Query Management Facility™

Installing and Managing QMF for VSE/ESA

Version 7

GC27-0721-00

Query Management Facility™

Installing and Managing QMF for VSE/ESA

Version 7

GC27-0721-00

Note!
Before using this information and the product it supports, be sure to read the general information under "Appendix E. Notices" on page 277.

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, (DB2 for VM and VSE), 5697-F42, (VSE environment only), and to any subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces and makes obsolete the previous publications, GC26-9574-00.

Contents

The QMF Library ix

About This Book xi
Terminology in This Guide xi
Locating Prerequisite Documentation xi
How to Use This Book xi
What You Should Know before You Begin . . xii
How National Language Feature Information is Represented xiii

Part 1. Installing QMF on VSE/ESA 1

Chapter 1. Before You Begin 3
Hardware 3
Prerequisite Software 3
QMF Storage Requirements 4
Apply Service 6
Check Space Requirements 6
Library Space 6
VSAM Catalog 6
dbspace 7
Check Your CICS Partition Size 7
Partition Size for Installation 8
Other Planning Considerations 8
Tailoring GDDM for QMF and CICS . . . 8
Running DB2 Guest Sharing 8
Customizing DB2 for Double-Byte Character Support 9
Installation Overview 9
Base Installation 9
Installing Language Support 10
CICS Tailoring 11

Chapter 2. Tailoring Your Installation . . . 13
Punch Members to an Editor 13
Install QMF Base 13
Catalog the Initialization Procedure . . . 13
Run the QMF Installation Job 15
Install QMF Base into DB2 Database . . . 16
Tailor QMF for NLF 18
Install NLF 18
Install QMF NLF into SQL Database . . . 19
Link-Edit Jobs for QMF 20
Link Jobs for NLF 21
Tailor CICS 21
Modify the DFHFCT and DFHDCT . . . 22
Define QMF Programs and Transactions to CICS 23
Run CEDA 24
Modify the DFHPCT and DFHPPT . . . 25
Modify the CICS Startup Job 25
Install QMF for VSE/ESA into a Second CICS System 26

Chapter 3. Installing QMF into Remote Database Servers. 27
Installing QMF Version 7 into a DB2 Universal Database® Remote Server 27
Prerequisites 27
Punch Members to an Editor 27
Installation steps 27
Installing QMF Version 7 into a DB2 for AS/400® Server 29

Chapter 4. Run the Installation Verification Procedure (IVP) 31
Before You Start QMF. 31
Start and Test QMF 31
Run an IVP for NLF 34
What if It Didn't Work? 34

Chapter 5. How to Maintain QMF 37
Adding New Components 37
Adding GDDM-PGF 37
Adding QMF to Another DB2 Database . . 37
Migrating to New Releases of DB2, CICS, or GDDM 37
Binding QMF 7.1 Packages at a Remote Server 37
Replacing Existing Components 38
Re-installing QMF 38
Re-installing an NLF 38
Applying Service Updates 39

Part 2. Managing QMF for VSE/ESA. 43

Chapter 6. Starting QMF 49

Before You Start QMF 49
Quick Start 49
Add QMF to the VSE/ESA Function Selection Menu 50
Starting QMF from a Cleared CICS Screen . . 51
Starting QMF from a CICS Application . . . 51
Starting a Noninteractive Session 52
Starting an Interactive Session 52

Chapter 7. Customizing Your Start Procedure 55
Quick Start 55
Customizing Report Storage and Report Performance 56
Adjusting GETVIS Storage Used for Report Data (DSQSBSTG) 56
Acquiring Extra Temporary Storage (DSQSPILL) 58
Specifying the Name of Spill Storage (DSQSSPQN). 62
Controlling the Number of Report Rows Retrieved for Display (DSQSIROW) . . . 63
Tracing QMF Activity at the Start of a Session 64
Setting the Level of Trace Detail (DSQSDBUG) 65
Specifying the Type of CICS Storage for Trace Data (DSQSDBQT). 66
Specifying the Name of CICS Storage for Trace Data (DSQSDBQN) 66
Controlling Initial Activities during a Session 67
Connecting to the Database (DSQSUSER) 67
Starting a Noninteractive QMF Session (DSQSMODE) 69
Naming a Procedure to Run When QMF Starts (DSQSRUN) 69
Setting Printing for Double-Byte Character Set Data (DSQSDBCS) 75

Chapter 8. The QMF Session Control Facility 77
Installing or Removing Q.SYSTEM_INI . . . 77
When Does the Q.SYSTEM_INI Procedure Run? 77
Using Q.SYSTEM_INI. 77
Example Shipped with QMF 77
User Session Procedure Example 78
Procedure that Displays an Object list . . 79
Security and Sharing Session Procedure. . . 80
Diagnosis Considerations 80

Chapter 9. Establishing QMF Support for End Users 81
Quick Start 81
Creating User Profiles to Enable User Access to QMF 82
Using the Q User Profile, a Special QMF Profile 82
Establishing a Profile Structure for Your Installation 82
Adding a New User Profile to the Q.PROFILES Table 83
Preventing Users Without Unique Profiles from Using QMF 84
Reading the Q.PROFILES Table 84
Providing the Correct Profile for the User's Operating Environment 88
Storing Profiles in VM DB2 in a Guest-Sharing Environment. 89
Updating User Profiles 89
Deleting Profiles from the Q.PROFILES Table 91
Controlling Access to QMF and Database Objects. 92
SQL Privileges Required to Access Objects 92
Granting and Revoking SQL Privileges . . 94
Sharing QMF Objects with Other Users . . 95
Allowing Uncommitted Read 96
Setting Standards for Creating Objects . . 96
Customizing a User's Database Object List. . 97
Using the Default Object Lists 97
Changing the Default List 98
Object List Storage Requirement 100
Enabling Users to Create Tables in the Database. 100
Choosing and Acquiring a dbspace for the User 102
Granting a User DB2 RESOURCE Authority 102
Enabling Users to Confirm Table Changes Before They are Made 103
Enabling Users to Support a Chart 104
Maintaining QMF Objects Using QMF Control Tables 104
Reading the Q.OBJECT_DIRECTORY Table 104
Reading the Q.OBJECT_DATA Table . . 105
Reading the Q.OBJECT_REMARKS Table 106
Listing QMF Queries, Forms, and Procedures 107

Displaying QMF Queries, Forms, and Procedures 107
Transferring Ownership of Queries, Forms, and Procedures 108
Deleting Obsolete Queries, Forms, and Procedures 108
Enlarging the dbspace for the QMF Object Control Tables 109
Maintaining Tables and Views Using DB2 System Tables 110
Listing Tables and Views 110
Transferring Ownership of a Table or View 111
Deleting a Table or View from the Database 111
Enabling English Support in an NLF Environment 111
Using Global Variables to Define the Currency Symbol 112

Chapter 10. Enabling Users to Print Objects 113
Quick Start 113
Printing Objects 114
Deciding Whether to Use QMF or GDDM Services for Printing 115
Using GDDM services to Handle Printing 116
Choosing a GDDM Nickname for Your Printer 116
Updating the GDDM Defaults Module (ADMADFC) with the Nickname . . . 120
Linking the Nickname with a Physical Device 121
How QMF Interfaces with your GDDM Nickname 122
Using QMF Services to Handle Printing . . 123
Choosing Between Temporary Storage Queues and Transient Data Queues . . . 123
Using the PRINT Command to Route Output to Queues. 123
Using Global Variables to Define Queues for Printing 124
Printing to VSE POWER using QMF . . 124
Updating User Profiles to Enable GDDM Printing 130

Chapter 11. Customizing QMF Commands 133
Quick Start 133
Creating the Command Synonym Table . . 133
Entering Command Synonym Definitions into the Command Synonym Table 135
Choosing a Verb 136
Choosing an Object Name 137
Choosing the Synonym Definition . . . 137
Activating the Synonyms 141
Minimizing Maintenance of Command Synonym Tables 142
Assigning One Synonym Table to All Users 142
Assigning Views of a Synonym Table to Individual Users 143

Chapter 12. Customizing QMF Function Keys 145
Quick Start 145
Choosing the Keys You Want to Customize 145
Default Keys on Full-Screen Panels . . . 146
Default Keys on Window Panels 147
Creating the Function Key Table 148
Entering Your Function Key Definitions into the Table. 149
Linking a Command with a Function Key 149
Labeling the Function Key and Positioning It on the Screen 150
Examples of Key Definitions 151
Identifying the Panel You Want to Customize 153
Full-Screen Panel Identifiers 153
Window Panel Identifiers 153
Activating New Function Key Definitions 156

Chapter 13. Creating Your Own Edit Codes for QMF Forms 159
Quick Start 159
Choosing an Edit Code 160
Calling Your Exit Routine to Format the Data 161
Passing Information To and From the Exit Routine 164
Fields of the Interface Control Block . . 165
Fields That Characterize the Input Area 166
Fields That Characterize the Output Area 167
Passing Control to the Exit Routine When QMF Terminates 167
Writing an Edit Routine in High-Level Assembler (HLASM). 168
How an HLASM Edit Routine Interacts with CICS 168
How an HLASM Edit Routine Interacts with QMF 169

Translating Your Program 172
Assembling Your Program 173
Link-Editing Your Program 173
Example JCL Statements 173
Defining the Edit Exit Phase to CICS . . 175
Writing an Edit Routine in VS COBOL II or COBOL for VSE/ESA 175
Using Literal Values in a COBOL Program 176
How a COBOL Edit Routine Interacts with CICS 176
How a COBOL Edit Routine Interacts with QMF 178
Translating Your Program 183
Compiling Your Program 184
Link-Editing Your Program 184
Example JCL Statements 184
Defining the Edit Exit Phase to CICS . . 186
Writing an Edit Routine in PL/I 186
Writing an Edit Routine in PL/I for CICS 186
How a PL/I Edit Routine Interacts with CICS 187
Translating Your Program 188
Link-Editing Your Program 188
CICS Program Definition 188
Example JCL Statements 188
How a PL/I Edit Routine Interacts with QMF 190
Handling Double-Byte Character Set Data 196
Edit Codes for DBCS Data 196
What the Edit Routine Receives 196
Ensuring the Edit Routine Returns the Right Results 197

Chapter 14. Controlling QMF Resources Using a Governor Exit Routine 199
Quick Start 199
Using the IBM-Supplied Governor Exit Routine 199
Activating the Default Limits for Number of Rows Retrieved 200
How a Governor Exit Routine Controls Resources 202
Defining Your Own Resource Limits . . 204
Creating your own Resource Control Table 206
Modifying the IBM-Supplied Governor Exit Routine or Writing Your Own. 208
Program Components of the Governor Exit Routine 208
How CICS Interfaces with the Governor Exit Routine 209
How and When QMF Calls the Governor Exit Routine. 211
Passing Resource Control Information to the Governor Exit. 215
Storing Resource Control Information for the Duration of a QMF Session 227
Canceling User Activity. 228
Providing Messages for Canceled Activities 228
Translating, Assembling, and Link-Editing Your Governor Exit Routine 229
Translating Your Governor Exit Program for CICS 229
Assembling Your Governor Exit 230
Link-Editing Your Governor Exit Routine 230
Example JCL Statements 230

Chapter 15. Troubleshooting and Problem Diagnosis 233
Quick Start 233
Troubleshooting Common Problems. . . . 234
Handling Initialization Errors 234
Handling Warning Messages 234
Handling GDDM Errors During Printing 235
Handling QMF Errors During Printing 237
Handling Display Errors 237
Solving Slow Performance Problems . . 238
Determining the Problem Using Diagnosis Aids 240
Choosing the Right Diagnosis Aid for the Symptoms 240
Diagnosing Your Problem Using QMF Message Support 240
Using the QMF Trace Facility 242
Using CICS Diagnostic Facilities 248
Using Error Log Reports from the Q.ERROR_LOG Table 250
Reporting a Problem to IBM 251
Using ServiceLink to Search for Previously Reported Problems 251
Working with Your IBM Support Center 254

Part 3. Appendixes 255

Appendix A. QMF for VSE/ESA Version 7 Product Limitations 257

Appendix B. Migrating from QMF VSE V1 to Version 7 259
Quick Start 259
Migrating Queries, Forms, and Procedures 260
Starting the Migration Utility 261
Specifying the Type of Object 262
Specifying the Owner of the Object . . . 263
Specifying the Name of the Object . . . 264
Migrating Version 1 Objects to Version 7 Control Tables 264
Viewing Messages from the Migration 265
Migrating User Profiles 265
Deleting QMF VSE V1 After You Migrate Your Objects 266
Deleting QMF VSE V1 from the VSE Sublibrary 266
Deleting QMF VSE V1 Information from the History File 267
Deleting QMF VSE V1 Objects from the VSE DB2 Database 268
Deleting QMF VSE V1 Definitions from the CICS System Tables 270

Appendix C. How QMF and GDDM Programs Are Defined to CICS 271
How QMF Programs Are Defined to CICS/VSE 271
Resident QMF Programs 271
How Nonresident Programs Affect Performance 271
Loading QMF to the 31-Bit Shared Virtual Area 272
How GDDM Definitions Are Loaded During QMF Installation 273
How Nonresident GDDM Programs Affect QMF 273
How Chart Formats Are Defined. 274
Adding Charting Function After QMF Installation 274
Using Transaction Routing to Control Resource Use 274

Appendix D. QMF Control Tables and dbspaces Used by QMF 275

Appendix E. Notices 277
Trademarks 280

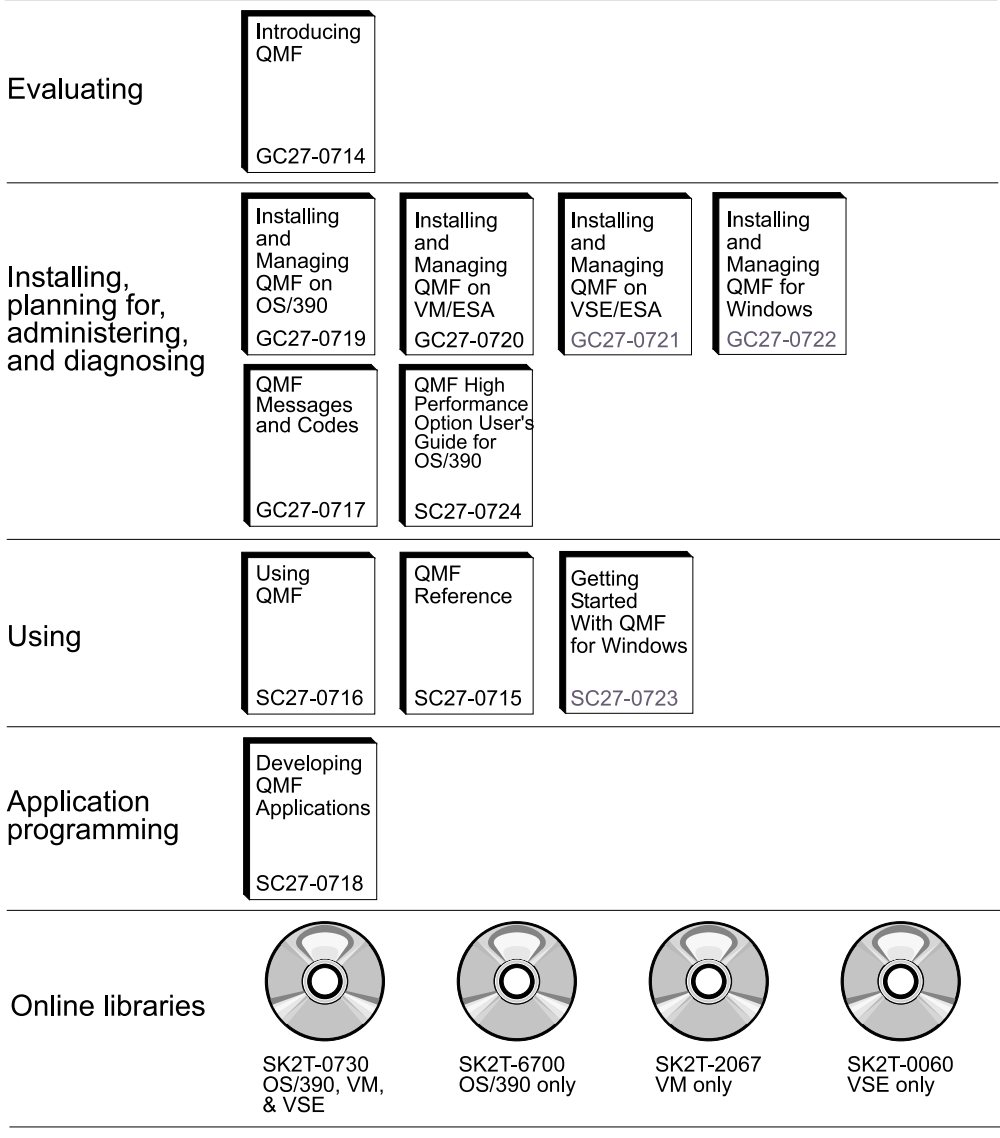
Bibliography 281
APPC Publications 281
CICS Publications. 281
COBOL Publications 282
DATABASE 2 Publications 282
DCF Publications 283
DRDA Publications 283
DXT Publications 283
Graphical Data Display Manager (GDDM) Publications. 283
HLASM Publications. 283
ISPF/PDF Publications 283
OS/390 Publications 284
PL/I Publications 284
REXX Publications 284
ServiceLink Publications 284
VM Publications 285
VSE Publications 285

Glossary of Terms and Acronyms . . . 287

Index 301

The QMF Library

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



About This Book

This book is intended to help database administrators and systems programmers install and maintain the Query Management Facility (QMF) product in the Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA)™ environment.

You should be familiar with VSE/ESA, VSE/VSAM, CICS/VSE®, GDDM/VSE, and DB2 for VSE. It is also helpful to know the Interactive Interface of VSE/ESA, because it is used as the primary installation tool.

To install QMF, you must first install and test CICS®, DB2 for VSE or VM™, and the Graphical Data Display Manager (GDDM)®.

Terminology in This Guide

To keep the installation task as simple as possible, many of the full IBM product names and titles are shortened. Each product is referred to by its generic, rather than specific, name. For example, VSE/ESA is just VSE; VSE/VSAM is VSAM; CICS/VSE is CICS; and DB2 for VSE or DB2® for VSE is DB2.

Locating Prerequisite Documentation

In addition to this guide, keep the *QMF Program Directory* and the *QMF Preventive Service Planning (PSP)* document ready during the installation.

The *QMF Program Directory* documents changes to the install process after this book is published. You'll find it packed in the shipping carton with your installation tape. The PSP document also has late-breaking information about installation. PSP documentation is described further in "Apply Service" on page 6.

For a complete list of QMF publications, see "The QMF Library" on page ix.

How to Use This Book

The management tasks in this book assume QMF was installed according to the installation procedures in "Part 1. Installing QMF on VSE/ESA" on page 1. If you decide to customize some default aspects of the installation, see "Appendix C. How QMF and GDDM Programs Are Defined to CICS" on page 271.

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, run the Installation Verification Procedure (IVP) to ensure that QMF is properly installed and configured for your site's needs. The IVP is the final step of the QMF installation process presented in "Part 1. Installing QMF on VSE/ESA" on page 1.

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

Each chapter in "Part 2. Managing QMF for VSE/ESA" on page 43 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.

What You Should Know before You Begin

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

- Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA), an operating system that supports QMF and its related products.
- Customer Information Control System (CICS), a general-purpose data communication and online transaction processing system. CICS/VSE provides the interface between QMF and VSE/ESA.
- Graphical Data Display Manager (GDDM), which makes it possible for QMF to display panels on the user's screen and create charts.
- DATABASE 2 (DB2)™, the database where users' objects are stored.
- High-Level Assembler (HLASM) 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 if you plan to create your own edit codes for QMF forms.
- VS COBOL II or COBOL for VSE/ESA, which you might use if you plan to create your own edit codes for QMF forms.
- PLI for VSE/ESA, which you might use if you plan to create your own edit codes for QMF forms.

Publications that discuss these products are listed in the bibliography at page281.

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 ix and the back cover of this book.

How National Language Feature Information is Represented

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

NLFs 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. For more information about NLF installation, see "Part 1. Installing QMF on VSE/ESA" on page 1.

All tasks discussed in this book can be performed for the base QMF product (English language) and for any NLF. 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**.

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 3 on page 10 that matches the NLF you installed. The table also shows the names by which QMF recognizes each language.

Part 1. Installing QMF on VSE/ESA

Chapter 1. Before You Begin 3
Hardware 3
Prerequisite Software 3
QMF Storage Requirements 4
Apply Service 6
Check Space Requirements 6
Library Space 6
VSAM Catalog 6
dbspace 7
Check Your CICS Partition Size 7
Partition Size for Installation 8
Other Planning Considerations 8
Tailoring GDDM for QMF and CICS . . . 8
Changing GDDM 2.3 Default Parameters 8
Run the Installation Verification Procedure (IVP) for GDDM 8
Running DB2 Guest Sharing 8
Customizing DB2 for Double-Byte Character Support 9
Installation Overview 9
Base Installation 9
Optional Job 10
Installing Language Support 10
NLF Install Process. 11
CICS Tailoring 11

Chapter 2. Tailoring Your Installation . . . 13
Punch Members to an Editor 13
Install QMF Base 13
Catalog the Initialization Procedure . . . 13
Run the QMF Installation Job 15
Install QMF Base into DB2 Database . . . 16
Tailor QMF for NLF 18
Install NLF 18
Install QMF NLF into SQL Database . . . 19
Link-Edit Jobs for QMF 20
Link Jobs for NLF 21
Tailor CICS 21
Modify the DFHFCT and DFHDCT . . . 22
Modify DFHFCT 22
Modify DFHDCT 22
Define QMF Programs and Transactions to CICS 23
Update the CSD. 23
Run CEDA 24
Modify the DFHPCT and DFHPPT . . . 25
Modify the DFHPCT 25
Modify the DFHPPT 25
Modify the CICS Startup Job 25
Install QMF for VSE/ESA into a Second CICS System 26

Chapter 3. Installing QMF into Remote Database Servers 27
Installing QMF Version 7 into a DB2 Universal Database® Remote Server 27
Prerequisites 27
Punch Members to an Editor 27
Installation steps 27
Installing QMF Version 7 into a DB2 for AS/400® Server 29

Chapter 4. Run the Installation Verification Procedure (IVP) 31
Before You Start QMF 31
Start and Test QMF 31
Run an IVP for NLF 34
What if It Didn't Work? 34

Chapter 5. How to Maintain QMF 37
Adding New Components 37
Adding GDDM-PGF 37
Adding QMF to Another DB2 Database . . 37
Migrating to New Releases of DB2, CICS, or GDDM 37
Binding QMF 7.1 Packages at a Remote Server 37
Replacing Existing Components 38
Re-installing QMF 38
Re-installing an NLF 38
Applying Service Updates 39
Replacing Text Decks or Phases . . . 39
Updating the QMF Panel File 39
Updating QMF GDDM Maps 40
Updating QMF SQL Packages 41

Chapter 1. Before You Begin

This chapter helps you plan for QMF installation. The key to success is having adequate resources. The following sections describe the hardware and software requirements, your planning considerations, and an overview of the installation task for QMF in the VSE/ESA environment.

Hardware

The required hardware consists of the following components:

Processor: You can install QMF for VSE/ESA on any processor supported by VSE/ESA Version 2 Release 2 or later in ESA mode.

Tape drive: You need a tape drive for loading the installation tape. You can use any tape drive supported by VSE/ESA Version 2 Release 2 or later.

System console: You can use any terminal supported by VSE/ESA Version 2 Release 2 or later.

Terminal: You need a terminal to install and test QMF. If you are installing support for a national language that requires the double-byte character set (DBCS), you'll need a terminal that also supports DBCS to run the installation verification procedure (IVP).

Prerequisite Software

The following table lists the program products with the minimum release levels required to support QMF for VSE/ESA Version 7. Later releases that are not available at the QMF Version 7 announcement time are not supported unless specifically stated otherwise.

Table 1. Prerequisite Software For QMF For VSE/ESA Version 7

Required product	Version and release	Number
IBM Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA)	Version 2 Release 2	5690–VSE
CICS/VSE	Version 2 Release 2	5686–026
GDDM/VSE	Version 3 Release 1	5686–057
DB2 for VSE	Version 7	5697-F42

The following table lists the program products with the minimum release levels required to support optional functions for QMF for VSE/ESA Version 7. Later releases that are not available at the QMF Version 7 announcement time are not supported unless specifically stated otherwise.

Table 2. Prerequisite software for optional functions for QMF for VSE/ESA Version 7

Product	Version and release	Number
CHARTS (Interactive Chart Utility):		
GDDM—PGF (for GDDM Version 3 release 1.1)	Version 2 Release 1.2	5668–812
GDDM—PGF (for GDDM Version 2 release 3)	Version 2 Release 1.1	5668–812
Callable Interface Programs using the callable interface can be written in:		
IBM C/370 Compiler and	Version 2	5688–187
C/370 Library	Version 2	5688–188
IBM HLASM	Version 1 Release 1 or Release 2	5696–234
VS COBOL II Compiler and Library	Version 1 Release 4	5688–023
COBOL for VSE/ESA	Version 1 Release 1	5686–068
PL/1 for VSE/ESA	Version 1 Release 1	5686–069
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 for VSE/ESA	Version 1 Release 1	5686–068
PL/1 for VSE/ESA	Version 1 Release 1	5686–069
Governor Exit Routine:		
IBM HLASM	Version 1	5696–234

QMF Storage Requirements

Before you start using QMF, you need to make sure that each CICS partition that runs QMF has enough storage to accommodate QMF programs and the QMF reports users create.

The partition must be large enough to accommodate:

All QMF phases: 2.8MB 31-bit storage, total

Storage for users to execute queries and hold QMF report data:	Average of 0.5MB to 1MB GETVIS storage per user. Some report options may require additional storage.

You can allocate storage for both purposes above 16MB.

For a QMF system with up to 20 users, allocate at least 24MB virtual storage for your CICS partition. The minimum acceptable partition size for any QMF system is 18MB, regardless of the number of users.

To specify your partition size, use the VSE ALLOC statement in the ALLOC.PROC data set of the IPL procedures, as shown in the following example. For systems with more than 20 QMF users, increase the ALLOC 0.5MB to 1MB for each additional user.

Also allow 9MB, within the 24MB, for your programs. Specify this space with the SIZE value in the IPL allocation data set:

```
// JOB ALLOC
// EXEC LIBR,PARM='MSHP'
ACC S=IJSYSRS.SYSLIB
CATALOG ALLOC.PROC    DATA=YES     REPLACE=YES
ALLOC S,F1=24M
SIZE F1=9M
⋮
```

Some users might require more than 1MB of GETVIS storage if they use complex formatting options for a report, or if a large amount of data is returned from a query. “Adjusting GETVIS Storage Used for Report Data (DSQSBSTG)” on page 56 explains how to calculate the desired allocation and size.

If a QMF transaction runs out of storage in the CICS partition, it might time out waiting for storage to become available. Therefore, when you adjust the GETVIS values for each user using the method explained in “Adjusting GETVIS Storage Used for Report Data (DSQSBSTG)” on page 56, ensure you increase the ALLOC to accommodate that additional storage.

You might also consider allocating a spill file for certain users, by defining CICS auxiliary temporary storage in DFHTEMP. The spill file is used for extra storage for data and reports. For information about this option, see “Acquiring Extra Temporary Storage (DSQSPILL)” on page 58.

The installation procedure in this book installs QMF to an individual CICS partition. If you have several users using QMF in different CICS partitions, you might consider loading QMF into the 31-bit shared virtual area, as explained on page 272. Or, if the QMF users you support routinely use more than 1MB of GETVIS storage for queries and reports, you might also consider

using CICS multiregion operations or intersystem communications to provide more efficient use of CICS resources at your site. For information about both of these alternatives to the default QMF installation, see "Using Transaction Routing to Control Resource Use" on page 274.

Apply Service

Ensure that the service level of your system is current. Call your IBM Software Service Support or use IBMLink™ (ServiceLink) in the United States or EMEA DIAL in Europe to request the latest program temporary fixes (PTFs) for QMF and its prerequisite products. Additionally, request QMF's preventive service planning (PSP) bucket, SUBSET: QMFVSE under UPGRADE QMF610. The bucket contains general hints, HIPER APARs and documentation changes. Subscribers that have access to either Information/Access or ServiceLink, can download the information directly.

Check Space Requirements

To ensure that there is adequate disk storage for QMF installation, you need to account for three kinds of space requirements. You must calculate all three requirements to get an accurate storage estimate.

Library Space

Your first task is to calculate your *exact* library space requirements, in blocks. You perform this calculation as part of the QMF installation described in the QMF Program Directory. The number of QMF library blocks can vary. Although there is a set number of blocks for the QMF base product, the library block number increases if you are adding support for a national language in addition to English.

If you need a rough estimate for *planning purposes only*, the number of library blocks is approximately 14,100, and the minimum number for each national language feature (NLF) is approximately 4,100.

VSAM Catalog

In addition to the library requirement, QMF needs to define a file in VSAM space. This file needs:

2.5MB of free VSAM space in a VSAM catalog.

0.5MB of free VSAM space in the catalog that holds the GDDM file, ADMF. You need free storage in the user catalog where the ADMF file resides.

If you're using an NLF: For each NLF, you'll need an additional 2.5MB of VSAM catalog space, and a corresponding 0.5MB space in the ADMF file. For further information on

how to add VSAM space to an existing catalog or how to define a new VSAM catalog, see *VSE/ESA Administration*.

dbspace

When you installed DB2, you created public and private dbspaces. QMF needs some of the public dbspaces for tables, queries, procedures, forms, and data.

A dbspace is a logical allocation of space in the database, which consists of 4K pages. To convert dbspaces to megabytes, cylinders and tracks, or to add dbspaces to the database, see *DB2 Server for VSE System Administration*.

If QMF tries to acquire the dbspace and there is not an exact match, it will try to acquire the next largest size available. To avoid wasting space, check that you have:

One 5120-page public dbspace
Three 256-page public dbspaces
Six 128-page public dbspaces

QMF needs this amount of space for each DB2 database; if you have multiple databases, you need to take that into account. To verify the size of the dbspaces, you can enter the following SQL statement from ISQL:

```
SELECT * FROM SYSTEM.SYSDBSPACES WHERE DBSPACETYPE=1 AND OWNER=' '
```

Your disk storage allotment is the sum of the dbspace, VSAM catalog space, and library block size calculated in cylinders or blocks.

Check Your CICS Partition Size

The minimum acceptable partition size for any QMF system is 18MB, regardless of the number of users. For a QMF system with up to 20 users, allocate 24MB virtual storage for your CICS partition. To specify your partition size, use the VSE ALLOC statement in the IPL procedure ALLOC.PROC, such as:

```
ALLOC F4=24M
SIZE F4=9M
```

For systems with more than 20 QMF users, increase the ALLOC by 0.5MB to 1MB for each additional user. Also allow 9MB, within the recommended 24MB, for your programs. You specify this space with the SIZE value in the IPL allocation data set.

Because the size of the GETVIS area is the difference between the partition size and the SIZE value, your GETVIS is 15MB. After installation, you can

adjust GETVIS space to maximize storage for a user's queries and reports. For more information on adjusting GETVIS, see "Chapter 6. Starting QMF" on page 49.

Partition Size for Installation

You need a partition to run the QMF installation job. This partition must have a partition size of at least 1.5MB.

Other Planning Considerations

Not all QMF installations are the same. The following sections describe some additional installation considerations that might apply to your situation.

Tailoring GDDM for QMF and CICS

Before you install QMF, GDDM must be fully installed, tailored, and tested. It is important to do a complete GDDM installation and not merely restore to a library. During QMF installation, QMF modifies GDDM's ADMF file. Additionally, you must define GDDM resources, such as programs and transactions, to CICS.

Changing GDDM 2.3 Default Parameters

If you are using GDDM 2.3, you might need to modify a parameter in the GDDM external defaults module. Ensure that the IOSYNCH parameter in ADMADFC is set to YES.

Run the Installation Verification Procedure (IVP) for GDDM

Check that you have GDDM properly installed by running the IVP for GDDM. The IVP minimizes installation problems and ensures that you are installing QMF onto a clean system.

Running DB2 Guest Sharing

You have the option to connect your CICS partition to a DB2 database on either VSE or VM. When VSE is a guest of VM and shares data with the host's applications through a common VM DB2 database, it is called *guest sharing*. The benefit of SQL guest sharing is that both VM and VSE users can use a common database. QMF requires minimum levels of VM/ESA® 1.1 and DB2 6 to use QMF in an SQL guest sharing environment. You do not have to install QMF for VM.

If you establish an SQL guest sharing environment and want to install QMF on VSE, complete the installation as if VSE owned the database. The VM DB2 database is transparent to the VSE user.

However, if you have both DB2 and QMF installed under VM/ESA, and you want to install an additional QMF product in VSE, you can skip the part of the QMF installation that deals with database installation. During the install

process, you are told when to skip that step. Otherwise, the remainder of this manual assumes that you are installing QMF into a DB2 for VSE database.

Customizing DB2 for Double-Byte Character Support

If you plan to install QMF with a national language that requires double-byte character support, you need to complete the database customization before installing QMF. For details on customizing DB2 for double-byte character support, see *DB2 Server for VSE System Administration*.

Installation Overview

Every data center is different, and every system in each data center can be configured numerous ways. You might have multiple DB2 databases, one or more CICS systems, VSE running as a guest under VM/ESA, DB2 guest sharing, another version of QMF (either on VM or VSE), or special national language requirements. Because your VSE system might be unique, we have designed the QMF installation so that you can easily install and re-install using only a few simple jobs.

Some of these jobs you will run only once, others you will run multiple times, depending on your configuration.

Base Installation

The base (English version) installation requires that you run the following jobs:

DSQ3INIT
: This job establishes the initialization criteria for the remainder of the installation. The other installation jobs use the procedure cataloged by DSQ3INIT to find information about installed products such as GDDM, DB2, and VSAM. It also contains information about the QMF installation. You run DSQ3INIT only once.

DSQ3EINS
: This job defines and loads the QMF panel file (DSQPNLE) into VSAM space. The panel file contains all of the panel definitions. DSQ3EINS also loads maps and sample charts; you run it only once.

DSQ3EDBI
: This job is the database installation job of QMF. DSQ3EDBI creates QMF control tables, loads QMF packages, and defines and loads sample tables into the DB2 database. You run DSQ3EDBI once for each local SQL database that you are connecting to CICS.

  Normally, DSQ3EDBI is a mandatory job. However, if you currently have QMF for VM/ESA 7.1 or later installed, and you want to add a QMF for VSE/ESA in an SQL guest sharing environment, you do not need to run this job. This is because the database portion of QMF was

installed during the QMF VM installation. However, if you want to define a DB2 VSE database in addition to the DB2 VM database, you do need to run DSQ3EDBI.

Optional Job

DSQ3ELNK

This job link-edits QMF with the current versions of GDDM, CICS, and DB2. You do not need to run this job if you have installed:

GDDM/VSE 3.2
CICS for VSE/ESA 2.3
DB2 for VSE 7.1

QMF is already linked with those versions.

Installing Language Support

The QMF base installation loads all of the panels and maps in English. If you require QMF in another or different language, you need to order one or more of the National Language Features (NLFs) of QMF.

You can distinguish base installation members from NLF members by a single character abbreviation known as the *National Language Identifier (NLID)*. In this manual, and throughout the QMF library, we use the letter *n* to represent the NLID. As you proceed through the installation process, you are told when to substitute the letter *n* in a member name with the NLID. For example, if you are installing DSQ3FINS.Z for French support, you substitute *F* in place of the *n* in member name DSQ3*n*INS.Z. The same member name for English support is DSQ3*E*INS.Z. Table 3 lists all of the languages and their associated NLIDs.

Table 3. NLIDs representing QMF base (English) and National Language Features (NLFs)

Language	NLID (n)
Brazilian Portuguese	P
Canadian-French	C
English	E
French	F
German	D
Italian	I
Japanese	K
Korean	H
Simplified Chinese	R
Spanish	S
Swiss French	Y

Table 3. NLIDs representing QMF base (English) and National Language Features (NLFs) (continued)

Language	NLID (n)
Swiss German	Z
Uppercase English	U

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.

NLF Install Process

You start an NLF installation by scanning the tape. You can do this when you scan the tape for the base product. After the tape is restored to disk, you continue through the installation procedures for the base install, up to the point of CICS tailoring. You must complete the installation of the base product, up to CICS tailoring, before you install the NLFs. While doing the base installation, ignore the sections in the procedures that apply to NLFs.

After the base product is installed, go back through the same procedures and follow the directions that apply for an NLF. Continue with CICS tailoring and customize both the base and the NLFs at the same time. Last, run the Installation Verification Procedures (IVPs) for the base product and for each NLF.

The NLF-specific installation jobs are:

DSQ3nINS
: This job is the same as DSQ3EINS, except that it loads the panel files in your national language. It is a mandatory step.

DSQ3nDBI
: This job is the same as DSQ3EDBI, except that it loads the sample tables and profile table in your national language. It is a mandatory step for each database that you connect to CICS.

DSQ3nLNK
: This job is the same as DSQ3ELNK, except it link-edits the national language parts of QMF. It is an optional step depending on the product versions you have installed.

CICS Tailoring

After you install the product base and any NLFs, you are ready to tailor CICS. You can tailor CICS for both the base and any NLFs at the same time. However, you must make the modifications to every CICS system that works with QMF.

Before You Begin

During CICS tailoring, you modify four CICS tables.

- Destination control table (DCT)
- File control table (FCT)
- PROGRAM resource (CSD) or processing program table (PPT)
- TRANSACTION resource (CSD) or program control table (PCT)

All four of the tables can be modified by assembling resource definition tables, but some of the tables can also be modified by defining resources online (RDO) in a CICS system definition (CSD) data set. You'll use copybooks and copy statements to change the tables. There are similar copybooks and copy statements for the NLF versions.

When CICS tailoring is complete, you need to check the installation by following the IVPs for the base product and for each NLF.

Chapter 2. Tailoring Your Installation

Before anyone can use QMF, you need to customize it. This chapter leads you through the necessary steps to tailor or customize QMF and CICS for your system. You must have completed the QMF for VSE 7.1 installation from tape per the program directory before proceeding with this chapter.

Punch Members to an Editor

Because you cannot edit members in a VSE sublibrary, you need to punch the members to a facility that has an editor such as ICCF or Virtual Machine (VM). The following procedure assumes you want to punch QMF jobs to an ICCF library.

1. Return to the main VSE/ESA Function Selection panel.
2. Select the `Command Mode` option to allow you to enter commands directly.

   You can switch to a secondary library by entering

   ```
   /SWI nn
   ```

   where `nn` represents the target ICCF library number.
3. Punch the following members to ICCF. Press Enter after typing each command.

   ```
   LIBRP PRD2.PROD DSQ3INIT.Z DSQ3INIT (REPLACE
   LIBRP PRD2.PROD DSQ3EINS.Z DSQ3EINS (REPLACE
   LIBRP PRD2.PROD DSQ3EDBI.Z DSQ3EDBI (REPLACE
   ```
4. **For NLF**:

   Punch NLF installation members to ICCF. Using the NLID for your NLF listed in Table 3 on page 10, substitute the *n* with the NLID before punching the job. Press Enter after typing each command.

   ```
   LIBRP PRD2.PROD DSQ3nINS.Z DSQ3nINS (REPLACE
   LIBRP PRD2.PROD DSQ3nDBI.Z DSQ3nDBI (REPLACE
   ```

Install QMF Base

In this section you install the QMF base, which involves:

- Modifying and cataloging the initialization procedure
- Running the install job
- Installing to the DB2 database

Catalog the Initialization Procedure

DSQ3INIT is the QMF initialization procedure. It establishes the installation criteria for the remainder of the installation, which is stored in

DSQ3INIT.PROC. Because the information stored in DSQ3INIT is critical to the success of the installation, ensure your entries are correct before running the job. An incorrect entry in this job causes errors in subsequent job steps.

You must edit DSQ3INIT before it can run.

1. Delete the first line of the file, which begins with CATALOG.
2. Change all instances of ..* to * with the following command:

   ```
   ch /..*/*/ *
   ```

3. Delete the last two lines of the file, leaving the end-of-job statement.
4. Verify or change the name of the QMF library and sublibrary, if necessary. If you are using anything other than the default library, PRD2.PROD, change the name to your library name.

   ```
   // EXEC LIBR,PARM='MSHP'
      ACC S=PRD2.PROD
      CATALOG DSQ3INIT.PROC

   // SETPARM QMFLIB=PRD2              *-- QMF for VSE LIBRARY
   // SETPARM QMFSLIB=PROD             *-- QMF for VSE SUBLIBRARY
   ```

5. Check the default library and sublibrary for GDDM/VSE and DB2 for VSE. Compare, and change if necessary, the defaults to actual library and sublibrary names being used. The default values are:

   ```
   // SETPARM ADMLIB=PRD2       *-- GDDM/VSE LIBRARY
   // SETPARM ADMSLIB=PROD      *-- GDDM/VSE SUBLIBRARY
   // SETPARM SQLLIB=PRD2       *-- DB2 LIBRARY
   // SETPARM SQLSLIB=DB2610    *-- DB2 SUBLIBRARY
   ```

6. Check the VSAM catalog name and ID. The catalog name and ID specify the target VSAM catalog for the QMF panel file and any NLF panel files. Compare the fields and change if necessary.

   ```
   // SETPARM UCAT=VSESPUC                   *-- FILE NAME OF CATALOG
   // SETPARM UCATID='VSESP.USER.CATALOG'    *-- FILE ID OF CATALOG
   ```

7. Determine whether you changed any of the defaults for GDDM/VSE. You defined ADMF in a VSAM catalog during the GDDM install. QMF installation loads maps and forms to ADMF. QMF requires the file ID of ADMF (ADMFID), the catalog name (ACAT), and the catalog ID (ACATID).

   If you changed the defaults, change the following statements to match your naming convention.

   ```
   // SETPARM ADMFID='ADMF'         *-- FILE ID OF GDDM/VSE FILE ADMF
   // SETPARM ACAT=VSESPUC          *-- CATALOG NAME AND CATALOG ID IN
   // SETPARM ACATID='VSESP.USER.CATALOG'    *-- WHICH ADMF IS DEFINED
   ```

8. File the job and run DSQ3INIT. Check the system console to ensure that the job ran with a return code of 0.

   If the job did not run with a return code of 0:

   a. Check for an error message on the system console.

b. Check the list output to find the cause of the problem.
c. Correct the problem.
d. Rerun DSQ3INIT.
e. Recheck the return code.

Run the QMF Installation Job

DSQ3EINS is the QMF installation job.

- It defines and loads the QMF panel file
- It loads sample charts
- It loads maps

During the initial installation, to successfully load and execute this job, you must:

1. Edit DSQ3EINS to change or supply the required parameters.
   a. Delete the first line of the file, starting with CATALOG.
   b. Change all instances of ..* to * with the following command:

      ```
      ch /..*/*/ *
      ```

   c. Delete the last two lines of the file, leaving the end-of-job statement.
2. Return to the top of the job and ensure the four job steps are set to YES.

   DSQ3EINS contains four job steps. When a SETPARM is set to YES, you are indicating that you want to run that step; when set to NO, the step is skipped. Under most circumstances, these steps are run only once, because VSE can share files with multiple CICS systems. For subsequent installations, or under error conditions, you might need to set some of the job steps to NO.

   ```
   // SETPARM STEP1=YES    *-- DEFINE CLUSTER DSQPNLE
   // SETPARM STEP2=YES    *-- LOAD DSQPNLE
   // SETPARM STEP3=YES    *-- LOAD QMF CHARTS
   // SETPARM STEP4=YES    *-- LOAD QMF MAPS
   ```

3. Verify or change all instances of the library and sublibrary names. If you installed QMF to a library other than PRD2.PROD, you need to change the library and sublibrary names.
4. Supply one or more volume IDs for the VSAM cluster that holds the QMF panel file. Locate this cluster definition under job step 1.

   ```
   // EXEC IDCAMS,SIZE=AUTO
    DEFINE CLUSTER ( -
           NAME (QMF610.DSQPNLE ) -
           RECORDS (1200 50) -
           SHAREOPTIONS (3) -
           RECORDSIZE (1920   32756 ) -
           VOLUMES (--V001--) -
   ```

   Replace the variable **--V001--** with the volume IDs, such as DOSRES or SYSWK1. For example:

   ```
   VOLUMES (DOSRES SYSWK1) -
   ```

or

```
VOLUMES (DOSRES) -
```

5. File and run the job. Check the system console to ensure that the job ran with a return code of 0.

   If the job did not run with a return code of 0:

   a. Check for an error message on the system console.
   b. Check the list output to find the cause of the error.
   c. Locate the job step that failed.
   d. Correct the problem.
   e. Reset the setparms that appear before the error to NO so that successful job steps are not run again.
   f. Rerun DSQ3EINS.
   g. Recheck the return code.

Install QMF Base into DB2 Database

DSQ3EDBI is the database installation job; you run it once for each SQL database that you are connecting to QMF.

Skip this procedure if you have a DB2 guest sharing environment and you installed QMF for VM. Under these conditions, you can continue with "Tailor QMF for NLF" on page 18, if you have an NLF to install, or skip to "Link-Edit Jobs for QMF" on page 20 for a base installation.

DSQ3EDBI:

- Creates QMF control tables
- Loads QMF packages
- Defines and loads sample tables

1. Edit DSQ3EDBI (first use of the job only).

   a. Delete the first line of the file, starting with CATALOG.
   b. Change all instances of ..* to * with the following command:

      ```
      ch /..*/*/ *
      ```

   c. Delete the last two lines of the file, leaving the end-of-job statement.

2. Ensure that the first three job steps are set to YES.

   ```
   // SETPARM STEP1=YES   -- CREATE QMF CONTROL TABLES IN SQL DB
   // SETPARM STEP2=YES   -- LOAD QMF PACKAGES INTO SQL DB
   // SETPARM STEP3=YES   -- LOAD QMF SAMPLES INTO SQL DB
   ```

   For subsequent installations, or under error conditions, you might need to set some of the job steps to NO. By changing a setparm to NO, you skip the step.

3. Locate the `// SETPARM DBNAME=SQLDS` parameter and verify or change SQLDS to the name of the database that you are using.

DBNAME is the database name that is specified in the SQL DBNAME directory.

4. Determine whether this database has an earlier version of QMF for VSE installed. The possible earlier versions are QMF V1R1, QMF V3R1M1, QMF V3R2, QMF V3R3, or QMF V6R1.

   If there is an earlier version installed, locate the following statement and change the value from NO to V1R1, V3R1M1, V3R2, or V3R3.

   ```
   // SETPARM MIGRATE=NO
   //
   ```

   Using this parameter prevents you from creating duplicate control tables for existing customers.

5. Verify or change all instances of the QMF library and sublibrary names. If you installed QMF to a library other than PRD2.PROD, change the library and sublibrary names.
6. Ensure that DB2 is up and running in multiple user mode.
7. Ensure the SQLDBA's password is set to SQLDBAPW.

   The QMF installation procedures assume that the password for SQLDBA is set to SQLDBAPW. If the password is set to anything else, you need to update DSQ3SETQ.A and re-catalog DSQ3SETQ.A into the QMF installation library.

   To update DSQ3SETQ.A:

   a. Punch the member DSQ3SETQ.A to an editor, such as ICCF.
   b. Modify the CONNECT statement by replacing SQLDBAPW with your existing password for SQLDBA.

      ```
      CONNECT SQLDBA IDENTIFIED BY 'new-SQLDBAPW'
      ```

   c. Recatalog DSQ3SETQ.A into the QMF installation library (PRD2.PROD).

8. File and run DSQ3EDBI. As the job runs, the system console displays messages that indicate which job step is executing. At the end of the job, check the system console to ensure that the job ran with a return code of 0. If you are migrating from an earlier version of QMF for VSE, you might get a return code of 6 in job step 1. Under this condition, you can ignore the return code and continue.

   If the job did not run with a return code of 0 or 6:

   a. Check for an error message on the system console.
   b. Determine from the console the last job step that completed successfully and the job step that failed.
   c. Check the list output to find the cause of the problem.
   d. Correct the problem.
   e. Reset the setparms to NO that occur before the error so that successful jobsteps are not run again.

f. Rerun DSQ3EDBI.

g. Recheck the return code.

If you want to install QMF into additional DB2 databases, repeat this procedure for each database, starting with step 3 on page 16. Otherwise, go to "Tailor QMF for NLF", or to "Link-Edit Jobs for QMF" on page 20.

Tailor QMF for NLF

If you are adding support for a national language, you punched two additional members, DSQ3*n*INS and DSQ3*n*DBI. These jobs need to be edited and executed.

Install NLF

As with DSQ3EINS, you run DSQ3*n*INS only once.

1. Edit DSQ3*n*INS for the following:
   a. Delete the first line of the file, starting with CATALOG.
   b. Change all instances of ..* to * with the following command:

      ```
      ch /..*/*/ *
      ```

   c. Delete the last two lines of the file, leaving the end of job statement.
   d. Locate the three setparms and ensure that they are set to YES.

      ```
      // SETPARM STEP1=YES    *-- DEFINE CLUSTER DSQPNLn
      // SETPARM STEP2=YES    *-- LOAD DSQPNLn
      // SETPARM STEP3=YES    *-- LOAD QMF MAPS TO ADMF
      ```

   e. Verify or correct the QMF library and sublibrary names. If you changed from the default PRD2.PROD library, change the names accordingly.
   f. Supply one or more volume IDs for the VSAM cluster that holds the national language version of the QMF panel file. Locate this cluster definition under job step 1.

      ```
      // EXEC IDCAMS,SIZE=AUTO
       DEFINE CLUSTER ( -
         NAME (QMF610.DSQPNLn ) -
         RECORDS (1200 50) -
         SHAREOPTIONS (3) -
         RECORDSIZE (1920   32756) -
         VOLUMES (--V001--) -
      ```

2. File DSQ3*n*INS and run the job. Check the system console to ensure that the job ran with a return code of 0.

   If the job did not run with a return code of 0:
   a. Check for an error message on the system console.
   b. Determine from the console the last job step that completed successfully and the job step that failed.
   c. Check the list output to find the cause of the problem.

d. Correct the problem.

e. Reset the setparms that appear before the error to NO so that successful job steps are not run again.

f. Rerun DSQ3*n*INS.

g. Recheck the return code.

Install QMF NLF into SQL Database

Edit DSQ3*n*DBI for each database that needs NLF support:

1. Delete the first line of the file, starting with CATALOG.
2. Change all instances of ..* to * with the following command:

   ```
   ch /..*/*/ *
   ```

3. Delete the last two lines of the file, leaving the end-of-job statement.
4. Ensure that the first three job steps are set to YES.:

   ```
   // SETPARM STEP1=YES  *-- Create Profile
   // SETPARM STEP2=YES  *-- Drop Sample Tables
   // SETPARM STEP3=YES  *-- Create and Load Sample Tables
   ```

   For subsequent installations, or under error conditions you might need to set some of the job steps to NO. By changing a setparm to NO, you skip the step.

5. Locate the database name SQLDS, and verify that it is set for the correct database.

   ```
   // SETPARM DBNAME=SQLDS   *-- TARGET DB2 DBNAME FOR QMF on VSE/ESA
   ```

6. Determine whether this database has an earlier version of QMF for VSE installed. The possible earlier versions are QMF V1R1, QMF V3R1M1, QMF V3R2 or QMF V3R3 or QMF V6R1. If there is an earlier version installed, locate the following statement and change the value from NO to V1R1, V3R1M1, V3R2, V3R3, or QMF V6R1.

   ```
   // SETPARM  MIGRATE=NO
   ```

   Using this parameter prevents you for creating duplicate control tables and samples.

7. Verify or correct the QMF library and sublibrary names. If you changed from the default PRD2.PROD library, change the names accordingly.
8. Ensure that DB2 is up and running in multiple user mode.
9. File DSQ3*n*DBI and run the job. Check the system console to ensure that the job ran with a return code of zero.

   If the job did not run with a return code of 0:

   a. Check for an error message on the system console.

   b. Determine from the console the last job step that completed successfully and the job step that failed.

   c. Check the list output to find the cause of the problem.

d. Correct the problem.
e. Reset the setparms that appear before the error to NO so that successful jobsteps are not run again.
f. Rerun DSQ3*n*DBI.
g. Recheck the return code.

Repeat this procedure, starting with step 5 on page 19, for every database that needs NLF support.

Link-Edit Jobs for QMF

QMF is prelinked with the following release levels of products:
GDDM/VSE 3.2
CICS/VSE 2.3
DB2 6

If you have different releases of these products, you need to run the following job.

1. Punch DSQ3ELNK to a library, and edit the job. The following example uses ICCF to perform the punch.

   ```
   LIBRP PRD2.PROD DSQ3ELNK.Z DSQ3ELNK (REPLACE
   ```

   Press Enter.

   For NLF:

   Punch and edit DSQ3*n*LNK.

   ```
   LIBRP PRD2.PROD DSQ3nLNK.Z DSQ3nLNK (REPLACE
   ```

   Press Enter.
2. Delete the first line of the file, starting with CATALOG.
3. Change all instances of ..* to * with the following command:

   ```
   ch /..*/*/ *
   ```
4. Verify or change the name of the QMF library and sublibrary. If you are using anything other than the default PRD2.PROD library, change the name globally to your naming convention.
5. Verify or change the search chain so that it contains the library and sublibrary for QMF, DB2, GDDM/VSE, and CICS.

   ```
   // LIBDEF OBJ,SEARCH=(PRD2.PROD,PRD2.DB2610,PRD1.BASE)
   ```
6. File and run the job. Check the system console to ensure that the job ran with a return code of 4. You do not receive a return code of 0 because of weak external references (WXTRNs) that are not resolved during the linkage editor run.

If the job did not run with a return code of 4, recheck the LIBDEF statement for the above products, and rerun the link-edit job.

Following is an example of the kind of output you can expect from this link-edit:

```
2165I WARNING - RMODE=ANY ASSIGNED TO PHASE, BUT THE PHASE
       CONTAINS 2 AND/OR 3 BYTE RELOCATABLE ADDRESS CONSTRAINTS
UNRESOLVED EXTERNAL REFERENCES               WXTRN     ADMUFO
                                             WXTRN     GERHND
                                             WXTRN     ADME000C
                                             WXTRN     ADMADFC
                                             WXTRN     ADMACIN
                                             WXTRN     ADMUOFF
                                             WXTRN     DSQCLDQ
                                             WXTRN     LTTBAS
                                             WXTRN     LTTBASX
                                             WXTRN     DSNHLI
                                             WXTRN     DSQIRDB2
```

Additionally you will see several messages about the use of WXTRNs and using 2- or 3-byte ADCONs. These messages are expected and should not cause a problem for QMF.

Link Jobs for NLF

Repeat the procedure in “Link-Edit Jobs for QMF” on page 20 for your NLF version of DSQ3*n*LNK.

Tailor CICS

You need to modify the following CICS tables for QMF to run in CICS:

- Destination control table (DCT)
- File control table (FCT)
- PROGRAM resource (CSD) or processing program table (PPT)
- TRANSACTION resource (CSD) or program control table (PCT)

These modifications need to be made for every CICS system that works with QMF.

You must define resources, such as QMF programs and transactions, that CICS controls for a particular run. You can define resources through two methods: either by defining resources online (RDO) in a CICS system definition (CSD) data set, or by assembling resource definition tables using macros.

You must use the macro method to define the FCT and DCT. QMF supplies copybooks for modifying these tables.

You can also modify your PCT and PPT by using the QMF supplied copybooks. However, the recommended way to modify these tables is with

RDO to a CSD data set. RDO allows you to interactively create resource definitions and store them in a data set.

CICS offers a utility program (DFHCSDUP) to update the CSD with a batch job. QMF provides a job to allow you to define the QMF programs and transactions to CICS without reassembling the DFHPCT and DFHPPT.

In the following procedures you need to know how to locate tables and assemble them using the Interactive Interface. Using the Interactive Interface is described in *IBM VSE/ESA Administration*

Modify the DFHFCT and DFHDCT

Change these tables using the macro method.

Before editing, locate the source used to create the FCT and DCT for this CICS system. In the following examples, we assume that you used the skeletons provided with VSE/ESA to bring up another CICS system. If you did not use those skeletons, you can use *CICS for VSE/ESA Resource Definition (Macro)* to locate the appropriate place to insert the changes. Also note that these examples have a suffix of C2; your tables might be different. The chapter on installing a second predefined CICS in *IBM VSE/ESA Administration* gives more details on using these skeletons.

Modify DFHFCT

Modify the source of your DFHFCT to define the QMF panel file to CICS.

1. Find the LIBDEF statement and ensure that the search chain contains the library and sublibrary for QMF. Otherwise, VSE/ESA cannot find the copybook.

   ```
   // LIBDEF *,SEARCH=(PRD1.BASE,PRD2.PROD)
   ```

2. Add a local entry for the QMF panel file in the FCT. If you are adding an NLF, also add a copy statement for the NLF member. For example:

   ```
   *-------------------------------------------------------------------
   *         LOCAL ENTRIES SHOULD BE PLACED BELOW THIS BOX
   *-------------------------------------------------------------------
            COPY DSQ3EFCT
            COPY DSQ3nFCT
            SPACE 3
   ```

   Substitute the appropriate NLID for the *n*.

3. Assemble and link-edit the member to create a new DFHFCTC2 phase.

Modify DFHDCT

Locate the source of your DFHDCTC2 and make the following changes:

1. Find the LIBDEF statement and ensure that the search chain contains the library and sublibrary for QMF. Otherwise, VSE cannot find the copybook.

   ```
   // LIBDEF *,SEARCH=(PRD1.BASE,PRD2.PROD)
   ```

2. Find the local entry for TYPE=SDSCI and add a copy statement for DSQ3DCTS as shown in the following example:

```
*-------------------------------------------------------------------
*   LOCAL ENTRIES FOR TYPE=SDSCI SHOULD BE PLACED BELOW THIS BOX
*-------------------------------------------------------------------
         COPY DSQ3DCTS
         SPACE 3
```

3. Install the QMF trace facility.

   Find where local entries are specified and add a copy statement, (DSQ3DCTE), for TYPE=EXTRA, as shown in the following example:

```
*-------------------------------------------------------------------
*        OTHER LOCAL ENTRIES SHOULD BE PLACED BELOW THIS BOX
*-------------------------------------------------------------------
         COPY DSQ3DCTE
         SPACE 3
```

4. Assemble and link-edit the member to create a new DFHDCTC2 phase.

Ensure the job completes with a return code of 0 or 4. If you receive higher return codes, check the list output and correct the error.

Define QMF Programs and Transactions to CICS

QMF provides help to define the QMF programs and transactions to CICS:

- By providing a batch job that defines the QMF resources to the CICS CSD
- By providing copybooks that can be included in the CICS PPT and PCT

We recommend that you define the QMF programs to the CSD using the job, DSQ3ECSD (and DSQ3*n*CSD for NLF installs).

Update the CSD

This procedure creates a new LIST called QMF, which is defined in the CSD. Additionally, for each language, a GROUP called QMF610*n* is defined in the LIST QMF. QMF610*n* contains the definition of the QMF programs and transactions (E for English).

The following procedure assumes you punch QMF jobs to an ICCF library.

1. Punch the following member to ICCF using this command:

   ```
   LIBRP PRD2.PROD DSQ3ECSD.Z DSQ3ECSD (REPLACE
   ```

   Press Enter.

2. Punch equivalent NLF members to ICCF. Substitute the NLID for the *n* in the following example.

   ```
   LIBRP PRD2.PROD DSQ3nCSD.Z DSQ3nCSD (REPLACE
   ```

   Press Enter.

3. Edit DSQ3ECSD (or DSQ3*n*CSD for NLF)

a. Delete the first line of the file, starting with CATALOG.
b. Change all instances of ..* to * with the following command:
   ```
   ch /..*/*/ *
   ```
c. Delete the last two lines of the file, leaving the end-of-job statement.

4. Check that the file ID and catalog reflect your CSD.
   ```
   // DLBL DFHCSD,'CICS.CSD',,VSAM,CAT=VSESPUC
   ```
5. Verify that the library and sublibrary names in the POWER statements are the QMF library and sublibrary names.
   ```
   * $$ SLI MEM=DSQ3ECDN.A,S=PRD2.PROD
   * $$ SLI MEM=DSQ3BCDB.A,S=PRD2.PROD
   ```

   For NLF:
   ```
   * $$ SLI MEM=DSQ3nCDN.A,S=PRD2.PROD
   ```
6. File and run the job. Ensure the job completes with a return code of 0 or 4. If you receive higher return codes, check the list output and correct the error. If you are having difficulty with RDO, see *CICS for VSE/ESA Resource Definition (Online)*

Run CEDA

CEDA is one of three interactive online transactions that comprise RDO. Using CEDA, you can modify, delete, check, and browse definitions while CICS is running. CEDA provides commands for managing groups and lists, which include installing a group of resource definitions on an active system.

To activate your resource definitions from the main VSE/ESA Function Selection panel:

1. Select `7 CICS - Supplied Transactions` and press Enter.
2. Select `2 Invoke CEDA` from the CICS-Supplied Transaction panel and press Enter.
3. Type `AP LIST(QMF) TO(VSELIST)` and press Enter.

   The VSELIST parameter must be the name specified for the GRPLIST parameter specified in DFHSIT of this CICS. CEDA appends the LIST QMF to the LIST VSELIST and ensures that the QMF definitions are known to CICS after the next cold start.

   To activate these QMF definitions immediately, you can either:

   - Perform a CICS cold start, or
   - Issue `CEDA INSTALL GR(QMF710e)` for a temporary (but immediate) change

   For NLF:
4. Repeat the procedure for applicable NLF members. Your CEDA command looks like:
   ```
   CEDA INSTALL GR(QMF710n)
   ```

Your next step is to modify the CICS startup job, as described in "Modify the CICS Startup Job".

Modify the DFHPCT and DFHPPT

If you didn't define the QMF programs and transactions to the CSD, then you need to make the following modifications.

Modify the DFHPCT

1. Locate the source deck of your DFHPCT.
2. Find where local entries are made and enter copy statements for DSQ3EPCT and for any NLF PCT entries, as shown in the following example:

```
*-------------------------------------------------------------------
*           LOCAL ENTRIES SHOULD BE PLACED BELOW THIS BOX
*-------------------------------------------------------------------
         SPACE 3
         COPY DSQ3EPCT          *****  QMF for VSE BASE ENTRIES
```

   For NLF:

```
         COPY DSQ3nPCT          *****  QMF for VSE NLF ENTRIES
```

   Substitute the appropriate NLID for the *n*.
3. Assemble and link-edit the member to create a new DFHPCTC2 phase.

Modify the DFHPPT

1. Locate the source of your DFHPPT.
2. Find where local entries are made and enter copy statements for DSQ3EPPT and for any NLF PPT entries, as shown in the following example:

```
*-------------------------------------------------------------------
*           LOCAL ENTRIES SHOULD BE PLACED BELOW THIS BOX
*-------------------------------------------------------------------
         SPACE 3
         COPY DSQ3EPPT          *****  QMF for VSE BASE ENTRIES
```

   For NLF:

```
         COPY DSQ3nPPT          *****  QMF for VSE NLF ENTRIES
```

   Substitute the appropriate NLID for the *n*.
3. Assemble and link-edit the member to create a new DFHPPTC2 phase.

Modify the CICS Startup Job

In this step, you modify the CICS startup job.

1. Locate the LIBDEF statement with the following search string and ensure that it contains the QMF library and sublibrary.

```
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD1.BASE,PRD2.DB2610, -
                    PRD2.PROD),PERM
```

2. Define the labels for the base QMF panel file and for the NLF equivalent member with your other CICS DLBL statements.

   ```
   // DLBL DSQPNLE,'QMF710.DSQPNLE',,VSAM,CAT=VSESPUC
   ```

   For NLF:

   ```
   // DLBL DSQPNLn,'QMF710.DSQPNLn',,VSAM,CAT=VSESPUC
   ```

   Optionally, you can include the above DLBL statement in the system standard label procedure.

3. Ensure that you expanded virtual storage allocated to the CICS partition. Partition size considerations are discussed under “Check Your CICS Partition Size” on page 7.

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

Install QMF for VSE/ESA into a Second CICS System

When installing QMF to several CICS systems, determine if you have a single CSD that is shared among the CICS systems, or individual CSDs per CICS system. You can determine if all the CICS systems use the same CSD by comparing the GRPLIST parameter in the SIT for each CICS.

If you have a single CSD for all of your CICS systems, you can skip further customization steps after you have initially defined QMF programs and transactions to the CSD. You still need to modify the FCT and DCT of the second CICS. These procedures begin with “Modify DFHFCT” on page 22 and “Modify DFHDCT” on page 22. and continue through modifying the CICS startup routine, as described in “Modify the CICS Startup Job” on page 25.

If you have individual CSDs for every CICS system, you need to repeat the entire CICS customization procedure for each CICS system. These procedures begin with “Modify the DFHFCT and DFHDCT” on page 22.

Chapter 3. Installing QMF into Remote Database Servers

In order to install QMF into remote database servers from VSE, TCP/IP communications must be in place between the local DB2 for VSE requester and the remote database server.

Installing QMF Version 7 into a DB2 Universal Database® Remote Server

DB2 Universal Database V5 or higher is supported.

Prerequisites

TBD

Punch Members to an Editor

Because you cannot edit members in a VSE sublibrary, you need to punch the members to a facility that has an editor such as ICCF or Virtual Machine (VM). The following procedure assumes you want to punch QMF jobs to an ICCF library.

1. Return to the main VSE/ESA Function Selection panel.
2. Select `Command Mode` option to allow you to enter commands directly.

   You can switch to a secondary library by entering

   ```
   /SWI nn
   ```

   where `nn` represents the target ICCF library number.
3. Punch the following members to ICCF. Press Enter after typing each command.

   ```
   LIBRP PRD2.PROD DSQ3EDBU.Z DSQ3EDBU (REPLACE
   LIBRP PRD2.PROD DSQ3BPKG.Z DSQ3BPKG (REPLACE
   ```
4. **For NLF**:

   Punch NLF installation members to ICCF. Using the NLID for your NLF listed in Table 3 on page 10, substitute the *n* with the NLID before punching the job. Press Enter after typing each command.

   ```
   LIBRP PRD2.PROD DSQ3nDBU.Z DSQ3nDBU (REPLACE
   ```

Installation steps

The steps listed here describe a QMF server installation to a remote DB2 UDB V5 or higher database server. The QMF server installation utilizes DB2 for VSE batch requester services and assumes that all connections are active and working. These steps install QMF control and sample tables and reload packages at the remote server.

1. Run job DSQ3EDBU - install QMF control tables and sample tables. You must edit job DSQ3EDBU before it can be run.
   a. Delete the first line of the file, which begins with CATALOG.
   b. Change all instances of `..*` to `*` with the following command:
      ```
      ch /..*/*/ *
      ```
   c. Delete the last two lines of the file, leaving the end-of-job statement.
   d. Carefully read the detailed comments in the file and change all appropriate values.
   e. File and run DSQ3EDBU.

   If the job did not run with a return code of 0:
   a. Check for an error message on the system console.
   b. Check the list output to find the cause of the problem.
   c. Correct the problem.
   d. Rerun DSQ3EDBU.
   e. Recheck the return code.
2. Run job DSQ3BPKG - RELOAD QMF packages at remote server. You must edit job DSQ3BPKG before it can be run.
   a. Delete the first line of the file, which begins with CATALOG.
   b. Change all instances of `..*` to `*` with the following command:
      ```
      ch /..*/*/ *
      ```
   c. Delete the last two lines of the file, leaving the end-of-job statement.
   d. Carefully read the detailed comments in the file and change all appropriate values.
   e. File and run DSQ3BPKG.

   The job will not end with a return code of 0. This is expected. A successful return code for this job will be 41 or less. If a return code of 42 or more is received:
   a. Check for an error message on the system console.
   b. Check the list output to find the cause of the problem.
   c. Correct the problem.
   d. Rerun DSQ3BPKG.
   e. Recheck the return code.
3. Install the QMF NLF at the remote DB2 UDB server. If you are adding support for a national language, run DSQ3nDBU. You must edit job DSQ3nDBU before it can be run.
   a. Delete the first line of the file, which begins with CATALOG.
   b. Change all instances of `..*` to `*` with the following command:
      ```
      ch /..*/*/*
      ```

c. Delete the last two lines of the file, leaving the end-of-job statement.
d. Carefully read the detailed comments in the file and change all appropriate values.
e. File and run DSQ3nDBU.

If the job did not run with a return code of 0:

a. Check for an error message on the system console.
b. Check the list output to find the cause of the problem.
c. Correct the problem.
d. Rerun DSQ3nDBU.
e. Recheck the return code.

Installing QMF Version 7 into a DB2 for AS/400® Server

The steps listed here describe a QMF server installation to a remote DB2 for AS/400 4.4 or higher database server. The QMF server installation utilizes DB2 for VSE batch requester services and assumes that all connections are active and working. These steps install QMF control and sample tables and reload packages at the remote server.

1. Run job DSQ3EDBA - install QMF control tables and sample tables. You must edit job DSQ3EDBA before it can be run.
   a. Delete the first line of the file, which begins with CATALOG.
   b. Change all instances of `..*` to `*` with the following command:

      ```
      ch /..*/*/ *

      ch /..*/*/ *
      ```

   c. Delete the last two lines of the file, leaving the end-of-job statement.
   d. Carefully read the detailed comments in the file and change all appropriate values.
   e. File and run DSQ3EDBA.

   If the job did not run with a return code of 0:

   a. Check for an error message on the system console.
   b. Check the list output to find the cause of the problem.
   c. Correct the problem.
   d. Rerun DSQ3EDBA
   e. Recheck the return code.
2. Run job DSQ3BPKG - Reload QMF packages at remote server. You must edit job DSQ3BPKG before it can be run.
   a. Delete the first line of the file, which begins with CATALOG.
   b. Change all instances of `..*` to `*` with the following command: `ch /..*/*/ *`

c. Delete the last two lines of the file, leaving the end-of-job statement.
d. Carefully read the detailed comments in the file and change all appropriate values.
e. File and run DSQ3BPKG.

The job will not end with a return code of 0. This is expected. A successful return code for this job will be 41 or less. If a return code of 42 or more is received:

a. Check for an error message on the system console.
b. Check the list output to find the cause of the problem.
c. Correct the problem.
d. Rerun DSQ3BPKG.
e. Recheck the return code.

3. Install the QMF NLF at the remote AS/400 server. If you are adding support for a national language, run DSQ3nDBA.You must edit job DSQ3nDBA before it can be run.
   a. Delete the first line of the file, which begins with CATALOG.
   b. Change all instances of `..*` to `*` with the following command: `ch /..*/*/ *`
   c. Delete the last two lines of the file, leaving the end-of-job statement.
   d. Carefully read the detailed comments in the file and change all appropriate values.
   e. File and run DSQ3nDBA.

   If the job did not run with a return code of 0:

   a. Check for an error message on the system console.
   b. Check the list output to find the cause of the problem.
   c. Correct the problem.
   d. Rerun DSQ3nDBA.
   e. Recheck the return code.

Chapter 4. Run the Installation Verification Procedure (IVP)

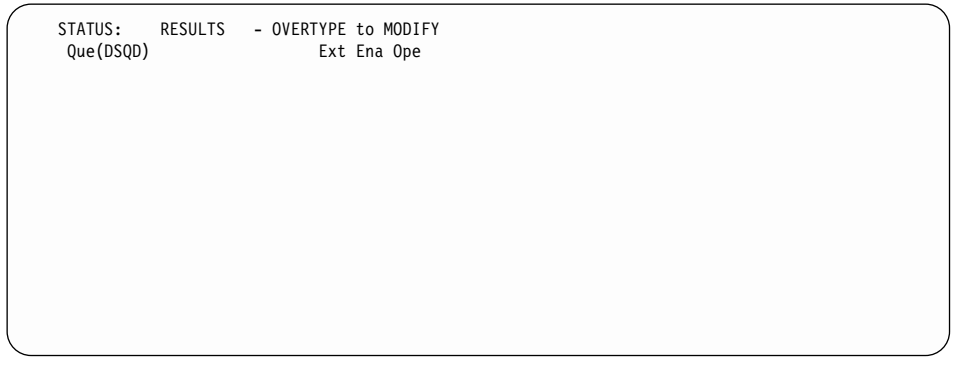
This chapter leads you through the final testing of QMF, called the installation verification procedure (IVP). To test that you have installed QMF for VSE properly, you need to start QMF and issue a few QMF commands. Most elements of the QMF product installation are tested by simply starting QMF.

Before You Start QMF

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

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

   You should see a screen similar to this:



   `Ena Ope` indicates that the queue is open and enabled. If you do not see that DSQD is enabled and open, you need to review your modifications to the CICS DCT. See “Modify DFHDCT” on page 22, for more details on modifying the DCT.

Start and Test QMF

This procedure starts the QMF for VSE product and tests that the product is properly installed. If you receive an error message during any part of the procedure, it indicates that QMF did not start properly. Under these circumstances, start by investigating some of the more common problems as described in “What if It Didn’t Work?” on page 34.

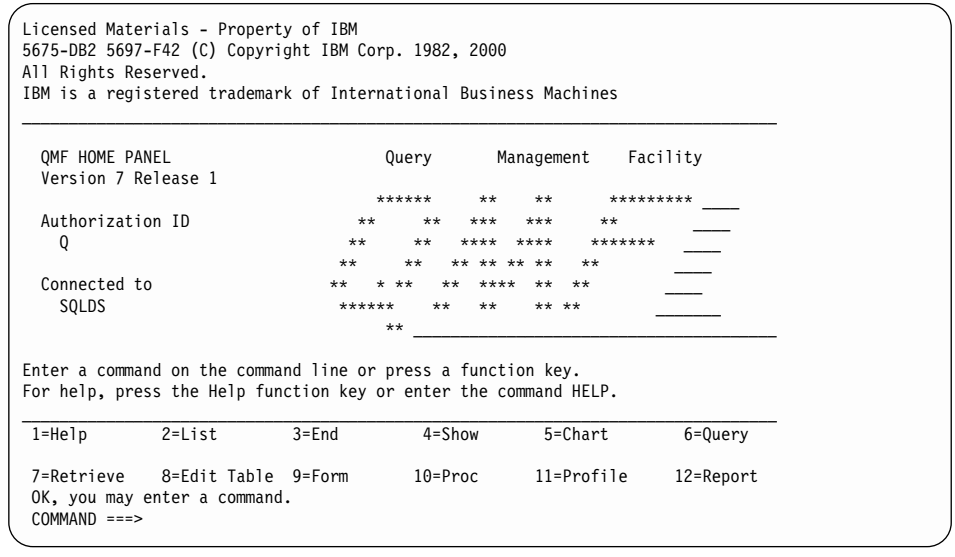
Run the Installation Verification Procedure (IVP)

1. Sign on to the CICS system that is connected to QMF.
2. Press the Escape function key to begin a native CICS session.
3. Start QMF by issuing the CICS transaction, QMFE, and specifying the DB2 authorization ID and password that you want to use for your test. Also specify the use of the temporary storage queue (DSQSDBQT) so that you can view any warning messages online. To start QMF and use the Q user ID, with the temporary storage queue name, DSQD, specify:

   ```
   QMFE UID=Q/QMF,DSQSDBQT=TS,DSQSDBQN=DSQD
   ```

   Where QMF is the password for Q.

   You should see the QMF Home panel.



4. Verify existence of QMF online help.

   Press the Help function key. You should see this Help panel:

```
Licensed Materials - Property of IBM
5675-DB2 5697-F42 (C) Copyright IBM Corp. 1982, 2000
All Rights Reserved.
IBM is a registered trademark of International Business Machines
+-------------------------------------------------------------------------------+
|                       Help: Query Management Facility                         |
|                                                                               |
| Select a topic.                                                               |
|                                                                 1  to 7  of 14 |
|     1. What's new in Version 7                                                |
|     2. Profile                                                                |
|     3. QMF commands                                                           |
|     4. Prompted Query                                                         |
|     5. SQL (Structured Query Language)                                        |
|     6. Table Editor                                                           |
|     7. Forms                                                                  |
+-------------------------------------------------------------------------------+
| F1=Help  F3=Exit  F7=Backward  F8=Forward  F9=Keys  F12=Cancel                |
+-------------------------------------------------------------------------------+


OK, HELP performed. Please proceed.
```

Exit from the Help panel by pressing either PF3 or PF12.

5. Obtain a list of QMF-supplied sample tables.

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

```
+-------------------------------------------------------------------------------+
|                                 Table List                                    |
|                                                                               |
| Action   Name               Owner                                             |
|                                                           1  to  7 of 36      |
|          APPLICANT          Q                                                 |
|          DSQ_RESERVED       Q                                                 |
|          OBJECT_DATA        Q                                                 |
|          OBJECT_DIRECTORY   Q                                                 |
|          OBJECT_REMARKS     Q                                                 |
|          ORG                Q                                                 |
|          PARTS              Q                                                 |
+-------------------------------------------------------------------------------+
| F1=Help  F4=Command  F5=Describe  F6=Refresh  F7=Backward  F8=Forward         |
| F9=Clear  F10=Comments  F11=Sort  F12=Cancel                                  |
+-------------------------------------------------------------------------------+
OK, BACKWARD performed. Please proceed.
```

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

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

```
CEBR DSQD
```

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

Run an IVP for NLF

You can also test your installation of QMF with a national language feature. Rerun the IVP, beginning with "Before You Start QMF" on page 31, and start QMF by issuing a different transaction ID. Use QMF*n*, where *n* is the NLID for the language, as given in Table 3 on page 10. For example, if you are installing German, your transaction ID is QMFD.

If you are testing a language that uses the double-byte character set, such as Japanese, Korean, or simplified Chinese, your terminal must be double-byte enabled.

If you encounter problems, see the messages found in "What if It Didn't Work?".

What if It Didn't Work?

If QMF is unable to start, an error message is generated. Descriptions of some of the more common errors follow. However, this list is not meant to replace the messages and codes manual for QMF or for other products. If you do not find the message on this list, consult the appropriate messages and codes manual.

The following list includes some of the most common errors occurring when running the IVP:

AEY9 ABEND
: The database connection between the CICS partition and DB2 is not active. Issue a CIRB command.

G050 ABEND
: The release level of GDDM being used in CICS partition doesn't match the version with which QMF was link-edited. Follow the procedure in "Link-Edit Jobs for QMF" on page 20.

Gxxx ABEND
: Issued by GDDM, see *GDDM Diagnosis* for Version 3.2 and the *GDDM Diagnosis and Problem Determination Guide* for Version 2.3.

DFH1599
: Region/Partition size is insufficient to initialize CICS.

  Increase the partition size.

DSQ40083

GDDM ERROR ADM0962 E MAPGROUP 'DXYKIMD5' not found. SEVERITY 8 FUNCTION MSQGRP.

The double-byte character set language feature requires a terminal that is also double-byte enabled. Before restarting QMF, ensure your terminal can display double-byte characters. If your terminal is double-byte enabled and you still have the error, check the CICS Terminal Control Table (TCT) for proper entries.

DSQ51304

File DSQPNLn not found in CICS. *** CMD=HELP

The VSAM file that contains QMF screen images is not available. Check the results of DSQ3EINS or DSQ3nINS for NLF. Also, verify that the panel files were defined in the CICS startup and in the CICS FCT.

DSQI0041

Unable to load module(s) nnnnnnn

If nnnnnn = ADMASP, verify that the GDDM product library was available when running the QMF job DSQ3ELNK job. If nnnnnn = ARIPRDI, verify that the DB2 product library was available when running QMF job DSQ3ELNK job. Other modules should be available from QMF product library.

What about warning messages? QMF generates warning messages for conditions it detects while starting QMF. Your QMF trace data contains helpful information for analyzing warning messages. For example, the messages might concern the initialization of the QMF governor DSQUEGV3, or the availability of the edit exit phase DSQUECIC.

You can use `CEBR DSQD` to browse the temporary storage queue for warning messages.

Chapter 5. How to Maintain QMF

QMF maintenance includes adding new components (because they were not installed initially) and replacing existing ones (because of service updates). Both types of activities are described in this chapter. To effectively use the following procedures, we assume that QMF and all of the prerequisite products are installed.

Adding New Components

Because very few systems are static, you might want to add new products, new versions or releases of products, or additional databases.

Adding GDDM-PGF

GDDM-PGF is an optional product that you might choose to install after installing QMF. Because all GDDM objects (such as charts and forms) are loaded into the ADMF file during QMF installation, no further action is required on the QMF side.

Adding QMF to Another DB2 Database

As your system grows, you might want to connect QMF to another DB2 database. Simply repeat the procedures in Install QMF Base into DB2 Database starting with step 3 on page 16 to add your new database. If national language support is also needed for this database, follow the procedure in Install QMF NLF into SQL Database, beginning with step 4 on page 19.

Migrating to New Releases of DB2, CICS, or GDDM

Because QMF is prelinked to specific release levels of DB2, CICS, and GDDM, you need to relink-edit anytime you migrate to a new release of those products. See “Link-Edit Jobs for QMF” on page 20 for the supported release levels and the procedures to link new releases to the QMF base and NLF versions.

Binding QMF 7.1 Packages at a Remote Server

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.

But, for those servers containing QMF 3.3 or above where migration is not a current option, users can run the job DSQ3BPKG. This job binds QMF 7.1 packages at any server specified.

Binding Packages at a Remote Server

Read, tailor and submit job DSQ3BPKG to perform the binds. Check the job output for error messages and re-run as necessary.

Scenerio for use: Local DB2 for VSE subsystem, DB2VSE, is migrated from QMF 3.3 to QMF 7.1. QMF users in subsystem DB2VSE regularly communicate with a DB2 for VM server, SQLV61A, which contains QMF 3.3. The DB2 for VM DBA cannot perform a QMF migration to 7.1 at the VM server. In order for the QMF 7.1 installation in DB2VSE to communicate with QMF on SQLV61A, job DSQ3BPKG must be run to bind packages at the DB2 for VM server.

Replacing Existing Components

This section describes the steps necessary to replace or re-install QMF, and how to apply service updates to QMF.

Re-installing QMF

To re-install QMF:

1. Install the tape as described in the QMF Program Directory.
2. Perform "Run the QMF Installation Job" on page 15 to re-install QMF.

   You do not need to rerun the initialization procedure, DSQ3INIT, unless you change product information from the last installation. For example, if you are using another release of GDDM or installing to a different sublibrary, then you would follow Catalog the Initialization Procedure beginning at step 4 on page 14.

   Begin the installation procedure at step 2 on page 15. When you run this job, the panel file is deleted, redefined, and reloaded. QMF charts and maps are also reloaded into the GDDM object file, ADMF.
3. Perform "Install QMF Base into DB2 Database" on page 16, to re-install QMF packages.

   Set the first three job steps as follows:

   ```
   // SETPARM STEP1=NO     -- CREATE QMF CONTROL TABLES IN SQL DB
   // SETPARM STEP2=YES    -- LOAD QMF PACKAGES INTO SQL DB
   // SETPARM STEP3=NO     -- LOAD QMF SAMPLES INTO SQL DB
   ```

   Continue with the remaining procedure.
4. Determine whether you need to relink-edit your products, as described in "Link-Edit Jobs for QMF" on page 20.
5. Skip the remaining steps, because it is not necessary to tailor CICS again.

Re-installing an NLF

To re-install a national language feature (NLF), follow the procedure in Install NLF starting with step 1d on page 18. You do not need to run DSQ3nDBI.

Determine whether you need to relink-edit your products, as described in "Link-Edit Jobs for QMF" on page 20. You do not need to tailor CICS again.

Applying Service Updates

You might need to apply maintenance or service updates to QMF periodically. These updates are in the form of a Program Temporary Fix (PTF) tape from IBM. All QMF tapes are shipped in MSHP format for easy installation and tracking. For more detail on how to apply PTFs using MSHP, see *VSE/ESA Installation and Service*

When you receive your PTF tape from IBM, you also receive detailed instructions for installation of that specific fix. To help you understand those specific instructions, the following overview familiarizes you with some of the unique aspects of applying service for QMF.

Like most IBM products, QMF consists of phases; but unlike most IBM products, it also consists of objects such as panels, GDDM maps, and SQL packages.

Replacing Text Decks or Phases

This is the most common and straight forward type of replacement. Simply apply the PTF that contains the new text deck (object) or phase. The PTF specifies to MSHP which link book to use, if necessary.

There are, however, QMF objects that require your handling, as they cannot be handled automatically. The MSHP process does keep track of these changes and restores the objects to the QMF sublibrary. The PTF documentation provides details if one of the following steps is to be performed after the PTF installation.

Updating the QMF Panel File

If changes are necessary to the QMF Panel file (DSQPNL*n*), you do not have to replace the entire file. Instead, single panels are shipped using the following naming convention:

DXY*n*name.N

where:

n is the NLID

DXY*n*name

is the complete name of the panel

1. Install the PTF; the panel (or panels) are restored in the QMF sublibrary.
2. Close the existing panel file using the CICS transaction, CEMT:

   ```
   CEMT SET DA(DSQPNLn) CLOSE
   ```

3. Load the panels to the VSAM panel file DSQPNLn. Use the following sample job to load the panel DSY**n**name to file DSQPNLn.

```
* $$ JOB JNM=REPPANEL,DISP=D,CLASS=0
// JOB REPPANEL  Replace panel in the QMF panel file
// DLBL DSQPNLn,'QMF610.DSQPNLn',,VSAM,CAT=VSESPUC
// LIBDEF *,SEARCH=(qmflib.sublib)
// EXEC DSQCVS80,SIZE=AUTO
* $$ SLI MEM=DXYnname.N S=qmflib.sublib
        ..........
/*
/&
* $$ EOJ
```

Check and modify if necessary the following values:

n NLID

The single character that represents the language of the panel. QMF NLIDs are listed in Table 3 on page 10.

VSESPUC
VSAM catalog name where the panel file was originally defined during QMF installation.

qmflib.sublib
Library and sublibrary for QMF

DXYnname
Name of the panel to be replaced. (This information is provided with the PTF.)

..... Repeat, if necessary, the last statement for every panel being replaced by the PTF.

4. Reopen the panel file with:

```
CEMT SET DA(DSQPNLn) OPEN
```

Updating QMF GDDM Maps

QMF GDDM maps can also be affected by a PTF. As with the panels, those objects are restored to the QMF sublibrary when you apply the PTF. More details are included with the PTF.

1. Install the PTF.
2. Modify the SETPARM statements in the QMF installation job, DSQ3EINS.

```
// SETPARM STEP1=NO        *-- DEFINE CLUSTER DSQPNLE
// SETPARM STEP2=NO        *-- LOAD DSQPNLE
// SETPARM STEP3=NO        *-- LOAD QMF CHARTS
// SETPARM STEP4=YES        *-- LOAD QMF MAPS
```

3. File and run the job. The first three job steps are skipped, and execution begins with loading the QMF maps.

NLF Maps: Because GDDM maps are language dependent, your PTF might require you to change those objects, as well.

1. Rerun the job DSQ3*n*INS, with the following SETPARM settings:

```
// SETPARM STEP1=NO        *-- DEFINE CLUSTER DSQPNLP
// SETPARM STEP2=NO        *-- LOAD DSQPNLP
// SETPARM STEP3=YES        *-- LOAD QMF MAP GROUPS TO ADMF
```

2. File and run the job. Check the system console to ensure that the job ran with a return code of 0. If the return code is not 0, the console should indicate the error. Correct the error and rerun DSQ3*n*INS.

Updating QMF SQL Packages

If QMF SQL packages are changed with a PTF, then the packages must be loaded into each database where QMF is installed. Use the original installation job, DSQ3EDBI, to update the packages.

1. Modify the SETPARMs in the QMF installation job, DSQ3EDBI, as follows:

```
// SETPARM STEP1=NO        *-- CREATE QMF CONTROL TABLES INSQLDB
// SETPARM STEP2=YES        *-- LOAD QMF PACKAGES INTO SQL DB
// SETPARM STEP3=NO        *-- LOAD QMF SAMPLES INTO SQL D B
```

2. Locate the `// SETPARM DBNAME=SQLDS` parameter and verify or change SQLDS to the name of the database that you are using.
3. File and run the job. Ensure that the job ran with a return code of 0. If the job did not run with a return code of 0:
   a. Check for an error message on the system console.
   b. Check the list output to find the cause of the problem.
   c. Correct the problem.
   d. Rerun DSQ3EDBI.
   e. Recheck the return code.
4. Repeat the procedure to load the packages into every database.

Part 2. Managing QMF for VSE/ESA

Chapter 6. Starting QMF 49
Before You Start QMF 49
Quick Start 49
Add QMF to the VSE/ESA Function Selection Menu 50
Starting QMF from a Cleared CICS Screen . . 51
Starting QMF from a CICS Application . . . 51
Starting a Noninteractive Session 52
Starting an Interactive Session 52

Chapter 7. Customizing Your Start Procedure 55
Quick Start 55
Customizing Report Storage and Report Performance 56
Adjusting GETVIS Storage Used for Report Data (DSQSBSTG) 56
Choosing the Right Amount of GETVIS Storage for Each User 57
Performance Tradeoffs 58
Acquiring Extra Temporary Storage (DSQSPILL) 58
Estimating the Space Required for a Spill File 59
Using a Spill File in a Noninteractive QMF Session 61
Solving Some Spill File Problems . . . 61
Specifying the Name of Spill Storage (DSQSSPQN). 62
Controlling the Number of Report Rows Retrieved for Display (DSQSIROW) . . . 63
Performance with Small DSQSIROW Values 63
Performance with Large DSQSIROW Values 64
Tracing QMF Activity at the Start of a Session 64
Setting the Level of Trace Detail (DSQSDBUG) 65
Specifying the Type of CICS Storage for Trace Data (DSQSDBQT) 66
Specifying the Name of CICS Storage for Trace Data (DSQSDBQN) 66
Controlling Initial Activities during a Session 67
Connecting to the Database (DSQSUSER) 67
Starting a Noninteractive QMF Session (DSQSMODE) 69
Naming a Procedure to Run When QMF Starts (DSQSRUN) 69
Running an Initial Procedure Noninteractively 70
Performing Interactive QMF Work with an Initial Procedure 71
Passing Variable Values to an Initial Procedure 72
Setting Printing for Double-Byte Character Set Data (DSQSDBCS) 75

Chapter 8. The QMF Session Control Facility 77
Installing or Removing Q.SYSTEM_INI . . . 77
When Does the Q.SYSTEM_INI Procedure Run? 77
Using Q.SYSTEM_INI. 77
Example Shipped with QMF 77
User Session Procedure Example 78
Procedure that Displays an Object list . . 79
Security and Sharing Session Procedure. . . 80
Diagnosis Considerations 80

Chapter 9. Establishing QMF Support for End Users 81
Quick Start 81
Creating User Profiles to Enable User Access to QMF 82
Using the Q User Profile, a Special QMF Profile 82
Establishing a Profile Structure for Your Installation 82
Adding a New User Profile to the Q.PROFILES Table 83
Preventing Users Without Unique Profiles from Using QMF 84
Reading the Q.PROFILES Table 84
Providing the Correct Profile for the User's Operating Environment 88
Storing Profiles in VM DB2 in a Guest-Sharing Environment. 89
Updating User Profiles 89
Using the SET PROFILE Command . . 89

Using SQL UPDATE Statements . . . 90
Updating the SYSTEM Profile 91
Deleting Profiles from the Q.PROFILES Table 91
Controlling Access to QMF and Database Objects. 92
SQL Privileges Required to Access Objects 92
SQL Privileges Required for QMF Commands 92
SQL Privileges Required for Prompted and QBE Queries 93
SQL Privileges Required for the Table Editor 94
Granting and Revoking SQL Privileges . . 94
Using the SQL GRANT Statement. . . 94
Using the SQL REVOKE Statement . . 95
Sharing QMF Objects with Other Users . . 95
Allowing Uncommitted Read 96
Setting Standards for Creating Objects . . 96
Customizing a User's Database Object List . . 97
Using the Default Object Lists 97
Changing the Default List 98
Object List Storage Requirement 100
Enabling Users to Create Tables in the Database. 100
Choosing and Acquiring a dbspace for the User 102
Using the SQL ACQUIRE Statement 102
Sizing a dbspace 102
Granting a User DB2 RESOURCE Authority 102
Enabling Users to Confirm Table Changes Before They are Made 103
Enabling Users to Support a Chart 104
Maintaining QMF Objects Using QMF Control Tables 104
Reading the Q.OBJECT_DIRECTORY Table 104
Reading the Q.OBJECT_DATA Table . . 105
Reading the Q.OBJECT_REMARKS Table 106
Listing QMF Queries, Forms, and Procedures 107
Displaying QMF Queries, Forms, and Procedures 107
Transferring Ownership of Queries, Forms, and Procedures 108
Deleting Obsolete Queries, Forms, and Procedures 108
Enlarging the dbspace for the QMF Object Control Tables 109
Maintaining Tables and Views Using DB2 System Tables 110
Listing Tables and Views 110
Transferring Ownership of a Table or View 111
Deleting a Table or View from the Database 111
Enabling English Support in an NLF Environment 111
Using Global Variables to Define the Currency Symbol 112

Chapter 10. Enabling Users to Print Objects 113
Quick Start 113
Printing Objects 114
Deciding Whether to Use QMF or GDDM Services for Printing 115
Using GDDM services to Handle Printing 116
Choosing a GDDM Nickname for Your Printer 116
Choosing the Right Type of GDDM Device 117
Creating the Nickname Specification 117
Example Nickname for a Family 1 or 2 GDDM Printer 118
Example Nickname for a Family 3 GDDM Printer 119
Defining Multiple Nicknames with One Definition 119
Examples of Nickname Definitions . . 119
Updating the GDDM Defaults Module (ADMADFC) with the Nickname . . . 120
Linking the Nickname with a Physical Device 121
Linking a Family 1 or 2 Nickname with a Physical Device 121
Linking a Family 3 Nickname with a Physical Device 121
How QMF Interfaces with your GDDM Nickname 122
Using QMF Services to Handle Printing . . 123
Choosing Between Temporary Storage Queues and Transient Data Queues . . . 123
Using the PRINT Command to Route Output to Queues. 123
Using Global Variables to Define Queues for Printing 124
Printing to VSE POWER using QMF . . 124
Modifying Your CICS Startup JCL . . 124

Modifying Your DCT 125
Modifying Your Synonym and Function Key Tables 125
Sample Program to Segment POWER Output 126
Creating the QMF Procedures. . . . 129
Modifying Your User's Profile. . . . 130
Using Your New Print Procedure . . 130
Updating User Profiles to Enable GDDM Printing 130

Chapter 11. Customizing QMF Commands 133
Quick Start 133
Creating the Command Synonym Table . . 133
Entering Command Synonym Definitions into the Command Synonym Table 135
Choosing a Verb 136
Rules for the VERB Column 136
Using Base QMF Verbs as Command Synonym Verbs 136
Choosing an Object Name 137
Choosing the Synonym Definition . . . 137
Using a Linear Procedure in the Synonym Definition 138
Using Variables in the Synonym Definition 139
Keying Information into the SYNONYM_DEFINITION Column . . 140
Activating the Synonyms 141
Minimizing Maintenance of Command Synonym Tables 142
Assigning One Synonym Table to All Users 142
Assigning Views of a Synonym Table to Individual Users 143
Synonyms for Public or Private Use 143
Synonyms for Public or Group Use 143
Synonyms Paired with an Authorization Table 144

Chapter 12. Customizing QMF Function Keys 145
Quick Start 145
Choosing the Keys You Want to Customize 145
Default Keys on Full-Screen Panels . . . 146
Default Keys on Window Panels. . . . 147
Creating the Function Key Table 148
Entering Your Function Key Definitions into the Table. 149
Linking a Command with a Function Key 149
Labeling the Function Key and Positioning It on the Screen 150
Examples of Key Definitions 151
Entering a Definition for a Key on a Full-Screen Panel 151
Entering a Definition for a Key on a Window Panel 152
Entering a Key Definition for a Help or Prompt Panel 152
Identifying the Panel You Want to Customize 153
Full-Screen Panel Identifiers 153
Window Panel Identifiers 153
Command Windows. 154
Forms Windows 154
Global Variable Windows 154
Help and Prompt Windows 154
Location Windows 154
Object List Windows. 154
Prompted Query Windows. 155
Activating New Function Key Definitions 156

Chapter 13. Creating Your Own Edit Codes for QMF Forms 159
Quick Start 159
Choosing an Edit Code 160
Calling Your Exit Routine to Format the Data 161
Passing Information To and From the Exit Routine 164
Fields of the Interface Control Block . . 165
Fields That Characterize the Input Area 166
How U-Type Edit Codes are Represented in the Input Area . . . 167
How V-Type Edit Codes are Represented in the Input Area . . . 167
Fields That Characterize the Output Area 167
Passing Control to the Exit Routine When QMF Terminates 167
Writing an Edit Routine in High-Level Assembler (HLASM). 168
How an HLASM Edit Routine Interacts with CICS 168
How an HLASM Edit Routine Interacts with QMF 169
Translating Your Program 172
Assembling Your Program 173
Link-Editing Your Program 173
Example JCL Statements 173
Defining the Edit Exit Phase to CICS . . 175

Writing an Edit Routine in VS COBOL II or COBOL for VSE/ESA 175
Using Literal Values in a COBOL Program 176
How a COBOL Edit Routine Interacts with CICS 176
How a COBOL Edit Routine Interacts with QMF 178
Translating Your Program 183
Compiling Your Program 184
Link-Editing Your Program 184
Example JCL Statements 184
Defining the Edit Exit Phase to CICS . . 186
Writing an Edit Routine in PL/I 186
Writing an Edit Routine in PL/I for CICS 186
How a PL/I Edit Routine Interacts with CICS 187
Translating Your Program 188
Link-Editing Your Program 188
CICS Program Definition 188
Example JCL Statements 188
How a PL/I Edit Routine Interacts with QMF 190
Handling Double-Byte Character Set Data 196
Edit Codes for DBCS Data 196
What the Edit Routine Receives 196
Data from Graphic Columns 196
Data from Character Columns . . . 196
Ensuring the Edit Routine Returns the Right Results 197
Overflowing the ECSRSLT Field . . . 197
Printing the Report Column 197

Chapter 14. Controlling QMF Resources Using a Governor Exit Routine 199
Quick Start 199
Using the IBM-Supplied Governor Exit Routine 199
Activating the Default Limits for Number of Rows Retrieved 200
How a Governor Exit Routine Controls Resources 202
How the Governor Knows What the Resource Limits Are 202
How the Governor Knows When You Reach a Resource Limit 203
What Happens When You Reach a Resource Limit. 204
Defining Your Own Resource Limits . . 204
Creating your own Resource Control Table 206
Modifying the IBM-Supplied Governor Exit Routine or Writing Your Own. 208
Program Components of the Governor Exit Routine 208
How CICS Interfaces with the Governor Exit Routine 209
How and When QMF Calls the Governor Exit Routine. 211
Points at Which QMF Calls the Governor 211
What Happens Upon Entry to the Governor Exit Routine 212
Establishing Addressability for Function Calls 213
Passing Resource Control Information to the Governor Exit. 215
Structure of the DXEGOVA Control Block 215
Addressing the Resource Control Table 219
Structure of the DXEXCBA Control Block 220
Storing Resource Control Information for the Duration of a QMF Session 227
Canceling User Activity. 228
Providing Messages for Canceled Activities 228
Translating, Assembling, and Link-Editing Your Governor Exit Routine 229
Translating Your Governor Exit Program for CICS 229
Assembling Your Governor Exit 230
Link-Editing Your Governor Exit Routine 230
Example JCL Statements 230

Chapter 15. Troubleshooting and Problem Diagnosis 233
Quick Start 233
Troubleshooting Common Problems. . . . 234
Handling Initialization Errors 234
Handling Warning Messages 234
Handling GDDM Errors During Printing 235
Handling QMF Errors During Printing 237
Handling Display Errors 237
Using the HEX Function 237
Using QMF-Provided Hex and Bit Edit Codes. 238
Handling Binary Data with User-Written Edit Routines. 238

Solving Slow Performance Problems . . 238
Resetting the Data Object to Improve Performance 238
Increasing the User's Report Storage 239
Increasing the Size of the CICS Partition 239
Determining the Problem Using Diagnosis Aids 240
Choosing the Right Diagnosis Aid for the Symptoms 240
Diagnosing Your Problem Using QMF Message Support 240
Determining which QMF Function Issued an Error Message 241
Handling System Error Messages . . 242
Handling SQL Return Codes 242
Using the QMF Trace Facility 242
Allocating Storage for Trace Data . . 243
Starting the Trace Facility 244
Getting the Right Level of Detail in Your Trace Output 244
Tracing at the Module Level 246
Viewing QMF Trace Data 247
Determining the QMF Service Level 247
Turning off the Trace Facility 248
Using CICS Diagnostic Facilities 248
Identifying QMF in CICS Diagnostic Output 248
Defining the Display for a CICS Abend Message 249
Using Error Log Reports from the Q.ERROR_LOG Table 250
Reporting a Problem to IBM 251
Using ServiceLink to Search for Previously Reported Problems 251
Working with Your IBM Support Center 254

Chapter 6. Starting QMF

This chapter describes the methods you can use to start QMF. You can start QMF from a cleared CICS screen or from a CICS application.

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

Before You Start QMF

Before you start QMF, you need to establish the connection between CICS and DB2. The CIRB transaction connects a CICS partition to VSE DB2. It establishes the DB2 online support, allowing users at CICS terminals to communicate with the VSE DB2 application server through CICS.

CIRB is usually run as part of the job that starts the CICS partition. You can also run CIRB as follows:

1. Start the VSE DB2 application server in multiple user mode, according to instructions in *DB2 Server for VSE System Administration*.
2. Run the CICS CIRB transaction once for each CICS partition where QMF is installed. You can run the transaction three ways:
   - By starting it from any CICS terminal
   - By starting it from the VSE console
   - Automatically when CICS comes up, provided you make the appropriate changes in the CICS startup facilities

   For more information about online support and using the CIRB transaction, see *DB2 Server for VSE System Administration*.

Quick Start

Table 4 outlines ways to start QMF. For information about the command syntax for starting QMF with parameters, see “Chapter 7. Customizing Your Start Procedure” on page 55.

The *n* symbol in each example represents the national language identifier (NLID). Substitute the NLID from Table 3 on page 10 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.

Starting QMF

Table 4. Options for starting QMF

To do this task:	See:
To start QMF from the VSE/ESA Function Selection Menu, using procedures described in the VSE/ESA documentation for your VSE system. You can add QMF to the VSE/ESA Function Selection Menu with or without values for the QMF program parameters explained in "Chapter 7. Customizing Your Start Procedure" on page 55.	Page 50
To start QMF from a cleared CICS screen, enter: `QMFn` Follow `QMFn` with values for the QMF program parameters explained in "Chapter 7. Customizing Your Start Procedure" on page 55.	Page 51
To start QMF from a CICS application, use the command `EXEC CICS START TRANSID('QMFn') FROM('...') TERMID('name')`, and enclose values for the QMF program parameters in single quotes following the FROM keyword. You can use any QMF program parameter in a CICS application. A terminal ID (TERMID) is required for interactive sessions, optional for noninteractive.	Page 51

Add QMF to the VSE/ESA Function Selection Menu

QMF is provided as a standard CICS transaction. As such you can add the invocation of QMF to the VSE/ESA Function Selection Menu. To add QMF, you use procedures as described in the VSE/ESA documentation for your VSE system. The basic steps are:

1. Create a new Application Profile for QMF.
2. Add or Change the QMF Application Profile.
   a. Ensure that the CODE field specifies QMF as a NON-CONVERSATIONAL transaction with data.
   b. Specify the name of the QMF transaction ID, QMFn, in the ACTIVATE field where n is the nlfid. The QMF transaction ID for English is QMFE.
   c. Specify any QMF program parameters that you want to use in the ″DATA″ field. See "Chapter 7. Customizing Your Start Procedure" on page 55 for more information.
3. Add the new QMF application profile to a selection panel.

After you add QMF to the VSE/ESA function selection menu, the menu might look like this:

```
Enter the number of your selection and press the ENTER key:

      1  Operations
      2  Problem Handling
      3  Program Development
      4  Command Mode
      5  CICS-Supplied Transactions
```

```
6  CIRB - Start SQL Connection
7  ISQL - Interactive SQL Facility
8  QMF  - Query Management Facility
```

Starting QMF from a Cleared CICS Screen

QMF runs as a conversational transaction in CICS, and is defined in CICS resource tables during QMF installation. You can start QMF by issuing the QMFn transaction from a cleared CICS screen, as shown in the example in Figure 1.

```
QMFE B=600000,F=200,L=YES
```

Figure 1. Starting QMF from a cleared CICS screen

The letters following QMFE represent abbreviated forms of some of the QMF program parameters you can use to customize the behavior of a QMF session. For example, the values shown here start an interactive English QMF session, retrieve 200 rows of data before displaying the first screen of the report, and activate extra storage for report data when the amount of data retrieved into GETVIS storage reaches 600 000 bytes.

You can specify the parameters in any order on the QMFn transaction. Ensure you:

- Specify each value in a *parameter_name=value* format. You can use the short form if the parameter has one.
- Specify only one value for each parameter.
- Enter a blank, a comma, or both after each value.
- Capitalize all letters in the parameter string.

QMF uses default values explained in “Chapter 7. Customizing Your Start Procedure” on page 55 for any parameter you don’t enter following the QMFn transaction. The values you supply remain effective throughout the QMF session, except for the parameter that specifies the level of detail in the trace data. Users can change this trace parameter directly from their profiles using the SET PROFILE command. See “Creating User Profiles to Enable User Access to QMF” on page 82 for more information on user profiles.

Starting QMF from a CICS Application

QMF can interact with existing QMF applications you might have at your site. You can use the EXEC CICS START command with the transaction ID QMFn to start a QMF session from within a CICS application. An example of the command is shown in Figure 2 on page 52. Replace the n symbol with the

NLID from Table 3 on page 10 that matches the national language you're using.

```
EXEC CICS START TRANSID('QMFn') FROM('M=B,I=START_PROC,UID=Q/QMF') TERMID('MYT5')
```

Figure 2. Starting QMF from a CICS application

The command in Figure 2 starts a noninteractive QMF session, connects QMF to DB2 using a user ID of Q and a password of QMF, then runs a QMF procedure named START_PROC.

Use the same rules for passing QMF program parameters as you would to start QMF from a cleared CICS screen, as discussed in "Starting QMF from a Cleared CICS Screen" on page 51. You can use any QMF program parameter in a CICS application.

A terminal ID (TERMID) is required for an interactive session (when DSQSMODE = I), and optional for a noninteractive session (when DSQSMODE = B). If the terminal ID specifies the terminal where the calling CICS application is running, the QMF session starts when the CICS application finishes. If you do specify a terminal ID, the terminal must exist and be available. Also ensure the ID is defined as either a local or a remote terminal on the system in which the START command is issued.

If you do not know the TERMID, issue the EXEC CICS ADDRESS EIB(xxx) parameter to retrieve it.

Starting a Noninteractive Session

You might choose to run a noninteractive QMF session to conserve resources, as explained in the example in "Running an Initial Procedure Noninteractively" on page 70. In this case, use a value of B for the DSQSMODE parameter and ensure you use the DSQSRUN parameter to pass the name of an initial procedure to perform the necessary QMF tasks. These parameters are explained in "Controlling Initial Activities during a Session" on page 67. You might also choose to use the DSQSUSER parameter to ensure you connect to the database using the appropriate SQL authorization ID and password.

If you do not specify a terminal ID, the QMF session runs without a terminal.

Starting an Interactive Session

You might also choose to start an interactive QMF session from within a CICS application. For example, the CICS application might be a menu application that allows users to start QMF from a menu of other products.

A terminal ID is required to start an interactive session. Because the session runs interactively, you don't need to supply an initial procedure that runs when QMF starts, nor do you need to supply a value for the DSQSMODE parameter. If you want to connect to DB2 explicitly, supply values for the DSQSUSER parameter; otherwise, QMF connects to DB2 using the default VSE operator ID and password defined in the system catalog.

Chapter 7. 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 5 shows how to use the program parameters to customize aspects of the QMF session. The command syntax in the examples applies to starting QMF from a cleared CICS screen. If you start QMF from a CICS application, use the same names and values for the program parameters, but use the command syntax shown in "Starting QMF from a CICS Application" on page 51.

The *n* symbol in each example represents the national language identifier (NLID). Substitute the NLID from Table 3 on page 10 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 5. Passing parameters

To do this task:	See:
To set limits on the amount of GETVIS storage used for QMF queries and reports, use the DSQSBSTG parameter if you want any limit other than 500 000 bytes. For example, to specify a limit of 1 000 000 bytes: `QMFn B=1000000`	Page 56
To use temporary storage (a spill file) as extra storage for report data, use the DSQSPILL parameter. For example, enter: `QMFn L=YES`	Page 58
To name the spill file something other than DSQSnnnn (where nnnn is the CICS terminal ID), use the DSQSSPQN parameter. For example, to indicate the name MYSPILL, enter: `QMFn DSQSSPQN=MYSPILL`	Page 62
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: `QMFn F=200`	Page 63
To log QMF activity in the trace data, including activity before the user's profile is established, use the DSQSDBUG parameter. For example, enter: `QMFn T=ALL`	Page 65

Table 5. Passing parameters (continued)

To do this task:	See:
To indicate that you want to use temporary storage (TS) rather than transient data (TD) queues for trace data, use the DSQSDBQT parameter. For example, enter: `QMFn DSQSDBQT=TS`	Page 66
To name the queue for trace data (whether temporary storage or transient data) something other than DSQD, use the DSQSDBQN parameter. For example, to get a temporary storage queue named MYTRACE, enter: `QMFn DSQSDBQN=MYTRACE`	Page 66
To use an ID other than the default VSE operator ID to connect to the database when starting QMF, use the DSQSUSER parameter to specify the user ID and password. For example, for user JONES who has the password MYPW, enter: `QMFn UID=JONES/MYPW`	Page 67
To run QMF without user interaction (either with or without a terminal ID), use the DSQSMODE parameter and specify an initial procedure using the DSQSRUN parameter. You might also choose to use the DSQSUSER parameter to ensure you connect to the database using the proper ID. For example, to do some noninteractive QMF work using the Q user ID and an example procedure named STARTPROC, enter: `QMFn M=B, UID=Q/QMF, I=STARTPROC`	Page 69
To run an initial procedure when QMF starts, use the DSQSRUN parameter. For example, to run a procedure called STARTPROC, enter: `QMFn I=STARTPROC`	Page 69
To print DBCS data from non-DBCS terminals, use the DSQSDBCS parameter. For example, enter: `QMFn K=YES`	Page 75

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 GETVIS storage. This section explains QMF program parameters that help you customize:

- The maximum amount of GETVIS storage used for report data
- Auxiliary storage used when GETVIS storage for reports is full
- How many rows of data are retrieved before QMF displays the first screen of the report

Adjusting GETVIS Storage Used for Report Data (DSQSBSTG)

Parameter name

DSQSBSTG

Short form

B

Valid values
From 0 to 99 999 999 bytes
Default
500 000 bytes

To produce reports and temporarily store data, QMF uses GETVIS storage, which is virtual storage within the CICS partition. VSE/ESA 1.3 limits GETVIS storage according to the partition size you define for CICS. To ensure each user has enough storage for QMF queries and reports, first adjust the CICS partition size according to the number of QMF users and the size and complexity of reports they are creating. "QMF Storage Requirements" on page 4 provides guidelines for sizing the CICS partition.

After you size the CICS partition, use the DSQSBSTG parameter to specify the maximum amount of GETVIS storage QMF uses to run queries and produce reports. Specify the storage amount in bytes. The user can specify the GETVIS storage from a cleared CICS screen.

For example, the following command starts QMF from a cleared CICS screen and specifies that a maximum of 0.8 MB of GETVIS storage can be used to store the user's report data:

```
QMFn B=800000
```

Choosing the Right Amount of GETVIS Storage for Each User

QMF needs a minimum amount of GETVIS storage to display a report in the default format. This minimum is between 15 000 and 31 000 bytes (15 to 31 KB), depending on how the storage in your CICS partition is distributed. Set DSQSBSTG to zero when you want QMF to use the minimum value for GETVIS storage.

The default value of 0.5 MB can accommodate most QMF transactions. However, the amount of virtual storage needed varies for users using a report format other than the default, or for users working with very large reports. These users might need up to 1 MB or more of virtual storage. See *QMF Reference* for information on report formatting options.

Important: QMF requires a minimum of 15 MB GETVIS storage for up to 20 users (24 MB total virtual storage for the partition). When you increase a user's GETVIS storage using the DSQSBSTG parameter or when you add more QMF users, ensure you increase the value of the CICS ALLOC parameter so each user has enough GETVIS storage to run queries and produce reports. A QMF transaction might time out waiting for storage to become available if there is not enough GETVIS storage to support the activity of all QMF users in the partition.

Performance Tradeoffs

You can use the DSQSPILL parameter to provide users with a spill file. If the spill file is full, QMF continues to retrieve data into GETVIS storage in amounts specified by the DSQSBSTG parameter. Thus, if you use too low a value for DSQSBSTG, performance might be poor even if you use 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 GETVIS 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 GETVIS storage is used for queries and reports. For more information about governor exit routines, see "Chapter 14. Controlling QMF Resources Using a Governor Exit Routine" on page 199.

Acquiring Extra Temporary Storage (DSQSPILL)

Parameter name
DSQSPILL

Short form
L

Valid values
YES or NO

Default
NO (no spill file is used)

Because large amounts of report data in GETVIS storage might affect the operation of other CICS transactions, QMF allows you to allocate a spill file (auxiliary temporary storage used when a user's GETVIS 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.

Set the DSQSPILL parameter to YES to activate the spill file:

```
QMFn L=YES
```

Data is written to the spill file until:

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

- The data retrieved into the spill file exceeds 32 767 rows, the maximum amount a CICS temporary storage queue can hold. (Each row can hold 4K of data.)

Estimating the Space Required for a Spill File

Temporary storage for the spill file is limited to 32 767 buffers of size 4 KB each. If the data written to the spill file goes over this limit, QMF does not use the data from the spill file, but instead retrieves it again from the database, using GETVIS storage to hold it.

To accommodate QMF's storage requirements, ensure CICS temporary storage file DFHTEMP 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 DFHTEMP 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 6 on page 60). See Table 7 on page 60 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 6. 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 7 shows a sample calculation for a spill file.

Table 7. Sample row width calculation for a spill file

Content of Row	Calculation	Contribution to Width
Two SMALLINT columns	2 x 4 =	8 bytes
One INTEGER column		6 bytes
One DECIMAL(3,2) column	(3+1)/2+2 =	4 bytes
One DECIMAL(6,0) column	(6+2)/2+2 =	6 bytes
One FLOAT column		10 bytes
One CHAR(10) column	10 + 2 =	12 bytes
One VARCHAR(16) column	16 + 4 =	20 bytes
Total width of row		59 bytes

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

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

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

When R=6000, 2 buffers/row:

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

Using a Spill File in a Noninteractive QMF Session

A spill file is most useful for improving performance in an interactive QMF session, when the DSQSMODE parameter is set to I. 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 “Starting a Noninteractive QMF Session (DSQSMODE)” on page 69.

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 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 have 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.

Specifying the Name of Spill Storage (DSQSSPQN)

Parameter name

DSQSSPQN

Short form

(no short form)

Valid values

Any name that follows CICS naming conventions for queues

Default

DSQSnnnn (nnnn is the CICS terminal ID)

When you choose to use a spill file, you can also specify a name for the CICS temporary storage queue to use for QMF spill data. For example, to specify the name MYDATA:

```
QMFn DSQSSPQN=MYDATA
```

If you start a noninteractive QMF session from within a CICS application and you choose not to specify a CICS terminal ID, then you need to code the

DSQSSPQN parameter. You must explicitly specify a value for DSQSSPQN, or QMF does not start. For more information on starting QMF without a terminal ID, see "Starting a Noninteractive Session" on page 52.

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

Parameter name
DSQSIROW

Short form
F

Valid values
Any number from 0 through 99 999 999

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 59 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.

See “Resetting the Data Object to Improve Performance” on page 238 for a list of commands that complete the data object.

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 “Starting a Noninteractive QMF Session (DSQSMODE)” on page 69 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 GETVIS storage for the partition is full, QMF waits for virtual storage to become available to finish retrieving rows for the database. When you plan your values for DSQSBSTG and DSQSIROW, remember that QMF might time out waiting for storage to become available.

Tracing QMF Activity at the Start of a Session

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

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

Setting the Level of Trace Detail (DSQSDBUG)

Parameter name
DSQSDBUG
Short form
T
Valid values
ALL or NONE
Default
NONE (no trace data)

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

The tracing you set with this parameter is effective until the user issues a `SET PROFILE (TRACE=value` command to change it, or, in the case of NONE, until the profile is loaded. For more information on valid trace values, see "Getting the Right Level of Detail in Your Trace Output" on page 244.

Set DSQSDBUG to ALL when you want to trace QMF activity at the highest level of detail, including program initialization errors and other errors that might occur before the user's profile is established:

```
QMFn T=ALL
```

When you use a value of ALL, ensure the type of storage queue you choose is large enough to hold the trace output. "Specifying the Type of CICS Storage for Trace Data (DSQSDBQT)" on page 66 explains how to specify the queue type when you start QMF.

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:

- 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.

"Starting a Noninteractive QMF Session (DSQSMODE)" on page 69 explains interactive and noninteractive sessions in more detail.

After QMF starts, you can turn tracing off by using the command `SET PROFILE (TRACE=NONE`. You can also set more specific levels of trace detail using this

command, by replacing NONE with various values that represent different QMF functions. See “Using the QMF Trace Facility” on page 242 for more information.

Specifying the Type of CICS Storage for Trace Data (DSQSDBQT)

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

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

```
QMFn DSQSDBQT=TS
```

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

A transient data queue named DSQD is predefined for you during QMF installation. If you use the DSQSDBQN parameter to name the transient data queue something other than DSQD, you must predefine the queue to CICS before you use it for the first time. Use the definition shown in Figure 101 on page 243 as an example.

For more information on specifying the amount of detail in the QMF trace and viewing trace data, see “Using the QMF Trace Facility” on page 242.

Specifying the Name of CICS Storage for Trace Data (DSQSDBQN)

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

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

If you specify transient data for DSQSDBQT and you want to name the queue something other than DSQD, define the queue in the CICS DCT if it is not yet available. Figure 101 on page 243 shows the DCT entries for the DSQD queue.

Ensure the queue name conforms to CICS specifications for the type of queue specified by DSQSDBQT. TD queues have names from 1 to 4 characters. TS queues have names from 1 to 8 characters.

You do not need to predefine temporary storage queues to CICS. For example, the following statement dynamically allocates a temporary storage queue named MYTRACE to hold trace data for the QMF session:

```
QMFn DSQSDBQN=MYTRACE
```

QMF issues CICS ENQ and DEQ commands around single trace entries in the queue, so that a single queue can be used by more than one user. See "Viewing QMF Trace Data" on page 247 for information on how to view the trace after it is written to the queue.

Controlling Initial Activities during a Session

This section explains program parameters that help you control initial QMF activities, such as:

- Specifying an ID and password for the connection to the database
- Starting a noninteractive session (with or without a CICS terminal ID)
- 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 without a CICS terminal ID, 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 temporary storage queue for later analysis.

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

Connecting to the Database (DSQSUSER)

Parameter name
DSQSUSER

Short form
UID

Valid values
ID and password that conform to CONNECT command rules

Default

3-character VSE operator ID and password defined in the DB2 system catalog

When a user starts QMF, DB2 uses an authorization ID to determine whether the user is authorized to connect to the database. DB2 uses this same ID to determine a user's authorization to access objects and perform database activities.

You can use the DSQSUSER parameter to provide DB2 with an authorization ID and password to use for the database connection. For example, the following command connects user JONES, who has a password of MYPW:

```
QMFn UID=JONES/MYPW
```

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

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

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

- The DB2 authorization ID and password you supply for DSQSUSER must conform to the rules for the CONNECT command for VSE DB2. For more information about these rules, see the explanation of the CONNECT command in *DB2 Server for VSE & VM SQL Reference*.
- The SQL authorization ID and password must both be in the DB2 system table SYSTEM.SYSUSERAUTH. For more information about this table, see *DB2 Server for VSE & VM SQL Reference*.

If you don't supply an SQL authorization ID and password, DSQSUSER defaults to the 3-character default VSE operator ID and password defined in the DB2 system catalog. You can issue the following SQL statements from the SQL query panel at any time during the QMF session to determine the ID that DB2 is currently using for database authorization:

```
SELECT DISTINCT USER FROM Q.ORG
```

If you supply a user ID but no password, QMF displays an error message. The password you supply doesn't have to be identical to the password associated with the VSE logon ID.

For more information on setting up a DB2 authorization ID, see *DB2 Server for VSE System Administration*.

Starting a 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.

Use a value of B to start a noninteractive session:

```
QMFn M=B,I=STARTPROC
```

Although a value of B specifies QMF batch mode, the QMF session does not run in the BG partition in CICS and cannot use VSE's Batch Utility services.

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 DSQSUSER parameter to specify an ID and password for the database connection, if you do not want to use the default VSE operator ID and password.

When you use the DSQSMODE parameter from a cleared CICS screen, as shown in Figure 1 on page 51, the QMF session runs using the ID of the CICS terminal where the command was issued. You can run a noninteractive QMF session without a terminal if you start QMF from a CICS application. For more information, see "Starting a Noninteractive Session" on page 52.

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:

```
QMFn 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:

```
QMFn 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 query in the following example. 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 the query in the following example. For this example, we'll call this QMF procedure INVENTORY_PROC:

```
RUN QUERY INVENTORY_QUERY
PRINT REPORT (QUEUENAME=Q1,QUEUETYPE=TS)
EXIT
```

The procedure includes an EXIT command because, when QMF is running noninteractively, no user is present to end the QMF session. EXIT ends the QMF session and frees the resources being held by QMF. Always use an EXIT command in an initial procedure that runs noninteractively.

Because the tasks involved in creating the report do not change (only the data changes), you could use the DSQSRUN parameter to query the INVENTORY table off-shift Sunday night and print the report to the storage queue named Q1, so you can have it Monday morning:

```
QMFn I=INVENTORY_PROC, M=B
```

You might later use the CICS CEBR transaction to browse the Q1 temporary storage queue.

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. For this example, we'll call this query INVENTORY_QUERY:

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

The procedure you use to run this query might look something like the query in the following example. For this example, we'll call this QMF procedure INVENTORY_PROC. 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 could then use the DSQSRUN parameter without specifying the DSQSMODE parameter, so that you start an interactive session for the user:

```
QMFn I=INVENTORY_PROC
```

The INVENTORY_PROC procedure prompts the user for the &LOWSTOCK variable value. For 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.

When you specify the name of an initial procedure, QMF issues a RUN PROC command that runs the procedure. When you use variables in your procedure,

the values you supply for these variables must conform to the syntax used for passing variables on a RUN command. For information about this syntax, see *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 3 shows an example query and procedure. The figure also shows how to pass a value for the variable when you enter the DSQSRUN parameter, and shows the RUN PROC command that QMF issues.

Query (named JONES.QUERY2)

```
SELECT NAME, &COL
FROM Q.STAFF
```

Procedure (named JONES.PROC2)

```
RUN QUERY JONES.QUERY2 (&&COL=&COL
```

DSQSRUN parameter

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

Resulting RUN command

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

Figure 3. Passing a QMF column name using DSQSRUN

Figure 4 shows a similar example, but instead of passing one column name to the procedure, it allows you to pass several, which return the employee's name, the department, and the employee's salary.

Query (named JONES.QUERY3)

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

Procedure (named JONES.PROC3)

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

DSQSRUN parameter

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

Resulting RUN command

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

Figure 4. 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 5. When you pass the character string MGR on the DSQSRUN parameter, be sure to enclose the value in single quotes.

```
Query (named JONES.QUERY4)
        SELECT JOB, NAME, ID
        FROM Q.STAFF
        WHERE JOB=&JOB
Procedure (named JONES.PROC4)
        RUN QUERY JONES.QUERY4 (&&JOB=&JOB
DSQSRUN parameter
        QMFn I=JONES.PROC4(&JOB='MGR')
Resulting RUN command
        RUN PROC JONES.PROC4 (&JOB='MGR')
```

Figure 5. Passing a string within single quotes using DSQSRUN

Figure 6 shows how to pass variable values that contain commas. Enclose the value SAN JOSE, CA in single quotes because it contains a comma.

```
Query (named JONES.QUERY5)
        SELECT *
        FROM Q.APPLICANT
        WHERE ADDRESS=&CITY
Procedure (named JONES.PROC5)
        RUN QUERY JONES.QUERY5 (&&CITY=&CITY
DSQSRUN parameter
        QMFn I=JONES.PROC5(&CITY='SAN JOSE,CA')
Resulting RUN command
        RUN PROC JONES.PROC5 (&CITY='SAN JOSE,CA')
```

Figure 6. Passing a comma within a string using DSQSRUN

Figure 7 on page 75 shows how to pass variable values that contain single quotes (for example, an apostrophe in a name). When you pass the value on the DSQSRUN parameter, be sure to enclose the value in single quotes and use two single quotes for the apostrophe instead of one.

```
Query (named JONES.QUERY6)
        SELECT *
        FROM Q.STAFF
        WHERE NAME=&NAME
Procedure (named JONES.PROC6)
        RUN QUERY JONES.QUERY6 (&&NAME=&NAME
DSQSRUN parameter
        QMFn I=JONES.PROC6(&NAME='O''BRIEN')
Resulting RUN command
        RUN PROC JONES.PROC6 (&NAME='O''BRIEN')
```

Figure 7. Passing an apostrophe as part of a string using DSQSRUN

Figure 8 shows how to pass values for variables in two different parts of the query.

```
Query (JONES.QUERY7)
        SELECT *
        FROM Q.STAFF
        WHERE DEPT IN &DEPT
        AND JOB  = &JOB
Procedure (named JONES.QUERY7)
        RUN JONES.QUERY7 (&&DEPT=&V1 &&JOB=&V2
DSQSRUN parameter
        QMFn I=JONES.PROC7(&V1=(((10,38))) &V2='MGR')
Resulting RUN command
        RUN PROC JONES.PROC7(&V1=(((10,38))) &V2='MGR')
```

Figure 8. 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 cleared CICS screen and allows the user to print DBCSTABLE using a command such as PRINT DBCSTABLE (PRINTER=DBCSPRT.

```
QMFU K=YES
```

For more information on how to establish a GDDM nickname for the DBCSPRT printer, see “Chapter 10. Enabling Users to Print Objects” on page 113.

Chapter 8. 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')
```

When Does the Q.SYSTEM_INI Procedure Run?

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

Using Q.SYSTEM_INI

Your QMF session procedure Q.SYSTEM_INI, can be as simple as setting some QMF global variables or profile values or as complex as 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 9. The Q.SYSTEM_INI shipped with QMF

Q.SYSTEM_INI is located in the QMF product as DSQ0BINI.

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 10. Q.SYSTEM_INI example that calls a user defined procedure

```
 PROC               WILLIAMS.USER_INI                          LINE
1

 -- This QMF procedure example shows how to setup QMF session defaults for
 -- A QMF user. The following settings replace any settings set by the
 -- SYSTEM_INI proc.

--
    QMF SET GLOBAL (DSQEC_NLFCMD_LANG=1)   -- Process English Commands
    QMF RESET PROC                         --  Hide Contents of this PROC
    QMF SET PROFILE (SPACE=MYSPACE)        --  Store data in MYSPACE.
    QMF SET PROFILE (PRINTER=MYROOM)       --  Print reports at My Printer
    QMF SET GLOBAL (DSQDC_LIST_ORDER=3A)   --  Object List Sorted by Object Name
    QMF SET GLOBAL (DSQEC_RESET_RPT=2)     --  Always ResetReports
    QMF SET GLOBAL (DSQEC_SHARE = 1)       --  Always Share My QMF Objects
    QMF SET GLOBAL (DSQEC_NLFCMD_LANG=0)   -- Return to Presiding Language
```

Figure 11. User session procedure example: user.USER_INI

Procedure that Displays an Object list

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

```
PROC                 Q.SYSTEM_INI                        LINE    1


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

Figure 12. 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 distinghished from others. See “Using the QMF Trace Facility” on page 242 for more information on QMF trace options.

Chapter 9. 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.

Before you set up user access to QMF as explained in this chapter, you need to make sure the user is known to CICS. Define a 3-character CICS terminal operator ID by:

- Defining a VSE user ID and mapping it to a CICS terminal operator ID
- Redefining the CICS IDin the default sign-on table (SNT) shipped with VSE

Both methods are explained in *VSE/ESA Planning.*

If your users need to connect to DB2 explicitly, as explained in 67, grant them DB2 CONNECT authority:

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

Quick Start

Use the steps in Table 8 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 8. Establishing QMF support for end users

To do this task:	See:
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 82
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 92
Customize a user's database object list, using the DSQEC_TABS_SQL and DSQEC_COLS_SQL global variables.	Page 97
Enable users to create tables (if necessary) by assigning a private dbspace or by granting DB2 RESOURCE authority and assigning a public dbspace.	Page 100
Enable users to support a chart using the Interactive Chart Utility (ICU) of GDDM.	Page 104

Table 8. Establishing QMF support for end users (continued)

To do this task:	See:
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 104
When necessary, enlarge the dbspace for the QMF object control tables using the DB2 DBS utility UNLOAD and RELOAD commands. Recreate indexes and any views you defined on the tables.	Page 109
Maintain your users' database tables and views by updating and reorganizing DB2 system tables.	Page 110

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 84 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 275.
- 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 97.

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 84. 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 13 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 9 on page 85 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', 'STUT2BER'
'DEUTSCH', 'DEUTASTEN'
'COMMAND_SYNONYM_D', 'SCHICHT'
'CICSVSE')
```

Figure 13. 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 13 shows only a subset of all possible profile values. Use "Reading the Q.PROFILES Table" on page 84 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 14 shows SQL statements that delete this row. You can also use the Table Editor, as explained in *Using QMF*.

Base QMF (English)
German NLF

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

Figure 14. 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 13 on page 83.

Reading the Q.PROFILES Table

Table 9 on page 85 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 for 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 DSQ3nPRO phase to see the default values for each NLF. (Replace the *n* symbol with an NLID from Table 3 on page 10.)

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 9. Structure of the Q.PROFILES table

Column Name	Data Type and Length	Nulls Allowed	Function and Possible Values
CREATOR	CHAR (8)	No	**Function:** Specifies the SQL authorization ID (the user) who owns the profile. **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.
CASE	CHAR (18)	Yes	**Function:** Specifies whether terminal input is converted to uppercase. **Values:** UPPER (default), STRING, or MIXED. See *QMF Reference* for descriptions of these values. CASE might have a different default for NLF users.
DECOPT	CHAR (18)	Yes	**Function:** Specifies what separators QMF puts in numeric report columns. **Values:** PERIOD (default), COMMA, and FRENCH. See *QMF Reference* for more information. DECOPT is translated and might have a different default for NLF users.
CONFIRM	CHAR (18)	Yes	**Function:** Controls display of confirmation panels. **Values:** YES (default) if you want confirmation panels displayed before database changes; NO if you don't. See page 103 for information on confirming table changes.

Table 9. Structure of the Q.PROFILES table (continued)

Column Name	Data Type and Length	Nulls Allowed	Function and Possible Values
WIDTH	CHAR (18)	Yes	**Function:** Controls number of printed columns per page. **Values:** 22 to 999. Default = 132.
LENGTH	CHAR (18)	Yes	**Function:** Controls number of printed lines per page. **Values:** 1 to 999, or CONT if you want no page breaks. Default = 60.
LANGUAGE	CHAR (18)	Yes	**Function:** Controls which query language QMF uses after a RESET QUERY command is issued. **Values:** SQL (default), QBE (for Query-by-Example), or PROMPTED (for Prompted Query).
SPACE	CHAR (50)	Yes	**Function:** Specifies a dbspace that holds tables created using SAVE DATA and IMPORT commands. **Values:** Any valid dbspace name. See page 102 for more information on using dbspaces.
TRACE	CHAR (18)	Yes	**Function:** Controls the level of detail in trace output. **Values:** ALL traces all functions at the most detailed level. NONE (default) inhibits normal levels of tracing. A character string of function codes and numbers indicates the level of tracing for individual QMF functions. See page 242 for more information on the QMF trace facility. See page 65 to specify a trace value when QMF starts. Only the values ALL and NONE are translated in NLFs.

Table 9. Structure of the Q.PROFILES table (continued)

Column Name	Data Type and Length	Nulls Allowed	Function and Possible Values
PRINTER	CHAR (8)	Yes	**Function:** Controls where printer output is routed. **Values:** Use a null (default) or blank value to route print output to CICS temporary storage or transient data queues. Use a GDDM nickname to direct output to a GDDM-defined printer. See Chapter 10 on page 113 for information on choosing and specifying values.
TRANSLATION	CHAR (18)	No	**Function:** Indicates English or NLF environment. **Values:** English (default) or the name of an NLF. The right-hand column of Table 3 on page 10 shows the translated names you need to use in this column.
PFKEYS	VARCHAR (31)	Yes	**Function:** Indicates the table or view (if any) where user's customized function key definitions are stored. **Values:** Any valid DB2 table or view name. If blank or null (default), QMF's default keys are used. "Chapter 12. Customizing QMF Function Keys" on page 145 describes how to create this table.
SYNONYMS	VARCHAR (31)	Yes	**Function:** Indicates the table or view (if any) where user's customized command definitions are stored. **Values:** Any valid DB2 table or view name. If blank or null (default), no customized definitions are used. "Chapter 11. Customizing QMF Commands" on page 133 describes how to create this table.

Table 9. 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 14. Controlling QMF Resources Using a Governor Exit Routine" on page 199 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:** This value is CICSVSE if you access the profiles through CICS/VSE. If profiles are stored in DB2 for VSE but are being accessed from a DB2 for OS/390® or DB2 for VM application requester, the value can be any one of the following: CMS, TSO, CICSMVS, 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. If running in CICS, CREATOR=SQL ID, ENVIRONMENT=CICS
3. CREATOR=SQL ID, ENVIRONMENT=NULL
4. CREATOR=SYSTEM, ENVIRONMENT=*current operating environment*
5. If running in CICS, CREATOR=SYSTEM, ENVIRONMENT=CICS
6. CREATOR=SYSTEM, ENVIRONMENT=NULL

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

Current operating environment is CICSVSE if the profiles are stored in VSE DB2 and are being accessed through CICS/VSE.

The value for current operating environment can also be CICSMVS, CICS, or TSO if the profiles are stored in VSE DB2 but are being accessed from an OS/390 DB2 application requester. The value can be CMS if the profiles are stored in VSE DB2 but are being accessed from a VM 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.

Storing Profiles in VM DB2 in a Guest-Sharing Environment

If you store QMF VSE profiles in a VM DB2 database, add the value CICSVSE to the ENVIRONMENT column of the user's QMF VM profile to ensure that your users can access QMF.Figure 15 shows how a site using DB2 guest sharing might use QMF VSE to access profiles and other objects stored in VM DB2.

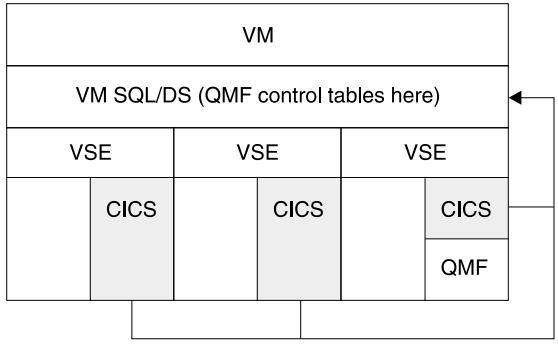


Figure 15. Possible guest-sharing scenario for profiles

Updating User Profiles

You can change the values in a user's profile by using either the SET PROFILE command or SQL UPDATE statements.

Using the SET PROFILE Command

Using this command is quicker than using SQL UPDATE statements, because you can enter it from the QMF command line with minimal typing.

Values set using SET PROFILE remain effective only until the user's session ends; use the SAVE PROFILE command to save values you changed. For more information on the SET PROFILE command and its parameters, see *QMF Reference*.

Because no special SQL privileges are required to use this command, your users can easily update their own profiles. However, they cannot use SET PROFILE to update fields you might use to customize their QMF sessions. These fields are PFKEYS, SYNONYMS, and RESOURCE_GROUP. You can use SQL UPDATE statements or the QMF Table Editor to update these Q.PROFILES fields. The Table Editor is explained in *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 9 on page 85 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 11. Customizing QMF Commands" on page 133
- "Chapter 12. Customizing QMF Function Keys" on page 145
- "Chapter 14. Controlling QMF Resources Using a Governor Exit Routine" on page 199

Use an SQL UPDATE query similar to the one in Figure 16 to update existing user profiles. This example changes the name of the table that stores a user's command synonyms. On the left is an example query for user JONES in base (English) QMF; on the right is the same query for user SCHMIDT in the German NLF.

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

Figure 16. 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 13 on page 83.

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 104.
- For instructions for database tables and views, see “Maintaining Tables and Views Using DB2 System Tables” on page 110.

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 authorities have been granted. For sample queries you can use, see “Transferring Ownership of Queries, Forms, and Procedures” on page 108.

Use a query similar to the one shown in Figure 17 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 17. 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 94. "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 95.

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 `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 10 on page 93, 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 94 for examples of SQL GRANT statements.

Table 10. 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 94 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 100.

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 11, 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 11. 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 queries or QBE queries, see *Using QMF*.

SQL Privileges Required for the Table Editor

Using Table 12, 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 12. 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. 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 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 during QMF installation.

When granting or revoking privileges on objects you do not own, qualify the object with the SQL authorization ID of the owner:

```
JONES.ORDER_BACKLOG
```

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 18 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 18. 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 19 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 19. Granting an SQL privilege to all QMF users

For more information on 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. If a user is editing a table required during QMF initialization, that table can be locked to prevent QMF from starting for other users.

Using the SQL REVOKE Statement

Use the SQL REVOKE statement to remove privileges:

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

Figure 20. Revoking an SQL privilege from a QMF user

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

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

For more information on 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 21 saves the current query as ORDER_QUERY and allows any other user to display and run it:

```
SAVE QUERY AS ORDER_QUERY (SHARE=YES
```

Figure 21. Sharing a QMF object

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

The owner of an object can change its shared status at any time, using a DISPLAY command followed by a SAVE command, as shown in Figure 22:

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

Figure 22. 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.

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 V7R1 the value '0' is only effective with DB2 for VM/VSE V5 or higher.

Setting Standards for Creating Objects

The objects in your installation might be shared among many users, so each should have a name that indicates what the object is and how it should be used. Encourage users to provide comments for queries, forms, procedures, and tables to describe for other users the purpose of the object. 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 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 two default views and automatically assigns these views to the user Q during QMF installation:

Q.DSQEC_TABS_SQL
Q.DSQEC_COLS_SQL

The view Q.DSQEC_TABS_SQL selects only the database tables the user is authorized to see. Figure 23 on page 98 shows the type of information the view provides.

```
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 TABLES/VIEWS LIST';
GRANT SELECT ON Q.DSQEC_TABS_SQL TO PUBLIC;
```

Figure 23. Default view that provides a list of tables for the LIST command

To override the default view Q.DSQEC_TABS_SQL, issue the command:

```
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 24 shows the type of information the view provides.

```
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 24. 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 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 25 on page 99 to create your own view. This query eliminates duplicate rows in the view and, although DB2 spends more time before returning rows to QMF, there is less data transfer between the database and the user machine, producing better performance.

You can name your customized view any name that is valid in QMF. See *QMF Reference* for information on QMF naming conventions.

```
CREATE VIEW Q.DATABASE_OBJECTS
  (OWNER,TNAME,TYPE,SUBTYPE,MODEL,RESTRICTED,REMARKS,
   CREATED,MODIFIED,LAST_USED,LABEL,LOCATION,OWNER_AT_LOCATION,
   NAME_AT_LOCATION)
AS SELECT CREATOR,TNAME,
'TABLE',TABLETYPE,' ',' ',REMARKS,
   ' ',' ',' ',TLABEL,
   ' ',' ',' '
FROM SYSTEM.SYSCATALOG A
   WHERE TNAME IN (SELECT TTNAME
                FROM SYSTEM.SYSTABAUTH
                 WHERE TCREATOR = A.CREATOR
                   AND GRANTEETYPE = ' &'
                   AND GRANTEE IN (USER, 'PUBLIC'))
```

Figure 25. Customizing your object lists using global variables

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

```
SET GLOBAL (DSQEC_TABS_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.

Follow these rules if you're creating a list view of your own:

- 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
- 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. If you use a new name, a new table is created. 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 13 on page 101.

Table 13. Creating tables in the database

If you're creating tables for your users:	**If users are creating tables themselves:**
Step 1:	**Step 1:**
Acquire a dbspace as shown in Figure 26 on page 102 and define it to DB2 beforeits first use. Use *DB2 Server for VSE Database Administration* to help you decide on a private or public dbspace.	Acquire a dbspace as shown in Figure 26 on page 102 and define it to DB2 before its first use. Use *DB2 Server for VSE Database Administration* to help you decide on a private or public dbspace.
Step 2:	**Step 2:**
Issue an SQL CREATE TABLE statement, a QMF DISPLAY command followed by a SAVE DATA command, *or* an IMPORT TABLE command to create the table. See *Using QMF* for examples of creating tables.	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 89. You can update the SYSTEM profile if you need to change its default values.
Step 3:	**Step 3:**
Create one or more indexes on the tables you create, to improve DB2 performance. See *DB2 Server for VSE & VM SQL Reference* for information on the CREATE INDEX statement and details on logical design of tables.	Grant DB2 RESOURCE authority to users creating their own tables in public dbspaces, or acquire a private dbspace for the user. Users automatically have all SQL privileges on tables they create.
Step 4:	**Step 4:**
Fill the tables with data. Use the DB2 DBS Utility, QMF IMPORT commands (for transferring small tables), or other methods. *DB2 Server for VSE & VM Database Services Utility* explains how to use the DBS Utility. *Using QMF* explains exporting and importing objects in QMF.	Provide education on the SQL CREATE TABLE statement, QMF SAVE DATA and IMPORT commands, and other guidelines your site has for creating tables. See *QMF Reference* for more information on these commands.
Step 5:	**Step 5:**
Grant SQL privileges for the tables to users who need them, as discussed in "SQL Privileges Required to Access Objects" on page 92.	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. SQL privileges for QMF functions and commands are discussed starting in "SQL Privileges Required to Access Objects" on page 92.

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 RESOURCE authority can create tables in a *public* dbspace. If the dbspace is *private*, only the assignee is allowed to create tables in it. For additional guidance on types of dbspaces, see *DB2 Server for VSE 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 26. You can enter this statement from the QMF SQL query panel, then press the Run function key to run the query.

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

Figure 26. Acquiring a dbspace

Substitute `PRIVATE` for `PUBLIC` in the statement if you're acquiring a private 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 59 shows an algorithm for estimating the size of a spill file.

Whatever size you choose, first search the DB2 storage pools for an existing dbspace close to the size you need. If no dbspace of convenient size already exists, use the ADD DBSPACE statement to create a dbspace. Instructions for adding dbspaces are provided in *DB2 Server for VSE System Administration*.

Granting a User DB2 RESOURCE Authority

You need to grant DB2 RESOURCE authority to any user who needs to create tables in a public dbspace. To grant a user RESOURCE authority, issue the SQL statement shown in Figure 27, where *userid1*, *userid2*, and *userid3*, represent SQL authorization IDs.
A user with RESOURCE authority can:

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

Figure 27. SQL statements to grant RESOURCE authority to more than one user

- 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.

For more information on acquiring a dbspace and a discussion of DB2 authority levels, see *DB2 Server for VSE Database Administration*.

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 choose for the user is recoverable. Because changes to DB2 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
- 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 creates charts using the Interactive Chart Utility (ICU) supplied by the GDDM-PGF product. Chart formats are templates for various types of charts (such as pie charts or histograms) that don't contain data. When a user creates a chart, QMF associates the data used with the chart format. Then, when the user enters a QMF DISPLAY CHART or EXPORT CHART command, the chart format and the data are merged to produce graphics data file (GDF) data.

User-defined chart objects are saved in the GDDM file ADMF. This file is defined during GDDM tailoring for CICS.

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"
- The Q.OBJECT_DATA table, which is described in "Reading the Q.OBJECT_DATA Table" on page 105
- The Q.OBJECT_REMARKS table, which is described in "Reading the Q.OBJECT_REMARKS Table" on page 106

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 14 on page 105:

Table 14. 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 always REL for QMF VSE 3.2, indicating relational data.
CREATED	TIMESTAMP		Yes	Shows the timestamp value for when an object was created. The value is recorded after SAVE or IMPORT commands.
MODIFIED	TIMESTAMP		Yes	Shows the timestamp value for when an object was last modified. The value is recorded after SAVE or IMPORT commands.
LAST_USED	TIMESTAMP		Yes	Shows the date value for when an object was last used. The value is updated 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_DATAX, with the UNIQUE attribute. Keyed columns are OWNER, NAME, and SEQ.

The table has the structure shown in Table 15:

Table 15. 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.
APPLDATA	LONG VARCHAR	3600	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.

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 16:

Table 16. 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 28.

```
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 28. 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:

- 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:

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

Figure 29. Sharing another user's objects with all users

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

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 30 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 30. Transferring QMF objects to another user

In the queries shown in Figure 30, `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 29 on page 107 to change the RESTRICTED column value to Y if you decide you want to share the object after transferring it.

Deleting Obsolete Queries, Forms, and Procedures

Use the SQL in Figure 31 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 31. Deleting unnecessary objects from the QMF control tables

You can also delete obsolete objects by using the date and time sorting capabilities in Q.OBJECT_DIRECTORY. You can select every object where the 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 DBS utility to enlarge the dbspace for these tables:

1. Archive the database, so that a backup copy is available for recovery if you need it.
2. Unload the dbspace using the `UNLOAD dbspace` command of the DBS utility.

   Table 17 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 275.

   All dbspaces for the QMF control tables are public. The sizes are given in pages, where each page is one 4096-byte block.

Table 17. 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_DATAX and is keyed on the OWNER, NAME, and SEQ columns.
- The index name for the Q.OBJECT_REMARKS table is OBJECT_REMARKSX and is keyed on the OWNER and NAME columns.

7. Recreate views if the dbspaces for Q.OBJECT_DIRECTORY or Q.OBJECT_REMARKS were dropped. For example:
   To provide access to this view to all QMF users, grant SELECT authority

```
CREATE VIEW Q.DSQEC_QMFOBJS
  (OWNER, TNAME, TYPE, SUBTYPE, MODEL, RESTRICTED, REMARKS, LABEL,
   LOCATION, OWNER_AT_LOCATION, NAME_AT_LOCATION)
AS SELECT
  A.OWNER, A.NAME, A.TYPE, SUBTYPE, MODEL, RESTRICTED,
    REMARKS, ' ', ' ', ' ', ' '
  FROM Q.OBJECT_DIRECTORY A, Q.OBJECT_REMARKS B
  WHERE A.OWNER = B.OWNER AND A.NAME = B.NAME
  AND (A.OWNER = USER OR RESTRICTED = 'N')
```

Figure 32. Recreating a view after dropping dbspaces

   to PUBLIC:

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

8. Alter the dbspace to allow the free space on occupied pages to be used. For example:

```
ALTER DBSPACE PUBLIC.DSQTSCT1 (PCTFREE=5)
```

For more information on enlarging dbspaces, see *DB2 Server for VSE 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 System Tables

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

Listing Tables and Views

The query in Figure 33 on page 111 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 33. Listing DB2 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. However, for more information on this task, see *DB2 Server for VSE & VM Database Services Utility*.

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 VSE Database Administration*.

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 in only 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.

Establishing QMF Support for End Users

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 need to use a non-keyboard character for your currency symbol, you can specify the currency symbol by using the HEX value in a Procedure with Logic. For example, the following PROC sets the currency symbol to HEX ’9F’:

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

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

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

The command can be used in either the command line or in a linear PROC.

Chapter 10. 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 75 for more information.

Quick Start

Use Table 18 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 234 to help you solve the problem.

Table 18. 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 114 and 123
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 to CICS temporary storage queues or transient data queues and allows printing only to devices that support the American National Standards Institute (ANSI) code of carriage control characters.	Page 115
To print using GDDM services: Define a GDDM nickname for your printer and update the GDDM defaults module ADMADFC with the nickname. Update CICS resource definition tables so CICS can link the nickname with a physical device.	Page 116

Table 18. Printing QMF objects (continued)

To do this task:	See:
To print using QMF services: Define a transient data queue or use a temporary storage queue to receive output. If you're using temporary storage queues, consider writing a program to route output from the queue to the VSE POWER LST queue. A sample program is provided with QMF, and is located in the QMF sublibrary under the name DSQCRPT2.Z.	Page 123
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 130

Printing Objects

The rules for printing QMF and database objects vary, depending on the type of object. Table 19 summarizes the requirements for each object.

Table 19. 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.	Destination associated with TONAME in the ADMMNICK specification.
Form	Yes	GDDM always gets control when the PRINT command is issued.	Destination associated with TONAME in the ADMMNICK specification.
QBE query	No	Only if the nickname is supplied on the PRINT command or in profile.	Temporary storage or transient data queue. If a nickname is supplied, output goes to the destination associated with TONAME in the ADMMNICK specification.
Procedure	No	Only if the nickname is supplied on the PRINT command or in profile.	Temporary storage or transient data queue. If a nickname is supplied, output goes to the destination associated with TONAME in the ADMMNICK specification.
Profile	No	Only if the nickname is supplied on the PRINT command or in profile.	Temporary storage or transient data queue. If a nickname is supplied, output goes to the destination associated with TONAME in the ADMMNICK specification.

Table 19. Summary of print requirements for QMF and database objects (continued)

Object type	Nickname required	GDDM gets control when...	Where output is routed
Prompted query	Yes	GDDM always gets control when the PRINT command is issued.	Destination associated with TONAME in ADMMNICK specification.
Report	No	Only if the nickname is supplied on the PRINT command or in profile.	Temporary storage or transient data queue. If a nickname is supplied, output goes to the destination associated with TONAME in the ADMMNICK specification.
SQL query	No	Only if the nickname is supplied on the PRINT command or in profile.	Temporary storage or transient data queue. If a nickname is supplied, output goes to the destination associated with TONAME in the ADMMNICK specification.
Table	No	Only if the nickname is supplied on the PRINT command or in profile.	Temporary storage or transient data queue. If a nickname is supplied, output goes to the destination associated with TONAME in the ADMMNICK specification.

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 specifying the type and name of the storage queue through which those objects are routed to the printer.

- If you need to handle routing automatically (rather than writing a program to route output), use GDDM or define transient data queues for use with QMF.

GDDM does the routing for you by using the transient data queue definitions you define to CICS. QMF takes care of the routing in the same way if you're using transient data queues to hold your output. If you print to temporary storage, you need to write a program to send the temporary storage queue to the printer or display the printed output online with the CICS-supplied transaction CEBR.

- If you need to print more than 32 767 rows of output, use GDDM or define transient data queues for use with QMF.

  Temporary storage queues cannot handle more than 32 767 rows of data.

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 books:

- *GDDM Installation and System Management for VSE* (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 116.

   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, printer, or VSE POWER queue. Nicknames can define both local and remote devices.
2. Update the GDDM defaults module, ADMADFC, with the specifications of your nickname. This is explained in "Updating the GDDM Defaults Module (ADMADFC) with the Nickname" on page 120.
3. Update CICS resource definitions with the values in the nickname specification, so that CICS can link the nickname with the physical device it manages. This is explained in "Linking the Nickname with a Physical Device" on page 121.
4. 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 130.

Choosing a GDDM Nickname for Your Printer

When a user enters a printer name on the PRINTER keyword of the QMF PRINT command, GDDM searches the defaults module, ADMADFC, 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).

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.
- **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 ADMUPRTC 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 ADMUPRTC utility, see *GDDM System Customization and Administration* for GDDM 3.1 or *GDDM Installation and System Management for VSE* for GDDM 2.3. These publications also provide more information on each type of GDDM device.

Creating the Nickname Specification

To create a nickname, first define a GDDM ADMMNICK specification. This specification indicates the device characteristics to GDDM, such as the number of lines per page the printer can handle, and how the printer is managed by CICS.

Use the format shown in Figure 34 for your ADMMNICK specification.

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

Figure 34. 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 **?** 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 VSE for GDDM 2.3
- The TONAME field points to entries in the TCT or DCT so that CICS is able to properly manage communication between GDDM and the printer. Use TONAME to point to the name of a 1-character to 4-character printer definition name with a value that depends on the type of device:
  – If the nickname defines a Family 1 or 2 printer, TONAME points to a matching entry in the CICS terminal control table (TCT), which defines the printer to CICS. In the matching TCT entry, the TRMIDNT field has the same value as TONAME.

    If you define the printer to CICS using CICS resource definition online (RDO) to update the CICS system definition (CSD) file, the TERMINAL attribute has the same value as TONAME.
  – If the nickname defines a Family 3 printer, TONAME points to a matching entry in the CICS destination control table (DCT), which defines the printer to CICS. In the matching DCT entry, the DESTID field has the same value as TONAME.

Example Nickname for a Family 1 or 2 GDDM Printer

To define the nickname GRAPHIC for a Family 1 or 2 GDDM printer, you might use an ADMMNICK specification similar to the one in Figure 35. 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 35. Using the ADMMNICK specification to define a nickname for a Family 2 printer

After you create the ADMMNICK specification, link the name with a physical device by updating the TCT, as shown in the example in Figure 37 on page 121. Make sure TONAME in the ADMMNICK specification and TRMIDNT in the TCT have matching values.

You can also use CICS RDO facilities to update the CSD online. If you define the printer this way, make sure the TERMINAL attribute in the CSD and TONAME in the ADMMNICK specification have matching values.

Example Nickname for a Family 3 GDDM Printer

To define the nickname 370PRINT for a Family 3 GDDM printer, you might use an ADMMNICK specification similar to the one in Figure 36.

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

Figure 36. Using the ADMMNICK specification to define a nickname for a Family 3 printer

After you create the ADMMNICK specification, link the name with a physical device by updating the DCT, as shown in the example in Figure 39 on page 122. Make sure TONAME in the ADMMNICK specification and DESTID in the DCT have matching values.

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 examples on defining nicknames for Family 4 devices, see:

GDDM System Customization and Administration for 3.1
GDDM Installation and System Management for VSE for 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:

  ```
  ADMMNICK NAME=GDDMPRT1,TOFAM=3,DEVTOK=S3800N6,TONAME=PRT1
  ```

- **3800, 3812, or 3820 printer, 8 lines per inch:** Use the following definition to define the nickname GDDMPRT2 for a Family 3 printer:

  ```
  ADMMNICK NAME=GDDMPRT2,TOFAM=3,DEVTOK=S3800N8,TONAME=PRT2
  ```

- **Non-3800 system printer, 132 columns, 8 lines per inch:** Use the following definition to define the nickname GDDMPRT3 for a Family 3 printer:

  ```
  ADMMNICK NAME=GDDMPRT3,TOFAM=3,DEVTOK=S1403W8,TONAME=PRT3
  ```

- **A remotely attached 3287 (suitable for printing charts):** Use the following definition to define the nickname GDDMPRT4 for a Family 2 printer:

  ```
  ADMMNICK NAME=GDDMPRT4,TOFAM=2,DEVTOK=R87S,TONAME=PRT4
  ```

- **Any destination without print control options:** Use the following definition to define the nickname GDDMPRT5 for a Family 3 printer:

  ```
  ADMMNICK NAME=GDDMPRT5,TOFAM=3,PROCOPT=((PRINTCTL,0)),TONAME=PRT5
  ```

  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.

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

  ```
  ADMMNICK NAME=PCPRINT,TOFAM=1,FAM=0,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.
- **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:

  ```
  ADMMNICK TOFAM=1,NAME=GDDMOS2P,FAM=0,TONAME=(*,ADMOS2P)
  ```

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

  To print to a PC printer connected to OS/2, GDDM-OS/2 Link must be installed on your workstation.

Updating the GDDM Defaults Module (ADMADFC) with the Nickname

The ADMMNICK nickname specifications reside in the GDDM external defaults module ADMADFC, which is supplied with the GDDM product. The ADMADFC module also contains default values for the GDDM product. It is stored as an A-type member in the GDDM sublibrary.

To update the ADMADFC module with your nickname specification:

1. Punch ADMADFC to ICCF or another editor, and edit the member to update it with the nickname specification.
2. Enter your ADMMNICK specification after the ADMMDFT statements in the module.
3. Reassemble and link-edit ADMADFC.

4. Use the CEMT transaction to load a new copy of the ADMADFC phase into CICS storage. Use a statement similar to the following example:

   ```
   CEMT S PROG(ADMADFC) NEW
   ```

For more information on the ADMADFC defaults module, see:
- *GDDM System Customization and Administration* for GDDM 3.1
- *GDDM Installation and System Management for VSE* for GDDM 2.3

Linking the Nickname with a Physical Device

After you update the ADMADFC module, you need to update the CICS resource definitions so that CICS can link the nickname with a physical device it manages.

Linking a Family 1 or 2 Nickname with a Physical Device

For a Family 1 or 2 printer, you can use macros to update CICS resource definitions in the TCT, or use CICS resource definition online (RDO) to update the CICS system definition (CSD) file.

For example, for this nickname specification:

```
ADMMNICK NAME=GRAPHIC,TOFAM=2,DEVTOK=R87S,TONAME=GRAP
```

you can update the CICS TCT using a macro similar to the example shown in Figure 37.

```
GRAP     DFHTCT TYPE=TERMINAL,
                ACCMETH=VTAM,
                TRMIDNT=GRAP,
                TRMTYPE=SCSPRT,
                    . . .
                    . . .
                    . . .
```

Figure 37. Defining to CICS a nickname for a Family 2 GDDM printer

All Family 1 and 2 devices must be described to CICS as queued.

For more information on using macros to update the TCT, see *CICS/VSE Resource Definition (Macro)*. For more information on using RDO to update the CSD file, see *CICS/VSE Resource Definition (Online)*.

Linking a Family 3 Nickname with a Physical Device

For a Family 3 printer, you need to update the DCT using macros. For example, for this nickname specification:

```
ADMMNICK NAME=370PRINT,TOFAM=3,DEVTOK=S3800N6,TONAME=S04E
```

you can update the CICS DCT using entries similar to the examples shown in Figure 38 on page 122 and Figure 39 on page 122.

Add the TYPE=SDSCI entry shown in Figure 40 on page 125 after all other TYPE=SDSCI entries in the DCT. The device address (SYS097) corresponds to the printer 04E, according to the assign statement in the startup JCL. If you use SYSLST, CICS STATS is part of your QMF report. Instead, use an alternate printer.

```
**********************************************************************
*       SYSTEM PRINTER FOR QMF OUTPUT.                               *
*       BLKSIZE:  132 + 1 FOR CTLCHR=ASA + 4 FOR RECFORM=VARUNB      *
**********************************************************************
        DFHDCT TYPE=SDSCI,                                           +
               BLKSIZE=137,                                          +
               DSCNAME=UTMS04E,                                      +
               RECFORM=VARUNB,                                       +
               DEVADDR=SYS097,                                       +
               DEVICE=1403,                                          +
               TYPEFLE=OUTPUT,                                       +
               CTLCHR=ASA
```

Figure 38. TYPE=SDSCI entry for the DCT

Add the TYPE=EXTRA entry shown in Figure 41 on page 125 after all other TYPE=EXTRA and TYPE=INDIRECT DCT entries. The TYPE=EXTRA entry corresponds to the preceding TYPE=SDSCI entry by the matching value for DSCNAME.

```
**********************************************************************
*       SYSTEM PRINTER FOR QMF OUTPUT.                               *
**********************************************************************
        DFHDCT TYPE=EXTRA,                                           +
               DESTID=S04E,                                          +
               DSCNAME=UTMS04E,RSL=1
```

Figure 39. TYPE=EXTRA entry for the DCT

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 Services to Handle Printing

To use QMF services to handle printing, specify the type of storage you want to use and provide CICS with a name for the storage.

Choosing Between Temporary Storage Queues and Transient Data Queues

CICS temporary storage queues are limited to 32 767 rows of output. They route data only to local print destinations. If you use temporary storage, you need to write a program that routes the data from the queue to the VSE POWER LST queue.

CICS transient data queues are limited only by the amount of storage you specify in the CICS DCT before CICS is started. You can define the transient data queue as an intrapartition or extrapartition data queue. You can use a transient data queue that is defined as an extrapartition data queue to print data to a file, SYSLST, or SYSPCH. A transient data queue that you define as an intrapartition data queue is limited to 32 767 rows and can be used only to print data to a file.

Using the PRINT Command to Route Output to Queues

You can specify on the QMF PRINT command both the name of the queue and the type of storage defined for that queue. For example, to print a report to a temporary storage queue named XYZ, enter this command:

```
PRINT REPORT (QUEUET=TS,QUEUEN=XYZ
```

To print from a transient data queue named XYZ, you can enter the following command. Ensure that the transient data queue is defined to CICS before its first use.

```
PRINT REPORT (QUEUET=TD,QUEUEN=XYZ
```

QUEUET and QUEUEN are abbreviations for QUEUETYPE and QUEUENAME.

QMF issues an ENQ statement on the queue name to prevent writing to the queue if another program is using it. If the name is enqueued by another application, CICS indicates to QMF that the queue is unavailable at that time. Use the SUSPEND (S) keyword to tell QMF what to do when the queue is unavailable. Use the value YES (or Y) to hold the report until the queue is available, then write to it. For example:

```
PRINT REPORT (QUEUET=TS,QUEUEN=XYZ,S=YES
```

The value NO is the default and cancels the PRINT command.

Using Global Variables to Define Queues for Printing

If you don't specify a value on the PRINT command, QMF uses values stored in the global variables DSQAP_CICS_PQNAME and DSQAP_CICS_PQTYPE.

If you're using temporary storage queues for printing, set the global variable DSQAP_CICS_PQTYPE to TS. If you're using transient data queues, set DSQAP_CICS_PQTYPE to TD. TS is the default.

Use the global variable DSQAP_CICS_PQNAME to define the name of the temporary storage or transient data queue. Names for transient data queues can be from 1 to 4 bytes. Names for temporary storage queues can be from 1 to 8 bytes. The default temporary storage queue name is DSQPnnnn, where *nnnn* is the user's 4-byte CICS terminal ID. For example, DSQPA085 is a valid name.

Printing to VSE POWER using QMF

You can print to a POWER-controlled printer using QMF. To do this, you need to write a program (see Figure 43 on page 127) that segments the POWER output queue. You also need a QMF procedure to execute the QMF PRINT command and execute a CICS transaction that will execute the program. You can then run the QMF procedure using the PRINT PF key from the QMF Report screen.

You need to customize your CICS startup JCL, the DCT, and your synonym and function key tables to print using the LST POWER queue.

(In your JCL and DCT entries, you may modify the references to POWER printer 04E, print class P, and logical unit SYS097, to values that are appropriate for your installation. Be sure that you make the modifications to all of the steps. Remember that printer 00E (usually print class A and logical unit SYSLST) is not recommended for QMF printing, because CICS statistics are routed there and would be segmented with your QMF report.)

Modifying Your CICS Startup JCL

The following POWER LST statements need to be in the CICS startup JCL. 04E is the printer address for the QMF output.

```
* $$ LST LST=04E,CLASS=P
```

Include the following assignment in the CICS startup JCL.

```
// ASSGN SYS097,04E
```

You don't need printer DLBLs/EXTENTs in the CICS startup JCL for the QMF printer destination

Modifying Your DCT

Add the TYPE=SDSCI entry shown in Figure 40 after all other TYPE=SDSCI entries in the DCT. The device address (SYS097) corresponds to the printer 04E, according to the assign statement in the startup JCL.

```
*********************************************************************
*       SYSTEM PRINTER FOR QMF OUTPUT.                              *
*       BLKSIZE:  132 + 1 FOR CTLCHR=ASA + 4 FOR RECFORM=VARUNB     *
*********************************************************************
        DFHDCT TYPE=SDSCI,                                          +
               BLKSIZE=137,                                         +
               DSCNAME=UTMS04E,                                     +
               RECFORM=VARUNB,                                      +
               DEVADDR=SYS097,                                      +
               DEVICE=1403,                                         +
               TYPEFLE=OUTPUT,                                      +
               CTLCHR=ASA
```

Figure 40. TYPE=SDSCI entry for the DCT

Add the TYPE=EXTRA entry shown in Figure 41 after all other TYPE=EXTRA and TYPE=INDIRECT DCT entries. The TYPE=EXTRA entry corresponds to the preceding TYPE=SDSCI entry by the matching value for DSCNAME.

```
*********************************************************************
*       SYSTEM PRINTER FOR QMF OUTPUT.                              *
*********************************************************************
        DFHDCT TYPE=EXTRA,                                          +
               DESTID=S04E,                                         +
               DSCNAME=UTMS04E,RSL=1
```

Figure 41. TYPE=EXTRA entry for the DCT

Modifying Your Synonym and Function Key Tables

Run the job shown in Figure 42 on page 126 to create and initialize two tables. The tables define a user-defined command synonym and a function key setting.

Ensure that column definitions that can contain nulls are not changed to NOT NULL, or QMF does not process the table entry properly.

```
* $$ JOB JNM=QMFPRINT,CLASS=0
* $$ LST CLASS=A
// JOB QMFPRINT SQL TABLES
// LIBDEF *,SEARCH=PRD2.SQL340
// EXEC ARIDBS
CONNECT Q IDENTIFIED BY ????????;      --  MODIFY PASSWORD
SET AUTOCOMMIT ON;
--
CREATE TABLE Q.UTM_SYN
  (VERB                 CHAR(18)      NOT NULL,
   OBJECT               VARCHAR(31)           ,
   SYNONYM_DEFINITION   VARCHAR(254)  NOT NULL)
   IN PUBLIC.TEMP1;
--
CREATE UNIQUE INDEX Q.UTM_SYN_IDX1 ON Q.UTM_SYN (VERB, OBJECT);
--
GRANT SELECT ON Q.UTM_SYN TO PUBLIC;
--
INSERT INTO Q.UTM_SYN
  (VERB,SYNONYM_DEFINITION)
   VALUES ('SYSPRINT','RUN PROC Q.SYSPRINT_PROC');
--
CREATE TABLE Q.UTM_PFK
  (PANEL                CHAR(18)      NOT NULL,
   ENTRY_TYPE           CHAR(1)       NOT NULL,
   NUMBER               SMALLINT      NOT NULL,
   PF_SETTING           VARCHAR(254)          )
   IN PUBLIC.TEMP1;
--
CREATE UNIQUE INDEX Q.UTM_PFK_IDX1 ON Q.UTM_PFK
   (PANEL, ENTRY_TYPE, NUMBER);
--
GRANT SELECT ON Q.UTM_PFK TO PUBLIC;
--
INSERT INTO Q.UTM_PFK
   (PANEL, ENTRY_TYPE, NUMBER, PF_SETTING)
    VALUES ('REPORT','K',2,'SYSPRINT');
--
INSERT INTO Q.UTM_PFK
   (PANEL, ENTRY_TYPE, NUMBER, PF_SETTING)
    VALUES ('REPORT','L',1,
    '1=Help   2=Sys Print   3=End   4=Dflt Print   5=Chart   6=Query');
/*
/&
* $$ EOJ
```

Figure 42. User-defined command synonym and function key setting

Sample Program to Segment POWER Output

The CICS program shown in Figure 43 on page 127 issues the POWER SEGMENT macro. You need to add a PCT entry for transaction S04E and a

PPT entry for assembler program SAMSEGMP. This is command-level assembler, so TWASIZE is zero in the PCT. This program segments the 04E POWER printer output.

```
* $$ JOB JNM=SAMSEGMP,CLASS=0,DISP=D
* $$ LST CLASS=A
// JOB SAMSEGMP ASSEMBLER CMD LEVEL CICS
// LIBDEF PHASE,CATALOG=????.??????         -- your target library
* ****************************************
* STEP 1: CICS COMMAND LEVEL TRANSLATION
* ****************************************
// DLBL IJSYSPH,'ASM.TRANSLATION',0,SD
// EXTENT SYSPCH,,1,0,??????,????         -- specify start, length
ASSGN  SYSPCH,DISK,VOL=DOSRES,SHR
// EXEC       DFHEAP1$,SIZE=512K
*ASM      XOPTS(NOEPILOG)
*********************************************************************
*                                                                   *
* PROGRAM NAME:   SAMSEGMP                                          *
* TRANSID:        S04E                                              *
*                                                                   *
* PURPOSE:        SEGMENT POWER-CONTROLLED PRINTER OUTPUT.          *
*                 TRANSID IDENTIFIES THE DCT TYPE=EXTRA ENTRY       *
*                 THAT CORRESPONDS TO A SPECIFIC SYSTEM PRINTER.    *
*                                                                   *
* NOTES:          STARTED BY QMFE, TO RUN AS ASYNC TRANS.           *
*                 THEREFORE, NO TERMINAL MESSAGES ARE DISPLAYED.    *
*********************************************************************
         EJECT
* ******************************************************************
* REGISTER USEAGE
* ******************************************************************
*
R1       EQU   1
R2       EQU   2
R11      EQU   11                     EXEC INTERFACE BLOCK
R12      EQU   12                     PROGRAM BASE
R13      EQU   13                     EIB DYNAMIC STORAGE
*
* ******************************************************************
         SPACE
         DFHEISTG                     EIB DYNAMIC STORAGE
         EJECT
         PRINT NOGEN
```

Figure 43. Program to issue the POWER segment MACRO (Part 1 of 3)

```
* ********************************************************************
* PROGRAM ENTRY
* ********************************************************************
*
SAMSEGMP DFHEIENT DATAREG=R13,CODEREG=R12,EIBREG=R11
*
         B     STRT                    ==> BRANCH AROUND
         DC    CL8'SAMSEGMP'           PROGRAM NAME.
         DC    CL8'01/20/94'           LAST MOD DATE.
STRT     DS    0H
         EXEC  CICS ENQ RESOURCE(EIBTRNID) LENGTH(4).
         MVC   USERNM(8),BLANKS           * CLEAR SAVE AREA *  9/94
         EXEC  CICS RETRIEVE INTO(USERNM) LENGTH(USERLEN)
         MVC   JOBJECL,JOBINIT
         MVC   JOBJECL+13(8),USERNM    * MOVE QMF USERID TO JOB STATEMENT
*
         CLC   EIBTRNID,=C'S04E'       SEGMENT LST=04E ?
         BE    SEG04E
         DC    H'0'                    ABEND IF NOT 04E.
*
SEG04E   DS    0H
         SEGMENT DEVADDR=SYS097,JECL=JOBJECL
         SEGMENT DEVADDR=SYS097,JECL=S04EJECL
         B     DONE
DONE     DS    0H
         EXEC  CICS DEQ RESOURCE(EIBTRNID) LENGTH(4).
         EXEC  CICS RETURN.
*
S04EJECL DC    CL71'* $$ LST LST=04E,CLASS=P'
JOBINIT  DC    CL71'* $$ JOB JNM=        '
JOBJECL  DS    CL71
USERNM   DS    CL8
BLANKS   DC    CL8'        '
USERLEN  DC    H'8'

*
         LTORG ,
*
**********************************************************************
         SPACE
         END SAMSEGMP
/*
```

Figure 43. Program to issue the POWER segment MACRO (Part 2 of 3)

```
* ********************************************************************
* STEP 2: ASSEMBLE PRINT SEGMENTING PROGRAM
* ********************************************************************
CLOSE SYSPCH,00D
// DLBL   IJSYSIN,'ASM.TRANSLATION',0,SD
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=DOSRES,SHR
// OPTION CATAL
   PHASE SAMSEGMP,*
         INCLUDE DFHEAI
         INCLUDE DFHEAI0
// EXEC ASSEMBLY
CLOSE SYSIPT,SYSRDR
/*
* ********************************************************************
* STEP 3: LINKEDIT PRINT SEGMENTING PROGRAM
* ********************************************************************
// EXEC LNKEDT
/*
/&
* $$ EOJ
```

Figure 43. Program to issue the POWER segment MACRO (Part 3 of 3)

Creating the QMF Procedures

Create two procedures to send your report to a specified printer.

The first procedure is:

```
SET GLOBAL (Q='
RUN Q.SYSPRINT_PROC2
```

Save the procedure using SHARE=YES to enable your users to access the segmenting program.

```
SAVE PROC AS Q.SYSPRINT_PROC (SHARE=YES
```

Create the following two-line QMF procedure. The first line is a QMF command; the second line causes QMF to start the transaction S04E asynchronously. After QMF routes a report to the printer destination, transaction S04E causes POWER to segment the output and print it immediately. Without segmentation, you don't get the printout until CICS is shut down (like waiting for CICS STATS).

```
PRINT REPORT (QUEUENAME=S04E QUEUETYPE=TD LENGTH=62 WIDTH=132
CICS S04E (FROM=&Q&DSQAO_CONNECT_ID&Q)
```

Save the procedure using SHARE=YES:

```
SAVE PROC AS Q.SYSPRINT_PROC2 (SHARE=YES
```

Modifying Your User's Profile

Modify your user's profile to ensure that the SYNONYM column name is the name of the created synonym table (Q.UTM_SYN) and the PFKEY column name is the name of the created function key table (Q.UTM_PFK). To update one or more profiles in the QMF control table Q.PROFILE, run an SQL command. For example, to enable JONES to use the synonym, run the following command:

```
UPDATE Q.PROFILES
  SET SYNONYM = 'Q.UTM_SYN',
      PFKEYS  = 'Q.UTM_PFK'
      WHERE CREATOR = 'JONES'
```

Using Your New Print Procedure

To produce the startup JCL and DCT changes, you need to cycle CICS. (Recycling CICS also replaces your old PCT and PPT tables with your changed tables, unless you used CEDA.) All other changes can be made before or after CICS is cycled, but we suggest that you do the table definition job and the segmentation program assembly before restarting CICS. Also, if a user is using QMF when the profile is updated, the changes won't take effect until the user exits QMF and starts a new session. You can test your changes as follows:

1. User logs on to QMF, runs a query, and the report screen is displayed.
2. User sees customized function key line, and presses PF2 (instead of PF4).
3. QMF associates PF2 with the synonym SYSPRINT.
4. Synonym SYSPRINT becomes the command RUN PROC Q.SYSPRINT_PROC.
5. Q.SYSPRINT_PROC sets a global then invokes Q.SYSPRINT_PROC2.
6. Q.SYSPRINT_PROC2 issues the print report command.
7. Q.SYSPRINT_PROC2 also starts transaction S04E to segment printer.
8. User sees message: `OK, Your procedure was run.`

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 ADMADFC defaults module, or QMF displays an error message.

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=' '
```

Chapter 11. 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 CICS commands. A synonym might simply be another word for a QMF or CICS 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 20 to create a command synonym. If you need more information on any step, see the page listed at the right.

Table 20. Creating synonyms for QMF commands

To do this task:	See:
1. **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 133
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 135
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 141
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 142

Creating the 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 the Command Synonym Table" on page 135 for instructions on entering your synonyms and their definitions.

1. If necessary, acquire or add a dbspace to hold the command synonyms table. Figure 26 on page 102 shows how to acquire a dbspace. If you need to add a dbspace, see *DB2 Server for VSE System Administration*.
2. From the QMF SQL query panel, run an SQL CREATE TABLE statement similar to the one in Figure 44 on page 135 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.

   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 on page 135 .

   ```
   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.

```
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

If you use DB2 for VSE as a server for a DB2 for OS/390 or DB2 for VM database, ensure that you use separate synonym tables for each operating environment. If the synonym table stored in DB2 for VSE has TSO, CMS, or MVS/CICS commands meant for use with DB2 for OS/390 or DB2 for VM, QMF issues warning messages for each TSO or CMS command it finds. See either *Installing and Managing QMF for MVS* or *Installing and Managing QMF for VM/ESA* for examples of command synonyms specific to the OS/390 and VM operating environments.

Entering Command Synonym Definitions into the 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*.

```
INSERT INTO COMMAND_SYNONYMS (VERB,OBJECT,SYNONYM_DEFINITION)
  VALUES('COMPUTE', 'MONTHLY_SALES', 'RUN PROC JONES.SALES_FIGURES')
```

Figure 45. Creating a command synonym definition

After it is activated according to the procedure in “Activating the Synonyms” on page 141, 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:

```
INSERT INTO COMMAND_SYNONYMS (VERB,SYNONYM_DEFINITION)
  VALUES('EXECUTE','RUN QUERY')
```

Figure 46. Creating a command synonym definition

After it is activated, the synonym EXECUTE runs the query currently in the QMF temporary storage area.

The synonyms in Figures 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.

The verb is your own word for the QMF RUN command or CICS command stored in the SYNONYM_DEFINITION column. For example, you might create the synonym COMPUTE for the QMF base verb RUN if your company has financial analysts who run only procedures that return financial results.

Rules for the VERB Column

Ensure entries in the VERB column of the synonym table:

- Are 1 to 18 characters long.
- Do not contain blanks.
- Do not include the verb *QMF* (other base QMF commands are allowed).
- Have an alphabetic or national character as the first character. (In English, national characters are #, @, and $.)

Characters after the first letter can be alphabetic, national characters, decimal digits, or the underscore. No other characters are allowed.

The following examples demonstrate these rules. QMF ignores rows that have invalid entries in the VERB column, and displays a warning message.

Valid Verbs:

Invalid Verbs:

COMPUTE

DO SALES (blanks not allowed)

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 tables.
- Must be surrounded by double quotes if the object name has blanks or other special characters. (Both QMF and the database manager remove the double quotes when the name is processed.)

The following examples show valid and invalid objects.

Valid Objects:
Invalid Objects:

PFKEYS
80CAT (first character is numeric)

MONTH_2_REPORT
ADJ%AGE (% not allowed)

"User x"."Net Sales"
JANUARY_PRODUCTIVITY (over 18 characters)

"Net Sales"
JONES GROSS (double quotes required for blanks)

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 CICS command that starts another CICS transaction.

Your synonym definition can even include both types of commands if the definition runs a QMF linear procedure.

For information about developing complex applications to run in a command synonym, see *Developing QMF Applications.*

Using a Linear Procedure in the Synonym Definition

A linear procedure is a QMF procedure that executes QMF commands sequentially. Your synonym definition can include a linear procedure that does the work of several QMF commands. For example, the procedure in Figure 47 performs the following tasks:

1. Runs the following query, called SALES_DATA, which creates a report that shows all the customers handled by sales representative number 20:

   ```
   SELECT QUANTITY, CUSTNO
   FROM Q.SALES
   WHERE SALESREPNO = 20
   ```

2. Routes the report from QMF to CICS temporary storage. In Figure 47, XYZ is the name of the temporary storage queue.
3. Runs a CICS transaction to route the report from temporary storage to a predefined print destination. In Figure 47, RPTX is the transaction name. It runs asynchronously with QMF to route output to a destination named REPORTX.

Your definition for a synonym that runs this procedure might look similar to

```
-- Procedure name: SALES_PROC
 RUN QUERY SALES_DATA
 PRINT REPORT (QUEUENAME=XYZ,QUEUETYPE=TS)
 CICS RPTX (FROM=('REPORTX, XYZ'))
```

Figure 47. Sample procedure to run using a command synonym

the one in Figure 48:

```
                                SYNONYM
VERB           OBJECT           DEFINITION
-----------    ---------------  ------------------
SHOW           SALES            RUN PROC SALES_PROC
```

Figure 48. Using a command synonym to run a linear 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 111.

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

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

Figure 49. Using variables in command synonym definitions

MYQUERY might look something like:

```
SELECT * FROM &TABLENAME
```

Ampersands are doubled in a variable name in the synonym definition because they become single ampersands when QMF executes the RUN command.

Use double ampersands in the synonym definition for all variables except the variable &ALL. &ALL is a special QMF variable that allows you to enter variable values when you enter the synonym, rather than including them in the synonym definition. When you use the variable &ALL in a synonym definition, QMF uses as variable values any information you enter to the right of the synonym. You can use the 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.

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

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

The query named STAFFQUERY might look something like the following:

```
SELECT * FROM Q.STAFF
WHERE DEPT=&DEPT and JOB=&EMPLOYEE_JOB
```

After activating the defined SHOW_INFO synonym, 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*.

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 139, QMF prompts the user for values for the &DEPT and &EMPLOYEE_JOB variables if the user enters SHOW_INFO by itself on the command line.

Keying Information into the SYNONYM_DEFINITION Column

Follow these rules when keying your synonym definitions into the synonym table:

- Add single quotes around a variable in your synonym definition.

  Single quotes around a variable eliminate the need for the user to add quotes to the command synonym when running a query. For example, &ALL has single quotes in this synonym definition:

  ```
  RUN MYQUERY (&NAMEVALUE='&ALL'
  ```

  If you search for the name O'BRIEN, you do not need to enter `'O'BRIEN'`, because QMF does this for you.

- Enter base verbs and keywords in uppercase.

  Literal information in the synonym definition is not converted to uppercase.

- Qualify all object names if their owners are different from the SQL authorization ID of the user who uses the synonym.

  QMF leaves names unqualified when searching for a synonym that contains the object name specified. For example, if your synonym definition includes a query named MY_SALES owned by user ID JONES, ensure that the object name in the synonym definition reads JONES.MY_SALES. Otherwise, JONES is the only user that can use that command synonym.

- Use only capital letters for letters that lie outside of delimited identifiers.

  If QMF converts user input (the synonym) to uppercase and the synonym definition is in lowercase, QMF 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='CICSVSE'
        AND ENVIRONMENT='CICSVSE'
```

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 13 on page 83.

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 142.

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 8 on page 81 for how to grant a user authority to connect to the database. Users who do not have DB2 CONNECT authority can end the current QMF session and start another to activate the synonyms.

Command synonyms follow the same rules for abbreviation as QMF commands. Any abbreviation must indicate a unique QMF command or command synonym. For example, the minimum valid abbreviation for the synonym EXECUTE is EXE. If you enter only EX, QMF can't distinguish the command synonym EXECUTE from the base QMF command EXPORT. See *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 VSE 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='CICSVSE'
```

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 will be 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 IDs.

```
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 on page 144 to create the view on the synonym table. The example in the figure shows a view created for a group of users that have a common user ID, DEPTD02. All users in the DEPTD02 group can use all public synonyms in the table and any synonyms assigned specifically to the group.

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

Figure 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 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 12. 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 21 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 21. 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 145
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 148
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 149
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 156

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.

Customizing QMF Function Keys

The tables in “Default Keys on Full-Screen Panels” show the default QMF function key labels and commands for both full-screen and window panels; use them to decide which function keys you want to change.

You cannot customize function keys on Table Editor panels. On other panels, you can choose QMF or installation-defined commands to associate with any function key label you modify.

Default Keys on Full-Screen Panels

Key	Executed Command
Backward	BACKWARD
Cancel	CANCEL
Change	CHANGE
Chart	DISPLAY CHART or SHOW CHART
Check	CHECK
Clear	CLEAR
Command	SHOW COMMAND
Comments	SWITCH COMMENTS
Delete	DELETE
Describe	DESCRIBE
Draw	DRAW
Edit Table	EDIT TABLE
End	END
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

Key	Executed Command
Report	DISPLAY REPORT or SHOW REPORT
Retrieve	RETRIEVE
Right	RIGHT
Run	RUN QUERY or RUN PROC
Save	SAVE PROFILE
Show	SHOW
Show Field	SHOW FIELD
Show SQL	SHOW SQL
Sort	SORT
Specify	SPECIFY
Specify View	SPECIFY VIEW

Default Keys on Window Panels

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

Key	Executed Command
Next Definition	NEXT DEFINITION
Previous Column	PREVIOUS COLUMN
Previous Definition	PREVIOUS DEFINITION
Refresh	REFRESH
Show Entity	SHOW ENTITY
Show Field	SHOW FIELD
Show View	SHOW VIEW
Sort	SORT
Specify Attributes	SPECIFY ATTRIBUTES
Specify Condition	SPECIFY CONDITION
Switch	HELP SWITCH

On the global variable list panel, RESET GLOBAL is the command executed when the Delete 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_PFKEYS and your own dbspace for DBSPACE1.

```
CREATE TABLE MY_PFKEYS
  (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

See “Choosing and Acquiring a dbspace for the User” on page 102 for information on acquiring a dbspace to hold the table. See *DB2 Server for VSE Database Administration* for information on creating a new dbspace.

2. Add comments to the SYSTEM.SYSCATALOG table using an SQL statement similar to the following:

   ```
   COMMENT ON TABLE MY_PFKEYS IS 'PF KEYS RESERVED FOR FINANCIAL ANALYSTS'
   ```

   The phrase PF KEYS RESERVED FOR FINANCIAL ANALYSTS appears in the REMARKS column of the SYSTEM.SYSCATALOG table. For more information on adding comments to the system catalog, see *DB2 Server for VSE 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 runs 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 following values:

Table 22. Values to customize your function key table

Column	Value	Information
PANEL	ID of the QMF panel you're customizing	"Full-Screen Panel Identifiers" on page 153 shows the IDs you need to use for full-screen panels. "Window Panel Identifiers" on page 153 shows the IDs you need to use for specific window panels. If you want to define the same set of keys to appear on every panel in a class of window panels, use the *class ID* shown at the bottom of the tables. For example, to customize the Specify panel of a Forms window, use the panel ID FOSPEC if you want the Specify panel to have different keys than the rest of the panels in the forms class. Otherwise, use the panel ID FOXXXX, which characterizes all panels in that class. Changes you make using a class ID apply to *all* panels customized by that class ID. Help and prompt windows don't have a set of unique IDs; they can be customized only by using class IDs.
ENTRY_TYPE	K	K indicates that this row defines the command QMF issues when the key is pressed
NUMBER	Number of the function key you're customizing	If you're changing the definition for PF5, enter a 5 in this column.
PF_SETTING	Text of the command that runs when the key is pressed	Make sure this command is appropriate for the panel on which it appears. For example, the ENLARGE command is appropriate only for the QUERY panel in a QBE query. Because QMF doesn't check if the command is appropriate for the panel until the user presses the key, test each of your new function keys before your end users need them. Enter the command in uppercase, because QMF does not convert terminal input to uppercase when it retrieves the commands associated with function keys. The command won't run if this value is lowercase and the CASE field of the user's profile has the value UPPER. Ensure that each panel you customize has a key set to END or CANCEL. Without a key defined to one of these commands, users might not be able to exit the panel.

If you'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 following values:

Table 23. 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 149.
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 149). 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 152 for an example. For window panels, you need to type only the label of the key in this column. See Figure 58 on page 152 and Figure 59 on page 153 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 153. 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 152 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 in Figure 57 on page 152) that you want to customize the command executed when a user presses PF2.

```
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 PF keys. For help, press PF1 or type HELP.
_____________________________________________________________________________
1=Help        2=Import      3=End        4=Show        5=Chart       6=Query
7=Retrieve    8=Edit Table  9=Form       10=Proc       11=Profile    12=Report
OK, cursor positioned.
COMMAND ===>
```

In the PF_SETTING column of the second query, be sure to type exactly what will appear 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 PF keys. For help, press PF1 or type HELP.
_____________________________________________________________________________
Import
7=Retrieve    8=Edit Table  9=Form       10=Proc       11=Profile    12=Report
OK, cursor positioned.
COMMAND ===>
```

Entering a Definition for a Key on a Window Panel

The SQL queries in Figure 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 on page 153 add a PF13 key to all help panels. The function key executes the CANCEL command, and is labeled CancelMe.

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

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

All help and prompt panels are customized using a single class ID. Because any changes you make to one panel in the class appear on *all* panels that are defined with that class ID, ensure changes you make to one help or prompt panel are appropriate for all the help and prompt panels in that class.

Identifying the Panel 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 four characters when it appears on the screen.

Window panels not named in the tables do not have unique panel IDs, and can be customized using the class ID shown at the bottom of each table. All class IDs have the character string XXXX in them. These characters are not variable characters; they are actually part of the ID.

Command Windows

Panel Identifier	Title or Description
COENTR	Command Entry
COXXXX	Command Window Class

Forms Windows

Panel Identifier	Title or Description
FOALIG	Alignment
FODFIN	Definition
FOSPEC	Specify
FOXXXX	Form Window Class

Global Variable Windows

Panel Identifier	Title or Description
GLADVA	Add Variables
GLSHVA	Show Variables
GLXXXX	Global Variables Window Class

Help and Prompt Windows

Panel Identifier	Title or Description
HEXXXX	Help Window Class
PRXXXX	Prompt Window Class

Location Windows

Panel Identifier	Title or Description
PLLOCA	Location Window List

Object List Windows

Panel Identifier	Title or Description
OBDESC	Object Description
OBLIAC	Object List: Action
OBLIMU	Object List: Multi-selection

Panel Identifier	Title or Description
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
QPROWS	Rows (Row Conditions)
QPSHFI	Show Field

Panel Identifier	Title or Description
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 = 'CICSVSE'
        AND ENVIRONMENT = 'CICSVSE'
```

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 142. 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 8 on page 81 for how to grant a user authority to connect to the database. Users who do not have DB2 CONNECT authority can end the current QMF session and start another to activate the new function keys.

Chapter 13. Creating Your Own Edit Codes for QMF Forms

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 with QMF do not meet the report editing needs of your site, you can use the information in this chapter to create your own edit codes to be used in the EDIT column of the FORM.MAIN and FORM.COLUMNS panels. *QMF Reference* shows the edit codes supplied with QMF.

This chapter also shows you how to write an edit exit routine in High-Level Assembler (HLASM), VS COBOL II, COBOL for VSE/ESA, or PLI for VSE/ESA 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.

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 24 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 24. 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 160
Request that your exit routine format the data by using fields of the IBM-supplied interface control block.	Page 161

Table 24. Creating a user edit exit routine (continued)

To do this task:	See:
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 164
Request that control be passed 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 167
To write your edit exit routine in HLASM, start with the sample HLASM program provided by IBM. After you write your program, translate, assemble, and link-edit the program and define it to CICS.	Page 168
To write your edit exit routine in VS COBOL II or IBM COBOL for VSE/ESA start with the sample COBOL program provided by IBM. COBOL refers to either VS COBOL II or IBM COBOL for VSE/ESA. After you write your program, translate, compile, and link-edit the program and define it to CICS.	Page 175
To write your edit exit routine in PLI for VSE/ESA start with the sample PLI program provided by IBM. After you write your program, translate, compile, and link-edit the program and define it to CICS.	Page 186

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, COBOL 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 25.

Table 25. 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 user edit routine is written to accept edit codes in mixed case, enter the edit codes when case is set to mixed.

Locally defined date and time edit codes (TTL and TDL) available in other QMF operating environments are not available in QMF VSE. If you choose to write an edit exit routine to carry out these functions that are not supplied by IBM, you cannot use TTL and TDL as the edit codes; instead, use Uxxxx or Vxxxx edit codes to identify your local date and time exit routines.

Calling Your Exit Routine to Format the Data

Figure 62 on page 162 shows how QMF and your edit exit routine work together to format data using the edit codes you define.

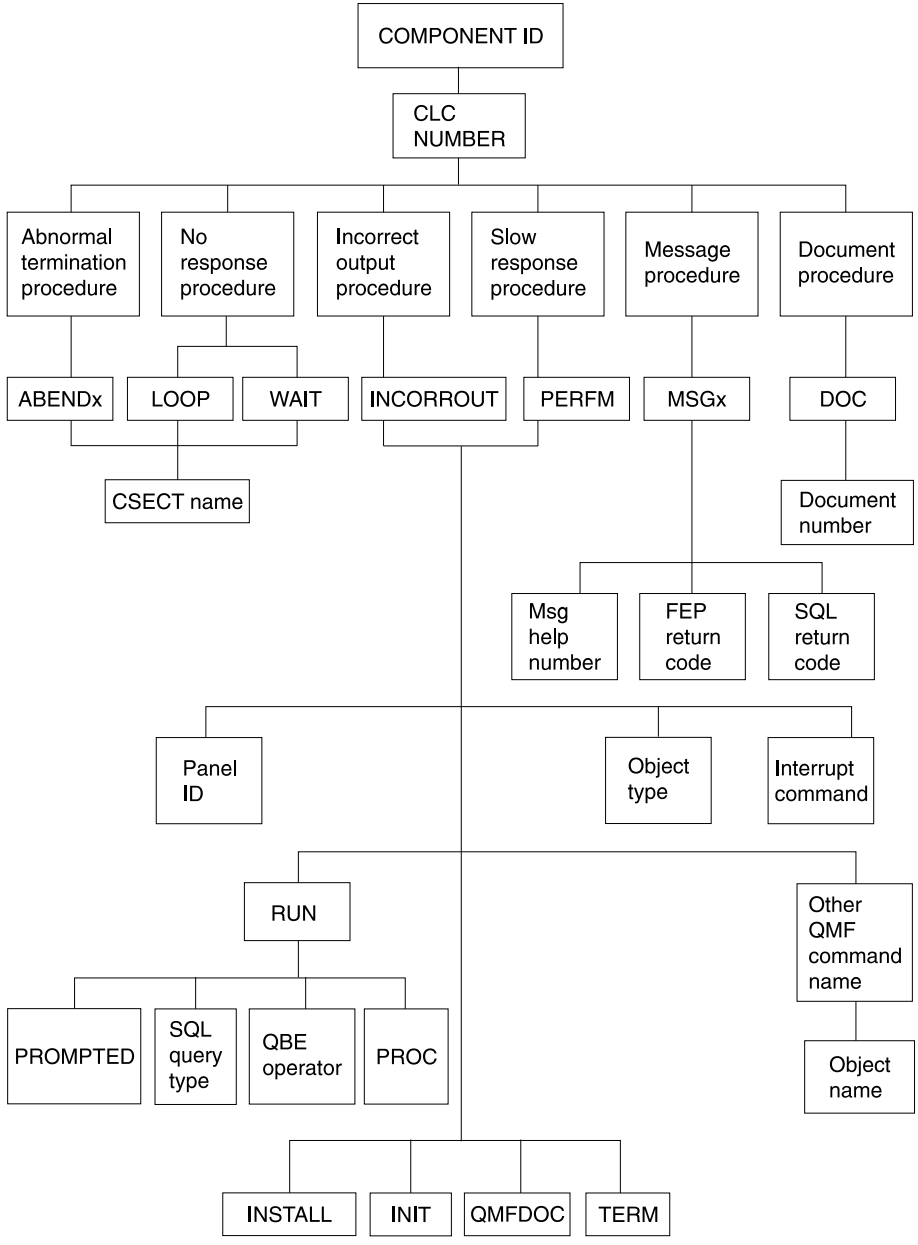


Figure 62. How a user edit routine works with QMF

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 at the left of Figure 62. These characteristics reside in specific fields of the control block, which are discussed in “Fields of

the Interface Control Block" on page 165. 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 HLASM (named DSQUXCTA), one is for PLI for VSE/ESA (named DSQUXCTP), and the other is for VS COBOL II (named DSQUXCTC). 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.

The DSQUECIC phase supplied with QMF is a sample meant to be used with the sample edit programs. Because of this, the phase simply returns an error code when it is called, and QMF displays a message indicating you attempted to use an unsupported edit code.

After you write your edit exit program, name it DSQUECIC. Then translate, assemble (or compile), and link-edit the program to form the edit exit phase named DSQUECIC. You need to replace the IBM-supplied phase with your new phase. Do not change the name of the phase; it remains DSQUECIC.

Figure 63 on page 164 shows the general program structure for edit routines.

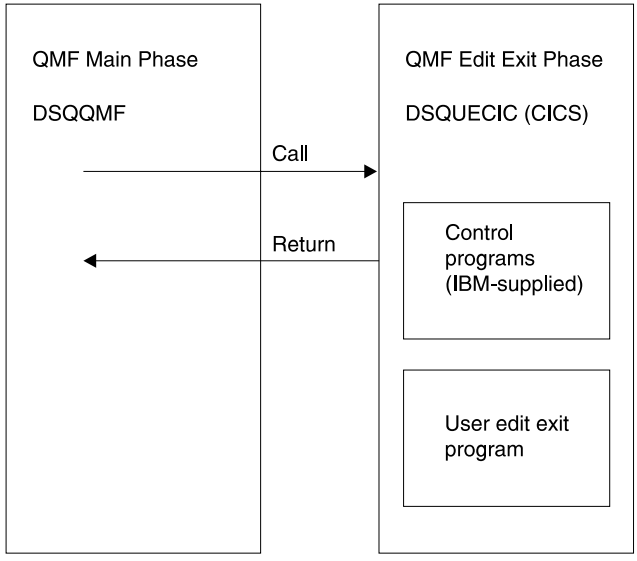


Figure 63. General program structure for edit routines

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. ECSRLEN contains the amount of storage actually passed to this output area on each call.

Important: Do not use more memory in the output area than is indicated in the ECSRLEN field, or results could be unpredictable.

ECSINPT, ECSRSLT, and ECSRLEN are fields of the interface control block, explained in Table 26.

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 interface with HLASM, COBOL, or PLI, 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 26. Unless otherwise stated, each field relates to all formatting calls.

These same fields appear in each sample program shipped with QMF. You can include these field names in your own source program. The sublibrary where QMF is installed contains the sample programs.

Table 26. Fields of the QMF interface control block

Name	Contents
ECSDECPT	Contains the current decimal point symbol as determined by the DECOPT option of PROFILE (period or comma).
ECSECODE	Contains the user edit code.
ECSERRET	Contains a zero at the point of call. Set this to a nonzero return code to record an error. Use one of the values in the following list for an error of the indicated type: **Number** — **Error** **99101** Unrecognized edit code **99102** Improper input data type for edit code **99103** Invalid input value for item to be formatted **99104** Item to be formatted is too short **99105** Not enough room for result in ECSRSLT (result is too wide for the space allotted) The error codes listed (and their associated messages and help panels) are specific to the error. For any other code, a general error message, with a general backup help panel, is displayed.
ECSFREQ	Holds E for a formatting call, T for a termination call.
ECSINLEN	Contains the length, in bytes, of the value to be formatted.
ECSINNUL	Holds an N if the value to be formatted is null. These values are not passed to a user edit routine: `overflow`, `undefined`, `no instance`, `no rel data`.
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.

Table 26. Fields of the QMF interface control block (continued)

Name	Contents
ECSINSGN	Holds the sign of a converted numeric value (blank or -). Applies only to V codes when the character string to be formatted was derived from numeric data.
ECSINTYP	Indicates, in database terms, how the value to be formatted is represented. Applies to edit codes of every type. Values can be: **384** DATE data type **388** TIME data type **392** TIMESTAMP data type **448** VARCHAR data type **452** CHAR data type **456** LONG VARCHAR data type **464** VARGRAPHIC data type **468** GRAPHIC data type **472** LONG VARGRAPHIC data type **480** FLOAT data type **484** DECIMAL data type **496** INTEGER data type **500** SMALLINT data type **940** Extended floating point data type The extended floating point data type is not supported by the database (or by VS COBOL II); it is limited to functions such as AVERAGE and STDEV. Extended floating point values are precise to more than 30 digits.
ECSNAME	Contains the name of the control block, which is DXEECS. Serves as an eye catcher in storage dumps.
ECSRQMF	Set this to T to request a termination call.
ECSRSLEN	Contains the length of the output area, in bytes.
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

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 for the decimal format are in the ECSINSCL and ECSINPRC fields. Length (in bytes) is in ECSINLEN.

Numeric data from certain summary values is returned as extended floating point values, a data type not explicitly supported by VSE DB2. The length (16 bytes) is in the ECSINLEN field.

Character or graphic values are represented in their internal, character-string format, with one exception: for variable-length strings (for example, VARCHAR data type), only the string itself appears and not the preceding length field. For all character values, the string length (in bytes) is in the ECSINLEN field.

How V-Type Edit Codes are Represented in the Input Area

Numeric values are represented by a numeric character string. The length is contained in the field ECSINLEN. Leading or trailing zeros fill out the string if required.

The string contains no sign or decimal point. Instead, the sign appears as a blank or a minus sign in the field ECSINSGN, and the position of the decimal point is in the field ECSINSCL. For example, suppose that the string in ECSINPT is 12345, that ECSINSGN is blank, and that ECSINSCL is equal to 3; then the value represented is +12.345.

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 ECSRSLEN. QMF blanks out ECSRSLT before calling the edit routine. The output area is temporary storage and can hold no more than 32 767 rows of output.

Passing Control to the Exit Routine When QMF Terminates

Use the ECSRQMF field of the control block to indicate that you want your exit routine to receive control whenever QMF terminates. The ECSRQMF value should be updated the first time the edit exit routine receives control.

When your edit exit routine receives control upon termination of QMF, the parameters passed to the routine are the control block, the input area, and the output area. Only the control block contains usable information.

Writing an Edit Routine in High-Level Assembler (HLASM)

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

- Interface control block, which is shipped with QMF as DXEECSA.A. Use a COPY statement to include the interface control block in your source deck.
- CICS prolog macro DFHEIENT, shipped with CICS. (The CICS epilog macro, DFHEIRET, is not needed for an EXEC CICS RETURN.)
- CICS command interface modules DFHEAI and DFHEAI0, supplied by IBM (shipped with CICS).
- Your edit exit program (named DSQUECIC).

Figure 64 shows the program structure of an HLASM edit exit routine for CICS.
The IBM-supplied sample edit program for HLASM, DSQUXCTA.Z, is located

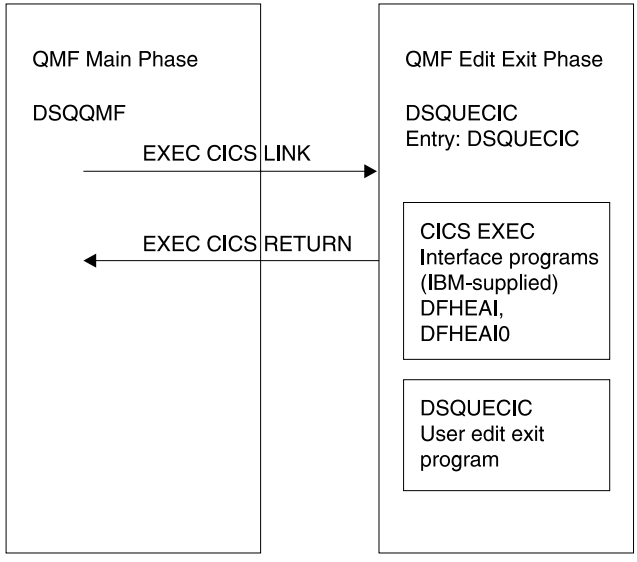


Figure 64. Program structure of an HLASM edit exit routine for CICS

in the sublibrary where QMF is installed. The sample program is commented so that you can modify it to suit your needs. If you plan to use this program, copy it to your program sublibrary and change its name to DSQUECIC.

How an HLASM Edit Routine Interacts with CICS

The user edit program is called by using the standard CICS LINK command interface. Your program executes on a different program level than the main QMF program. Upon entry to your edit exit program, the following conditions exist:

- Register 1 contains the address of a standard CICS parameter list suitable for processing by the CICS-supplied macro DFHEIENT, as shown in Figure 65.

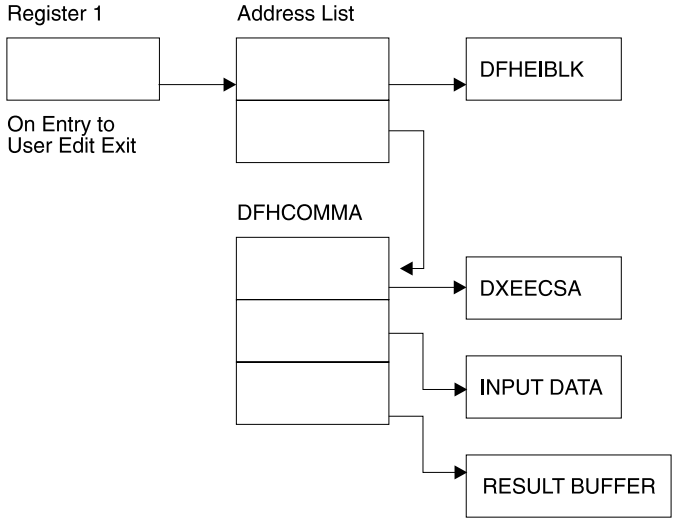


Figure 65. Registers of the program interface in HLASM

- Register 13 contains the address of a standard CICS working storage area as described by CICS-supplied macro DFHEISTG.

An HLASM DSECT for DXEECS is shipped with QMF as DXEECSA.A and is located in the sublibrary where QMF is installed. Include DXEECSA.A in your program using the HLASM COPY statement.

Return control to QMF by using the standard CICS RETURN command.

How an HLASM Edit Routine Interacts with QMF

The interface control block between QMF and the user edit interface DSQUECIC 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. 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.

Figure 66 on page 170 shows the DXEECS control block for HLASM.

```
************************************************************************ 00001000
*                                                                      * 00002000
*        CONTROL BLOCK NAME: DXEECS (ASSEMBLER 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
*        STATUS: VERSION 7 RELEASE 1 LEVEL 0                           * 00023000
*                                                                      * 00024000
*        INNER CONTROL BLOCKS: NONE                                    * 00025000
*                                                                      * 00026000
*        CHANGE ACTIVITY:                                              * 00027000
*                                                                      * 00028000
*        CHANGE DATE:                                                  * 00029000
*                                                                      * 00030000
************************************************************************ 00031000
*                                                                        00032000
DXEECS   DSECT                                                          00033000
ECSNAME  DS    CL8           -- CONTROL BLOCK IDENTIFICATION            00034000
         SPACE                                                          00035000
ECSEDCTL DS    XL40          -- EDIT CONTROL                            00036000
         ORG   ECSEDCTL                                                 00037000
ECSFREQ  DS    CL1           ----- FUNCTION REQUEST                     00038000
ECSFEDIT EQU   C'E'          -------- EDIT FUNCTION                     00039000
ECSFTERM EQU   C'T'          -------- TERMINATE FUNCTION                00040000
*                                       (TO FREE RESOURCES... QMF       00041000
*                                        WILL CALL THE USER EDIT        00042000
*                                        ROUTINE FOR THIS FUNCTION      00043000
*                                        ONLY IF THE USER EDIT ROUTINE  00044000
*                                        HAS PREVIOUSLY REQUESTED IT.   00045000
*                                        SEE ECSRQMF BELOW.)            00046000
ECSPAD10 DS    CL3           ----- RESERVED FIELD                       00047000
ECSECODE DS    CL5           ----- EDIT CODE FROM FORM OBJECT           00048000
```

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

```
ECSPAD20 DS    CL3          ----- RESERVED FIELD                        00049000
ECSDECPT 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
ECSFLTX  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
ECSPAD40 DS    CL10         ----- RESERVED FIELD                        00097000
         SPACE                                                          00098000
ECSRSDTA DS    XL16         -- DESCRIPTION OF THE RESULT BUFFER        00099000
```

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

```
         ORG   ECSRSDTA                                              00100000
ECSRSLEN 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 HLASM version of DXEECS control block (Part 3 of 3)

&rbl;

Translating Your Program

Translate your program using the CICS translator for HLASM. When you translate your program, CICS normally supplies the standard CICS prolog DFHEIENT, which sets up addressability and saves registers in the standard CICS working storage area.

Return control to QMF by using the CICS return command EXEC CICS RETURN.

Assembling Your Program

When you assemble your program, ensure the LIBDEF search chain includes the CICS and QMF sublibraries so that the CICS macros and the edit exit interface control block (DXEECSA.A) can be found.

Use the following HLASM compiler options to assemble the routine:

```
'LIBMAC,USING(NOLIMIT,NOWARN),EXIT(LIBEXIT(EDECKXIT(ORDER=EA)))'
```

These compiler options require that you specify an E-deck exit. EDECKXIT is a library exit for HLASM that enables the processing of E-decks. This exit is required here to process CICS E-decks.

VSE/ESA provides a skeleton to help you set up the E-deck exit. You can use the skeleton without modifying it; however, before you use the skeleton, ensure you enable the exit according to instructions provided in *VSE Guide to System Functions*.

Link-Editing Your Program

Create a new QMF edit exit phase DSQUECIC by including your edit program DSQUECIC with EXEC CICS interface control modules DFHEAI and DFHEAI0, both located in the CICS PRD1.BASE sublibrary. Ensure the EXEC CICS modules DFHEAI and DFHEAI0 are the first modules in the edit exit phase DSQUECIC.

The phase DSQUECIC must be executable in 31-bit addressing mode.

Example JCL Statements

Figure 67 on page 174 shows the sample job DSQ3XCTA.Z, which is shipped with QMF. This job translates, compiles, and link-edits the example HLASM program (DSQUXCTA.Z), which is also shipped with QMF. Use the sample job as a starting point to create JCL that translates, assembles, and link-edits your own edit exit routine.

For more information on installing an assembler program in CICS, see *CICS System Definition Guide*.

```
// JOB DSQ3XCTA Install  QMF Edit Exit for COBOL
*  ---------------------------------------------------------------
*  Install QMF Edit Exit (HLASM)
*  ---------------------------------------------------------------------
// SETPARM VOLID=volid      *-- update volid for syspch
// SETPARM START=rtrk       *-- update start track/block (syspch)
// SETPARM SIZE=ntrks       *-- update number of tracks/blocks (syspch)
*  ---------------------------------------------------------------------
// DLBL   IJSYSPH,'ASM.TRANSLATION',0
// EXTENT SYSPCH,,1,0,&START.,&SIZE.
ASSGN SYSPCH,DISK,VOL=&VOLID.,SHR
*  Library search chain must contain the QMF, CICS and HLASM sublibries
// LIBDEF *,SEARCH=(PRD2.PROD,PRD1.BASE,PRD2.CONFIG)
// LIBDEF PHASE,CATALOG=PRD2.PROD
*  ---------------------------------------------------------------------
*  Step 1: Translate Edit Exit program
*  ---------------------------------------------------------------------
// EXEC   DFHEAP1$
   :
   Assembler source program here
   :
/*
*  ---------------------------------------------------------------------
*  Step 2: Assemble Edit Exit program
*  ---------------------------------------------------------------------
CLOSE SYSPCH,00D
// DLBL   IJSYSIN,'ASM.TRANSLATION',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=&VOLID.,SHR
// OPTION CATAL,DECK,SYM,ERRS

   PHASE DSQUECIC,*,SVA

         INCLUDE DFHEAI
         INCLUDE DFHEAI0
```

Figure 67. Example JCL for translating, assembling, and link-editing an HLASM routine (Part 1 of 2)

```
// EXEC ASMA90,SIZE=(ASMA90,50K),                                      C
               PARM='LIBMAC,USING(NOLIMIT,NOWARN),EXIT(LIBEXIT(EDECKXITC
               (ORDER=EA)))'
CLOSE SYSIPT,SYSRDR
/*
*  ---------------------------------------------------------------------
*  Step 3: Link-edit Edit Exit program
*  ---------------------------------------------------------------------
// EXEC LNKEDT,PARM='AMODE=31,RMODE=ANY'
/*
/&
// JOB  RESET
ASSGN SYSIPT,SYSRDR    IF 1A93D, CLOSE SYSIPT,SYSRDR
ASSGN SYSPCH,00D       IF 1A93D, CLOSE SYSPCH,00D
/&
```

Figure 67. Example JCL for translating, assembling, and link-editing an HLASM routine (Part 2 of 2)

Defining the Edit Exit Phase to CICS

The edit exit phase DSQUECIC is defined to CICS during the default QMF installation. The phase is described in the CICS program processing table (PPT) or the CICS system definition (CSD) file using the ASSEMBLER keyword for the LANGUAGE parameter.

Writing an Edit Routine in VS COBOL II or COBOL for VSE/ESA

The edit exit interface for COBOL consists of three parts:

- Interface control block (supplied by IBM; shipped with QMF as DXEECSC.C)
- CICS command interface module (supplied by IBM; shipped with CICS as DFHECI)
- Your edit exit program (named DSQUECIC)

Figure 68 on page 176 shows the structure of a COBOL edit exit routine.

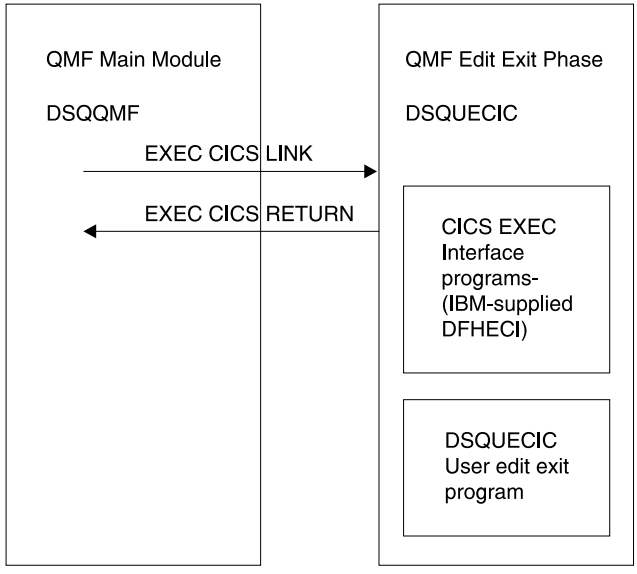


Figure 68. Program structure for a COBOL edit exit routine in CICS

The IBM-supplied sample edit exit program, DSQUXCTC.Z, is commented so that you can browse it online, print it, or modify it to suit your needs. A sample job named DSQ3XCTC.Z is shipped with QMF; this job compiles and link-edits the sample COBOL program DSQUXCTC.Z.

Using Literal Values in a COBOL Program

Use either single or double quotes to delimit literal values in a COBOL program. You can specify the delimiter of your choice in the CICS translation process, and to the COBOL compiler by specifying QUOTE or APOST keywords. Make sure the APOST or QUOTE option in effect for the COBOL compiler matches that of the CICS translator.

The edit control block DXEECSC.C and the sample COBOL program DSQUXCTC.Z, as distributed by QMF, use double quotes to delimit literals. If your installation or program uses single quotes instead, change DXEECSC.C (as distributed by QMF) or copy the structure to your program and change the double quotes to single quotes.

How a COBOL Edit Routine Interacts with CICS

The user edit program is called by using the standard CICS LINK command interface. Your program runs on a different program level than the main QMF program. Translate your edit exit program using the CICS translator for COBOL.

The CICS communications area DFHCOMMAREA is used to provide pointers to the user edit routine program parameters, DXEECS, input data, and output data as shown in Figure 69.

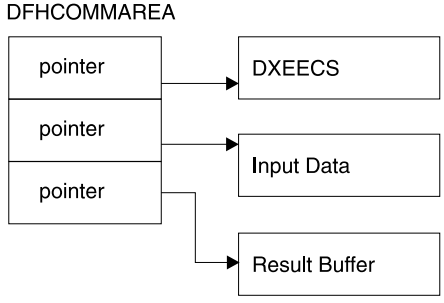


Figure 69. The CICS communications area, DFHCOMMAREA

After translation, the CICS translator provides a procedure statement that describes the CICS environment block DFHEIBLK and the CICS communications block DFHCOMMAREA, as shown in the following example:

```
PROCEDURE DIVISION USING DFHEIBLK DFHCOMMAREA.
```

QMF provides pointers to the user edit routine control block DXEECS, input data, and output data in the CICS communications area DFHCOMMAREA. Provide your own description of the DFHCOMMAREA in the COBOL program linkage section as shown in Figure 70.

```
LINKAGE SECTION.

 01  DFHCOMMAREA.
     02 ECSADR   POINTER.
     02 ECSINADR POINTER.
     02 ECSRLADR POINTER.
```

Figure 70. Example description of the CICS communications area in program linkage

To provide addressability to the user edit routine control block DXEECS, input data area ECSINPT, and the result data area ECSRSLT, set the pointers in these data areas to the values located in the DFHCOMMAREA, as shown in Figure 71 on page 178.

```
SETUP SECTION.

    SET ADDRESS OF DXEECS   TO ECSADR.
    SET ADDRESS OF ECSINPT  TO ECSINADR.
    SET ADDRESS OF ECSRSLT  TO ECSRLADR.
```

Figure 71. Establishing addressability to the control block and the input and output fields

A COBOL copybook is shipped with QMF as DXEECSC.C, located in the sublibrary where QMF is installed. Include this copybook in your program.

Return control to QMF using a standard CICS RETURN command as shown in Figure 72.

```
EXEC CICS

     RETURN

END-EXEC.
```

Figure 72. Returning control to QMF from a COBOL edit exit routine

How a COBOL Edit Routine Interacts with QMF

DXEECS is the name of the interface control block between QMF and the user edit exit DSQUECIC. 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.

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.

Figure 73 on page 179 shows the DXEECS control block for COBOL.

```
*****************************************************************  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
```

Figure 73. User edit routine field definitions for COBOL version of DXEECS control block (Part 1 of 5)

```
     02 ECSEDCTL.                                                  00041000
*                    -- EDIT CONTROL                              00042000
                                                                  00043000
        03 ECSFREQ     PICTURE X(1).                              00044000
*                    -- FUNCTION REQUEST                          00045000
           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-INTEGER            VALUE IS +496.               00080000
           88 ECS-SMALL-INTEGER      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
```

Figure 73. User edit routine field definitions for COBOL version of DXEECS control block (Part 2 of 5)

```
           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
*                     -- 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
                                                                00111000
        03 ECSINNUL    PICTURE X(1).                             00112000
*                     -- NULL INPUT DATA INDICATOR              00113000
           88 ECS-NULL-DATA VALUE "N".                          00114000
                                                                00115000
        03 ECSPAD40    PICTURE X(10).                            00116000
*                     -- RESERVED FIELD                         00117000
                                                                00118000
     02 ECSRSDTA.                                               00119000
*                     -- DESCRIPTION OF THE RESULT BUFFER       00120000
                                                                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
                                                                00128000
     02 ECSUCTL.                                                00129000
*                     -- USER CONTROL AREA                      00130000
                                                                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
                                                                00140000
     02 ECSUSERS.                                               00141000
```


Figure 73. User edit routine field definitions for COBOL version of DXEECS control block (Part 3 of 5)

```
*                    -- USER SCRATCH PAD AREA                  00142000
                                                               00143000
        03 ECSUSERS-ARRAY                                      00144000
                     PICTURE X(1)                              00145000
                     OCCURS 256 TIMES.                         00146000
                                                               00147000
                                                               00148000
********             -- EDIT ROUTINE INPUT DATA                00149000
 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-INTEGER-OVL                                    00155000
                      REDEFINES ECSINPTC.                      00156000
        03 ECSINPT-INTEGER                                     00157000
                      PICTURE S9(9)                            00158000
                      USAGE IS COMPUTATIONAL.                  00159000
        03 FILLER      PICTURE X(1)                            00160000
                      OCCURS 32763 TIMES.                      00161000
     02 ECSINPT-SMALL-INTEGER-OVL                              00162000
                      REDEFINES ECSINPTC.                      00163000
        03 ECSINPT-SMALL-INTEGER                               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 73. User edit routine field definitions for COBOL version of DXEECS control block (Part 4 of 5)

```
                                                                        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 ECSRSLEN.                  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 73. User edit routine field definitions for COBOL version of DXEECS control block (Part 5 of 5)

Translating Your Program

Before you translate your program, include in the LIBDEF statement the QMF edit exit interface control block DXEECSC.C, which is located in the sublibrary where QMF is installed.

Translate your program using the CICS translator for COBOL. When you translate your program, CICS normally supplies the standard procedure and linkage sections. If you do not specify a definition for DFHCOMMAREA in

your program, the translator automatically creates a definition for you. Because this default DFHCOMMAREA is not structured to allow proper communication between QMF and the edit exit routine, you need to prevent the translator from creating a default definition by providing a structure as shown in Figure 70 on page 177.

Compiling Your Program

Specify compiler options RENT, RES, NODYNAM for the COBOL compiler.

Because QMF distributes the user edit routine control block DXEECSC.C using quotes as literal delimiters, use the QUOTE compiler option if you use the DXEECSC control block distributed by IBM.

Link-Editing Your Program

Create a new QMF edit exit phase, DSQUECIC, by including your edit exit program (also named DSQUECIC) with the EXEC CICS interface control module DFHECI. DFHECI is located in the CICS module sublibrary, which is usually PRD1.BASE. Ensure DFHECI is the first module in the edit exit phase. Also ensure the phase DSQUECIC is executable in 31-bit addressing mode.

Example JCL Statements

Figure 74 on page 185 shows the sample job DSQ3XCTC.Z, which is shipped with QMF. This job translates, compiles, and link-edits the example COBOL program (DSQUXCTC.Z), which is also shipped with QMF. Use the sample job as a starting point to create JCL that translates, assembles, and link-edits your own edit exit routine.

Ignore weak external references unresolved by the linkage editor, and also the associated messages about unresolved address constants.

For more information on installing a program in CICS, see *CICS System Definition Guide*.

```
// JOB DSQ3XCTC  Install QMF Edit Exit for COBOL
*  ------------------------------------------------------------------
*  Install QMF edit exit (COBOL Version)
*  ------------------------------------------------------------------
// SETPARM VOLID=volid         *-- update volid for syspch
// SETPARM START=rtrk          *-- update start track/block
// SETPARM SIZE=ntrks          *-- update number of tracks/blocks
*  ------------------------------------------------------------------
// DLBL   IJSYSPH,'CICS.TRANSLAT.OUTPUT',0
// EXTENT SYSPCH,,1,0,&START.,&SIZE.
ASSGN SYSPCH,DISK,VOL=&VOLID.,SHR
*  Library search chain must contain the QMF, CICS and COBOL sublibraries
// LIBDEF *,SEARCH=(PRD2.PROD,PRD1.BASE,PRD2.CONFIG)
// LIBDEF PHASE,CATALOG=PRD2.PROD
*  ------------------------------------------------------------------
*  Step 1: Translate user edit exit program
*  ------------------------------------------------------------------
// EXEC   DFHECP1$,SIZE=256K,PARM='XOPTS(CICS,QUOTE)'
  :
 COBOL source program here
  :
 /*
 *  ------------------------------------------------------------------
 *  Step 2: Compile  translated user edit exit program
 *  ------------------------------------------------------------------
 CLOSE SYSPCH,00D
 // DLBL   IJSYSIN,'CICS.TRANSLAT.OUTPUT',0
 // EXTENT SYSIPT
 ASSGN SYSIPT,DISK,VOL=&VOLID.,SHR
// OPTION CATAL
   PHASE DSQUECIC,*,SVA
         INCLUDE DFHECI
// EXEC IGYCRCTL,PARM='SZ(MAX),OBJECT,MAP,RES,NODYNAM,QUOTE,LIB,RENT
CLOSE SYSIPT,SYSRDR
/*
```

Figure 74. Example JCL for translating, compiling, and link-editing a COBOL routine (Part 1 of 2)

```
*  ------------------------------------------------------------------
*  Step 3: Link-edit user edit exit program
*  ------------------------------------------------------------------
// EXEC LNKEDT,PARM='AMODE=31,RMODE=31'
/*
/&
// JOB  RESET
ASSGN SYSIPT,SYSRDR    IF 1A93D, CLOSE SYSIPT,SYSRDR
ASSGN SYSPCH,00D       IF 1A93D, CLOSE SYSPCH,00D
/&
```

Figure 74. Example JCL for translating, compiling, and link-editing a COBOL routine (Part 2 of 2)

Defining the Edit Exit Phase to CICS

During QMF installation, the QMF edit exit program is installed with a programming language of HLASM. To use the COBOL edit exit program, you must define the routine to CICS using the COBOL keyword in the CICS program processing table (PPT) or the CICS system definition (CSD) file.

Writing an Edit Routine in PL/I

You can write an edit routine in PL/I for CICS.

Writing an Edit Routine in PL/I for CICS

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

- Interface control block, which is shipped with QMF as DXEECSP.C
- CICS command interface modules, which are shipped with CICS as DFHPL1I
- Your edit exit program, which is named DSQUECIC

Figure 75 shows the program structure of a PL/I edit exit routine in CICS.

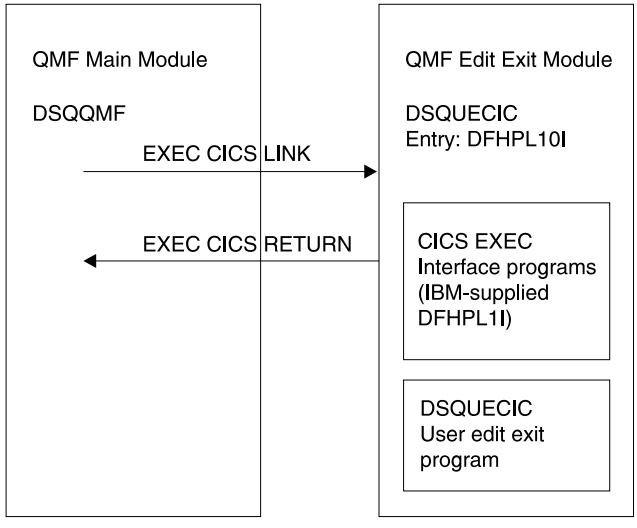


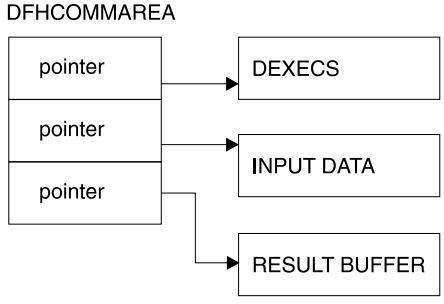
Figure 75. Program structure for PL/I edit exit routine in CICS

The IBM-supplied sample edit exit program, DSQUXCTP.Z, is commented so that you can browse it online, print it, or modify it to suit your needs. A sample job named DSQ3XCTP.Z is shipped with QMF; this job compiles and link-edits the sample PL/I program DSQUXCTP.Z.

How a PL/I Edit Routine Interacts with CICS

The user edit program is called by using the standard CICS LINK command interface. Your program is executing on a different program level than the main QMF program. The user edit program must be translated using the CICS translator for PL/I.

The CICS communications area DFHCOMMAREA is used to provide addresses to the user edit routine program parameters, DXEECS, input data, and output data as shown in the following diagram.



After translation, the CICS translator provides a procedure statement that describes the CICS environment block DFHEIBLK. Provide a parameter that is a pointer to the CICS communications block DFHCOMMAREA such as the following example:

```
DSQUXDT:
  PROCEDURE(DFHCOMMP) OPTIONS(REENTRANT,MAIN);
```

QMF provides addresses to the user edit routine control block DXEECS, input data, and output data in the CICS communications area DFHCOMMAREA. Provide your own description of the DFHCOMMAREA in the PL/I program as follows:

```
/*****************************************************************/
/*  CICS DFHCOMM ARE DESCRIPTION OF EDIT EXIT PARAMETERS         */
/*****************************************************************/
DECLARE
  DFHCOMMP PTR;
DECLARE
  1  DFHCOMM BASED(DFHCOMMP),
     02 DFHCOMM_ECSPTR PTR,
     02 DFHCOMM_INPTR  PTR,
     02 DFHCOMM_OUTPTR PTR;
```

If you do not specify a definition for DFHCOMMAREA in your program, the translator automatically creates a definition for you. Because this default DFHCOMMAREA is not structured to allow proper communication between

QMF and the edit exit routine, you need to prevent the translator from creating a default definition by providing a structure as shown in the preceding example.

To provide addressability to the user edit routine control block DXEECS, input data area ECSINPT, and the result data area ECSRSLT, set the addresses of these data areas to the values located in DFHCOMMAREA as in the following example:

```
ECSPTR   = DFHCOMM_ECSPTR;    /* ADDRESS OF DXEECS:
                                 EDIT CODE SPECIFICATIONS       */
ECSINPTP = DFHCOMM_INPTR;     /* ADDRESS OF INPUT DATA          */
ECSRSLTP = DFHCOMM_OUTPTR;    /* ADDRESS OF RESULT AREA         */
```

A PL/I data structure is shipped with QMF as DXEECSP.C, located in the sublibrary where QMF is installed. Include this structure in your program.

Return control to QMF using a standard CICS RETURN command such as the following:

```
EXEC CICS RETURN;
```

Translating Your Program

Before you translate your program, include in the LIBDEF statement the QMF edit exit interface control block DXEECSP.C, which is located in the sublibrary where QMF is installed.

Translate your program using the CICS translator for PL/I. During translation, CICS normally supplies an input parameter and data structure definition for the CICS environment control block EIB.

Link-Editing Your Program

Create a new QMF edit exit phase, DSQUECIC, by including your edit exit program (also named DSQUECIC) with the EXEC CICS interface control module DFHPL1I. DFHPL1I is located in the CICS module sublibrary, which is usually PRD1.BASE. Ensure DFHPL1I is the first module in the edit exit phase. Also ensure the phase DSQUECIC is executable in 31-bit addressing mode.

CICS Program Definition

When QMF is installed, the QMF edit exit program is installed with a program language of assembler. To use the PL/I edit exit program, you must change the program language of module DSQUECIC to PL/I using the CICS program control table (PCT) macro or resource definition online (RDO).

Example JCL Statements

Figure 76 on page 190 shows the sample job DSQ3XCTP.Z, which is shipped with QMF. This job translates, compiles, and link-edits the example PL/I

program (DSQUXCTP.Z), which is also shipped with QMF. Use the sample job as a starting point to create JCL that translates, assembles, and link-edits your own edit exit routine.

Ignore weak external references unresolved by the linkage editor, and also the associated messages about unresolved address constants. For more information on installing a program in CICS, see the *CICS System Definition Guide*.

```
..* $$ JOB JNM=DSQ3XCTP,DISP=D,CLASS=0
// JOB DSQ3XCTP  Sample Job to Install QMF Edit Exit for PL/I
*  ---------------------------------------------------------------------
*  Install QMF edit exit (PL/I)
*  ---------------------------------------------------------------------
// SETPARM VOLID=volid         *-- update volid for syspch
// SETPARM START=rtrk          *-- update start track/block
// SETPARM SIZE=ntrks          *-- update number of tracks/blocks
*  ---------------------------------------------------------------------
// DLBL   IJSYSPH,'CICS.TRANSLAT.OUTPUT',0
// EXTENT SYSPCH,,1,0,&START,&SIZE
ASSGN SYSPCH,DISK,VOL=&VOLID,SHR
*  Library search chain must contain the QMF, CICS and PL/I sublibrary
// LIBDEF *,SEARCH=(PRD2.PROD,PRD1.BASE,PRD2.CONFIG)
// LIBDEF PHASE,CATALOG=PRD2.PROD
*  ---------------------------------------------------------------------
*  Step 1: Translate user edit exit program
*  ---------------------------------------------------------------------
// EXEC   DFHECP1$,SIZE=256K,PARM='XOPTS(CICS,QUOTE)'
..* $$ SLI MEM=DSQUXCTP.Z,S=PRD2.PROD
/*
*  ---------------------------------------------------------------------
*  Step 2: Compile  translated user edit exit program
*  ---------------------------------------------------------------------
CLOSE SYSPCH,00D
// DLBL   IJSYSIN,'CICS.TRANSLAT.OUTPUT',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=&VOLID,SHR
// OPTION CATAL
   PHASE DSQUECIC,*,SVA
         INCLUDE DFHPL1I
// EXEC PLIOPT
CLOSE SYSIPT,SYSRDR
/*
*  ---------------------------------------------------------------------
*  Step 3: Link-edit user edit exit program
*  ---------------------------------------------------------------------
// EXEC LNKEDT,PARM='AMODE=31,RMODE=31'
/*
/&
// JOB  RESET
ASSGN SYSIPT,SYSRDR    IF 1A93D, CLOSE SYSIPT,SYSRDR
ASSGN SYSPCH,00D       IF 1A93D, CLOSE SYSPCH,00D
/&
..* $$ EOJ
```

Figure 76. Sample job control from DSQ3XCTP

How a PL/I Edit Routine Interacts with QMF

The interface control block between QMF and the user edit interface DSQUECIC is DXEECS. It contains the user's edit code, identifies the source data and the target location for the edited result, and provides a scratchpad area for the user edit routine's use. The control block is persistent between

calls to the user edit routine. The scratchpad area is not modified by QMF after the initial invocation of the exit routine.

Figure 77 on page 192 shows the DXEECS control block for PL/I.

```
/*****************************************************************/ 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
/*      STATUS: VERSION 7 RELEASE 1 LEVEL 0                      */ 00023000
/*                                                               */ 00024000
/*      INNER CONTROL BLOCKS: NONE                               */ 00025000
/*                                                               */ 00026000
/*      CHANGE ACTIVITY:                                         */ 00027000
/*                                                               */ 00028000
/*      CHANGE DATE:                                             */ 00029000
/*                                                               */ 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 ECSDECPT 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
                                       IN CURRENT PROFILE OBJECT)  */ 00048000
      5 ECSPAD30 CHARACTER(6),     /* RESERVED FIELD              */ 00049000
```

Figure 77. User edit routine field definitions for PL/I DXEECS control block (Part 1 of 4)

```
   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 ECSINNUL 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 77. User edit routine field definitions for PL/I DXEECS control block (Part 2 of 4)

```
                                                                    00093000
DECLARE                          /* INPUT DATA PARAMETER...      */ 00094000
  ECSINPT  CHARACTER(32767)      /* CHARACTER INPUT DATA         */ 00095000
           BASED(ECSINPTP),                                         00096000
  ECSINSIN FIXED BINARY(15)      /* SMALL INTEGER INPUT DATA     */ 00097000
           BASED(ECSINPTP),                                         00098000
  ECSININT FIXED BINARY(31)      /* INTEGER INPUT DATA           */ 00099000
           BASED(ECSINPTP),                                         00100000
  ECSINFLT FLOAT BINARY(53)      /* FLOATING POINT INPUT DATA    */ 00101000
           BASED(ECSINPTP);                                         00102000
                                                                    00103000
DECLARE                          /* RESULT BUFFER PARAMETER...   */ 00104000
  ECSRSLT CHARACTER(32767)       /* EDIT ROUTINE RESULT BUFFER   */ 00105000
          BASED(ECSRSLTP);                                          00106000
                                                                    00107000
DECLARE                                                             00108000
 (ECSPTR,                        /* MUST CONTAIN DXEECS ADDRESS  */ 00109000
  ECSINPTP,                      /* 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
  ECSFLTX   INITIAL(940)         /*   EXTENDED FLOATING POINT    */ 00130000
       )  FIXED BINARY(31) STATIC;                                  00131000
                                                                    00132000
                                                                    00133000
DECLARE (                        /* FUNCTION REQUEST CONSTANTS   */ 00134000
```

Figure 77. User edit routine field definitions for PL/I DXEECS control block (Part 3 of 4)

```
                                    (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
DECLARE (                          /* PLUS/MINUS SIGN CONSTANTS    00148000
                                      (SEE ECSINSGN ABOVE)        */ 00149000
  ECSPLUS   INITIAL(' '),          /*   INPUT DATA IS POSITIVE    */ 00150000
  ECSMINUS  INITIAL('-')           /*   INPUT DATA IS NEGATIVE    */ 00151000
        ) CHARACTER(1) STATIC;                                    00152000
                                                                  00153000
                                                                  00154000
DECLARE (                          /* NULL INDICATION CONSTANT     00155000
                                      (SEE ECSINNUL ABOVE)        */ 00156000
  ECSNULL   INITIAL('N')           /*   INPUT DATA IS NULL        */ 00157000
        ) CHARACTER(1) STATIC;                                    00158000
                                                                  00159000
                                                                  00160000
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
DECLARE (                          /* QMF-DEFINED ERROR RETURN CODE 00169000
                                      CONSTANTS                   00170000
                                      (SEE ECSERRET ABOVE)        */ 00171000
  ECSERR01  INITIAL(99101),        /* UNRECOGNIZED EDIT CODE     */ 00172000
  ECSERR02  INITIAL(99102),        /* IMPROPER INPUT DATA TYPE FOR */ 00173000
                                   /* REQUESTED EDIT EDIT CODE   */ 00174000
  ECSERR03  INITIAL(99103),        /* INVALID INPUT DATA VALUE   */ 00175000
                                   /* RECEIVED                   */ 00176000
  ECSERR04  INITIAL(99104),        /* LENGTH OF INPUT DATA IS TOO */ 00177000
                                   /* SHORT                      */ 00178000
  ECSERR05  INITIAL(99105)         /* LENGTH OF RESULT BUFFER IS */ 00179000
                                   /* TOO SHORT                  */ 00180000
        ) FIXED BINARY(31) STATIC;                                00181000
```

Figure 77. User edit routine field definitions for PL/I DXEECS control block (Part 4 of 4)

Handling Double-Byte Character Set Data

Double-byte character set (DBCS) data can appear in character columns or in columns with a graphic data type (GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC). If you need to devise edit routines that process this type of data, read this section.

Among the characters represented by the Japanese DBCS are Latin characters and Katakana characters. A Latin character has these characteristics:

- The first (leftmost) byte of the character has the value X'42'.
- The second byte of the character contains the EBCDIC equivalent.

A Katakana character has these characteristics:

- The first byte of the character contains X'43'.
- The second byte contains the EBCDIC equivalent.

Edit Codes for DBCS Data

You can use either Uxxxx or Vxxxx edit codes for DBCS data. The data that the edit routine receives is the same.

What the Edit Routine Receives

The data to be formatted is in the field ECSINPT, and the length of that data, in bytes, is in ECSINLEN. What you find in ECSINPT depends to some extent on where the data originates. More precisely, it depends on whether the column containing that data is a character column or one with a graphic data type.

Data from Graphic Columns

If the data to be formatted is from a column with a graphic data type, then the text in ECSINPT consists of this data preceded by one shift character and followed by another. Both shift characters are single bytes. For DBCS terminals, shift characters mark the start and end of a string of DBCS characters.

So denotes the shift character that introduces a DBCS string, and Si denotes the one that marks its end. So has the value X'0E'. Si has the value X'0F'.

The shift characters are included in the data length recorded in ECSINLEN. Thus, the length appearing in ECSINLEN is always greater by two than the length of the actual data. Because the data is presumably a string of DBCS characters, its length (in bytes) is always an even number.

Data from Character Columns

If the data to be processed comes from a character column, then the data in ECSINPT is just a copy of the column data. Unlike data from a graphic column, this data can hold single-byte characters and shift characters, as well as DBCS characters. To locate DBCS characters, you must search for the So

and Si characters that bracket the DBCS strings. If there are no So or Si characters in ECSINPT, the string contains no DBCS data. For example, ECSINPT contains the following string:

```
ccccSodedededededededeSiccSodedededededeSi
```

Here, c, d, and e stand for any possible byte, and So and Si are shift bytes. From the placement of the shift bytes, you can see that every occurrence of c represents a single-byte character, and that every occurrence of de represents a DBCS character.

Single-byte characters can represent Latin letters, Arabic numerals, and special characters such as plus signs and parentheses. For Japanese DBCS, they can also be Katakana characters. Some bytes meant to represent lowercase Latin might be displayed as Katakana symbols. You might have to devise edit codes that distinguish between columns containing lowercase English and those containing Katakana.

Ensuring the Edit Routine Returns the Right Results

Return the results in the ECSRSLT field, with trailing blanks for unused bytes. Make the results readable to the user's screen. This means that the resulting DBCS and EBCDIC characters must have the appropriate representations, and that the beginning and end of any string of DBCS characters are marked by So and Si characters.

Overflowing the ECSRSLT Field

Be careful not to overflow the ECSRSLT field, whose length is contained in the ECSRSLEN field. If your results do not fit, truncate them on the right. If the last character represented in the truncated results is a DBCS character, be certain to retain its rightmost byte, and to follow that character with an Si character.

Printing the Report Column

QMF copies the ECSRSLT field into the corresponding report column. The result is exactly as wide as the report column. If you 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:

- 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 14. 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 number of rows a user can retrieve from the database. You can use this default exit routine, or use High-Level Assembler (HLASM) to modify the routine or write one of your own.

Quick Start

Use the steps in Table 27 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 27. 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 200
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 above.	Page 204
To limit activities other than the number of rows retrieved from the database, use High-Level Assembler (HLASM) to modify the IBM-supplied governor exit routine or write a routine of your own.	Page 208
If you modify the IBM-supplied governor exit routine or write your own routine, translate, assemble, and link-edit the routine.	Page 229

Using the IBM-Supplied Governor Exit Routine

The governor exit routine supplied by IBM controls how many rows a user can retrieve from the database. The governor exit routine is shipped with two predefined values for the number of rows:

- A row prompt value warns users when the number of rows retrieved reaches 25 000, at which time the user sees the message shown in Figure 78:

```
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 78. Message displayed when a resource limit is approaching. The n symbol in the figure represents an NLID from Table 3 on page 10

- 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 78. When the IBM-supplied governor cancels data retrieval, the user sees the message shown in Figure 79:

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

Figure 79. 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 79. For example, if your procedure contains a command that requires the report to complete (such as ERASE), you receive the message shown in Figure 80:

```
Procedure canceled.
```

Figure 80. Message displayed when a procedure is canceled

Users using the SYSTEM profile, discussed in “Establishing a Profile Structure for Your Installation” on page 82, 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 for Number of Rows Retrieved”.

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 204.

Activating the Default Limits for Number of Rows Retrieved

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 81 from the SQL query panel:

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

Figure 81. 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 82 activate default values for user JONES (using English QMF) and user SCHMIDT (using German QMF).

   Important: Always specify a value for the TRANSLATION column, or you might change more rows in Q.PROFILES than you intend.

```
Base QMF (English)
        German NLF
UPDATE Q.PROFILES
        UPDATE Q.PROFILES
SET RESOURCE_GROUP = 'SYSTEM'
        SET RESOURCE_GROUP = 'SYSTEM'
WHERE CREATOR='JONES' AND
        WHERE CREATOR='SCHMIDT' AND
TRANSLATION='ENGLISH'
        TRANSLATION='DEUTSCH'
```

Figure 82. 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 82.

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 8 on page 81 for how to grant a user authority to connect to the database. Users who do not have DB2 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 204.

How a Governor Exit Routine Controls Resources

The governor uses two types of information to control resources:

- Information about the resource limits you set for a user, defined in a resource control table called Q.RESOURCE_TABLE.
- Information about the state of the user's session, which tells the governor how close the user's activity is coming to the resource limits defined for the resource group the user is in. This information is passed to the governor exit routine in the IBM-supplied control blocks DXEGOVA and DXEXCBA.

How the Governor Knows What the Resource Limits Are

Each row of the IBM-supplied Q.RESOURCE_TABLE contains:

- The name of a resource group (RESOURCE_GROUP), which characterizes one or more users whose activities you want to govern in the same manner.
- The name of the resource (RESOURCE_OPTION) you want to limit for the group of users named in RESOURCE_GROUP.
- Values (INTVAL, FLOATVAL, or CHARVAL) that define the limit for the resource option. Resource options can have integer values, floating-point values, or character values.

Table 28 on page 207 shows the structure of the Q.RESOURCE_TABLE as it is shipped by IBM. Q.RESOURCE_TABLE has the index Q.RESOURCE_INDEX. Keyed columns are RESOURCE_GROUP and RESOURCE_OPTION.

The Q.RESOURCE_TABLE is shipped by IBM with a predefined resource group called SYSTEM. The SYSTEM resource group has three predefined resource options, as shown in Figure 83. Use the CHARVAL column to indicate the limits defined in each row, as shown.

```
RESOURCE          RESOURCE
 GROUP             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
```

Figure 83. Default resource group and options for the IBM-supplied governor exit

SCOPE = 0
Activates governing for a particular resource group.

ROWLIMIT = 100 000
If the user decides to continue when warned, the governor exit routine cancels data retrieval activities after 100 000 rows are retrieved. (Retrieval is for FETCH only.)

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

IBM also supplies a view of this table, called Q.RESOURCE_VIEW, that includes all five columns of Q.RESOURCE_TABLE. Each time QMF calls the governor exit routine, QMF passes to the routine the resource control information stored in Q.RESOURCE_VIEW. The governor exit routine uses this resource information to help determine when the user reaches a resource limit.

How the Governor Knows When You Reach a Resource Limit

On a call to the governor exit routine, QMF queries Q.RESOURCE_VIEW, which shows what resource limits are defined in the resource control table for the resource group to which the user belongs. To determine the resource group, QMF checks the value of the RESOURCE_GROUP field of the user's row in the Q.PROFILES table and checks Q.RESOURCE_VIEW for a matching value.

QMF uses two control blocks, DXEGOVA and DXEXCBA, to pass information to the governor exit routine. The DXEGOVA control block contains information from Q.RESOURCE_VIEW about the limits you set for each user. The DXEXCBA control block contains information about the activities the user is performing in the current QMF session, which tells the governor how close the user is coming to the resource limits.

Figure 84 on page 204 shows how the governor limits use of resources.

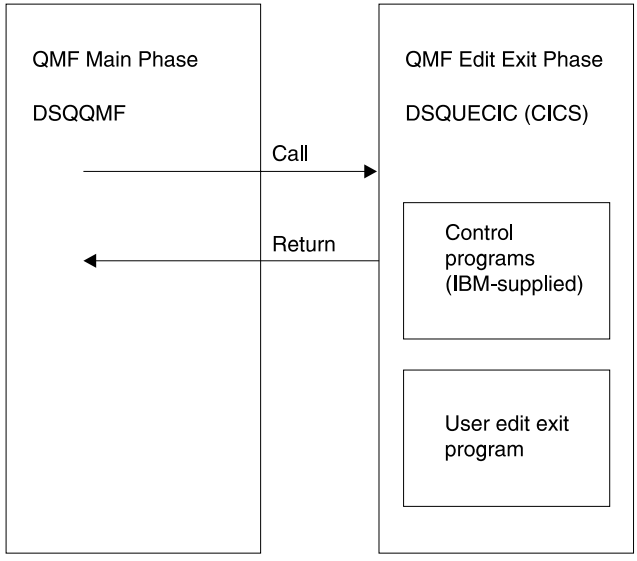


Figure 84. How a governor exit routine works with QMF

QMF calls the governor exit routine at a number of different points within the QMF session, as shown in Figure 84. These calls are called *function calls*. For more information about function calls, see "Points at Which QMF Calls the Governor" on page 211.

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 79 on page 200.

If you use the default limits for number of rows as discussed in Activating the Default Limits for Number of Rows Retrieved, the IBM-supplied governor exit routine also displays a warning before canceling the activity, as shown in Figure 78 on page 200. See "Defining Your Own Resource Limits" for how to activate this warning if you are not using the default values for the number of rows retrieved.

The IBM-supplied governor exit routine resets its count of the number of rows upon returning control to QMF, so that the number of rows is not cumulative across calls to the governor.

Defining Your Own Resource Limits

This section explains how to create a new resource group, for which the resource is the number of rows retrieved from the database. If you want to

define resource limits other than the number of rows, you need to modify the IBM-supplied governor exit routine or write an exit routine of your own. See "Modifying the IBM-Supplied Governor Exit Routine or Writing Your Own" on page 208 for more information on the facilities you can use.

Use the following procedure to add a resource group to the resource control table. This procedure adds a resource group named GROUP1, where the governor prompts a user in GROUP1 when the number of rows reaches 10 000, and cancels the user's activity when the number of rows reaches 15 000. The procedure also shows an example of how to add a user to a resource group.

1. Run the query in Figure 85 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 85. Activating prompting for row limit

2. Run the query in Figure 86 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 86. Activating cancellation of activities when user reaches row limit

3. Run the query shown in Figure 87 to turn on governing for the GROUP1 resource group.

   SCOPE is a resource option that activates or deactivates governing. Each

```
INSERT INTO Q.RESOURCE_VIEW (RESOURCE_GROUP,RESOURCE_OPTION,INTVAL)
   VALUES('GROUP1','SCOPE',0)
```

Figure 87. Turning on the governor for a particular resource group

   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 will not be governed. Set INTVAL to 1 to deactivate governing.

4. Run a query similar to the one in Figure 88 on page 206 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 88. 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 3 on page 10.

5. 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 8 on page 81 for how to grant a user authority to connect to the database. Users who do not have DB2 CONNECT authority can end the current QMF session and begin another to activate the new resource group.

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 28 on page 207.

Figure 89 on page 207 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 89 on page 207 and Figure 90 on page 207.

```
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 89. 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 90 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 90. Redefining the Q.RESOURCE_VIEW

Table 28. Structure of the Q.RESOURCE_TABLE table

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
RESOURCE_GROUP	CHAR	16	No	Contains the name of the resource group. Update the RESOURCE_GROUP field of the user's row in Q.PROFILES to activate governing for that user.
RESOURCE_OPTION	CHAR	16	No	Your own name for a resource you want to monitor.
INTVAL	INTEGER		Yes	Reflects resource limit for resource options that have integer values. For example, number of rows retrieved from the database is a resource that has an integer value.
FLOATVAL	FLOAT		Yes	Reflects resource limit for resource options that have floating point values. FLOATVAL is null for the IBM-supplied governor.

Table 28. Structure of the Q.RESOURCE_TABLE table (continued)

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
CHARVAL	VARCHAR	80	Yes	Reflects resource limit for resource options that have character values. For example, you might establish a DAY_OF_WEEK resource option and assign MONDAY to CHARVAL so that QMF users can log on to QMF only on Mondays. CHARVAL is used as a comment column in the IBM-supplied governor.

Modifying the IBM-Supplied Governor Exit Routine or Writing Your Own

If you decide to govern resources other than the number of rows returned from the database, 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 211 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 215 explains this step.
3. Establish addressability to the QMF cancellation service to cancel activities. "Canceling User Activity" on page 228 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 228 explains this step.
5. Translate, assemble, and link-edit your governor exit routine, whether you modified the IBM-supplied governor exit routine or wrote your own. "Translating, Assembling, and Link-Editing Your Governor Exit Routine" on page 229 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 29 on page 209 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 29 with the NLID (from Table 3 on page 10) that matches the NLF you're using.

Table 29. IBM-supplied governor components

Member Name	Sublibrary	Function
DSQUnGV3.PHASE	PRD2.PROD	Executable phase installed during QMF installation.
DSQUnGV3.Z	PRD2.PROD	Source code for governor exit routine.
DXEGOVA.A	PRD2.PROD	DSECT for the DXEGOVA control block.
DXEXCBA.A	PRD2.PROD	DSECT for the DXEXCBA control block.
DXEUnGV3.A	PRD2.PROD	Contains text and related definitions for the governor exit routine cancellation message in CICS.
DXEUnGM.Z	PRD2.PROD	Contains CICS basic mapping support (BMS) information, which describes how the governor prompts appear on the screen.
DSQ3nGLK.Z	PRD2.PROD	Job that translates, assembles, and link-edits the IBM-supplied governor exit routine and the BMS map.

You can find these members in the sublibrary where QMF was installed (the default sublibrary is PRD2.PROD).

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 phase DSQU*n*GV3 for each language environment. For example, if you have both English and German QMF installed, use the phase DSQUEGV3 for English and the phase DSQUDGV3 for German.

You can share the resource control table (Q.RESOURCE_TABLE or one you create yourself) and the Q.RESOURCE_VIEW between language environments, just as the Q.PROFILES table can contain profiles for English or any NLF.

How CICS Interfaces with the Governor Exit Routine

At the start of a user's session, QMF issues an EXEC CICS LOAD command to bring the governor into the user's virtual storage. For performance reasons, an assembler call interface is used between QMF and the governor exit routine. The governor exit routine must provide fast performance because, depending on which resources you are trying to control, it could be called on every row retrieved from the database.

The CICS control block interface to the governor exit consists of the following parts:

- Interface control blocks DXEXCBA.A and DXEGOVA.A (shipped with QMF)
- CICS-supplied prolog and epilog macros DFHEIENT and DFHEIRET (shipped with CICS)
- Command interface modules DFHEAI and DFHEAI0 (shipped with CICS)
- The governor exit program (named DSQU*n*GV3)

Figure 91 shows the program structure of a governor exit routine:

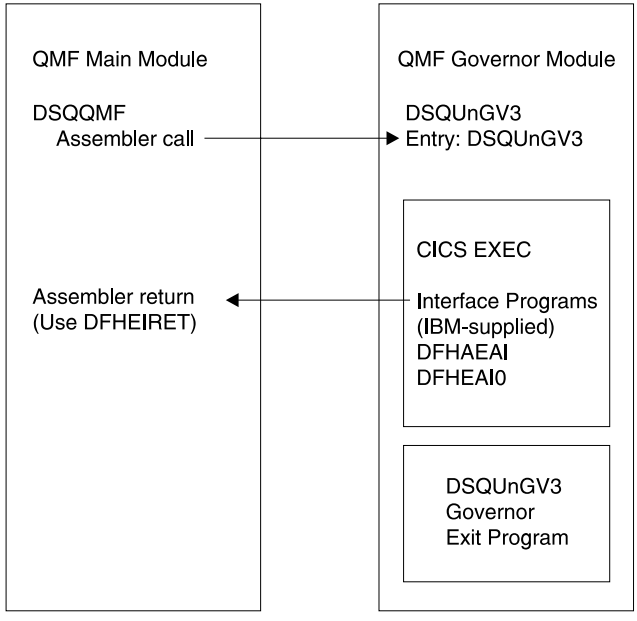


Figure 91. CICS processing that interfaces QMF with the governor exit

The governor exit routine executes on the same program level as the main QMF program.

The entry point to the governor exit routine is DSQU*n*GV3. When it calls the governor exit routine, QMF always branches to the address returned by CICS as the result of an EXEC CICS LOAD command.

If the load fails or the phase doesn't support 31-bit addressing mode, QMF issues a warning message, disables the governor exit, and continues the session without the governor.

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:

- 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

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 30 on page 215. Each type of function call has a specific value for the GOVFUNCT field. These values are shown in Figure 93 on page 214.

What Happens Upon Entry to the Governor Exit Routine

QMF calls the governor exit routine by branching to the address of the entry point DSQU*n*GV3. Upon entry to the governor exit routine:

- Register 1 contains a CICS parameter list suitable for processing by CICS-supplied macros DFHEIENT and DFHEIRET. Figure 92 shows the contents of Register 1 on a call to the governor.

  DFHEIBLK is the address of the CICS communications area. DFHCOMMA contains two pointers, one to the DXEXCBA control block and the other to the DXEGOVA control block.

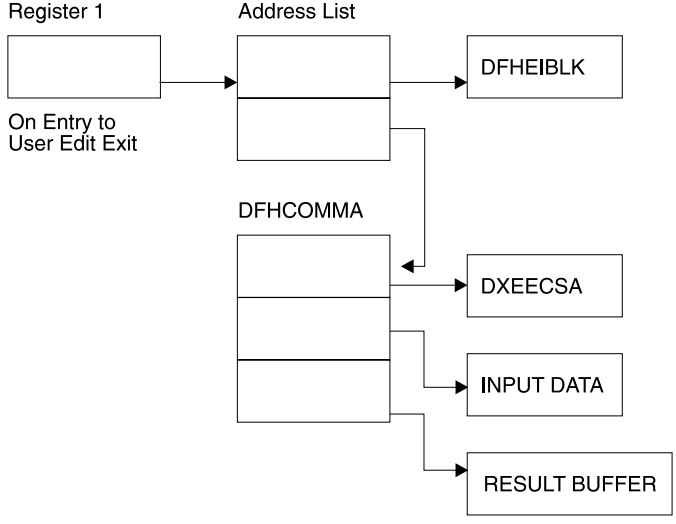


Figure 92. Contents of Register 1 on a call to the governor exit routine

- Register 13 contains the address of a standard CICS working storage area as described by CICS-supplied macro DFHEISTG.
- Register 14 contains the return address.

Establishing Addressability for Function Calls

Because QMF always branches to an entry point named DSQU*n*GV3 when it calls the governor, you 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 9. 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 93 on page 214. (If you need more information about the activity that

occurs at each function call, see “Points at Which QMF Calls the Governor” on page 211 .)

```
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
```

Figure 93. Character and numeric values for the GOVFUNCT field of DXEGOVA

To improve performance in your own exit routine, you can follow the convention the IBM-supplied governor uses, and equate the values of GOVFUNCT with binary numbers by using a branch table. QMF uses the branch table to find the addresses to branch to for each type of function call.

Figure 94 shows an example of some code that identifies branch addresses for the IBM-supplied governor.

```
         XR    R07,R07              ZERO REGISTER 7
         IC    R07,GOVFUNCT         IDENTIFY EXIT TYPE
         SLL   R07,2                DETERMINE BRANCH TABLE OFFSET
         LA    R15,FUNBTAB(R07)     GET BRANCH TABLE ADDRESS
         L     R15,0(R15)           GET BRANCHING ADDRESS
         BALR  R14,R15              BRANCH TO THE APPROPRIATE CODE
            . . .
            . . .
            . . .
            . . .
FUNBTAB  DS    0F
         DC    A(BYPASS)          VALUE "0"  - UNUSED
         DC    A(INIT)            VALUE "1"  - QMF INITIALIZATION
            . . .
            . . .
            . . .
```

Figure 94. Identifying the type of function call and branching to the appropriate address

Because the governor program runs on the same program level as QMF, use caution when using any EXEC CICS commands that change the environment (for example, CICS HANDLE CONDITION). If you need to use the CICS HANDLE CONDITION, use EXEC CICS PUSH and EXEC CICS POP to save and restore the QMF environment.

Use the standard HLASM RETURN statement to return control to QMF after every call.

Attention: Do not use the EXEC CICS RETURN command to return to QMF. Using EXEC CICS RETURN ends the QMF session or causes unpredictable results.

Passing Resource Control Information to the Governor Exit

If you have not done so already, read the following sections, which describe how to set up resource control information in a format the governor can use:

- "How a Governor Exit Routine Controls Resources" on page 202
- "Defining Your Own Resource Limits" on page 204

QMF passes resource control information using two control blocks named DXEGOVA and DXEXCBA. These are shown in Figure 95 on page 218 and Figure 97 on page 225. Their addresses are passed to the governor on every function call. The DSECT DXEXCBA (shipped as DXEXCBA.A) and the DSECT DXEGOVA (shipped as DXEGOVA.A) are located in the sublibrary where QMF is installed. Include these DSECTs in your program using the HLASM 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 202 for more information on how this view is used.

Table 30 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 VSE/ESA environment because it is used in only OS/390 or VM operating environments.

Table 30. 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 228.

Table 30. Fields of the DXEGOVA interface control block to the governor (continued)

Field	Data Type	Purpose
GOVFUNCT	XL1	Indicates the type of function call. Possible values are: • GOVINIT (session initialization); GOVTERM (session termination) • GOVSCMD (start command); GOVECMD (end command) • GOVCONN (connect command) • GOVSDBAS (start database retrieval operation); GOVEDBAS (end database retrieval operation) • GOVSACTV (suspend QMF activity for user think time); GOVRACTV (resume QMF activity) Code to use this field appears in "Establishing Addressability for Function Calls" on page 213.
GOVGROUP	CL16	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 202.
GOVNAME	CL8	Contains the name of the control block (DXEGOVA). This value does not change during a session. It can serve as an eye catcher in a dump of virtual storage.
GOVOROWS	F	Contains the number of rows for the user's resource group in the resource control table. This value does not change during a session, and can be zero.
GOVRESC	10XL128	Contains information from the resource control table. This information is divided into 10 contiguous blocks of storage that are structured like DSECT GOVRESCT. A block contains information about one of the rows for the user's resource group in the QMF resource control table. • If the resource group has less than 10 rows, unused blocks are those at the end of the field. • If the resource group has more than 10 rows, use the field named GOVNEXTR (in the GOVRESCT DSECT) to access additional rows. All blocks are part of a chain, as described in "Addressing the Resource Control Table" on page 219. The value of this field does not change during a session.

Table 30. Fields of the DXEGOVA interface control block to the governor (continued)

Field	Data Type	Purpose
GOVRESCT	DSECT	Describes the block of storage containing information on one of the user's rows of the resource control table. All such blocks are linked together in a chain discussed in "Addressing the Resource Control Table" on page 219. The following fields are within the block: **GOVOPTN(CL16)** Contains the value in the RESOURCE_OPTION column of the resource control table. Blocks in the chain are ordered alphabetically on the content of this field. **GOVNULLI(H)** Null indicator for INTVAL column. **GOVINTVL(F)** Value of INTVAL column. **GOVNULLF(H)** Null indicator for FLOATVAL column. **GOVFLOAT(D)** Value of FLOATVAL column. **GOVNULLC(H)** Null indicator for CHARVAL column. **GOVCHLEN(H)** Length of data in CHARVAL column. **GOVCHAR(CL80)** Value in CHARVAL column. **GOVNEXTR(A)** Points to the block of data for the next resource table row. Contains zero if this is the last row. Any null indicator in the structure is zero when its corresponding column value isn't null. If the column value is null, the indicator is not zero.
GOVSQLCA	A	Address of the SQL communications area (SQLCA), which holds information about the SQL SELECT query on the resource control view (Q.RESOURCE_VIEW).
GOVSQLRC	F	Return code from the SQL SELECT query on the resource control view (Q.RESOURCE_VIEW). If it is nonzero, the query failed and no rows are passed to the governor.
GOVUSERS	CL2048	Scratchpad area, retained between session calls. QMF does not change this value.

Figure 95 on page 218 shows the structure of the DXEGOVA control block.

```
************************************************************************ 00001000
*                                                                      * 00002000
*        CONTROL BLOCK NAME: DXEGOVA                                   * 00003000
*                                                                      * 00004000
*        FUNCTION:                                                     * 00005000
*                                                                      * 00006000
*            THIS IS THE INTERFACE CONTROL BLOCK BETWEEN QMF AND        * 00007000
*            THE GOVERNOR EXIT ROUTINE.                                * 00008000
*                                                                      * 00009000
*        STATUS: VERSION 7 RELEASE 1 LEVEL 0                           * 00010000
*                                                                      * 00011000
*        INNER CONTROL BLOCKS: NONE                                    * 00012000
*                                                                      * 00013000
*        CHANGE ACTIVITY: NA                                           * 00014000
*                                                                      * 00015000
*        CHANGE DATE: NA                                               * 00016000
*                                                                      * 00017000
************************************************************************ 00018000
*                                                                        00019000
DXEGOVA  DSECT                                                          00020000
         DS    0D                                                       00021000
GOVNAME  DS    CL8           -- CONTROL BLOCK IDENTIFICATION            00022000
         SPACE                                                          00023000
GOVEXCTL DS    XL72          -- EXIT CONTROL                            00024000
         ORG   GOVEXCTL                                                 00025000
GOVFUNCT DS    XL1           ----- FUNCTION CODE                        00026000
GOVINIT  EQU   1             -------- INITIALIZATION OF SESSION         00027000
GOVTERM  EQU   2             -------- TERMINATION OF SESSION           00028000
GOVSCMD  EQU   3             -------- START COMMAND                     00029000
GOVECMD  EQU   4             -------- END COMMAND                       00030000
GOVCONN  EQU   5             -------- CONNECT COMMAND                   00031000
GOVSDBAS EQU   6             -------- START DATA BASE                   00032000
GOVEDBAS EQU   7             -------- END DATA BASE                     00033000
GOVSACTV EQU   8             -------- SUSPEND QMF ACTIVITY             00034000
GOVRACTV EQU   9             -------- RESUME QMF ACTIVITY              00035000
GOVABEND EQU   10            -------- QMF ABEND OPERATION              00036000
GOVPAD10 DS    CL7           ----- RESERVED FIELD                       00037000
         SPACE                                                          00038000
GOVCADDR DS    A             ----- ADDR TO BRANCH TO FOR CANCELLATION   00039000
         SPACE                                                          00040000
GOVOROWS DS    F             ----- NUMBER OF OPTION ROWS RETRIEVED      00041000
         SPACE                                                          00042000
GOVSQLRC DS    F             ----- RESOURCE TABLE SQL RETURN CODE       00043000
         SPACE                                                          00044000
GOVSQLCA DS    A             ----- ADDRESS OF SQLCA FOR ERROR CONDITION 00045000
         SPACE                                                          00046000
GOVGROUP DS    CL16          ----- GROUP NAME                           00047000
GOVPAD20 DS    CL32          ----- RESERVED FIELD                       00048000
         SPACE                                                          00049000
GOVUCTL  DS    XL304         -- USER CONTROL AREA                       00050000
```

Figure 95. The DXEGOVA control block (Part 1 of 2)

```
         ORG   GOVUCTL                                                  00051000
GOVUSERS DS    CL2048       ----- USER SCRATCH PAD AREA                 00052000
GOVPAD30 DS    CL48         ----- RESERVED FIELD                        00053000
         SPACE                                                         00054000
         DS    0D                                                      00055000
GOVRESC  DS    10XL128      -- RESOURCE CONTROL TABLE                  00056000
         ORG   GOVRESC                                                 00057000
GOVRESCT DSECT              -- RESOURCE CONTROL TABLE MAPPING          00058000
         DS    0D                                                      00059000
GOVOPTN  DS    CL16         ----- RESOURCE OPTION                      00060000
GOVNULLI DS    H            ----- INTEGER NULL INDICATOR               00061000
GOVPAD40 DS    CL2          ----- RESERVED FIELD                       00062000
GOVINTVL DS    F            ----- INTEGER OPTION REPRESENTATION        00063000
GOVNULLF DS    H            ----- FLOATING POINT NULL INDICATOR        00064000
GOVPAD50 DS    CL6          ----- RESERVED FIELD                       00065000
GOVFLOAT DS    D            ----- FLOATING POINT OPTION REPRESENTATION 00066000
GOVNULLC DS    H            ----- CHARACTER NULL INDICATOR             00067000
GOVCHLEN DS    H            ----- LENGTH OF THE CHARACTER OPTION       00068000
GOVCHAR  DS    CL80         ----- CHARACTER OPTION REPRESENTATION      00069000
GOVNEXTR DS    A            ----- POINTER TO NEXT RESOURCE CONTROL ROW 00070000
```

Figure 95. The DXEGOVA control block (Part 2 of 2)

Addressing the Resource Control Table

The GOVGROUP field of the DXEGOVA control block holds the value of the RESOURCE_GROUP column of Q.RESOURCE_VIEW, the view defined on the resource control table.

All information about the user's resource options is stored in blocks; there is one block for each of the user's resource options you decide to monitor.

The first block defines the first resource option and is stored in the DXEGOVA control block as the DSECT GOVRESCT. This DSECT is shown in the last part of Figure 95. The address of this DSECT is defined in the DXEGOVA field GOVRESC. You can establish addressability to the GOVRESC field in your own routine using the address of the GOVRESCT DSECT.

Negative half-word integers in the DSECT represent null values entered for INTVAL, CHARVAL or FLOATVAL in the Q.RESOURCE_VIEW; zero or positive half-words indicate a value in that column of Q.RESOURCE_VIEW.

The blocks that store the resource control information form a chain in which a pointer in one block points to the beginning of the next block (the next resource option) in the chain. For example, the GOVNEXTR DS statement in the GOVRESCT DSECT in Figure 95 contains the address of the next block in the chain of resource control information. Each block in the chain has a GOVNEXTR DS statement. In the final block, the GOVNEXTR DS statement contains zeros to mark the end of the user's resource control information.

Figure 96 shows a part of the code for the IBM-supplied governor that processes the blocks of resource control information. In this code, GOVRESC points to the GOVRESCT DSECT.

```
         L     R08,GOVOROWS      GET NUMBER OF RESOURCE TABLE ROWS
         LTR   R08,R08           ANY RESOURCE TABLE ROWS?
         BZ    ENDRESST          NO, SKIP RESOURCE INITIALIZATION
         LA    R05,GOVRESC       GET ADDRESS OF 1ST RESOURCE ROW
         USING GOVRESC,R05       BASE RESOURCE RECORD ENTRY
LOOK4RES DS    0H                MAIN LOOP THRU RESOURCE ROWS
         LTR   R05,R05           ANY MORE RESOURCE TABLE ROWS?
         BZ    ENDRESST          NO, END RESOURCE INITIALIZATION
⋮
         L     R05,GOVNEXTR      GET ADDRESS ON NEXT RESOURCE ROW
         B     LOOK4RES          BEGIN NEXT ITERATION
ENDRESST DS    OH                -- BRANCH HERE WHEN FINISHED READING ALL ROWS

         . . .
         . . .
         . . .
         . . .

DXEGOVA  DSECT

         .  .  .
         .  .  .
         .  .  .

GOVRESC  DS    10XL128           -- RESOURCE CONTROL TABLE
         ORG   GOVRESC
GOVRESCT DSECT                   -- DSECT FOR RESOURCE ROW
         .  .  .
         .  .  .
         .  .  .
GOVNEXTR DS    A                 -- POINTER TO NEXT RESOURCE ROW

         .  .  .
         .  .  .
         .  .  .
```

Figure 96. Resource initialization

Structure of the DXEXCBA Control Block

The DXEXCBA control block passes to the governor exit routine information about the state of the QMF session upon entry to the governor. The governor combines this information with information on resource limits (contained in DXEGOVA) to determine when the resource limits are exceeded and when to cancel the user's activity.

For example, you can define a resource option that does not allow user JONES to use the EDIT TABLE command. You can then write your governor

exit routine so that, if the XCBQRYP field of the DXEXCBA control block indicates an EDIT TABLE command, the governor exit calls the QMF cancellation service to cancel the command.

Table 31 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 VSE/ESA environment because it is used in only OS/390 or VM operating environments.

Table 31. 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. 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.
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, which might change on a CONNECT command.
XCBCAN	CL1	Indicates whether the user or the governor requested cancellation of the current command. The field is set to 1 if cancellation is requested. Zero indicates that no cancellation was requested. This field is reset to zero before the function call for the command's termination.
XCBCLOC	CL18	Contains the current location name. The current location is always a VSE DB2 database name for QMF for VSE/ESA Version 3.2.
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.

Table 31. Fields of the DXEXCBA interface control block to the governor (continued)

Field	Data Type	Purpose
XCBCMDP	A	Points to the string containing the command to be run. This field is reset when QMF validates a command at some point before the function call for the start of the command. The field is reset to zeros before the function call when the command completes. If a command synonym is being run, it appears here.
XCBCVERB	CL18	Holds the verb of the current command. This field changes value on the function call for the start of a command. The value does not change between calls.
XCBDBMG	CL1	Identifies the database manager. This value is always set to 1 for QMF for VSE/ESA Version 3.2, indicating DB2 for VM/ESA or VSE/ESA
XCBEMODE	CL1	Indicates the current mode of the QMF session: **1** Interactive **2** Noninteractive (batch) This value does not change during a session. See “Starting a Noninteractive QMF Session (DSQSMODE)” on page 69 for more information on starting a noninteractive session.
XCBERRET	F	Contains the return code to be used in the default cancellation message. For more information about this message, see “Providing Messages for Canceled Activities” on page 228.
XCBINCI (TSO ISPF only)	CL1	This field is not supported in CICS/VSE.
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 75 for more information about this parameter.
XCBLOGM	CL1	Indicates if QMF should log a message in the QMF trace data. 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 228. Using the QMF trace facility is described in “Using the QMF Trace Facility” on page 242.
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 page 228.
XCBMSGNO (TSO ISPF only)	CL8	This field is not supported in CICS/VSE.

Table 31. Fields of the DXEXCBA interface control block to the governor (continued)

Field	Data Type	Purpose
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 3 on page 10.) Value does not change during a session.
XCBPANEL (TSO ISPF only)	CL8	This field is not supported in CICS/VSE.
XCBPLAN	CL8	This field applies to TSO only, which is not supported in CICS/VSE.
XCBQCE	F	Contains the value of the SQLDERRD(4) field in the SQLCA returned from VSE DB2. The value is a short floating-point number, divided by X'1000' and converted to a QMF internal decimal. The integer part of this decimal appears in the database status (“relative cost estimate”) panel. The value is set to zero on the function call when the command finishes running. The field contains zeros if the operation is not a data retrieval query.
XCBQERR	CL1	Tells whether a QMF error occurred since the previous function call: 0 indicates no error occurred; 1 indicates an error occurred.
XCBQMF	CL10	Identifies the current release of QMF. This value is `QMF V7R1`, and does not change during a session.
XCBQRYP	A	Contains the address of a copy of the query that QMF passes to the database for execution. The governor inspects the query upon a call to start database activity (before any data retrieval) and determines whether to cancel the activity. The address is set to zero either at the beginning of the session or when the data object is reset or imported to temporary storage. This field contains information only when data retrieval is requested through one of the commands in the following list; no information is provided for queries on VSE DB2 system tables or QMF control tables. **DISPLAY TABLE** EDIT TABLE **ERASE TABLE** EXPORT TABLE **IMPORT TABLE** PRINT TABLE **RUN QUERY** SAVE DATA
XCBREFR	CL1	Indicates whether QMF refreshes the screen after returning from the governor; 1 indicates a refresh; 0 indicates no refresh. If your governor displays any screen information, set this field to 1.

Table 31. Fields of the DXEXCBA interface control block to the governor (continued)

Field	Data Type	Purpose
XCBRELN	CL2	Identifies the QMF release level. For QMF for VSE/ESA Version 3.2, this is 09. The value does not change during a session.
XCBRGRP	CL16	Contains the name of the user's resource group. This value can change after a CONNECT command.
XCBROWSF	F	Reflects the number of rows retrieved into the data object. Initially zero, this field changes value whenever more rows are retrieved. All data retrieval is counted whether data is retrieved from the database or imported from CICS temporary storage or transient data queues. QMF does not reset this field, but the governor can. For example, if your governor exit routine monitors the number of database rows retrieved, you can set this field to zero on the function call for the end of the command that began the data retrieval.
XCBSYST	CL1	Identifies the current operating system. The value does not change during a session, and is usually set to 5, indicating CICS. 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 VSE/ESA 6, see "Providing the Correct Profile for the User's Operating Environment" on page 88.
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 228. 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 242.
XCBUSER	CL8	This field is not used in CICS; it contains blanks.
XCBUSERS	CL2048	Scratchpad area in which you can store results you want the governor to save from one call to the next. It is initially set to blanks. QMF does not change this value.

Figure 97 on page 225 shows the structure of the DXEXCBA control block.

```
*************************************************************************** 00001000
*                                                                         * 00002000
*        CONTROL BLOCK NAME: DXEXCBA                                      * 00003000
*                                                                         * 00004000
*        FUNCTION:                                                        * 00005000
*                                                                         * 00006000
*            THIS IS THE INTERFACE CONTROL BLOCK BETWEEN QMF AND           * 00007000
*            EXIT ROUTINES.                                               * 00008000
*                                                                         * 00009000
*        STATUS: VERSION 7 RELEASE 1 LEVEL 0                              * 00010000
*                                                                         * 00011000
*        INNER CONTROL BLOCKS: NONE                                       * 00012000
*                                                                         * 00013000
*        CHANGE ACTIVITY:                                                 * 00014000
*                                                                         * 00015000
*                                                                         * 00016000
*************************************************************************** 00017000
*                                                                           00018000
DXEXCBA  DSECT                                                             00019000
         DS    0D                                                          00020000
XCBNAME  DS    CL8           -- CONTROL BLOCK IDENTIFICATION               00021000
         SPACE                                                             00022000
XCBEXCTL DS    XL190         -- EXIT CONTROL                               00023000
         ORG   XCBEXCTL                                                    00024000
XCBAUTH  DS    CL8           ----- AUTHORIZATION ID                        00025000
XCBUSER  DS    CL8           ----- USER ID                                 00026000
XCBPLAN  DS    CL8           ----- PLAN ID                                 00027000
         SPACE                                                             00028000
XCBQMF   DS    CL10          ----- CURRENT VERSION/RELEASE                 00029000
         SPACE                                                             00030000
XCBRELN  DS    CL2           ----- QMF RELEASE LEVEL                       00031000
         SPACE                                                             00032000
XCBTRACE DS    CL1           ----- QMF EXIT TRACE LEVEL                    00033000
XCBTOFF  EQU   C'0'          -------- NO TRACING                           00034000
XCBTPART EQU   C'1'          -------- PARTIAL TRACING                      00035000
XCBTFULL EQU   C'2'          -------- FULL TRACING                         00036000
         SPACE                                                             00037000
XCBSYST  DS    CL1           ----- OPERATING SYSTEM                        00038000
XCBSYSTX EQU   C'3'          -------- MVS/ESA or XA (TSO,APPC, native)    00039000
XCBSYSTV EQU   C'4'          -------- CMS/VM/ESA                           00040000
XCBSYSTY EQU   C'5'          -------- CICS (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
```

Figure 97. The DXEXCBA control block (Part 1 of 3)

```
XCBKPARN EQU   C'0'          -------- LATIN                                00048000
XCBKPARY EQU   C'1'          -------- DBCS                                 00049000
         SPACE                                                             00050000
XCBDBMG  DS    CL1           ----- DATA BASE MANAGER                       00051000
XCBDBMGS EQU   C'1'          -------- DB2 FOR VM/VSE                       00052000
XCBDBMGD EQU   C'2'          -------- DB2 FOR OS/390                       00053000
XCBDBMGW EQU   C'3'          -------- WORKSTATION DB2                      00054000
         SPACE                                                             00055000
XCBEMODE DS    CL1           ----- CURRENT EXECUTION MODE                  00056000
XCBIACTV EQU   C'1'          -------- INTERACTIVE MODE                     00057000
XCBBATCH EQU   C'2'          -------- BATCH MODE                           00058000
         SPACE                                                             00059000
XCBAIACT DS    CL1           ----- CURRENT INTERACT MODE                   00060000
XCBAIACY EQU   C'1'          -------- INTERACTIVE EXECUTION                00061000
XCBAIACN EQU   C'0'          -------- NOT INTERACTIVE EXECUTION            00062000
         SPACE                                                             00063000
XCBINCI  DS    CL1           ----- CURRENT COMMAND INTERFACE STATE         00064000
XCBINCIY EQU   C'1'          -------- COMMAND INTERFACE ACTIVE             00065000
XCBINCIN EQU   C'0'          -------- COMMAND INTERFACE NOT ACTIVE         00066000
         SPACE                                                             00067000
XCBINPRC DS    CL1           ----- PROCEDURE OR LIST CMD EXEC STATE        00068000
XCBPRCY  EQU   C'1'          -------- RUNNING A PROCEDURE OR LIST CMD      00069000
XCBPRCN  EQU   C'0'          -------- NOT RUNNING PROCEDURE OR LIST CMD    00070000
         SPACE                                                             00071000
XCBCVERB DS    CL18          ----- CURRENT COMMAND VERB                    00072000
         SPACE                                                             00073000
XCBCAN   DS    CL1           ----- CANCEL CURRENT COMMAND INDICATOR        00074000
XCBCANN  EQU   C'0'          -------- NO CANCELLATION                      00075000
XCBCANY  EQU   C'1'          -------- CANCELLATION IN PROGRESS             00076000
         SPACE                                                             00077000
XCBACTIV DS    CL1           ----- TYPE OF DATA BASE ACTIVITY              00078000
XCBOPEN  EQU   C'1'          -------- OPEN                                 00079000
XCBFETCH EQU   C'2'          -------- FETCH                                00080000
XCBPREP  EQU   C'3'          -------- PREPARE                              00081000
XCBDESCR EQU   C'4'          -------- DESCRIBE                             00082000
XCBCLOSE EQU   C'5'          -------- CLOSE                                00083000
XCBEXEC  EQU   C'6'          -------- EXECUTE                              00084000
XCBEXECI EQU   C'7'          -------- EXECUTE IMMEDIATE                    00085000
XCBPAD20 DS    CL9           ----- RESERVED FIELD                          00086000
         SPACE                                                             00087000
XCBRGRP  DS    CL16          ----- RESOURCE GROUP NAME                     00088000
                                                          00100000
```

Figure 97. The DXEXCBA control block (Part 2 of 3)

```
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 XCBQERR  DS    CL1           ----- QMF ERROR INDICATOR   00101000
XCBQERRN EQU   C'0'          -------- NO QMF ERROR DETECTED             00102000
XCBQERRY EQU   C'1'          -------- QMF ERROR DETECTED                00103000
XCBCLOC  DS    CL18          ----- CURRENT LOCATION NAME                00104000
XCBPAD40 DS    CL41          ----- RESERVED FIELD                       00105000
         SPACE                                                          00106000
XCBQRYP  DS    A             ----- POINTER TO SQL QUERY                 00107000
*                            -------- QUERY LENGTH IS FIRST HALFWORD     00108000
         SPACE                                                          00109000
XCBUCTL  DS    XL432         -- USER CONTROL AREA                       00110000
         ORG   XCBUCTL                                                  00111000
XCBERRET DS    F             ----- EXIT ERROR RETURN CODE               00112000
XCBMGTXT DS    CL78          ----- EXIT ERROR MESSAGE TEXT              00113000
XCBMSGNO DS    CL8           ----- ISPF MESSAGE NUMBER                  00114000
XCBPANEL DS    CL8           ----- ISPF MESSAGE HELP PANEL              00115000
XCBLOGM  DS    CL1           ----- LOG MESSAGE INDICATOR                00116000
XCBLOGMN EQU   C'0'          -------- QMF SHOULD NOT LOG MESSAGE        00117000
XCBLOGMY EQU   C'1'          -------- QMF SHOULD LOG MESSAGE            00118000
XCBREFR  DS    CL1           ----- REFRESH SCREEN INDICATOR             00119000
XCBREFRN EQU   C'0'          -------- QMF DOES NOT HAVE TO REFRESH SCR  00120000
XCBREFRY EQU   C'1'          -------- QMF SHOULD REFRESH SCREEN         00121000
XCBPAD50 DS    CL28          ----- RESERVED FIELD                       00122000
         SPACE                                                          00123000
XCBUSERS DS    CL2048        -- USER SCRATCH PAD AREA                   00124000
XCBPAD60 DS    CL48          ----- RESERVED FIELD                       00125000
```

Figure 97. The DXEXCBA control block (Part 3 of 3)

Storing Resource Control Information for the Duration of a QMF Session

You can use the information passed to the governor on the first call of a session for subsequent calls to the governor routine. Although your governor exit routine can issue an EXEC CICS GETMAIN command to obtain the necessary storage to hold the resource control information, it might be more efficient for you to use the 2048-byte scratchpad areas provided in the DXEGOVA and DXEXCBA control blocks. These fields can contain any information you need to store. The information persists from one call to the governor to the next.

The IBM-supplied governor uses the code shown in Figure 98 on page 228 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 Figure 98 with XCBUSERS.

```
LA    WORKPTR,GOVUSERS
USING WORK,WORKPTR
```

Figure 98. Establishing addressability to the governor scratchpad area

In Figure 98, 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.

Canceling User Activity

When users reach their resource limits, you can call the QMF cancellation service to cancel user activity. For example, your governor exit routine might cancel the following:

- A QMF session during a function call at the start of a QMF session
- The current command during a number of different function calls, and any commands that start database activity

The code for canceling either of these activities is contained in the source program DSQU*n*GV3. To have your governor call the QMF cancellation service to cancel an activity, branch to the address that appears in the DXEGOVA control block field named GOVCADDR. Figure 99 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 99. 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.

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 XCBMGTXT and XCBERRET fields of the DXEXCBA control block:

XCBMGTXT
Contains the message text.

XCBERRET
Contains the error return code.

Upon entry to the governor, XCBMGTXT contains blanks, and XCBERRET contains binary zeros. The value of XCBERRET determines what message is displayed on the screen:

- If you want to use the message `OK, command canceled`, leave the zero value in XCBERRET.
- If you want to use the message `A governor exit cancel occurred with return code xxxxx`, use a nonzero value for XCBERRET; this nonzero value appears in the message in place of `xxxxx`.

If QMF initialization is canceled by the governor exit, the preceding messages for XCBMGTXT and XCBERRET appear in the user's trace data rather than on the screen.

Set XCBLOGM to 1 to log a message in the user's trace data for any function call in your own governor exit routine. If the value of XCBERRET is nonzero, the IBM-supplied governor logs cancellation messages in the user's trace data by setting the XCBLOGM field of the DXEXCBA control block to a value of 1.

The trace facility writes messages to CICS temporary storage queues or transient data queues 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 to get trace output. For additional details on using the QMF trace facility, see "Using the QMF Trace Facility" on page 242.

The IBM-supplied governor does not log messages for termination function calls.

Translating, Assembling, and Link-Editing Your Governor Exit Routine

Whether you're modifying the IBM-supplied governor exit routine or writing a routine of your own, you need to translate, assemble, and link-edit the routine. Use the sample JCL shown in this section to help you.

Translating Your Governor Exit Program for CICS

Translate your program using the CICS translator for HLASM. When you translate your program, CICS supplies the standard CICS prolog (DFHEIENT), which establishes addressability and saves registers in the standard CICS working storage area. The standard prolog also provides a standard CICS epilog (DFHEIRET).

Assembling Your Governor Exit

Before you assemble your governor exit routine, establish a VSE library exit to handle macro processing of E-decks. *VSE Guide to System Functions* provides a description of how to establish this exit.

Use the HLASM compiler options shown in the following example to assemble the routine. The LIBEXIT parameter includes CICS macro definitions created by the CICS translation process.

```
'LIBMAC,USING(NOLIMIT,NOWARN),EXIT(LIBEXIT(EDECKXIT(ORDER=EA)))'
```

In the source library search specification, specify the QMF governor exit interface control blocks DXEXCBA.A and DXEGOVA.A, located in the QMF sublibrary.

Link-Editing Your Governor Exit Routine

Create a new QMF governor exit phase named DSQU*n*V3, by including the EXEC CICS interface control modules DFHEAI and DFHEAI0 (located in the CICS sublibrary PRD1.BASE) and your governor exit program, DSQU*n*V3. The EXEC CICS module DFHEAI must be the first module in your governor exit phase, and the entry point must be the QMF module DSQU*n*V3.

The module DSQU*n*V3 must be executable in 31-bit addressing mode. Remember to replace the *n* symbol with an NLID from Table 3 on page 10 that corresponds to the national language you're using.

Example JCL Statements

Figure 100 on page 231 shows the JCL used to install, translate, assemble, and link-edit the IBM-supplied governor exit routine. This JCL is supplied in the QMF sublibrary, under the name DSQ3GV3.Z. For more information on installing your own program into CICS, see *CICS System Definition Guide*.

```
 ...* $$ JOB JNM=DSQ3GV3,DISP=D,CLASS=0
// JOB DSQ3GV3 Sample Job to Install Customer Written QMF Governor
*  ---------------------------------------------------------------------
*  Install QMF Governor Exit (HLASM)
*  ---------------------------------------------------------------------
// SETPARM VOLID=volid      *-- update volid for syspch
// SETPARM START=rtrk       *-- update start track/block (syspch)
// SETPARM SIZE=ntrks       *-- update number of tracks/blocks (syspch)
*  ---------------------------------------------------------------------
*  Library search chain must contain the QMF, CICS and HLASM sublibrary
*  ---------------------------------------------------------------------
// LIBDEF *,SEARCH=(PRD2.PROD,PRD1.BASE)
// LIBDEF PHASE,CATALOG=PRD2.PROD
*  ---------------------------------------------------------------------
*  Step 1: Translate Governor exit program
*  ---------------------------------------------------------------------
// DLBL   IJSYSPH,'ASM.TRANSLATION',0
// EXTENT SYSPCH,,1,0,&START.,&SIZE.
ASSGN SYSPCH,DISK,VOL=&VOLID.,SHR
// EXEC   DFHEAP1$
         :
         :
         Your governor program
         :
         :
/*
*  ---------------------------------------------------------------------
*  Step 2: Assemble Governor exit program
*  ---------------------------------------------------------------------
CLOSE SYSPCH,00D
// DLBL   IJSYSIN,'ASM.TRANSLATION',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=&VOLID.,SHR
// OPTION CATAL,DECK,SYM,ERRS
   PHASE DSQUEGV3,*,SVA
         INCLUDE DFHEAI
         INCLUDE DFHEAI0
// EXEC ASMA90,SIZE=(ASMA90,50K),                                       C
               PARM='LIBMAC,USING(NOLIMIT,NOWARN),EXIT(LIBEXIT(EDECKXITC
               (ORDER=EA)))'
CLOSE SYSIPT,SYSRDR
/*
```

Figure 100. Example JCL for translating, assembling, and link-editing a governor exit (Part 1 of 2)

```
*  ---------------------------------------------------------------------
*  Step 3: Link-edit Governor exit program
*  ---------------------------------------------------------------------
// EXEC LNKEDT,PARM='AMODE=31,RMODE=ANY'
/*
/&
// JOB  RESET
ASSGN SYSIPT,SYSRDR    IF 1A93D, CLOSE SYSIPT,SYSRDR
ASSGN SYSPCH,00D       IF 1A93D, CLOSE SYSPCH,00D
/&
 ...* $$ EOJ
```

Figure 100. Example JCL for translating, assembling, and link-editing a governor exit (Part 2 of 2)

Chapter 15. Troubleshooting and Problem Diagnosis

Use this chapter to help solve problems your users might have while using QMF. "Troubleshooting Common Problems" on page 234 provides possible solutions to common problems, while "Determining the Problem Using Diagnosis Aids" on page 240 provides explanations of diagnosis aids that help you solve more complex problems.

Quick Start

Use the steps in Table 32 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 32. 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 237
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 234
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 237
If you're getting slow response, you likely need to either reset the data object or increase your storage.	Page 238
If the problem is none of the above, determine which QMF or CICS diagnostic aids can help you further diagnose the problem.	Page 240
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 240
To determine the problem using the QMF trace facility, turn the trace on by setting the DSQSDBUG 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 DSQSDBUG parameter is either ALL or NONE. You can also specify a selected trace level just before running your trace. For example, you can specify `SET T=C2D2L2`.	Page 242
To determine the problem using CICS diagnostic facilities, identify QMF in a CICS transaction dump, using information in this book and in *CICS Problem Determination Guide*.	Page 248

Table 32. 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 250
To report a problem to IBM, use IBM's ServiceLink facility (if you have it) or call your IBM Support Center.	Page 251

Troubleshooting Common Problems

Use this section to help determine how to solve initialization errors, printing errors, warning messages on the display, incoherent report displays, and slow response times or other performance problems.

Handling Initialization Errors

If you cannot start QMF, there are several common fixes:

- Determine if all QMF users at your shop cannot get into QMF, and it is not just one user.
- Check whether there are any messages on the terminal screen, and look up the explanation for the DSQDEBUG file message in *QMF Messages and Codes*.
- If nothing appears on the screen and nothing is in DSQDEBUG, go into ISQL and issue a SELECT * FROM Q.ERROR_LOG command to see if any entries appear during the time you were trying to access QMF.
- QMF initializes DB2 and GDDM during QMF initialization. If any ARI (DB2) and ADM (GDDM) error messages appear, look them up in the messages and codes book for the appropriate product.
- Check that the DB2 database is initialized and working properly. If all users are getting a type of ADMxxxx message upon startup, check that the base GDDM product is working correctly by running the GDDM IVPs.

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 92.
- The table conforms to the proper structure:
  - The structure for command synonym tables is shown in Figure 44 on page 135
  - The structure for function key tables is shown in Figure 56 on page 148
  - The structure for the resource control table is shown in Table 28 on page 207.
- All rows of the table contain valid data. If this might be the problem, see:
  - Page135 for information on valid command synonym definitions
  - Page149 for information key definitions
  - Page207 for information on valid resource control
- 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 a transient data queue named DSQD, unless you changed the type or name using the DSQSDBQT or DSQSDBQN program parameter when you started the QMF session.

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 247. Search the trace data for the numeric portion of the message number to see information about the error.

Handling GDDM Errors During Printing

If a GDDM error occurred during printing, QMF displays this message:

```
GDDM error using nnnnnnnn.  See message help for details.
```

The character string nnnnnnnn in the message represents a GDDM printer nickname. Press the Help key to display the help panel, which contains an explanation of the error. This section discusses some common errors and what you can do to fix them.

DSQ50623

GDDM error. ADM0307 E FILE 'ADMPRINT.REQU—QUEUE' NOT FOUND. Severity 8. Function DSOPEN. * CMD=PRINT**

If you see a message like this, QMF can’t find a nickname definition for the printer name the user specified. You need to set up a nickname

definition for the printer name, or supply one that is already defined. See “Choosing a GDDM Nickname for Your Printer” on page 116 for additional information.

DSQ50623

GDDM error. ADM0314 E UNABLE TO OPEN 'MYPRINT'. DD OR DLBL STATEMENT MISSING. Severity 8. Function DSOPEN. * CMD=PRINT**

If you see a message like this, QMF was able to find a nickname but no DLBL statement that links the nickname with a physical device. You need to provide DLBL information in the CICS startup JCL. Sample JCL is shown in Figure 39 on page 122.

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 116 for information on how to create an ADMMNICK specification for a GDDM printer. For a list of valid device tokens for each family of GDDM printers, see *GDDM System Customization and Administration* for 3.1 or *GDDM Installation and System Management for VSE* for 2.3.

DSQ50631

GDDM error. ADM0904 E ALPHANUMERIC FIELDS ARE NOT SUPPORTED FOR THIS DEVICE. Severity 8. Function ASDFLD. * CMD=PRINT**

If you see a message like this, the output the user is trying to print is not valid for the type of printer defined by the GDDM nickname. Certain types of output, such as QMF charts, are restricted to specific families of GDDM printers. For more information on what families of printers handle your type of output, see *GDDM System Customization and Administration* for 3.1 or *GDDM Installation and System Management for VSE* for 2.3.

DSQ90551

GDDM error. ADM0055 E SPINIT, AT '82F810C2'X ADM0050 E DEFAULTS ERROR. INVALID SYNTAX OR VALUE AT '...JIP,ADMMNICK'

If you see a message like this when starting QMF, you made a syntax error somewhere in the ADMMNICK specification for the nickname. See “Choosing a GDDM Nickname for Your Printer” on page 116 for examples of syntax for the ADMMNICK specification. After you fix the syntax error, reload the ADMADFC GDDM defaults module.

DSQ50633

GDDM error ADM0327 E 'TD WRITEQ' ERROR CODE '08000000'X, ON 'SYSP'. Severity 8. Function FSFRCE. * CMD=PRINT**

A message like this indicates that the temporary storage or transient data queue to which QMF is attempting to print is closed. Use the CICS CEMT transaction to open the queue, specifying the file identifier SYSP.

Handling QMF Errors During Printing

The following message is a QMF error message that indicates that the object the user is trying to print needs a printer name, and QMF can't find a printer name in the user's profile or a default name.

```
GDDM printer nickname is required for PRINTER
```

Instruct users who see this message to press the Enter key to display a prompt panel on which they can enter a printer name and other print parameters. Update the user's profile with a valid printer nickname so QMF does not display this message again. "Updating User Profiles" on page 89 explains how to update the user's profile with the nickname.

"Using GDDM services to Handle Printing" on page 116 explains how to create a GDDM nickname for your printer.

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 either VS COBOL II or High-Level Assembler to convert the binary data to the character string you want. You might consider predefining some QMF forms that use the new edit codes you create. You can then make the forms available to your users. See "Chapter 13. Creating Your Own Edit Codes for QMF Forms" on page 159 for more information.

Solving Slow Performance Problems

If your users notice slow performance in running queries or formatting reports, the problem can 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
DISPLAY *tablename* (from the database)
DRAW *tablename*
EDIT TABLE
ERASE
EXPORT (from the database)
IMPORT (to the database)
LIST
PRINT (from the database)
REFRESH (of a database object list)
RESET QUERY (with LANGUAGE=PROMPTED and modifying query)
RUN (an object in the database)
RUN QUERY
SAVE (data, form, procedure, or profile)

Depending on the value specified for 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 in order 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 AMF 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.

Increasing the User's Report Storage

Users might also experience slow performance if they do not have enough virtual storage to accommodate a large report. For example, if you set the DSQSBSTG parameter at a very low value and the user runs a query that retrieves hundreds of thousands of rows, QMF retrieves the data in small amounts. 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 GETVIS storage for the user, using the DSQSBSTG parameter. This parameter is discussed in "Adjusting GETVIS Storage Used for Report Data (DSQSBSTG)" on page 56. 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 Temporary Storage (DSQSPILL)" on page 58. However, using primarily virtual storage for QMF operations provides better performance. Users who rely on a spill file and have little virtual storage might notice slow performance for large reports. Because a spill file can hold a maximum of 32 767 rows of size 4K each, setting DSQSBSTG higher ensures that QMF completes the report.

Increasing the Size of the CICS Partition

If a QMF transaction runs out of virtual storage in the CICS partition, the transaction might time out waiting for storage to become available. Ensure you size the CICS partition according to the recommended values in "QMF Storage Requirements" on page 4. These recommendations are in addition to any storage required by additional products installed.

Determining the Problem Using Diagnosis Aids

If you aren't able to solve your problem using the troubleshooting techniques discussed in "Troubleshooting Common Problems" on page 234, use this section to find out which QMF and CICS diagnosis aids can help you determine the problem.

Choosing the Right Diagnosis Aid for the Symptoms

Use Table 33 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, CICS status information, and QMF messages and help to determine the problem.

Table 33. Types of problems and the best diagnosis aids to use for them

	QMF Msg. No.	QMF Trace	CICS Trans-action Dump	CICS Status Info.	Help Message	Non-QMF Msg. No.	Error Log Output
Abend	x		x	x			
Callable interface	x	x	x	x		x	
Display panel	x	x			x	x	x
Error messages	x	x			x	x	x
Termination	x	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
Loop		x		x		x	x
Noninteractive session	x	x		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
User edit routine	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 34. 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 244.

Table 34. QMF functions and the message numbers they issue

Function	Trace ID	Message Numbers
Database Services	I	DSQ10000 - DSQ19999 DSQ30000 - DSQ39999
Dialog Command Processing	D	DSQ20000 - DSQ29999
Display Services	E	DSQ40000 - DSQ49999
Common Services and Systems Interface	C	DSQ50000 - DSQ59999
Report Formatting	F	DSQ60000 - DSQ69999
Charting	P	DSQ70000 - DSQ79999
Full-screen Windows	G	DSQ80000 - DSQ89999

In addition to the message numbers in Table 34, the following ranges of message numbers might be generated during QMF initialization:

DSQI0001 - DSQI0100
DSQ90000 - DSQ99999

Handling System Error Messages

A system error might indicate a system problem, a resource problem, or an unexpected condition. These might be problems within QMF, the database manager, or possibly some other software component. System errors are indicated by the 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 247 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.

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 FFFFFD42 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 VSE 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 Storage for Trace Data

Choose either a CICS temporary storage or transient data queue to store trace data. If the trace data for the user's session does not exceed 32 767 rows, you can use CICS temporary storage or intrapartition transient data queues to contain it. If the trace data exceeds 32 767 rows, define in the CICS DCT an extrapartition transient data queue that routes the output to a VSE file or SYSLST.

Temporary storage queues can be browsed online; however, you need to stop CICS to view the output in a transient data queue. See "Viewing QMF Trace Data" on page 247 for more information.

To define a transient data queue, update the CICS DCT with a 1-byte to 7-byte entry that points to the location that receives your trace data.

Figure 101 shows the definitions for the default queue, a transient data queue named DSQD that is allocated to a SYSLST. The default location is DSQDBUG.

```
DFHDCT TYPE=EXTRA,        QUEUE FOR QMF EXTRA PROCESSING
       DESTID=DSQD,
       RSL=PUBLIC,
       DSCNAME=DSQDBUG

DFHDCT TYPE=SDSCI,        DCT ENTRY FOR DEBUG OF QMF
       DSCNAME=DSQDBUG,
       RECFORM=VARUNB,
       BLKSIZE=136,
       TYPEFLE=OUTPUT,
       CTLCHR=ASA,
       DEVADDR=SYSLST,
       DEVICE=1403
```

Figure 101. Describing a SYSLST to contain trace data

If you want to use a temporary storage queue for the trace data, you can use the DSQSDBQT parameter when you start QMF. If you want to name the queue something other than DSQD, you can use the DSQSDBQN parameter. Both parameters are explained in "Tracing QMF Activity at the Start of a Session" on page 64.

The trace data queue can be shared by all the users in the same CICS partition, because QMF issues CICS ENQ and DEQ commands around single trace entries. Because tracing is an aspect of a user's profile, you can also set

the level of trace detail individually for each user, using the SET PROFILE command with the TRACE keyword. Records in the trace data identify individual terminal IDs for different QMF sessions on the header line.

Starting the Trace Facility

You can start the trace facility when you start QMF, from a QMF session, or by using CONNECT authority. To start the trace facility do one of the following:

- Specify a value of ALL on the DSQSDBUG program parameter when you start QMF, as explained in "Setting the Level of Trace Detail (DSQSDBUG)" on page 65. This value traces QMF activity at the highest level of detail, including program failures that might occur during QMF initialization.

  See page 66 for how to specify a type and name for the queue to hold the trace output. You need to use a transient data queue to hold the output if it exceeds 32 767 rows.
- 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".
- Use SQL UPDATE statements for the TRACE field in the user's profile, which has the same effect as the previous step. Instruct the user to reconnect to the database to initialize the new values. For example, user JONES with password MYPW can enter:

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

  Users who do not have DB2 CONNECT authority can end the current QMF session and begin another to initialize the values.

Getting the Right Level of Detail in Your Trace Output

If you want to trace all QMF functions at the most detailed level, use a value of ALL for the trace. When you start QMF, use a transient data queue (the default) to hold the trace output if it might exceed 32 767 rows. Specifying the type and name of the queue is explained in "Specifying the Type of CICS Storage for Trace Data (DSQSDBQT)" on page 66.

If you want to trace individual QMF functions, update the TRACE column of Q.PROFILES with a character string that has letters for the QMF functions you want to trace and numbers for the level of detail you want in the trace data for each function. You need to pair each letter with a number:

The value 1 traces a function at a medium level of detail.

The value 2 traces a function at the highest level of detail.

Only the functions you specify in the character string are traced. The letter for each QMF function is shown in the following list.

Trace ID	QMF Function
A	Application Support Services
C	Common Services and Systems Interface
D	Dialog Command Processing
E	Display services for parts of QMF such as Prompted Query, QBE, Table Editor, global variable lists, and database object list
F	Report formatting
G	QBE, Prompted Query, and table editor full-screen windows
I	Database services
L	Message and command logging
P	Charting (Interactive Chart Utility)
R	Storage management functions
U	User exits, such as 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 102 on page 246 shows an example of a RUN QUERY command that failed because the user named columns that were not in the table.

```
------------------------------------------------------------------------
----------       ****** 93/12/15  20:39 ******     -----------------
USERID: KRIS
AUTHORIZATION-ID: KRIS
COMMAND TEXT:
RUN QUERY
------------------------------------------------------------------------
------------------------------------------------------------------------
----------       ****** 93/12/15  20:39 ******     -----------------
USERID: KRIS
AUTHORIZATION-ID: KRIS
MESSAGE NUMBER: DSQ12405
MESSAGE TEXT:
Column name DATE is not in table STAFF.
&01:  DATE
&02:  STAFF
&09:  -205
------------------------------------------------------------------------
```

Figure 102. Using the L2 trace code to trace a user's commands and messages

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 102 appears in *QMF Messages and Codes* as:

```
Column &01 is not in table &02.
```

The bottom half of Figure 102 shows that the value for &01 in the message is DATE and that the value for &02 is STAFF. Substitute these values into the message to help a user solve the problem.

These variables might also appear in the definition of the help panels associated with the error message. Use the variable values from the trace data together with the help command to reconstruct the message help panel.

Tracing at the Module Level

Important: Perform a trace at the module level only under IBM Service Level 2 guidance.

You can turn on a trace for certain modules using the SET PROFILE command and the module DSQUTRAC. For example, you can trace the formatter buffer manager without tracing the line manager or the summary manager. The values for module-level tracing are:

The value 3 provides a detailed trace for specific programs in a component, and traces entry and exit for all other programs in the component.

The value 4 traces a module only.

To create a module-level trace, list the modules you want traced in the DSQUTRACE module. Then assemble and link-edit the module. After the module has been created, you must make it available as a phase. You can then run the following command:

```
SET PROFILE (TRACE F4
```

Viewing QMF Trace Data

Depending on the number of users and the levels of detail at which their sessions are traced, the trace data might be very long. Browse the data before you decide to print it.

Viewing Data in a Temporary Storage Queue: You can use the CICS transaction CEBR to browse a temporary storage queue. For example, to browse a queue named MYTRACE, enter the following from a cleared CICS screen:

```
CEBR MYTRACE
```

If the trace output is less than 32 767 rows, we recommend using temporary storage queues to hold the trace data. See “Specifying the Type of CICS Storage for Trace Data (DSQSDBQT)” on page 66 for information on how to specify this type of queue when you start QMF.

If the output is more than 32 767 rows, you must use a transient data queue for the trace data.

Viewing Data in a Transient Data Queue: The default queue for trace data is a transient data queue named DSQD, defined as shown in Figure 101 on page 243. Trace output routed to this queue goes to the SYSLST, and can be found in the list output of your CICS job. To transfer the data from the CICS LST queue to the SYSLST so you can view it, you need to stop CICS. Then you can browse or print the SYSLST using VSE POWER, ICCF, or another facility available to you.

If you want to view the data without bringing down CICS, consider redefining the transient data queue in the DCT so that the output goes to a file.

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

You can determine the QMF service level using the following procedure:

1. Enter the SET PROFILE command (T=ALL.
2. Enter the SET PROFILE command (T=NONE.

3. Exit QMF.
4. Look at the DSQDEBUG file.

The resulting trace shows the program with its version, date, and time. The trace can also show an Authorized Program Analysis Report (APAR) number if the module has a Program Temporary Fix (PTF) applied, as in the following trace example:

```
** DSQFQWRM: ENTERED FROM DSQFMCTL ***
   V3R3.00   96/06/30  12:00  PNxxxxx
```

APAR PNxxxxx is the most recent APAR for which service was applied.

Turning off the Trace Facility

After you capture diagnostic details using the trace facility, you might want to turn tracing off, because the storage queue for the trace data can fill up very quickly.

To turn tracing off, issue the following command from within QMF:

```
SET PROFILE (T=NONE
```

If you leave tracing on until you end the QMF session, when you start QMF the next time, the tracing is set to NONE by default. The program parameter DSQSDBUG, explained in "Setting the Level of Trace Detail (DSQSDBUG)" on page 65, controls this tracing when QMF is started.

Using CICS Diagnostic Facilities

To diagnose an abend in QMF, you might need to use procedures in *CICS Problem Determination Guide*. Because another program might have caused QMF to abend, these procedures can help you find much of the information you need in a CICS dump of the transaction. A transaction dump shows detailed activity of the programs that were running in the CICS partition at the time of the abend.

The program that caused the abend might be QMF or it might be another program. You can use the CICS Execution Diagnostic Facility (CEDF) to help you diagnose a QMF abend if the QMF diagnostic facilities explained in this chapter do not contain enough information about the cause of the error.

Identifying QMF in CICS Diagnostic Output

If you use CICS diagnostic facilities to help you diagnose an abend in QMF, the following information might help you identify QMF programs in CICS output.

- QMF program names begin with the prefix DSQ.
- QMF is an assembler-language program and issues standard assembler calls, not CICS LINK statements.

- QMF issues standard EXEC CICS statements for all system services when running in CICS.
- QMF uses an internal call interface to the GDDM product.
- QMF issues standard EXEC SQL statements to the database.
- QMF does not issue any EXEC CICS ABEND commands.

Most QMF programs contain a stamp that you can use to help identify them in diagnostic output. Figure 103 shows an example.

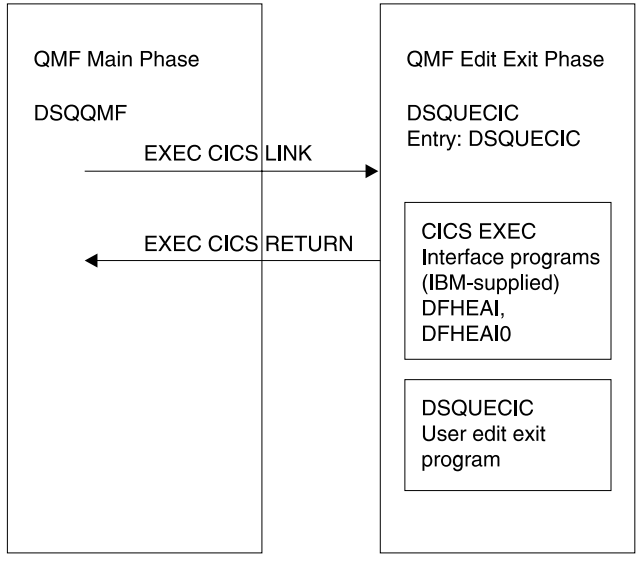


Figure 103. Example of a stamp that identifies a QMF program

Defining the Display for a CICS Abend Message

In some cases, such as if QMF abends or when the operator cancels the transaction, CICS sends a message to the user's terminal indicating the abnormal ending. Because QMF is a full-screen application that uses GDDM to provide display services, you need to define to CICS how you want the abend message should be displayed.

Using the CICS Resource Definition Online (RDO) facility, set diagnostic display attributes of the CICS error message in the CICS TYPETERM definition. A TYPETERM is a partial terminal definition that makes it easy for you to define many terminal displays with one definition. Figure 104 on page 250 shows an example of diagnostic display attributes you might use.

The definition shown in Figure 104 on page 250 displays the message at the bottom of the screen, beneath the QMF message line. The message appears in red, underlined, and with a higher intensity than the rest of the screen

display. This definition is useful if you defined the QMF transaction to time out when the user does not enter input for a certain amount of time. In this type of transaction timeout, the QMF display remains on the screen, so the message is readable only at the bottom of the screen.

```
DIAGNOSTIC DISPLAY
 ERRLastline     : Yes              No | Yes
 ERRIntensify    : Yes              No | Yes
 ERRColor        : Red              NO | Blue | Red | Pink | Green
                                    | Turquoise | Yellow | NEutral
 ERRHilight      : Underline        No | Blink | Reverse | Underline
```

Figure 104. TYPETERM specification for CICS diagnostic display

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 35.

Table 35. Structure of the Q.ERROR_LOG table

Column name	Data type	Length (bytes)	Nulls allowed?	Function/values
DATESTAMP	CHAR	8	No	The date on which the error occurred. It is in the form yyyymmdd.
TIMESTAMP	CHAR	5	No	The time at which the error occurred. It is in the form hh:mm, where hh is the hour and mm is the minute.
USER ID	CHAR	8	No	CICS terminal ID of the user who experienced the error.
MSG_NO	CHAR	8	No	The QMF message number that was issued with the error.
MSGTEXT	VARCHAR	254	No	Text of the message. SQL errors might have data from the SQLCA in this column.

A long error message might need more than one row of the table to represent it. If it does, the values of every column except the MSGTEXT column repeat. Within the MSGTEXT column, each row carries a fragment of the message. A fragment begins with 1), 2), 3), and so on, to indicate its relative position in the message.

To help diagnose problems, you can query the Q.ERROR_LOG table for information about errors. You need to know the terminal ID of the user who

experienced the problem and the approximate time the problem occurred. Figure 105 shows the format of the query.

```
SELECT TIMESTAMP, MSG_NO, MSGTEXT
  FROM Q.ERROR_LOG
  WHERE USERID = 'terminal_id'
    AND DATESTAMP = 'date'  __yyyymmdd
 AND TIMESTAMP BETWEEN 'time1' AND 'time2'
ORDER BY TIMESTAMP, MSG_NO, MSGTEXT
```

Figure 105. Querying the error log for problem information

Be sure to use valid formats for the date and times you supply. These formats are shown in Table 35 on page 250.

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" 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 254.

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 VSE/ESA 6

begins with the component ID 566872101 and a CLC number (shown in Table 36) that matches the QMF national language environment in which you experienced the problem.

Table 36. CLC numbers for QMF base product and NLFs

NLF	ID
Brazilian Portuguese	1K3
English	1JT
French	1JY
German	1JZ
Italian	1K0
Japanese	1K1

Table 36. CLC numbers for QMF base product and NLFs (continued)

NLF	ID
Korean	1K2
Simplified Chinese	1JW
Spanish	1K4
Swiss French	1K6
Swiss German	1K7
Uppercase English	1JU

The flowchart in Figure 106 on page 253 shows how to develop your search words as you determine each characteristic of the problem.

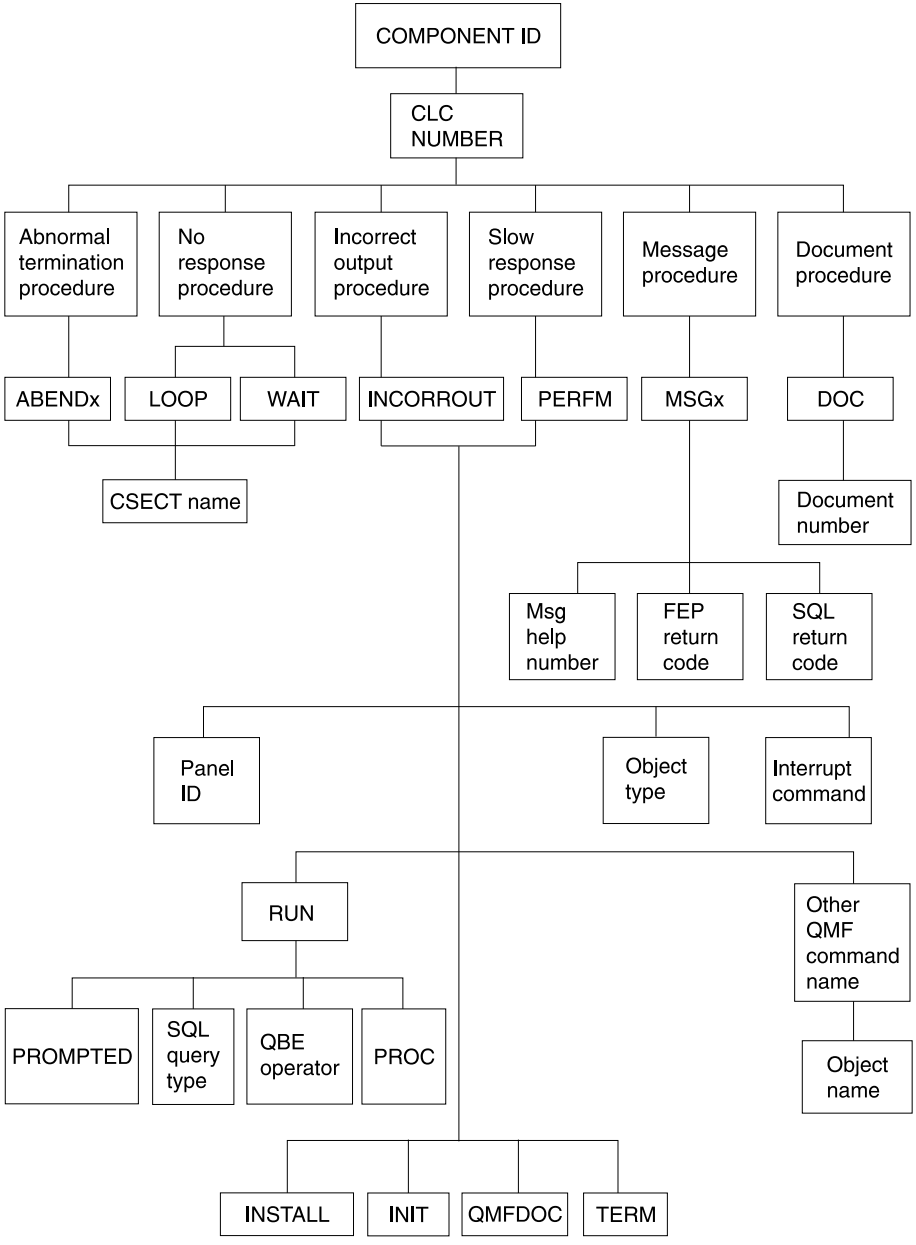


Figure 106. Chart of keyword types. Move from the top to the bottom of this chart to determine your keywords.

For example, if the problem you are searching for is an abend type of 0C4 that occurred in the DSQFDTBL control section (CSECT) when a user was running an English QMF session, use this search phrase:

```
566872101 09 ABEND0C4 DSQFDTBL
```

To find the CSECT name, look in the section of the trace output that has the heading ABEND CSECT NAME. The CSECT name is set off by asterisks. See "Using the QMF Trace Facility" on page 242 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 242. The IBM representative might also need documentation produced by other diagnosis aids shown in Table 33 on page 240. This documentation can help the representative recreate the problem.

Part 3. Appendixes

Appendix A. QMF for VSE/ESA Version 7 Product Limitations

Some functions provided by QMF are dependent on underlying system services and other program products that are available in VM/CMS and MVS/TSO, but not in CICS/VSE. ISPF is not available in CICS. REXX is not available in QMF CICS, even though REXX is available in VSE/ESA 1.3. The following QMF functions or programs are not supported in QMF for VSE/ESA Version 7.

These functions depend on ISPF (as well as other services in some cases):

ISPF command
DPRE applications
BATCH application
EXTRACT application
QMF command interface
LAYOUT command synonym

These functions depend on REXX (as well as other services in some cases):

Report calculations
Conditional formatting
Column definition
Procedures with logic

Other products are not available in CICS:

Repository Manager
Document Interface

The EDIT PROC and EDIT QUERY commands are not available in CICS. However, it is possible to edit procedures and queries using the DISPLAY command with QMF. Other products are not available in QMF for VSE/ESA Version 7:

- CMS command (VM only)
- TSO command (TSO only)
- CONNECT command (when issued to connect to another database)
- Remote unit of work and distributed unit of work

  DB2 on VSE is a server, not a requester. It can be accessed by other application requesters where QMF is installed, but QMF users on VSE cannot connect to another application server with the QMF CONNECT command.
- QMF client/server components

QMF for VSE/ESA Version 7 Product Limitations

The function of QMF on VSE is synchronized with that of QMF on other platforms. As a result, some functions that exist in QMF Version 1 are not supported in QMF for VSE/ESA Version 7:

- Command canceling.
- IMPORT ISQL queries.
- Table plot utility. To create charts, see the information on charts in *Using QMF*.
- QMF VSE V1 defaults module for starting QMF.
- VSE/POWER support (use methods supplied by CICS or GDDM to print your objects).
- QMF-supplied views.
- QMF sample queries from QMF VSE V1. QMF for VSE/ESA Version 7 does not supply sample objects except for tables. (Sample queries are not discussed in the documentation.)

It is not abnormal that, under some circumstances, a QMF Version 1 report looks slightly different in Version 7. You can make it look th e same as it did in Version 1 with a minor adjustment on the FORM.OPTIONS panel.

Appendix B. Migrating from QMF VSE V1 to Version 7

If you are migrating to QMF for VSE/ESA Version 7 from QMF VSE V1, you can use a migration utility to migrate Version 1 queries, forms, procedures, and ISQL queries. IBM provides the migration utility exclusively for the purpose of migrating objects to QMF for VSE/ESA Version 7. You don't need to customize any part of QMF to use the migration utility, because the utility is installed during QMF installation. The utility is not provided with QMF for any other versions or in any other operating environments.

After you successfully migrate Version 1 objects, you might consider deleting Version 1 from the database using the example JCL shown in this chapter.

Quick Start

Use the steps in Table 37 to help you migrate your QMF VSE V1 objects to QMF for VSE/ESA Version 7. To decide which Version 1 objects you want to migrate, you can use QMF for VSE/ESA Version 7 to view QMF objects in the Q.INTERNAL_DATA table, and ISQL queries in the SQLDBA."STORED QUERIES" table.

If you need more information on any step, see the page listed at the right of the table.

Table 37. Migrating QMF VSE V1 objects to QMF for VSE/ESA Version 7

To do this task:	See:
1. **Migrate QMF queries, forms, and procedures** using the migration utility provided by IBM. Start the migration utility using the transaction ID QMFM, and specify parameters that tell QMF which objects to migrate.	Page 260
1. **View the messages that resulted from the migration** by entering `CEBR msglog` from a cleared CICS screen, where `msglog` is the value you specified for the MSGLOG parameter during the migration.	Page 265
1. **Migrate Version 1 user profiles** by inserting values from the Version 1 Q.USERS table into the Version 7 Q.PROFILES table. Use an SQL INSERT statement.	Page 265
1. **Delete Version 1 from the VSE sublibrary after successfully migrating objects**, using the VSE librarian program.	Page 266
1. **Delete Version 1 information from the VSE history file**, using JCL statements. Examples are provided in this chapter.	Page 267

Table 37. Migrating QMF VSE V1 objects to QMF for VSE/ESA Version 7 (continued)

To do this task:	See:
1. **Delete Version 1 objects from the database**, if necessary. Use the DSQ3BDEL.Z job, shipped with QMF, to delete the objects.	Page 268
1. **Delete Version 1 QMF program definitions** from the CICS system tables.	Page 270

Migrating Queries, Forms, and Procedures

IBM supplies a migration utility that migrates QMF queries, forms, procedures, and ISQL queries from the Version 1 Q.INTERNAL_DATA and SQLDBA."STORED QUERIES" tables to the Version 7 Q.OBJECT_DIRECTORY, Q.OBJECT_REMARKS, and Q.OBJECT_DATA tables. The utility leaves the objects in the Version 1 tables intact after the migration, and you can still use these objects with Version 1 if you need to.

Table 38 shows how the columns of the Q.INTERNAL_DATA table map to the three new Version 1 control tables that store QMF objects. The Version 7 tables have the OWNER, NAME, and TYPE columns in common so that QMF is able to uniquely reference any QMF object of any type.

See "Maintaining QMF Objects Using QMF Control Tables" on page 104 for the complete structure of Q.OBJECT_DIRECTORY, Q.OBJECT_REMARKS, and Q.OBJECT_DATA, as well as more information on how QMF uses these tables.

Table 38. How the Q.INTERNAL_DATA table (V1) maps to QMF for VSE/ESA Version 7 control tables that contain objects

Version 1 column	Meaning	Migrates to this column	Of this QMF Version 7 control table
CREATOR	User ID of user who created object	OWNER	Q.OBJECT_DIRECTORY Q.OBJECT_DATA Q.OBJECT_REMARKS
NAME	Name given to object by CREATOR	NAME	Q.OBJECT_DIRECTORY Q.OBJECT_DATA Q.OBJECT_REMARKS
SEQ	Object row number	SEQ	Q.OBJECT_DATA
TYPE	QMF-assigned object type (such as query or form)	TYPE	Q.OBJECT_DIRECTORY (TYPE *and* SUBTYPE) Q.OBJECT_DATA Q.OBJECT_REMARKS

Table 38. How the Q.INTERNAL_DATA table (V1) maps to QMF for VSE/ESA Version 7 control tables that contain objects (continued)

Version 1 column	Meaning	Migrates to this column	Of this QMF Version 7 control table
SHARED	Whether an object has been shared with other QMF users. When SHARED=YES, RESTRICTED=NO. When SHARED=NO, RESTRICTED=YES.	RESTRICTED	Q.OBJECT_DIRECTORY
RELEVEL	Release level of QMF. Has value of REL 1.0 in QMF VSE V1.	RELEVEL not migrated	RELEVEL not migrated
REMARKS	Comments on the object given by CREATOR	REMARKS	Q.OBJECT_REMARKS
DATA	Internal representation of object itself	APPLDATA	Q.OBJECT_DATA

The DATA field of the Q.INTERNAL_DATA table is 3600 bytes long. If the internal representation of the Version 7 object is more than 3600 bytes, more than one row in Q.OBJECT_DATA is used to store the object. The APPLDATA column contains the continuation of the internal representation of the object. The SEQ column is a number that represents each additional APPLDATA field.

If you migrate data into the QMF for VSE/ESA Version 7 Q.OBJECT_DIRECTORY table from the QMF VSE V1 table, the new timestamp columns in the Q.OBJECT_DIRECTORY become nulls in your old object.

Starting the Migration Utility

You can start the migration utility three ways:

- Enter QMFM from a cleared CICS screen, followed by values for the migration utility parameters. These are explained beginning with “Specifying the Type of Object” on page 262.

  For example, the following command migrates a query named MYQUERY, owned by user JONES. It also specifies that the Version 1 object should not be migrated if a Version 7 object with the same name exists.

  ```
  QMFM TYPE(QUERIES) CREATOR(JONES) NAME(MYQUERY)
       REPLACE(NO) MSGLOG(MYMSGS)
  ```

  After the migration, user JONES can use the command CEBR MYMSGS to view the messages from the migration.

- Enter `CICS QMFM...` from the QMF Home panel, where "..." represents the migration utility parameters and values. The Home panel command line has a limit of 55 characters of input.
- Write a CICS application that includes the following command:

  ```
  EXEC CICS START TRANSID('QMFM') FROM('...')
  ```

  where "..." represents values for the migration utility parameters.

If you are migrating QMF VSE V1 objects that have uppercase names, you can use the QMFM transaction from a cleared CICS screen.

If you need to migrate QMF VSE V1 objects that have been saved in lower or mixed case, you need to start the migration utility from the QMF Home panel (using the CICS command) or from within a CICS application using the EXEC CICS START command.

Follow these guidelines when you use the migration utility parameters:

- Use uppercase if you are migrating objects from the QMF Home panel or from a CICS application. CICS converts all input entered from a cleared CICS screen to uppercase.
- Use each parameter only once.
- Enclose each value you supply in parentheses.
- Always supply a value for the TYPE and MSGLOG parameters. These parameters are required. If you omit values for the other parameters, defaults are used as explained in this chapter.
- Ensure the values you specify for CREATOR and NAME are identical to the Version 1 names, because the CREATOR and NAME parameters are case sensitive.

Specifying the Type of Object

Use the TYPE parameter to specify the type of object you want to migrate. The migration utility migrates QMF queries, forms, procedures, and ISQL queries. The following values are valid for the TYPE parameter:

If you use this value for TYPE:
The migration utility migrates:

ALL All QMF objects

QUERIES
QBE and SQL queries

FORMS
QMF forms

PROCS
QMF procedures

ISQLQUERIES
ISQL queries

The value ALL migrates only QMF queries, forms, and procedures. Ensure you specify ALL by itself on the TYPE parameter. For example, QMF displays an error message if you enter the following:

```
QMFM TYPE(PROCS,ALL) MSGLOG(MYMSGS)
```

Other values can appear together as options on the same TYPE parameter, provided the values are separated by commas. For example, the following migrates all Version 1 queries and forms, regardless of their owners and names:

```
QMFM TYPE(QUERIES,FORMS) MSGLOG(MYMSGS)
```

After you migrate QMF Version 1 forms, specify YES in answer to the AUTOMATIC REORDERING? question on the Form.Options panel if you want the Version 7 form to look identical to Version 1. QMF Version 1 provided automatic reordering as the default, which ordered all aggregation summary columns on the right and all break columns on the left. Automatic reordering is not the default in Version 7.

The value ISQLQUERIES migrates Version 1 ISQL queries to Version 7 SQL queries. Ensure you specify the value ISQLQUERIES by itself on the TYPE parameter.

ISQL queries are stored in the SQLDBA."STORED QUERIES" table in Version 1. The STMTNAME, CREATOR, and STMTTEXT columns of this Version 1 table are migrated to the NAME, OWNER, and APPLDATA columns of Version 7 QMF control tables. Format data associated with the STMTTEXT column is not migrated.

After the migration, the TYPE column is set to QUERY, and the SUBTYPE column is set to SQL. The SEQ column of Q.OBJECT_DATA is used if necessary. See Table 38 on page 260 for descriptions of how Version 1 table columns map to Version 7.

Specifying the Owner of the Object

Use the CREATOR parameter to migrate objects by Version 1 user ID. If you omit the CREATOR parameter, all objects that meet the criteria you specified for other parameters (NAME and TYPE, for example) are migrated.

You can use the CREATOR parameter with the TYPE option to migrate objects by user ID, regardless of their names. For example, the following command migrates all QMF queries for user JONES:

```
QMFM TYPE(QUERIES) CREATOR(JONES) MSGLOG(MYMSGS)
```

Ensure the value you use for CREATOR is 8 characters or less. Also ensure that you surround the entire value in double quotes if it contains blanks. Double quotes are not allowed as part of the CREATOR value itself.

Characters you normally use in the LIKE predicate of an SQL statement can be used to represent one or more characters in the CREATOR value. These characters are the underscore (_) and the percent sign (%). For example, the following command migrates all objects for user IDs that start with S:

```
QMF TYPE(ALL) CREATOR(S%) MSGLOG(MYMSGS)
```

The migration utility always treats both the percent sign and the underscore as characters in an SQL LIKE predicate. If these characters are part of the user ID whose objects you need to migrate, use the NAME and other parameters to migrate that user's objects.

Specifying the Name of the Object

Use the NAME parameter to migrate specific objects by name. If you omit the NAME parameter, all objects that meet the criteria you specified for other parameters (CREATOR and TYPE, for example) are migrated.

You can use the NAME parameter with the TYPE option to migrate a specific set of objects, regardless of who created them. For example, the following command migrates all QMF queries named SAMPLE, regardless of who created them:

```
QMFM TYPE(QUERIES) NAME(SAMPLE) MSGLOG(MYMSGS)
```

For QMF object names, ensure the value you use for NAME is 18 characters or less. For ISQL queries, ensure the value is 8 characters or less.

Also ensure that you surround the entire value in double quotes if it contains embedded blanks. Trailing blanks are processed as part of the name. For example, "MYQUERY" and "MYQUERY" are processed as two different names.

See "Specifying the Owner of the Object" on page 263 for how to use characters you normally use in the LIKE predicate of an SQL statement.

Migrating Version 1 Objects to Version 7 Control Tables

If an object you are migrating has the same values for OWNER, NAME, and TYPE as a QMF for VSE/ESA Version 7 object, the REPLACE parameter indicates whether the Version 1 object replaces the Version 7 object.

Use YES to indicate that you want QMF to replace the Version 7 object upon finding a match. For example, the following command replaces all Version 7 QMF queries that begin with SAMPLE:

```
QMFM TYPE(QUERIES) NAME(SAMPLE%) REPLACE(YES)
```

If you omit the REPLACE parameter, QMF assumes a value of NO.

Viewing Messages from the Migration

Use the MSGLOG parameter to name the CICS temporary storage queue that holds the messages from the migration. Ensure the name you use is 8 characters or less and that no other queue by that name exists.

For example, the following command migrates all QMF objects and stores the messages generated from the migration in a temporary storage queue named MYMSGS:

```
QMFM TYPE(ALL) MSGLOG(MYMSGS)
```

You can use the CICS CEBR transaction or any other facility appropriate for reading a temporary storage queue. If you plan to use the CEBR transaction to view the messages, ensure you specify an uppercase name for the storage queue that holds the message log, because the CEBR transaction folds terminal input to uppercase.

Migrating User Profiles

To migrate user profiles from the Version 1 Q.USERS table to the Version 7 Q.PROFILES table, run an SQL query similar to the one in Figure 107. Specify for the value of the &CREATOR variable the SQL authorization ID of the user whose profile you're migrating. To copy all QMF VSE V1 profiles to QMF for VSE/ESA Version 7, use a percent sign (%) for the value of the &CREATOR variable.

```
INSERT INTO Q.PROFILES (CREATOR,CASE,DECOPT,CONFIRM,WIDTH,LENGTH,
                        LANGUAGE,SPACE,TRACE,TRANSLATION)
    SELECT CREATOR,CASE,DECOPT,CONFIRM,WIDTH,LENGTH,LANGUAGE,SPACE,
         TRACE,'ENGLISH')
    FROM Q.USERS
         WHERE CREATOR LIKE &CREATOR
```

Figure 107. Query that migrates QMF VSE V1 user profiles from Q.USERS table to QMF for VSE/ESA Version 7 Q.PROFILES table

So that Version 7 profiles do not get overwritten, the query fails if you attempt to insert a row that has the same CREATOR value as a row already in Q.PROFILES.

The CREATOR and TRANSLATION columns do not allow null values. The TRANSLATION column does not exist in Q.USERS and gets a value of ENGLISH for rows inserted from Q.USERS.

If you're using an NLF: Run an SQL UPDATE query after migrating the profiles to change the value of the TRANSLATION column.

Columns not in Q.USERS are automatically given null values when the profiles are migrated. Because Q.USERS has a separate control table for printer names (Q.PRINTER), the PRINTER column in Q.PROFILES is also given a null value. See Table 9 on page 85 for information about the Q.PROFILES table.

Deleting QMF VSE V1 After You Migrate Your Objects

When you are certain you successfully migrated all Version 1 objects to Version 7, consider deleting QMF VSE V1 to make more storage available on your system. You need to perform four steps, each explained in this section:

1. Delete the QMF VSE V1 objects from the VSE sublibrary.
2. Delete information about QMF VSE V1 from the history file.
3. Remove QMF VSE V1 objects from VSE DB2.
4. Remove QMF VSE V1 definitions from the CICS system tables (for example, the PPT and PCT).

Each of these steps is described in this section.

Attention: Before you begin this procedure, ensure you migrated all Version 1 objects successfully. Check the migration utility message log to make sure there were no errors. See “Viewing Messages from the Migration” on page 265 for more information about how to view the message log.

Deleting QMF VSE V1 from the VSE Sublibrary

Use the VSE librarian program to delete QMF VSE V1 from the VSE sublibrary. Run a job similar to the following:

```
      . . .
      . . .
      . . .

EXEC LIBR,PARM='MSHP'
ACC S=library.sublibrary
DELETE DSQ*.*
DELETE $$$COQMF.OBJ
DELETE HD292F67.Z
```

Figure 108. Job that deletes QMF VSE V1 from the VSE sublibrary

Enter for **library.sublibrary** the library and sublibrary where QMF VSE V1 is installed.

If you installed the QMF VSE V1 Uppercase Feature (UCF), ensure you add the following as the last line of the job:

```
DELETE HD292A16.Z
```

Deleting QMF VSE V1 Information from the History File

How you delete QMF VSE V1 from the history file depends on how you installed QMF VSE V1. Retrace your history file to find out which of the following configurations you have for QMF VSE V1:

- Only the Version 1 base product is installed.

  In this case, run a job similar to the one shown in Figure 109 to delete the history file entries.

```
// EXEC MSHP,SIZE=900K
 REMOVE 292F67            /* QMF REL 1.0 BASE   */
 REMOVE 5666-292-01-F67
/*
// EXEC DTRIPST,SIZE=500K
/*
/&
```

Figure 109. Job that deletes QMF VSE V1 history file entries for base QMF

- The base product and UCF are installed to the same sublibrary.

  In this case, run a job similar to the one shown in Figure 110 to delete the history file entries.

```
// EXEC MSHP,SIZE=900K
 REMOVE 292A16            /* QMF REL 1.0 UPPER CASE ENGLISH FEATURE */
 REMOVE 5666-292-01-A16
/*
// EXEC DTRIPST,SIZE=500K
/*
/&
```

Figure 110. Job that deletes QMF VSE V1 history file entries for QMF base and UCF (same sublibrary)

- The base product and UCF are installed to different sublibraries.

  In this case, run a job similar to the one shown in Figure 111 on page 268 to delete the history file entries.

```
// EXEC MSHP,SIZE=900K
 REMOVE 292F67            /* QMF REL 1.0 BASE   */
 REMOVE 5666-292-01-F67
 REMOVE 292A16            /* QMF REL 1.0 UPPER CASE ENGLISH FEATURE */
 REMOVE 5666-292-01-A16
/*
// EXEC DTRIPST,SIZE=500K
/*
/&
```

Figure 111. Job that deletes QMF VSE V1 history file entries for QMF base and UCF (different sublibrary)

Deleting QMF VSE V1 Objects from the VSE DB2 Database

IBM ships a job named DSQ3BDEL.Z with QMF for VSE/ESA Version 7. This job removes Version 1 QMF objects (queries, forms, and procedures) stored in the VSE DB2 database.

Check the target DB2 database where QMF VSE V1 is installed. The target database name is specified on the DBNAME parameter:

```
// SETPARM DBNAME=SQLDS
```

The default name for the target database is **SQLDS**. If you installed QMF VSE V1 into a different database, run DSQ3BDEL with a different database name.

Figure 112 on page 269 shows the DSQ3BDEL job.

```
* $$ JOB JNM=DSQ3BDEL,DISP=D,CLASS=0
// JOB DSQ3BDEL  DROP QMF V1 OBJECTS FROM DB2
// LIBDEF *,SEARCH=PRD2.DB2610
/. C  * ------------------------------------------------------- *
/. C  *                 ** CAUTION **                           *
/. C  * THIS JOB DROPS ALL QMF V1 OBJECTS FROM THE DB2         *
/. C  * DATABASE, INCLUDING PROCS, QUERIES, AND FORMS.  RUNNING    *
/. C  * THIS JOB IS OPTIONAL AFTER SUCCESSFUL MIGRATION OF QMF     *
/. C  * V1 OBJECTS TO QMF V7R1M0.                                 *
/. C  *                                                         *
/. C  * NOTE:  APPLICATION SERVER MUST BE UP IN MULTIPLE USER MODE *
/. C  * ------------------------------------------------------- *
// SETPARM DBNAME=SQLDS   *-- TARGET DB2 DBNAME FOR THIS JOB
// PAUSE *CAUTION* HIT ENTER TO DROP QMF V1 OBJECTS FROM DB2
// EXEC ARIDBS,SIZE=AUTO,PARM='DBNAME(&DBNAME)'
CONNECT Q IDENTIFIED BY QMF;
SET AUTOCOMMIT OFF;
DROP PROGRAM DSQIBOR ;
DROP PROGRAM DSQICNCT;
DROP PROGRAM DSQICSQL;
DROP PROGRAM DSQIDTVQ;
DROP PROGRAM DSQIESQL;
DROP PROGRAM DSQIFSQL;
DROP PROGRAM DSQIICVS;
DROP PROGRAM DSQIIPEL;
DROP PROGRAM DSQILD  ;
DROP PROGRAM DSQIMIG ;
DROP PROGRAM DSQIPR  ;
DROP PROGRAM DSQIPRNT;
DROP PROGRAM DSQISDTA;
DROP PROGRAM DSQISV  ;
DROP PROGRAM DSQIUPRF;
```

Figure 112. Job that deletes QMF VSE V1 objects from VSE DB2 (DSQ3BDEL) (Part 1 of 2)

```
DROP VIEW Q.AUTHS;
DROP VIEW Q.COLUMNS;
DROP VIEW Q.FORMS;
DROP VIEW Q.ISQLQUERIES;
DROP VIEW Q.PROCS;
DROP VIEW Q.QUERIES;
DROP VIEW Q.TABLES;
DROP TABLE Q.INTERNAL_DATA;
DROP TABLE Q.PRINTER;
DROP TABLE Q.USERS;
DROP DBSPACE CONTROL_TABLE1;
COMMIT WORK;
/*
/&
* $$ EOJ
```

Figure 112. Job that deletes QMF VSE V1 objects from VSE DB2 (DSQ3BDEL) (Part 2 of 2)

Deleting QMF VSE V1 Definitions from the CICS System Tables

CICS definitions for QMF VSE V1 programs and transactions are in the PPT, the PCT, or the CSD:

- If the definitions are in the PPT or PCT, remove them and reassemble the tables.
- If the definitions are in the CSD, remove the entries for the programs and transactions.

For more information on how to remove definitions and reassemble the tables, see *CICS/VSE Resource Definition (Online).*

Appendix C. How QMF and GDDM Programs Are Defined to CICS

Installing and Managing QMF on VSE/ESA provides the jobs necessary to define QMF programs to CICS and load GDDM definitions and chart formats for QMF panels. If you need to modify the default installation, use this section to find out how QMF programs are defined and how GDDM definitions are loaded during QMF installation.

How QMF Programs Are Defined to CICS/VSE

During QMF installation, the default transaction ID QMFn is defined for QMF, where n is a national language identifier from Table 3 on page 10. The transaction ID is defined in either the CICS program control table (PCT) or the system definition (CSD) file. If you need to, you can change this default transaction ID:

- To update the CSD, see *CICS/VSE Resource Definition (Online)*
- To update the PCT, see *CICS/VSE Resource Definition (Macro)*

Resident QMF Programs

During QMF installation, the following programs are defined as resident in CICS:

DSQQMF
DSQQMFn
DSQCBST
DSQCnLTT
DSQCnBLT

CICS/VSE treats programs with RMODE(ANY) as permanently resident, because of the large amount of virtual storage available above the 16MB line. Programs defined as resident are loaded during CICS system initialization. Nonresident programs are loaded on the first reference to the program.

The first QMF transaction to start causes certain GDDM programs to be loaded. See “How Nonresident GDDM Programs Affect QMF” on page 273 for more information.

How Nonresident Programs Affect Performance

If several users use QMF, removing QMF programs from resident storage might affect QMF and CICS performance, because QMF must be loaded each time a user starts the program. However, if the needs of your installation require that you remove these programs from resident storage, change the definition for QMF programs from resident to nonresident.

You can specify RESIDENT=NO on the CEDA DEFINE PROGRAM command to interactively change the program definition in the CSD, or specify RES=NO on the DFHPPT TYPE=ENTRY macro to change the value in the program processing table (PPT).

For more information on the performance implications of nonresident programs, see *CICS for VSE/ESA 2.3 Performance Guide*

Loading QMF to the 31-Bit Shared Virtual Area

The default installation loads QMF to an individual CICS partition. Depending on the configuration of your system, you might consider loading QMF programs to the 31-bit VSE shared virtual area (SVA).

If several CICS partitions run QMF, consider loading QMF to the SVA rather than to the resident area in the individual CICS partitions where QMF is running. Loading QMF to the SVA:

- Allows two or more CICS systems in the same processor to share QMF programs. The CICS systems do not have to be using intercommunication facilities to benefit from sharing programs.
- Automatically protects the programs from being overwritten by other programs, such as CICS applications. This integrity also applies to a single CICS system within the processor.

If you decide to load programs to the SVA, you need to use the SVA command to allocate space in the SVA for QMF modules and their system directory list (SDL) entries at IPL time. The space you allocate is in addition to any required for other phases in the SVA.

The following QMF base programs can be loaded to the SVA; they take approximately 2.8 MB of space:

DSQQMF
: Main QMF program

DSQCBST
: Driver for the callable interface

DSQQMFE
: Identifies the environment and language being started

DSQCELTT
: Holds messages and constants for QMF objects and screen displays

DSQCEBLT
: Holds command definitions and permits bilingual support

DSQUEGV3
: Required for the governor exit routine (discussed in Chapter 14)

DSQUECIC
Required for user-written edit exit routines (discussed in Chapter 13)

If you're using an NLF: You can also load the following QMF NLF programs to the SVA. These programs require a total of approximately 300 KB in the SVA, per NLF. The n symbol represents an NLID from Table 3 on page 10.

DSQQMFn
DSQCnLTT
DSQCnBLT
DSQUnGV3

To load programs into the SVA, issue a SET SDL command from the background (BG) partition, naming the selected programs. You can issue this command at any time after IPL, but you must issue it before bringing up any CICS system that uses programs from the SVA.

VSE loads SVA phases from the libraries of the BG partition library search chain. You need to specify the QMF library in the SEARCH operand of the LIBDEF statement for the BG partition. Figure 113 shows an example of a load list for QMF programs:

```
// JOB CICS SVALOAD
*  CICS/VSE/ESA SVA LOAD LIST
*  LIBDEF statement for the QMF sublibrary
SET SDL
DSQQMFE,SVA
DSQCBST,SVA
DSQCELTT,SVA
   . . .
   . . .
   . . .
/*
/&
```

Figure 113. Loading QMF programs to the SVA

How GDDM Definitions Are Loaded During QMF Installation

QMF for VSE/ESA Version 7 uses GDDM services for printing and displaying QMF screens. The VSAM panel file DSQPNLn contains text for QMF screens and is described to CICS during QMF installation. QMF also uses the GDDM-PGF product to create charts of many types, such as scatter, pie, histogram, and others.

How Nonresident GDDM Programs Affect QMF

GDDM programs are not predefined as resident. When you tailor GDDM for CICS, consider making GDDM programs resident, because certain GDDM programs are loaded when QMF is started, whether or not you use QMF's

charting functions. See the *CICS/VSE Performance Guide* for more information on how to decide which programs should be resident. For more information on tailoring GDDM for CICS, see:

GDDM System Customization and Administration for GDDM 3.1
GDDM Installation and System Management for VSE for GDDM 2.3

How Chart Formats Are Defined

The QMF default installation stores chart formats, chart data, and GDF data in the GDDM file ADMF. You can change the name of this GDDM object file or create additional GDDM object files to store chart objects by modifying the OBJFILE section of the GDDM external defaults module, ADMADFC. For example, you might have separate files for chart formats, chart data, and GDF data.

Adding Charting Function After QMF Installation

If you install GDDM-PGF after you install QMF, you need to fully install and tailor GDDM-PGF for CICS, rather than merely restoring the product to a sublibrary.

If you use GDDM 3.2, you need to install GDDM-PGF 2.1.3.
If you use GDDM 2.3, you need GDDM-PGF 2.1.1.

After you install GDDM-PGF and tailor it, you can verify the installation by running the CICS ADMC transaction, which is predefined by GDDM during GDDM tailoring for CICS. No further customization of the chart formats is necessary; these formats were defined for you during QMF installation.

Using Transaction Routing to Control Resource Use

To protect high-speed transactions in your system from potential long-running QMF queries that might consume extra resources, consider isolating execution of QMF transactions to a single partition, using multiregion operations or intersystem communications. Define one CICS terminal-owning partition and route QMF transaction requests to other partitions by using multiple transaction IDs or dynamic routing exits. Both methods are described in *CICS for VSE/ESA 2.3 Intercommunication Guide*

See “Customizing Report Storage and Report Performance” on page 56 for information on how QMF uses GETVIS storage in the CICS partition.

Appendix D. QMF Control Tables and dbspaces Used by QMF

QMF uses the control tables shown in Table 39 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 39. 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 82 to 92
Q.OBJECT_DIRECTORY	DSQTSCT1	256 pages	Contains general information about all QMF queries, forms, and procedures in the database.	Page 104
Q.OBJECT_DATA	DSQTSCT3	5120 pages	Contains queries, forms, and procedures represented in an internal QMF format.	Page 105
Q.OBJECT_REMARKS	DSQTSCT2	256 pages	Contains comments that were saved when queries, forms, and procedures were created (or replaced).	Page 106
Q.RESOURCE_TABLE	DSQTSGOV	128 pages	Contains resource control information passed to the governor exit routine.	Page 199
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 250

Table 39. 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. **Important: Do not modify this table.**	This table is not discussed in this book.

In addition to the dbspaces shown in Table 39 on page 275 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 and the SAVE DATA or IMPORT TABLE commands, see *Using QMF*.

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
Advanced Peer-to-Peer Networking
AIX
AIX/6000
AS/400
C/370
CICS
CICS/ESA
CICS/MVS
CICS/VSE
COBOL/370
DATABASE 2
DataJoiner
DB2
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
DXT
GDDM
IBM
IBMLink
IMS
Language Environment
MVS
MVS/ESA
MVS/XA
OfficeVision/VM
OS/2
OS/390
PL/I
PROFS
QMF
RACF
S/390
SQL/DS
Virtual Machine/Enterprise Systems Architecture
Visual Basic
VM/XA
VM/ESA
VSE/ESA
VTAM

Java or all Java-based trademarks and logos, and Solaris are trademarks of Sun Microsystems, Inc. in the United States, other counrtries, or both.

Lotus and 1-2-3 are trademarks of Lotus Development Corporation in the Unites States, other counrties, or both.

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

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 ProgrammerÆs 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*

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 Uitility 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

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: UserÆs 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

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

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.

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.

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.

IXF. Integration Exchange Format: A protocol for transferring tabular data among various software products.

JCL. Job control language for OS/390.

job control. In VSE, a program called into storage to prepare each job or job step to be run. Some of its functions are to assign I/O devices to symbolic names, set switches for program use, log (or print) job control statements, and fetch the first phase of each job step.

JIS (Japanese Industrial Standard) format. A format that represents date and time values as follows:
- Date: yyyy-mm-dd
- Time: hh:mm:ss

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*.

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.

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 priviledge for the Q.PROFILES control table.

QMF administrator. A QMF user with QMF administrative authority.

QMF command. Refers to any command that is part of the QMF language. Does **not** include installation-defined commands.

QMF session. All interactions between the user and QMF from the time the user invokes QMF until the EXIT command is issued.

qualifier. When referring to a QMF object, the part of the name that identifies the owner. When referring to a TSO data set, any part of the name that is separated from the rest of the name by periods. For example, ‘TCK’, ‘XYZ’, and ‘QUERY’ are all qualifiers in the data set name ‘TCK.XYZ.QUERY’.

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.

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.

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”.

Index

A

abbreviating command synonyms 142
abbreviations
  for command synonyms 142
abend 34
abend during installation 34
access
  to objects
    controlling 92
    database object list, customizing 97
    queries, forms, procedures 95
    SQL GRANT statement 94
    SQL REVOKE statement 95
  to QMF
    connecting to DB2 68
    enabling 82
    restricting 84
ACQUIRE dbspace statement 102
add
  new products 37
adding new products 37
address, governor function calls 213
ADMADFC defaults module 120, 274
ADMF file 104
administration
  acquiring dbspace 102
  DB2 system tables 110
  granting and revoking privileges 94
  listing user's tables/views 110
  object
    controlling access 95
    deleting 108
    displaying user's 107
    listing user's 107
    transferring ownership 108, 111
  Q user profile 82
  resources 82
  tables, creating 100
  user profiles and objects 91
ADMMNICK specification 118
ALL keyword 65
ALL keyword 65 *(continued)*
  DSQSDBUG program parameter 65
ALLOC parameter, CICS 5
ALLOC.PROC data set 7
ALTER DBSPACE statement 110
ALTER statement
  DBSPACE keyword 110
ampersand (&)
  in command synonyms 139
ampersands in command synonyms 139
ANSI support for printing 123
APAR (Authorized Program Analysis Report) 249, 251
  description 249
APPLDATA column 106
application
  CICS, starting QMF 51
applying PTFs 39
ASCPUT services, printing 123
assembler
  compiling HLASM edit routine 173
assembling HLASM edit routine 173
asynchronous processing, printing 116
authorization
  cascading 95
  command synonyms 141, 144
  creating tables 100
  DBA, user Q 82
  implicit connection to DB2 68
  messages 92
  RESOURCE 102
  to access QMF 82
automatic routing, print output 115
auxiliary storage 58

B

B parameter 57
base QMF commands as synonyms 136
batch
  starting QMF in 52
  updating CSD in 22
batch QMF session 69
bilingual support
  forms 111
bilingual support, QMF forms 111
binary data
  in reports 237
binary data in reports 237
blanks, restrictions for user ID and procedure syntax 70
blocks, library 6
BOTTOM command 63
branch addresses, governor 213
Brazilian Portuguese NLF 11

C

calculating
  spill file space 59
Canadian French NLF 11
canceling
  governor 228
cancellation service, governor 228
cascading authority 95
case, setting 85
CASE column (Q.PROFILES) 85
CEBR transaction
  viewing migration messages 265
  viewing trace data 247
CEDA 24
chart
  format 274
  printing
    GDDM vs QMF 115
charts
  formats 274
  printing 114, 115
    specific objects 114
  support 104
CHARVAL column, Q.RESOURCE_TABLE 202
Chinese NLF 11
CICS
  application, starting QMF 51
  calculating storage for QMF 56
  command interface modules, edit routine 168
  connecting to DB2 49
  CSD modification 23
  customization 11, 21
  DB2 connection 49
  defining edit routine
    COBOL 186
    HLASM 175
  defining GDDM printers 121

CICS *(continued)*
deleting QMF V1 270
destination control table (DCT) 121
DFHEISTG macro 169
ENVIRONMENT values, QMF profile 88
installing additional 26
interface to governor 209
location of QMF programs 273
partition, defining storage 5
partition size 7
prolog macro, edit routine 168
QMF CICS command
command synonym 137
starting migration utility 262
QMFE transaction 32
startup job 25
tailoring 21
temporary storage 123
terminal operator ID 81
transient data queue 123
TYPETERM entries, QMF display 249

CICS (Customer Information Control System) 5
application, starting QMF 51
calculating storage for QMF 56
COBOL edit routine 184
command interface modules, edit routine 168
connecting to DB2 49
DB2 connection 49
defining edit routine 175
COBOL 186
HLASM 175
defining GDDM printers 121
deleting QMF V1 270
destination control table (DCT) 121
defining Family 3 printer 121
entries for trace data queue 243
DFHEISTG macro 169
diagnostic facilities 248
ENVIRONMENT values, QMF profile 88
HLASM edit routine 173
interface control block 208
interface to governor 209
JCL to define GDDM printer 121
location of QMF programs 273
multiple transaction IDs 271

CICS (Customer Information Control System) 5 *(continued)*
partition, defining storage 5
prolog macro, edit routine 168
QMF CICS command 137
command synonym 137
starting migration utility 262
starting QMF 51
storage 56
support for QMF 271
temporary storage queue 56
printing using QMF services 123
reports 56, 58
trace data 66, 247
terminal control table (TCT), defining printers 118
terminal operator ID 81
trace storage 66
transient data queue 66
printing using QMF services 123
routing output 115
trace data 66, 247
TYPETERM entries, QMF display 249
CIRB transaction 49
class ID, customizing function keys 150
CLC numbers 251
COBOL 175
COBOL edit routine
defining to CICS 186
command 87
BOTTOM 63
cancellation messages 228
cancellation service 228
CICS, synonym definition 137
CONNECT 67
DSQSUSER parameter 67
customizing 133
EXEC CICS statement
LINK command 168
RETURN command 169
START command 51
EXIT in procedures 72
function keys, assigning 145
governor exit routine calls 211
interface initialization messages 234
interface modules (CICS), edit routine 168
PRINT 113
privileges required 92
program parameters 76

command 87 *(continued)*
RESET DATA 63
RUN
initial procedure 69
synonym definition 137
SAVE
DATA keyword 63
SHARE parameter 95
SET PROFILE 89
starting QMF 51
synonyms 133
window IDs 153
command synonyms table
creating 134
maintaining 142
views 143
comment
on function keys table 148
on synonyms table 134
comments
on function keys table 148
on synonyms table 134
compiler options
COBOL edit routine 184
governor exit routine 230
HLASM edit routine 173
CONFIRM column (Q.PROFILES) 85
confirmation panel
displaying 85
table editor 103
confirmation panels
displaying 85
table editor 103
CONNECT authority, granting 81
CONNECT command
errors 234
QMF and DB2 68
connecting DB2 to CICS 49
connection to the database
VSE DB2 to CICS/VSE 49
console 3
control programs, edit routine 163
control section (CSECT), diagnosis 253
control tables 83
dbspace names/sizes 275
ownership 82
Q.ERROR_LOG 250
Q.OBJECT_DATA 105
Q.OBJECT_DIRECTORY 104
Q.OBJECT_REMARKS 106
Q.PROFILES 84
Q.RESOURCE_TABLE 202
Q.RESOURCE_VIEW 203

copybooks 12
CREATE TABLE statement
  command synonyms 134
  customized function keys 148
  privileges for SAVE DATA 93
  resource control table 206
  tables for users 100
CREATOR column (Q.PROFILES)
  defined 85
  role in profile initialization 89
CREATOR parameter,
  migration 263
CSD data set 12, 23
CSECT (control section),
  diagnosis 253
cursor stability 96
Customer Information Control
  System (CICS) 5
  application, starting QMF 51
  calculating storage for QMF 56
  COBOL edit routine 184
  command interface modules, edit
    routine 168
  connecting to DB2 49
  DB2 connection 49
  defining edit routine 175
    COBOL 186
    HLASM 175
  defining GDDM printers 121
  deleting QMF V1 270
  destination control table
    (DCT) 121
    defining Family 3
      printer 121
    entries for trace data
      queue 243
  DFHEISTG macro 169
  diagnostic facilities 248
  ENVIRONMENT values, QMF
    profile 88
  HLASM edit routine 173
  interface control block 208
  interface to governor 209
  JCL to define GDDM
    printer 121
  location of QMF programs 273
  multiple transaction IDs 271
  partition, defining storage 5
  prolog macro, edit routine 168
  QMF CICS command 137
    command synonym 137
    starting migration utility 262
  starting QMF 51
  storage 56
  support for QMF 271
Customer Information Control
  System (CICS) 5 *(continued)*
  temporary storage queue 56
    printing using QMF
      services 123
    reports 56, 58
    trace data 66, 247
  terminal control table (TCT),
    defining printers 118
  terminal operator ID 81
  trace storage 66
  transient data queue 66
    printing using QMF
      services 123
    routing output 115
    trace data 66, 247
  TYPETERM entries, QMF
    display 249
customizing
  additional database 18
  CICS 21
  function keys 145
  object lists 99
  QMF 13, 26
  QMF commands 133
  QMF environment 13, 26
  QMF session behavior
    using QMF program
      parameters 51, 75
    using user profile 82

D

data formats 159
data object
  BOTTOM command 63
  limiting initial rows retrieved 63
  performance 238
  privileges for SAVE DATA 93
  RESET DATA command 63
  retrieval 59
  SAVE command 63
database
  confirming changes 103
  connection
    authority 82
    CICS and DB2 49
    QMF to DB2 68
  object list, customizing 97
  objects
    access 92
    granting privileges 94
    ownership 94
    revoking privileges 94
    storage for 56
  requirements for QMF 7
database *(continued)*
  slow performance 63, 238
DATABASE 2 (DB2) 49
DB2
  acquiring a dbspace 102
  authority
    DBA authority 94
    displaying users' 91
  change confirmation panels 103
  CICS connection 49
  CICS partitions 49
  DBA authority 94
  deleting QMF V1 objects 268
  displaying users' authority 91
  enlarging dbspaces 109
  explicit connection 68
  granting and revoking
    privileges 94
  granting CONNECT
    authority 81
  granting RESOURCE
    authority 102
  guest sharing 8, 89, 135
  implicit connection 68
  running without user
    interaction 68
  SQL authorization ID 68
  system tables 110
DB2 customization 13
DB2 for VSE
  customization 13
DB2 guest sharing 8
DBCS (double-byte character set)
  installing under VSE 35
  printing support 75
DBCS (double-byte character set)
  support
  DSQSDBCS 75
  edit codes 196
  Katakana characters 196
  Latin characters 196
  shift characters 196
dbspace 7
  acquiring 102
  ADD dbspace statement 102
  calculating size 102
  creating tables 101
  deleting 91
  enlarging 109
  nonrecoverable 103
  QMF-supplied tables 275
  requirements 7
  specifying in user profile 86
DCT (Destination Co ntrol
  Table) 121

DCT (Destination Co ntrol Table) 121 *(continued)*
defining Family 3 printer 121
DCT (Destination Control Table)
entries for trace data queue 243
decimal data, edit routine 161
DECOPT column (Q.PROFILES) 85
default
function keys 146
GDDM module ADMADFC 120
QMF profile 84
default function keys 146
default QMF profile 84
defaults module, GDDM
printing 120
deleting
QMF VSE V1 266
deleting QMF V1 266
DEQ command
print queues 123
trace data 67
DEQ command, storage queues 67
DESCRIBE command
customizing 97
DESCRIBE command, customizing 97
destination control table (DCT) 12, 22, 23
Destination Control Table (DCT) 121
defining Family 3 printer 121
entries for trace data queue 243
Deutsch NLF 11
DEVTOK keyword, ADMMNICK specification 117
DFHEAI/DFHEAI0 modules 168, 173
DFHEIENT prolog macro 168
DFHEISTG macro 169
DFHTEMP, sizing 58
diagnosis 233
aids 240
CICS 248
dumps 248
message support 240
problem reporting 251
Q.ERROR_LOG table 250
SQL return codes 242
symptoms 240
system error messages 242
termination messages 249
trace facility 242
disk storage requirements 6
DISPLAY command, SQL privileges required 92
DOS printers 120
double-byte character set (DBCS) 9, 35
double-byte character set (DBCS) support
DSQSDBCS 75
edit codes 196
Katakana characters 196
Latin characters 196
shift characters 196
DRAW command
SQL privileges required 92
DRAW command, SQL privileges required 92
DSQ3BDEL job 268
DSQ3EDBI 16, 41
DSQ3EINS 9, 15
DSQ3ELNK 10
DSQ3GV3 sample job 230
DSQ3INIT 9, 13
DSQ3nCSD 23
DSQ3nDBI 11, 19
DSQ3nINS 11, 18
DSQ3nLNK 11
DSQ3SETQ 17
DSQ3XCTA sample job 173
DSQ3XCTC sample job 184
DSQAP_CICS_PQNAME variable 124
DSQAP_CICS_PQTYPE variable 124
DSQCP global variables 103
DSQDBUG, routing trace data 243
DSQDC_SHOW_PANID global variable 241
DSQEC_COLS_SQL variable 97
DSQEC_FORM_LANG variable 111
DSQEC_NLFCMD_LANG variable 111
DSQEC_RERUN_IPROC global variable 72
DSQEC_TABS_SQL variable 97
DSQSBSTG parameter 57
DSQSDBCS parameter 75
DSQSDBQN parameter 66
DSQSDBQT parameter 66
DSQSDBUG parameter 65
DSQSIROW parameter 63
DSQSMODE parameter 51, 69
DSQSPILL parameter 58
DSQSRUN parameter
defined 69
noninteractive session 72
passing variables 72
used within CICS application 51
DSQSSPQN parameter 58, 62
DSQSUSER parameter 51, 67
defined 67
used within CICS application 51
DSQUECIC edit program 163
DSQUEGV3 phase, governor exit 212
DSQUnGV3 phase, governor exit 208
DSQUXCTA sample edit routine 168, 176
dumps for diagnosis 248
DXEECS control block 169, 178
DXEGOVA control block 215
DXEXCBA control block 220

E

E-deck processing 173, 230
edit
codes 160
binary data 237
CASE field of profile 161
DBCS data 196
locally defined 161
numeric data processing 161
types 160
UDN 163
VSS 163
exit interface 159
control block fields 165
formatting calls 161
HLASM 169
input area 166
output area 166, 167
termination calls 167
VS COBOL II 178
exit phase 165
routine 159
DBCS data 196
defining to CICS (HLASM) 175
defining to CICS (VS COBOL II) 186
general structure 163
HLASM 168
sample program (COBOL) 176
sample program (HLASM) 168
scratchpad area 178
storing data between calls 169
VS COBOL II 175
EDIT TABLE command
concurrent editing 95
SQL privileges required 92

English QMF, NLID 11
English support in NLF session 111
ENQ command
  print queues 123
  trace data 67
ENQ command, storage queues 67, 123
entry point, governor 210
ENTRY_TYPE column (function key table) 150
environment
  changing in QMF profile 88
  customizing 82
  default setup 271
ENVIRONMENT column (Q.PROFILES)
  DB2 guest sharing 89
  role in profile initialization 89
error
  incomplete data object 59
  initialization 65, 72
  insufficient storage 63
  messages
    authorization 92
    during install 34
    warning 234
  printing 130
  QMF log 250
  QMF transaction timeout 57
  reporting to IBM 251
  startup failure 63, 67
error messages 34
estimating
  spill file size 59
EXEC CICS LINK command 168
EXEC CICS RETURN command 169
EXEC CICS START command
  migration utility 262
  QMF session 51
EXIT command
  procedures 72
EXIT command, procedures 72
exits, E-deck processing (VSE) 173, 230
explicit connection
  defined 68
  granting CONNECT authority 81
EXPORT TABLE, SQL privileges 92
extended floating point, edit routine 161

F

F parameter 63
Family 1 printer 117, 118
Family 2 printer 117, 118
Family 3 printer 117, 119
file control table (FCT) 12, 22
floating point data, edit routine 161
FLOATVAL column, Q.RESOURCE_TABLE 202
formatting calls, edit routine 161
forms
  controlling user access 95
  creating new edit codes 160
  displaying 107
  internal stored format 105
  listing 107
  migrating 260
  NLF support 111
  printing 114
  window IDs 154
French NLF 11
FSFRCE services, printing 123
full-screen panels 151, 153
  customized function key examples 151
  panel IDs 153
function calls
  branch addresses 213
  GOVFUNCT values 212
  types 211
function keys
  customizing 87
    activating new definitions 156
    appearance on screen 150
    command 150
    examples 151
    full-screen panel 151
    guidelines 149
    help panel 152
    problems activating 149
    prompt panel 152
    updating function key table 149
    user profile modification 156
    window panel 152
  default settings 146
  index on table 149
  initialization messages 234
  panels 145
  table 148
    authorizing users 156
    creating 148
    entering definitions 149
    maintenance 156
    panel IDs 153
function selection menu 50

G

GDDM (Graphical Data Display Manager)
  ADMADFC defaults module 120, 274
  chart formats 104, 274
  default setup 273
  error 34
  error messages, printing 235
  modifying ADMF file 8
  nonresident programs, performance 273
  PGF product 274
  printer nicknames 116
    ADMADFC defaults module 120
    ADMMNICK specification 117
    multiple addresses 121
  printing 116, 123
  PROCOPT parameter 120
  QMF considerations 8
  version 2.3 tailoring 8
  VSAM requirements 6
GDDM-PGF 37, 274
generic QMF profile 82
German NLF 11
GETVIS 8
GETVIS storage
  amounts per user 5, 57
  installation requirements 8
  setting when QMF starts 57
global variables
  confirming database changes 103
  DSQDC_SHOW_PANID 241
  DSQEC_RERUN_IPROC 72
  English support for NLFs 111
  object list, displaying 97
  printing 124
  window IDs 154
governor exit routine
  branch table 213
  cancellation service 228
  CICS control block interface 208
  command processing 213
  control information, storing 227
  description 199
  entry point 210
  exit routine information 220
  flow of control 208
  function calls 213
  passing resource control information 215
  performance 214

governor exit routine *(continued)*
program structure 208
resource control table 199
RESOURCE_GROUP 87
sample JCL 230
scratchpad area 227
sharing 209
specifying for resource groups 206
think time 212
translating, assembling, link-editing 230
types of function calls 211
GOVFUNCT values for function calls 212
GRANT statement
CONNECT authority 81
PUBLIC keyword 94
RESOURCE authority 102
WITH GRANT OPTION 94
Graphical Data Display Manager (GDDM)
ADMADFC defaults module 120, 274
chart formats 104, 274
default setup 273
error messages, printing 235
nonresident programs, performance 273
PGF product 274
printer nicknames 116
ADMADFC defaults module 120
ADMMNICK specification 117
multiple addresses 121
printing 116, 123
PROCOPT parameter 120
graphics printers, defining nicknames 116
guest sharing 8, 89

H

Hangeul NLF 11
hardware requirements 3
help
customizing panel function keys 152, 154
panel test during IVP 32
help panel test during IVP 32
help panels
customized function key example 152
panel ID 154
HEX function 237
history file, deleting QMF V1 267
HLASM (High-Level Assembler)
edit routine
assembling 173
compiler options 173
defining to CICS 175
edit exit routine structure 168
edit function call 168
interface control block 169
link-edit 173
program translation 172
governor exit routine 230
printing, sample program 124
home panel 32
during IVP 32

I

I parameter 69
ICU (Interactive Chart Utility) 104, 274
ID 67
CICS terminal operator 81
QMF panels 153
implicit connection 68
IMPORT TABLE command
creating tables 100
default dbspace 275
SQL privileges required 92
incomplete data object 59
index
command synonyms table 134
function key table 149
Q.OBJECT_DATA 105
Q.OBJECT_DIRECTORY table 104
Q.OBJECT_REMARKS 106
Q.PROFILES table 85
Q.RESOURCE_TABLE 202
recreating 109
initial procedure 69
initial procedures
specifying 69
initialization 49
errors 63, 234
message numbers 241
performance 271
procedure 13
QMF profile values 89
slow performance 72
tracing errors 65
initialization procedure 13
input area
control for formatting 164
control for termination 167
installation
job 15
overview 9
prerequisite software 3
for optional features 4
process 9
verification procedure (IVP) 31
insufficient storage message 63
integer data, edit routine 161
Interactive Chart Utility (ICU) 104, 274
interactive session 69
interface control block
DXEGOVA 215
DXEXCBA 215
HLASM edit routine 169
VS COBOL II 178
INTVAL column, Q.RESOURCE_TABLE 202
IPL procedures 7
ISC (intersystem communicaton) 274
isolation levels
cursor stability 96
uncommited read 96
ISQL queries 263
Italian NLF 11
IVP (Installation Verification Procedure) 31

J

Japanese NLF 11
JCL samples
deleting QMF Version 1 266
governor exit routine 230
printing using VSE POWER 124
using edit routines
COBOL (DSQ3XCTC.Z) 184
HLASM (DSQ3XCTA.Z) 173

K

K parameter 75
Katakana terminals
setting up DBCS support 75
UCF support 11
Katakana terminals, DBCS support 11, 75
keywords, reporting problems 251
Korean NLF 11

L

L parameter 58
L1 tracing 66
L2 tracing 65
LENGTH column (Q.PROFILES) 85
library exits, E-deck processing 173

library space 6
linear procedures in command synonyms 138
link-edit jobs 20, 37
link-edit statements
  COBOL edit exit routine 184
  governor exit routine 230
  HLASM edit exit routine 173
LIST command
  ALL keyword 107
  customizing 97
list view
  rules 99
literals in command synonyms 140
location window IDs 154
locks on tables
  releasing control tables 63
  SQL GRANT statement 95
logon to QMF
  enabling 82
  restricting 84
loop problems, initialization 72

M

M parameter 69
macros to define printers 121
maintenance
  command synonym table 142
  displaying objects 107
  enlarging dbspace for objects 109
  function key table 156
  listing objects 107
  listing tables 110
  listing views 110
  QMF and database objects 104
message
  authorization 92
  canceling user activity, governor 228
  incomplete data object 59
  insufficient storage 63
  log from migration 265
  printer name 130
  printing errors 235, 237
  QMF message services 240
  row limit exceeded 200
  tracing 66
  warning, QMF Home panel 234
migrating
  CICS 37
  DB2 37
  GDDM 37
migration 259
  column values 260
  ISQL queries 262, 263
migration 259 *(continued)*
  message log 265
  parameter values 262, 265
  replacing objects 264
  user profiles 265
  utility 260
MODEL column 88, 105
MRO (multiregion operation) 274
MSGLOG parameter, migration 265
multiple user mode (DB2) 49

N

name
  ADMMNICK specification 117
  column in control tables 105
  printers 116
  spill file storage 62
  trace data storage 66
NAME parameter, migration 264
nickname
  defined 116
  defining multiple printers 119
  errors during printing 235
Nihongo NLF 11
NLF
  CLC numbers, ServiceLink 251
  command synonyms 138
  English support 111
  governor, sharing 209
  multiple profiles 84
  QMF profile values 84
  starting QMF 51
  TRANSLATION column (Q.PROFILES) 83
NLF (national language feature)
  customize QMF 18
  installation verification procedure (IVP) 34
  NLID 10
  overview 10
  re-installing 38
  updating maps 40
  VSAM catalog requirements 6
NLF (National Language Feature)
  changing in QMF profile 87
  DBCS printing 75
NONE keyword 65
noninteractive session
  ID and password 67
  method for connecting to DB2 68
  starting 69
nonrecoverable dbspace 103
Notices 277
NUMBER column (function key table) 151
numeric data conversion, edit routine 161

O

object 5, 81
  authorization to use 92
  cascading authority 95
  control tables 104
  deleting 108, 111
  displaying 107
  enlarging dbspace 109
  internal representation 104, 260
  list
    customizing 98
    default views 97
    window IDs 154
  listing 107
  maintenance 104
  migration 262
  name, command synonym 136
  ownership 94
  privileges 92
  retrieval storage 58
  sharing 95, 107
  standards for creating 96
  storage 56
  transferring ownership
    queries, forms, procedures 108
    tables, views 111
OBJECT column (synonyms table) 134, 137
object lists
  customizing 99
OBJECTLEVEL column, QMF control tables 105
online support, DB2 49
open enrollment 83
OS/2 printers 120
output area
  control for formatting 164
  control for termination 167
OWNER column, QMF control tables 105
ownership
  control tables 82
  how QMF tracks 104
  transferring 108, 111

P

page sizes for dbspace 102
panel
  class ID 150
  confirmation 85, 103
  customized function keys 145

panel *(continued)*
file (DSQPNLn)
error message 35
loading 15
updating 39
governor prompt 199
IDs 153
print and display support 273
PANEL column (function key table) 150
panel file (DSQPNLn)
error message 35
loading 15
updating 39
panels
class ID 150
confirmation 85, 103
customized function keys 145
governor prompt 199
IDs 153
print and display support 273
parameters
migration utility 262
passed to edit routine 164
QMF program 55, 76
partition size
CICS 7
installation only 8
partition storage 5, 57
password, connection to DB2 67
passwords 17
PC printers 120
performance
data retrieval 63
eliminating duplicate rows 99
governor exit routine 214
initialization 72
loading QMF to SVA 273
QMF transaction timeout 57
reports 58, 63
resident programs 271
slow, causes 238
table indexes 101
using spill file 58, 61
PF keys 87
PF_SETTING column (function key table) 151
PFKEYS column (Q.PROFILES) 87
phases 39
Portuguese NLF 11
POWER queues, printing 124
prerequisite software 3
for optional features 4
preventive service 6
PRINT command 115, 124
PRINT command 115, 124 *(continued)*
routing to named destinations 115, 124
PRINT TABLE command, SQL privileges required 92
printer
ANSI support
GDDM nicknames 121
graphic device 115
QMF-provided services 123
control keywords (PRINTCTL) 120
DBCS support 75
DOS 120
Family 1 117
Family 2 118
Family 3 119
length of output 85
multiple addresses 117, 121
nicknames 116, 117
OS/2 120
PROCOPT parameter 120
width of output 85
PRINTER column (Q.PROFILES) 86
printing 113
defining printers to CICS 121
errors 235, 237
QMF vs. GDDM 115
sample program 126
summary 114
temporary storage queue 123
to PC printers 120
transient data queue 123
updating user profiles 130
using GDDM services 116
using QMF services 123
VSE POWER 124
private dbspace 102
privileges 89
commands 92
database objects 92
for table editor 94
GRANT statement 94
granting to all users (PUBLIC) 94
queries 93
REVOKE statement 95
revoking 95
privileges required for QMF tasks 92
problem reporting 251
procedures
controlling user access 95
displaying 107
procedures *(continued)*
initial 69
internal stored format 105
listing 107
maintaining objects 105, 111
migrating 260
printing 114
running without user interaction 69
using in command synonyms 138
processing program table (PPT) 12, 25
processor 3
PROCOPT parameter, printing 120
profile
CASE setting, customized function keys 151
command synonyms 141
creating 82
DB2 guest sharing 89
default values 84
deleting 84, 91
function key customization 156
initialization search order 89
maintenance 104
migrating Version 1 265
multiple (NLFs) 84
print parameters 130
printing 114
Q user ID 82
SET PROFILE command 89
sizing printed reports 130
updating 89, 91
program control table (PCT) 12, 25
program parameters 51, 76
CONNECT ID and password (DSQSUSER) 67
DBCS support (DSQSDBCS) 75
default values 55
initial procedure (DSQSRUN) 69
interactive session (DSQSMODE) 69
maximum rows retrieved (DSQSIROW) 63
name of spill file (DSQSSPQN) 62
names 55
noninteractive session (DSQSMODE) 69
report storage (DSQSBSTG) 57
short forms 55
trace data storage (DSQSDBQT) 66

program parameters 51, 76
(continued)
trace data storage name (DSQSDBQN) 66
tracing errors (DSQSDBUG) 65
prolog macro, edit routine 168
prompt panel
customized function key example 152
panel ID 154
prompted query
printing 114, 116
SQL privileges 93
window IDs 155
PTFs 39
PTFs, applying
under VSE 39
public dbspace 102
PUBLIC keyword 95

Q

Q.DSQ_RESERVED control table 275
Q.ERROR_LOG control table 250, 275
Q.INTERNAL_DATA table (V1) 260
Q.OBJECT_DATA control table
default dbspace 275
enlarging dbspace 109
migrating objects 260, 265
Q.OBJECT_DIRECTORY control table
default dbspace 275
enlarging dbspace 109
maintaining objects 104, 111
migrating objects 260, 265
Q.OBJECT_REMARKS control table
default dbspace 275
enlarging dbspace 109
maintaining objects 106, 111
migrating objects 260, 265
Q.PROFILES control table
adding user profiles 83
DB2 guest sharing 89
default dbspace 275
deleting user profile 84
table structure 84
updating 89
updating PFKEYS field 156
updating RESOURCE_GROUP field 200
updating SYNONYMS field 141
user modifications 89
Q.RESOURCE_TABLE control table
default dbspace 275
governor exit routines 202
Q.RESOURCE_VIEW, governor 203
Q user ID 32
Q user profile 82
QBE query
printing 114
SQL privileges 93
QMF 81
control tables 16
deleting Version 1 266
error messages, printing 237
establishing user support 81, 112
maps 15, 40
objects 5, 57
packages 16
printing 123
program stamps 249
session 82
storage required 5
QMF command
rules 51
QMFE transaction 32
QMFM transaction ID 261
QMFn transaction ID 51
passing program parameters 51, 75
used from cleared CICS screen 51
with CICS application 51
queries
changing default type 86
controlling user access 95
deleting 108
displaying 107
GRANT 94
internal stored format 105
listing 107
migrating 260
printing 115
required privileges 93
storage for execution 57
transferring object ownership 108
query
changing default type 86
deleting
SQL statements 108
displaying 107
internal stored format 105
listing
SQL statements 107
migrating 260
required privileges 93
storage for execution 57
question mark (?)
printer nicknames 121
question mark, printer nicknames 121
QUEUENAME, QUEUETYPE keywords 124

R

re-installing QMF 38
RELOAD dbspace command 109
REMARKS column 106
remote database servers, installing QMF into 27
REPLACE parameter, migration 264
replacing
existing products 38
text decks and phases 39
replacing existing products 38
replacing text decks and phases 39
reports
binary data 237
data formats 159
printing 114
Q.ERROR_LOG table 250
slow performance 63, 239
storage
allocating extra 58
estimating spill file size 58
setting amount of GETVIS 57
width/length 85, 130
RESET DATA command 63, 238
resident QMF programs 271
resource
controlling 199
governor exit routine 199
group 84
default (SYSTEM) 202
limiting 199
profile 87
ownership 82
passing control information 215
problem log 250
profile management 84
RESOURCE authority 102
RESOURCE_GROUP column 87, 202
RESOURCE_OPTION column 202
restricted access to QMF 84
RESTRICTED column
changing value to NO 107
defined 105
return codes, SQL 242
REVOKE statement 95

rows, controlling number retrieved 199
rules
command synonyms 136
customizing function keys 149
rules for command synonyms 136, 141
rules for creating a list view 99
rules for customizing function keys 149
rules for QMF command 51
RUN command
command synonym 137
initial procedure 69
SQL privileges required 92

S

sample
charts 15
tables
installing 16, 33
predefined dbspaces 275
sample charts 15
sample tables 16, 33, 275
SAVE command
DATA keyword 92
default dbspace, tables 275
performance 63
SHARE parameter 95
SQL privileges required 92
TABLE keyword 92
SCOPE resource option 202
scratchpad area
edit routines 178
governor exit routine 227
SEQ column 106
ServiceLink 251
servicing QMF 37
under VSE 37
session 82
cancellation service 228
customizing
at initialization 51
user profile 82
ID and password 67
interactive vs. noninteractive 69
SET PROFILE command 89
share locks on data 63
SHARE parameter 95
shared virtual area (SVA) 272
shift characters 196
sign-on table (SNT) 81
Simplified Chinese NLF 11
small integer data, edit routine 161
SNT (sign-on table) 81
software requirements 3
under VSE 3
Software Support Facility (SSF) 251
SPACE column (Q.PROFILES) 86
space requirements 6
Spanish NLF 11
spill file 58
activating 62
allocating 58
estimating size 59
name 62
performance problems 61
sample calculations 60
SQL
ID 68
attached to user profile 85
command synonym table 144
connecting to DB2 67
default 68
how QMF stores 106
Q 82
privileges 68
for prompted, QBE queries 93
for QMF commands 92
for table editor 94
granting 94
Q.PROFILES table update 89
revoking 94
table and view access 92
queries, printing 114
return codes 242
statement 68
ACQUIRE DBSPACE 102
ALTER DBSPACE 110
CREATE TABLE 100
GRANT 94
INSERT (new user profile) 83
REVOKE 95
UPDATE 90
SQLDBA."STORED QUERIES" table 263
SQLDBA password 17
SSF (Software Support Facility) 251
stamps, QMF programs 249
starting QMF 49, 63
as a CICS transaction 51
DB2 connection to CICS 49
errors 63, 67
from a CICS application 51
passing program parameters 51, 75
QMF connection to DB2 68
starting QMF 49, 63 *(continued)*
QMF profile initialization 89
table lock failure 95
startup job 25
storage
allocating extra report storage 57, 58
CICS partition 5, 56
data from edit routine 164, 169
dbspace
calculating size 102
increasing size 109
for printing 123
GETVIS 5, 57
incomplete data object 59
insufficient storage prompt 63
limiting users' storage 58
object retrieval 58
QMF phases and objects 57
reports
allocating extra (spill file) 58
allocating GETVIS 57
sizing DFHTEMP 58, 59
spill file 58
SVA requirements for QMF 272
temporary storage queue
printing using QMF services 123
reports 56, 58
trace data 66, 247
trace data 66, 243
transient data queue
printing using QMF services 123
routing output 115
trace data 66, 247
troubleshooting 239
storage violation 34
SUBTYPE column, QMF control tables 105
support products
CICS 271
GDDM 273
setup 271
SUSPEND keyword 124
SVA (shared virtual area) 272
Swiss French NLF 11
Swiss German NLF 11
SYNONYM_DEFINITION column 137
SYNONYMS column (Q.PROFILES) 87
synonyms for QMF commands 133, 135
abbreviations 142

synonyms for QMF commands 133, 135 *(continued)*
activating for users 141
creating synonyms table 133
index 134
initialization messages 234
object name 136
problems activating 136
quotation marks 140
synonym definition 137
syntax 140
table maintenance 142
using variables 139
verb 136
SYSLST
printing 123
routing trace data 243
SYSTABAUTH system table 91
system
error messages 242
printing errors 237
tables, DB2 110
SYSTEM profile
changing default values 91
deleting 84
SYSTEM resource group 202
system tables
CICS, deleting QMF V1 270
DB2 91
SYSUSERAUTH system table 91

T

T parameter 65
Table Editor
confirmation panels 103
SQL privileges required 94
tables
command synonym 134
concurrent editing 95
control tables 83
controlling access 92
creating 100
DB2 system 110
deleting 111
enlarging dbspaces 109
function keys 145
indexes 101
listing 110
locks 95
maintenance 110
printing 114
QMF control tables 83
resource control, governor exit 202
transferring ownership 111
tailoring 21
tape
drive 3
TCT (Terminal Control Table), defining printers 121
TD keyword 66
TDL edit code 161
temporary storage queue
printing using QMF services 123
reports 56, 58
trace data 66, 247
terminal 3
changing case 85
DBCS data support 75
GDDM nicknames 116
installation requirements 3
operator ID 81
running without 69
UCF support for Katakana 11
Terminal Control Table (TCT) 121
TERMINAL field, CICS TCT 118
termination calls, edit routine 167
termination messages 249
testing QMF 31
text decks 39
think time 212
timeout, QMF transaction
CICS partition size 239
defining message display 249
TOFAM keyword, ADMMNICK specification 117
toggle switch, governor exit 202
TONAME keyword, ADMMNICK specification 118
trace
data
level of detail 65
storage 66
storage queue name 66
viewing 247
facility
file allocation 242
functions 244
starting 65, 244
stopping 248
level of detail 86, 244
message logging 234
storage queue 67
TRACE column (Q.PROFILES) 86
transaction
QMFM 261
QMFn 51
routing requests with MRO and ISC 274
transferring object ownership 108
transient data queue
printing using QMF services 123
routing output 115
trace data 66, 247
translation
edit routine
COBOL 183
HLASM 172
governor exit routine 229
TRANSLATION column (Q.PROFILES) 87
TRMIDNT field, CICS TCT 118
troubleshooting 233
diagnostic aids 240
initialization errors 234
performance problems 238
printing errors 237
storage requirements 239
TS keyword 66
TTL edit code 161
TYPE column, QMF control tables 105
TYPE parameter, migration 262
TYPETERM specification 249
TYPETERMs for QMF display 249

U

U edit codes, forms 160, 166
defined 160
input area 166
UCF (Uppercase Feature) 11
UDN edit code 163
UID parameter 51
uncommited read 96
UNLOAD dbspace command 109
updating
packages 41
products 38
updating packages 41
updating products 38
Uppercase Feature (UCF) 11
user
adding new 83
authorization for objects 92
CONNECT ID 67
edit routines 159
limiting resources 199
objects 107
password 67
support 81
creating tables 100
object access 92
profile and object maintenance 104

V

V edit codes, forms 160, 166
  defined 160
  input area 166
variables
  global 72
  in synonym definitions 139
  passing using DSQSRUN parameter 72
  using &ALL 139
VERB column (synonyms table) 134, 136
verification procedure 31
Version 1 QMF, deleting 266
views
  controlling access 92
  deleting 111
  listing 110
  maintenance 110
  object lists, customizing 97
  privileges for queries 93
  privileges for table editor 94
  Q.RESOURCE_VIEW, governor exit 203
  recreating 110
virtual storage, estimating 5, 57
virtual storage error 34
Virtual Storage Extended (VSE)
  DB2 guest sharing 89, 135
  library exit, E-deck processing 173
  operator ID 68, 81
  printing to POWER queues 124
VS COBOL IIedit routine
  compiling 184
  copybook 178
  delimiters 186
  edit exit routine structure 175
  edit program call 176
  interface control block 178
  link-edit 184
  translating for use 183
VSAM catalog requirements 6
VSE (Virtual Storage Extended)
  ALLOC statement 7
  DB2 guest sharing 89, 135
  library exit, E-deck processing 173
  operator ID 68, 81
  printing to POWER queues 124
VSE ALLOC statement 7
VSE POWER
  printing to 124
VSS edit code 163

W

warning messages 35, 234
WIDTH
  column in Q.PROFILES 85
WIDTH column (Q.PROFILES) 85
window panels
  customized function key examples 152
  IDs 153

Readers' Comments — We'd Like to Hear from You

Query Management Facility™
Installing and Managing QMF for
VSE/ESA
Version 7

Publication No. GC27-0721-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

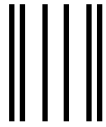
Company or Organization _____

Phone No. _____

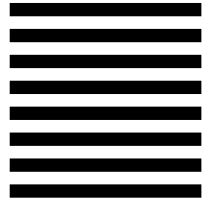
Readers' Comments — We'd Like to Hear from You
GC27-0721-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM CORPORATION
Department HHX/H3
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

Fold and Tape **Please do not staple** Fold and Tape

GC27-0721-00

Cut or Fold
Along Line

Program Number: 5697-F42

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC27-0721-00

Spine information:

Query Management Facility™ Installing and Managing QMF for VSE/ESA Version 7