Query Management Facility
12.2

Getting Started with QMF Z Client

GC27-9133

Note

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

August 2, 2021 edition

This edition applies to Version 12 Release 2 of IBM Query Management Facility (QMF) Enterprise Edition Advanced, which is a feature of IBM Db2 12 for z/OS (5650-DB2), Version 12.1. It also applies to Version 12 Release 2 of IBM QMF for z/OS (5697-QM2), which is a stand-alone IBM Db2 for z/OS tool. This information applies to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this information

This information describes how to use the QMF Z Client application.[1]

Always check the Db2® and IMS Tools Library page for the most current version of this publication:

http://www.ibm.com/software/data/db2imstools/db2tools-library.html

Who should read this information

This information is intended for all QMF Z Client users.

Service updates and support information

To find service updates and support information, including software fix packs, PTFs, Frequently Asked Questions (FAQs), technical notes, troubleshooting information, and downloads, see http://www.ibm.com/software/data/qmf/support.html

[1] Throughout this information, the IBM® QMF Z Client client is referred to as QMF.

Chapter 1. QMF overview

QMF features

The QMF Z Client solution offers a set of business intelligence functions for mainframe users.

Relational queries
Creation of relational queries is facilitated by different query interfaces that are tailored to different skill and knowledge levels.

Reports
QMF offers flexible design environment for reports, allowing you to group, aggregate, and summarize data, add calculation expressions, and conditionally format the report depending on the query results.

Data editing capabilities
QMF provides built-in table editing capabilities that allow you to add, delete, and change entire rows or individual cells within a table. You can also create, edit, and run sophisticated procedures to carry out a variety of tasks.

Procedures with logic
Procedures with logic combine QMF commands with REXX statements and functions allowing you to create powerful application programs.

Configuration and invocation

QMF Z Client can be started only from z/OS. QMF Z Client can be set up to run under TSO, ISPF, or as a batch job.

Allocating files used by QMF Z Client

Important:

Before you can use QMF Z Client, you must install and configure QMF Server. For detailed information, see Installing and Managing QMF Server.

Before starting QMF Z Client, you must allocate and customize certain files that are used by the application.

Complete the following steps:

1. Access the FQMINI properties file that is stored in the FQMprfx.SFQMPARM target library. The file contains configuration information for the QMF Z Client session. For more information, see “QMF Z Client program parameters” on page 5. If you chose to tailor the default environment, copy the file to a working version of the data set. Note, that you must keep the member name as FQMINI.
2. Allocate your FQMprfx.SFQMPARM(FQMINI) or working copy data set to the DDNAME FQMPARM.
3. Edit and customize the FQMprfx.SFQMSKEL(FQMBATCH) ISPF skeleton as appropriate for your environment. This skeleton is used to generate JCL jobs when the QMF Z Client BATCH command is invoked. Note that FQMprfx.SFQMSKEL, or a tailored version of this data set, must be allocated to the DDNAME ISPSLIB.

   You can customize the FQMprfx.SFQMSKEL(FQMBATCH) ISPF file by changing certain parameters. For more information, see “Customizing FQMprfx.SFQMSKEL(FQMBATCH) ISPF file and parameters” on page 2.
4. To allow QMF Z Client run on different national languages, allocate the library containing national language files with the FQMLANG DDNAME. The national language files are located in the FQMprfx.SFQMLANG target library.

5. Allocate the QMF Z Client trace output data set to the DDNAME FQMDEBUG. When allocating, use following DCB parameters:

```
DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
```

Table 1. QMF Z Client Version 12.2 target libraries and their descriptions

Library name	Description
SFQMLOAD	QMF Z Client load library.
SFQMPARM	Contains the QMF Z Client configuration file.
SFQMLANG	National language files library.
SFQMSKEL	ISPF skeleton library, which is required only if you are running QMF Z Client under ISPF.
SFQMEXEC	Exec library. Contains sample invocation script.

Customizing FQMprfx.SFQMSKEL(FQMBATCH) ISPF file and parameters

To begin you can modify the following:

1.
```
//       &JOBNAME JOB MSGLEVEL(1,1),
//       MSGCLASS=&O,
//       USER=&TSOUSER,
)SEL &TSOPWORD ¬= &Z
//       PASSWORD=&TSOPWORD,
)ENDSEL
//       NOTIFY=&ZUSER
```

   Use your login ID and login password in USER and PASSWORD field respectively. Use MSGLEVEL and MSGCLASS as per your environment.

2.
```
//       QMFINVOK EXEC PGM=IKJEFT01,
//       TIME=&TIME,
//       DYNAMNBR=30,
//       REGION=&REGION
```

   - This statement calls TSO (PGM=IKJEFT01). This statement should not be changed.
   - Specify value for the TIME parameter. For example, TIME = NOLIMIT.
   - Specify an adequate number of allowable dynamic allocations. For example, DYNAMNBR=30.
   - Specify a sufficiently large region for QMF Z Client, for example you can use REGION=0M.

3.
```
//STEPLIB  DD  DSN=&ISPFPRE..SISPLOAD,DISP=SHR
//         DD  DSN=&QMFPRE..LOAD,DISP=SHR
//ISPPLIB  DD  DSN=&ISPFPRE..SISPPENU,DISP=SHR
//ISPMLIB  DD  DSN=&ISPFPRE..SISPMENU,DISP=SHR
//ISPSLIB  DD  DSN=&ISPFPRE..SISPSENU,DISP=SHR
//         DD  DSN=&ISPFPRE..SISPSLIB,DISP=SHR
//ISPTLIB  DD  DSN=&ISPFPRE..SISPTENU,DISP=SHR
//ISPPROF  DD  UNIT=SYSDA,SPACE=(TRK,(9,1,4)),
//         DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//SYSUDUMP DD   SYSOUT=&O
```

   This statement assigns program load libraries and allocate data set used by ISPF. Use the following DCB parameters for ISPFPROF:

   - UNIT=SYSDA,SPACE=(TRK,(9,1,4))
   - DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)

4.
```
//SHZDEBUG DD  SYSOUT=&O,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
```

   Use DCB values as above for allocating data set used by QMF Z Client. You can use SYSOUT= A.

This data set contains the message output from TSO and ISPF. For this data set, specify DD statements. For example, you can use SYSOUT= A or you can set it as follows:

```
//SYSTSPRT DD  SYSOUT=&O
```

5. **SYSTSIN**

   SYSTSIN holds the TSO statements that run during the job step. To include these statements in your JCL, specify DD statements like those shown here:

```
//SYSTSIN  DD  *
ISPSTART PGM(SHUTTLEZ) +
PARM(/&QMFPRE +
M=B +
)SEL &DBUSER ¬= &Z
"DBLogin=&DBUSER" +
)ENDSEL
)SEL &DBPWORD ¬= &Z
"DBPassword=&DBPWORD" +
)ENDSEL
)SEL &REP ¬= &Z
"Repository=&REP" +
)ENDSEL
)SEL &REPUSER ¬= &Z
"RepositoryLogin=&REPUSER" +
)ENDSEL
)SEL &REPPWORD ¬= &Z
"RepositoryPassword=&REPPWORD" +
)ENDSEL
)SEL &DS ¬= &Z
"Datasource=&DS" +
)ENDSEL
)SEL &DSUSER ¬= &Z
"DatasourceLogin=&DSUSER" +
)ENDSEL
)SEL &DSPWORD ¬= &Z
"DatasourcePassword=&DSPWORD" +
)ENDSEL
"I=&PNAME") +
NEWAPPL(SHZ&LCHAR)
```

   The ISPSTART statement invokes batch-mode QMF Z Client with ISPF.

   - DBLogin: Give Db2 user login
   - DbPassword: Give Db2 user password
   - Repository: Give the repository name
   - RepositoryLogin: Give the repository login user name
   - RepositoryPassword: Give the repository login password
   - Datasource: Give datasource name
   - DatasourceLogin: Give datasource login name
   - Datasource Password: Give datasource password

Configuration files and program parameters

The FQMprfx.SFQMPARM target library contains the installation level QMF Z Client configuration file. The configuration file, also known as the properties file, contains descriptions and default settings for QMF Z Client program parameters. In this file, you can specify program parameters that will affect all users of your QMF Z Client installation.

There are three ways to override the default program parameters that are set in the file:

- Specify new program parameters in the user level configuration file. The name of the user level configuration file is specified by the UserParmlib parameter in the installation level configuration file. QMF Z Client user may allocate their own library that contains the configuration file. Program parameters specified in that file will override the program parameters specified in the installation level configuration file.

- Use the OPTFILE program parameter to get additional override values from a DDNAME, Unix path name or data set name. This option might be useful when you run QMF Z Client is running in BATCH mode, so you can specify the program parameters in an in-stream data set. Program parameters specified by using this option will override the user and installation level program parameters.
- Specifying the program parameter and a new value upon QMF Z Client invocation. Program parameters specified this way will override any other program parameters.

Running QMF Z Client on TSO

To run QMF Z Client on TSO, use the TSO CALL command. With the command, specify the name of the QMF Z Client load library and pass program parameters after the dataset name. Consider the following example:

```
ALLOC DD(FQMPARM) DA('FQM.SFQMPARM') SHR REUSE
ALLOC DD(FQMLANG) DA('FQM.SFQMLANG') SHR REUSE
ALLOC DD(FQMDEBUG) SYSOUT(A) RECFM(F B A) LRECL(121) REUSE
CALL 'FQM.SFQMLOAD(FQMQMF)' 'FQM MODE=I'
```

The first program parameter is the QMF Z Client installation prefix, and it cannot be omitted. Other parameters are optional.

The QMF Z Client load library is allocated as a task library for the duration of the CALL command. However, you must give QMF Z Client access to the GDDM load library. In most cases, GDDM library is not part of the TASKLIB. If GDDM library is not available, QMF Z Client terminates with an error.

Running QMF Z Client on ISPF

To run QMF Z Client on ISPF, start the QMF program dialog using the ISPF SELECT service. Access hlq.SFQMEXEC(FQMQMF) sample invocation script, copy it, and customize it for your environment.

Starting QMF in a batch environment

You can use QMF batch mode in TSO, ISPF, and native z/OS. To start QMF in a batch mode, use a specific JCL script.

To start QMF in batch mode, your JCL script must resemble the following example:

```
//BATCH JOB
//*************************************************************
//*        QMF Z CLIENT STEP                                  *
//*************************************************************
//QMFINVOK EXEC PGM=IKJEFT01,
//       TIME=NOLIMIT,
//       DYNAMNBR=30,
//       REGION=0M
//*************************************************************
//*        PROGRAM LOAD LIBRARIES                             *
//*************************************************************
//STEPLIB  DD  DSN=ISP.SISPLOAD,DISP=SHR
//         DD  DSN=FQM.SFQMLOAD,DISP=SHR
//*************************************************************
//*        DATASETS USED BY ISPF                              *
//*************************************************************
//ISPPLIB  DD  DSN=ISP.SISPPENU,DISP=SHR
//ISPMLIB  DD  DSN=ISP.SISPMENU,DISP=SHR
//ISPSLIB  DD  DSN=ISP.SISPSENU,DISP=SHR
//         DD  DSN=ISP.SISPSLIB,DISP=SHR
//ISPTLIB  DD  DSN=ISP.SISPTENU,DISP=SHR
//ISPPROF  DD  UNIT=SYSDA,SPACE=(TRK,(9,1,4)),
//        DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB)
//SYSUDUMP DD   SYSOUT=A
//*************************************************************
//*        DATASETS USED BY QMF                               *
//*************************************************************
//FQMPARM  DD  DSN=FQM.SFQMPARM,DISP=SHR
//FQMLANG  DD  DSN=FQM.SFQMLANG,DISP=SHR
//FQMDEBUG DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
//SYSTSPRT DD  SYSOUT=A
//OPTIONS  DD  *
```

```
Mode=BATCH
PasswordEncrypted=1
ServerHost=sys1
ServerPort=9001
Repository=qmfz
Datasource=ABC1
Run=USER.PROC
/*
//SYSTSIN  DD  *
ISPSTART PGM(FQMQMF) +
PARM(/FQM +
OPTFILE=DD:OPTIONS +
) +
NEWAPPL(FQM)
/*
```

Testing QMF Z Client installation on a sample database

Before you start using QMF Z Client for the first time, complete the following steps to ensure that it is installed and configured properly:

1. Run QMF Z Client.
2. Make sure that the **Switch Repository** panel is displayed and the list of available repositories is not empty. If the list is empty, contact your QMF administrator or use the QMF Server application to create a repository.

QMF Z Client program parameters

Long name	Short name	Description
ServerHost		**Controls or sets** Host name of the QMF Server to connect to. **Valid values** Any valid host name or IP address. **Defaults** The default behavior is to fetch the list of available QMF Servers from QMF Server Registry and ask the user to select one.
ServerPort		**Controls or sets** QMF Server port number. **Valid values** 0 to 65535 **Defaults** 2206
IOTimeout		**Controls or sets** Network IO operations timeout interval in milliseconds. **Valid values** 0 to 99999999 **Defaults** 30000

Long name	Short name	Description
UserParmlib		**Controls or sets** Name of the partitioned data set without the first qualifier where the user defined PARMLIB can be found. Specific user TSO PREFIX or user ID will be used as the first qualifier. The specified data set will also be used to store application cache in FQMCACHE member. If UserParmlib is specified in user defined PARMLIB, it will be ignored. **Valid values** Any valid partitioned data set name. **Defaults** None.
Language		**Controls or sets** Application language. **Valid values** E (English), Q (Danish), C (Canadian French), D (German), F (French), I (Italian), K (Japanese), H (Korean), P (Brazilian Portuguese), S (Spanish), V (Swedish), Y (Swiss French), Z (Swiss German) **Defaults** E
FormatLanguage		**Controls or sets** Numbers, dates, and currencies formatting. **Valid values** E (English), Q (Danish), C (Canadian French), D (German), F (French), I (Italian), K (Japanese), H (Korean), P (Brazilian Portuguese), S (Spanish), V (Swedish), Y (Swiss French), Z (Swiss German) **Defaults** E
NumFavs		**Controls or sets** Maximum number of objects in the Favorite Objects list. **Valid values** 0-999 **Defaults** 50
NumComms		**Controls or sets** Maximum number of commands in the command history. **Valid values** 0-999 **Defaults** 20

Long name	Short name	Description
NumRecly		**Controls or sets** Maximum number of objects in the Recently Used Objects list. **Valid values** 0-999 **Defaults** 20
NumMyActs		**Controls or sets** Maximum number of actions in the Actions list. **Valid values** 0-999 **Defaults** 50
Mode	M	**Controls or sets** QMF mode of operation. **Valid values** B (BATCH) or I (INTERACTIVE). **Defaults** I (INTERACTIVE)
PasswordEncrypted[1] [1] This parameter is available in QMF 12.2 Fix Pack 5 (12.2.0.5) and later versions.		**Controls or sets** Encryption of the following password parameters: DBPassword, RepositoryPassword, and DatasourcePassword. **Valid values** 1 **Defaults** 1
Run	I	**Controls or sets** Name of the QMF procedure to run when QMF starts, before the QMF home panel is displayed. **Valid values** Any valid QMF procedure name. **Defaults** None.
Trace	T	**Controls or sets** Level of trace detail. **Valid values** ALL (traces all functions at the highest level of detail) or NONE (no trace data is recorded). **Defaults** NONE in INTERACTIVE mode, L2 in BATCH mode.

Long name	Short name	Description
DBLogin		**Controls or sets** User name to connect to the repository storage. **Valid values** Any valid user name. **Defaults** Your TSO login.
DBPassword		**Controls or sets** Password to connect to the repository storage. **Valid values** Any valid password. **Defaults** None.
RepositoryLogin		**Controls or sets** User name to connect to the repository. **Valid values** Any valid user name. **Defaults** Your TSO login.
RepositoryPassword		**Controls or sets** Password to connect to the repository. **Valid values** Any valid password. **Defaults** None.
DatasourceLogin		**Controls or sets** User name to connect to the data source. **Valid values** Any valid user name. **Defaults** Your TSO login.
DatasourcePassword		**Controls or sets** Password to connect to the data source. **Valid values** Any valid password. **Defaults** None.
Repository		**Controls or sets** Name of repository connection to connect to. **Valid values** Any valid repository connection name. **Defaults** The default repository for the host that is being used.

Long name	Short name	Description
Datasource		**Controls or sets** Name of data source to connect to. **Valid values** Any valid data source name. **Defaults** The default datasource for the repository that is being used.
CCSID		**Controls or sets** Application will translate Unicode text to this CCSID before displaying. **Valid values** Any valid CCSID number. **Defaults** Depends on Language parameter value.
APLSupport		**Controls or sets** The level of APL characters support. **Valid values** FULL (all APL characters), LINE (only box drawing characters), NONE (no APL characters). **Defaults** FULL in case terminal emulator reported that it support APL, NONE otherwise.
OPTFILE		**Controls or sets** UNIX path or data set name or dd name (in format DD:DDNAME) where additional program parameters could be found. **Valid values** Any valid UNIX path or data set name or DD name. **Defaults** None.
QSRHost		**Controls or sets** Host name of the currently used QMF Server Registry. **Valid values** Any valid host name or IP address. **Defaults** The default behavior is to discover the QMF Server Registry address automatically, which is possible if the DNS server was configured for that. For more information, see Installing and Managing QMF Server.
QSRPort		**Controls or sets** QMF Server Registry port number. **Valid values** 0 to 65535 **Defaults** 8080

Long name	Short name	Description
QMFServerPath		**Controls or sets** QMF Server context path. **Valid values** Any valid context path. **Defaults** Empty.

Typical QMF workflow overview

When you work with QMF Z Client, you typically perform the following tasks:

Connecting to a repository
To be able to access data, you must connect to a repository. For more information, see “Connecting to repositories” on page 17.

Connecting to a data source
To be able to access tables, queries, procedures, and other QMF objects, you must connect to a data source. For more information, see “Connecting to data sources” on page 18.

Creating a query
To process data that is stored in a table, you must create and run a query. For more information, see “Creating queries using SQL editor” on page 21 and “Creating queries using prompted query editor” on page 21.

Creating a report
To present the data from the query result set in a comprehensible way, you must create a report. For more information, see “Creating reports” on page 23.

QMF interface overview

The following are the key elements of QMF interface:

Command line

The command line is located at the very bottom of the screen. Use the command line to issue QMF commands and navigate among panels. The command line is your primary way to interact with the application, unless your terminal emulator supports vector graphics. In this case, you can configure the terminal to support a mouse.

If the command that you are typing is too long to fit in the command line, you can open the extended command line on a separate panel. To do so, position the cursor on the word **Command** and press Enter.

To view the complete list of QMF commands, see Appendix C, “QMF Commands,” on page 65.

Scroll field

In the **Scroll** specify the default value for the scroll. Valid values are:

A number in the range 1 - 9999
Scrolls the number of pages or rows.

MAX
Scrolls to the end.

HALF
Scrolls by half a page.

PAGE
Scrolls by one page.

DATA
Scrolls to the line before the end of the page.

CSR
Scrolls based on the position of the cursor. If the cursor is in a scrollable area, scrolls to the end. If the cursor is outside of or at the end the scrollable area, scrolls one page.

Message line

The message line is located at the bottom of the screen directly above the command line. The Message line displays informational, warning, and error messages.

Function keys

Function keys are located at the bottom of the screen above the Message line and can be assigned to the programmable function keys on your keyboard. Each function key can be configured to perform a specific QMF command. For information about configuring function keys, see "Customizing function keys" on page 13.

Action bar

The action bar is located at the top of the screen. It allows you to perform certain actions without typing anything in the command line. Note that the list of action bar items may vary from panel to panel.

Context menu

On some QMF panels, you can right-click an object to access a context menu that contains the list of actions that you can perform on the object.

Quick access areas on the Home panel

The following quick access areas are available on the **Home** panel:

Favorite Objects
Displays the contents of the **Favorite Objects** panel. The **Favorite Objects** panel displays the list of objects that you added to the list of favorites.

Favorite Actions
Displays the contents of the **Favorite Actions** panel. The **Favorite Actions** panel displays the list of available favorite QMF actions.

Recently Used
Displays the contents of the **Recently Used** panel. The **Recently Used** panel displays the list of objects that you have worked with recently.

To display an object or run an action from one of the areas, click the object or action that you want to work with and press Enter.

Chapter 2. Setting preferences

Customizing function keys

Each QMF panel has a set of pre-defined function keys, that you can configure to perform specific QMF commands.

Procedure

1. Open the panel that you want to work with.
2. On the command line, enter `SHOW KEYS`.
3. On the **Keys** panel, position the cursor on the line that corresponds to the key that you want to customize.
4. In the **Label** field, type the name for the function key. If the name is too long for the field, press the **Show Field** function key to open the **Key Editor** panel.
5. In the **Command** field, type the QMF command to associate with the key. If the command is too long for the field, press the **Show Field** function key to open the **Key Editor** panel.
6. Optional: To reset all function keys to their default values, enter `reset key(panelid=ID keyid=all` in the command line, where `ID` is the ID of the panel whose function keys you want to reset. You can find this ID enclosed in parentheses in the **Edit keys for panel** field.

   Note: To view the complete list of QMF panels and their IDs, see Appendix H, "IDs of QMF panels," on page 137.
7. Press the **End** function key to save the changes and close the **Keys** panel.

Restoring default values for function keys

You can restore the default values for all function keys on a panel.

Procedure

1. Open the panel that you want to work with.
2. On the command line, enter `SHOW KEYS`.
3. On the command line, enter `reset key(panelid=ID keyid=all`, where `ID` is the ID of the panel whose function keys you want to reset. You can find this ID enclosed in parentheses in the **Edit keys for panel** field.

   Note: To view the complete list of QMF panels and their IDs, see Appendix H, "IDs of QMF panels," on page 137.

Global variables

QMF features a number of global variables that help you control various aspects of your QMF session, QMF commands, and panel display.

QMF has two types of global variables:

System global variables
: During installation, system global variables are created. The name of each system global variable begins with the DSQ prefix. You cannot create or delete system global variables; you only can edit their default values.

  Use system global variables to control various aspects of your QMF session, QMF commands, and panel display. For example, use the DSQAO_CONNECT_ID system global variable to set the user ID that is used to connect to the current database.

User-defined global variables

You can create user-defined global variables. You specify whether the value of a user-defined global variable is permanent or applies only to the current QMF session. You can create, edit, and delete user-defined global variables.

Use user-defined defined global variables to control the aspects of your QMF session that are not covered by system global variables.

You can use the DSQEC_USERGLV_SAV system global variable to restore all system global variables to their default values and delete all user-defined global variables. For more information about system global variables, see “DSQEC global variables” on page 117.

The following topics describe working with global variables:

Creating user-defined global variables

Use the **Globals** panel to create user-defined global variables.

Procedure

1. On the command line, type SHOW GLOBALS.
2. On the **Globals** panel, press the **Add** function key.
3. In the **Variable name** field on the **Add Global Variable** panel, type a name for the new global variable. To avoid confusing user-defined global variables with system global variables, do not use the DSQ prefix.
4. In the **Variable value** field, enter the value for your variable.
5. Optional: In the **Variable description** field, type a description of the variable.
6. In the **Variable lifetime** field, specify whether the variable exists for the current QMF session or permanently.
7. Press Enter to create the global variable.

Editing global variables

You cannot edit the names of system global variables; however, you can edit the default values of both system global variables and user-defined global variables.

Procedure

1. On the command line, enter SHOW GLOBALS.
2. On the **Globals** panel, position the cursor on the variable to edit. Press the **Show Field** function key.
3. On the **Show Global Variable** panel, edit the **Variable name** and **Variable value** fields.
4. Optional: In the **Variable description** field, edit the description.

   Note: You can edit the description only of user-defined global variables whose LIFETIME parameter is set to PERMANENT.
5. In the **Variable lifetime** field, specify whether the variable exists only during the current QMF session or permanently.
6. Press Enter to save the changes.

Deleting global variables

You can delete user-defined global variables.

Procedure

1. On the command line, enter SHOW GLOBALS.
2. On the **Globals** panel, position the cursor on the variable to delete. Press the **Delete** function key.

3. On the **Prompt** panel, select **Yes**. Press Enter to delete the variable.

Chapter 3. Accessing data

Repositories and data sources

To work with QMF Z Client, you must connect to a repository, which stores data sources and application objects.

A repository is a centralized storage area created by your QMF administrator. It is the place where your objects such as queries, procedures, forms, and reports can be saved. It is also where QMF will look for the information necessary to connect to any data sources that you need to access.

A data source stores the connection information that is required to access a database. In a repository, each data source is classified by the type of the database that it represents:

Hive
Data is stored in Apache Hive™ data warehouses. This kind of storage is designed for summarizing, querying, and analyzing of large volumes of data with the help of HiveQL, a language that is similar to SQL.

JavaScript
Data is provided by online services and stored in JavaScript tables.

QMF Data Service
Data is stored in tables on the QMF Data Service server.

Relational
Data is stored in interrelated tables. Each table comprises a number of columns and rows.

Virtual
Data is stored in virtual and JavaScript tables that collect information from different sources and present it as a single database. Virtual databases cache data from original databases so that you can work with it without referring to original data sources separately.

Note: In QMF Z Client, you can connect to existing repositories and use existing data sources. To create a repository or a data source, use QMF Server. For more information, see Installing and Managing QMF Server.

Connecting to repositories

In order to access a repository and run repository objects by using QMF Z Client, you must be connected to a repository.

About this task

To connect to a repository, complete the following steps:

Procedure

1. Click **File** > **Switch Repository**.
2. On the **Switch Repository** panel, select the repository to which you want to connect.

   Note: To view the properties of a repository connection, position the cursor on the repository and press the **Describe** function key.
3. Press Enter to connect to the specified repository.

   Note: If you are trying to connect to a secured repository, QMF prompts you to enter the user credentials for that repository.

Connecting to data sources

QMF data sources store data in database tables. Each database table comprises a number of columns and rows. Queries for QMF data sources are written in SQL.

Procedure

1. Click **File** > **Connect To**.
2. On the **Connect to** panel, select the data source to which you want to connect.
3. Press Enter to connect to the specified data source.

Accessing QMF objects

Use the **Object List** panel to access the list of QMF objects that are available to you in the current data source.

Procedure

1. On the command line, enter `LIST ALL`.
2. On the **Object List** panel, use the **Name**, **Type**, and **Owner** fields to filter the list and find the object that you want to work with.
3. To sort the list, press the **Sort** function key, specify the sort order that you want to apply, and press Enter.
4. To filter the list by date, use the **Created** and **Modified** fields. Use the following syntax: `[ >, <, =] N [d, m, y]`, where `N` is the number of days (`d`), months (`m`), or years (`y`). For example, enter `<5d` in the **Created** field to display the objects that were created fewer than five days ago.
5. In the **Action** field corresponding to the object, enter the command that you want to perform on the object or right-click the field to view the list of available commands. For more information about QMF commands, see Appendix C, “QMF Commands,” on page 65.

Saving QMF objects

You can save QMF objects to the database by using the Action bar. This is the equivalent of using the SAVE command or the SAVE AS command.

About this task

To save an object, complete the following steps:

Procedure

1. If the object that you are working with is already saved in the database and you only want to save the latest changes, click **File** > **Save** in the action bar and skip the remaining steps.
2. If you want to save an object to the database, click **File** > **Save As** in the Action bar.

   The **Command Prompt** panel opens.
3. In the **Object name** field, specify the name for the object. If the name includes spaces or mixed case symbols, make sure that you enclose the name in double quotation marks.
4. Optional: In the **Comment** field, specify a side note for the object.
5. Optional: Press the **Forward** function key to display the second half of the panel.
6. In the **Confirm** field, specify whether to display a confirmation dialog when saving the changes to the object or replacing it.
7. Optional: In the **Folder** field, specify the folder to which you want to save the link to the object.

   Note: QMF folders only contain links to the QMF objects that are stored in a database, but not the actual objects.

8. Optional: In the **Share** field, specify whether you want to make the saved object available to other users. Valid values are YES and NO.
9. Press Enter to save the object.

Working with folders

In QMF Z Client, workspace folders store objects or other folders, while QMF Catalog folders store links to the objects.

About this task

Typical QMF workflow suggests the following order of operations when working with folders:

Procedure

1. On the command line, enter `list folders` to access the list of folders that are available on the current data source.
2. To open a folder, enter `sel` in the **Action** field near the folder.
3. In the **HOME:/<location>/<folder name>** field, click the folder name to return to parent folder or click the location name to display the list of objects for the location.

Creating folders

Create a workspace folder to store QMF objects and other folders.

About this task

To create a workspace folder, complete the following steps:

Procedure

1. On the command line, type `CREATE FOLDER ?` and press Enter to open the CREATE FOLDER command prompt.
2. In the **Folder Name** field, specify the name for the new folder.
3. In the **Comment** field enter a text that you want to associate with the folder.
4. In the **Folder** field, specify the parent workspace folder for the folder that you are creating. To view the list of available parent folders, position the cursor on the field and press **List**.
5. Press Enter to create the folder.

Chapter 4. Working with data

Working with queries

To request information from a relational data source, use the SQL editor or the prompted query editor to create a query.

The following topics describe working with QMF queries:

Creating queries using SQL editor

Use the SQL editor to create and run queries against relational data sources.

Procedure

1. To open the query editor, type `CREATE QUERY` on the command line. Press Enter.
2. Position the cursor in the editor area.

   Note: Maximum of 600000 number of rows are allowed in an editor. Documents upto size 2 MB are allowed in an editor.
3. Type one or more SQL statements. Use a `;` (semicolon) to separate multiple statements.

   Note: To insert, remove, copy, and reposition the lines in the editor area, see Appendix E, "SQL editor line commands," on page 129.
4. Press the **Run** function key to run the query and display the result set.
5. Once the query result set is displayed, you have the following options:

   Note: If the query contained multiple SQL statements, click **Query** > **Specify Result Set** to display a specific result set.

Creating queries using prompted query editor

To create a query without typing SQL statements, use the prompted query editor.

Procedure

1. On the command line, type `RESET QUERY(LANG=PROMPTED` and press Enter.
2. On the **Tables** panel, complete the following steps to specify one or more tables to add to the query:
   a) In the **Table owner** field, specify the owner of the table that you want to work with.
   b) In the **Table name** field, specify the name of the table that you want to work with.

      Note: To view the list of all tables that belong to the specified owner, press the **List** function key.
   c) Press the **Add** function key to add the table to the query.
   d) Repeat the procedure for each table to include in the query and press the **Cancel** function key to save the changes.

   Each time that you specify an additional table, you use the **Joins** panel to specify the joining options.
3. Optional: To customize the list of columns that are included in the query result set, complete the following steps:
   a) Position the cursor in the **Columns** area and press the **Insert** function key.
   b) On the **Columns** panel, press the **List** function key to view the list of available table columns.
   c) On the **Column List** panel, position the cursor on the column to include in the query result set and press the **Add** function key.

d) Repeat the previous step for each column to include in the result set.

e) Press the **Cancel** function key to save the changes.

4. Optional: To specify row conditions for the query, complete the following steps:
   a) On the main editor panel, press the **Switch** function key to display the **Row Conditions** and **Sort Conditions** areas.
   b) Position the cursor in the **Row Conditions** area and press the **Insert** function key.
   c) On the **Row Conditions** panel, select the column whose rows to filter or enter an expression in the **Expression** field. Press Enter.
   d) On the **Comparison Operators** panel, specify the comparison operators that you want to use. Press Enter.
   e) On the next panel, specify the values for the comparison operator that you have selected. Press Enter to save the changes.
5. Optional: To specify sort conditions for the query, complete the following steps:
   a) Position the cursor in the **Sort Conditions** area and press the **Insert** function key.
   b) In the **Order** field on the **Row Conditions** panel, specify the sort order that you want to apply to the query result set.
   c) In the **Select column or enter expression** field, select the column by which to filter the result set or enter an expression. Press Enter to save the changes.
6. Press the **Run** function key to run the query.

Running existing queries

You can access the list of existing QMF queries to re-run, edit, or delete each one.

About this task

This topic describes running existing queries manually. To run a query unattended, that is without interacting with the application, use QMF in batch mode. For more information about using QMF in batch mode, see “Working with batch objects” on page 30.

Procedure

1. On the command line, type `LIST QUERIES` and press Enter.
2. On the **Object List** panel, use the **Name** and **Owner** fields to filter the list and find the query that you want to work with.
3. In the **Action** field corresponding to the query, access the context menu and select one of the following actions:

   Run
   Runs the query.

   Display
   Displays the query.

   Edit
   Opens the query editor where you can edit the query.

   Add To Favorites
   Adds the query to the list of favorites.

   Describe
   Opens the panel where you can view the query metadata and enter a comment.

   Rename
   Opens the panel where you can rename the query.

   Erase
   Deletes the query.

Creating reports

After you run a query or display a table, use the Form Editor to create a report that is based on the result set.

About this task

Use the Form Editor panels to configure different aspects of your report. The **Form.Main** panel allows you to specify general preferences for your report. Other Form panels allow you to specify the detailed preferences. To display a particular Form panel, click **View** and select the panel that you want to work with. The following Form panels are available:

Form.Break
Specify the break options for the report. You can configure up to 6 break levels for your report and specify distinct break options for each level. Specify each set of break level options on the corresponding Form.Break panel (Form.Break1 to Form.Break6).

Form.Calculations
Specify calculations for the report.

Form.Columns
Work with the columns that you want to include in the report.

Form.Conditions
Specify conditional expressions for the report.

Form.Detail
Specify detail block options for the report.

Form.Final
Specify the text to display at the end of the report.

Form.Options
Specify detailed formatting options for the report.

Form.Page
Specify headers and footers for the pages of the report.

Procedure

1. On the command line, type `CREATE FORM` to open the Form Editor for the Form.Main panel, where you can specify general preferences for your report.

   If you previously specified the data source object for the report, skip step "2" on page 23 and continue with step "3" on page 23.
2. To specify the data source object for the report, complete the following steps:
   a) Click **Form** > **Data Source Object**.
   b) On the **Data Source Object** panel, specify whether to use an object from a repository or from a data source. Press Enter.
   c) Specify the object that you want to work with, and press Enter.
3. In the **Num** field, view the order in which the columns are arranged in the query result set.
4. In the **Column heading** field, enter headings for the columns of the report. By default, column headings come from the result set.
5. Optional: In the **Usage** field, enter a usage code for each column. For more information about QMF usage codes, see Appendix F, "QMF usage codes," on page 131.
6. Optional: In the **Indent** field, enter the number of spaces to insert before the column. The default value comes from the result set.
7. Optional: In the **Width** field, enter the width of the column. The default value comes from the result set.
8. In the **Edit** field, enter an edit code for the column. For more information about QMF edit codes, see Appendix G, "QMF edit codes," on page 133.

9. In the **Seq** field, specify the order for the columns in the report.
10. In the **Page heading** and **Page footing** fields, specify the text for the header and the footer of the report.
11. In the **Final text** field, enter the text to display at the end of the report.
12. In the **Break 1** and **Break 2** fields, enter the text to place in the report breaks.
13. In the **Options** field, use the **Outline** check box to specify whether to enable the outlining option for the report.
14. Use the **Default break text** check box to specify whether to put the default text at break levels of the report. The default break text is a string of 1-6 asterisks (*).

    The **Form.Main** panel allows you to specify general preferences for your report. Other Form panels allow you to specify the detailed preferences. To display a particular Form panel, click **View** and select the panel that you want to work with. The following Form panels are available:

    Form.Break
    Use this panel to specify the break options for your report. You can configure up to 6 break levels for your report and specify distinct break options for each level. Each break level options can be specified on the corresponding Form.Break panel (Form.Break1 to Form.Break6).

    Form.Calculations
    Use this panel to specify calculation expressions for your report.

    Form.Columns
    Use this panel to work with the query result set columns that you want to include in your report.

    Form.Conditions
    Use this panel to specify conditional expressions for your report.

    Form.Detail
    Use this panel to specify detail block options for your report.

    Form.Final
    Use this panel to display at the end of the report.

    Form.Options
    Use this panel to specify the detailed formatting options for your report.

    Form.Page
    Use this panel to specify heading and footing for the pages of your report.

Working with procedures

Use a procedure to execute a series of QMF commands within a single RUN command, call other applications, and start QMF in batch mode.

The following topics describe working with procedures:

Creating procedures

To create a procedure that executes a series of QMF commands, use the procedure editor.

Before you begin

If the procedure includes running an object, make sure to create the object and save it before you start working on your procedure.

Procedure

1. On the command line, type CREATE PROC and press Enter.
2. On the **Editor** panel, type one or more QMF commands.

Note: If the command is too long to fit on one line, finish the line with the + character and continue the command on the next line. Consider the following example:

```
show
+query
```

Maximum of 600000 rows are allowed in the editor.

Note: Text upto 2 MB size is allowed in the editor.

3. Press the **Run** function key to run the procedure.

Working with existing procedures

You can access the list of procedures and run or edit each one.

Procedure

1. On the command line, type `LIST PROC` and press Enter.
2. On the **Object List** panel, use the **Name** and **Owner** fields to filter the list and find the procedure that you want to work with.
3. In the **Action** field corresponding to the procedure, access the context menu and select the action to perform on the procedure.

   Run
   Runs the procedure.

   Display
   Displays the procedure.

   Edit
   Opens the procedure editor where you can edit the procedure.

   Add To Favorites
   Adds the procedure to the list of favorites.

   Describe
   Opens the panel where you can view the procedure metadata and enter a comment.

   Rename
   Opens the panel where you can rename the procedure.

   Erase
   Deletes the procedure.

   Note: To run a procedure unattended, that is without interacting with the application, use QMF in batch mode. For more information about using QMF in batch mode, see “Working with batch objects” on page 30.

Procedures with logic

Standard procedures can execute a series of QMF commands. Procedures with logic enable you to add REXX programming statements and functions along with QMF commands to develop your own applications.

What is supported

The following features are supported:

- ADDRESS QRW command environment

  A command environment that allows execution of QMF commands while running REXX procedures. For more information, see “ADDRESS QRW and the QMF command environment” on page 27.

- REXX variables

You can assign REXX variables to QMF global variables and QMF global variables to REXX variables. For more information, see “REXX variables in procedures with logic” on page 28.

Note:

For QMF TSO users:

The following features available in QMF TSO are not currently supported in QMF Z Client.

- Callable interface
- Command interface
- Command synonyms

Creating REXX procedures

About this task

The steps for creating a procedure with logic are similar to that documented under Working with procedures. The only differences are:

- The first line of the procedure must indicate the start of a REXX procedure.
- Commands directed to QMF must be preceded by the ADDRESS QRW statement.
- The QMF commands must be enclosed in double quotes.

Example 1

```
/* REXX */
      REXX statements
      .
      .
      ADDRESS QRW
      qmf_command_1
      qmf_command_2
      .
      .
      REXX statements
      .
      .
```

Example 2

```
/* REXX */
      SAY "HI, THIS IS A REXX SCRIPT"
      ADDRESS QRW
      "RUN PROC EXAMPLE_PROCEDURE"
      SAY "REXX SCRIPT IS DONE"
```

Example 3

This example shows how to export a dataset. It adds REXX logic to check the day of the week and to print weekly reports on Mondays.

Create the following procedure EXAMPLE_PROCEDURE:

```
-- MONDAY MORNING REPORT.
-- PROCEDURES MAY CONTAIN COMMENT LINES; THEY BEGIN
-- WITH TWO HYPHENS.
-- A TITLE OR IDENTIFIER AT THE BEGINNING IS USEFUL.

 RUN QUERY MYQUERY (FORM=MYFORM
-- THIS COMMAND RUNS YOUR QUERY AND FORMATS THE REPORT.

 SAVE DATA AS LASTWEEKDATA (CONFIRM=NO
-- THIS COMMAND SAVES YOUR DATA IN A TABLE AND OVERRIDES THE VALUE OF
-- CONFIRM IN YOUR PROFILE FOR THE DURATION OF THE COMMAND.

 EXPORT TABLE LASTWEEKDATA TO LASTWEEK
-- THIS COMMAND EXPORTS THE DATA THAT WAS SAVED IN THE PREVIOUS COMMAND
-- TO THE DATASET 'LASTWEEK'.
```

```
-- PROCEDURE HAS FINISHED.
```

Create a REXX procedure that calls the procedure EXAMPLE_PROCEDURE. The REXX procedure adds logic for checking the day of week.

```
/* REXX */
/* This procedure checks to see what day it is.  If it's
    Monday, it runs a query and saves result to dataset.  If it
    isn't, a message is displayed informing the user.   */

ADDRESS QRW
signal on error
if date('w') = 'Monday' then
   do
     "RUN PROC EXAMPLE_PROCEDURE"
   end
else
   do
     SAY "Sorry, it is not Monday.  Report cannot be created."
   end
exit 0       /*Exit without errors */
error:
exit 8      /*Exit with error condition*/
*** END ***
```

ADDRESS QRW and the QMF command environment

The QMF Z Client creates a command environment called QRW to allow execution of QMF commands from REXX procedures. When you are executing a REXX program, you can set the default command environment to QRW by issuing the following REXX ADDRESS command:

```
ADDRESS QRW
```

With ADDRESS QRW, QMF remains the default command environment until you issue another ADDRESS command.

You can also direct a single command to be executed by the QRW environment by issuing the REXX ADDRESS command followed by the QMF command:

```
ADDRESS QRW qmf_command
```

In this situation, QMF is the command environment only for the command that follows the ADDRESS QRW statement. For example,

```
ADDRESS QRW RUN PROC EXAMPLE_PROCEDURE
```

When you are using a QMF procedure with logic, TSO is the default command environment. You need to explicitly set it to the QRW environment by using the ADDRESS QRW instruction.

The following example shows how to use the QMF command environment when issuing multiple QMF commands:

```
/* REXX */
      SIGNAL ON ERROR
      .
      ADDRESS QRW
      "RUN PROC EXAMPLE_PROCEDURE"
      "EXPORT PROC EXAMPLE_PROCEDURE TO EXAMPLE_DATASET"
      .
      EXIT(0)
      ERROR:
      EXIT RC
```

REXX variables in procedures with logic

REXX variables can be used in procedures with logic. The values for these variables are known only within the procedure in which you defined them. There are three ways in which the variables can be used:

- Copy a REXX variable to a QMF variable with the SET GLOBAL command.

```
/* REXX */
      RUN_MT = 1
      ADDRESS QRW
      "SET GLOBAL (DSQEC_RUN_MQ=RUN_MT"
```

- Copy a global variable to a REXX variable with the GET GLOBAL command.

```
/* REXX */
      ADDRESS QRW
      "GET GLOBAL (QMF_REL=DSQAO_QMF_RELEASE"
      SAY QMF_REL
```

- Use REXX variables in your REXX statements.

In a procedure with logic, you can use REXX SAY and PULL statements to prompt for variable values.

Use a SAY statement (or a sequence of SAY statements) to display text on the screen. The following example shows some sample SAY statements:

```
say 'Hello,' whoisuser'.'
say 'Please enter the letter of the weekly report you would like, '
say 'or NONE to exit:'
say
say '                A. Sales results (Monday Only)'
say '                B. Tax figures'
say '                C. Cumulative salaries'
```

When you use these SAY statements, the output shown is as follows:

```
Hello, username.
Please enter the letter of the weekly report you would like,
or NONE to exit:

          A. Sales results (Monday Only)
          B. Tax figures
          C. Cumulative salaries
```

Specify a REXX PULL statement to retrieve the input and place it in the REXX variable answer as shown in the following example.

```
/*    This procedure can produce any of three weekly reports
      regularly produced by the Acme Company (Sales Results,
      Tax Figures, or Cumulative Salaries). It prompts the user
      for the type of report required, runs the necessary
      queries, and checks for errors.                      */

/* REXX */
ADDRESS QRW
arg report .               /* get any arguments from RUN PROC   */
ok = 'NO'                    /*   set variable for do loop        */
"GET GLOBAL (WHOISUSER = DSQAO_CONNECT_ID"     /*  identify user  */

if report = '' then          /*  check to see if no arg entered   */

    /* if no arg entered, prompt user until A,B,C, or NONE is entered */
  do until ok = 'YES'

    say 'Hello,' whoisuser'.'
    say 'Please enter the letter of the weekly report you would like, '
    say 'or None to exit:'
    say
    say '                A. Sales results (Monday Only)'
    say '                B. Tax figures'
    say '                C. Cumulative salaries'

    pull answer                /*   get answer from user     */
    answer = strip(answer)   /* strip any leading or trailing blanks  */
```

```
    if answer = 'NONE' then exit 3     /*  exit immediately if NONE   */
    if pos(answer,'ABC') ¬= 0 then ok = 'YES'  /* if invalid value,   */
  end                                 /* keep prompting.     */
else answer = report
```

An exit code of 3 was selected here to indicate the exit condition when the user enters None. As with any exit code, you choose the number to indicate an exit condition.

Working with database tables

Use SQL to view, edit, save, erase, or export a database table.

Procedure

1. On the command line, type LIST TABLES and press Enter.
2. On the **Object List** panel, use the **Name** and **Owner** to filter the list and find the table that you want to work with.
3. In the **Action** field, access the context menu and select one of the following actions:

   Display
   Displays the table on the **Results** panel where you can view it or use it to create a query or report. For more information about creating queries, see “Creating queries using SQL editor” on page 21. For more information about creating reports, see “Creating reports” on page 23.

   Edit
   Opens the table editor where you can edit the table. For more information about editing tables, see the Editing database tables topic.

   Add To Favorites
   Adds the table to the list of favorites.

   Describe
   Opens the panel where you can view the table metadata and enter a comment.

   Rename
   Opens the panel where you can rename the table.

   Erase
   Deletes the table.

Editing database tables

You can edit database tables that are accessible to you in your data source.

About this task

To edit a database table, complete the following steps:

Procedure

1. On the command line, type EDIT *t_owner.t_name*, where *t_owner* is the name of the table owner and *t_name* is the name of the table. Press Enter.

   Note: If the table that you want to edit belongs to the user account under which you are currently logged in, you can omit the table owner from the command.
2. Optional: By default, every edit that you make is automatically saved and committed. To make multiple edits and avoid unwanted commits, click **Table** > **Disable Immediate Commit**.

   Note: Uncommitted changes to the table are marked with the * (asterisk) character.
3. Optional: To quickly find the row that you want to work with, complete the following steps:

   a) Press the **Search** function key.

b) On the **Search** panel, specify the search information for the row that you want to work with and press Enter.

4. To edit a row, complete the following steps:
   a) Position the cursor on the row and press the **Change** function key.
   b) On the **Edit Row** panel, make the necessary edits and press Enter.
5. To insert a new row, complete the following steps:
   a) Press the **Add** function key.
   b) On the **Add Row** panel, enter the appropriate information in each cell and press Enter.
6. To remove a row, position the cursor on the row and press the **Delete** function key.
7. If you have disabled the **Immediate Commit** option in step "2" on page 29, you have the following options when you are done editing the table:
   - To save the edits, click **Table** > **Commit**.
   - To cancel the edits, click **Table** > **Roll Back**.

Working with batch objects

A batch object is a set of parameters that creates a JCL batch job, which you use to run QMF queries and procedures in background mode.

The following topics describe working with batch objects:

Creating batch objects

Use the batch wizard to create a batch object.

About this task

To create a batch object, complete the following steps:

Procedure

1. On the command line, type BATCH and press Enter.
2. Press the **Add** function key.
3. On the **Batch Wizard - Main Parameters** panel, complete the following steps:
   a. In the **Batch object name** field, enter a name for the object.
   b. In the **Batch PROC name** field, specify the full path to the batch procedure that you want to use.
   c. Specify whether to create an object or use an existing one:
      - To create a batch procedure for a query, select **Create batch PROC for QUERY**. Then continue with step 4.
      - To create a batch procedure for a procedure, select **Create batch PROC for PROC** option. Then continue with step 5.
      - To use an existing batch procedure, select the **Use existing batch PROC** option.
   d. Press the **Next** function key to open the next panel of the wizard.
4. On the **Batch Wizard - Parameters for QUERY**, complete the following steps:
   a. In the **QUERY name** field, enter the name of the query to use for the batch object. To use the query that is currently open in the editor, select **Use query from work area**. Note that the work area can contain several open objects. If you select the **Use query from work area** option, the most recently opened query is used. Also note, that if the **Use query from work area** check box is selected, the currently open query is saved with the name that is specified in the **Object name** field.
   b. In the **FORM name** field, enter the name of the form to use for your batch object. To use the form that is currently open in the editor, select **Use form from work area**. Note, that the work area can

contain several open objects. If you select the **Use form from work area** option, the most recently opened form is used. Also note, that if the **Use form from work area** check box is selected, the currently open form is saved with the name that is specified in the **Object name** field.

c. In the **TABLE name to save result DATA** field, specify the name for the results file and the full path to the location where you want to save it.

d. Press the **Next** function key to open the next panel of the wizard.

5. On the **Batch Wizard - Parameters for PROC** panel, complete the following steps:

   a. In the **PROC name** field, specify the name of the procedure that you want to use for your batch object. If you want to use the procedure that is currently open in the editor, select the **Use procedure from work area** option. Note, that the work area can contain several open objects. If you select the **Use procedure from work area** option, the most recently opened procedure is used.

   b. Press the **Next** function key to open the next panel of the wizard.

   The **Batch Wizard - REPORT Parameters** panel opens. This panel allows you to specify the email address to which you want to send your report. If you do not want to send your report in an email, press the **Next** function key and go to step 8.

6. On the **Batch Wizard - REPORT Parameters** panel, complete the following steps:

   a. In the **Emails to send REPORT** field, specify one or more email addresses to which you want to send your report.

   b. In the **From** field, specify the email address of the sender.

   c. In the **Subject** field, type the subject of your email.

   d. In the **Report type** field, specify the format to which you want to convert your report before sending the email. Valid values are: TEXT, PDF, and HTML. If you leave the field blank, the report is automatically converted to the text format.

   e. Press the **Next** function key to open the next panel of the wizard.

7. On the **Batch Wizard - SMTP Settings** panel, complete the following steps:

   a. In the **SMTP server** field, type the address of the SMTP server to use.

   b. In the **Port** field, type the number of the server port to use.

   c. In the **User** and **Password** fields, specify your QMF Z Client user credentials.

   d. Press the **Next** function key to open the next panel of the wizard.

8. On the **Batch Wizard - Common Parameters** panel, complete the following steps:

   a. If you want to run a batch job for another user, use the **TSO login for batch job** field and the **TSO password for batch job** field to specify login information of the user for which you want to run the batch job.

   b. Use the **Login to database** field and the **Password to database** field, specify the user credentials that you use to connect to the database that you want to work with.

   c. In the **Name of repository** field, see the name of the repository that you are working with.

   d. Use the **Login to repository** and **Password to repository** fields to specify the login information for the repository.

   e. Use the **Name of data source**, **Login to data source**, and **Password to data source** fields to specify the data source that you want to work with and the login information for it.

   f. Press Enter to create your batch object.

Working with batch objects

Run, edit, or remove existing QMF batch objects.

Procedure

1. On the command line, enter BATCH and press Enter.

2. On the **Batch List** panel, position the cursor on the batch object that you want to work with and press one of the following function keys:

   Submit
   Runs the specified batch object.

   Edit
   Opens the specified object in Batch Wizard where you can edit it. Editing an object is similar to creating an object. For more information about creating batch objects, see “Creating batch objects” on page 30.

   Add
   Creates a batch object. For more information about creating batch objects, see “Creating batch objects” on page 30.

   Remove
   Deletes the specified batch object.

   JCL Export
   Exports the specified batch object to a TSO data set or a UNIX file.

Chapter 5. The callable interface and QMF Z Client applications

The callable interface and QMF Z Client applications

QMF Z Client callable interface provides a mechanism using which you can develop your own programs for running QMF commands.

What is the callable interface?

The QMF Z Client callable interface provides standard interfaces for different programming languages.

When an application program needs to run a QMF command, it must start communication between the program and QMF. This communication is made by issuing a call to a QMF interface routine.

The application program can issue one or more QMF commands after the initial START call. The application program can call the routine to issue QMF commands.

After the QMF command finishes processing, QMF provides a return code that indicates the status of QMF command execution. The callable interface gathers other information about the processing of the command and stores this information in variables accessible to both QMF and the application program. These variables are contained in an *interface communications area*. When the callable interface returns control to the calling application program, the application can refer to these variables but not alter them.

When the application program no longer needs to use QMF, the program should issue a call to terminate communication between the program and QMF. QMF Z Client provides a routine to terminate this communication.

Considerations for using the QMF callable interface

Follow the guidelines provided below, when you write application programs to be used with the QMF Z Client callable interface:

- A call to QMF returns control to the calling application program only after QMF finishes processing the QMF command.
- The application program and QMF communicates using the interface communication area.
- All QMF commands must be coded in uppercase English letters.

This diagram shows how the application passes commands through the callable interface to QMF.

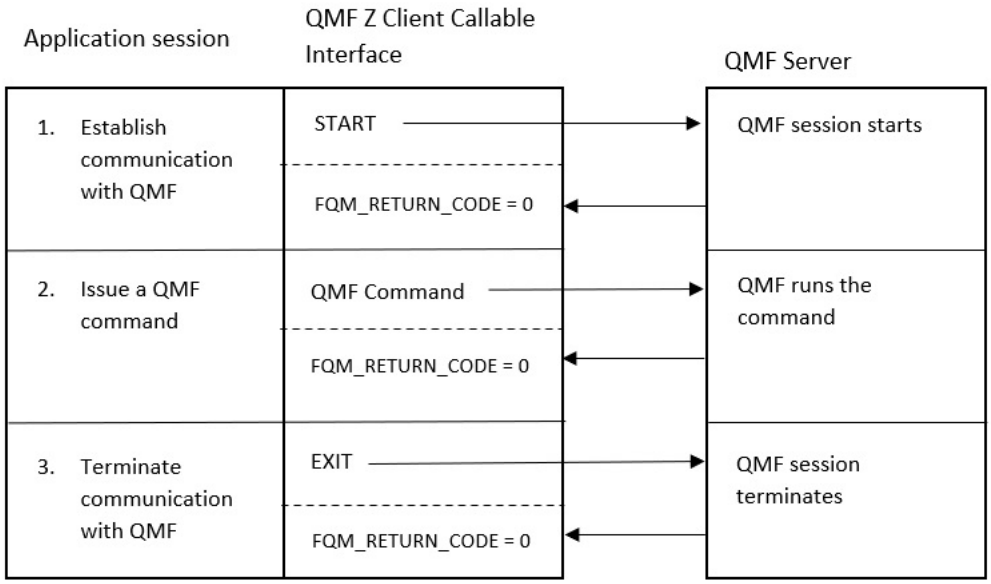

Figure 1. QMF Z Client Callable Interface

The interface communications area (FQMCOMM)

QMF Z Client provides an interface communications area for each supported programming language. This area contains definitions of return and reason codes.

The interface communications area defines storage for the interface communications variables. The variables stored in this area are accessible to both QMF and the calling application program. However, only QMF can alter the values. Ensure that the application program treats these variables as read-only.

The QMF callable interface communications area is required for all routine calls. Storage for the callable interface communications area is allocated by the program that is utilizing QMF.

The START command establishes a unique instance or occurrence of a QMF session. The START command can establish only one QMF session.

When running the START command, QMF updates the variables within the interface communications area.

All calls that follow the START command must pass the address of the interface communications area that corresponds to the QMF instance. The application program is responsible for pointing to the correct interface communications area.

Each supported programming language has a unique interface communications area. Application programs must reference variables by variable name rather than by value, if they are to be portable, because the values can be different on the other systems.

The variables within the interface communications area contain the information shown in this table:

Table 2. FQMCOMM fields that must not be altered

Information provided by the variable	**Description**
Return code	Indicates the status of QMF processing after QMF processes a command.
Instance identifier	Identifies the instance of QMF that was started by the START command

Table 2. FQMCOMM fields that must not be altered (continued)	
Information provided by the variable	**Description**
Completion message ID	Contains the message ID of the message that QMF returns. This field is set after the completion of every QMF command. It contains the message QMF displays at the end of a command.
Query message ID	Not populated.
START command parameter in error	Not populated.
Cancel indicator	Indicates whether the user canceled processing while QMF was running the command.
Completion message	Contains the completion message that QMF returns.
Query message	Not populated.

Return codes

Return codes are returned after each call to the QMF Z Client . Return code values are described by the interface communications area provided by QMF.

The values of return codes can be different on other systems. If you want your applications to be portable across systems, the applications must reference the values of these codes by the variable names. The names of the return-code variables within the interface communications area are documented with the programming language specification.

This table shows the possible return codes for callable interface conditions.

Table 3. Callable interface return codes	
Value	**Explanation**
0	Successful execution
4	QMF session marked for termination by an EXIT or END command
8	Execution failed, but the error did not mark the session for termination
16	Severe error: session marked for termination

Commands for using the callable interface

You can use the callable interface to issue any QMF command that you would use in a procedure. However, some commands have special syntax for the callable interface: START and SET GLOBAL.

The START command work only in the callable interface. To use the SET GLOBAL commands in a callable interface application use the extended syntax for these commands.

For examples of the START and SET GLOBAL commands in a programming language, see the specification for that language. For more information, see chapter, *Programming language specifications for using the callable interface*.

Running your application program

You can run your application program in Interactive or Batch mode.

In either mode, you can set your program variables. For more information on program variables, please see chapter, *Programming language specifications for using the callable interface.*

For information about setting up your environment and compiling and running your application program, see the coding sample in the appropriate language specification.

Error handling

At the completion of every QMF command, the FQMCOMM communications area contains message text in the fqm_message_text variable and a return code in the fqm_return_code variable.

The return code is assigned one of the following values:

fqm_success
Successful completion of the command

fqm_warning
Completion with warnings

fqm_failure
Command did not run correctly

fqm_severe
Severe error; QMF session is stopped

The variables and fields in each FQMCOMM area are documented with the programming language specifications.

Chapter 6. Programming language specifications for using the callable interface

Introduction

The QMF application program interface is available for several programming languages. This chapter provides information about how to assemble (or compile) and link-edit the programs and how to run them using the QMF application program interface.

C language interface

You can use the C language with the callable interface.

Interface communications area mapping for C language (FQMCOMMC)

FQMCOMMC provides FQMCOMM mapping for C language programs.

The following table provides specifications for Interface communications area mapping for C language (FQMCOMMC).

Table 4. Interface communications area for FQMCOMMC

Structure field	Data type	Description
FQM_RETURN_CODE	signed long integer	Indicates the status of a QMF command after it is run. Its values are: **FQM_SUCCESS** Successful run of the request **FQM_WARNING** Normal completion with warnings **FQM_FAILURE** Command did not run correctly **FQM_SEVERE** Severe error; QMF session stopped
FQM_INSTANCE_ID	signed long integer	Identifier established by QMF during running of the START command
FQM_COMM_LEVEL	character, length 12	Identifies the version of the FQMCOMM structure. In your application, initialize this variable to the value of FQM_CURRENT_COMM_LEVEL before issuing the QMF START command.
FQM_PRODUCT	character, length 2	Identifies the IBM query product in use. Initialize the value of this variable to FQM_QMF.
FQM_PRODUCT_RELEASE	character, length 2	Version of QMF in use. Variable FQM_QMF_V12R2 specifies QMF Version 12 Release 2.

Table 4. Interface communications area for FQMCOMMC (continued)		
Structure field	**Data type**	**Description**
FQM_RESERVE1	character, length 28	Reserved for future use
FQM_MESSAGE_ID	character, length 8	Completion message ID
FQM_Q_MESSAGE_ID	character, length 8	Not populated
FQM_START_PARM_ERROR	character, length 8	Not populated
FQM_CANCEL_IND	character, length 1	Contains one of two values, depending on whether the user canceled while a QMF command was running: • FQM_CANCEL_YES • FQM_CANCEL_NO
FQM_RESERVE2	character, length 23	Reserved for future use
FQM_RESERVE3	character, length 156	Reserved for future use
FQM_MESSAGE_TEXT	character, length 128	Completion message text
FQM_Q_MESSAGE_TEXT	character, length 128	Not populated

Function calls for the C language

QMF provides two function calls for the C language: FQMCIC and FQMCICE.

FQMCIC

This call is for the QMF commands that do not require access to application program variables. Use this call for most of the QMF commands; its syntax is as follows:

```
FQMCIC (&FQMCOMM,&CMDLTH,&CMDSTR)
```

The parameters have the following values:

FQMCOMM
The interface communications area

CMDLTH
Length of the command string (CMDSTR); a long type parameter

CMDSTR
The QMF command to run, specified as an array of unsigned character type of the length specified by CMDLTH

The QMF command must be in uppercase.

FQMCICE

This call has an extended syntax for the QMF commands that require access to application program variables. The START and SET GLOBAL commands use the extended syntax.

```
FQMCICE (&FQMCOMM,&CMDLTH,&CMDSTR,
         &PNUM,&KLTH,&KWORD,
         &VLTH,&VALUE,&VTYPE);
```

The parameters have the following values:

FQMCOMM
The interface communications area.

CMDLTH
Length of the command string (CMDSTR); a long integer parameter.

CMDSTR
QMF command to run; an array of unsigned character type. The QMF command must be in uppercase.

PNUM
Number of command keywords; a long integer parameter.

KLTH
The length of each specified keyword (KWORD); a long integer parameter or an array of long integer parameters.

KWORD
QMF keyword, keywords; each is a character, array of characters.

VLTH
The length of each value that is associated with the keyword; a long integer parameter or array of long integer parameters.

VALUE
The value that is associated with each keyword.

Its type is specified in the VTYPE parameter and can be an unsigned character array, a long integer parameter, or array of long integer parameters.

VTYPE
Data type of the contents of the VALUE parameter.

This parameter has one of the two values, which are provided in the interface communications area, FQMCOMMC:

- FQM_VARIABLE_CHAR for unsigned character type
- FQM_VARIABLE_FINT for long integer

All of the values that are specified in the VALUE field must have the data type that is specified by the VTYPE.

The C language interface has the following parameter considerations:

- Command strings and the START and SET command parameters are all input character strings. With these strings, C requires you to pass a storage area that ends with a null value, the null termination character must be included in the length of the parameter. Use the compile-time length function to obtain the parameter length that is passed to the QMF interface.
- If the string does not end by a null value before reaching the end of the string, an error is returned by QMF. The null value (X'00') indicates the end of a character string.

C language programming example

The following sample program for the C language application program performs the following functions:

- Starts QMF
- Sets three global variables

- Runs a query called Q1
- Ends the QMF session

QMF does not provide query Q1, but the sample program uses this object.

```
/*
 * Sample Program
 * C version of the callable interface
 */

#include <stdio.h>
#include <stdlib.h>

/*
 * Include and declare query interface communications area
 */
#include "FQMCOMMC"

#define    NUM_KEYWORDS    4
#define    SIZE_KEYVAL     32
#define    SET_KEYWORDS    3
#define    SIZE_VAL        8

int main(int argc, char** argv)
{
    struct fqmcomm communication_area;    /* FQMCOMM from include */

    /*
     * Query interface command length and commands
     */
    static char start_query_interface[] = "START";
    static char set_global_variables[] = "SET GLOBAL Lifetime=Permanent";
    static char run_query[] = "RUN QUERY Q1";
    static char end_query_interface[] = "EXIT";

    signed long command_length;
    signed long number_of_parameters;             /* number of variables */

    signed long keyword_lengths[NUM_KEYWORDS];    /* lengths of keyword names */
    signed long data_length[NUM_KEYWORDS];        /* lengths of variable data */

    /*
     * Variable data type constants
     */
    static char char_data_type[] = FQM_VARIABLE_CHAR;
    static char int_data_type[]  = FQM_VARIABLE_FINT;

    /*
     * Keyword parameter and value for START command
     */
    static char start_keywords[NUM_KEYWORDS][SIZE_KEYVAL] =\
    {"Mode"};

    static char start_keyword_values[NUM_KEYWORDS][SIZE_KEYVAL] =\
    {"I"};

    /*
     * MAIN PROGRAM
     */

    /*
     * Start a query interface session
     */
    strncpy(communication_area.fqm_comm_level,
            FQM_CURRENT_COMM_LEVEL,\
            sizeof(communication_area.fqm_comm_level));

    number_of_parameters = 1;
    command_length = sizeof(start_query_interface);

    keyword_lengths[0] = SIZE_KEYVAL;
    data_length[0] = SIZE_KEYVAL;

    FQMCICE(&communication_area, &command_length,\
      &start_query_interface[0], &number_of_parameters, keyword_lengths,\
      &start_keywords[0][0], data_length, &start_keyword_values[0][0],\
      &char_data_type[0]);

    /*
     * Keyword parameter and values for SET command
```

```
     */
    char set_keywords[SET_KEYWORDS][SIZE_VAL];
    signed long set_values[SET_KEYWORDS];

    /*
     * Set numeric values into query using SET command
     */
    number_of_parameters = 3;
    command_length = sizeof(set_global_variables);

    strcpy(set_keywords[0],"MYVAR01");
    strcpy(set_keywords[1],"SHORT");
    strcpy(set_keywords[2],"MYVAR03");

    keyword_lengths[0] = SIZE_VAL;
    keyword_lengths[1] = SIZE_VAL;
    keyword_lengths[2] = SIZE_VAL;
    data_length[0] = sizeof(long);
    data_length[1] = sizeof(long);
    data_length[2] = sizeof(long);
    set_values[0] = 20;
    set_values[1] = 40;
    set_values[2] = 84;

    FQMCICE(&communication_area, &command_length,\
      &set_global_variables[0], &number_of_parameters, keyword_lengths,\
      &set_keywords[0][0], data_length, set_values, &int_data_type[0]);

    /*
     * Run a query
     */
    command_length = sizeof(run_query);
    FQMCIC(&communication_area, &command_length, &run_query[0]);

    /*
     * End the query interface session
     */
    command_length = sizeof(end_query_interface);
    FQMCIC(&communication_area, &command_length,\
      &end_query_interface[0]);

    return 0;
}
```

FQMCOMM for C

The interface communications area of the C language is defined below.

```
/* C include for query callable interface */

/*  Structure declaration for callable interface communication area */
struct fqmcomm
{
    long int fqm_return_code;        /* Function return code */
    long int fqm_instance_id;        /* ID established in START cmd */
    char fqm_comm_level[12];         /* Communications level id */
    char fqm_product[2];             /* Query product id */
    char fqm_product_release[2];     /* Query product release */
    char fqm_reserve1[28];           /* Reserved */
    char fqm_message_id[8];          /* Completion message ID */
    char fqm_q_message_id[8];        /* Query message ID */
    char fqm_start_parm_error[8];    /* Start parameter in error */
    char fqm_cancel_ind[1];          /* Cmd cancelled indicator, 1 = cancelled,\
                                        0 = not cancelled */
    char fqm_reserve2[23];           /* RESERVED AREAS */
    char fqm_reserve3[156];
    char fqm_message_text[128];      /* Message text */
    char fqm_q_message_text[128];    /* Query message text */
};


/* RETURN CODES */

#define FQM_SUCCESS                0
#define FQM_WARNING                4
#define FQM_FAILURE                8
#define FQM_SEVERE                16

/* Communications Level */
```

```
#define FQM_CURRENT_COMM_LEVEL    "FQML>001202<"

/* Query Product Codes */

#define FQM_QRW                    "01"
#define FQM_QFM                    "02"
#define FQM_QM3                    "03"

/* Query Product Release Levels */

#define FQM_QMF_V12R2             "12"
#define FQM_QMF_CURRENT           "12"

/* CANCELLED INDICATOR */

#define FQM_CANCEL_YES             "1"
#define FQM_CANCEL_NO              "0"

/* VARIABLE TYPES */

#define FQM_VARIABLE_CHAR          "CHAR"
#define FQM_VARIABLE_FINT          "FINT"

/* The Callable Interface functions */

int FQMCIC(struct fqmcomm* fqmcom, signed long* cmdlen, char* cmdstr);
int FQMCICE(struct fqmcomm* fqmcom, signed long* cmdlen, char* cmdstr,\
        signed long* pnum, signed long* klth,\
        char* kword, signed long* vlth, void* value, char* vtype);
```

Running your C programs in TSO

To run your C program in TSO, compile and link-edit the program, and then run it with or without ISPF.

Compiling and link-editing in TSO

You must compile and link-edit your C program before you can run it in TSO.

This job compiles and link-edits your application program by using the IBM C compiler for z/OS®. Some parameters might vary from one QMF installation to the another installation.

```
//sampleC JOB
//STEP1 EXEC PROC=EDCCB,
// INFILE='name of dataset that contains source code',
// OUTFILE='name of dataset that contains executable'
//* Provide Access to QMF Communications Macro FQMCOMM
//* Copy FQMCOMM to QMZ1210.SAMPLIB
//USERLIB DD DSN=QMZ1210.SAMPLIB,DISP=SHR
//BIND.SYSIN DD DSN=QMZ1210.SFQMLOAD(FQMCINT),DISP=SHR
```

Running your programs in TSO without ISPF

After your C program compiles successfully, you can run it without ISPF.

Run your program in TSO without ISPF by writing a program similar to the following REXX script:

```
/*- REXX ---------------------------------------------------------------*/
/*      THIS EXEC INVOKES QMF Z CLIENT CALLABLE INTERFACE              */
/*---------------------------------------------------------------------*/
/*                                                                     */
/* This exec invokes the Z Client Callable Interface without ISPF     */
/* services.                                                           */
/* No ISPF allocations are done in this exec.                          */
/*                                                                     */
/* Change the QMF Z Client installation prefix (FQMPRFX) according to*/
/* your installation. Check also *.SFQMPARM and *.SFQMSKEL members.  */
/*                                                                     */
/*---------------------------------------------------------------------*/
/*---------------------------------------------------------------------*/
/* Note: C load libraries must be allocated before running             */
/*       this Exec.                                                    */
/*---------------------------------------------------------------------*/
TRACE OFF
```

```
/*--------------------------------------------------------------------*/
/* Defaults for this startup                                          */
/*--------------------------------------------------------------------*/

/* File allocation Prefix                                             */
FQMPRFX = 'FQM.PREFIX'

/*--------------------------------------------------------------------*/
/* Allocations                                                        */
/*--------------------------------------------------------------------*/

ADDRESS TSO
"ALLOC DD(FQMPARM) DA('"FQMPRFX".SFQMPARM' SHR REUSE"
"ALLOC DD(FQMLANG) DA('"FQMPRFX".SFQMLANG') SHR REUSE"

"ALLOC DD(FQMDEBUG) SYSOUT(A) RECFM(F B A) LRECL(121) REUSE"

/*--------------------------------------------------------------------*/
/* ›STEPLIB                                                           */
/*--------------------------------------------------------------------*/

ADDRESS TSO
"STEPLIB ADD DATASETS('"FQMPRFX".SFQMLOAD') FIRST QUIET"

/*--------------------------------------------------------------------*/
/* ALTLIBS for CLIST and EXEC libraries                               */
/*--------------------------------------------------------------------*/

ADDRESS TSO
"ALTLIB ACTIVATE APPLICATION(CLIST) DATASET('"FQMPRFX".SFQMSKELS')"

/*--------------------------------------------------------------------*/
/* INVOKE QMF Z CLIENT CALLABLE INTERFACE                             */
/*--------------------------------------------------------------------*/
CALL sampleC
/*--------------------------------------------------------------------*/
/* Deactivate ALTLIBs                                                 */
/*--------------------------------------------------------------------*/
ADDRESS TSO
"ALTLIB DEACTIVATE APPLICATION(*)"

/*--------------------------------------------------------------------*/
/* Free allocations                                                   */
/*--------------------------------------------------------------------*/

ADDRESS TSO
"FREE DD(FQMDEBUG)"
"FREE DD(FQMLANG)"
"FREE DD(FQMPARM)"

/*--------------------------------------------------------------------*/
/* Remove ›STEPLIB                                                    */
/*--------------------------------------------------------------------*/
"STEPLIB REMOVE DATASETS('"FQMPRFX".SFQMLOAD') FIRST QUIET"

EXIT
```

Running your programs in TSO under ISPF

After your C program compiles successfully, you can run it under ISPF.

Run your program in TSO under ISPF by writing a program similar to the following REXX script:

```
/*- REXX --------------------------------------------------------------*/
/*      THIS EXEC INVOKES QMF Z CLIENT CALLABLE INTERFACE             */
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Change the QMF Z Client installation prefix (FQMPRFX) according to*/
/* your installation. Check also *.SFQMPARM and *.SFQMSKEL members.  */
/*                                                                    */
/*--------------------------------------------------------------------*/
/*--------------------------------------------------------------------*/
/* Note: C load libraries must be allocated before running            */
/*       this Exec.                                                   */
/*--------------------------------------------------------------------*/

/*--------------------------------------------------------------------*/
/* Defaults for this startup                                          */
/*--------------------------------------------------------------------*/
```

```
/* File allocation Prefix                                             */
 FQMPRFX = 'FQM.PREFIX'
/*--------------------------------------------------------------------*/
/* ALLOCATIONS                                                        */
/*--------------------------------------------------------------------*/
ADDRESS TSO
"ALLOC DD(FQMPARM) DA('"FQMPRFX".SFQMPARM') SHR REUSE"
"ALLOC DD(FQMLANG) DA('"FQMPRFX".SFQMLANG') SHR REUSE"

"ALLOC DD(FQMDEBUG) SYSOUT(A) RECFM(F B A) LRECL(121) REUSE"

ADDRESS ISPEXEC
"LIBDEF ISPLLIB DATASET ID('"FQMPRFX".SFQMLOAD') STACK"
"LIBDEF ISPSLIB DATASET ID('"FQMPRFX".SFQMSKEL') STACK"

/*--------------------------------------------------------------------*/
/* INVOKE QMF Z CLIENT CALLABLE INTERFACE                             */
/*--------------------------------------------------------------------*/
ADDRESS TSO
"ISPEXEC SELECT CMD(sampleC) NEWAPPL(FQM) PASSLIB"

/*--------------------------------------------------------------------*/
/* FREE ALLOCATIONS                                                   */
/*--------------------------------------------------------------------*/
ADDRESS TSO
"FREE DD(FQMDEBUG)"
"FREE DD(FQMLANG)"
"FREE DD(FQMPARM)"

ADDRESS ISPEXEC
"LIBDEF ISPSLIB"
"LIBDEF ISPLLIB"
```

C++ language interface

You can use the C++ language with the callable interface in QMF.

Interface communications area mapping for C++ language (FQMCOMMP)

FQMCOMMP provides FQMCOMMP mapping for C++ language programs.

The following table provides specifications for Interface communications area mapping for C++ language (FQMCOMMP).

Table 5. Interface communications area for FQMCOMMP

Structure field	Data type	Description
FQM_RETURN_CODE	signed long integer	Indicates the status of a QMF command after it is run. Its values are: **FQM_SUCCESS** Successful execution of the request **FQM_WARNING** Normal completion with warnings **FQM_FAILURE** Command did not run correctly **FQM_SEVERE** Severe error; QMF session stopped.
FQM_INSTANCE_ID	signed long integer	Identifier established by QMF during running of the START command

Table 5. Interface communications area for FQMCOMMP (continued)		
Structure field	**Data type**	**Description**
FQM_COMM_LEVEL	character, length 12	Identifies the version of the FQMCOMM structure In your application, include instructions that initialize this variable to the value of FQM_CURRENT_COMM_LEVEL before issuing the QMF START command.
FQM_PRODUCT	character, length 2	Identifies the IBM query product in use. Initialize the value of this variable to FQM_QMF.
FQM_PRODUCT_RELEASE	character, length 2	Version of QMF in use. Variable FQM_QMF_V12R2 specifies QMF Version 12 Release 2.
FQM_RESERVE1	character, length 28	Reserved for future use
FQM_MESSAGE_ID	character, length 8	Completion message ID
FQM_Q_MESSAGE_ID	character, length 8	Not populated
FQM_START_PARM_ERROR	character, length 8	Not populated
FQM_CANCEL_IND	character, length 1	Contains one of two values, depending on whether the user canceled while a QMF command was running: • FQM_CANCEL_YES • FQM_CANCEL_NO
FQM_RESERVE2	character, length 23	Reserved for future use
FQM_RESERVE3	character, length 156	Reserved for future use
FQM_MESSAGE_TEXT	character, length 128	Completion message text
FQM_Q_MESSAGE_TEXT	character, length 128	Not populated

Function calls for the C++ language

QMF provides two function calls for the C++ language: FQMCIC and FQMCICE.

FQMCIC

This call is for QMF commands that do not require access to application program variables. Use this call for most QMF commands; its syntax is as follows:

```
FQMCIC (&FQMCOMM,&CMDLTH,&CMDSTR)
```

The parameters have the following values:

FQMCOMM
The interface communications area

CMDLTH
Length of the command string (CMDSTR); a long type parameter

CMDSTR
The QMF command to run, specified as an array of unsigned character type of the length specified by CMDLTH

The QMF command must be in uppercase.

FQMCICE

This call has an extended syntax for the QMF commands that require access to application program variables. The START and SET GLOBAL commands use the extended syntax.

```
FQMCICE (&DSQCOMM,&CMDLTH,&CMDSTR,
         &PNUM,&KLTH,&KWORD,
         &VLTH,&VALUE,&VTYPE);
```

The parameters have the following values:

FQMCOMM
The interface communications area.

CMDLTH
Length of the command string (CMDSTR); a long integer parameter.

CMDSTR
QMF command to run; an array of unsigned character type. The QMF command must be in uppercase.

PNUM
Number of command keywords; a long integer parameter.

KLTH
The length of each specified keyword (KWORD); a long integer parameter or an array of long integer parameters.

KWORD
QMF keyword, keywords, or address; each is a character, array of characters.

VLTH
The length of each value that is associated with the keyword; a long integer parameter or array of long integer parameters.

VALUE
The value that is associated with each keyword.

Its type is specified in the VTYPE parameter and can be an unsigned character array, a long integer parameter, or array of long integer parameters.

VTYPE
Data type of the contents of the VALUE parameter.

This parameter has one of two values, which are provided in the interface communications area, FQMCOMMP:

- FQM_VARIABLE_CHAR for unsigned character type
- FQM_VARIABLE_FINT for long integer

All of the values that are specified in the VALUE field must have the data type that is specified by VTYPE.

The C++ language interface has the following parameter considerations:

- Command strings and the START and SET command parameters are all input character strings. With these strings, C++ requires you to pass a storage area that ends with a null value, which must be

included in the length of the parameter. Use the compile-time length function to obtain the parameter length that is passed to the QMF interface.

- If the string does not end by a null value before reaching the end of the string, an error is returned by QMF. The null value (X'00') indicates the end of a character string.

C++ language programming example

The following sample program for the C++ language application program performs the following functions:

- Starts QMF
- Sets three global variables
- Runs a query called Q1
- Ends the QMF session

QMF does not supply query Q1, but the sample program uses these object.

```
/*
 * Sample Program
 * C++ version of the callable interface
 */

#include <iostream>
#include <cstdlib>
#include <cstring>

/*
 * Include and declare query interface communications area
 */
#include "FQMCOMMP"

#define    NUM_KEYWORDS    4
#define    SIZE_KEYVAL     32
#define    SET_KEYWORDS    3
#define    SIZE_VAL        8

int main(int argc, char** argv)
{
    struct fqmcomm communication_area;    /* FQMCOMM from include */

    /*
     * Query interface command length and commands
     */
    static char start_query_interface[] = "START";
    static char set_global_variables[] = "SET GLOBAL Lifetime=Permanent";
    static char run_query[] = "RUN QUERY Q1";
    static char end_query_interface[] = "EXIT";

    signed long command_length;
    signed long number_of_parameters;             /* number of variables */

    signed long keyword_lengths[NUM_KEYWORDS];    /* lengths of keyword names */
    signed long data_length[NUM_KEYWORDS];        /* lengths of variable data */

    /*
     * Variable data type constants
     */
    static char char_data_type[] = FQM_VARIABLE_CHAR;
    static char int_data_type[]  = FQM_VARIABLE_FINT;

    /*
     * Keyword parameter and value for START command
     */
    static char start_keywords[NUM_KEYWORDS][SIZE_KEYVAL] =\
    {"Mode"};

    static char start_keyword_values[NUM_KEYWORDS][SIZE_KEYVAL] =\
    {"I"};

    /*
     * MAIN PROGRAM
     */

    /*
```

```
     * Start a query interface session
     */
    strncpy(communication_area.fqm_comm_level,
            FQM_CURRENT_COMM_LEVEL,\
            sizeof(communication_area.fqm_comm_level));

    number_of_parameters = 1;
    command_length = sizeof(start_query_interface);

    keyword_lengths[0] = SIZE_KEYVAL;
    data_length[0] = SIZE_KEYVAL;
    FQMCICE(&communication_area, &command_length,\
      &start_query_interface[0], &number_of_parameters, keyword_lengths,\
      &start_keywords[0][0], data_length, &start_keyword_values[0][0],\
      &char_data_type[0]);

    /*
     * Keyword parameter and values for SET command
     */
    char set_keywords[SET_KEYWORDS][SIZE_VAL];
    signed long set_values[SET_KEYWORDS];

    /*
     * Set numeric values into query using SET command
     */
    number_of_parameters = 3;
    command_length = sizeof(set_global_variables);

    strcpy(set_keywords[0],"MYVAR01");
    strcpy(set_keywords[1],"SHORT");
    strcpy(set_keywords[2],"MYVAR03");

    keyword_lengths[0] = SIZE_VAL;
    keyword_lengths[1] = SIZE_VAL;
    keyword_lengths[2] = SIZE_VAL;
    data_length[0] = sizeof(long);
    data_length[1] = sizeof(long);
    data_length[2] = sizeof(long);
    set_values[0] = 20;
    set_values[1] = 40;
    set_values[2] = 84;

    FQMCICE(&communication_area, &command_length,\
      &set_global_variables[0], &number_of_parameters, keyword_lengths,\
      &set_keywords[0][0], data_length, set_values, &int_data_type[0]);

    /*
     * Run a query
     */
    command_length = sizeof(run_query);
    FQMCIC(&communication_area, &command_length, &run_query[0]);

    /*
     * End the query interface session
     */
    command_length = sizeof(end_query_interface);
    FQMCIC(&communication_area, &command_length,\
      &end_query_interface[0]);

    return 0;
}
```

FQMCOMM for C++

The interface communications area of the C++ language is defined below.

```
/* C++ include for query callable interface */

/*  Structure declaration for callable interface communication area */
struct fqmcomm
{
    long int fqm_return_code;        /* Function return code */
    long int fqm_instance_id;        /* ID established in START cmd */
    char fqm_comm_level[12];         /* Communications level id */
    char fqm_product[2];             /* Query product id */
    char fqm_product_release[2];     /* Query product release */
    char fqm_reserve1[28];           /* Reserved */
    char fqm_message_id[8];          /* Completion message ID */
    char fqm_q_message_id[8];        /* Query message ID */
```

```
    char fqm_start_parm_error[8];    /* Start parameter in error */
    char fqm_cancel_ind[1];          /* Cmd cancelled indicator, 1 = cancelled,\
                                        0 = not cancelled */
    char fqm_reserve2[23];           /* RESERVED AREAS */
    char fqm_reserve3[156];
    char fqm_message_text[128];      /* Message text */
    char fqm_q_message_text[128];    /* Query message text */
};


/* RETURN CODES */

#define FQM_SUCCESS                 0
#define FQM_WARNING                 4
#define FQM_FAILURE                 8
#define FQM_SEVERE                 16

/* Communications Level */

#define FQM_CURRENT_COMM_LEVEL    "FQML>001202<"

/* Query Product Codes */

#define FQM_QRW                    "01"
#define FQM_QFM                    "02"
#define FQM_QM3                    "03"

/* Query Product Release Levels */

#define FQM_QMF_V12R2             "12"
#define FQM_QMF_CURRENT           "12"

/* CANCELLED INDICATOR */

#define FQM_CANCEL_YES             "1"
#define FQM_CANCEL_NO              "0"

/* VARIABLE TYPES */

#define FQM_VARIABLE_CHAR          "CHAR"
#define FQM_VARIABLE_FINT          "FINT"

/* The Callable Interface functions */

extern "C"
{
int FQMCIC(struct fqmcomm* fqmcom, signed long* cmdlen, char* cmdstr);
int FQMCICE(struct fqmcomm* fqmcom, signed long* cmdlen, char* cmdstr,\
        signed long* pnum, signed long* klth,\
        char* kword, signed long* vlth, void* value, char* vtype);
}
```

Running your C++ programs in TSO

To run your C++ program in TSO, compile and link-edit the program, and then run it in either with or without ISPF.

Compiling and link-editing in TSO

You must compile and link-edit your C++ program before you can run it in TSO.

This job compiles and link-edits your application program by using the IBM C++ compiler for z/OS. Some parameters might vary from one QMF installation to the next.

```
//samCPP JOB
//STEP1 EXEC PROC=CBCCB,
// INFILE='name of dataset that contains source code',
// OUTFILE='name of dataset that contains executable'
//* Provide Access to QMF Communications Macro FQMCOMM
//* Copy FQMCOMM to QMZ1210.SAMPLIB
//USERLIB DD DSN=QMZ1210.SAMPLIB,DISP=SHR
//BIND.SYSIN DD DSN=QMZ1210.SFQMLOAD(FQMCINT),DISP=SHR
```

Running your programs in TSO without ISPF

After your C++ program compiles successfully, you can run it without ISPF.

Run your program in TSO without ISPF by writing a program similar to the following REXX script:

```
/*- REXX ---------------------------------------------------------------*/
/*      THIS EXEC INVOKES QMF Z CLIENT CALLABLE INTERFACE              */
/*---------------------------------------------------------------------*/
/*                                                                     */
/* This exec invokes the Z Client Callable Interface without ISPF      */
/* services.                                                           */
/* No ISPF allocations are done in this exec.                          */
/*                                                                     */
/* Change the QMF Z Client installation prefix (FQMPRFX) according to*/
/* your installation. Check also *.SFQMPARM and *.SFQMSKEL members.    */
/*                                                                     */
/*---------------------------------------------------------------------*/
/*---------------------------------------------------------------------*/
/* Note: C++ load libraries must be allocated before running           */
/*       this Exec.                                                    */
/*---------------------------------------------------------------------*/
TRACE OFF

/*---------------------------------------------------------------------*/
/* Defaults for this startup                                           */
/*---------------------------------------------------------------------*/

/* File allocation Prefix                                              */
FQMPRFX = 'FQM.PREFIX'

/*---------------------------------------------------------------------*/
/* Allocations                                                         */
/*---------------------------------------------------------------------*/

ADDRESS TSO
"ALLOC DD(FQMPARM) DA('"FQMPRFX".SFQMPARM' SHR REUSE"
"ALLOC DD(FQMLANG) DA('"FQMPRFX".SFQMLANG') SHR REUSE"

"ALLOC DD(FQMDEBUG) SYSOUT(A) RECFM(F B A) LRECL(121) REUSE"

/*---------------------------------------------------------------------*/
/* ›STEPLIB                                                            */
/*---------------------------------------------------------------------*/

ADDRESS TSO
"STEPLIB ADD DATASETS('"FQMPRFX".SFQMLOAD') FIRST QUIET"

/*---------------------------------------------------------------------*/
/* ALTLIBS for CLIST and EXEC libraries                                */
/*---------------------------------------------------------------------*/

ADDRESS TSO
"ALTLIB ACTIVATE APPLICATION(CLIST) DATASET('"FQMPRFX".SFQMSKELS')"

/*---------------------------------------------------------------------*/
/* INVOKE QMF Z CLIENT CALLABLE INTERFACE                              */
/*---------------------------------------------------------------------*/
CALL samCPP
/*---------------------------------------------------------------------*/
/* Deactivate ALTLIBs                                                  */
/*---------------------------------------------------------------------*/
ADDRESS TSO
"ALTLIB DEACTIVATE APPLICATION(*)"

/*---------------------------------------------------------------------*/
/* Free allocations                                                    */
/*---------------------------------------------------------------------*/

ADDRESS TSO
"FREE DD(FQMDEBUG)"
"FREE DD(FQMLANG)"
"FREE DD(FQMPARM)"

/*---------------------------------------------------------------------*/
/* Remove ›STEPLIB                                                     */
/*---------------------------------------------------------------------*/
"STEPLIB REMOVE DATASETS('"FQMPRFX".SFQMLOAD') FIRST QUIET"

EXIT
```

Running your programs in TSO under ISPF

After your C++ program compiles successfully, you can run it under ISPF.

Run your program in TSO under ISPF by writing a program similar to the following REXX script:

```
/*- REXX ---------------------------------------------------------------*/
/*      THIS EXEC INVOKES QMF Z CLIENT CALLABLE INTERFACE              */
/*---------------------------------------------------------------------*/
/*                                                                     */
/* Change the QMF Z Client installation prefix (FQMPRFX) according to*/
/* your installation. Check also *.SFQMPARM and *.SFQMSKEL members.  */
/*                                                                     */
/*---------------------------------------------------------------------*/
/*---------------------------------------------------------------------*/
/* Note: C++ load libraries must be allocated before running          */
/*       this Exec.                                                    */
/*---------------------------------------------------------------------*/

/*---------------------------------------------------------------------*/
/* Defaults for this startup                                           */
/*---------------------------------------------------------------------*/
/* File allocation Prefix                                              */
 FQMPRFX = 'FQM.PREFIX'
/*---------------------------------------------------------------------*/
/* ALLOCATIONS                                                         */
/*---------------------------------------------------------------------*/
ADDRESS TSO
"ALLOC DD(FQMPARM) DA('"FQMPRFX".SFQMPARM') SHR REUSE"
"ALLOC DD(FQMLANG) DA('"FQMPRFX".SFQMLANG') SHR REUSE"

"ALLOC DD(FQMDEBUG) SYSOUT(A) RECFM(F B A) LRECL(121) REUSE"

ADDRESS ISPEXEC
"LIBDEF ISPLLIB DATASET ID('"FQMPRFX".SFQMLOAD') STACK"
"LIBDEF ISPSLIB DATASET ID('"FQMPRFX".SFQMSKEL') STACK"

/*---------------------------------------------------------------------*/
/* INVOKE QMF Z CLIENT CALLABLE INTERFACE                              */
/*---------------------------------------------------------------------*/
ADDRESS TSO
"ISPEXEC SELECT CMD(samCPP) NEWAPPL(FQM) PASSLIB"

/*---------------------------------------------------------------------*/
/* FREE ALLOCATIONS                                                    */
/*---------------------------------------------------------------------*/
ADDRESS TSO
"FREE DD(FQMDEBUG)"
"FREE DD(FQMLANG)"
"FREE DD(FQMPARM)"

ADDRESS ISPEXEC
"LIBDEF ISPSLIB"
"LIBDEF ISPLLIB"
```

COBOL language interface

You can use the COBOL language with the callable interface in QMF.

Interface communications area mapping for COBOL (FQMCOMMB)

FQMCOMMB provides FQMCOMM mapping for COBOL language programs.

The following table provides specifications for Interface communications area mapping for COBOL language (FQMCOMMB).

Table 6. Interface communications area for COBOL (FQMCOMMB)		
Structure field	**Data type**	**Description**
FQM-RETURN-CODE	PIC 9(8)	Indicates the status of a QMF command after is run. Its values are: **FQM-SUCCESS** Successful run of the request **FQM-WARNING** Normal completion with warnings **FQM-FAILURE** Command did not run correctly **FQM-SEVERE** Severe error; QMF session stopped
FQM-INSTANCE-ID	PIC 9(8)	Identifier established by QMF during running of the START command
FQM-COMM-LEVEL	PIC X(12)	Identifies the version of the FQMCOMM structure In your application, include instructions that initialize this variable to the value of FQM_CURRENT_COMM_LEVEL before issuing the QMF START command.
FQM-PRODUCT	PIC X(2)	Identifies the IBM query product in use Initialize the value of this variable to FQM_QMF.
FQM-PRODUCT-RELEASE	PIC X(2)	Version of QMF in use. Variable FQM_QMF_V11R2 specifies QMF Version 12 Release 2.
FQM-RESERVE1	PIC X(28)	Reserved for future use
FQM-MESSAGE-ID	PIC X(8)	Completion message ID
FQM-Q-MESSAGE-ID	PIC X(8)	Not populated
FQM-START-PARM-ERROR	PIC X(8)	Not populated
FQM-CANCEL-IND	PIC X(1)	Contains one of two values, depending on whether the user canceled while a QMF command was running: • FQM-CANCEL-YES • FQM-CANCEL-NO
FQM-RESERVE2	PIC X(23)	Reserved for future use
FQM-RESERVE3	PIC X(156)	Reserved for future use
FQM-MESSAGE-TEXT	PIC X(128)	Completion message text
FQM-Q-MESSAGE-TEXT	PIC X(128)	Not populated

Function calls for COBOL

QMF provides two function calls for the COBOL language: FQMCIC and FQMCICE (extended format).

FQMCIC

This call is for QMF commands that do not require access to application program variables. Use this call for most QMF commands.

```
    CALL "FQMCIC" USING FQMCOMM CMDLTH CMDSTR
```

The parameters have the following values:

FQMCOMM
The interface communications area

CMDLTH
Length of the command string (CMDSTR); an integer parameter

CMDSTR
QMF command to run; an uppercase character string of the length specified by CMDLTH

FQMCICE (extended format)

This call has an extended syntax for the QMF commands that require access to application program variables. The START and SET GLOBAL commands use as the extended syntax.

```
     "FQMCICE" USING
             FQMCOMM CMDLTH CMDSTR
             PNUM KLTH KWORD VLTH VALUE VTYPE
```

The parameters have the following values:

FQMCOMM
The interface communications area.

CMDLTH
The length of the command string (CMDSTR); an integer parameter.

CMDSTR
The QMF command to run; an uppercase character string of the length specified by CMDLTH.

PNUM
The number of command keywords; an integer parameter.

KLTH
The length of each specified keyword; an integer parameter or an array of integer parameters.

KWORD
QMF keyword, keywords.

Each is a character, array of characters. If all the keywords have the same length, you can use an array of characters.

VLTH
The length of each value that is associated with the keyword; an integer parameter or an array of integer parameters.

VALUE
The value that is associated with each keyword.

Its type is specified in the VTYPE parameter, and can be a character, an array of characters, an integer parameter, or an array of integer parameters.

VTYPE
Data type of the contents of the VALUE parameter.

This parameter has one of two values, which are provided in the communications area, FQMCOMMB:

- FQM-VARIABLE-CHAR for character values
- FQM-VARIABLE-FINT for integer values

All values that are specified in the VALUE field must have the data type that is specified by VTYPE.

COBOL programming example

The following sample program for the COBOL application program performs the following functions:

- Starts QMF
- Sets three global variables
- Runs a query called Q1
- Ends the QMF session

QMF does not supply query Q1, but the sample program uses these objects.

```
*****************************************************************
*   The following is a COBOL version of the query
*   callable interface *** FQMCOINF **.
*****************************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  FQMCOINF.
   DATE-COMPILED.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*************************
* Copy FQMCOMMB definition - contains query interface variables
*************************
   COPY FQMCOMMB.

* Query interface commands
 01  STARTQI      PIC X(5)  VALUE "START".
 01  SETG         PIC X(29) VALUE "SET GLOBAL LifeTime=Permanent".
 01  QUERY        PIC X(12) VALUE "RUN QUERY Q1".
 01  ENDQI        PIC X(4)  VALUE "EXIT".

* Query command length
 01  QICLTH       PIC S9(9) USAGE IS BINARY.
* Number of variables
 01  QIPNUM       PIC S9(8) USAGE IS BINARY.
* Keyword variable lengths
 01  QIKLTHS.
   03   KLTHS     PIC S9(9) OCCURS 10 USAGE IS BINARY.
* Value Lengths
 01  QIVLTHS.
   03   VLTHS     PIC 9(9) OCCURS 10 USAGE IS BINARY.
* Start Command Keyword
 01 SNAMES.
   03 FQMMODE    PIC X(4) VALUE "MODE".
* Start Command Keyword Value
 01 SVALUES.
   03 FQMMODE   PIC X(1)  VALUE "I".
* Set GLOBAL Command Variable Names to set
 01  VNAMES.
   03  VNAME1   PIC X(7) VALUE "MYVAR01".
   03  VNAME2   PIC X(5) VALUE "SHORT".
   03  VNAME3   PIC X(7) VALUE "MYVAR03".
* Variable value parameters
 01  VVALUES.
   03   VVALS     PIC S9(9) OCCURS 10 USAGE IS BINARY.

 01   FQMSPACE PIC X              VALUE SPACE.

 PROCEDURE DIVISION.

*
* Start a query interface session
     MOVE FQM-CURRENT-COMM-LEVEL TO FQM-COMM-LEVEL.
     MOVE 5 TO QICLTH.
     MOVE 4 TO KLTHS(1).
     MOVE 1  TO VLTHS(1).
     MOVE 1 TO QIPNUM.
* Start a query interface session
     CALL "FQMCICE" USING FQMCOMM, QICLTH, STARTQI,
                          QIPNUM, QIKLTHS, SNAMES,
```

```
                         QIVLTHS, SVALUES, FQM-VARIABLE-CHAR.

*
* Set numeric values into query variables using SET GLOBAL command
     MOVE 29 TO QICLTH.
     MOVE 7 TO KLTHS(1).
     MOVE 5 TO KLTHS(2).
     MOVE 7 TO KLTHS(3).
     MOVE 4 TO VLTHS(1).
     MOVE 4 TO VLTHS(2).
     MOVE 4 TO VLTHS(3).
     MOVE 20 TO VVALS(1).
     MOVE 40 TO VVALS(2).
     MOVE 84 TO VVALS(3).
     MOVE 3 TO QIPNUM.
     CALL "FQMCICE" USING FQMCOMM, QICLTH, SETG,
                          QIPNUM, QIKLTHS, VNAMES,
                          QIVLTHS, VVALUES, FQM-VARIABLE-FINT.

*
* Run a Query
     MOVE 12 TO QICLTH.
     CALL "FQMCIC" USING FQMCOMM, QICLTH, QUERY.

*
* End the query interface session
     MOVE 4 TO QICLTH.
     CALL "FQMCICE" USING FQMCOMM, QICLTH, ENDQI,
                          FQMSPACE, FQMSPACE, FQMSPACE,
                          FQMSPACE, FQMSPACE, FQMSPACE.

     STOP RUN.
```

FQMCOMM for COBOL

The interface communications area for the COBOL language is defined below.

The interface communications area of the COBOL language is defined below:

```
************************************************************
* COBOL INCLUDE FOR QUERY CALLABLE INTERFACE
************************************************************

* STRUCTURE DECLARE FOR COMMUNICATIONS AREA

  01 FQMCOMM.

     03 FQM-RETURN-CODE PIC S9(9) USAGE IS BINARY.
*                                FUNCTION RETURN CODE      *
     03 FQM-INSTANCE-ID PIC S9(9) USAGE IS BINARY.
*                                IDENTIFIER FROM START CMD *
     03 FQM-COMM-LEVEL  PIC X(12).
*                                COMMUNICATIONS LEVEL      *
     03 FQM-PRODUCT     PIC X(2).
*                                QUERY PRODUCT ID          *
     03 FQM-PRODUCT-RELEASE PIC X(2).
*                                QUERY PRODUCT RELEASE     *
     03 FQM-RESERVE1 PIC X(28).
*                                RESERVED AREA             *
     03 FQM-MESSAGE-ID PIC X(8).
*                                COMPLETION MESSAGE ID     *
     03 FQM-Q-MESSAGE-ID PIC X(8).
*                                QUERY MESSAGE ID          *
     03 FQM-START-PARM-ERROR PIC X(8).
*                                START PARAMETER IN ERROR  *
     03 FQM-CANCEL-IND PIC X(1).
*                                1 = COMMAND CANCELLED     *
*                                0 = COMMAND NOT CANCELLED *
     03 FQM-RESERVE2 PIC X(23).
*                                RESERVED AREA             *
     03 FQM-RESERVE3 PIC X(156).
*                                RESERVED AREA             *
     03 FQM-MESSAGE-TEXT PIC X(128).
*                                QMF MESSAGE TEXT          *
     03 FQM-Q-MESSAGE-TEXT PIC X(128).
*                                QMF QUERY MESSAGE TEXT    *
*                                512 BYTES TOTAL           *
```

```
*  VALUES FOR FQM-RETURN-CODE

  01 FQM-SUCCESS PIC 9(8) USAGE IS COMP VALUE 0.
  01 FQM-WARNING PIC 9(8) USAGE IS COMP VALUE 4.
  01 FQM-FAILURE PIC 9(8) USAGE IS COMP VALUE 8.
  01 FQM-SEVERE  PIC 9(8) USAGE IS COMP VALUE 16.

*  VALUES FOR FQM-COMM-LEVEL

  01 FQM-CURRENT-COMM-LEVEL PIC X(12) VALUE "FQML>001202<".

*  VALUES FOR FQM-PRODUCT

  01 FQM-QRW             PIC X(2) VALUE "01".
  01 FQM-QMF             PIC X(2) VALUE "02".
  01 FQM-QM4             PIC X(2) VALUE "03".

*  VALUES FOR FQM-PRODUCT-RELEASE

  01 FQM-QMF-V12R2       PIC X(2) VALUE "12".
  01 FQM-QMF-CURRENT     PIC X(2) VALUE "12".

*  VALUES FOR FQM-CANCEL-IND

  01 FQM-CANCEL-YES      PIC X(1) VALUE "1".
  01 FQM-CANCEL-NO       PIC X(1) VALUE "0".

*  VALUES FOR VARIABLE TYPE ON CALL PARAMETER

  01 FQM-VARIABLE-CHAR PIC X(4) VALUE "CHAR".
  01 FQM-VARIABLE-FINT PIC X(4) VALUE "FINT".
```

Considerations for running your COBOL callable interface program

Pay attention to the details about running a COBOL program that uses the QMF Z Client callable interface.

When you translate, compile, and link-edit a program that uses the QMF Z Client callable interface, consider the following conditions:

- The execution environment

  Your COBOL program must call the QMF Z Client interface program, FQMCICE and FQMCIC, by using a COBOL static call.
- Whether to use quotation marks or apostrophes

  You must use either double quotation marks (") or apostrophes (') to delimit literals in a COBOL program.

  The communications area (FQMCOMMB) and the sample COBOL program as distributed by QMF Z Client use quotations to delimit literals. If your site or program uses apostrophes instead of quotation marks, change FQMCOMMB or copy the structure to your program, changing quotation marks to apostrophes.
- Availability of the communications area (FQMCOMMB)

  The communications area FQMCOMMB must be available to the COBOL compile step or copied into your program as a control structure.
- Availability of the interface module (FQMCINT)

  The QMF interface module must be available during the link-edit step of your program.

Running your COBOL programs in TSO

To run your COBOL program in TSO, compile and link-edit the program, and then run it in either with or without ISPF.

Compiling and link-editing in TSO

You must compile and link-edit your COBOL program before you can run it in TSO.

This job uses the COBOL compiler to compile your application program. It then link-edits your application. Some parameters might vary from one QMF installation to the next.

```
//samCOBOL   JOB
//STEP1      EXEC PROC=IGYWCL
//* Provide access to Z Client communications macro FQMCOMM
//* Copy FQMCOMM to QMZ1210.SAMPLIB
//COBOL.SYSLIB   DD DSN=QMZ1210.SAMPLIB,DISP=SHR
//COBOL.SYSIN    DD  *
          .
          .
          .
          Your program or copy of Z Client sample
          .
          .
          .
//* Provide access to Z Client interface module
//* Allocation for target library
//LKED.SYSLMOD DD
//* Allocation for Z Client load library
//LKED.QMZLOAD DD DSN=QMZ1210.SFQMLOAD,DISP=SHR
//LKED.SYSIN    DD  *
          INCLUDE QMZLOAD(FQMCINT)
          ENTRY samCOBOL
          MODE  AMODE(31) RMODE(31)
          NAME  samCOBOL(R)
 /*
```

Running your programs in TSO without ISPF

After your COBOL program compiles successfully, you can run it with JCL without ISPF.

Run the COBOL compiler and linkage editor in TSO without ISPF by writing a program similar to the following CLIST:

```
/*- REXX ---------------------------------------------------------------*/
/*              THIS EXEC STARTS QMF Z CLIENT PROGRAM                  */
/*---------------------------------------------------------------------*/
/*                                                                     */
/* This exec invokes the Z Client Callable Interface without ISPF     */
/* services.                                                           */
/* No ISPF allocations are done in this exec.                          */
/*                                                                     */
/* Change the QMF Z Client installation prefix (FQMPRFX) according to*/
/* your installation. Check also *.SFQMPARM and *.SFQMSKEL members.  */
/*                                                                     */
/*---------------------------------------------------------------------*/
/*---------------------------------------------------------------------*/
/* Note: COBOL load libraries must be allocated before running        */
/*       this REXX Exec.                                               */
/*---------------------------------------------------------------------*/
TRACE OFF

/*---------------------------------------------------------------------*/
/* Defaults for this startup                                           */
/*---------------------------------------------------------------------*/

/* File allocation Prefix                                              */
FQMPRFX = 'FQM.PREFIX'

/*---------------------------------------------------------------------*/
/* Allocations                                                         */
/*---------------------------------------------------------------------*/

ADDRESS TSO
"ALLOC DD(FQMPARM) DA('"FQMPRFX".SFQMPARM' SHR REUSE"
```

```
"ALLOC DD(FQMLANG) DA('"FQMPRFX".SFQMLANG') SHR REUSE"

"ALLOC DD(FQMDEBUG) SYSOUT(A) RECFM(F B A) LRECL(121) REUSE"

/*--------------------------------------------------------------------*/
/* ›STEPLIB                                                           */
/*--------------------------------------------------------------------*/

ADDRESS TSO
"STEPLIB ADD DATASETS('"FQMPRFX".SFQMLOAD') FIRST QUIET"

/*--------------------------------------------------------------------*/
/* ALTLIBS for CLIST and EXEC libraries                               */
/*--------------------------------------------------------------------*/

ADDRESS TSO
"ALTLIB ACTIVATE APPLICATION(CLIST) DATASET('"FQMPRFX".SFQMSKELS')"

/*--------------------------------------------------------------------*/
/* Start ZClient                                                      */
/*--------------------------------------------------------------------*/
CALL samCOBOL
/*--------------------------------------------------------------------*/
/* Deactivate ALTLIBs                                                 */
/*--------------------------------------------------------------------*/
ADDRESS TSO
"ALTLIB DEACTIVATE APPLICATION(*)"

/*--------------------------------------------------------------------*/
/* Free allocations                                                   */
/*--------------------------------------------------------------------*/

ADDRESS TSO
"FREE DD(FQMDEBUG)"
"FREE DD(FQMLANG)"
"FREE DD(FQMPARM)"

/*--------------------------------------------------------------------*/
/* Remove ›STEPLIB                                                    */
/*--------------------------------------------------------------------*/
"STEPLIB REMOVE DATASETS('"FQMPRFX".SFQMLOAD') FIRST QUIET"

EXIT
```

Running your programs in TSO under ISPF

After your COBOL program compiles successfully, you can run it under ISPF.

Run your program in TSO under ISPF by writing a program similar to the following REXX script:

```
/*- REXX --------------------------------------------------------------*/
/*      THIS EXEC INVOKES QMF Z CLIENT CALLABLE INTERFACE             */
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Change the QMF Z Client installation prefix (FQMPRFX) according to*/
/* your installation. Check also *.SFQMPARM and *.SFQMSKEL members.  */
/*                                                                    */
/*--------------------------------------------------------------------*/
/*--------------------------------------------------------------------*/
/* Note: COBOL load libraries must be allocated before running        */
/*       this Exec.                                                   */
/*--------------------------------------------------------------------*/

/*--------------------------------------------------------------------*/
/* Defaults for this startup                                          */
/*--------------------------------------------------------------------*/
/* File allocation Prefix                                             */
 FQMPRFX = 'FQM.PREFIX'
/*--------------------------------------------------------------------*/
/* ALLOCATIONS                                                        */
/*--------------------------------------------------------------------*/
ADDRESS TSO
"ALLOC DD(FQMPARM) DA('"FQMPRFX".SFQMPARM') SHR REUSE"
"ALLOC DD(FQMLANG) DA('"FQMPRFX".SFQMLANG') SHR REUSE"

"ALLOC DD(FQMDEBUG) SYSOUT(A) RECFM(F B A) LRECL(121) REUSE"

ADDRESS ISPEXEC
"LIBDEF ISPLLIB DATASET ID('"FQMPRFX".SFQMLOAD') STACK"
```

```
"LIBDEF ISPSLIB DATASET ID('"FQMPRFX".SFQMSKEL') STACK"

/*--------------------------------------------------------------------*/
/* INVOKE QMF Z CLIENT CALLABLE INTERFACE                             */
/*--------------------------------------------------------------------*/
ADDRESS TSO
"ISPEXEC SELECT CMD(samCOBOL) NEWAPPL(FQM) PASSLIB"

/*--------------------------------------------------------------------*/
/* FREE ALLOCATIONS                                                   */
/*--------------------------------------------------------------------*/
ADDRESS TSO
"FREE DD(FQMDEBUG)"
"FREE DD(FQMLANG)"
"FREE DD(FQMPARM)"

ADDRESS ISPEXEC
"LIBDEF ISPSLIB"
"LIBDEF ISPLLIB"
```

Appendix A. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use a software product successfully.

Accessibility in QMF Z Client

QMF Z Client includes several accessibility features.

Accessibility features in QMF Z Client enable users to:

- Use assistive technologies such as screen readers and screen magnifier software. Consult the assistive technology documentation for specific information when using it to access z/OS® interfaces.
- Customize display attributes such as color and font size.
- Operate specific or equivalent features by using only the keyboard. Refer to the following publications for information about accessing ISPF interfaces:
  - *z/OS ISPF User's Guide, Volume 1*
  - *z/OS TSO/E Primer*
  - *z/OS TSO/E User's Guide*

Navigation in QMF Z Client

The means that you can use to navigate among QMF panels will differ depending on whether your terminal emulator supports vector graphics.

If you are using a mouse and a keyboard, you can navigate among QMF panels by clicking the Action bar items. To perform some of the actions, such as accessing the list of favorites or recently used objects from the **Home** panel, you must position the cursor on the item that you want to access and press Enter.

If you are using only a keyboard, use the command lines and specific QMF commands to navigate among panels. To access the Action bar, enter the `ACTIONS X` command, where X is the underscored letter in the name of the Action bar item.

Some of the QMF panels feature clickable + and > characters. The + character marks the fields that support the `LIST` command. Clicking the + character is the equivalent of pressing the **List** function key. The > character marks the fields that can be opened on a separate panel. Clicking the > character is the equivalent of pressing the **Show Field** function key.

Appendix B. Troubleshooting

Diagnose and correct problems that you might experience with QMF.

QMF trace feature

QMF provides a means of tracing QMF activity during a user-session. Trace output can help you analyze errors such as incorrect or missing output, performance problems, and loops. This section shows you how to allocate the storage data set for the trace output, how to start the facility, and how to view the trace data for diagnosis.

Allocating the trace data set

Trace information is recorded in the FQMDEBUG data set, which is used only for trace purposes. Before you start a QMF session, this data set must be allocated, either automatically or manually.

To determine if the data set is automatically allocated, consult your TSO administrator. To manually allocate the data set, issue the following TSO statement before you start QMF for the diagnostic session:

```
ALLOC DDNAME(FQMDEBUG) SYSOUT(A) RECFM(F B A) LRECL(121)
```

Tracing QMF activity

1. Allocate a data set whose ddname is FQMDEBUG.
2. Use the DSQSDBUG parameter to start QMF Z Client. The value of this parameter determines the level of detail in the trace output. Valid values are:

   ALL
   QMF activity is traced at the highest level of detail, including the program failures that might occur during QMF initialization. If the trace output exceeds 32,767 rows, you must use a transient data queue to hold it.

   NONE
   No QMF activity is traced.

   X
   Enables QMF Z Client internal debug trace. If you use this value, specify the level of detail in the trace output. Specify X1 for the medium level of detail. Specify X2 for the highest level of detail.

   L
   Traces QMF Z Client messages and commands. If you use this value, specify the level of detail in the trace output. Specify L1 to log all messages, specify L2 to log all of the L1 records and additional records that describe the execution of QMF commands. Use the L2 value to log each command that the user issues and QMF response to the command.

Use the values X1, X2, L1, and L2 in any combination to provide various levels of detail in the trace output.

Printing or displaying trace output

To allocate the FQMDEBUG data set for printing, issue the following statements:

```
FREE FILE(FQMDEBUG)
ATTR DEBUG RECFM(F B A) LRECL(121)
ALLOC DDNAME(FQMDEBUG) SYSOUT(A) USING(DEBUG)
```

The allocated data set contains 121-character records. The first character of each record is an ANSI carriage-control character. The trace information is formatted with 120 characters per line, not including the ANSI carriage-control character.

If you allocated the output from the FQMDEBUG data set to go to the HOLD queue, issue the following TSO command to release the output to the OUTPUT queue:

```
FREE DDNAME(FQMDEBUG)
```

To allocate the FQMDEBUG data set as a sequential data set that you can display by using an online editor, issue the following statements:

```
FREE FILE(FQMDEBUG)
ATTR DEBUG RECFM( F B A) LRECL(81)
ALLOC DDNAME(FQMDEBUG) DSNAME(DEBUG.LIST) NEW KEEP
```

The allocated data set consists of 81-character records. The first character of each record is an ANSI carriage-control character. The trace information is formatted with 80 characters per line, not including the ANSI carriage-control character.

Interrupting QMF commands

Use the Attention function to interrupt the execution of a QMF command.

In TSO, the QMF interrupt handler can be activated even though a QMF command is inactive. To interrupt QMF, press the Attn key.

Appendix C. QMF Commands

ACTIONS command

Use the ACTIONS command to access the Action bar items from the command line.

Syntax

```
>>--ACtions-+-------+-<<
            +-value-+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

To expand a specific Action bar item, use the ACTIONS command with the underscored letter of the Action bar item name as the command value. If you use the ACTIONS command without any value, it positions the cursor on the Action bar, without expanding any specific item.

Examples

ACTIONS F
Expands the **File** item of the Action bar.

AC F
Also expands the **File** item of the Action bar.

ACTIONS V
Expands the **View** item of the Action bar.

ADD command

Use the ADD command to create objects on certain QMF panels.

Syntax

```
>>--ADd--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

Use the ADD command on the **Globals** panel to define new global variables or in the Table Editor to add rows to database tables.

Examples

```
ADD
```

```
AD
```

BACKWARD command

Use the BACKWARD command to scroll the scrollable area towards the top.

Syntax

```
>>--BACkward-----+---------+--<<
                 +--value--+
                 +--Max----+
                 +--Half---+
                 +--Page---+
                 +--CSR----+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the BACKWARD command:

A number in the range 1 - 9999
Scrolls the number of pages or rows.

MAX
Scrolls to the top.

HALF
Scrolls by half a page.

PAGE
Scrolls by one page.

CSR
Scrolls based on the position of the cursor. If the cursor is in a scrollable area, scrolls to the top. If the cursor is outside of or at the end the scrollable area, scrolls one page.

If you issue the BACKWARD command without a parameter, the default parameter is used. You can view or change the default parameter in the **Scroll** field, which is located in the lower right corner of the screen.

Examples

```
BACKWARD MAX
```

```
BACKWARD 4
```

```
BAC
```

BATCH command

Use the BATCH command to open the **Batch List** panel, which you use to create and edit QMF batch objects and to run and export JCL jobs.

Syntax

```
>>---BATch---<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Examples

```
BATCH
```

```
BAT
```

BOTTOM command

Use the BOTTOM command to scroll to the last line of the scrollable area. BOTTOM is equivalent to FORWARD MAX.

Syntax

```
>>--BOttom--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Examples

```
BOTTOM
```

```
BO
```

CHANGE command

Use the CHANGE command to change any table, column, joining option, or condition in a prompted query.

Syntax

```
>>--CHAnge--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

To change a table, column, joining option, or condition in a prompted query, type CHANGE on the command line, position the cursor on the element to change, and press Enter.

Examples

```
CHANGE
```

```
CHA
```

CHECK command

Use the CHECK command to check a FORM panel for errors. Note that you have to be on one of the FORM panels of your report in order to use the CHECK command.

Syntax

```
>>--CHEck--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Examples

```
CHECK
```

```
CHE
```

CLEAR command

Use the CLEAR command to clear all **Action** fields on the **Object List** panel.

Syntax

```
>>--CLEar--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Examples

```
CLEAR
```

```
CLE
```

CLOSE command

The CLOSE command closes the currently open document.

Syntax

```
>>--CLOse--+-----+--<<
           +-All-+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

If you specify ALL as the parameter for the CLOSE command, the command closes all currently open documents.

Examples

```
CLOSE
```

```
CLOSE ALL
```

```
CLO A
```

CONNECT command

Use the CONNECT command to connect to a remote database server.

Syntax

To connect to a database server, issue the following command:

```
>>--CONNect--TO--servername--<<
```

To connect to a database server and set the user, issue the following command:

```
>>--CONNect---authorizationid--TO--servername--(Password=password--<<
```

Uppercase letters in each diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the CONNECT command:

servername
Specifies the name of the server to which you want to connect.

authorizationid
Specifies the user ID for the database user. The user must be granted the CONNECT permission with a password.

PASSWORD
Specifies the password for the database user.

Examples

```
CONNECT TO example_server
```

```
CONN example_auth_id TO example_server(PASSWORD=abc
```

CONVERT command

The CONVERT command converts a prompted query or an SQL query into a query with standard SQL syntax. The original query remains unaffected by this operation.

Syntax

To convert the currently open query, use the following command:

```
>>--CONVert--QUERY--<<
```

To convert a query that is stored in a database, use the following command:

```
>>--CONVert--queryname--+-------------------+-<<
                        +-Substitute=Yes/No-+
```

Uppercase letters in each diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the CONVERT command:

queryname
The name of the query that you want to convert.

SUBSTITUTE
Specifies whether the variables in the query will be assigned values or not. Valid values for this parameter are:

YES
If the query uses one or more variables, QMF attempts to assign a value to each variable. If all the variables are defined either via the `&variable` parameter, or via a predefined global variable, no prompt panel will be displayed. If QMF cannot assign a value to the variable, it will prompt the user to enter the value.

NO
No values are assigned to the variables.

Examples

```
CONVERT QUERY
```

```
CONV query01
```

CREATE command

Use the CREATE command to create QMF objects.

Syntax

```
>>--CReate-+-Query--+--<<
           +-Proc---+
           +-FORm---+
           +-FOLder-+-foldername-+(--------------------+
                                  +-Folder=parentfolder-+
                                  +-Comment=text--------+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the CREATE command:

QUERY
Creates a query and opens the query editor.

PROC
Creates a procedure and opens the procedure editor.

FORM
Creates a default form for data and displays it on the screen.

FOLDER
Creates a folder in the specified location.

Note: To create a workspace folder, you must specify the full path to the workspace as the value for the `Folder` parameter.

The following parameters can be specified for the `CREATE FOLDER` command:

foldername
Specifies the name of the folder that you are creating.

FOLDER
Specifies the name of the parent workspace folder.

COMMENT
Specifies a comment for the folder. Make sure to enclose the comment text in quotation marks or parentheses.

Examples

```
CREATE QUERY
```

```
CREATE Q
```

```
CREATE PROC
```

```
CREATE FORM
```

```
CREATE FOLDER NEW_FOLDER (FOLDER=parent_folder
```

```
CREATE FOLDER NEW_FOLDER (RSBI:/.WORKSPACES/WORKSPACENAME
```

DELETE command

Use the DELETE command to remove specific items from some QMF panels.

Syntax

```
>>--DELete--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Commentary

Use the DELETE command to remove any of the following items:

- A column on the **Form.Main** panel or the **Form.Columns** panel.
- A calculation expression on the **Form.Calculations** panel.
- A conditional expression on the **Form.Conditions** panel.
- A line of text on the **Form.Break** panel, or on the **Form.Detail**, **Form.Final**, or **Form.Page** panels.
- A row from a database table when using the Table Editor.
- A list item on any of the Prompted Query Editor panels.
- A user-defined global variable on the **Globals** panel.

To remove an item, complete the following steps:

1. Type DELETE in the command line.
2. Position the cursor on the item that you want to remove.
3. Press Enter.

Examples

```
DELETE
```

```
DEL
```

DESCRIBE command

Use the DESCRIBE command to view detailed information about QMF objects.

Syntax

```
>>--DEScribe--<<
```

Commentary

Use the DESCRIBE command on the **Object List** panel to view detailed information about the following objects:

- Forms
- Procedures
- Queries

- Tables
- Views
- Folders

To view the detailed information about an object, complete the following steps:

1. Type DESCRIBE in the command line.
2. Position the cursor on the item whose detailed information you want to view.
3. Press Enter.

Examples

```
DESCRIBE
```

```
DES
```

DISPLAY command

The DISPLAY command displays an object from the temporary storage or from a database. The DISPLAY command can also be used to navigate among panels.

Syntax

To display a query, a procedure, or a database table, use the following command:

```
>>--DIsplay-+-------+--objectname-+--<<
            +-QUERY-+-------------+
                    +--objectname-+
            +-PROC--+-------------+
                    +--objectname-+
            +-TABLE-+-------------+
                    +--objectname-+
```

To display a form that is stored in the temporary storage, use the following command:

```
>>--DIsplay--FORM-+-------------+--<<
                  +-.MAIN-------+
                  +-.BREAK1-----+
                  +-.BREAK2-----+
                  +-.BREAK3-----+
                  +-.BREAK4-----+
                  +-.BREAK5-----+
                  +-.BREAK6-----+
                  +-.COLUMNS----+
                  +-.CONDITIONS-+
                  +-.DETAIL-----+
                  +-.OPTIONS----+
                  +-.PAGE-------+
                  +-.FINAL------+
                  +-.CALC-------+
```

To display a form that is stored in a database, use the following command:

```
>>--DIsplay-+------+-objectname-+--<<
            +-FORM-+
```

To view or edit the set of function keys for a panel, use the following command:

```
>>--DIsplay--KEYS-+-----------------+-<<
                  (+-panelid=panelid-+
```

Uppercase letters in each diagram show the minimum set of letters required to issue the command.

Parameters

To display an object, such as query, procedure, form, or table, you must specify its name as the parameter for the DISPLAY command.

Note: For QMF Catalog objects that belong to the currently logged in owner, you can specify the object name alone. For QMF Catalog objects that belong to other owners, use the following template: `display ownername.objectname`. For repository objects, type the full object key of the object that you want to display.

If you specify the QMF object type without the object name, the current object opens.

If several objects of different types have the same name in the database, you must specify the type of object along with its name.

For the DISPLAY KEYS command, you can specify the `panelid` parameter, which is the ID of the panel whose set of function keys you want to view or edit. If you issue the command without the `panelid` parameter, QMF will display the list of function keys for the currently open panel.

To view the complete list of QMF panels and their IDs, see Appendix H, "IDs of QMF panels," on page 137.

Examples

```
DISPLAY QUERY EXAMPLE_QUERY_1
```

```
DI PROC
```

```
DISPLAY FORM.MAIN
```

```
DI rsbi:/.workspaces/workspace1/object1
```

DRAW command

The DRAW command creates a basic SQL query for the specified table based on the description of that table in the database.

Syntax

```
>>--DRaw-tablename-+--------------------------+--<<
                   (+-Type=Select/Insert/Update-+
                    +-Identifier=corrname-------+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the DRAW command:

TYPE
Specifies the type of the query that you are creating. The default value is `Select`.

IDENTIFIER
Specifies the correlation name that will be associated with the table in the resulting query. This parameter is ignored when the value of the Type parameter is set to `Insert`.

Examples

```
DRAW Q.STAFF (Type=Select
```

```
DR Q.STAFF (Identifier=A
```

EDIT command

Use the EDIT command to edit a specified object.

Syntax

To edit a database object, use the following command:

```
>>--EDit-+--------+-username.objectname-+----------------------+-<<
         +-QUERY--+                     (+-&&variablename=value-+
         +-PROC---+
         +-FORM---+
         +-TABLE--+
         +-REPORT-+
```

```
>>--EDit--rsbi:/.workspaces/workspacename/objectname--<<
```

To edit an object that is stored in the temporary storage, use the following command:

```
>>--EDit--+-QUERY--+-----<<
          +-FORM---+
          +-PROC---+
          +-REPORT-+
          +-TABLE--+
```

Uppercase letters in each diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the EDIT command:

username
Login of the current user.

objectname
Name of the object that you want to display.

workspacename
The name of the workspace where the object is stored.

&&variablename
If the object is a query or a procedure, assigns a value to each variable that the object uses.

Examples

```
EDIT MYLOGIN.QUERY1
```

```
ED rsbi:/.workspaces/MY_WORKSPACE/QUERY1
```

END command

The END command closes the currently open panel or, if you are on the **Home** panel, terminates your QMF session.

Syntax

```
>>--ENd--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Examples

```
END
```

```
EN
```

ERASE command

The ERASE command removes an object from the database.

Syntax

```
>>-ERase--+--------+--ownername.objectname-----------------------<<
>>-ERase--+--------+--rsbi:/.workspaces/workspacename/objectname--<<
          +-QUERY--+          ( +--Confirm=Yes/No-----+
          +-FORM---+            +--Folder=foldername--+
          +-PROC---+
          +-TABLE--+
          +-FOLDER-+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the ERASE command:

ownername
The name of the user who owns the object.

objectname
The name of the object that you want to erase.

Note: When a QMF query, procedure, or form is erased, it is also erased from every folder that references it.

When a FOLDER object is erased, none of the objects that it references are erased.

CONFIRM
Specifies whether the confirmation dialog is displayed before erasing the object.

FOLDER
Specifies the folder in QMF Catalog that stores the object that you want to erase. When you specify the FOLDER parameter, the QMF object is erased only from the specified folder; the QMF object itself is not erased.

workspacename
The name of the workspace where the object is stored.

Examples

```
ERASE QUERY USERNAME.OBJECTNAME (CONFIRM=YES
```

```
ER QUERY rsbi:/.workspaces/WORKSPACENAME/OBJECTNAME (C=Y
```

```
ERASE QUERY MYQUERY (FOLDER=SALES
```

EXIT command

The EXIT command terminates the QMF session.

Syntax

```
>>--EXIt--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Examples

```
EXIT
```

```
EXI
```

EXPORT command

The EXPORT command saves the currently open object or the object that is stored in a database to a data set or a file.

Syntax

Use the following command to export a QMF query, procedure, form, report, or data from the temporary storage:

```
>>--EXPort-+-QUERY--+-TO-+-datasetname-+--+--------------------+--<<
           +-PROC---+    +-pathname----+ (+-Member=membername---+
                                         +-CONfirm=Yes/No------+
                                         +-Saveatserver=Yes/No-+
           +-FORM---+-TO-+-datasetname-+--+--------------------+
                         +-pathname----+ (+-Language=value------+
                                         +-Member=membername---+
                                         +-CONfirm=Yes/No------+
                                         +-Saveatserver=Yes/No-+
           +-REPORT-+-TO-+-datasetname-+--+--------------------+
                         +-pathname----+ (+-Dataformat=value----+
                                         +-Member=membername---+
                                         +-CONfirm=Yes/No------+
                                         +-Saveatserver=Yes/No-+
                                         +-Width=integer-------+
                                         +-Length=integer------+
                                         +-CCsid=value---------+
           +-DATA---+-TO-+-datasetname-+--+--------------------+
                         +-pathname----+ (+-Dataformat=value----+
                                         +-Outputmode=value----+
                                         +-Member=membername---+
                                         +-CONfirm=Yes/No------+
                                         +-Saveatserver=Yes/No-+
                                         +-DATEformat=value----+
                                         +-Timeformat=value----+
                                         +-Outputmode=value----+
                                         +-LOBSInfile=Yes/No---+
                                         +-LOBSTo=pth1;pth2;---+
                                         +-LOBFile=value-------+
                                         +-CCsid=value---------+
                                         +-Unicode=Yes/No------+
                                         +-Mode=GRID/RAW-------+
                                         +-Columnheadings=Yes/No-+
```

Use the following command to export a QMF query, procedure, form, or table from a database:

```
>>-EXPort-+-QUERY-+-objectname-TO-+-datasetname-+--+--------------------+-<<
          +-PROC--+               +-pathname---+ (+-Member=membername---+
                                                 +-CONfirm=Yes/No------+
                                                 +-Saveatserver=Yes/No-+
          +-FORM--+-formname-TO-+-datasetname-+----+--------------------+
```

```
                              +-pathname----+   (+-Language=value------+
                                             +-Member=membername---+
                                             +-CONfirm=Yes/No------+
                                             +-Saveatserver=Yes/No-+
          +-TABLE-+-tablename-+-TO-+-datasetname-+-+---------------------+
                                  +-pathname----+(+-Dataformat=value----+
                                             +-Outputmode=value----+
                                             +-Member=membername---+
                                             +-CONfirm=Yes/No------+
                                             +-DATEformat=value----+
                                             +-Timeformat=value----+
                                             +-LOBSInfile=Yes/No---+
                                             +-LOBSTo=pth1;pth2;---+
                                             +-LOBFile=value-------+
                                             +-CCsid=value---------+
                                           +-Columnheadings=Yes/No-+
                                           +-Unicode=Yes/No--------+
                                           +-Saveatserver=Yes/No---+
```

Uppercase letters in each diagram show the minimum set of letters required to issue the command.

Unix path is case sensitive. Provide the exact path in the command. For example:

```
export query to /u/ts3157/Test.QUERY
```

or

```
export query to /u/ts3157/Test
```

While running the export command to PS (physical sequential file) or PDS (partioned data set), if you specify the path name in quotes, the path name is not modified and the export command is run as is. For example:

```
export data to 'TS3157.MYDATA.DATA'
```

If you specify the path name without quotes, *<tso user id>* is prefixed and *<dot & object type>* is suffixed to the path name. For example:

```
"export form to MYFORM" is converted to "export form to ts3157.MYFORM.FORM"
```

Parameters

objectname, formname, tablename
The name of the object that you want to export.

datasetname
The name of the TSO data set where you want to export the object.

pathname
The name of the UNIX file where you want to export the object.

MEMBER
Indicates that the object will be exported to a member of TSO partitioned data set.

membername
The name of the member that receives the exported object. Member names are limited to 8 characters. The member name is added (in parentheses) as a suffix to the data set name.

CONFIRM
Specifies whether the confirmation dialog must be displayed before replacing an existing file.

DATEFORMAT
Specifies how the date is formatted in the HTML, CSV, or TXT export file. Date formats are specified by Java date pattern strings. Within date pattern strings, letters from 'A' to 'Z' and from 'a' to 'z' that are not enclosed in quotation marks are interpreted as pattern letters representing the components of a date string. To avoid interpretation, text can be enclosed in single quotation marks (').

Note: If the format string includes spaces, enclose it in quotation marks. For more information about Java format strings, see the Java 2 SDK, Standard Edition Documentation.

TIMEFORMAT
Specifies how the time is formatted in the HTML, CSV, or TXT export file. Time formats are specified by Java time pattern strings. Within time pattern strings, letters from 'A' to 'Z' and from 'a' to 'z' that are not enclosed in quotation marks are interpreted as pattern letters representing the components of a time string. To avoid interpretation, text can be enclosed in single quotation marks (').

Note: If the format string includes spaces, enclose it in quotation marks. For more information about Java format strings, see the Java 2 SDK, Standard Edition Documentation.

LOBSINFILE
Specifies whether the LOBs must be included in the exported data.

Note: For the EXPORT TABLE and EXPORT DATA commands, this parameter is available only for the IXF data format.

LOBSTO
Specifies the location to save the LOBs.

Note: For the EXPORT TABLE and EXPORT DATA commands, this parameter is available only for the IXF data format.

LOBFILE
Specifies the base name of the exported LOBs.

Note: For the EXPORT TABLE and EXPORT DATA commands, this parameter is available only for the IXF data format.

CCSID
Specifies the code page (coded character set identification number) to use when saving the file. This value can either be an integer or the Java™ encoding name of the code page.

WIDTH
Specifies the width in units for a report page.

LENGTH
Specifies the length in units for a report page.

COLUMNHEADINGS
Specifies whether the column headers will be exported. This parameter is only available for export into HTML, CSV, or TEXT files.

UNICODE
Specifies whether the graphic columns will be saved as UNICODE. This option is only applicable when saving data in IXF format.

MODE
Specifies whether the query result set is saved with formatting and added calculated columns. You can specify one of the following values:

- **GRID** - specifies that all of the data as it is currently formatted in the current query result set will be saved. All calculated columns that have been added to the query result set will be saved.

  This is the default value for the PDF, XLS, and XLSX formats.

  Note: MODE **GRID** exports labels if the value of the DSQDC_COL_LABELS global variable is set to 1. MODE **GRID** exports names if the value of the DSQDC_COL_LABELS global variable is set to 0.
- **RAW** - specifies that all of the data in the current query result set will be saved. None of the formatting that has been applied to the data will be saved. None of the calculated columns that have been added to the query results will be saved.

  This is the default value for all formats other than PDF, XLS, and XLSX.

  Note: MODE **RAW** always exports names. This also applies when the MODE parameter is omitted.

Note: The parameter is ignored if Dataformat=XLS, XLSX.

SAVEATSERVER
Specifies whether or not to include the **Root output directory** in the export path of an object. In QMF Z Client, you can specify only the path that lies in the root directory set by administrator. Otherwise, the export will be forbidden and an error will occur.

LANGUAGE
Specifies whether a form is exported in English or in the current session language. A form that is exported in English can be run in any session. A form exported in the session language can only be run in a session of the same language. The default value is provided by the DSQEC_FORM_LANG global variable.

DATAFORMAT
Specifies the file format for the object that you are exporting. Valid values are:

HTML
The HyperText Markup Language format. You can specify HTML only when you are exporting a report. This is the default format for UNIX files. The TSO data set or UNIX file can be transferred to a web server where it can be viewed via a web browser. The maximum length of an exportable data row for this format is 32 KB. You can use the XML format to export character data if you need support for record lengths beyond this limit; the XML format supports record lengths of up to 2 GB.

IXF
The Integrated Exchange Format. This can be used only when exporting data objects and tables. The maximum length of an exportable data row for this format is 32 KB. You can use the XML format to export character data if you need support for record lengths beyond this limit; the XML format supports record lengths of up to 2 GB.

DBF
The dBase database file format. This option can be used only when exporting data objects and tables.

XML
The Extensible Markup Language format. The data is exported as an XML document in Unicode UTF-8 format with a CCSID of 1208. You can use this option only when exporting data objects or tables, and this is the only option when exporting data or tables to a UNIX file.

The maximum length of an exportable data row for this format is 32 KB.

If you are working with this format, ensure that all characters in the XML data that you want to export are supported by the XML parser.

PDF
The Adobe Portable Document Format. This option can be used only when exporting reports.

XLS
The Microsoft Excel Binary File format. This option can be used only when exporting data objects and tables.

XLSX
The Microsoft Excel Binary File format used in Microsoft Excel 2007 and later. This option can be used only when exporting data objects and tables.

TEXT
The format for exporting reports without control information. This option can be used only when exporting reports.

CSV
The Comma-Separated Values format. This option can be used only when exporting data objects and tables.

The maximum LRECL of data to be exported in this format is 32756.

OUTPUTMODE
Specifies how to represent numeric data in the exported object. This option can only be specified when the export file format is IXF. Valid values are:

BINARY
Numeric column data is encoded in its native internal format.

This does not apply to any numeric data in the header records of the exported object. These are always represented in a character format.

CHARACTER
Numeric column data is converted to a character representation in EBCDIC.

Examples

```
EXPORT PROC KATIE.PANELID TO dataset
```

```
EXPORT QUERY FIRSTQ TO LOREN (MEMBER=GAMMA
```

FAVORITE command

The FAVORITE command adds an object to the list of favorite objects.

Syntax

```
>>--FAvorite--objectname--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Examples

```
FAVORITE example_object
```

```
FA example_object
```

FORWARD command

The FORWARD command scrolls the scrollable area towards the bottom.

Syntax

```
>>--FOrward-----+---------+--<<
                +--value--+
                +--Max----+
                +--Half---+
                +--Page---+
                +--CSR----+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the FORWARD command:

A number in the range 1 - 9999
Scrolls the number of pages or rows.

MAX
Scrolls to the bottom.

HALF
Scrolls by half a page.

PAGE
Scrolls by one page.

CSR
Scrolls based on the position of the cursor. If the cursor is in a scrollable area, scrolls to the bottom. If the cursor is outside of or at the end the scrollable area, scrolls one page.

If you issue the FORWARD command without a parameter, the default parameter is used. You can view or change the default parameter in the **Scroll** field, which is located in the lower right corner of the screen.

Examples

```
FORWARD 4
```

```
FORWARD MAX
```

```
FO M
```

HELP command

The HELP command displays the Help topic for the specified panel or for the currently displayed panel or error message.

Syntax

```
>>--Help--+-----------+--<<
          +-panelid---+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

If you use the HELP command without any parameters, it displays the help topic for the currently open panel or for the error message that is displayed above the command line.

To display the help topic for a specific panel, specify the ID of the panel that you want to display as the value for the HELP command. To view the full list of QMF panels and their IDs, see Appendix H, “IDs of QMF panels,” on page 137.

If the topic that matches the specified ID is not found, the Help contents is displayed.

Examples

```
HELP
```

```
H
```

IMPORT command

The IMPORT command copies the contents of a TSO data set or UNIX file into QMF temporary storage or into the database.

Syntax

Use the following command to import a QMF object into temporary storage:

```
>>--IMport-+-QUERY-+-FROM-+-datasetname-+--+----------------+-<<
           +-PROC--+      +-pathname----+ (+Member=mbrname--+
           +-FORM--+      +-datasetname-+--+----------------+-<<
                          +-pathname----+ (+-Member=mbrname-+
                                          +-Language=value-+
           +-DATA--+      +-datasetname-+--+----------------+-<<
```

```
                          +-pathname----+ (+Member=mbrname--+
                                          +-Lobsfrom=value-+
```

Note: The MEMBER parameter is accepted only when you import from a TSO data set.

Use the following command to import a QMF query, procedure, form, or table into the database:

```
>>-IMport-+-QUERY-+-objname-FROM-+-datasetname-+--------------------+-<<
          +-PROC--+              +-pathname----+ (+-Member=mbrname--+
                                                 +-CONfirm=YES/NO--+
                                                 +-SHare=value-----+
                                                 +-COMment=value---+
                                                 +-Folder=value----+
          +-FORM--+-objname-FROM-+-datasetname-+--------------------+
                                 +-pathname----+ (+-Language=value--+
                                                 +-Member=mbrname--+
                                                 +-CONfirm=YES/NO--+
                                                 +-SHare=YES/NO----+
                                                 +-COMment=value---+
                                                 +-Folder=value----+
         +-TABLE-+-tblname-FROM-+-datasetname-+---------------------+
                                +-pathname----+(+-Action=value------+
                                               +-Member=mbrname----+
                                               +-CONfirm=YES/NO----+
                                               +-COMment=value-----+
                                               +-ACCElerator=value-+
                                               +-SPACE=value-------+
                                             +-SPACE DATABASE=value-+
```

Unix path is case sensitive. Provide the exact path in the command. For example:

```
import query from /u/ts3157/Test.QUERY
```

or

```
import query from /u/ts3157/Test
```

While running the import command from PS or PDS, if you specify the path name in quotes, the path name is not modified and the import command is run as is. For example:

```
import data from 'TS3157.MYDATA.DATA'
```

If you specify the path name without quotes, *<tso user id>* is prefixed and *<dot & object type>* is suffixed to the path name. For example:

```
"import form from MYFORM" is converted to "import form from ts3157.MYFORM.FORM"
```

Parameters

The following parameters can be specified for the IMPORT command:

objname, tblname
Specifies the name of the object that you want to import.

datasetname, pathname
Specifies the name of the TSO data set or the UNIX pathname whose contents you want to import.

MEMBER
Indicates that the imported object is a member in a TSO partitioned data set.

mbrname
Specifies the name of the member whose contents you want to import. Member names are limited to 8 characters. The member name is added (in parentheses) as a suffix to the data set name.

CONFIRM
Specifies whether you want to display a confirmation dialog before replacing an existing object.

COMMENT
Specifies a comment with the imported object. Make sure to enclose the comment text in quotation marks.

SHARE
Specifies whether other users are allowed to use the imported object.

LANGUAGE
Specifies whether the QMF keywords that are contained within the imported form are recorded in English or in the current NLF session language. Valid values are `ENGLISH` and `SESSION`.

ACTION
Specifies whether the entire database table will be replaced or the new data will be appended to the existing table. Valid values are: `REPLACE` and `APPEND`.

FOLDER
Specifies the folder to which you want to import the object.

SPACE
Specifies both the database name and the table space name to save the table in a particular database container and table space.

Note:

- `database.tablespace` is used for Db2 for z/OS databases.
- `tablespace` is used for Db2 for LUW databases.

SPACE DATABASE
Specifies only the database name to save the table in a particular database container with the table space created automatically under the name of the created table.

Note: This parameter is used only for z/OS databases.

ACCELERATOR
Specifies the name of the accelerator that you want to use to save your data. The ACCELERATOR keyword can be up to 128 characters long. The ACCELERATOR keyword cannot be specified if the SPACE keyword is already specified for the command, unless the value of the DSQEC_SAV_ALLOWED global variable is set to 5. The default value for the ACCELERATOR keyword is taken from the DSQEC_SAV_ACCELNM global variable.

The ACCELERATOR keyword is supported only on Db2 z/OS servers that support IDAA.

LOBSFROM
Specifies the location where the saved LOBs are stored.

Example

```
IMPORT TABLE MYTABLE FROM NEW.ROWS (ACTION=APPEND
```

INSERT command

Use the `INSERT` command to create certain items on certain QMF panels.

Syntax

```
>>--INSert--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Commentary

Use the `INSERT` command to create the following items:

- A column on the **Form.Main** panel or the **Form.Columns** panel.

- A calculation expression on the **Form.Calculations** panel
- A conditional expression on the **Form.Conditions** panel.
- A line of text on the **Form.Break**, **Form.Detail**, **Form.Final**, or **Form.Page** panels.
- A list item on any of the Prompted Query editor panels.

To create an item, complete the following steps:

1. Type INSERT in the command line.
2. Position the cursor on the item after which you want to insert a new one.
3. Press Enter.

Examples

```
INSERT
```

```
INS
```

ISPF command

The ISPF command calls the Interactive System Product Facility (ISPF).

Syntax

```
>>--ISpf--+--------+--<<
          +-option-+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

You can specify the OPTION parameter for the ISPF command. This parameter specifies the initial option to pass to ISPF. For example, if you enter 3, the third ISPF panel option is selected.

Examples

```
ISPF 3
```

```
IS 4
```

LEFT command

The LEFT command scrolls toward the left boundary of a panel.

Syntax

```
>>--LEft-----+---------+--<<
             +--value--+
             +--Max----+
             +--Half---+
             +--Page---+
             +--CSR----+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the LEFT command:

value
Scrolls the scrollable area to the left by this number of pages or columns (a whole number ranging from 1 through 9999). The scrolling unit, that is pages or columns, depends on the currently open panel.

MAX
Scrolls to the leftmost boundary of the panel.

HALF
Scrolls the scrollable area to the left by half a page.

PAGE
Scrolls the scrollable area to the left by one page.

CSR
The scroll is based on the position of a cursor. The column on which the cursor is placed is moved to the left boundary of the scrollable area. If the cursor is positioned outside of the scrollable area or on its left boundary, a full-page scroll occurs.

If you issue the LEFT command without a parameter, the default parameter is used. You can view or change the default parameter in the **Scroll** field, which is located in the lower right corner of the screen.

Examples

```
LEFT
```

```
LE MAX
```

```
LE M
```

LIMIT LOCAL command

The `LIMIT LOCAL` command creates a set of selectable values for a local variable. With this command issued, the **Prompt Variables** dialog allows you to select one of the pre-defined values. Variables that were created via the `LIMIT LOCAL` command are available only for the current object (query, report), do not appear in the Global Variables list, and do not affect other procedures.

Syntax

```
>>--LIMit-Local-(variablename=value, ...--<<
```

Parameters

For the `LIMIT LOCAL` command, you can specify the `variablename` parameter. It specifies the name of the local variable that you want to work with.

Each value that you specify for a local variable can be from 1 to 55 characters long. To create a set of selectable values, separate them with a semicolon.

Example

```
LIMIT LOCAL (Var1=1;2;3 Var2=2;4;5
```

```
LIM L (Var1=1;2;3 Var2=2;4
```

LIST command

The LIST command displays the **Object List** panel.

Syntax

```
>>-LISt--+---------+--+-----------------------------------------+-<<
         +-QUeries-+ (+-Folder=rsbi:/.workspaces/workspacename--+
         +-Tables--+  +-Owner=authorizationid/patternstring/ALL-+
         +-FORms---+  +-Name=ALL/objectname/patternstring-------+
         +-Procs---+  +-Location=servername--------------------+
         +-FOLders-+
         +-All-----+
         +-QMF-----+
         +-Home----+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Depending on the object that you specify for the LIST command, the **Object List** panel displays the following objects:

LIST
The **Object List** panel displays the object list that was viewed the last. When issued for the first time during a session, the LIST command behaves like the LIST HOME command.

LIST QUERIES
The **Object List** panel displays queries that can be accessed from the current data source.

LIST TABLES
The **Object List** panel displays tables that can be accessed from the current data source.

LIST FORMS
The **Object List** panel displays forms that can be accessed from the current data source.

LIST PROCS
The **Object List** panel displays procedures that can be accessed from the current data source.

LIST FOLDERS
The **Object List** panel displays folders that can be accessed from the current data source.

LIST ALL
The **Object List** panel displays all objects that can be accessed from the current data source.

QMF
The **Object List** panel displays QMF queries, forms, procedures, and folders that can be accessed from the current data source.

LIST HOME
Opens the **List** panel where you must specify the data source or the workspace whose objects you want to display and press Enter. After you press Enter, QMF displays all objects that can be accessed from the selected data source or workspace.

Parameters

The following parameters can be specified for the LIST command:

FOLDER
Specifies the folder in QMF Catalog or the workspace folder whose contents you want to list. The default value is provided by the DSQEC_CURR_FOLDER global variable.

OWNER
Specifies the owner whose objects you want to list.

To filter the list by owner names, use the % and _ characters. Use the % character to substitute any string of characters and use the _ to substitute any single character.

For example, to get a list of all objects whose owner name includes a certain character string, type that character string followed by or surrounded by the % character.

NAME
Specifies the full name of the object that you want to display or a part of it.

To filter the list by object names, use the % and _ characters. Use the % character to substitute any string of characters and use the _ to substitute any single character.

For example, to get a list of all objects whose names include a certain character string, type that character string followed by or surrounded by the % character.

LOCATION
Specifies the location that contains the objects that you want to list.

Examples

```
LIST QUERIES
```

```
LIS AL (F=rsbi:/.workspaces/MY_WORKSPACE
```

```
LIST TABLES (N=%TA%
```

MAIL TO command

The MAIL TO command sends the specified object as an Internet Mail attachment.

Syntax

To email an object that is stored in a database, issue the following command:

```
>>-MAil-+---------+-objectname-TO-emailaddress--<<
        +--QUERY--+             (+-FRom=address-------------+
        +--PROC---+              +-CClist=address1;address2-+
        +--FORM---+              +-SUbject=subject----------+
                                 +-Body=text----------------+
                                 +-FOrmat=text/HTML---------+
                                 +-SMTPServer=server_name---+
                                 +-SMTPPOrt=port_number-----+
                                 +-SMTPUser=username--------+
                                 +-SMTPPAssword=password----+
                       +-DATEformat=java_date_format_string-+
                       +-Timeformat=java_time_format_string-+
```

```
>>-MAil-+---------+-objectname-TO-emailaddress--<<
        +--TABLE--+             (+-FRom=address-------------+
                                 +-CClist=address1;address2-+
                                 +-Dataformat=value---------+
                                 +-SUbject=subject----------+
                                 +-Body=text----------------+
                                 +-FOrmat=text/HTML---------+
                                 +-SMTPServer=server_name---+
                                 +-SMTPPOrt=port_number-----+
                                 +-SMTPUser=username--------+
                                 +-SMTPPAssword=password----+
                       +-DATEformat=java_date_format_string-+
                       +-Timeformat=java_time_format_string-+
```

Uppercase letters in each diagram show the minimum set of letters required to issue the command.

To email an object that is stored in the temporary storage, issue the following command:

```
>>-MAil-+-QUERY-+-TO-emailaddress--<<
        +-PROC--+             (+-FRom=address-------------+
        +-FORM--+              +-CClist=address1;address2-+
                               +-SUbject=subject----------+
                               +-Body=text----------------+
                               +-FOrmat=text/HTML---------+
                               +-SMTPServer=server_name---+
                               +-SMTPPOrt=port_number-----+
                               +-SMTPUser=username--------+
                               +-SMTPPAssword=password----+
```

```
                        +-DATEformat=java_date_format_string-+
                        +-Timeformat=java_time_format_string-+
```

```
>>-MAil-+-DATA-+-TO-emailaddress-+-----<<
                                 (+-FRom=address-------------+
                                  +-CClist=address1;address2-+
                                  +-Dataformat=value---------+
                                  +-SUbject=subject----------+
                                  +-Body=text----------------+
                                  +-FOrmat=text/HTML---------+
                                  +-SMTPServer=server_name---+
                                  +-SMTPPOrt=port_number-----+
                                  +-SMTPUser=username--------+
                                  +-SMTPPAssword=password----+
                           +-DATEformat=java_date_format_string-+
                           +-Timeformat=java_time_format_string-+
```

```
>>-MAil-+-REPORT-+-TO-emailaddress--<<
                                 (+-FRom=address-------------+
                                  +-CClist=address1;address2-+
                                  +-SUbject=subject----------+
                                  +-Body=text----------------+
                                  +-FOrmat=text/HTML---------+
                                  +-SMTPServer=server_name---+
                                  +-SMTPPOrt=port_number-----+
                                  +-SMTPUser=username--------+
                                  +-SMTPPAssword=password----+
                           +-DATEformat=java_date_format_string-+
                           +-Timeformat=java_time_format_string-+
                                  +-Method=value-------------+
                                  +-Type=value---------------+
```

Uppercase letters in each diagram show the minimum set of letters required to issue the command.

To email a message, issue the following command:

```
>>-MAil-+--MESSAGE--+-TO-emailaddress--<<
                                (+-FRom=address-------------+
                                 +-CClist=address1;address2-+
                                 +-SUbject=subject----------+
                                 +-Body=text----------------+
                                 +-FOrmat=text/HTML---------+
                                 +-SMTPServer=server_name---+
                                 +-SMTPPOrt=port_number-----+
                                 +-SMTPUser=username--------+
                                 +-SMTPPAssword=password----+
                                 +-Attachment=file1;file2---+
                        +-DATEformat=java_date_format_string-+
                        +-Timeformat=java_time_format_string-+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the `MAIL TO` command:

emailaddress
Specifies the email address to which you want to send your object.

FROM
Specifies the email address of the sender.

CCLIST
Specifies one or several recipient email addresses.

DATAFORMAT
Specifies the file format of the attached data object. Valid values are `CSV`, `DBF`, `HTML`, `IXF`, `PDF`, `QMF`, `SHP`, `TEXT`, `WQML`, `XLS`, `XLSX`, and `XML`.

If you omit this parameter, the DSQQW_EXP_DT_FRMT global variable supplies the format to be used. For detailed information about the DSQQW_EXP_DT_FRMT global variable, see “DSQQW global variables” on page 109.

SUBJECT
Specifies the email subject line reference.

BODY
Specifies the contents of the email message.

FORMAT
Specifies the email format. Supported formats are Text and HTML.

SMTPSERVER
Specifies the name of the SMTP server that you want to use.

SMTPPORT
Specifies the number of the SMTP server port that you want to use.

SMTPUSER
Specifies the user name for authorization on the SMTP server.

SMTPPASSWORD
Specifies the password for authorization on the SMTP server.

DATEFORMAT
Specifies how the date is formatted in the HTML, CSV, or TXT export file.

Date formats are specified by Java date pattern strings. Within date pattern strings, letters from 'A' to 'Z' and from 'a' to 'z' that are not enclosed in quotation marks are interpreted as pattern letters representing the components of a date string.

To avoid interpretation, the text must be enclosed in single quotation marks (') .

If the format string includes spaces, enclose it in quotation marks. For more information about Java format strings, see the Java 2 SDK, Standard Edition Documentation.

TIMEFORMAT
Specifies how the time is formatted in the HTML, CSV, or TXT export file.

Time formats are specified by Java time pattern strings. Within time pattern strings, letters from 'A' to 'Z' and from 'a' to 'z' that are not enclosed in quotation marks are interpreted as pattern letters representing the components of a time string.

To avoid interpretation, the text must be enclosed in single quotation marks (').

If the format string includes spaces, enclose it in quotation marks. For more information about Java format strings, see the Java 2 SDK, Standard Edition Documentation.

ATTACHMENT
Specifies the name and path to the files that will be attached to the email. If you want to attach a data set or a UNIX file, make sure to enclose the path to the object in double quotation marks.

TYPE
Specifies the format into which the report is converted. Valid values are: PDF, HTML, and TEXT.

METHOD
Specifies whether the report is divided into pages. Valid values are: SPLIT and CONT.

Examples

```
MAIL QUERY TO abc@mail.com (SU="subj", SMTPS=smtp.example.com
```

```
MA QUERY TO abc@mail.com (SU="subj", SMTPS=smtp.example.com
```

REFRESH command

The REFRESH command refreshes the list on the **Object List** panel.

The REFRESH command can be used on the **Object List** panel, to update the list.

Syntax

```
>>--REFresh--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Examples

```
REFRESH
```

```
REF
```

RENAME command

The RENAME command changes the name of the specified object.

Syntax

```
>>--REName-+--------+--source_object_name-TO-new_object_name--<<
           +-QUERY--+
           +-FORM---+
           +-PROC---+
           +-TABLE--+
           +-FOLDER-+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the RENAME command:

source_object_name
Specifies the current name of the object that you want to rename.

new_object_name
Specifies the new name for the object that you want to rename.

Note: If the object that you are renaming is stored in QMF Catalog or belongs to the currently logged in owner, type only the name of the object as the value for both parameters. If the object is stored in a repository or belongs to a different owner, type the full path to the object as the value for both parameters.

Examples

```
RENAME QUERY QUERY_old TO QUERY_new
```

```
REN PROC rsbi:/.workspaces/.../PROC1 TO rsbi:/.workspaces/.../PROC2
```

RESET command

The RESET command restores the specified object to its default state. The RESET command functions a little bit differently depending on the object that is being reset.

Syntax

```
>>--RESet-+---------+-----------------<<
          +-Query---+
                    (+-(Language=value-+
                     +--Model=Rel------+
          +-Proc----+
```

```
          +-Data----+
          +-CONtext-+
          +-FORM----+-------------+
                    +-.BREAK1-----+
                    +-.BREAK2-----+
                    +-.BREAK3-----+
                    +-.BREAK4-----+
                    +-.BREAK5-----+
                    +-.BREAK6-----+
                    +-.CALc-------+
                    +-.COLumns----+
                    +-.CONditions-+
                    +-.Detail-+-----------------+
                              (+-Variation=value-+
                               +-Using=value-----+
                    +-.Final------+
                    +-.Options----+
                    +-.Page-------+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The RESET QUERY and RESET PROC commands create a new object and close the currently open one.

For the RESET QUERY command, you can specify the following parameters:

LANGUAGE
Specifies which query language to use. Valid values are:

SQL
Specifies that the query that you want to reset was written in SQL.

PROMPTED
Specifies that the query that you want to reset was created by using the prompted query editor.

MODEL
Specifies the data model used for queries. REL for relational data is the only supported value.

The RESET DATA command closes the currently open query result set.

The RESET CONTEXT command restores the user context of the current user (that is the list of global variables, the list of favorites, the list of recently used objects, the contents of the **Favorite Actions** panel, and so on) to its default state.

The RESET FORM command restores the currently displayed Form panel to its default state.

The following options can be specified for the RESET FORM.DETAIL command:

VARIATION
Specifies which detail variation to reset. If this option is omitted, the current detail variation is reset. Valid values are integers from 1 to 99 or ALL. The ALL value resets all detail variations to their default values.

USING
Specifies which detail variation to use as a template to reset or create another variation. This can be helpful if you make a number of modifications to a detail panel and want to create another panel with similar changes. Valid values are integers from 1 to 99.

Examples

```
RESET QUERY
```

```
RESET QUERY(LANG=PROMPTED
```

```
RES FORM.F
```

RESET GLOBAL command

The RESET GLOBAL command deletes global variables that were created by an administrator or a user, leaving only the global variables that were pre-defined by the application developers.

Syntax

```
>>--RESet Global--+-All-----------------+--<<
                  +-(varname1, varname2-+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the RESET GLOBAL command:

varname
Specifies the name of a variable that you want to delete. You can name up to 10 variables. Use comma or blank space as a separator.

ALL
Deletes all global variables that were created by an administrator or a user.

Examples

```
RESET GLOBAL ALL
```

```
RES G (example_variable1, example_variable2
```

RESET KEY command

The RESET KEY command resets the specified function key to its default state.

Syntax

```
>>--RESet Key(Panelid=+----ALL---+, Keyid=+--ALL---+--<<
                      +--CURRENT-+        +-key_id-+
                      +-panel_id-+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the RESET KEY command:

PANELID
Specifies the panel that contains the key that you want to reset. Possible values are:

- ALL - resets the specified key on all the panels that use it.
- CURRENT - resets the specified key on the currently open panel.
- panel_id - resets the specified key on the panel whose ID you enter as the value for the PANELID parameter. To view the complete list of QMF panels and their IDs, see Appendix H, “IDs of QMF panels,” on page 137.

KEYID
Specifies the key that you want to reset. Possible values are:

- ALL - resets all keys on the specified panel.
- key_id - resets the specific key on the specified panel. Valid values are integers from 1 to 24.

Examples

```
RESET KEY (PANELID=CURRENT, KEYID=ALL
```

```
RES K (P=example_panel, K=ALL
```

```
RES K (P=example_panel, K=10
```

RETRIEVE command

The RETRIEVE command re-displays the most recent command line entry.

Syntax

To display the most recent command line entry, type RETRIEVE in the Command line.

```
>>--RETrieve--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

To go back more than one command line entry, enter several ? characters in the command line, where the number of ? characters determines how far you get in the command line history.

```
>>--??--<<
```

Examples

```
RETRIEVE
```

```
RET
```

```
???
```

RIGHT command

The RIGHT command scrolls towards the right boundary of a panel.

Syntax

```
>>--RIght----+---------+--<<
             +--value--+
             +--Max----+
             +--Half---+
             +--Page---+
             +--CSR----+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the RIGHT command:

value
Scrolls the scrollable area to the right by this number of pages or columns (a whole number ranging from 1 through 9999). The scrolling unit, that is pages or columns, depends on the currently open panel.

MAX
Scrolls to the rightmost boundary of the panel.

HALF
Scrolls the scrollable area to the right by half a page.

PAGE
Scrolls the scrollable area to the right by one page.

CSR
The scroll is based on the position of a cursor. The column on which the cursor is placed is moved to the right boundary of the scrollable area. If the cursor is positioned outside of the scrollable area or on its left boundary, a full-page scroll will occur.

If you issue the RIGHT command without a parameter, the default parameter is used. You can view or change the default parameter in the **Scroll** field, which is located in the lower right corner of the screen.

Examples

```
RIGHT
```

```
RIGHT MAX
```

```
RI HALF
```

RUN command

Use the RUN command to run queries or procedures.

Syntax

To run a query, use the following command:

```
>>-RUn-+-------+-objectname-+-------------------------+-<<
       +-QUERY-+           (+-ACCElerator=value-------+
                            +-ACCELERATORDATABASE=name+
                            +-ACTion=append/replace---+
                            +-COMment=comment_text----+
                            +-CONfirm=Yes/No----------+
                            +-Form=FORM/formname------+
                            +-METHOD=method_name------+
                            +-MODE=GRID/RAW-----------+
                            +-ROWIDADD=YES/NO---------+
                            +-ROWIDDISP=value---------+
                            +-ROWIDNAME=text----------+
                            +-ROWLimit=integer--------+
                            +-SPACE=value-------------+
                            +-SPACE DATABASE=database-+
                            +-SCOPE=integer-----------+
                            +-Table=tablename---------+
                            +-&&variablename=value----+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

To run a procedure, use the following command:

```
>>-RUn-+------+-objectname-+--+----------------------+-<<
       +-PROC-+              (+-&&variablename=value-+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the RUN QUERY command:

objectname
Specifies the name of the query that you want to run.

CONFIRM
Specifies whether or not to display a confirmation dialog before replacing or changing an object as a result of this command.

FORM
If you are running a query that must return a report, use this parameter to indicate which QMF form to use to format the selected data. You can specify the keyword FORM to use the form object that is currently stored in the temporary storage, or specify the name of a form that is saved in the database. Note, that if the temporary storage contains more than one open form at the same time, the most recently opened one will be used.

ROWLIMIT
Specifies the maximum number of table rows to include in the query result set.

&&variablename
Assigns a value to a variable in the query. The variable name can be 1 to 17 characters long and the value can be 1 to 55 characters long. You can specify any number of variables and values with the RUN command. The variable name must be prefaced with two ampersands and enclosed in quotation marks.

ACTION
Specifies whether you want to replace the entire database table with the data returned by the query or to append the data to the existing table. This option is valid only if the **TABLE** option is also specified. Valid values are REPLACE and APPEND.

TABLE
Specifies that you want to insert the query results into a table. Valid value for this parameter is the name of the table.

COMMENT
Creates a comment and stores it with the data that is returned by the query and inserted into the specified table. This option is valid only if the **TABLE** option is also specified.

SPACE
Specifies the storage space to hold any tables that are created by the SAVE DATA command. If you leave the value of this parameter blank, the application uses the default space chosen by the database manager program.

SPACE DATABASE
Specifies the database name to save the table in a particular database container. The table space is created automatically under the name of the created table. This parameter is used only for z/OS databases.

ACCELERATOR
Specifies the name of the accelerator in which the table will be created.

ACCELERATORDATABASE
Specifies the name of the database that you want to use to save accelerator-only tables. The ACCELERATORDATABASE keyword can be up to 128 characters long. The default value of the ACCELERATORDATABASE parameter is taken from the DSQEC_SAV_ACCELDB global variable. If the DSQEC_SAV_ACCELDB global variable value is not empty, the database specified by the SPACE keyword is ignored.

MODE
Specifies whether the query result set is saved with the formatting and added calculated columns. Valid values are:

RAW
Saves the query result set without the formatting and added calculated columns.

GRID
Saves the query result set with the formatting and added calculated columns.

METHOD
Specifies the method of saving the query result set. Valid values are:

REGULAR
Sends the query result set data from the client back to the database server where it is inserted into the table.

FAST
Reruns the query at the server and inserts the query results directly into the table.

FASTSAFE
Reruns the query at the server without the ORDER BY clauses and inserts the query results directly into the table.

ROWIDADD
Specifies whether to add the Row ID column to the table.

ROWIDDISP
Specifies the disposition of the new Row ID column.

ROWIDNAME
Specifies the name for the new Row ID column.

SCOPE
Specifies the commit scope of the data.

The following parameters can be specified for the RUN PROC command:

objectname
Specifies the name of the procedure that you want to run.

&&variablename
Assigns a value to a variable in the procedure. The variable name can be 1 to 17 characters long and the value can be 1 to 55 characters long. You can specify any number of variables and values with the RUN command. The variable name must be prefaced with two ampersands and enclosed in quotation marks.

Examples

```
RUN PROC EXAMPLE_PROCEDURE (&&VAR='example_value'
```

```
RU QUERY EXAMPLE_QUERY (&&VAR='example_value' rowlimit=5
```

RUNTSO command

Use the RUNTSO command to start the Q.DSQQMFSP stored procedure from a CALL statement. The RUNTSO command passes the name of a query or procedure to run on QMF for TSO. The query or procedure that is named in this command must exist in the QMF catalog in the subsystem against which the RUNTSO command runs.

Syntax

```
>>-RUNTSO-objectname--+---------------------+-<<
                      (+-Tracelevel=+-None-+
                                    +-L2---+
                                    +-All--+
                                    +-Ptf--+
                       +-L2DESTINATION=None/Dsqdebuf-+
                       +-LANGUAGE=value------+
```

Parameters

The following parameters can be specified for the RUNTSO command:

objectname
Specifies the name of a QMF procedure or query that will run after QMF starts. All types of QMF queries are accepted. The procedure can be either a QMF linear procedure or a procedure with logic.

The query or procedure that is named in this parameter must exist in the QMF catalog subsystem on which the stored procedure interface components are installed.

One result set is returned if the specified object is a query. Up to 21 result sets can be returned from a procedure, including the trace output which is returned as the last result set when the TRACELEVEL parameter is set to L2 and the L2DESTINATION parameter is set to `blank` or `null`.

TRACELEVEL
Specifies the level of trace detail. Valid values are:

NONE
Trace output is not generated. This is the default option.

L2
Traces QMF messages and commands at the highest level of detail. The destination of the trace output depends on the **L2DESTINATION** setting.

ALL
Traces QMF activity at the highest level of detail, including program initialization errors and other errors that might occur before the user profile is established. Trace output is sent to the DSQDEBUG DD card.

PTF
This option is used to verify that the stored procedure interface is running correctly. Do not use this option unless instructed to do so by an IBM® Software Support representative.

L2DESTINATION
Specifies the destination for the trace log when the TRACELEVEL is set to L2. Valid values are:

NONE
Returns the trace output as the last result set from the stored procedure run. This is the default value.

DSQDEBUG
Sends the trace output to the DSQDEBUG DD card.

LANGUAGE
Specifies the language in which QMF runs.

This parameter can be a one-letter language identifier from the following table. Valid values are:

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

The default value depends on the *DSQEC_NLFCMD_LANG* variable. For example, if *DSQEC_NLFCMD_LANG*=0 then *DSQAO_NLF_LANG* is used as the language.

If DSQEC_NLFCMD_LANG=1 then E is used.

Example

```
RUNTSO Q.STAFF (TRACELEVEL=NONE L2DESTINATION=NONE LANGUAGE=E
```

SAVE AS command

The SAVE AS command saves the object that is currently displayed in the editor to a database.

Syntax

Use the SAVE AS command to save objects to QMF catalog or to a workspace.

```
>>--SAve-+-Query-+-AS-objectname-(+----------------+--<<
         +-Proc--+
                                  +-CONfirm=Yes/No-+
                                  +-Share=Yes/No---+
                                  +-COMment=value--+
                                  +-Folder=name----+
```

```
>>--SAve--Form-AS-objectname-(+----------------+--<<
                              +-Language=value-+
                              +-CONfirm=Yes/No-+
                              +-Share=Yes/No---+
                              +-COMment=text---+
                              +-Folder=name----+
```

```
>>--SAve--Data-AS-tablename-(+----------------+--<<
                             +-ACTion=value------+
                             +-CONfirm=Yes/No----+
                             +-COMment=text------+
                             +-SPACE=value-------+
                             +-ACCElerator=value-+
                        +-ACCELERATORDATABASE=name+
                        +-METHOD=method_name------+
                        +-MODE=GRID/RAW-----------+
                        +-RESULTSET=integer-------+
                        +-ROWIDADD=YES/NO---------+
                        +-ROWIDDISP=value---------+
                        +-ROWIDNAME=text----------+
                        +-ROWLimit=integer--------+
                        +-SPACE=value-------------+
                        +-SPACE DATABASE=database-+
                        +-SCOPE=integer-----------+
```

Uppercase letters in each diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the SAVE AS command:

objectname
The name to assign to the object (query, form, procedure, or table) when it is saved. When `objectname` refers to an object of the same type that already exists in the database, QMF replaces the existing object with the one that you are saving.

If you want to save the object to a workspace, type the following string as a value for the `objectname` parameter: `rsbi:/.workspaces/WORKSPACENAME/OBJECTNAME`.

tablename
The name for the table in the database. If the object already exists, the application replaces or appends the existing table according to the value of the ACTION parameter on the SAVE command. If the table does not exist, a new table is created using the specified column names and labels.

ACTION
Specifies whether to replace the entire database table or append the data to an existing table. Valid values are REPLACE and APPEND. A table can replace or be appended only to a table with the same number of columns, and the corresponding columns must have the same data type and length. If corresponding columns do not have the same data type or length, they might be automatically converted from one data type or length to another, depending on the level of support that your database management software offers for implicit casting.

LANGUAGE
Specifies whether a form is saved in English or in the current session language. Valid values are ENGLISH and SESSION. A form that is saved in English can be run in any NLF session. A form that is saved in the session language can only be run in a session of the same language.

CONFIRM
Specifies whether or not to display a confirmation dialog before replacing or changing an object as a result of this command.

SHARE
Specifies whether other users are allowed to use the saved object.

COMMENT
Stores a comment with the saved object. Enclose the comment text in quotation marks, double quotation marks, or parentheses.

FOLDER
Specifies the folder to which you want to save your object.

SPACE
Specifies a storage space to hold the data created by the SAVE DATA command. A blank value specifies that the default storage space is determined by the database at the current location.

SPACE DATABASE
Specifies the database name to save the table in a particular database container. The table space is created automatically under the name of the created table. This parameter is used only for z/OS databases.

ACCELERATOR
Specifies the name of the accelerator in which you want to save your table.

ACCELERATORDATABASE
Specifies the name of the database that you want to use to save accelerator-only tables. The ACCELERATORDATABASE keyword can be up to 128 characters long. The default value of the ACCELERATORDATABASE parameter is taken from the DSQEC_SAV_ACCELDB global variable. If the DSQEC_SAV_ACCELDB global variable value is not empty, the database specified by the SPACE keyword is ignored.

MODE
Specifies whether the query result set is saved with the formatting and added calculated columns. Valid values are:

RAW
Saves the query result set without the formatting and added calculated columns.

GRID
Saves the query result set with the formatting and added calculated columns.

METHOD
Specifies the method of saving the query result set. Valid values are:

REGULAR
Sends the query result set data from the client back to the database server where it is inserted into the table.

FAST
Reruns the query at the server and inserts the query results directly into the table.

FASTSAFE
Reruns the query at the server without the ORDER BY clauses and inserts the query results directly into the table.

RESULTSET
Specifies the number of the result set that you want to save.

ROWIDADD
Specifies whether to add the Row ID column to the table.

ROWIDDISP
Specifies the disposition of the new Row ID column.

ROWIDNAME
Specifies the name for the new Row ID column.

SCOPE
Specifies the commit scope of the data.

Examples

```
SAVE QUERY AS QUERY1 (CONFIRM=NO
```

```
SA Q AS rsbi:/.workspaces/MY_WORKSPACE/QUERY1
```

SAVE command

The SAVE command saves the changes in the currently open object that has already been saved to a database. If the object has not been saved to a database yet, QMF displays a prompt panel that allows you to specify the location where you want to save the object.

Syntax

```
>>--SAve--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Examples

```
SAVE
```

```
SA
```

SEARCH command

Use the SEARCH command in the Table Editor to open the **Search** panel, where you can specify the information that you want to locate in a database table.

Syntax

```
>>--SEArch--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Examples

```
SEARCH
```

```
SEA
```

SET GLOBAL command

The SET GLOBAL command sets the values of existing global variables or creates global variables and values for them. Note, that the names of variables that you create cannot begin with the DSQ prefix. This prefix identifies system global variables. You cannot add or delete system global variables, you only can edit their default values.

Syntax

```
>>--SEt Global-(+-variable_name=value-------------+--<<
                +-Lifetime=Current/Permanent------+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the SET GLOBAL command:

variable_name
Specifies the name of the global variable to which you want to assign a value.

value
Specifies the value that you want to assign to the global variable.

LIFETIME
Specifies the period of time during which the variable is available for use. Valid values are:

CURRENT
The variable is available only in the current session (default value). When the session ends, the variable will be deleted.

PERMANENT
The variable is available permanently, .

Examples

```
SET GLOBAL(EXAMPLE_VARIABLE=EXAMPLE_VALUE
```

```
SET G(DSQEC_RUN_MQ=0
```

SET INVISIBLE command

The SET INVISIBLE command hides the specified local variables from the **Prompt Variables** dialog. The **Prompt Variables** dialog will not request values for the variables that are set invisible. If all local variables are set invisible, the **Prompt Variables** dialog will not be displayed at all.

Syntax

```
>>--SEt-Invisible-(variablename1, variablename2, ...-----<<
```

Parameters

For the SET INVISIBLE command, you can specify the variablename parameter. Valid values are the names of variables that you do not want to display in the **Prompt Variables** dialog.

Example

```
SET INVISIBLE (Var1, Var2
```

SET KEY command

The SET KEY command allows you to assign a command to a function key.

Syntax

```
>>--SEt Key(+-Panelid=+-ALL-----+--<<
                       +-CURRENT-+
                       +-panelid-+
             +-Keyid=keyid-------+
             +-Label=text--------+
             +-Command=text------+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the SET KEY command:

PANELID
Specifies the panel that contains the key that you want to set. Possible values are:

- ALL - allows you to set the specified key on all the panels that use it.
- CURRENT - allows you to set the specified key on the currently open panel.
- panelid - allows you to set the specified on the panel whose ID you specify as the value for the PANELID parameter. To view the complete list of QMF panels and their IDs, see Appendix H, "IDs of QMF panels," on page 137.

KEYID
Specifies the number of the function key that you want to set. Valid values are integers from 1 to 24.

LABEL
Specifies the label text associated with the key. If the value of the LABEL parameter contains more than one word, the whole value must be enclosed in quotation marks.

COMMAND
Specifies the command that you want to assign to the key. If the value of the COMMAND parameter contains more than one word, the whole value must be enclosed in quotation marks.

Examples

```
SET KEY(PANELID=FQMPHOME, KEYID=5, LABEL=GLOBALS, COMMAND="SHOW GLOBALS"
```

```
SET KEY(P=FQMPHOME, K=5, L=GLOBALS, C="SHOW GLOBALS"
```

SET LOCAL command

The SET LOCAL command sets the values for existing local variables or creates new local variables and assigns values to them. Variables that were created via the SET LOCAL command are available only for the current object (query, report), do not appear in the Global Variables list, and do not affect other procedures.

Syntax

```
>>--SEt-Local-(variablename=value, ...--<<
```

Parameters

For the SET LOCAL command, you can specify the variablename parameter. It specifies the name of the local variable that you want to set or create. A local variable name can be from 1 to 17 characters long. Variables whose names begin with DSQ are restricted, and cannot be created or deleted.

The value of a local variable can be from 1 to 55 characters long. The values of variables whose names begin with DSQ are restricted.

Example

```
SET LOCAL (Var1=abc, Var2=def
```

SET LOCAL WITH VALUES command

The SET LOCAL WITH VALUES command creates a set of selectable values for a local variable. With this command issued, the **Prompt Variables** dialog allows you to select one of the pre-defined values or type another one by hand.

Syntax

```
>>-SEt-Local-With-Values-(variablename=value1; value2;...-<<
```

Parameters

For the SET LOCAL WITH VALUES command, you can specify the variablename parameter. It specifies the name of the local variable that you want to set or create. A local variable name can be from 1 to 17 characters long. Variables whose names begin with DSQ are restricted, and cannot be created or deleted.

The values specified for a variable will be available to choose from in the Prompt Variables window. The value can be from 1 to 55 characters long. The values of variables whose names begin with DSQ are restricted.

Example

```
SET LOCAL WITH VALUES (Var1=abc; def, Var2=ghi
```

SET OPTIONS command

The SET OPTIONS command specifies the procedure execution options.

Syntax

```
>>-SEt-Options--+------------------------+--<<
                (+-SToponerror=Yes/No-----+
                 +-SUppressmessages=value-+
```

Parameters

For the SET OPTIONS command, you can specify the following parameters:

STOPONERROR
Specifies whether the procedure stops running when an error occurs. Valid values are YES and NO.

If you do not specify any value for the STOPONERROR parameter, the value is taken from the DSQQW_PROC_FAIL_ON_ERROR global variable.

SUPPRESSMESSAGES
Specifies which type of messages to suppress while the procedure is running. Valid values are:

- ALL - suppresses all messages.
- INFORM - suppresses information messages.
- ERROR - suppresses error messages.

Example

```
SET OPTIONS (STOPONERROR=YES SUPPRESSMESSAGES=ALL
```

SHOW command

The SHOW command displays the specified panel.

Syntax

```
>>-SHow--+-Query--------+--------------------------+-<<
                        (+-View=value----------------+
                         +-Resultset=value-----------+
         +-Proc---------+
         +-Globals------+
         +-Home---------+
         +-REPort-------+
         +-Keys---------+------------------+
                        (+-PANELID=panel_id-+
         +-FORM---------+-------------+
                         +-.Main-------+
                         +-.BREAK------+
                         +-.COLumns----+
                         +-.CONditions-+
                         +-.Detail-+-----------------+
                                   (+-Variation=value-+
                         +-.Options----+
                         +-.Page-------+
                         +-.Final------+
                         +-.CAlc-------+
         +-FIeld--------+
         +-Actions------+
         +-FAvorites----+
         +-RECentlyused-+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

For the SHOW FORM.DETAIL command, you can specify the VARIATION parameter. This parameter specifies the detail variation that you want to display. If this option is omitted, the current detail variation is displayed. Valid values are integers from 1 to 99. If the specified detail variation has not been created yet, the number is reduced to the next sequential number following all existing detail variations, and a new detail variation is created. Thus, to create a new detail variation, type 99 as the value for the VARIATION parameter of the SHOW FORM.DETAIL command.

For the SHOW QUERY command, you can specify the following parameters:

VIEW
Specifies the type of the target query view. Valid values are: SQL, PROMPTED, and RESULTS.

RESULTSET
Specifies the number of the result set that you want to display.

Note: This parameter is available only if the value of the VIEW parameter is set to RESULTS.

For the SHOW KEYS command, you can specify the PANELID parameter. This parameter specifies the ID of the panel whose set of function keys you want to display. You can find the complete list of QMF panels and their IDs in the *Getting Started with QMF Z Client* guide.

The SHOW FIELD command displays detailed information about a field or a line on a panel and can be issued in the following situations:

- On the **Globals** panel, to display or edit the info about a global variable on a separate panel.
- On the **Object List** panel, to enlarge the input area of the **Action** field.
- In the Table Editor and on several other panels, to enlarge the input area of a field.
- In the Keys Editor, to display or change the function keys definitions.

Examples

```
SHOW QUERY
SHOW PROC
SHOW FORM.MAIN
SHOW FORM.DETAIL (VARIATION=2
SHOW FORM.DETAIL (VARIATION=99
```

SORT command

The SORT command sorts the items in the list of database objects. When you enter the SORT command or press the **Sort** function key, a panel is displayed that allows you to specify the sorting options.

Syntax

```
>>--SOrt--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Examples

```
SORT
```

```
SO
```

SPECIFY command

The SPECIFY command is available only on the **Prompted Query** panel and the **Form.Columns** panel. The SPECIFY command displays a panel on which you can specify the alignment options for the query result set columns or type an expression to define a calculated column (on the **Form.Columns** panel) or specify the information that is needed to compose a prompted query (on the **Prompted Query** panel).

Syntax

On the **Form.Columns** panel:

```
>>--SPecify--+------------+-<<
             +-Alignment--+
             +-Definition-+
```

On the **Prompted Query** panel:

```
>>--SPecify--+------------+-<<
             +-Columns----+
             +-Joins------+
             +-Rows-------+
             +-Sort-------+
             +-Tables-----+
```

Uppercase letters in each diagram show the minimum set of letters required to issue the command.

Parameters

If you issue the SPECIFY command without any parameters, the **Specify** panel opens. On the panel, you can select one of the following items:

Tables
Opens the **Tables** panel, where you can specify the tables that are used in the query.

Columns
Opens the **Columns** panel, where you can specify the columns that you want to include in your query result set.

Join Conditions
Opens the **Joins** panel, where you can specify the joining options for the tables in your query.

Row Conditions
Opens the **Row Conditions** panel, where you can specify row conditions.

Sort Conditions
Opens the **Sort Conditions** panel, where you can specify sort conditions.

The following parameters can be specified for the SPECIFY command:

ALIGNMENT
Opens the **Alignment** panel, where you can specify the text alignment options for the columns of your report.

DEFINITION
Opens the **Definition** panel, where you can type an expression to define the calculated column.

COLUMNS
Opens the **Columns** panel, where you can specify the columns that you want to include in your query result set.

JOINS
Opens the **Joins** panel, where you can specify the joining options for the tables in your query.

ROWS
Opens the **Row Conditions** panel, where you can specify row conditions.

SORT
Opens the **Sort Conditions** panel, where you can specify sort conditions.

TABLES
Opens the **Tables** panel, where you can specify the tables that are used in the query.

Examples

```
SPECIFY
```

```
SP
```

SWITCH command

The SWITCH command is used in the Prompted Query Editor to display the **Row Conditions** and **Sort Conditions** areas. The SWITCH COMMENT command is used on the **Object List** panel to display the **Comments** field.

Syntax

```
>>--SWitch-+---------+--<<
           +-Comment-+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The SWITCH command can be issued from the **Command** line in the **Prompted Query Editor** panel, to switch between the **Row Conditions** and **Sort Conditions** areas and the **Tables**, **Columns**, and **Join** areas.

The SWITCH COMMENT command can be issued from the **Command** line on the **Object List** panel to switch between the **Modified** and **Created** fields and the **Comments** field.

Examples

```
SWITCH
```

```
SWITCH COMMENT
```

```
SW C
```

TOP command

The TOP command scrolls towards the top of a scrollable area. TOP is equivalent to BACKWARD MAX.

Syntax

```
>>--TOp--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Examples

```
TOP
```

```
TO
```

TSO command

Use the TS0 command to enter a command in the TSO environment without terminating the QMF session.

Syntax

```
>>--TSo---commandstring--<<
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

You can specify the commandstring parameter for the TS0 command. The commandstring parameter is a character string that constitutes a valid command or exec in the TSO environment.

Every character that comes after the word TS0 is sent to TSO and interpreted there.

If the execution is successful, you return to the same QMF panel from which you entered the TSO command. If the execution is not successful, you return to the same QMF panel from which you entered the TSO command and receive an error message from TSO.

Example

To send the user JOHN5 a message with the TSO SEND command, type the following:

```
TSO SEND 'I RECEIVED YOUR PROC2. THANK YOU.' USER(JOHN5)
```

USE REPOSITORY command

The USE REPOSITORY command establishes a connection to the specified repository.

Syntax

```
>>--Use Repository---repositoryname--------------------<<
                                  (+-User=value-------+
                                   +-Password=value---+
                                   +-DBUser=value-----+
                                   +-DBPassword=value-+
```

Uppercase letters in the diagram show the minimum set of letters required to issue the command.

Parameters

The following parameters can be specified for the USE REPOSITORY command:

repositoryname
Specifies the name of the repository to which you want to connect. Make sure to enclose the name in double quotation marks.

USER
Specifies the user name that is used to connect to the secured repository.

PASSWORD
Specifies the password that is used to connect to the secured repository.

DBUSER
Specifies the user name that is used to connect to the database of the specified repository.

DBPASSWORD
Specifies the password that is used to connect to the database of the specified repository.

Examples

```
USE REPOSITORY "Default"
U R SomeRepository (USER=user PASSWORD=password
```

Appendix D. System global variables

Use system global variables to control various aspects of your QMF session, QMF commands, and panel display.

DSQQW global variables

Global variables that have names that begin with DSQQW provide information about the current query environment.

The following DSQQW global variables are currently available:

Name	Length	Description
DSQQW_AUTOMATION	1	Indicates whether the application was started as an Automation server.
DSQQW_CONNECTIONS	1	Controls the use of database server connections while running a procedure. Value can be zero (0) to minimize the number of connections or one (1) to allow a new connection for each RUN QUERY command. Specifying a value of zero (0) can force the distributed product to reset or complete a data object before continuing execution of a procedure. The default value is one (1).
DSQQW_DQ	1	The value of a double quote character. This variable can be used in queries and procedures to eliminate the need for the user to enter quotation marks with a text value. The default value is the double quote character.
DSQQW_EXP_DT_FRMT	1	The format to use when exporting data with the EXPORT DATA command in a procedure. Specify the value of: • zero (0) for text format • two (2) for HTML format • three (3) for CSV format • four (4) for IXF format • five (5) for dbase III files • six (6) for XML format • seven (7) for PDF format • eight (8) for XLS format • nine (9) for XLSX format
DSQQW_EXP_OUT_MDE	1	The IXF variation to use when exporting data to an IXF file. Value can be zero (0) for System/370 character-mode IXF or one (1) for PC/IXF. The default value is one (1).

Name	Length	Description
DSQQW_FST_SV_DATA	1	Controls the use of "fast mode" when saving data with the SAVE DATA command in a procedure. Value can be zero (0) to use regular save mode (not fast mode); one (1) to use fast mode with ORDER BY clause(s) stripped; or two (2) to use fast mode with ORDER BY clause(s). The default value is zero (0).
DSQQW_HTML_REFTEXT	55	The text that appears in a report when the &REF variable is used. The default value is "Back To".
DSQQW_ORIENTATION	0	The orientation of the application. The value is zero (0) for left-to-right orientation. The value is one (1) for right-to-left orientation.
DSQQW_PROC_FAIL_ON_ERROR	1	Stops procedure execution if any of the procedure commands fails. A value of zero (0) specifies the procedure will continue. A value of one (1) specifies the procedure will stop.
DSQQW_PROC_OUTPUT		Output file name for a procedure.
DSQQW_PROC_WNDWS	1	Controls what happens to intermediate result windows created by running a procedure. The value of zero (0) will close all intermediate windows, leaving only the final result window open at the end of the procedure. The value of one (1) will leave all windows open at the end of the procedure. The value of two (2) will close all intermediate windows, and will also close the procedure window if the procedure is run indirectly (run from another procedure or from the command line). The default value is one (1).
DSQQW_QUERY_LANG	1	Specifies the subtype of query created when a DISPLAY QUERY command is executed but no query object exists. Value can be zero (0) for a query in the SQL view or one (1) for a query in the prompted view. The default value is zero (0).
DSQQW_QUERY_PREP	1	Specifies whether the query on a RUN command is to be prepared or run. The results of prepared queries are not returned to the user's workstation. Value can be zero (0) to prepare the query, or one (1) to run the query. The default value is one (1).

Name	Length	Description
DSQQW_QUERY_PRESERVE_SORT	1	Specifies whether the query sorting order defined by a user is saved within the query and used every time the query is run. Value can be zero (0) - not to preserve the sorting order, or one (1) - to preserve the sorting order. The default value is one (1).
DSQQW_REMOTE_LAUNCH	1	Specifies whether analytical queries are executed on the server or on the client machine. Value can be zero (0) to execute analytical queries on the client machine, or one (1) to execute analytical queries on the server. Executing analytical queries on the server can improve the execution speed for large analytical queries. You can use this global variable only if you are using a web service repository connection. **Note:** If any node of your analytical query includes a prompt hierarchy, the query will be executed on the local machine.
DSQQW_REUSE_OBJS	1	Specifies whether existing windows displaying retrieved objects are reused, or if a new window opens every time an object is selected. Value can be zero (0) to always open objects in new windows, or (1) to activate an existing window if the selected object is already open. The default value is one (1).
DSQQW_RPT_COPIES	10	Specifies the number of copies to print when printing a report with the PRINT REPORT command in a procedure. The default value is one (1).
DSQQW_RPT_FONT	55	Specifies the font face name to use when printing a report with the PRINT REPORT command in a procedure. The default value is "Monospaced".
DSQQW_RPT_FONT_BD	1	Specifies the font bold attribute to use when printing a report with the PRINT REPORT command in a procedure. A value of zero (0) specifies not bold and a value of one (1) specifies bold. The default value is zero (0).
DSQQW_RPT_FONT_CS	3	The character set of the font to use when printing a report with the PRINT REPORT command in a procedure. The default value is zero (0).

Name	Length	Description
DSQQW_RPT_FONT_IT	1	Specifies the font italic attribute to use when printing a report with the PRINT REPORT command in a procedure. A value of zero (0) specifies not italic and a value of one (1) specifies italic. The default value is zero (0).
DSQQW_RPT_FONT_SZ	2	Specifies the font point size to use when printing a report with the PRINT REPORT command in a procedure. The default value is ten (10).
DSQQW_RPT_LEN_TYP	1	Specifies the type of page length that will be used when printing a report with the PRINT REPORT command or exporting a report with the EXPORT REPORT command in a procedure. Value can be zero (0) to automatically fit the length to the printed page, one (1) to specify an explicit number of lines, or two (2) to specify a continuous report with no page breaks. The default value is zero (0).
DSQQW_RPT_NUM_CHR	10	Specifies the number of characters to fit across a printed page when printing a report with the PRINT REPORT command or exporting a report with the EXPORT REPORT command in a procedure. This has an effect only when DSQQW_RPT_WID_TYP is one (1). The default value is eighty (80).
DSQQW_RPT_NUM_LNS	10	Specifies the number of lines to fit down a printed page when printing a report with the PRINT REPORT command or exporting a report with the EXPORT REPORT command in a procedure. This has an effect only when DSQQW_RPT_LEN_TYP is one (1). The default value is sixty (60).
DSQQW_RPT_ORIENT	1	The page orientation to use when printing a report with the PRINT REPORT command or exporting a report with the EXPORT REPORT command in a procedure. Value can be zero (0) for portrait or one (1) for landscape. The default value is zero (0).
DSQQW_RPT_OUT_TYP	1	The format to use when printing a report with the PRINT REPORT command in a procedure. Value can be zero (0) for text or two (2) for HTML. The default value is zero (0).

Name	Length	Description
DSQQW_RPT_TD_TYP	1	Date format for TD edit code. Value can be zero (0) for ISO format, one (1) for USA format, two (2) for EUR format or three (3) for JIS format. The default value is zero (0).
DSQQW_RPT_TT_TYP	1	Time format for TT edit code. Value can be zero (0) for ISO format, one (1) for USA format, two (2) for EUR format or three (3) for JIS format. The default value is zero (0).
DSQQW_RPT_USE_PS	1	Specifies what page formatting options (page length, page width, etc.) to use when printing a report with the PRINT REPORT command in a procedure. Value can be zero (0) to use the values specified on the PRINT REPORT command or in global variables, or one (1) to use the values specified in the form's page setup. The default value is one (1).
DSQQW_RPT_WID_TYP	1	Specifies the type of page width when printing a report with the PRINT REPORT command in a procedure. Value can be zero (0) to automatically fit the width to the printed page, one (1) to specify an explicit number of characters or two (2) to specify a continuous line. The default value is zero (0).
DSQQW_SHOW_QUERY	1	Specifies which view of a query to display when the SHOW QUERY command is issued from a procedure. Valid values are zero (0) for the **SQL** or **Prompted** view and one (1) for the **Results** view. The default value is zero (0).
DSQQW_SQ	1	The value of a single quotation mark. This variable can be used in queries and procedures to eliminate the need for the user to enter quotation marks with a text value. The default value is a single quotation mark (').
DSQQW_SV_DATA_C_S	10	The number of rows to insert before committing the unit of work when saving data with a SAVE DATA command in a procedure. Value can be zero (0) for all of the rows or an explicit number of rows. The default value is zero (0).

Name	Length	Description
DSQQW_SV_DATA_T_M	1	Specifies how source and target column data types are matched when using the SAVE DATA command. The value can be zero (0) to require exact data type matches, one (1) to allow data type conversions with no possible data loss, or two (2) to allow all data type conversions that are supported by the database. The default value is one (1).
DSQQW_UEDIT_JAR	55	The name of the JAVA archive file that contains user edit routines.

DSQAO global variables

Global variables that have names that begin with DSQAO provide information about the current state of the query session.

The following DSQAO global variables are available:

Name	Length	Description
DSQAO_BATCH	1	Batch or interactive mode. Value can be one (1) for an interactive session or two (2) for a batch session. See the BATCH command line parameter.
DSQAO_CONNECT_ID	8	The user ID that is used to connect to the current database.
DSQAO_CONNECT_LOC	18	If you are connected to a DB2 for z/OS database, this variable specifies the location name of the DB2 for z/OS database to which you are currently connected. For all other scenarios, this variable is not set with any value.
DSQAO_CURSOR_OPEN	1	The status of the current query object's database cursor. Value can be one (1) if the cursor is open or two (2) if the cursor is closed.
DSQAO_DATE_FORMAT	3	If you are connected to a DB2 for z/OS database, this variable contains the value that is specified in SYSIBM.DATE_FORMAT for z/OS DB2. For all other scenarios, this variable is not set with any value. Values can be ISO, USA, EUR, or JIS.
DSQAO_DBCS	1	DBCS support status. Value can be one (1) if DBCS support is present or two (2) if DBCS support is not present.

Name	Length	Description
DSQAO_DB_MANAGER	1	Database manager, indicated by one of the following values: **0** Unknown **1** DB2 for VM **2** DB2 for z/OS **3** DB2 for Windows **4** DB2 for iSeries **5** DB2 for Linux **6** DB2 for AIX **7** DB2 for Solaris **8** DB2 for HP-UX system
DSQAO_HOME_WORKSPACE	128	The current repository user's home workspace key, if the workspace exists. Valid values are: • `rsbi:/.workspaces/<user name>` This is the value if the user connected to a secured repository connection and if the `rsbi:/.workspaces/<user name>` object is viewable by the current user in the repository. *<user name>* is the login name of the repository user. • `rsbi:/.workspaces` This is the value if the user connected to a repository connection without security, or the `rsbi:/.workspaces/<user name>` is not viewable by the user, or it does not exist. • blank This is value if the previous situations do not exist. For example, if the user is not connected to any repository connection. **Note:** Workspace operations such as creating, deleting, and renaming performed by the current user affect the Global Variable value. Additionally, such operations performed by other users might also affect the value.

Name	Length	Description
DSQAO_LOCAL_DB2	18	If you are connected to a DB2 for z/OS database, this variable specifies the location name of the DB2 for z/OS database to which you are currently connected. For all other scenarios, this variable is not set with any value.
DSQAO_LOCATION	18	Location name of the current object. This variable provides the name of the current datasource.
DSQAO_NLF_LANG	1	National language of session. Value is "E" for the English language.
DSQAO_NUM_FETCHED	0	The number of rows fetched by the current query object.
DSQAO_OBJ_NAME	18	The name of the current query, form, or procedure object. If there is no current object, the value is blank.
DSQAO_OBJ_OWNER	8	The owner of the current query, form, or procedure object. If there is no current object, the value is blank.
DSQAO_PANEL_TYPE	1	Type of current panel, indicated by one of the following values: **1** HOME (Default value) **2** QUERY **3** REPORT **4** FORM **5** PROC **8** LIST **9** Table Editor **A** GLOBALS
DSQAO_REP_USER	8	The user name that is used to connect to the current repository.
DSQAO_QMF_RELEASE	2	Numeric release number of the application.
DSQAO_QMF_VER_RLS	10	External version and release number for the application.
DSQAO_QUERY_MODEL	1	Model of the current query object. Value can be one (1) for relational.
DSQAO_QRY_SUBTYPE	1	Subtype of the current query object. Value can be one (1) for SQL queries or three (3) for queries in a prompted view.

Name	Length	Description
DSQDC_SHOW_PANID	1	**1** Display Panel Identifiers (Default value) **0** Suppress Panel Identifiers
DSQAO_SUBSYS_ID	4	If you are connected to a DB2 for z/OS database, this variable specifies the ID of the local DB2 subsystem to which QMF is attached. For all other scenarios, this variable is not set with any value.
DSQAO_SYSTEM_ID	1	Current operating system. Values can be one of the following: • 8 - Windows NT and above • 9 - Linux® • 10 - HP-UX • 11 - AIX® • 12 - Solaris • 13 - iSeries • 14 - z/OS
DSQAO_TIME_FORMAT	3	If you are connected to a DB2 for z/OS database, this variable contains the value that is specified in SYSIBM.TIME_FORMAT for z/OS DB2. For all other scenarios, this variable is not set with any value. Values can be ISO, USA, EUR, or JIS.

DSQEC global variables

Global variables that have names that begin with DSQEC control how commands and procedures are executed.

The following DSQEC global variables are available:

Name	Length	Description
DSQEC_CON_ACC_RES	1	For executable SELECT queries that the application submits to Db2 for z/OS, this variable allows you to specify how you want the database to proceed when the data to be selected is locked by an insert, update, or delete operation. When you set this variable, the application specifies the clause associated with the variable value on the concurrent-access-resolution attribute of the PREPARE statement for the SELECT query. Executable SELECT queries can result not only from queries (such as SQL SELECT queries, prompted queries, or QBE P. queries), but also from other operations such as DISPLAY TABLE. Possible values are: • **0** - No concurrent access resolution options on the PREPARE statement associated with the pending SQL SELECT statement are specified. This is the default value. • **1** - SKIP LOCKED DATA. This value can be specified for executable SELECT statements directed to DB2® for z/OS Version 9 and Version 10 and to Db2 for z/OS Version 11 or later. • **2** - USE CURRENTLY COMMITTED. This value can be specified for executable SELECT statements directed to DB2 for z/OS Version 10 and to Db2 for z/OS Version 11 or later. • **3** - WAIT FOR OUTCOME. This value can be specified for executable SELECT statements directed to DB2 for z/OS Version 10 and to Db2 for z/OS Version 11 or later.
DSQEC_CURR_FOLDER	128	Use the variable to group links to objects in folders in QMF Catalog. Its value is used as a default one for the FOLDER parameter in the LIST, SAVE, and ERASE commands for Db2 databases. It can be up to 128 characters long. The variable is blank by default, no folder is used for LIST, SAVE, and ERASE commands.
DSQEC_DSALLOC_DIR	3	Specifies the number of directory blocks to be used when exporting a member of a new PDS data set in TSO. The value must be greater than zero for PDS data sets. If you are using the site default type of data set or PDSE data sets, QMF ignores the value of this global variable. To use the site default type of data set, set DSQEC_PO to 0. To use PDSE data sets, set DSQEC_PO to 2. If your site uses sequential data sets, set this global variable to zero.
DSQEC_DSALLOC_PRI	8	Specifies the primary quantity of tracks for the TSO data set that is used to store the results of the QMF EXPORT command. Values can be from 1 to the maximum size allowed by the storage device and operating system. The default value is 15. A value of zero is not allowed. **Note:** PS, PDS, and PDSE data sets can have a maximum value of 16777215 tracks.

Name	Length	Description
DSQEC_DSALLOC_SEC	8	Specifies the secondary quantity of tracks for the TSO data set that is used to store the results of the QMF EXPORT command. Values can be from zero to the maximum size allowed by the storage device and operating system. The default value is 105 tracks. **Note:** PS, PDS, and PDSE data sets can have a maximum value of 16777215 tracks.
DSQEC_DSQSFISO	1	For DB2 z/OS databases, specifies the format in which date time columns are displayed. The following values are used: **0** The date time columns are displayed in DATE FORMAT and TIME FORMAT fields on DB2 installation panel DSNTIP4. **1** The date time columns are displayed in ISO format. For non-DB2 z/OS databases, the value of this variable has no effect. The date and time columns will always be displayed in ISO format.
DSQEC_EXTND_STG	31	Specifies the number of megabytes of extended storage that the application will acquire on each request to the extended storage manager when spilling data to extended storage in QMF for TSO. When a user performs an operation that requires extended storage, the application issues repeated requests to the extended storage manager for the specified amount until the operation is complete or extended storage is exhausted. When setting this global variable, consider the average size of DATA objects with which your users work. If the average size is very large and you set the DSQEC_EXTND_STG variable too low, the application must issue many calls to the extended storage manager to complete the DATA object, which could affect overall performance. Values can be from 1 to 1000. The default value is 25, indicating that the application requests 25MB of storage on each request.
DSQEC_FORM_LANG	1	Defines the default NLF language in which a form will be saved or exported. Value can be zero (0) for the presiding NLF language or one (1) for English. The default value is one (1).
DSQEC_LAST_RUN	1	Specifies the set of commands that cause the LAST_USED column of the Q.OBJECT_DIRECTORY table to be updated. Possible values are: • **0** - Last used is updated on any activity. • **1** - Last used is updated when RUN, SAVE, or IMPORT commands are performed.

Name	Length	Description

DSQEC_LIST_OWNER	128	Provides the default value for the OWNER parameter of the LIST command. Specify an authorization ID up to 128 characters long. This variable is blank by default, resulting in a list of objects owned by the current authorization ID. You can use selection symbols in the variable value. Use an underscore (_) in place of a single character and a percent sign (%) in place of zero or more characters. For example, after you issue the following command, followed by a LIST command, the application lists only objects that are owned by user IDs that begin with the characters RO:
		`SET GLOBAL (DSQEC_LIST_OWNER=RO%`
		The following command sets the default owner to any user IDs that begin with I, have any character in the second position, and any characters in the remaining positions:
		`SET GLOBAL (DSQEC_LIST_OWNER=I_%`

DSQEC_LOB_COLMAX	10	Specifies the maximum data size of a LOB column that is to be retrieved, in bytes, up to the maximum LOB size of 2147483648 KB (2 GB). By default, LOB metadata is retrieved instead of LOB data. However, if an edit code other than M is specified or if the DSQEC_LOB_RETRV global variable is set to 3, LOB data is retrieved instead of metadata. In this case, if a user queries a table that contains LOB data that is larger than the maximum size specified for the variable, an error message is displayed. If a user issues an EXPORT TABLE, PRINT TABLE, SAVE DATA, or EXPORT DATA command for a table or data object that contains LOB data that is larger than the maximum, an error is displayed. The default value is 0, which specifies the maximum value. **Note:** This is a read-only variable.

Name	Length	Description
DSQEC_LOB_RETRV	1	Specifies how LOB data or metadata is retrieved. The valid values are: **0** When this option is displayed, you will not be able to query any table that contains LOB data and an error message is displayed. **1** Displays LOB metadata in results. To display actual LOB data, you can change the M edit code to another edit code. When this value is specified, QMF uses LOB locators to access LOB data, where a LOB Locator is a reference to a LOB value stored in a database. This is the default setting. **2** Displays LOB metadata only in results. The M edit code is the only valid edit code for LOB data. When this value is specified, QMF does not use LOB locators. **3** Retrieves and displays actual LOB data in results. When this value is specified, QMF does not use LOB locators to access LOB data. **Note:** This is a read-only variable.
DSQEC_LOB_SAVE	1	Specifies whether users can save LOB data to a table in the database using the QMF SAVE DATA or IMPORT TABLE command. The valid values are: **0 - Disable LOB Save** Specifies that users cannot issue the QMF SAVE DATA or IMPORT TABLE commands to save data to a table in the database if any column contains LOB data. An error message is displayed and no data is saved if a LOB column exists. **1 - Enable LOB Save** Specifies that users can save LOB data to a table in the database using the QMF SAVE DATA or IMPORT TABLE commands. This is the default value. **Note:** This is a read-only variable.
DSQEC_NLFCMD_LANG	1	Defines the expected NLF language for commands in procedures. Value can be zero (0) for the presiding NLF language or one (1) for English. The default value is zero (0).
DSQEC_PO	1	Specifies the type of partitioned (PO) data set to create when exporting a QMF object to a new TSO data set. Values can be: **0** Allocates a data set of the type listed as the default for your site. This type is specified in the IGDSMSxx member of the SYS1.PARMLIB. This value is the default value. **1** Allocates a PDS data set for the exported data. **2** Allocates a PDSE data set for the exported data.

Name	Length	Description
DSQEC_RESET_RPT	1	Determines whether a user will be prompted when an incomplete data object that will affect performance is encountered. Value can be zero (0) to complete the data object without prompting, one (1) to prompt the user asking whether the data object should be completed, or two (2) to reset the data object without prompting.

DSQEC_RUN_MQ	1	Specifies whether the RUN QUERY command supports multiple statements in an SQL query. Possible values are:
		• **0** - Multiple SQL statements are not supported.
		If you set this variable to 0 and run an SQL query that contains multiple statements, the application ignores all statements after the first semicolon.
		• **1** - Multiple SQL statements are supported.
		This is the default value.
		A semicolon can be placed at the end of each statement.
		You can substitute the semicolon with any other character by using `SET STATEMENT DELIMITER` comment at the beginning of the SQL text. For example, the following example is a valid use of SQL with multiple statements:
		`--SET STATEMENT DELIMITER="!"`
		`select * from q.staff!`
		`select * from q.org`

DSQEC_SAV_ACCELDB	128	Specifies the name of the database that you want to use to save accelerator-only tables.
DSQEC_SAV_ACCELNM	128	Contains the default name of the accelerator that you want to use when creating accelerator-only tables from SAVE DATA, IMPORT TABLE and RUN QUERY (with TABLE keyword) commands. This variable is only referenced if query acceleration is enabled and the ACCELERATOR keyword is not specified. You can leave this global variable blank, if the value of the DSQEC_SAV_ALLOWED global variable is not set to 2, 4, or 5.

Name	Length	Description
DSQEC_SAV_ALLOWED	1	This field specifies whether users can save data to a database table or to an accelerator table by using the SAVE DATA, IMPORT TABLE, and RUN QUERY (with the TABLE keyword) commands. Possible values of this global variable are: • **0** - Specifies that users cannot save data at all. • **1** - Specifies that users can save data only to database tables. This value is selected by default. • **2** - Specifies that users can save data only to accelerator tables. If this option is selected, the DSQEC_SAV_ACCELNM global variable must contain the name of the accelerator that you want to use by default. The DSQEC_SAV_ACCELNM global variable can be overridden with the ACCELERATOR keyword. Accelerator-only tables cannot be copied to more than one accelerator. • **3** - Specifies that users can save data to either a database table or an accelerator table. The data is saved to the database table if no command keyword overrides, such as SPACE or ACCEL, are present. • **4** - Specifies that users can save data to either a database table or an accelerator table. If no command keyword overrides, such as SPACE or ACCELERATOR, are present, the data is saved to the accelerator. When this option is chosen, DSQEC_SAV_ACCELNM global variable must contain the name of the accelerator that you want to use by default. • **5** - Specifies that users can save data to accelerator-shadow tables. These tables are saved in the database but also support accelerated read data queries and, therefore, can be saved to an accelerator as well. When this option is chosen, DSQEC_SAV_ACCELNM global variable must contain the name of the accelerator that you want to use. Accelerator-shadow tables can be copied to multiple accelerators.
DSQEC_SHARE	1	Specifies the default value for whether a saved object will be shared with other users. Value can be zero (0) to not share the object or one (1) to share the object.
DSQEC_SP_RS_NUM	1	Specifies the number of the result set that will be displayed for a stored procedure. The default result set number is minus one (-1).

Name	Length	Description
DSQEC_SQLQRYSZ_2M	1	Controls whether SQL queries greater than 32,767 bytes (32 KB) in length are supported by the RUN QUERY command. Possible values are: • **0** - SQL queries directed to DB2 for iSeries or Db2 for z/OS, Linux, UNIX, and Windows databases are limited to 32,767 bytes (32 KB). • **1** - SQL queries can be greater than 32 KB. The maximum supported size of queries directed to DB2 for iSeries or Db2 for Linux, UNIX, and Windows can be up to 65 KB in length. The maximum supported query size varies depending on the type of database to which the query is directed: – Queries directed to Db2 for z/OS can be up to 2 MB in length. – Queries directed to DB2 for iSeries or Db2 for Linux, UNIX, and Windows can be up to 65 KB in length. These maximums assume that the version of the database to which the RUN QUERY command is directed supports queries of this size. SQL queries directed to DB2 for VM and VSE are limited to 8 KB. The default value is 1.
DSQEC_USERGLV_SAV	1	Determines whether the global variables that were created or edited by the user during the current QMF session are saved when the session ends. Saved variables and values will be restored at the start of the next QMF session. Valid values are: • **0** - All system global variables are restored to their default state at the start of the next session. All user-defined global variables are discarded. • **1** - All global variables that were created by the user during the current session are discarded when the session ends. All global variables that were edited by the user during the current session are restored to their previous state. • **2** - All global variables that were created or edited by the user during the current session are saved when the session ends. Note that the global variables whose `LIFETIME` parameter was set to `CURRENT` are still discarded. This is the default value.

DSQDC global variables

Global variables that have names that begin with DSQDC control how information is displayed.

The following DSQDC global variables are available:

Name	Length	Description
DSQDC_COL_LABELS	1	Specifies whether column headings will be column names or database labels in Classic Reports. Value can be zero (0) to specify that column headings will be column names or one (1) to specify that column headings will be database labels. The default value is one (1).

Name	Length	Description
DSQDC_CURRENCY	18	Defines the custom currency symbol to use when the DC edit code is specified.
DSQDC_DISPLAY_RPT	1	Specifies whether a report is displayed after a RUN QUERY command in a procedure. Value can be zero (0) to not display a report or one (1) to automatically display a report with the default form. The default value is zero (0).
DSQDC_LIST_ORDER	2	Specifies the default sort order for objects in a list of database objects. Valid values for this variable are combinations of two characters that are typed together, without a blank space between them. Valid values for the first character are: **1** The list is sorted in the default order. **2** The list is sorted by object owner. **3** The list is sorted by object name. **4** The list is sorted by object type. **5** The list is sorted by the date modified. **6** The list is sorted by the date last used. Valid values for the second character are: **A** The list is sorted in the ascending order. **D** The list is sorted in the descending order. This variable applies only to the objects that are listed as a result of the LIST command. This variable does not apply to the list that were produced via other means.
DSQDC_POS_SQLCODE	1	Specifies what happens when a positive SQL code is returned from the database. Possible values are: • **0** - Neither log the message or display the message text. • **1** - Log the message associated with the SQL code. • **2** - Display the online help that is associated with the SQL code.

DSQCP global variables

Global variables that have names that begin with DSQCP control the operation of the table editor.

The following DSQCP global variables are available:

Name	Length	Description
DSQCP_CNFRM_DBUPD	1	Indicates if the **Confirm database updates** check box in the **Options** tab of the **Edit Resource Limits** dialog box is selected or not. Values include: **0** The **Confirm database updates** check box is *not*selected. **1** The **Confirm database updates** check box is selected. **Note:** This is a read-only variable. If this variable is set 1, the confirmation panel is displayed irrespective of the values set in DSQCP_TEADD, DSQCP_TECHG, and DSQCP_TEDEL.
DSQCP_TEADD	1	Determines if a confirmation panel is displayed when adding rows to a table during a table edit session. Values can be: **0** Confirmation panel is disabled. **1** Confirmation panel is enabled. Confirmation panel is enabled by default. **Note:** The value set in this variable is ignored if DSQCP_CNFRM_DBUPD is 1.
DSQCP_TECHG	1	Determines if a confirmation panel is displayed when modifying rows in a table during a table edit session. Values can be: **0** Confirmation panel is disabled. **1** Confirmation panel is enabled. Confirmation panel is enabled by default. **Note:** The value set in this variable is ignored if DSQCP_CNFRM_DBUPD is 1.
DSQCP_TEDEL	1	Determines if a confirmation panel is displayed when deleting rows in a table during a table edit session. Values can be: **0** Confirmation panel is disabled. **1** Confirmation panel is enabled. Confirmation panel is enabled by default. **Note:** The value set in this variable is ignored if DSQCP_CNFRM_DBUPD is 1.
DSQCP_TEDFLT	1	Defines the reserved character used in the Table Editor to indicate a default value for a column. The default value is "+".

Name	Length	Description
DSQCP_TENULL	1	Defines the reserved character used in the Table Editor to indicate a null value for a column. The default value is "-".

Appendix E. SQL editor line commands

Use line commands to insert, remove, copy, and reposition the lines in the SQL editor area. The line command area is located to the left of the editor area.

INSERT

The INSERT line command inserts one or more blank lines. Use the following syntax with the INSERT line command:

I
Inserts one blank line.

I<n>
Inserts <n> blank lines after the line that is marked with the I character.

DELETE

The DELETE line command removes one or more lines. Use the following syntax with the DELETE line command:

D
Removes one line.

D<n>
Removes <n> lines, starting from the line that is marked with the D character.

DD ... DD
Removes all lines between the two DD commands, including the lines on which the DD commands are entered.

COPY

The COPY line command copies one or more lines and pastes them before or after a specified line. Use the following syntax with the COPY line command:

C A/B
Copies the line that is marked with the C character and pastes it after the line that is marked with the A character or before the line that is marked with the B character.

C<n> A/B
Copies <n> lines, starting from the one that is marked with the C character, and pastes them after the line that is marked with the A character or before the line that is marked with the B character.

CC ... CC A/B
Copies all lines between the two CC commands, including the lines on which the CC commands are entered, and pastes them after the line that is marked with the A character or before the line that is marked with the B character.

REPEAT

The REPEAT line command duplicates one or more lines. Use the following syntax with the REPEAT line command:

R
Duplicates one line.

R<n>
Inserts <n> copies of the line that is marked with the R character.

RR ... RR
Duplicates the lines on which the RR commands are entered and all lines between them.

RR ... RR<n>
Inserts <n> copies of the lines that are enclosed in the RR commands.

MOVE

The MOVE line command repositions one or more lines. Use the following syntax with the MOVE line command:

M A/B
Places the line that is marked with the M character after the line that is marked with the A character or before the line that is marked with the B character.

M<n> A/B
Places <n> lines, starting from the line that is marked with the M character, after the line that is marked with the A character or before the line that is marked with the B character.

MM ... MM A/B
Places all lines between the two MM commands, including the lines on which the MM commands are entered, after the line that is marked with the A character or before the line that is marked with the B character.

Appendix F. QMF usage codes

When you create a report, you specify a usage code for each column. The usage code specifies the operation to perform on the data in the column.

ACROSS
Produces a report with horizontal control breaks. Note that:

- The number and titles of columns in your report are dependent on the values in the ACROSS column. There is only one set of report columns for each value in the ACROSS column and the header for each is the value of the column. The set of report columns includes a column for each one that uses an aggregation usage code, such as SUM, AVERAGE, COUNT.
- You can have only one ACROSS column in a report.
- The CSUM, PCT, CPCT, TPCT, and TCPCT usage codes are only partially supported when generating reports that also use the ACROSS usage code.

AVERAGE
Analyzes all values in a column and calculates the average. The calculated value appears as a total in the report. The calculated value is formatted with the edit code of the column. This usage code is only valid for numeric data.

BREAK*n*
Provides a control break level. The "n" symbol represents a number between 1 and 6. For example, the BREAK1 usage code specifies a control column for a level-1 break and BREAK2 specifies a control column for a level-2 break. Any change in the value of the column causes a section break in the report. Subtotals are displayed for columns whose usage code is one of the aggregation type. Also, the break text specified on the **Form.Break** panel is displayed.

BREAK*n*X
Same as BREAKn, except the control column is omitted from the report.

CALC*id*
Provides the evaluation of calculation expressions on the Form.Calculations panel. The "id" part represents the ID of the calculation expression.

COUNT
Counts the non-null values in the column. The calculated value appears as a total in the report and is formatted with the edit code K.

CPCT
Calculates the cumulative percentage of each value in the column relative to the current total.

CSUM
Calculates the cumulative sum of the values in the column. The calculated value replaces each detail line value and also appears as a total in the report. The calculated value is formatted with the edit code of the column. The CSUM usage code is only partially supported when generating reports that also use the ACROSS usage code.

FIRST
First value of the column. The calculated value appears as a total in the report. The calculated value is formatted with the edit code of the column.

GROUP
Displays only one line of summary data for each set of values in the column. More than one column can have usage code GROUP. If so, a change in value in any column starts a new group. All other columns with no usage code are omitted from the report.

LAST
Last value in the column. The calculated value appears as a total in the report. The calculated value is formatted with the edit code of the column.

MAXIMUM
Maximum value in the column. The calculated value appears as a total in the report. The calculated value is formatted with the edit code of the column.

MINIMUM
Minimum value in the column. The calculated value appears as a total in the report. The calculated value is formatted with the edit code of the column.

OMIT
Excludes the column from the report.

PCT
Calculates the percentage of each value in the column relative to the current total. The calculated value replaces each detail line value and also appears as a total in the report. The calculated value is formatted with the edit code of the column. The PCT usage code is only partially supported when generating reports that also use the ACROSS usage code.

STDEV
Calculates the standard deviation of the values in the column. This usage code is only valid for numeric data. The calculated value appears as a total in the report. The calculated value is formatted with the edit code of the column.

SUM
Calculates the sum of the values in the column. This usage code is only valid for numeric data. The calculated value appears as a total in the report. The calculated value is formatted with the edit code of the column.

TPCT
Calculates the percentage of each value in the column relative to the final total. The calculated value replaces each detail line value and also appears as a total in the report. The calculated value is formatted with the edit code of the column. The TPCT usage code is only partially supported when generating reports that also use the ACROSS usage code.

TCPCT
Calculates the cumulative percentage of each value in the column relative to the final total. The calculated value replaces each detail line value and also appears as a total in the report. The calculated value is formatted with the edit code of the column. The TCPCT usage code is only partially supported when generating reports that also use the ACROSS usage code.

Appendix G. QMF edit codes

An edit code is a set of characters that tells QMF how to format and punctuate the data in a specific column of a report. Edit codes do not change the data in the database; they merely control how the data is displayed. Below is the complete list of QMF edit codes.

Edit codes for character data

Use the character data edit codes to format the text fields in your report.

C
Display character data.

CW
Display character data with wrapping based on column width. If the value does not fit on one line in the column, as much data as possible is placed within the column, and then additional data is wrapped to subsequent lines in the column.

CT
Display character data with wrapping based on the text in the column. If the value does not fit on one line in the column, as much data as possible is placed on a line within the column, until a blank is found within the text that triggers the wrapping of additional data to subsequent lines in the column. If the string of text is too long to fit in the column and does not contain a blank, the data is wrapped by width until a blank is found.

CDx
Display character data with wrapping based on the specified delimiter. If the value does not fit on one line in the column, a new line of data begins in the column each time a special delimiter is encountered in the text. If the string of text is too long to fit in the column and does not contain a delimiter, the data is wrapped by width until a delimiter is found. The delimiter specified by "x" can be any single character, including a blank. The delimiter character does not appear in the report.

X
Format data as a series of hexadecimal characters.

XW
Format data as a series of hexadecimal characters with wrapping based on column width. Columns are wrapped according to the rules specified for the CW edit code.

B
Format data as a series of zeros and ones.

BW
Formats data as a series of zeros and ones with wrapping based on column width. Columns are wrapped according to the rules specified for the CW code.

Edit codes for date data

Use the date data edit codes to format the fields that contain date information. The letter "x" in the date edit codes represents the character to be used as the delimiter in the date value. The value of "x" can be any special character, including a blank, but excluding a letter or a number.

TDYx
Year (4 digits), month, day.

TDMx
Month, day, year (4 digits).

TDDx
Day, month, year (4 digits).

TDYAx
Year (last 2 digits), month, day.

TDMAx
Month, day, year (last 2 digits).

TDDAx
Day, month, year (last 2 digits).

TDL
Formats date according to the format specified as the default at the database server requesting data.

TD
Edit codes that appear on result set reports. That is, reports generated from a stored procedure CALL. They would be used on time or date data if the data is not in the ISO format. If these edit codes are found on column data, then the edit code cannot be changed for that column. Also, the report object cannot be exported if this edit code is present in the form.

Edit codes for graphic data

Use the graphic data edit codes to format the fields that contain graphic or pure DBCS information.

G
Display graphic data.

GW
Display graphic data with wrapping based on column width. If the value does not fit on one line in the column, as much data as possible is placed within the column, and then additional data is wrapped to subsequent lines in the column.

Edit codes for numeric data

Use the numeric data edit codes to format the fields that contain numeric information. The letters "nn" in the numeric data edit codes represents a number between 0 and 99. This number determines how many places to allow after the decimal point. Numbers with more places after the decimal are rounded and numbers with fewer places are padded.

E
Displays numbers in scientific notation. Up to 17 significant digits, or up to 34 significant digits when editing extended floating point data, are shown even if the width of the column can accommodate more. Used as the default form for columns with data type FLOAT.

EZ
Displays numbers in scientific notation with zero values in the column suppressed. Up to 17 significant digits, or up to 34 significant digits when editing extended floating point data, are shown even if the width of the column can accommodate more.

Dnn
Displays numbers in decimal notation formatted with a negative sign, thousands separator, and currency symbol.

DZnn
Displays numbers in decimal notation formatted with a negative sign, thousands separator, currency symbol and any zero values in the column suppressed.

DCnn
Displays numbers in decimal notation formatted with a negative sign, thousands separator, and a user-defined currency symbol. The currency symbol that will be used instead of the standard currency symbol is defined using the global variable DSQDC_CURRENCY.

DZCnn
Displays numbers in decimal notation formatted with a negative sign, thousands separator, a user-defined currency symbol and any zero values in the column suppressed. The currency symbol that will be used instead of the standard currency symbol is defined using the global variable DSQDC_CURRENCY. If both edit code options "Z" and "C" are used, "C" must follow "Z"

Inn
Displays numbers in decimal notation formatted with leading zeros displayed and a negative sign.

IZnn
Displays numbers in decimal notation formatted with leading zeros displayed, a negative sign and any zero values in the column suppressed.

Jnn
Displays numbers in decimal notation formatted with leading zeros displayed.

JZnn
Displays numbers in decimal notation formatted with leading zeros displayed and any zero values in the column suppressed.

Knn
Displays numbers in decimal notation formatted with a negative sign and a thousands separator.

KZnn
Displays numbers in decimal notation formatted with a negative sign, a thousands separator and any zero values in the column suppressed.

Lnn
Displays numbers in decimal notation formatted with a negative sign.

LZnn
Displays numbers in decimal notation formatted with a negative sign and any zero values in the column suppressed.

Pnn
Displays numbers in decimal notation formatted with a negative sign, thousands separator, and a percent sign.

PZnn
Displays numbers in decimal notation formatted with a negative sign, thousands separator, a percent sign and any zero values in the column suppressed.

Edit codes for time data

Use the time data edit codes to format the fields that contain time information. The letter "x" in the time data edit code represents the character that will be used as the delimiter in the time value. The value of "x" can be any character, including a space, but excluding a letter or a number.

TTSx
24 hour clock, seconds included.

TTCx
12 hour clock, seconds included.

TTAx
24 hour clock, seconds excluded.

TTAN
24 hour clock, seconds excluded, and no delimiter between hours and minutes.

TTUx
USA format (`HHxMM PM`, `HHxMM AM`).

TTL
Formats time data according to the format specified as the local default at the database server requesting data.

TT
Edit codes that appear on result set reports. That is, reports generated from a stored procedure CALL. They would be used on time or date data if the data was found not to be in ISO format. If these edit codes are found on column data, then the edit code cannot be changed for that column. Also, the report object cannot be exported if this edit code is present in the form.

Edit codes for timestamp data

Use the timestamp data edit codes to format the fields that contain timestamp information.

TSI
yyyy–mm–dd–hh.mm.ss.nnnnnnnnnnnn, where yyyy is the four digit year, mm is the two digit month, dd is the two digit day, hh is the two digit hour, mm is the two digit minute, ss is the two digit second, and nnnnnnnnnnnn is the twelve digit fractional seconds.

TSZ
yyyy–mm–dd–hh.mm.ss.nnnnnnnnnnnn±th:tm, where yyyy is the four digit year, mm is the two digit month, dd is the two digit day, hh is the two digit hour, mm is the two digit minute, ss is the two digit second, nnnnnnnnnnnn is the twelve digit fractional seconds, ±th is the two-digit value representing the time zone hour, shown as an offset relative to UTC, and tm is the two-digit value representing the time zone minutes between 0 and 59.

Note: To specify UTC, you can either specify a time zone of -24:00 or +24:00 or replace the time zone offset and its sign with an uppercase Z.

User defined edit codes

You can use the user defined edit codes Uxxxx and Vxxxx for special purposes. The value "xxxx" can be any combination of characters, excluding embedded blanks. The following user edit codes are predefined:

VSSN or USSN
Social security number format (xxx–xx–xxxx).

VTEL
Telephone number format ((xxx) xxx–xxxx).

VTEL2
Telephone number format (xxx.xxx.xxxx).

VZIP
Zip code format (xxxxx–xxxx).

Edit codes for metadata

Use the metadata edit code M to display the descriptive data for a report column rather than the actual data. The metadata for a column is found in the Descriptor Area (DA) and consists of the type and the length of the data that will be included in the column. If a column with the edit code M is null, a null indicator is displayed rather than the metadata. If the column size is less than the amount needed to display the metadata, the metadata is truncated in order to fit into the column space.

Edit code for LOB data

Use the LOB edit code to format the LOB fields in your report.

Supported Edit Codes:

- **For BLOB**: B, X
- **For CLOB**: C, B, X

where:

- **B**: Format LOB data as a series of zeros and ones.
- **C**: Display character LOB data.
- **X**: Format LOB data as a series of hexadecimal characters.

Appendix H. IDs of QMF panels

Full-screen panels

Form.Main panel
Panel ID is FQMPFMAN.

Form.Break panel
Panel ID is FQMPFBRK.

Form.Calculations panel
Panel ID is FQMPFCLC.

Form.Columns panel
Panel ID is FQMPFCOL.

Form.Conditions panel
Panel ID is FQMPFCON.

Form.Detail panel
Panel ID is FQMPFDET.

Form.Final panel
Panel ID is FQMPFFIN.

Form.Options panel
Panel ID is FQMPFOPT.

Form.Page panel
Panel ID is FQMPFPAG.

Globals panel
Panel ID is FQMPGLOB.

Home panel
Panel ID is FQMPHOME.

Keys panel
Panel ID is FQMPKEYS.

Object List panel
Panel ID is FQMPOBJL.

Procedure Editor panel
Panel ID is FQMPPEDT.

Query Editor panel
Panel ID is FQMPQEDT.

Prompted Query Editor panel
Panel ID is FQMPPQRY.

Results panel
Panel ID is FQMPRSLT.

Report panel
Panel ID is FQMPRPRT.

Table Editor panel
Panel ID is FQMPTBED.

Non-full-screen panels

About panel
Panel ID is FQMPABOT.

Action panel
Panel ID is FQMPACTE.

Action on <object_name> panel
Panel ID is FQMPOACT.

Add Global Variable panel
Panel ID is FQMPGLAD.

Add Row panel
Panel ID is FQMPTEAD.

Alignment panel
Panel ID is FQMPFCAL.

Attention Interrupt panel
Panel ID is FQMPATTN.

Batch List panel
Panel ID is FQMPBTLT.

Batch Wizard - Common Parameters panel
Panel ID is FQMPBTD4.

Batch Wizard - Main Parameters panel
Panel ID is FQMPBTD1.

Batch Wizard - Parameters for PROC panel
Panel ID is FQMPBTDP.

Batch Wizard - Parameters for QUERY panel
Panel ID is FQMPBTDQ.

Batch Wizard - REPORT Parameters panel
Panel ID is FQMPBTD3.

Batch Wizard - SMTP Settings panel
Panel ID is FQMPBTD5.

Columns panel
Panel ID is FQMPPQCE.

Column Description panel
Panel ID is FQMPCOLD.

Column List panel
Panel ID is FQMPPQCL.

Command panel
Panel ID is FQMPCMDS.

Command Prompt panel
Panel ID is FQMPCMPD.

Comparison Operator panel
Panel ID is FQMPPQOE.

Comparison Operators panel
Panel ID is FQMPPQCO.

Comparison Operator: Between panel
Panel ID is FQMPPQOB.

Connect to panel
Panel ID is FQMPCNDS.

Data Source Description panel
Panel ID is FQMPDSDS.

Data Source Object panel
Panel ID is FQMPFOB1.

Definition panel
Panel ID is FQMPFCDF.

Edit Row panel
Panel ID is FQMPTEED.

Export JCL panel
Panel ID is FQMPJEXP.

Favorite Actions panel
Panel ID is FQMPACTS.

Favorite Objects panel
Panel ID is FQMPFAVS.

Form Break Number panel
Panel ID is FQMPSPBK.

From Data Source panel
Panel ID is FQMPFOB2.

Form Detail Variation panel
Panel ID is FQMPSPDV.

From Open Object panel
Panel ID is FQMPFOB4.

From Repository panel
Panel ID is FQMPFOB3.

Help panel
Panel ID is FQMPHELP.

Help table of contents
Panel ID is FQMPMGSB.

JavaScript procedure prompt panel
Panel ID is FQMPPRMT.

Joins panel
Panel ID is FQMPPQJE.

Join Columns panel
Panel ID is FQMPPQJC.

Key Editor panel
Panel ID is FQMPKDLG.

List panel
Panel ID is FQMPLOCS.

Login panel
Panel ID is FQMPAUTH.

Object Description panel
Panel ID is FQMPODSC.

Prompt panel
Panel ID is FQMPMSGB.

Prompt Variables panel
Panel ID is FQMPVARS.

Query Number panel
Panel ID is FQMPSPQN.

Repository Description panel
Panel ID is FQMPREPD.

Recently Used panel
Panel ID is FQMPRCUS.

Result Set Number panel
Panel ID is FQMPSPRS.

Row Conditions panel
Panel ID is FQMPPQRC.

Save Objects panel
Panel ID is FQMPSVOB.

Screen Test panel
Panel ID is FQMPSTST.

Search panel
Panel ID is FQMPTESD.

Set Data Source panel
Panel ID is FQMPSTDS.

Select Data Source panel
Panel ID is FQMPSLDS.

Select Repository panel
Panel ID is FQMPSERP.

Select Object panel
Panel ID is FQMPOBLD.

Show Field panel
Panel ID is FQMPSHFD.

Show Global Variable panel
Panel ID is FQMPGLSH.

Sort Object List panel
Panel ID is FQMPOSRT.

Sort Conditions panel
Panel ID is FQMPPQSE.

Specify panel: Form.Columns
Panel ID is FQMPFCSP.

Specify panel: Prompted Query
Panel ID is FQMPPQSC.

Switch Repository panel
Panel ID is FQMPSRVR.

Tables panel
Panel ID is FQMPPQTE.

Windows panel
Panel ID is FQMPODOC.

Workspace Description panel
Panel ID is FQMPWSDS.

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, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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 information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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 illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A complete and current list of IBM trademarks is available on the web at http://www.ibm.com/legal/copytrade.shtml.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions:

Applicability: These terms and conditions are in addition to any terms of use for the IBM website.

Personal use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights: Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Glossary

The glossary provides brief descriptions of product terms.

accessibility

Features that help those with physical disabilities, such as restricted mobility or limited vision, use their computer.

batch objects

A batch object is a set of parameters that is used to create a JCL batch job.

calculated columns

Columns of data that you add to the query results.

reports

Text-based, tabular reports that are generated using query results as the data source and a form template.

command line

An interface that allows user to interact with the QMF application via entering commands.

data sources

Data sources are QMF entities that store connection information to access databases.

formatting options

You can customize how the query results will appear in the editor window. You can apply formatting options to entire columns, individual cells, column headings, and summary cells. You can also specify that column and cell formatting be applied based on the results of a conditional expression.

forms

Forms are considered objects and they can be saved in your repository, in the QMF catalog, or in a file. When you open a form object that has been saved, you are actually running the form object to generate the report. When opened, forms automatically use the currently active query results as the data source.

global variables

Global variables are variables that stay active while the current session of QMF is active. This is in contrast to substitution variables that are active only during the execution of an object (query, procedure, form). For objects that use global variables, the value currently defined for the global variable is used.

grouping and aggregation

Grouping and aggregation options can be applied to query result columns to organize the result data into logical or summarized groupings. By adding grouping and aggregation you can automatically obtain summary information about your data and display the data more logically.

LOB data

A large object (LOB) is a Db2 for z/OS and Db2 for UNIX® data type that houses nontraditional data such as text, multimedia, image, video, photograph, sound, or any very large data file inside a database table. Retrieving or saving LOB data can consume a substantial amount of resources.

object key

A unique identifier that is given to every database object. You can view the object key of a particular object by accessing the **Object List** panel, placing the cursor on the object, and pressing the **Describe** function key.

procedures

A set of commands that enable you to run queries, print reports, import and export data, as well as perform other actions.

Prompted Query Editor

When building queries using the Prompted Query editor, you supply tables as well as join, column, sort, and row information and the Prompted Query editor constructs the Structured Query Language (SQL) statements.

QMF catalogs

A set of database tables that contain saved objects (queries, procedures and forms); user resource limits and profiles; reports; and other miscellaneous settings and information. QMF catalogs reside on database servers that host a Db2 database.

Query Editor

An interface that allows you to open any database table that is accessible to you in your workspace.

query parameters

Query parameters contain the value that will be sent to the query and used at runtime.

relational query

A query is a request for information from a data source. To request information from a relational data source your query is constructed using SQL statements.

SQL Query Editor

For those with SQL experience, one way of creating a query is to type their own SQL statements in the SQL query editor. You can write a single SQL statement that will return a single result set or multiple SQL statements that will return multiple result sets.

substitution variables

Substitution variables are used to enter changing values to a SQL query at run time. This feature enables you to substitute a part of an SQL statement and make it more generic. Substitution variables are active only while the object (query, procedure or form) is running. As a result, only one object can access the substitution variable. The variable will not exist after the object is executed.

Table Editor

An interface that allows you to open any database table that is accessible to you in your workspace.

usage codes

Usage codes provide summary information about the data in a column. For example, usage codes can provide total summary information at the end of a column, or partial summaries at control breaks in a table. The usage codes available depend on the data in the column and the type of summary.

workspaces

All of the data sources and objects that you can access are contained in one or more workspaces that have been pre-populated for you by your administrator.

Index

F

G

H

I

L

M

N

P

Q

R

S

T

U

V

W

Product Number: 5697-QM2
5650-DB2
5615-DB2

GC27-9133