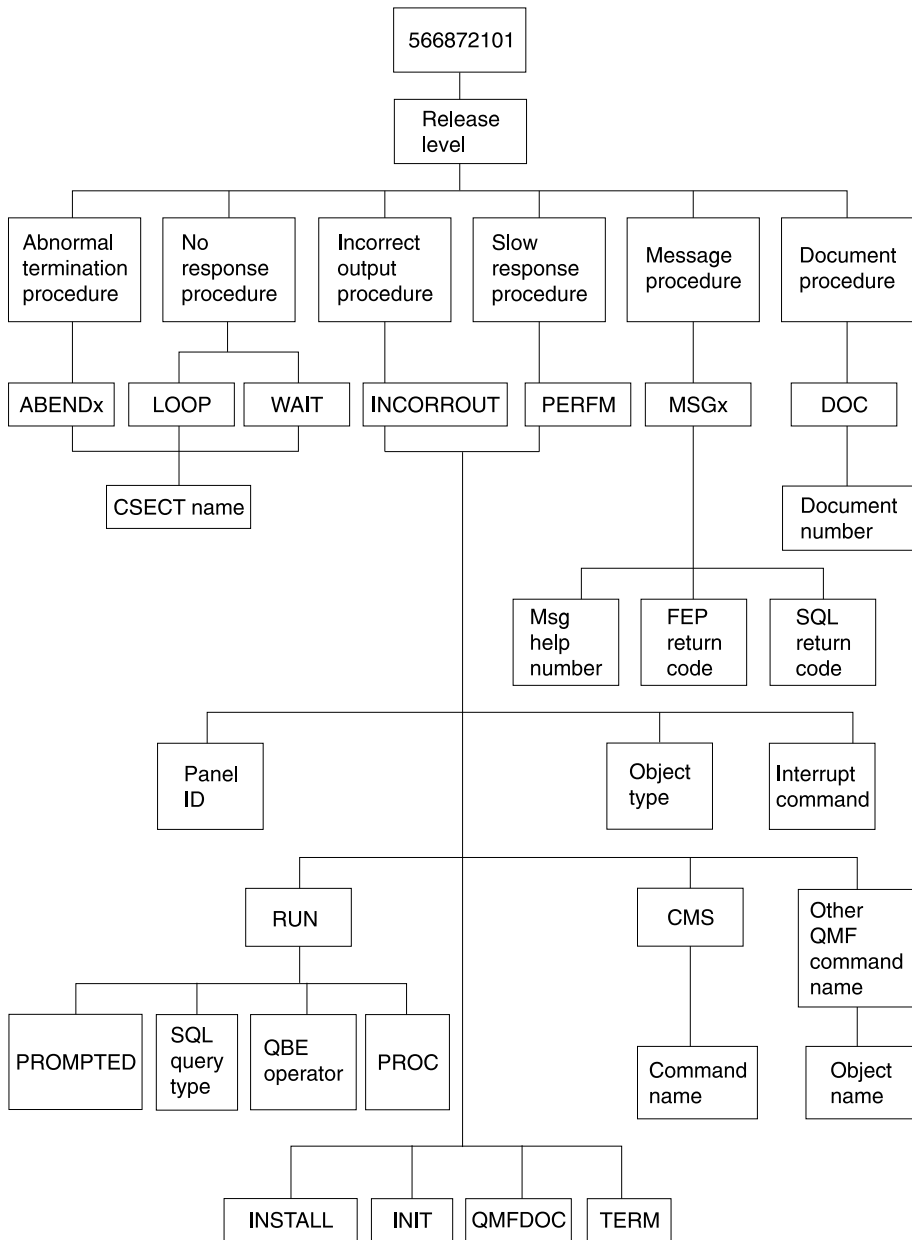


Figure 109. Chart of keyword types. Move from the top to the bottom of this chart to determine your keywords.

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:

Troubleshooting and Problem Diagnosis

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 300 for more information on how to use the QMF trace facility.

For more information on searching the SSF for known QMF problems, see *ServiceLink User's Guide*

Working with Your IBM Support Center

If you're 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 more information 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 300. The IBM representative might also need documentation produced by other diagnosis aids shown in Table 44 on page 298. This documentation can help the representative recreate the problem.

Part 3. Appendixes

Appendix A. Installation Checklists

This appendix is provided to assist you in QMF and QMF NLF installations. Checklists provide the major installation steps to be completed and the applicable file name or EXEC of each step.

QMF Installation Checklist

Table 48. Checklist for QMF V7R1

Step	Description	File or EXEC Name
	Read and follow installation steps in program directory	
1	Create QMF installation control file	DSQ2ECTL
2	Create DB2 for VM DBSPACES 10 (initial installation) 1 (migrating from QMF V2) 0 (migrating from QMF V3R1 or higher)	DSQ2DBSC
3	Run QMF installation EXEC	DSQ2EINS
4	Start QMF	DSQ2EINV
5	Run IVP for interactive mode	DSQ2EIVP
6	Install the QMF sample queries and procedures	DSQ2ESQD DSQ2ESQI
7	Run IVP for batch mode (optional)	DSQ2EBAT
8	Delete QMF 2.4 or higher (optional)	DSQ2BDEL
9	Post-installation cleanup	N/A
10	Load QMF DB Packages to Remote server (optional)	DSQ2BPKB

Installation Checklists

QMF NLF Installation Checklist

Table 49. Checklist for QMF National Language Feature

Step	Description	File or EXEC Name
	Read and follow the installation steps in the program directory	
1	Create the QMF NLF installation control file	DSQ2nCTL
2	Run the QMF NLF installation EXEC	DSQ2nINS
3	Invoke QMF NLF	DSQ2nINV
4	Run the NLF IVP	DSQ2nIVP
5	Install the QMF NLF sample queries	DSQ2nSQD DSQ2nSQI
6	Run the IVP for batch mode processing	DSQ2nBAT
7	Post-installation cleanup	N/A

Appendix B. QMF Objects Residing in DB2 for VM

The following tables show a DBA the QMF 7.1 objects that reside in the database. The tables are intended to summarize all the database objects that are needed to run QMF 7.1 in the DB2 for VM 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.

The DDL parts and jobs reside on the QMF distribution and products disks.

Input to DSQ2EINS or DSQ2nINS

Table 50 shows the files for DSQ2EINS or DSQ2nINS.

Table 50. Input to DSQ2EINS or DSQ2nINS

File name	Created by EXEC
QMFV710E INSTALL	DSQ2ECTL
QMFV710n INSTALL	DSQ2nCTL

QMF User ID

Table 51 shows the QMF user ID and its definition part.

Table 51. QMF user ID

Name	Definition part
Q	DSQ2SETQ

QMF Control Tables

Table 52 shows the control tables shipped with QMF.

Table 52. QMF control tables

Table	Index	View	dbspace*	DDL part
Q.OBJECT_DIRECTORY	Q.OBJECT_DIRECTORYX	none	DSQTSC1	DSQ2TBLI
Q.OBJECT_REMARKS	Q.OBJECT_REMARKSX	none	DSQTSC2	DSQ2TBLI
Q.OBJECT_DATA	Q.OBJECT_OBJDATA	none	DSQTSC3	DSQ2TBLI
Q.PROFILES	Q.PROFILEX	none	DSQTSPRO	DSQ2TBLU
Q.ERROR_LOG	none	none	DSQTSLOG	DSQ2TBLE

Table 52. QMF control tables (continued)

Table	Index	View	dbspace*	DDL part
Q.COMMAND_SYNONYMS	Q.COMMAND_SYNONYMSX	none	DSQTSSYN	DSQ2TBLN
Q.RESOURCE_TABLE	Q.RESOURCE_INDEX	Q.RESOURCE_VIEW	DSQTSGOV	DSQ2TBLN
Q.DSQ_RESERVED	none	none	DSQTSRDO	DSQ2TBLR

: *dbspaces are acquired using the DSQDBSP EXEC.

Default List Views

You might have customized the default list views. Table 53 describes the default views shipped with QMF.

Table 53. Default List Views

View Name	DDL Member
Q.DSQEC_TABS_SQL	DSQ2TBLV
Q.DSQEC_COLS_SQL	DSQ2TBLV
Q.DSQEC_QMFOBJS	DSQ2TBLV

QMF Packages

Table 54 describes the packages shipped with QMF.

Table 54. QMF packages

Package name	Preprocessing job
DSQCxxxx	DSQ2PREP

Note: For the job that runs the particular DDL member, see the index.

NLF Parts

Table 55 describes the NLF table shipped with QMF

Table 55. QMF NLF table

Table	Index	DDL member in DSQSAMPn
Q.OBJECT_SYNONYM_n	Q.COMMAND_SYNONYMX_n	DSQ2nSYC

Appendix C. Migration and Fallback Considerations

Note: Skip this section if you are installing QMF for the first time.

Your users may need certain kinds of help before they can operate the new release of QMF.

Migrating from a Previous QMF Release to QMF 7.1

This section describes changes to the global variables (and the governor exit control block), the callable interface communications area, the invocation procedure (use of), and the view Q.VPROFILE.

Global Variables and the Governor

These are the changes made to the global variables and the governor exit control block for QMF 7.1:

- Global variables
 - DSQA0_QMF_RELEASE is now set to '12'
 - DSQA0_QMF_VER_RELS is now set to 'QMF V7R1.0'
- Governor exit control block DXEXCBA fields
 - XCBRELN is now set to '12'
 - XCBQMF is now set to 'QMF V7R1.0'

Use of the Invocation Procedure

In QMF Version 2, if you start QMF with a value for the DSQSRUN program parameter, or issue the END command to leave QMF, QMF runs an “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 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

Migration and Fallback Considerations

migrating to QMF 7.1 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 7.1 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 Q.VPROFILE:

```
CREATE VIEW Q.VPROFILE AS
  SELECT * FROM Q.PROFILES
  WHERE CREATOR = 'SYSTEM'
```

Multiple Releases of QMF

Multiple releases of QMF can access one DB2 for VM database, and all releases use the same QMF control tables (and hence, QMF objects).

Migrating to a new DB2 for VM level

When upgrading your database to a new level of DB2 for VM, it is recommended that you follow the DB2 for VM migration installation procedure described in *DB2 Server for VM Installation Guide*. When you migrate to a new level of DB2 for VM, QMF requires its DCSS to be rebuilt.

If you install the new level of DB2 for VM as an initial installation and then decide to migrate to it from the old database, it is important to remember that QMF requires the existence of the following items in DB2 for VM:

- QMF DBSPACES and associated tables, indexes, and views
- QMF catalog views NOT associated with QMF DBSPACES:
 - Q.DSQEC_TABS_SQL
 - Q.DSQEC_COLS_SQL

These views are described in DSQ2TBLV on your QMF distribution minidisk.

- QMF program access modules

If QMF DBSPACES and QMF catalog views not associated with QMF DBSPACES already exist in the new database, proceed to step 3 of the installation process and follow the database-only installation procedure. When running DSQ2ECTL, specify v3r3 migration to indicate that the QMF control tables do not require alteration by the installation EXEC.

If you are migrating from SQL/DS V2R2 to a later release, you must drop the Q.OBJECT_DATA table and recreate it so that the APPLDATA column is defined as LONG VARCHAR FOR BIT DATA:

1. Use the DB2 for VM DBS Utility to unload the Q.OBJECT_DATA table.

- Drop and create the Q.OBJECT_DATA table with the following SQL queries:

```
CONNECT Q IDENTIFIED BY xxx ;

DROP TABLE OBJECT_DATA ;

CREATE TABLE Q.OBJECT_DATA
    ("OWNER"          CHAR(8)          NOT NULL,
     "NAME"           VARCHAR(8)      NOT NULL,
     "TYPE"           CHAR(8)          NOT NULL,
     "SEQ"            SMALLINT         NOT NULL,
     "APPLDATA"       LONG VARCHAR    FOR BIT DATA)
    IN DSQTSCT3;

COMMENT ON TABLE Q.OBJECT_DATA IS
    'QMF SAVED ITEM DATA TABLE';
COMMENT ON COLUMN Q.OBJECT_DATA.OWNER IS
    'AUTHORIZATION ID OF QMF ITEM OWNER';
COMMENT ON COLUMN Q.OBJECT_DATA.NAME IS
    'NAME OF QMF ITEM';
COMMENT ON COLUMN Q.OBJECT_DATA.TYPE IS
    'TYPE OF QMF ITEM';
COMMENT ON COLUMN Q.OBJECT_DATA.SEQ IS
    'ROW SEQUENCE NUMBER';
COMMENT ON COLUMN Q.OBJECT_DATA.APPLDATA IS
    'QMF ITEM DATA';
CREATE UNIQUE INDEX Q.OBJECT_DATA_X ON Q.OBJECT_DATA
    ("OWNER" ASC, "NAME" ASC, "SEQ" ASC);
```

- Use the DB2 for VM DBS Utility to reload data into the Q.OBJECT_DATA table.

Migration and 31-Digit Decimal Support

If you are using QMF 3.1 (or later) and are operating in SQL/DS 3.2 (or later), you have 31-digit decimal support. If you migrate from an earlier version of QMF, or from an earlier database version, you may want to determine which tables are impacted by the 31-digit decimal support. The following query retrieves a list of the user tables from the SQL/DS tables:

```
SELECT DISTINCT(TNAME) FROM SYSTEM.SYSCOLUMNS WHERE COLTYPE IN
    ('INTEGER', 'SMALLINT', 'FLOAT', 'DECIMAL', 'DATE', 'TIME', 'TIMESTAMP')
ORDER BY TNAME
```

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 VM. Cleanup is described in “Part 1. Installing QMF for VM/ESA” on page 1 and is not discussed here.

Fallback isn't likely to be done unless the two versions of QMF are running from the same DB2 for VM database.

Migrating to a New DB2 for VM Level

Re-establishing the Earlier Profiles

Because QMF Version 3 added an ENVIRONMENT column to Q.PROFILES, it is possible to have multiple rows where the CREATOR and TRANSLATION columns contain the same values (but with different values in the ENVIRONMENT column). Since there was no ENVIRONMENT column in QMF versions earlier than Version 3, this means that when you fall back from Version 3 to Version 2, some rows in Q.PROFILES that were unique in Version 3 appear as duplicates in Version 2. Therefore, you might need to re-establish the earlier profile. This is necessary only under certain conditions.

- If your SQL/DS is Version 3.2 or earlier, no action is needed.
- If your SQL/DS is Version 3.3 or later, and you are falling back to QMF Version 3.1, no action is needed.
- If your SQL/DS is Version 3.3 or later, and you are falling back to QMF Version 2.4 or earlier, delete the ENVIRONMENT column from the Q.PROFILES table.

Using Version 7 Objects Under Earlier QMF 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 fall back to the earlier QMF release, some objects created under QMF Version 7 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 objects in earlier releases.

- Forms

Form objects that were saved or exported under Version 7, and displayed or imported to earlier releases of QMF, can be expected to execute normally. However, form objects saved or exported from Version 7 cannot be used in Version 2.4 or earlier.

Before they can be used in earlier applications, forms exported from Version 7 that use break field numbers (or the object level in the header record) require the Form Application Migration Aid.

- Queries

Some restrictions apply to Version 7 queries for fallback to earlier releases:

- SQL queries: You can export SQL queries from Version 7 and import them on earlier releases, and they execute normally. However, SQL queries saved on Version 7 cannot be used in Version 2.4 or earlier.
- Prompted queries: You can display and import Version 7 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 can be displayed or imported in earlier releases and execute normally.

- Procedures
Procedure objects exported from Version 7 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 cannot be displayed with earlier releases unless you first export them from Version 7 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 could fail to run for a number of reasons. See “Using Version 7 QMF Commands with Earlier Releases”, for details.
- Applications that call the callable interface
Applications call the callable interface in their EXECs 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.
- Applications containing QMF commands that cannot be run in the earlier release.
These are the same commands that cannot be run in a procedure, and for the same reasons. See “Using Version 7 QMF Commands with Earlier Releases” for details.

Importing objects to Version 2.4

If you have queries or procedures created under Version 3 which do not have any of the characteristics just mentioned, it might be possible to use them under QMF Version 2.4. To use your Version 3 procedures or queries:

1. Export your Version 3 objects while running under Version 3.
2. Import the objects while running under QMF Version 2.4.

You might want more information on the differences between the earlier QMF release and QMF Version 3. If you do, compare the two releases of *QMF Reference*.

Using Version 7 QMF Commands with Earlier Releases

Version 7 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.

Migrating to a New DB2 for VM Level

31-Digit Decimal Support

If you are using QMF Version 3.1 (or later) and are operating in SQL/DS Version 3.2 (or later), you have 31-digit decimal support. Some expressions might produce unusual or unacceptable results when migrated from 15-digit to 31-digit decimal format. It is very difficult to fall back to a 15-digit decimal environment without backup data.

Appendix D. QMF Control Tables and dbspaces Used by QMF

QMF uses the control tables shown in Table 56 to manage QMF users and the objects they create. The dbspace sizes given for each block are in pages, where each page is one 4096-byte block. See the page listed at the right of the table if you need information on the table's structure and more detailed information on how QMF uses it.

Table 56. List of QMF control tables and dbspaces used by QMF

Control table name	dbspace	dbspace size	Table content	More information:
Q.PROFILES	DSQTSPRO	128 pages	Contains QMF profiles that hold information about individual users' access to resources and data during a QMF session.	Pages 97 to 107
Q.OBJECT_DIRECTORY	DSQTSCT1	256 pages	Contains general information about all QMF queries, forms, and procedures in the database.	Page 121
Q.OBJECT_DATA	DSQTSCT3	5120 pages	Contains queries, forms, and procedures represented in an internal QMF format.	Page 122
Q.OBJECT_REMARKS	DSQTSCT2	256 pages	Contains comments that were saved when queries, forms, and procedures were created (or replaced).	Page 123
Q.COMMAND_SYNONYMS	DSQTSCT2	128 pages	Contains information on the command synonyms	Page 159
Q.RESOURCE_TABLE	DSQTSGOV	128 pages	Contains resource control information passed to the governor exit routine.	Page 227
Q.ERROR_LOG	DSQTSLOG	128 pages	Contains information on system, resource, and "unexpected condition" errors. This information is more detailed than that found in error messages.	Page 308

QMF Control Tables and dbspaces Used by QMF

Table 56. List of QMF control tables and dbspaces used by QMF (continued)

Control table name	dbspace	dbspace size	Table content	More information:
Q.DSQ_RESERVED	DSQTSRDO	128 pages	Contains information used by QMF during initialization.	This table is not discussed in this book.

In addition to the dbspaces shown in Table 56 on page 325 for the QMF control tables, QMF uses dbspace DSQ2STBT for the QMF sample tables, and DSQTSDEF to store data from the QMF SAVE DATA or IMPORT TABLE commands. Both dbspaces have a default size of 128 pages.

For more information about the QMF sample tables, see *Using QMF*. For more information on the SAVE DATA or IMPORT TABLE commands, see *QMF Reference*.

Appendix E. 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
P.O. Box 49023
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	IBMLink
Advanced Peer-to-Peer Networking	IMS
AIX	Language Environment
AIX/6000	MVS
AS/400	MVS/ESA
C/370	MVS/XA
CICS	OfficeVision/VM
CICS/ESA	OS/2
CICS/MVS	OS/390
CICS/VSE	PL/I
COBOL/370	PROFS
DATABASE 2	QMF
DataJoiner	RACF
DB2	S/390
DB2 Universal Database	SQL/DS
Distributed Relational Database Architecture	Virtual Machine/Enterprise Systems Architecture
DRDA	Visual Basic
DXT	VM/XA
GDDM	VM/ESA
IBM	VSE/ESA
	VTAM

Java or all Java-based trademarks and logos, and Solaris are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Lotus and 1-2-3 are trademarks of Lotus Development Corporation in the United States, other countries, 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.

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.

Advanced Peer-to-Peer Networking. A distributed network and session control architecture that allows networked computers to communicate dynamically as equals. Compare with Advanced Program-to-Program Communication (APPC). An implementation of the SNA synchronous data link control LU 6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

Advanced Program-to-Program Communication (APPC). An implementation of the SNA synchronous data link control LU 6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

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.

APPC. Advanced Program-to-Program Communication

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

Glossary

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.

chart. A graphic display of information in a report.

CICS. Customer Information Control System.

client. A functional unit that receives shared services from a server.

CMS. Conversational Monitor System.

column. A vertical set of tabular data. It has a particular data type (for example, character or numeric) and a name. The values in a column all have the same data characteristics.

column function. An operation that is applied once to all values in a column, returns a single value as a result, and is expressed in the form of a function name followed by one or more arguments enclosed in parentheses.

column heading. An alternative to the column name that a user can specify on a form. Not saved in the database, as are the column name and label.

column label. An alternative descriptor for a column of data that is saved in the database. When used, column labels appear by default on the form, but they can be changed by users.

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.

connectivity. The enabling of different systems to communicate with each other. For example, connectivity between a DB2 UDB for OS/390 application requester and a DB2 for VM and VSE application server enables a DB2 UDB for OS/390 user to request data from a DB2 for VM and VSE database.

conversation. A logical connection between two programs over an LU 6.2 session that allows them to communicate with each other while processing a transaction.

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.

Glossary

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.

DATA. An object in temporary storage that contains the information returned by a retrieval query. Information represented by alphanumeric characters contained in tables and formatted in reports.

database. A collection of data with a given structure for accepting, storing, and providing on demand data for multiple users. In DB2 UDB for OS/390, a created object that contains table spaces and index spaces. In DB2 for VM and VSE, a collection of tables, indexes, and supporting information (such as control information and data recovery information) maintained by the system. In OS/2, a collection of information, such as tables, views, and indexes.

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. Designates a day, month, and year (a three-part value).

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.

DB2 UDB for OS/390. DB2 Universal Database for OS/390 (an IBM relational database management system).

DB2 for AIX. DATABASE2 for AIX. The database manager for QMF's relational data.

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. 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.

DOC. The keyword for a document problem.

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.

DRDA. Distributed Relational Database Architecture.

duration. An amount of time expressed as a number followed by one of seven keywords: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS, MICROSECONDS.

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

example element. A symbol for a value to be used in a calculation or a condition in a QBE query.

example table. The framework of a QBE query.

fixed area. That part of a report that contains fixed columns.

fixed columns. The columns of a report that remain in place when the user scrolls horizontally. On multiple-page, printed reports, these columns are repeated on the left side of each page.

Glossary

form. An object that contains the specifications for printing or displaying a report or chart. A form in temporary storage has the name of FORM.

function key table. A table containing function key definitions for one or more QMF panels, along with text describing the keys. Each user can be assigned one of these tables.

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.

GDDM. Graphical Data Display Manager.

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. 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.

HELP. Additional information about an error message, a QMF panel, or a QMF command and its options.

host. A mainframe or mid-size processor that provides services in a network to a workstation.

HTML. Hypertext Markup Language. A standardized markup language for documents displayed on the World Wide Web.

ICU. Interactive Chart Utility.

INCORROUT. The keyword for incorrect output.

index. A collection of data about the locations of records in a table, allowing rapid access to a record with a given key.

initial procedure. A QMF procedure specified by the DSQSRUN parameter on the QMF start command which is executed immediately after QMF is invoked.

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).

installation-defined command. A command created by an installation. QMF will process it as one of its own commands or as a combination of its commands.

installation-defined format. Date and time formats, also referred to as LOCAL formats, that are defined (or built) by the installation.

interactive execution. Execution of a QMF command in which any dialog that should take place between the user and QMF during the command's execution actually does take place.

interactive session. Any QMF session in which the user and QMF can interact. Could be started by another interactive session by using the QMF INTERACT command.

interactive switch. A conceptual switch which, when on, enables an application program to run QMF commands interactively.

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.

IXE. 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

join. A relational operation that allows retrieval of data from two or more tables based on matching columns that contain values of the same data type.

keyword parameter. An element of a QMF command consisting of a keyword and an assigned value.

like. Pertaining to two or more similar or identical IBM operating environments. For example, like distribution is distribution between two DB2 UDB for OS/390's with compatible server attribute levels. Contrast with "unlike".

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.

line wrapping. Formatting table rows in a report so they occupy several lines. The row of column names and each row of column values are split into as many lines as are required by the line length of the report.

local. Pertaining to the relational database, data, or file that resides in the user's processor. See also "local DB2 UDB for OS/390", and contrast with *remote*.

Glossary

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.

LU. Logical unit.

LU 6.2. Logical Unit type 6.2.

LOOP. The keyword for an endless-loop problem.

MSGx. The keyword for a message problem.

Multiple Virtual Storage. Implies the MVS/ESA product

MVS/ESA. Multiple Virtual Storage/Enterprise System Architecture (IBM operating system).

NCP. Network Control Program.

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.

node. In SNA, an end point of a link or a junction common to two or more links in a network. Nodes can be distributed to host processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities.

null. A special value used when there is no value for a given column in a row. *Null* is not the same as zero.

null value. See *null*.

object. A QMF query, form, procedure, profile, report, chart, data, or table. The report, chart, and data objects exist only in temporary storage; they cannot be saved in a database. The table object exists only in a database.

object name. A character string that identifies an object owned by a QMF user. The character string can be a maximum of 18 bytes long and must begin with an alphabetic character. The term “object name” does not include the “owner name” prefix. Users can access other user’s objects only if authorized.

object panel. A QMF panel that can appear online after the execution of one QMF command and before the execution of another. Such panels include the home, report, and chart panels, and all the panels that display a QMF object. They do not include the list, help, prompt, and status panels.

online execution. The execution of a command from an object panel or by pressing a function key.

owner name. The authorization id of the user who creates a given object.

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. An object that contains QMF commands. It can be run with a single RUN command. A procedure in temporary storage has the name of PROC. See also “linear procedure” and “procedure with logic.”

procedure termination switch. A conceptual switch that a QMF MESSAGE command can turn on. While on, every QMF procedure to which control returns terminates immediately.

Glossary

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.

prompt panel. A panel that is displayed after an incomplete or incorrect QMF command has been issued.

Prompted Query. A query built in accordance with the user’s responses to a set of dialog panels.

protocol. The rules governing the functions of a communication system that must be followed if communication is to be achieved.

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 privilege 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’.

query. An SQL or QBE statement, or a statement built from prompting, that performs data inquiries or manipulations. A saved query is an SQL query, QBE query, or Prompted Query that has been saved in a database. A query in temporary storage, has the name QUERY.

RDBMS. Relational database management system

relational database. A database perceived by its users as a collection of tables.

relational database management system (RDBMS). A computer-based system for defining, creating, manipulating, controlling, managing, and using relational databases.

remote. Pertaining to a relational DBMS other than the local relational DBMS.

remote data. Data that is maintained by a subsystem other than the subsystem that is attempting to access the data. Contrast with local data.

remote data access. Methods of retrieving data from remote locations. The two remote data access functions used by QMF are *remote unit of work* and DB2 UDB for OS/390-only distributed unit of work, which is called *system-directed access*.

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.

report. The formatted data produced when a query is issued to retrieve data or a DISPLAY command is entered for a table or view.

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. A horizontal set of tabular data.

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.

sample tables. The tables that are shipped with QMF. Data in the sample tables is used to help new QMF users learn the product.

saved object. An object that has been saved in the database. Contrast with current object.

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.

screen. The physical surface of a display device upon which information is presented to the user.

scrollable area. The view of a displayed object that can be moved up, down, left, and right.

server. A functional unit that provides shared services to workstations over a network.

session. All interactions between the user and QMF from the time the user logs on until the user logs off.

single-byte character. A character whose internal representation consists of one byte. The letters of the Latin alphabet are examples of single-byte characters.

SNA. Systems Network Architecture.

SNAP dump. A dynamic dump of the contents of one or more storage areas that QMF generates during an abend.

Glossary

sort priority. A specification in a retrieval query that causes the sorted values in one retrieved column to determine the sorting of values in another retrieved column.

SQL. Structured Query Language.

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.

stored object. An object that has been saved in permanent storage. Contrast with current object.

string. A set of consecutive items of a similar type; for example, a character string.

Structured Query Language. 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. The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

table. A named collection of data under the control of the relational database manager. A table consists of a fixed number of rows and columns.

Table Editor. The QMF interactive editor that lets authorized users make changes to a database without having to write a query.

table name area. The leftmost column of a QBE example table.

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.

time. Designates a time of day in hours and minutes and possibly seconds (a two- or three-part value).

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. The work that occurs between 'Begin Unit of Work' and 'Commit' or 'Rollback'.

transaction program. A program that processes transactions in an SNA network. There are two kinds of transactions programs: application transaction programs and service transaction programs.

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.

unlike. Refers to two or more different IBM operating environments. For example, unlike distribution is distribution between DB2 for VM and VSE and DB2 UDB for OS/390. Contrast with *like*.

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

value. A data element with an assigned row and column in a table.

Glossary

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.

view. An alternative representation of data from one or more tables. It can include all or some of the columns contained in the table or tables on which it is defined. (2) The entity or entities that define the scope of the data to be searched for a query.

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.

window. A rectangular portion of the screen in which all or a portion of a panel is displayed. A window can be smaller than or equal to the size of the screen.

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.)

wrapping. See “column wrapping” and “line wrapping”.

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.

APPC Publications

Communicating with APPC and CPI-C: A Technical Overview
Networking with APPC: An Overview

CICS Publications

CICS Transaction Server for OS390

CICS/OS390 User's Handbook
CICS/OS390 Application Programmer's Reference
CICS/OS390 Application Programming Guide
CICS/OS390 DB2 Guide
CICS/OS390 Resource Definition (Macro)
CICS/OS390 Resource Definition (Online)
CICS/OS390 Problem Determination Guide
CICS/OS390 System Definition Guide
CICS/OS390 Intercommunication Guide
CICS/OS390 Performance Tuning Handbook

CICS for VSE

- *CICS for VSE/ESA User's Handbook*
- *CICS for VSE/ESA Application Programmer's Reference*
- *CICS for VSE/ESA Application Programming Guide*
- *CICS for VSE/ESA Resource Definition (Macro)*
- *CICS for VSE/ESA Resource Definition (Online)*
- *CICS for VSE/ESA Problem Determination Guide*
- *CICS/OS390 System Definition Guide*
- *CICS for VSE/ESA Intercommunication Guide*
- *CICS for VSE/ESA Performance Tuning Handbook*

Bibliography

COBOL Publications

VS COBOL II Application Programming Guide for VSE
COBOL/VSE Language Reference
COBOL/VSE Programming Guide

DATABASE 2 Publications

DB2 UDB for OS390

DB2 UDB for OS390 Installation Guide
DB2 UDB for OS390 Administration Guide
DB2 UDB for OS390 SQL Reference
DB2 UDB for OS390 Command Reference
DB2 UDB for OS390 Application Programming and SQL Guide
DB2 UDB for OS390 Message and Codes
DB2 UDB for OS390 Utility Guide and Reference
DB2 UDB for OS390 Call Level Interface Guide and Reference
DB2 UDB for OS390 Reference for Remote DRDA Requesters and Servers

DB2 for VSE & VM

DB2 Server for VM Installation Guide
DB2 Server for VSE Installation Guide
DB2 Server for VSE & VM Database Administration
DB2 Server for VM System Administration
DB2 Server for VSE 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 VM Message and Codes
DB2 Server for VSE Message and Codes
DB2 Server for VSE & VM Diagnostic Guide and Reference
DB2 Server for VSE & VM Performance Tuning Handbook

DB2 for AS/400

DB2 for AS/400 SQL Reference
DB2 for AS/400 SQL Programming

Parallel Edition

DB2 Parallel Edition Administration Guide and Reference

DB2 Universal Database

DB2 Universal Database Command Reference
DB2 Universal Database SQL Reference
DB2 Universal Database Message Reference

DataJoiner

DataJoiner Application Programming and SQL Reference Supplement

DCF Publications

DCF and DLF General Information

DRDA Publications

DRDA Every Manager's Guide

DRDA Connectivity Guide

DXT Publications

DXT Guide to Dialogs

Data Extract: Planning and Administration Guide for Dialogs

Data Extract: UserÆs Guide

Learning to Use DXT

Graphical Data Display Manager (GDDM) Publications

GDDM General Information

GDDM Base Programming Reference

GDDM Base Programming Guide

GDDM Guide for Users

GDDM Installation and System Management for VSE

GDDM Messages

HLASM Publications

IBM High-Level Assembler Programmer's Guide for OS/390, VM and VSE

IBM High-Level Assembler Language Reference for OS/390, VM and VSE

ISPF/PDF Publications

OS/390

Interactive System Productivity Facility for OS/390 Installation and Customization

Interactive System Productivity Facility for OS/390 Dialog Management Guide

Interactive System Productivity Facility for OS/390 Dialog Management Services and Examples

VM

ISPF for VM Dialog Management Services and Examples

Bibliography

OS/390 Publications

Utilities

OS/390 Administration: Utilities
OS/390 Extended Architecture Utilities

JCL

OS/390 Extended Architecture JCL Reference
OS/390 Extended Architecture JCL User's Guide
OS/390 JCL Reference
OS/390 JCL Users 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

OS/390 TSO Primer
OS/390 User's Guide

SMP/E

OS/390 System Modification Program Extended Messages and Codes
OS/390 System Modification Program Extended Primer
OS/390 System Modification Program Extended Reference
OS/390 System Modification Program Extended User's Guide

PL/I Publications

PL/I VSE Language Reference
PL/I VSE Programming Guide

REXX Publications

OS/390 environment

IBM Compiler and Library for REXX/370: User/Es Guide and Reference
TSO Extensions REXX/MVS Reference

VM environment

Procedures Language VM/REXX Reference
Procedures Language VM/REXX User's Guide

ServiceLink Publications

ServiceLink User's Guide

VM Publications

Virtual Machine Planning Guide and Reference
Virtual Machine CMS Command and Macro Reference

VSE Publications

VSE Planning Guide
VSE Guide to System Functions
VSE System Utilities
VSE Guide for Solving Problems

Bibliography

Index

Numerics

- 31-digit decimal 321
- 31-digit decimal and fallback 324
- 31-digit decimal and migration 321

A

- abbreviating command
 - synonyms 170
- abbreviations
 - for command synonyms 170
- ABEND handler 294
- abend handling 305
- access
 - to objects
 - controlling 107
 - database object list, customizing 112
 - queries, forms, procedures 111
 - SQL GRANT statement 110
 - SQL REVOKE statement 110
 - to QMF
 - enabling 97
 - restricting 99
- ACQUIRE DBSPACE command 14
- ACQUIRE dspace statement 14, 118
- acquire the DB2 for VM DBSPACE(s) 28
- address, governor function calls 243
- ADMADFV defaults module 154
- administration
 - acquiring dspace 118
 - DB2 for VM system tables 127
 - granting and revoking privileges 109
 - listing user's tables/views 128
 - object
 - controlling access 111
 - deleting 125
 - displaying user's 124
 - listing user's 123
 - transferring ownership 124, 128
 - Q user profile 95, 97
 - resources 95, 97
 - tables, creating 116
 - user profiles and objects 106

- ALL keyword, tracing 81
- allocating DXT CMS files, example 131, 136
- ALTER DBSPACE statement 127
- ALTER statement
 - DBSPACE keyword 127
- ampersand (&)
 - in command synonyms 166
- ampersands in command synonyms 166
- APAR (Authorized Program Analysis Report) 309
- APPLDATA 270
- APPLDATA column 123
- application procedures
 - install in base 37
 - install in NLF 49
- application requester
 - definition 13
- application requester (AR) 13
- application requester for QMF
 - definition 13
- application server
 - definition 13
- application server (AS) 13
- assembler
 - compiling assembler edit routine 202
 - edit routine
 - assembling 202
 - compiler options 202
 - creating a DSQUEDIT module file 203
 - edit exit routine structure 197
 - edit function call 198
 - generate 203
 - interface control block 199, 205
 - linkage 198
 - returning control to the caller 198
- assembler edit routine
 - creating a DSQUEDIT module file 203
 - linkage 198
 - returning control to the caller 198

- assembling assembler edit routine 202
- asynchronous processing, printing 149
- authorization
 - cascading 110
 - command synonyms 168, 171
 - creating tables 116
 - DBA, user Q 95, 97
 - messages 108
 - RESOURCE 118
 - to access QMF 98

B

- B parameter 73
- base QMF commands as synonyms 164
- batch
 - DSQSMODE parameter 83
 - expected results for IVP 39
 - ISPSTART command 63
 - QMF procedures
 - identifying (DSQSRUN parameter) 63
 - running a query or procedure in 160
 - running IVP in 38
 - running the IVP (NLF) 50
 - starting QMF
 - from ISPF 63
 - starting QMF in 64
- BATCH application
 - filling in the prompt panel 283
 - MACLIBs required 283
 - modifying 286
 - starting 283
- batch-mode QMF
 - DSQSMODE parameter 83
- batch processing
 - customizing the program 277
 - debugging a batch procedure 282
 - enabling use 278
 - running jobs on your machine 281
 - sending a job to the CMS batch machine 279
 - using the BATCH application 282

- batch use of QMF, specifying through parameter 83
- bilingual support forms 144
- bilingual support, QMF forms 144
- binary data in reports 295
- binary data in reports 295
- branch addresses, governor 243
- Brazilian Portuguese NLF 20
- C**
- callable interface
 - DSQSCMDn program parameter 68
- canceling governor 259
- cancellation service, governor 259
- cascading authority 110
- case, setting 101
- CASE column (Q.PROFILES) 101
- catalog views QMF 17
- chart
 - printing 148
 - GDDM vs QMF 149
 - specific objects 148
- chart support 120
- charts
 - printing 149
- CHARVAL column, Q.RESOURCE_TABLE 231
- checklist, installation
 - NLF 316
 - QMF 315
- Chinese NLF 20
- class ID, customizing function keys 178
- CMS
 - command errors
 - DB2 for VM COMMIT command 295
 - DB2 for VM CONNECT command 294
 - QMF ABEND handler 294
 - QMF interrupt handler 306
 - ENVIRONMENT values, QMF profile 104
 - interface to governor 240
 - QMF CMS command
 - command synonym 165
 - setting report storage 73, 74
 - subset mode 294
- CMS (Conversational Monitor System)
 - access 96
 - CMS (Conversational Monitor System) (*continued*)
 - command errors
 - DB2 for VM COMMIT command 295
 - DB2 for VM CONNECT command 294
 - QMF ABEND handler 294
 - QMF interrupt handler 306
 - ENVIRONMENT values, QMF profile 104
 - interface to governor 240
 - non-SUBSET mode 62
 - QMF CMS command, command synonym 165
 - storage requirements for QMF 14
 - subset mode 294
 - SUBSET mode 62
- COBOL 213
 - edit routine
 - creating a DSQUEDIT module file 220, 222
 - returning to the caller 214
- COBOL (with LE) 221
- COBOL edit routine
 - compiling 219
 - creating a DSQUEDIT module file 220, 222
 - edit exit routine structure 213
 - edit exit routine structure (with LE) 221
 - edit program call 214
 - interface control block 214
 - linkage 214
 - returning to the caller 214
- Code-only installation
 - create DB2 for VM
 - dbspace(s) 26
 - installation EXEC 28
- code page support
 - APPLDATA 270
 - special characters 270
- command 103
 - cancellation messages 260
 - cancellation service 259
 - CMS, synonym definition 165
 - customizing 159
 - function keys, assigning 173
 - governor exit routine calls 241
 - interface
 - installation verification procedure (IVP) 36, 49
 - interface initialization messages 291
- command 103 (*continued*)
 - ISPSTART 61
 - PRINT 147
 - privileges required 108
 - RUN
 - initial procedure 84
 - synonym definition 165
 - SAVE
 - SHARE parameter 111
 - SET PROFILE 105
 - synonyms 159
 - window IDs 181
- command (QMF) interface test, IVP
 - base 36
 - NLF 49
- command interface
 - considerations for non-SUBSET and SUBSET modes 62
- command synonyms table
 - creating 162
 - maintaining 170
 - views 171
- comment
 - on function keys table 177
 - on synonyms table 162
- comments
 - on function keys table 177
 - on synonyms table 162
- COMMIT command, DB2 for VM 295
- compiler options
 - assembler edit routine 202
 - COBOL edit routine 220
- CONFIRM column (Q.PROFILES) 101
- confirmation panel
 - displaying 101
 - table editor 119
- confirmation panels
 - displaying 101
 - table editor 119
- CONNECT authority, granting 96
- CONNECT command
 - CMS command 294
 - errors 291
- CONNECT ID
 - user Q 14
- CONNECT ID "Q" 14
- control programs, edit routine 192
- control section (CSECT), diagnosis 311
- control tables
 - dbspace names/sizes 325
 - installation of 5
 - ownership 95, 97

- control tables (*continued*)
 - Q.ERROR_LOG 308
 - Q.OBJECT_DATA 122
 - Q.OBJECT_DIRECTORY 121
 - Q.OBJECT_REMARKS 123
 - Q.PROFILES 98
 - Q.RESOURCE_TABLE 231
 - Q.RESOURCE_VIEW 232
 - QMF 16
- controlling rows fetched from the database 79
- Conversational Monitor System (CMS)
 - access 96
 - command errors
 - DB2 for VM COMMIT command 295
 - DB2 for VM CONNECT command 294
 - QMF ABEND handler 294
 - QMF interrupt handler 306
 - ENVIRONMENT values, QMF profile 104
 - interface to governor 240
 - non-SUBSET mode 62
 - QMF CMS command, command synonym 165
 - subset mode 294
 - SUBSET mode 62
- CREATE TABLE statement
 - command synonyms 162
 - customized function keys 176
 - privileges for SAVE DATA 108
 - resource control table 236
 - tables for users 116
- CREATOR column (Q.PROFILES)
 - defined 101
 - role in profile initialization 104
- CSECT (control section), diagnosis 311
- cursor stability 111
- customizing
 - function keys 173
 - QMF commands 159
 - QMF session behavior using user profile 97
- D**
 - D parameter 82
 - Danish NLF 20
 - data
 - files 32
 - Data Extract 129
 - data formats 187
 - data object
 - limiting initial rows retrieved 79
 - performance 296
 - privileges for SAVE DATA 108
 - database
 - only installation
 - for NLF 44
 - for QMF base 24
 - migration 320
 - CONNECT ID "Q" 14
 - connection
 - authority 95, 97
 - location name parameter 90
 - remote 263
 - starting QMF 57
 - object list, customizing 112
 - objects
 - access 107
 - granting privileges 109
 - ownership 109
 - revoking privileges 109
 - storage for 73, 74
 - requirements for QMF 13
 - slow performance 79, 296
 - database-only installation
 - for NLF 44
 - for QMF 24
 - Database-only installation migration 320
 - DATE columns 190
 - date information, handling 189
 - date/time exit routines, local 128
 - DB2 for VM
 - acquiring a dbspace 118
 - authority
 - DBA authority 109
 - displaying users' 106
 - change confirmation panels 119
 - CMS command errors 294
 - COMMIT cmd and CMS cmd errors 295
 - CONNECT command errors 294
 - CONNECT ID 28
 - DBA authority 109
 - dbspace 26, 28
 - displaying users' authority 106
 - enlarging dbspaces 126
 - granting and revoking privileges 109
 - granting CONNECT authority 96
 - granting RESOURCE authority 118
 - knowledge required 12
 - required by QMF 13
 - DB2 for VM (*continued*)
 - system tables 127
 - use of 3
 - DB2 for VM CONNECT ID, establish 28
 - DB2 for VM DBSPACE(s)
 - acquire 28
 - create 26
 - DBCS (double-byte character set) printing support 90
 - DBCS (double-byte character set) support
 - edit codes 223
 - Katakana characters 223
 - Latin characters 223
 - dbspace
 - acquiring 118
 - ADD dbspace statement 118
 - calculating size 118
 - creating tables 117
 - deleting 106
 - enlarging 126
 - nonrecoverable 119
 - QMF-supplied tables 325
 - requirements 14, 18
 - specifying in user profile 102
 - DBSPACE requirements
 - number to create 16
 - overview 14
 - Q.COMMAND_SYNONYMS 16
 - Q.DSQ_RESERVED table 16
 - Q.ERROR_LOG 16
 - Q.OBJECT tables 16
 - Q.PROFILES 16
 - Q.RESOURCE table 16
 - QMF control tables 16
 - QMF sample tables 17
 - SAVE DATA command 13
 - dcssname program parameter 72
 - decimal data, edit routine 188
 - DECOPT column (Q.PROFILES) 101
 - default
 - function keys 174
 - GDDM module ADMADFV 154
 - QMF profile 99
 - default function keys 174
 - default QMF profile 99
 - defaults module, GDDM printing 154
 - DESCRIBE command
 - customizing 112
 - DESCRIBE command, customizing 112
 - Deutsch (NLF) 20

DEVTOK keyword, ADMMNICK specification 151
 diagnostic aids 298
 DISPLAY command, SQL privileges required 108
 displaying reports (DPRE) 160, 269
 distribution minidisk
 create DB2 for VM
 DBSPACE(s) 26
 create QMF installation control file 25, 45
 NLF 45
 QMF 25
 QMF installation EXEC 28, 46
 base 28
 NLF 46
 DMSFREE storage problems 73
 DOS printers 153
 double-byte character set (DBCS) support
 edit codes 223
 Katakana characters 223
 Latin characters 223
 DPRE 160
 remote unit of work 269
 DRAW command
 SQL privileges required 108
 DRAW command, SQL privileges required 108
 DRDA
 remote unit of work 263
 DSQ2EINS, installation EXEC 28
 preparation 28
 DSQ2EINV, QMF invocation EXEC 30
 DSQCP global variables 119
 DSQDC_SHOW_PANID global variable 299
 DSQDEBUG 32
 requirements 32
 DSQEC_ALIASES variable 113
 DSQEC_COLS_LDB2 variable 113
 DSQEC_COLS_SQL variable 113
 DSQEC_FORM_LANG variable 144
 DSQEC_NLFCMD_LANG variable 144
 DSQEC_RERUN_IPROC global variable 84
 DSQEC_TABS_LDB2 variable 113
 DSQEC_TABS_SQL variable 113
 DSQEDIT 33
 DSQLDLIB 33
 DSQMFE
 independent of ISPF 64
 return codes 64
 DSQPNLE 33
 DSQPRINT 33
 allocation
 using the QMF CMS command 33
 DSQQMFE module 61, 64
 DSQSBSTG (maximum storage for reports)
 general 73
 DSQSBSTG parameter 73
 DSQSCMDE (REXX initialization program) 69
 DSQSCMDn 68
 DSQSCMDn parameter 68
 DSQSDBCS (choose Kanji) 90
 DSQSDBCS parameter 90
 DSQSDBNM (initial location name) 82
 DSQSDBNM parameter 82
 DSQSDEBUG (set trace) 81
 DSQSDEBUG parameter 81
 DSQSIROW (control rows fetched) 79
 DSQSIROW parameter 79
 DSQSMODE (choose mode) 83
 DSQSMODE (operation mode)
 batch-mode QMF 63
 DSQSMODE parameter 83
 DSQSPILL 33
 DSQSPILL (spill file) 75
 DSQSPILL parameter 75
 file requirements 33
 DSQSRSTG (reserve storage) 74
 DSQSRSTG parameter 74
 DSQSRUN
 description 84
 initial procedure 84
 DSQSRUN (called procedure)
 batch-mode QMF 63
 DSQSRUN parameter 84
 passing variables 86
 DSQUECIC edit program 191
 DSQUEGV2 262
 DSQUEGV3 phase, governor exit 243
 DSQUnGV2 phase, governor exit 239
 DSQUXDTA sample edit routine 197
 DSQUXDTC sample edit routine 213
 DXEPCS control block 199, 205, 214
 DXEGOVA control block 245
 DXEXCBA control block 250
 DXT (Data Extract) 129

DXT (Data Extract) 129 *(continued)*
 allocating files 129
 EXTRACT command 129

E

edit
 codes 188
 binary data 295
 CASE field of profile 189
 DBCS data 223
 numeric data processing 188
 types 188
 UDN 191
 VSS 191
 exit interface 187
 assembler 199, 205
 COBOL 214
 control block fields 193
 formatting calls 191
 input area 195
 output area 195, 196
 termination calls 197
 exit routine 193
 routine 187
 assembler 197
 COBOL 213
 COBOL (with LE) 221
 DBCS data 223
 general structure 191
 sample program (assembler) 197
 scratchpad area 214
 storing data between calls 199, 205
 EDIT command 143
 EDIT command, support for 139
 edit routines
 creating a DSQUEDIT module file
 in assembler 203
 in PL/I 210
 in VS COBOL II 220, 222
 handling different codes 195
 specific languages
 PL/I 203, 209
 PL/I (with LE) 211
 VS COBOL II 219
 EDIT TABLE command
 concurrent editing 110
 SQL privileges required 108
 editing interfaces
 customizing 139
 editor, make available for the EDIT command 139
 English QMF, NLID 20

English support in NLF session 144
 enrolling users in QMF 98
 ENTRY_TYPE column (function key table) 179
 environment
 changing in QMF profile 104
 customizing 97
 ENVIRONMENT column (Q.PROFILES)
 role in profile initialization 104
 environmental considerations when starting QMF 61
 error
 initialization 81, 84
 insufficient storage 79
 messages
 authorization 108
 warning 291
 printing 157
 QMF log 308
 reporting to IBM 309
 errors
 CMS command 294
 EXEC
 starting QMF, CMS 61
 EXEC DSQ2EINV
 examples of invocation
 statements 32
 getting to the QMF Home Panel 36
 tailoring 30
 exit routines
 date/time 128
 explicit connection
 granting CONNECT
 authority 96
 EXPORT TABLE, SQL
 privileges 108
 extended floating point, edit routine 188
 EXTRACT command 129
 allocating resources 129
 allocation through an EXEC 130
 names for CMS files 129
 reallocation through an EXEC 135
 support 129

F

F parameter 79
 fallback
 31-digit decimal support 324
 installation considerations 18
 fallback, installation considerations 18

fallback and clean-up 321
 fallback considerations 324
 Family 1 printer 150
 Family 2 printer 150, 151
 Family 3 printer 150, 152
 Family 4 printer 152
 file (CMS)
 DSQDEBUG 32
 DSQPRINT 33
 DSQSPILL 33
 file requirements
 DSQDEBUG 32
 DSQPRINT 33
 DSQSPILL 33
 floating point data, edit routine 188
 FLOATVAL column,
 Q.RESOURCE_TABLE 231
 formatting calls, edit routine 191
 forms
 controlling user access 111
 creating new edit codes 188
 displaying 124
 internal stored format 122
 listing 123
 NLF support 144
 of the ISPSTART command 61
 printing 148
 window IDs 182
 French NLF 20
 full-screen panels 179, 181
 customized function key
 examples 179
 panel IDs 181
 function calls
 branch addresses 243
 GOVFNCT values 243
 types 241
 function keys
 customizing 103
 activating new
 definitions 184
 appearance on screen 179
 command 179
 examples 179
 full-screen panel 179
 guidelines 177
 help panel 181
 problems activating 177
 prompt panel 181
 updating function key
 table 177
 user profile modification 184
 window panel 180
 default settings 174
 index on table 177

function keys (*continued*)
 initialization messages 291
 panels 173
 table 176
 authorizing users 184
 creating 176
 entering definitions 177
 maintenance 184
 panel IDs 181

G

GDDM (Graphical Data Display Manager) 153
 ADMADSV defaults
 module 154
 considerations, QMF
 invocation 34
 error messages, printing 292
 printer nicknames 149
 ADMADSV defaults
 module 154
 ADMNNICK
 specification 151
 use of 3
 generate statements
 assembler edit exit routine 203
 generic QMF profile 98
 German NLF 20
 GETMAIN storage problems 73
 global variable
 DSQEC_RERUN_IPROC 84
 migration 319
 global variables
 confirming database
 changes 119
 DSQDC_SHOW_PANID 299
 English support for NLFs 144
 object list, displaying 112
 window IDs 182
 governor exit, assembling 261
 governor exit routine
 assembling 261
 branch table 243
 building module file 262
 cancellation service 259
 CICS control block interface 238
 command processing 243
 control information, storing 258
 description 228
 exit routine information 250
 flow of control 238
 function calls 243
 load library member 262
 passing resource control
 information 244

- governor exit routine (*continued*)
 - performance 244
 - program structure 238
 - resource control table 227
 - RESOURCE_GROUP 104
 - scratchpad area 258
 - sharing 239
 - specifying for resource groups 236
 - think time 242
 - types of function calls 241
- GOVFNCT values for function calls 243
- GRANT statement
 - CONNECT authority 96
 - PUBLIC keyword 110
 - RESOURCE authority 118
 - WITH GRANT OPTION 110
- H**
- Hangeul (NLF) 20
- hardware requirements
 - for NLF 44
 - for QMF 7
- help
 - customizing panel function keys 181, 182
 - panel test during IVP 36
- Help panel test, the IVP 36
- help panels
 - customized function key example 181
 - panel ID 182
- HEX function 295
- I**
- I parameter 84
- ICU (Interactive Chart Utility) 120
- ID
 - for NLFs 19
 - QMF panels 181
 - translation 275
- IMPORT TABLE command
 - creating tables 116
 - default dbspace 325
 - SQL privileges required 108
- incomplete
 - data object and the DSQSIROW parameter 79
- index
 - command synonyms table 163
 - function key table 177
 - Q.OBJECT_DATA 122
 - Q.OBJECT_DIRECTORY table 121
 - Q.OBJECT_REMARKS 123
- index (*continued*)
 - Q.PROFILES table 100
 - Q.RESOURCE_TABLE 231
 - recreating 126
- initial procedures 84
- initialization
 - errors 291
 - location name parameter 90
 - message numbers 299
 - QMF profile values 104
 - REXX program to start QMF 68
 - slow performance 84
 - tracing errors 81
- initialization errors 290
- input area
 - control for formatting 193
 - control for termination 197
- installation
 - checklist 315, 316
 - considerations 18, 19
 - control file, create 25, 45
 - EXEC
 - error messages 29, 47
 - function 28
 - preparation 28, 46
 - restart procedure 29
 - running 29, 46
 - overview 5
 - steps 21, 24, 44
 - NLF 44
 - verification procedure 36
 - worksheet
 - NLF 44
 - QMF 25
- Installation Verification Procedure (see IVP) 38
- insufficient storage message 79
- integer data, edit routine 188
- Interactive Chart Utility (ICU) 120
- interactive use of QMF, specifying through parameter 83
- interface control block
 - assembler edit routine 199, 205
 - COBOL 214
 - DXEGOVA 244
 - DXEXCBA 244
- interrupt handler, QMF 306
- INTVAL column,
 - Q.RESOURCE_TABLE 231
- invocation EXEC
 - GDDM considerations 34
 - parameters 30, 47
 - QMF dialog considerations 32
 - to start QMF 30, 47
 - NLF 47
- invocation EXEC (*continued*)
 - to start QMF 30, 47 (*continued*) QMF 30
- invocation procedure, migration 319
- invoking procedures through program parameters 84
- isolation levels
 - cursor stability 111
 - uncommitted read 111
- ISPEXEC command
 - batch-mode QMF 62
- ISPF (Interactive System Productivity Facility)
 - establishing QMF as dialog 35, 47
 - forms of the PARM operand 61
 - invocation EXEC 35
 - Master Application Menu
 - NLF 48
 - QMF 35
 - menu
 - starting batch-mode QMF 63
 - starting batch QMF session 62
 - starting QMF, CMS 61
 - use of 3
- ISPF/PDF editor 143
- ISPSTART command
 - batch-mode QMF 62
 - considerations when using user edit routines 211, 221
 - environmental considerations 61
 - from CMS 61
 - from READY mode 61
 - passing parameters 31
 - PGM form 61, 62
 - program segment form 62
 - specifying batch-mode 63
- ISPSTART command parameters 31
- Italian NLF 20
- IVP (installation verification procedure)
 - for QMF batch mode NLF 50
- IVP (Installation Verification Procedure)
 - for QMF batch mode
 - authorization required 39
 - QMF 38
 - what it tests 38
 - what the results are 38
 - for QMF interactive mode
 - NLF 48
 - QMF 36

J

Japanese NLF 20

K

K parameter 90

Katakana terminals

setting up DBCS support 90

UCF support 20

Katakana terminals, DBCS

support 20

keywords, reporting problems 310

Korean NLF 20

L

L parameter 75

L2 tracing 81

LENGTH column

(Q.PROFILES) 101

linear procedures in command

synonyms 166

LIST command

ALL keyword 124

customizing 112

list views

rules for creating 115

literals in command synonyms 168

load library member 262

load modules 5

local date/time exits 128

local installation 5

location

name

parameter 90

location window IDs 182

locks on tables 79, 110

log on to QMF

restricting 99

logon procedure

for VM 30

logon procedure for QMF 30

logon to QMF

enabling 97

loop problems, initialization 84

M

M parameter 83

maintenance

command synonym table 170

displaying objects 124

enlarging dspace for

objects 126

function key table 184

listing objects 123

listing tables 128

listing views 128

QMF and database objects 120

message

authorization 108

canceling user activity,

governor 260

insufficient storage 79

printer name 157

printing errors 292

QMF message services 298

row limit exceeded 228

warning, QMF Home panel 291

migrating to QMF Version 7

commands from earlier

releases 323

migration

considerations

31-digit decimal

migration 321

global variables 319

installation 18

invocation procedure 319

multiple releases of

QMF 320

Q.VPROFILE 319

fallback and clean-up 321

installation considerations 18

object compatibility

using Version 7 objects under

earlier releases 322

to a new DB2 for VM level 320

minidisk, distribution and

production 46

create DB2 for VM

DBSPACE(s) 26

create QMF installation control

file 25, 45

QMF installation EXEC 28

MODEL column 104, 122

module file

building 262

DSQUEGV2 262

module level tracing 304

multiple

releases of QMF 320

multiple releases of QMF 320

N

n symbol 19

name

ADMMNICK specification 151

column in control tables 122

translation 275

National Language Feature

(NLF) 19

NEWAPPL operand, starting

QMF 61

nickname

defining multiple printers 152

errors during printing 292

Nihongo (NLF) 20

NLF (National Language Feature)

administering 19

changing in QMF profile 103

command synonyms 166

DBCS printing 90

defined 19

English support 144

governor, sharing 239

multiple profiles 99

NLID 20

QMF profile values 100

release numbers,

ServiceLink 310

TRANSLATION column

(Q.PROFILES) 98

NLIDs, national languages 20

non-SUBSET mode of CMS,

considerations for 62

NONE keyword, tracing 81

nonrecoverable dspace 119

Notices 327

NUCXLOAD command

loading edit routines 192

with DSQQMFE 64

NUMBER column (function key

table) 179

numeric data conversion, edit

routine 188

O

object 95

authorization to use 107

cascading authority 110

control tables 120

deleting 125, 128

displaying 124

enlarging dspace 126

internal representation 121

list

customizing 114

default views 112

window IDs 182

listing 123

maintenance 120

name, command synonym 164

ownership 109

privileges 107

sharing 111, 124

standards for creating 112

storage 73, 74

- object 95 (*continued*)
 - transferring ownership
 - queries, forms,
 - procedures 124
 - tables, views 128
 - types 3
 - OBJECT column (synonyms table) 162, 165
 - object lists
 - customizing with global variables 114
 - OBJECTLEVEL column, QMF control tables 122
 - open enrollment 98
 - OS/2 printers 154
 - output area
 - control for formatting 193
 - control for termination 197
 - overview
 - of installation 5
 - of QMF 3
 - overview of the install process 5
 - OWNER column, QMF control tables 122
 - ownership
 - control tables 95, 97
 - how QMF tracks 121
 - transferring 124, 128
- P**
- page sizes for dbspace 118
 - panel
 - class ID 178
 - confirmation 101, 119
 - customized function keys 173
 - governor prompt 228
 - IDs 181
 - PANEL column (function key table) 178
 - panels
 - class ID 178
 - confirmation 101, 119
 - customized function keys 173
 - governor prompt 228
 - IDs 181
 - parameters
 - NLF invocation 47
 - passed to edit routine 193
 - QMF invocation 30
 - QMF program 67
 - PC printers 153
 - performance 84
 - data retrieval 79
 - governor exit routine 244
 - reports 73, 74, 79
 - performance 84 (*continued*)
 - slow, causes 296
 - table indexes 117
 - using a spill file 75
 - using spill file 78
 - PF keys 103
 - PF_SETTING column (function key table) 179
 - PFKEYS column (Q.PROFILES) 103
 - PGM form of ISPSTART
 - command 61, 62
 - PGM operand, starting QMF 61
 - PL/I edit routine
 - creating a CMS text file 210
 - creating a DSQUEDIT module file 210
 - linkage 204
 - returning to the caller 205
 - Portuguese NLF 20
 - post-installation cleanup
 - NLF 50
 - QMF 40
 - preprocess QMF modules
 - for sample table insert program 28
 - PRINT command 149
 - routing to named destinations 149
 - print data output 33
 - Print function key
 - defining a synonym for 156
 - PRINT TABLE command, SQL
 - privileges required 108
 - printer
 - control keywords (PRINTCTL) 153
 - DBCS support 90
 - DOS 153
 - Family 1 150
 - Family 2 151
 - Family 3 152
 - Family 4 152
 - length of output 101
 - multiple addresses 151
 - nicknames 149
 - OS/2 154
 - PROCOPT parameter 153
 - width of output 101
 - PRINTER column (Q.PROFILES) 103
 - printing
 - enabling users 147
 - errors 292
 - QMF vs. GDDM 149
 - summary 148
 - printing (*continued*)
 - to PC printers 153
 - trace file 304
 - updating user profiles 156
 - private dbspace 118
 - privileges 105
 - commands 108
 - database objects 107
 - for table editor 109
 - GRANT statement 110
 - granting to all users (PUBLIC) 110
 - queries 109
 - REVOKE statement 110
 - revoking 110
 - privileges required for QMF tasks 108
 - problem reporting 309
 - procedure
 - batch-mode QMF 83
 - initial (DSQSRUN) 84
 - invoking using program parameters 84
 - procedures
 - controlling user access 111
 - displaying 124
 - internal stored format 122
 - listing 123
 - maintaining objects 122, 128
 - printing 148
 - using in command synonyms 166
 - PROCOPT parameter, printing 153
 - production minidisk
 - create DB2 for VM DBSPACE(s) 26
 - create QMF installation control file 25, 45
 - NLF 45
 - QMF 25
 - QMF installation EXEC 28, 46
 - NLF 46
 - QMF 28
 - QMF invocation EXEC 30, 47
 - NLF 47
 - QMF 30
 - profile
 - CASE setting, customized function keys 179
 - command synonyms 168
 - creating 97, 98
 - default values 100
 - deleting 99, 106
 - function key customization 184
 - initialization search order 104

- profile (*continued*)
 - maintenance 120
 - multiple (NLFs) 99
 - print parameters 156
 - printing 148
 - Q user ID 95, 97
 - SET PROFILE command 105
 - sizing printed reports 156
 - updating 105, 106
- program
 - access packages 18
 - parameters 30
 - requirements
 - NLF 44
- program access modules, QMF 18
- program access packages, QMF 18
- program parameters 84
 - DSQSBSTG (maximum storage for reports) 73
 - DSQSCMDn 68
 - DSQSDBCS (choose Kanji) 90
 - DSQSDBNM (initial location name) 82
 - DSQSDEBUG (set trace) 81
 - DSQSIROW (control rows fetched) 79
 - DSQSMODE (choose mode) 83
 - DSQSPILL (spill file) 75
 - DSQSRSTG (reserve storage) 74
 - DSQSRUN (initial procedure) 84
 - DSQSRUN (invoke a procedure) 84
- program products
 - required
 - NLF 44
- program segment
 - running QMF as an ISPF dialog 72
 - start QMF as an ISPF dialog 59
- program segment form of ISPSTART command 62
- program segments
 - program parameter, starting QMF 72
- prompt panel
 - customized function key
 - example 181
 - panel ID 182
- prompted query
 - printing 148
 - SQL privileges 109
 - window IDs 183
- public dbspace 118
- PUBLIC keyword 110

Q

- Q.COMMAND_SYNONYMS
 - table 16
 - storage structure 16
- Q.DSQ_RESERVED control table 325
 - table structure 16
- Q.DSQ_RESERVED table 16
- Q.DSQIOLST_AU_VIEW catalog view 17
- Q.DSQIOLST_QT_VIEW catalog view 17
- Q.DSQIOLST_TB_VIEW catalog view 17
- Q.ERROR_LOG control table 308, 325
 - table structure 16
- Q.ERROR_LOG table 16
- Q.OBJECT_DATA control table
 - default dbspace 325
 - enlarging dbspace 126
 - table structure 16
- Q.OBJECT_DATA table 16
- Q.OBJECT_DIRECTORY control table
 - default dbspace 325
 - enlarging dbspace 126
 - maintaining objects 121, 128
 - table structure 16
- Q.OBJECT_DIRECTORY table 16
- Q.OBJECT_REMARKS control table
 - default dbspace 325
 - enlarging dbspace 126
 - maintaining objects 123, 128
 - table structure 16
- Q.OBJECT_REMARKS table 16
- Q.PROFILE, migration 319
- Q.PROFILES control table
 - adding user profiles 98
 - default dbspace 325
 - deleting user profile 99
 - table structure 16, 100
 - updating 105
 - updating PFKEYS field 184
 - updating RESOURCE_GROUP field 229
 - updating SYNONYMS field 168
 - user modifications 105
- Q.PROFILES table 16
- Q.RESOURCE_TABLE control table
 - default dbspace 325
 - governor exit routines 231
- Q.RESOURCE_TABLE table 16
- Q.RESOURCE_VIEW, governor 232
- Q user profile 95, 97

- QBE query
 - printing 148
 - SQL privileges 109
- QMF
 - error messages, printing 292
 - establishing user support 95, 145
 - interrupt handler 306
 - profile 95
 - session 97, 259
- QMF (Query Management Facility)
 - application procedures
 - install in base 49
 - install in NLF 37
 - control tables 16
 - data files 32
 - dbspace requirements 14
 - dialog on ISPF menu
 - NLF 48
 - QMF 35
 - establishing as an ISPF dialog
 - NLF 47
 - QMF 35
 - hardware requirements 7
 - invocation 30
 - objects 3
 - overview of 3
 - program
 - access modules 18
 - parameters 30
 - required files 32
 - starting
 - NLF 47
 - QMF 30
 - storage requirements 14
- QMF dialog and invocation 32
- QMF edit command 143
- QMF installation EXEC 46
- QMF program access modules 18
- QMF program access packages 18
- queries
 - changing default type 102
 - controlling user access 111
 - deleting 125
 - displaying 124
 - GRANT 110
 - internal stored format 122
 - listing 123
 - printing 149
 - required privileges 109
 - transferring object ownership 124
- queries, QMF sample
 - installing
 - NLF 49

- queries, QMF sample (*continued*)
 - QMF 37
 - query
 - changing default type 102
 - deleting
 - SQL statements 125
 - displaying 124
 - internal stored format 122
 - listing
 - SQL statements 123
 - required privileges 109
 - sample
 - installing 37, 49
 - storage for execution 73, 74
 - Query Management Facility (see QMF) 7
- R**
- R parameter 74
 - release numbers 310
 - RELOAD dbspace command 126
 - REMARKS column 123
 - remote installation 5
 - remote unit of work
 - access to objects 273
 - access to tables 274
 - administrator access 275
 - code page
 - APPLDATA 270
 - special characters 270
 - support 270
 - commands 269
 - connecting to local database 271
 - connecting to remote
 - database 271
 - creating command synonym
 - tables 267
 - customizing a remote database
 - connection 263
 - database connection, type 264
 - DB2 connections 266
 - DB2 for VM connections 266
 - defined 13
 - deleting users 275
 - distributed unit of work 265
 - DPRE command 269
 - DRDA 263
 - enabling user access 271
 - function key tables 269
 - location name 272
 - naming conventions 267
 - NLF 270
 - overview 4
 - preparing the locations 267
 - QMF governor control
 - tables 270
 - remote unit of work (*continued*)
 - remote location 265
 - SQL authorization ID 271
 - SQL errors 274
 - three-part name
 - directing a query 265
 - specifying a table or
 - view 265
 - user profile 271
 - report
 - slow performance 79
 - reports
 - binary data 295
 - data formats 187
 - printing 148
 - Q.ERROR_LOG table 308
 - slow performance 297
 - storage
 - allocating extra 75
 - width/length 101, 156
 - requirements for QMF
 - database 13
 - DBSPACE 14
 - files 32
 - hardware 7
 - MAINT machine 14
 - storage 14
 - reserve storage, specifying through
 - parameters 74
 - RESET DATA command 296
 - resource
 - controlling 228
 - governor exit routine 228
 - group 99
 - default (SYSTEM) 231
 - limiting 228
 - profile 104
 - ownership 95, 97
 - passing control information 244
 - problem log 308
 - profile management 99
 - RESOURCE authority 118
 - RESOURCE_GROUP column 104, 231
 - RESOURCE_OPTION column 231
 - restricted access to QMF 99
 - RESTRICTED column
 - changing value to NO 124
 - defined 122
 - restrictions, environmental, when
 - starting QMF 61
 - return codes
 - issued when QMF started
 - without ISPF 64
 - return codes, SQL 300
 - REVOKE statement 110
 - REXX
 - DSQSCMDn program
 - parameter 68
 - QMF initialization program 68
 - REXX initialization program (DSQSCMDE) 69
 - routines, edit
 - DATE, TIME, and TIMESTAMP
 - data 189, 190
 - rows, controlling number
 - retrieved 228
 - rows from database, controlling
 - number fetched 79
 - rules
 - command synonyms 163
 - customizing function keys 177
 - rules for command synonyms 163, 168
 - rules for customizing function
 - keys 177
 - RUN command
 - command synonym 165
 - initial procedure 84
 - SQL privileges required 108
- S**
- sample
 - queries
 - installation 37
 - installing 49
 - storage requirements 14
 - tables
 - creating 29
 - DBSPACE for 17
 - deleting 29
 - insert program 28
 - predefined dbspaces 325
 - space requirements 14
 - what they are 17
 - sample queries, QMF
 - installing
 - NLF 49
 - QMF 37
 - storage requirements for 14
 - sample tables 325
 - delete 29
 - insert program 28
 - install 29
 - QMF 17
 - storage requirements for 14
 - SAVE
 - DATA keyword 108
 - SHARE parameter 111
 - SQL privileges required 108

- SAVE (*continued*)
 - TABLE keyword 108
- SAVE command
 - default dbspace, tables 325
 - performance 79
- SCOPE resource option 231
- scratchpad area
 - edit routines 214
 - governor exit routine 258
- SEQ column 122
- ServiceLink 309
- session 97
 - cancellation service 259
 - customizing
 - user profile 97
 - interactive vs. batch 83
- SET PROFILE command 105
- SFS (Shared File System)
 - directories 3
- share locks on data 79
- shift characters 223
- Simplified Chinese NLF 20
- small integer data, edit routine 188
- Software Support Facility (SSF) 309
- SPACE column (Q.PROFILES) 102
- Spanish NLF 20
- special characters 270
- specifying interactive or batch mode 83
- spill file
 - allocating 75
 - estimating size 76
 - performance problems 75, 78
 - sample calculations 78
 - specifying characteristics through parameters 75
- SQL
 - ID 98
 - attached to user profile 101
 - command synonym table 171
 - how QMF stores 122
 - Q 95, 97
 - privileges 98
 - for prompted, QBE queries 109
 - for QMF commands 108
 - for table editor 109
 - granting 109
 - Q.PROFILES table update 105
 - revoking 109
 - table and view access 107
 - queries, printing 148
 - return codes 300
- SQL (*continued*)
 - statement 98
 - ACQUIRE DBSPACE 118
 - ALTER DBSPACE 127
 - CREATE TABLE 116
 - GRANT 110
 - INSERT (new user profile) 98
 - REVOKE 110
 - UPDATE 105
 - SQLADBSpace DB(dbname)
 - command 27
 - SQLSTART DB(dbname)
 - command 27
 - SSF (Software Support Facility) 309
 - STAE exit 294
 - START command
 - DSQSCMDn parameter 68
 - starting QMF
 - as an ISPF dialog 59
 - batch mode, base QMF environment
 - from an ISPF menu 63
 - independent of ISPF 64
 - environmental considerations
 - with ISPSTART 61
 - establishing a database connection 57
 - from an ISPF menu
 - adding an option to the menu 59
 - without an EXEC 61
 - independent of ISPF
 - commands for 64
 - return codes issued 64
 - independent of ISPF, DSQQMFE 61
 - passing parameters
 - PARM operand 61
 - QMF profile initialization 104
 - QMF program 57
 - REXX initialization program 68
 - sample definition for the ISPF menu 59
 - table lock failure 110
 - through ISPSTART 61
 - STAX exit 306
 - storage
 - combinations of values for DSQSBSTG and DSQSRSTG 75
 - data from edit routine 193
 - dbspace
 - calculating size 118
 - increasing size 126
- storage (*continued*)
 - insufficient storage prompt 79
 - limiting users' storage 73, 74
 - reports
 - allocating extra (spill file) 75
 - setting for reports 73, 74
 - specifying reserve storage through parameters 74
 - spill file 75
 - storage requirements
 - for DB2 for VM 13
 - QMF catalog views 17
 - QMF control tables 16
 - QMF DBSPACE 14
 - QMF program access modules 18
 - QMF sample tables 17
 - SUBSET mode of CMS, considerations for 62
 - SUBTYPE column, QMF control tables 122
 - Swedish NLF 20
 - Swiss French NLF 20
 - Swiss German NLF 20
 - SYNONYM_DEFINITION column 165
 - SYNONYMS column (Q.PROFILES) 103
 - synonyms for QMF commands 159, 163
 - abbreviations 170
 - activating for users 168
 - creating synonyms table 162
 - index 162
 - initialization messages 291
 - object name 164
 - problems activating 164
 - quotation marks 168
 - synonym definition 165
 - syntax 168
 - table maintenance 170
 - using variables 166
 - verb 164
 - SYSTABAUTH system table 106
 - system
 - error messages 300
 - printing errors 292
 - tables, DB2 for VM 127
 - SYSTEM profile
 - changing default values 106
 - deleting 99
 - SYSTEM resource group 231
 - system tables
 - DB2 for VM 106
 - SYSUSERAUTH system table 106

T

T parameter 81
Table Editor
 confirmation panels 119
 SQL privileges required 109
tables
 command synonym 162
 concurrent editing 110
 control tables 98
 controlling access 107
 creating 116
 DB2 for VM system 127
 deleting 128
 enlarging dbspaces 126
 function keys 173
 indexes 117
 listing 128
 locks 110
 maintenance 127
 printing 148
 QMF control tables 98
 resource control, governor
 exit 231
 transferring ownership 128
tables, control
 QMF 7
tables, sample
 (see sample tables) 7
tailor the QMF invocation EXEC
 NLF 47
 QMF 30
tailoring the QMF invocation
 EXEC 30, 47
terminal
 changing case 101
 DBCS data support 90
 UCF support for Katakana 20
termination calls, edit routine 197
think time 242
TIME data 190
TIMESTAMP data 190
 information, handling 189
TOFAM keyword, ADMMNICK
 specification 151
toggle switch, governor exit 231
trace
 data
 level of detail 81
 dump output 32
 facility
 file allocation 300
 starting 81, 301
 stopping 305
 level of detail 102
 message logging 291

trace (*continued*)
 viewing data 304
TRACE column (Q.PROFILES) 102
trace dump output 32
trace file
 allocating 301
 displaying 304
 printing 304
tracing
 module level 304
transferring object ownership 124
translation
 of user IDs 275
TRANSLATION column
 (Q.PROFILES) 103
troubleshooting
 abend handling 305
 aids 298
 common problems 290
 diagnostic aids 298
 GDDM errors 292
 initialization errors 290
 message support 298
 performance problems 296
 printing errors 292
 problem reporting 309
 Q.ERROR_LOG table 308
 SQL return codes 300
 symptoms 298
 system error messages 300
 trace facility 300
TYPE column, QMF control
 tables 122

U

U edit codes, forms 188, 195
 defined 188
 input area 195
UCF (Uppercase Feature) 20
UDN edit code 191
uncommitted read 111
UNLOAD dbspace command 126
Uppercase Feature (UCF) 20
user
 adding new 98
 authorization for objects 107
 edit routines 187
 limiting resources 227
 objects 123
 profiles 95
 support 95
 creating tables 116
 object access 107
 profile and object
 maintenance 120

user edit routines
 creating a DSQUEDIT module
 file
 in assembler 203
 in PL/I 210
 in VS COBOL II 220, 222
 DATE data 189, 190
 handling different codes 195
 specific languages
 PL/I 203, 209
 PL/I (with LE) 211
 VS COBOL II 219
 TIME data 189, 190
 TIMESTAMP data 189, 190

V

V edit codes, forms 188, 195
 defined 188
 input area 195
variables
 in synonym definitions 166
 passing using DSQSRUN
 parameter 86
 using &ALL 166
VERB column (synonyms
 table) 162, 164
views
 controlling access 107
 deleting 128
 listing 128
 maintenance 127
 object lists, customizing 112
 privileges for queries 109
 privileges for table editor 109
 Q.RESOURCE_VIEW, governor
 exit 232
 recreating 127
virtual storage
 requirements
 VM 12
virtual storage requirements 12
VSS edit code 191

W

warning messages 291
 where QMF objects reside 5
WIDTH
 column in Q.PROFILES 101
WIDTH column (Q.PROFILES) 101
window panels
 customized function key
 examples 180
 IDs 181
worksheet for installation
 QMF 25

workstation database servers
planning 20

Readers' Comments — We'd Like to Hear from You

Query Management Facility™
Installing and Managing QMF on
VM/ESA
Version 7

Publication No. GC27-0720-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

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

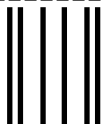
Phone No.



Fold and Tape

Please do not staple

Fold and Tape



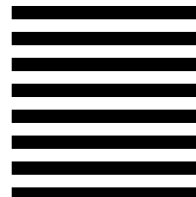
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM CORPORATION
Department HHX/H3
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.



Fold and Tape

Please do not staple

Fold and Tape



Program Number: 5697-F42



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC27-0720-00



Spine information:



Query Management Facility™

Installing and Managing QMF on VM

Version 7