Query Management Facility
Version 12 Release 2

QMF Reference

SC27-8880

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 3, 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

IBM® Db2® Query Management Facility for TSO and CICS® is a tightly integrated, powerful, and reliable tool that offers query and reporting functions that help you access and present data from any of the following relational databases:

- DB2® for z/OS®
- Db2 for Linux®, UNIX, and Windows
- DB2 for iSeries
- DB2 Server for VSE and VM

These topics are designed to help users, programmers, and database administrators of QMF for TSO and CICS to understand this information:

- The syntax and usage of commands
- How to use SQL keywords in QMF queries
- How to use forms, reports, and charts (including usage and edit codes)

What you should know before you begin

The topics in contain basic QMF information; the reference information provided here assumes that you have been through the tasks and concepts in that guide. In addition to the steps necessary to get started with QMF and learn how to write SQL queries, contains detailed scenarios that show how to build queries and forms step by step. It also contains information about Query-By-Example. You can obtain QMF publications from the or the IBM Publications Center.

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, refer to the following Web page:

IBM Software Support website

Highlighting conventions

This information uses the following highlighting conventions:

- **Boldface** type indicates commands or user interface controls such as names of fields, folders, icons, or menu choices.
- `Monospace` type indicates examples of text that you enter exactly as shown.
- *Italic* indicates the titles of other publications or emphasis on significant terms. It is also used to indicate variables that you should replace with a value.

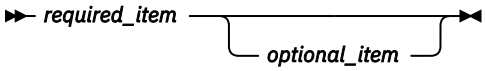
How to read syntax diagrams

The following rules apply to the syntax diagrams that are used in this information:

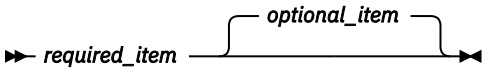
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol indicates that a syntax diagram is continued from the previous line.

  - The --->< symbol indicates the end of a syntax diagram.
- Required items appear on the horizontal line (the main path).

- Optional items appear below the main path.

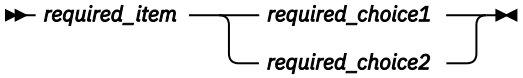


  If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

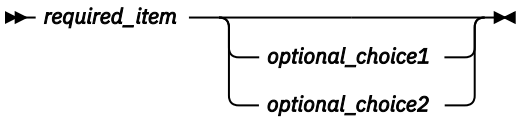


- If you can choose from two or more items, they appear vertically, in a stack.

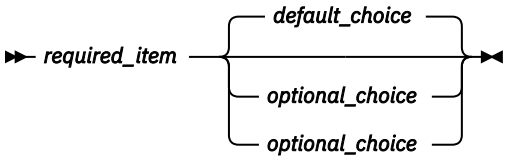
  If you *must* choose one of the items, one item of the stack appears on the main path.



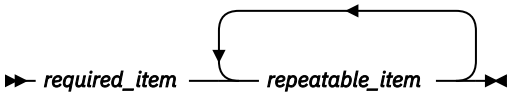
  If choosing one of the items is optional, the entire stack appears below the main path.



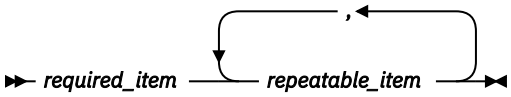
  If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



- An arrow returning to the left, above the main line, indicates an item that can be repeated.



  If the repeat arrow contains a comma, you must separate repeated items with a comma.



  A repeat arrow above a stack indicates that you can repeat the items in the stack.
- Keywords, and their minimum abbreviations if applicable, appear in upper case. They must be spelled exactly as shown. Variables appear in all lowercase italic letters (for example, *column-name*). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.

- Enter punctuation marks, parentheses, arithmetic operators, and other symbols exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses; for example, (1).

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other documentation, use either of the following options:

- Use the online reader comment form, which is located at:

  http://www.ibm.com/software/data/rcf
- Send your comments by e-mail to comments@us.ibm.com. Be sure to include the name of the book, the part number of the book, the version of your product, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).

Chapter 1. Db2 function level support

When you activate new Db2 function levels in a Db2 subsystem or data sharing group, enhancements might become available that impact QMF for TSO and CICS.

The levels of function level support are defined as follows:

Tolerated
The product works as it did on a previous release or function level of Db2 for z/OS, but it does not support the new features of this function level.

Supported
The product supports most, but not necessarily all, of the new function-level features that IBM deems the most significant for this product.

The following function levels are tolerated or supported by QMF for TSO and CICS and are provided with the corresponding PTF, if any:

- QMF for TSO and CICS PTFs in support of Db2 13 function levels
- QMF for TSO and CICS PTFs in support of Db2 12 function levels

Table 1. QMF for TSO and CICS PTFs in support of Db2 13 function levels

Db2 13 function level	Toleration PTF	Support PTF
FL504	No PTF required	No PTF required
FL503	No PTF required	No PTF required
FL502	No PTF required	No PTF required
FL501	No PTF required	No PTF required
FL500	No PTF required	No PTF required
FL100	No PTF required	No PTF required

Table 2. QMF for TSO and CICS PTFs in support of Db2 12 function levels

Db2 12 function level	Toleration PTF	Support PTF
FL510	No PTF required	No PTF required
FL509	No PTF required	No PTF required
FL508	No PTF required	No PTF required
FL507	No PTF required	No PTF required
FL506	No PTF required	No PTF required
FL505	No PTF required	No PTF required
FL504	APARs PH14686, PH15724	APARs PH14686, PH15724
FL503	No PTF required	No PTF required
FL502	No PTF required	No PTF required
FL501	No PTF required	No PTF required
FL500	No PTF required.	No PTF required

To access and use the features of the new Db2 for z/OS function levels, bind the QMF for TSO and CICS product packages at the new function level. For more information, see Exploiting New Db2® for z/OS® Function Levels.

Chapter 2. QMF commands

Look up syntax, option descriptions, and usage information for commands used with QMF.

QMF command environments

You can enter QMF commands from TSO or CICS environments. In TSO, you might also be using ISPF.

In a small table at the beginning of each command description, an X indicates which environments accept the command. An asterisk (*) indicates that only certain aspects of the command are accepted. For example:

TSO with ISPF	TSO without ISPF	CICS
X	X	*

In cases where there is only one environment to which the command applies, the name of the environment is included in the topic title and a table is not shown.

Entering commands

You can issue QMF commands in several ways: on the command line, with a function key, on a prompt panel, or from a procedure or application.

If your site defined a command synonym with the same name as a QMF command, you must precede the command with QMF to override the synonym.

On the command line

Where a command line appears, you can enter any QMF command by typing it in full after the arrow. For example:

```
COMMAND ===> RUN MYQUERY (FORM=FORM2
```

To run the command, press Enter.

With a function key

You can enter some commands by using function keys. QMF has a default set of function keys for each panel. The function keys that you see when you use QMF can differ from the defaults if your administrator customized them. This information refers to the default set of function keys.

To use parameters with a function key command, type the parameters on the command line, then press the function key. For example, when the query panel is displayed, type (FORM=FORM2, then press the Run function key. The following command is run:

```
RUN QUERY (FORM=FORM2
```

On a prompt panel

QMF displays a command prompt panel if you enter a command with a syntactical error (or a misspelling twice in succession), or when you enter the command name followed by a question mark on the command line. This prompt panel is useful when you enter long commands.

For example, when you enter RUN ?, the command prompt panel that is shown in the following figure is displayed, on which you can enter the required information:

```
                        RUN Command Prompt
                                                                    1 to 8 of 8
Type (                          )
Name (<------------------------------------------------>)+
.... (<------------------------------------------------>)+
.... (<------------------------------------------------>)+
.... (<------------------------------------------------>)+
.... (<------------------------------------------------>)+
.... (<------------------------------------------------>)+
     To run an object from temporary storage, enter its type:
     QUERY or PROC.
     To run an object from the database, enter its name (and
     optionally its type). Type can be QUERY or PROC.

F1=Help   F3=End   F4=List   F7=Backward   F8=Forward
```

Figure 1. RUN Command Prompt panel

If the command references an object name and the object name is too long to fit on one line, continue typing the name on the next line. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

Entries in the **Name** field that begin with the characters ALL must be delimited in double quotation marks. For example, if you want to list all objects whose names begin with ALL, type "ALL%" in the **Name** field and press the List key.

If QMF needs more information to complete a command, a second panel might be displayed to prompt you for command parameters.

You can skip the first panel of this two-step prompt by entering the command, the object type, and the object name, followed by a question mark on the command line. A panel appears containing the parameters that are applicable to that object.

A question mark is not valid in the parameters portion of a command (after the left parenthesis). Also, any parameters that follow the question mark are ignored. For example, (FORM=FORM2 is ignored in the following command:

```
RUN QUERY MYQUERY ? (FORM=FORM2
```

The following function keys appear on most prompt panels:

Help
Displays help information about the displayed message.

List
Displays a list of objects from which you can select.

End
Returns to the panel from which the prompt was issued.

From a procedure

You can include most QMF commands as a line in a procedure, including a RUN command that runs another procedure. This feature is helpful when you use commands that are too long to enter on the command line.

When you put commands into a procedure, use the full command names, parameters, and values rather than the abbreviations. The minimum acceptable abbreviation for an existing word might change in future releases and cause your procedure to fail.

When you use QMF commands in a procedure with logic, the commands:

- Must be in uppercase, regardless of the profile setting
- Can be continued by ending the line with a comma
- Can contain substitution variables

Commands in linear procedures can be continued over more than one line by placing a plus sign (+) as a continuation character in column 1 of each additional line. The continued line then starts in column 2.

An object name, authorization ID, or location must be within double quotation marks (delimited identifiers) when continued over more than one line, as shown in the following figure:

```
PROC MODIFIED LINE 1

ERASE QUERY
+"LOCATION12345678"."LONGOWNERID123456789121234567893123456789412345678
+12345678911234567892123456789312345678941234567895123456789612345678971234
+LONGNAME12345678911234567892123456789312345678941234567895123456789612345678971
+123456789112345678921234567893123456789412345"
```

Figure 2. Continuing a qualified object name over more than one line in a linear procedure

Use single quotation marks when you use the LIST command.

From an application

You can enter QMF commands from within applications that use the following interfaces. QMF commands within applications must be entered in uppercase, regardless of how the CASE option of the QMF profile is set.

Command interface
Receives QMF commands from ISPF. QMF must be started before the application or CLIST is run. The command interface is not available in CICS, as its function depends on ISPF.

Callable interface
Receives QMF commands directly from the QMF common programming interface (CPI). You can start and stop QMF from your application. ISPF is not required.

Related concepts

Procedures
When you start QMF, the system initialization procedure runs to configure the QMF session.

Related reference

LIST
Use the LIST command to display lists of QMF objects and database tables stored in the database. When you first issue the LIST command in a QMF session, ensure that you use one of these parameters: Queries, Forms, Procs, Analytics, Folders, QMF, Tables, or All.

Related information

The callable interface and QMF applicationsProgramming languages can use the QMF callable interface to run QMF commands.

QMF commands that access data at a remote server

Several points apply to QMF commands that access data at a remote server.

- Unless the QMF command specifies a three-part table or view name, it applies to data at the location to which you are currently connected.
- If you are using three-part names in your commands and your database administrator has set up QMF to use the multirow fetch feature, both databases you are working with must be Db2 for z/OS. Your database administrator can turn off this feature if necessary.
- QMF commands with three-part names cannot be directed to DB2 for VSE and VM.
- By default, three-part names cannot be used to access remote tables that contain LOB data. However, you can set the DSQEC_LOB_RETRV global variable to 2 or 3 to access LOB metadata or data with a three-part name. Or, you can use the CONNECT command to connect to the database, and then run the query to access the remote table.

- References to QMF procedures, queries, forms, folders, and analytics objects in the database apply to the current location. You cannot refer to a procedure, query, form, or analytics object with a three-part name.
- Data sets or files named in QMF commands must reside at the system on which QMF was started. QMF Version 12.1 can be started only on Db2 for z/OS Version 9.1 New Function Mode or later.
- CICS data queues named in QMF commands must be defined at the system on whichQMF is executing.
- References to stored profile values apply to the current location, except for the TRACE parameter.
- When QMF is running in , all database objects (tables, views, procedures, queries, forms, folders, and analytics objects) at remote databases are read-only.
- When QMF for TSO has been started as a stored procedure, you cannot access data from a remote server.
- Using three-part names, data may be accessed from a QMF Data Service server. The DSQEC_DS_SUPPORT global variable controls this behavior. QMF Data Service data may be accessed from the DISPLAY TABLE, DRAW, EXPORT TABLE, PRINT TABLE and RUN QUERY commands for SQL queries, Prompted queries, and Query-by-Example queries.

Confirmation panels

If there is a CONFIRM parameter on a command, you can specify YES or NO (or use the default in your profile).

If the command modifies the database and the CONFIRM parameter is YES, a confirmation panel like the following one is displayed:

```
                        RUN CONFIRMATION

WARNING:
Your RUN command modified this number of rows in the
database:         1

Do you want to make this change?
1 1. YES - Make the changes permanent in the database.
  2. NO  - Roll back the changes to the last COMMIT operation
           or to the beginning of the query. The database that you are
           using with QMF determines the rollback rules.
```

Figure 3. Example of a confirmation panel

If the query contains multiple SQL statements, your response to the confirmation panel applies to all statements in the query unless the query contains multiple COMMIT statements. If the query does not contain multiple COMMIT statements, the answer that you provide in response to the single prompt applies to all changes that are made by all SQL statements in the query. If the query contains multiple statements that change the database and these statements are of different types, the confirmation prompt asks about only one type of statement. For example, if the query contains a DROP statement and an UPDATE statement, the confirmation prompt refers to the UPDATE statement only; however, your response to the prompt applies to both the DROP and UPDATE statements in this case.

If the query contains multiple SQL statements and multiple COMMIT statements, a confirmation panel is displayed for every COMMIT statement. However, if a COMMIT statement follows SQL statements that change only a database catalog, a confirmation panel is not displayed for that COMMIT statement.

Many QMF confirmation panels for changes to the database are actually prompting you to do a commit (by entering YES to keep the changes) or a rollback (by entering NO). Because the changes were already made to the database, the database manager holds locks on the data until you reply YES or NO on the confirmation panel.

If you are connected to DB2 Server for VSE and VM, the tables you are working with might be in a nonrecoverable dbspace. If so, any changes you make are committed to the database immediately; you cannot execute a rollback. Therefore, if a table is in a nonrecoverable dbspace, specifying NO on the confirmation panel will not prevent the changes from taking place.

Related information

Search for information about dbspaces by referring to the administration information for DB2 Server for VM or VSE.

Canceling commands

The method you use to cancel a QMF command or query that is currently in process depends on the type of terminal connection you have and your environment.

Procedure

To cancel commands:

- In TSO:
  - If your terminal is connected directly to the system, press the Reset key, then the PA1 key.
  - If your terminal is connected by network, press the ATTN key.
  - If you are using a terminal emulator to simulate the operating environment, display the pop-up menu for the session you want to cancel. The PA1 and ATTN keys are on this menu.
- In CICS:

  The CICS operator must cancel the QMF transaction like any other CICS transaction. You cannot use the PA1 and ATTN keys in CICS. When a QMF transaction is canceled, all work is lost.

Command parameters

A command can allow positional parameters and keyword parameters.

Positional parameters must be placed in a certain position within a command. Keyword parameters are assigned a value and can be placed in any order within a command. The first keyword parameter used in a command must be preceded by a left parenthesis.

If a command allows keyword parameters, you can use as many as you need. If you use a keyword parameter more than once in a command and provide different values for the parameter, its last value takes effect. No parameter value can be longer than 80 characters.

All parameters are separated from each other with a blank or a comma followed by an optional blank. For example, all of the following specifications are correct:

```
(MEMBER=member CONFIRM=YES
(MEMBER=member, CONFIRM=YES
(MEMBER=member,CONFIRM=YES
(MEMBER member CONFIRM=YES
(MEMBER member CONFIRM YES
```

A right parenthesis is not required, but can be used to end the command. Anything you put after it is treated as a comment; it is not processed.

ADD

Use the ADD command to add rows to a table in the Table Editor or add global variables to the global variable list.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

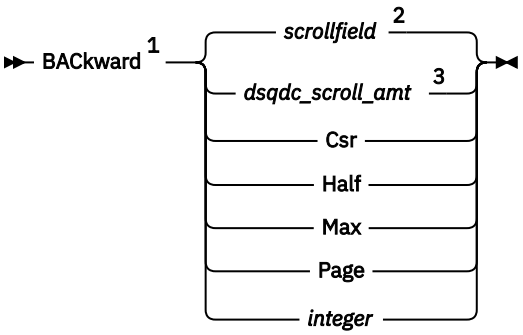
Usage notes

- In the Table Editor, a transaction is either saved immediately or when you end your Table Editor session, depending on what you specify for the SAVE option on the EDIT command.
- In the Global Variable List, the ADD command displays the Add Variable panel so you can add a new variable.

BACKWARD

The BACKWARD command scrolls toward the top of the active panel (or to the first field of the current row if you are using the Table Editor). In a panel, you can scroll backward to the cursor position, to the beginning, or scroll a half page, a full page, or a specific number of lines.

TSO with ISPF	TSO without ISPF	CICS
X	X	X



Notes:

[1] Specify scroll amounts only when there is a SCROLL field on the active panel. PAGE is assumed in all other situations.
[2] The value shown in the SCROLL field is used. This value is also maintained in the global variable DSQDC_SCROLL_AMT.
[3] The value set in this global variable is used.

Description

CSR
Scrolls the line where the cursor is positioned to the bottom of the scrollable area.

HALF
Scrolls backward half the depth of the scrollable area or to the top (if that is nearer).

MAX
Scrolls to the top of the scrollable area.

PAGE
Scrolls backward the depth of the scrollable area or to the top (if that is nearer).

integer
Scrolls backward this number of lines on the panel (a positive integer up to 9999).

Usage notes

- MAX is in effect only for the current command. This value will not remain in the SCROLL field after the command completes. You cannot set the global variable DSQDC_SCROLL_AMT to this value.
- To scroll backward in the footing text on form panels, position the cursor on the portion of the panel where the footing text is located and enter the BACKWARD command.

- You can also change the scroll amount QMF uses by setting the global variable DSQDC_SCROLL_AMT to Csr, Half, Page, or a positive integer up to 9999.

BATCH

BATCH is a command synonym supplied by QMF that accesses the QMF batch application. This application lets you run queries and procedures as QMF batch jobs rather than interactively.

TSO with ISPF	TSO without ISPF	CICS
X		

The QMF BATCH command supports object names of the lengths shown in the following table.

Table 3. Field lengths for object names on the QMF BATCH command

Field name	Maximum length
Object name (name of query or procedure)	77
Form name	77
Batch name (name of QMF batch procedure)	31
Save data (name of data to be saved)	77

The BATCH command also allows for the input of long variables. You can use the scroll indicator to help you enter these variables. The scroll indicator looks like the following:

```
< > 31 60
```

The left and right carets are directional indicators and the numbers represent beginning and ending positions.

BOTTOM

The BOTTOM command scrolls to the last line of queries, procedures, reports, global variable lists, and scrollable form panels.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Usage notes

- BOTTOM is equivalent to FORWARD MAX.
- To scroll to the bottom of footing text on form panels, position the cursor on the portion of the panel where the footing text is located and enter the BOTTOM command.

CANCEL

Use the CANCEL command to discard pending modifications made during a Table Editor session. You can also use the CANCEL command to return to a primary QMF panel from a help panel or to cancel a confirmation panel for a command.

When you press the Cancel function key from a confirmation panel, the command whose action you were asked to confirm is canceled and you return to the QMF panel on which you entered the command.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Usage notes

- The CANCEL command is only available as a function key. You can use the CANCEL function key from the Table Editor, QMF help panels, and confirmation panels.
- CANCEL is available in the Table Editor session depending on the SAVE option specified on the EDIT TABLE command:
  - When SAVE=END, changes are discarded when the Cancel function key is pressed.
  - When SAVE=IMMEDIATE, CANCEL is not accepted.

CHANGE

In Prompted Query, the CHANGE command displays a panel on which you can make changes. In the Table Editor, the CHANGE command modifies rows in a table or view.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Usage notes

- In Prompted Query, you can use one of the following methods to make changes:
  - In the echo area, position the cursor on the underscore character that appears to the left of the specification to be changed. To change a specification that is longer than one line, place the cursor on the first line of the specification. Then, press the Change function key.
  - Type CHANGE on the command line, then position the cursor on the underscore character that appears to the left of the specification to be changed. To change a specification that is longer than one line, place the cursor on the first line of the specification. Then, press Enter.
- In the Table Editor, when you press the Change function key:
  - When SAVE=IMMEDIATE, changes are saved when the transaction is processed.
  - When SAVE=END, changes are saved when the END command is processed.

CHECK

The CHECK command checks form panels for errors and conflicting entries.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Usage notes

- When a form panel is displayed, you can enter CHECK on the command line or press the Check function key. QMF checks for detectable errors in the displayed panel and then checks the remaining form panels.
- The message line describes the error that must be corrected before other errors are shown.
- When one error is displayed, you can display any additional errors by correcting the currently displayed error and pressing the Check key.
- CHECK cannot detect all errors. Some errors are not evident until you display the report, when QMF displays an error message.

Error conditions

If a form panel contains an error, QMF displays the panel on which the first error occurs, with the word ERROR at the top of the panel. If only one form panel contains an error, QMF displays the word ERROR on all the form panels. The entry area containing the error is highlighted, and the cursor is positioned next to it. The message on the message line describes the error.

You must correct the error before you can see the next error or create the report. For more information about the error and what you must do to correct it, press the Help function key. To identify the next error, enter the CHECK command again and correct the error. Continue in this way until you correct all errors.

If FORM.CALC, FORM.CONDITIONS, or a column definition panel in FORM.COLUMNS contains an expression with an error, this error might not be detected until QMF passes the values to REXX for evaluation.

Warning conditions

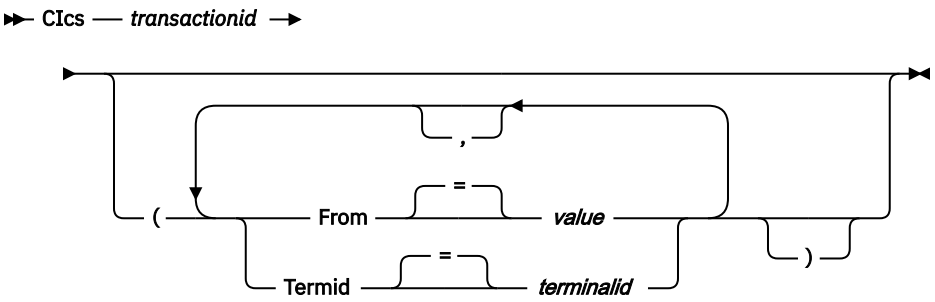
If the form panels have no errors, or if you corrected all of them, QMF checks for warning conditions. If a warning condition is found, QMF displays the form panel on which the first warning condition occurs, with the word `WARNING` at the top of the panel. In addition, the cursor is positioned next to the entry area containing the conflicting value, and a message describes the condition.

Unlike errors, warnings are not highlighted, and you can see all the warning conditions (without having to change the conflicting values) by repeatedly issuing the CHECK command. You do not need to change values that cause warning conditions – QMF can interpret the values and format your report. However, the report might not show the expected results.

CICS

The CICS command starts a CICS transaction when you are running QMF under CICS. The transaction can be started without ending your current QMF session.

TSO with ISPF	TSO without ISPF	CICS
		X



Description

transactionid
The name of a CICS transaction to be started. This is a one- to four-character value.

FROM
Specifies the data passed to the transaction. Up to 78 characters of data can be passed.

value
The character string that makes up the contents of your data.

A value that contains blank characters must be surrounded with delimiters. Valid delimiters for a data value are single quotation marks, parentheses, and double quotation marks.

TERMID
Specifies the CICS terminal associated with the transaction.

This option is required for any transaction that must communicate with a terminal. In any other case, omit this option to start the transaction without an associated terminal.

terminalid
A CICS terminal identifier. This is a one- to four-character alphanumeric value.

The current CICS terminal identifier for your QMF session is listed on the QMF CICS command prompt panel.

Usage notes

- The QMF CICS command parameters (transactionid, FROM, and TERMID) have the same meanings as the CICS START command options (TRANSID, FROM, and TERMID).
- The CICS transaction is scheduled to start immediately.
- The CICS transaction must conform to the rules governing CICS Basic Mapping Service, GDDM applications, and the CICS START command.

Examples

- To display a prompt panel for the QMF CICS command, enter:

```
CICS ?
```

- To use a global variable on the FROM parameter, surround the global variable with parentheses. For example:

```
CICS transid (FROM=(&DSQAP_CICS_PQNAME)
```

Do not surround the global variable with single quotation marks; it will not resolve correctly.

CLEAR

Use the CLEAR command to erase input from all fields in the Table Editor.

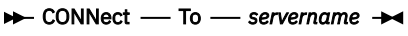
TSO with ISPF	TSO without ISPF	CICS
X	X	X

CONNECT in CICS

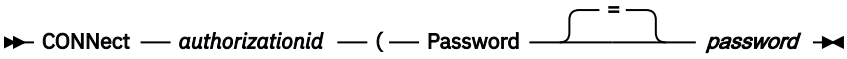
With the CONNECT command, you can connect to any database server that is part of the distributed network from within a QMF session. If you are connected to a DB2 Server for VSE and VM database, you can also use the CONNECT command to change the database user.

Syntax

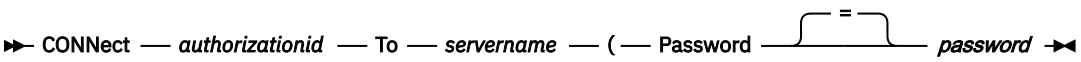
CONNECT to a database server



Change the database user (when connecting to DB2 Server for VSE databases only)



CONNECT to a server and set the user (when connecting to DB2 Server for VSE databases only)



Description

authorizationid

A user ID at a remote DB2 Server for VSE and VM database management system. The user ID must possess CONNECT authority to the database.

The user ID can be delimited with double quotation marks. If the user ID is "TO", or an abbreviation of "TO", it must be enclosed within double quotation marks. For example:

```
CONNECT "T" TO MIAMI (PASSWORD=password
```

When you specify an authorization ID on the CONNECT command, the QMF session operates with the privileges held by the newly established runtime authorization ID. Reconnecting to the database under a different authorization ID can be helpful if you need to perform privileged database administration tasks by changing the connection to an ID with DBA authority. Reconnecting to the database with a different authorization ID changes the USER special register in DB2 for VSE and VM.

The database authorization ID at a Db2 for z/OS server cannot be passed on the CONNECT command in QMF for CICS. Instead, it can be changed by running a QMF SQL query with a SET CURRENT SQLID statement. For example:

```
SET CURRENT SQLID = 'QMFADM'
```

The QMF session is connected to a Db2 for z/OS server when the global variable DSQAO_DB_MANAGER has the value of 2.

servername

The location parameter, which is the name of a database application server in the distributed network.

The server name can be delimited with double quotation marks.

A list of server names is available for this parameter when using the CONNECT command prompt panel.

Password
The password for the database user who is attempting to connect to the DB2 for VSE and VM database. The password cannot be blank.

The password can be surrounded with delimiters. Valid delimiters are single quotation marks or double quotation marks.

Usage notes

- When using CICS with a remote database server, all data at the server is restricted to read-only.
- Notes on database authorization IDs:
  - The default database authorization ID for each server is system-defined.
  - The maximum length of the database authorization ID used to make the connection, as well as the maximum length of any table and column names used afterward, is determined by the database to which the CONNECT command is directed.
- Differences between the CONNECT command and the DSQSDBNM program parameter include the following:
  - The DSQSDBNM parameter establishes the initial database server used for the QMF session.
  - The CONNECT command changes the database server after a QMF session is established.
- The CONNECT command cannot be used in a QMF query.

Examples

1. To display the CONNECT command prompt panel:

```
CONNECT ?
```

2. To connect to a remote database server with a location name of MIAMI:

```
CONNECT TO MIAMI
```

3. DB2 Server for VSE and VM only:
   - To change the database user to "QMFADM", having a password of "A12ZDT":

```
CONNECT "QMFADM" (PASSWORD="A12ZDT"
```

   - To connect to another location and change the database user:

```
CONNECT QMFADM TO MIAMI (PASSWORD=A12ZDT
```

Related reference
CONNECT in TSOFor more information about usage, see "Connecting to Db2 databases within a distributed network."

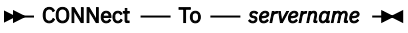
CONNECT in TSO

You can use the CONNECT command from within a QMF session to connect to any database server that is part of the distributed network.

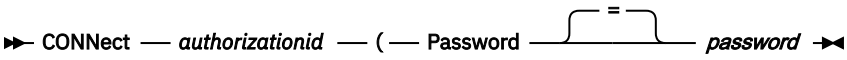
TSO with ISPF	TSO without ISPF
X	X

Syntax

CONNECT to a database server



Change the database user



CONNECT to a database server and set the user

Notes:

[1] The *servername* must specify a Db2 for z/OS Version 8.1.5 or higher server.

Description

authorizationid

A user ID at a remote database management system. The user must have been granted CONNECT authority with a password.

The default database authorization ID for each server is defined at the time of installation.

The authorization ID can be delimited with double quotation marks. If the authorization ID is "TO", or an abbreviation of "TO", it must be enclosed within double quotation marks. For example:

```
CONNECT "T" TO MIAMI (PASSWORD=password
```

When you specify an authorization ID on the CONNECT command, the QMF session operates with the privileges held by the newly established runtime authorization ID. Reconnecting to the database under a different authorization ID can be helpful if you:

- Need to perform privileged database administration tasks by changing the connection to a user ID with administrator authority
- Run batch jobs and need to set the database user to something other than the batch machine's user ID

The database authorization ID for a Db2 for z/OS server can be changed by running a QMF SQL query with a SET CURRENT SQLID statement. For example:

```
SET CURRENT SQLID = 'QMFADM'
```

If the authorization ID is long and therefore spans multiple lines, enter the information on a command prompt panel.

servername

The location parameter, which specifies the name of a database application server in the distributed network.

The server name can be delimited with double quotation marks.

A list of server names is available for this parameter when using the CONNECT command prompt panel.

Password

The password for the database user. The password cannot be blank. It can be surrounded by either single or double quotation marks.

Usage notes

- Passwords are necessary to ensure security and protect against unauthorized access to catalogs and control table spaces.
- Db2 for z/OS uses RACF® to define user IDs and passwords. If your site takes advantage of RACF support for mixed-case passwords, ensure that the CASE option of your QMF profile is set to MIXED. Otherwise, QMF converts all input to uppercase, causing the CONNECT command to fail.
- Notes on authorization IDs:
  - Connecting to a database server resets the database authorization ID.
  - Double quotation marks must be used when continuing an authorization ID across more than one line within a QMF linear procedure.
  - The default database authorization ID for each server is system-defined.
  - The database authorization ID at a Db2 for z/OS server can be changed by running a QMF SQL query with a SET CURRENT SQLID statement. For example:

    ```
    SET CURRENT SQLID = 'QMFADM'
    ```

    The QMF session is connected to a Db2 for z/OS server when the global variable DSQAO_DB_MANAGER has the value of 2. The database authorization ID cannot be changed when the global variable DSQAO_DB_MANAGER has a value other than 2.
  - The maximum length of the database authorization ID used to make the connection, as well as the maximum length of any table and column names that are used afterward, is determined by the database to which the CONNECT command is directed.
- The following are differences between the CONNECT command and the DSQSDBNM program parameter:
  - The DSQSDBNM parameter establishes the initial database server that is used for the QMF session.
  - The CONNECT command changes the database server after a QMF session is established.
- You cannot connect to a remote database if QMF is started as a stored procedure.
- The CONNECT command cannot be used in a QMF query.

Examples

1. To display the CONNECT command prompt panel:

   ```
   CONNECT ?
   ```

2. To connect to a remote database server with a location name of MIAMI:

   ```
   CONNECT TO MIAMI
   ```

The CONNECT command in a QMF procedure

You must use double quotation marks to continue an authorization ID across more than one line within a QMF linear procedure. All continuation lines must have a plus sign (+) in column one, as shown in the following figure:

```
PROC                        Test_Connect                          MODIFIED LINE   1
 CONNECT “A234567890123456789012345678901234567890123456789012345678901234567890
 +12345678901234567890123456789012345678901234567890123456789012345678” (PASSWORD=XYZ)
```

Figure 4. Continuing an authorization ID across more than one line in a QMF linear procedure

Connecting to Db2 databases within a distributed network

When you connect to a remote location, it becomes your current location. These connections can be made between like (Db2 for z/OS to Db2 for z/OS) and unlike locations (DB2 Server for VSE and VM Version 7.3 or later; Db2 for Linux, UNIX, and Windows Version 9.1 or later; and DB2 for iSeries Version 5.4 or later). You can establish this connection during QMF initialization by using the DSQSDBNM program parameter when you start QMF or by issuing the QMF CONNECT command from within a QMF session.

The maximum length of the database authorization ID used to make the connection, as well as the maximum length of any table and column names that are used afterward, is determined by the database to which the CONNECT command is directed.

When you are connected to a remote location, all SQL statements you issue (except CONNECT) are directed to the database at that location. Therefore, you can access data and QMF objects at a remote location in the same way you would access data and objects locally. For example, you can create a table or replace comments on a table at a remote location by first connecting to that location using the QMF CONNECT command.

CONVERT

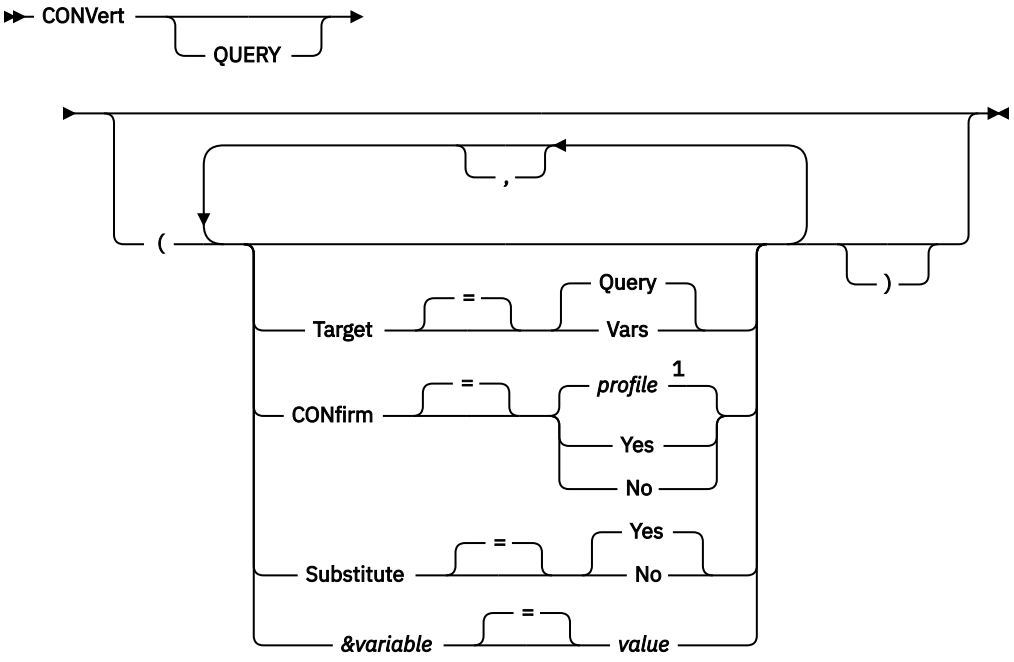
The CONVERT command converts a Prompted, SQL, or QBE query into a query with standard SQL syntax.

Converting a query can be useful if you want to expand a basic prompted or QBE query into a more complex query using the SQL language. The CONVERT command can also be used to improve the organization of an existing query on the SQL Query panel.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Substitution variables can be replaced with values you specify or with values defined by global variables. CONVERT assigns values to variables and removes all original comments from the query.

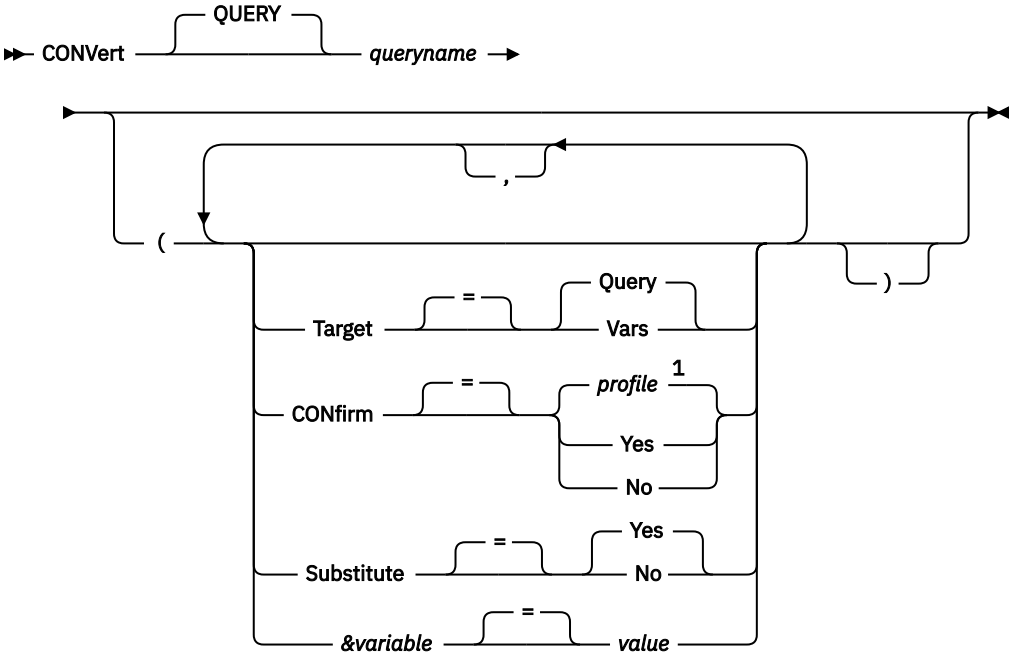
CONVERT a query in temporary storage



Notes:

[1] The value set in your profile is used.

CONVERT a query from the database



Notes:

[1] The value set in your profile is used.

Description

queryname
Name of a query stored in the database. The query stored in the database is unchanged, and the query in QMF temporary storage is replaced with a copy of the stored query.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.To display the panel, issue the following command:

```
CONVERT ?
```

TARGET
Controls the placement of the converted query.

QUERY
Places the converted query on the SQL Query panel. The query in your temporary storage area, regardless of its type, is replaced with the converted query. Thus, any existing prompted or QBE query in temporary storage is lost if it has not been saved in the database prior to the conversion.

A single QBE insert or delete query can result in multiple SQL statements after the conversion. These statements are all placed on the SQL Query panel. However, all of the statements after the first are turned into query comments (each line is preceded by two hyphens). Use the Delete key to remove the hyphens from any statements that you want to run. You must place a semicolon at the end of every SQL statement except the last. The DSQEC_RUN_MQ global variable controls support for multistatement queries.

VARS
Places the converted query and related information about the query in QMF global variables beginning with DSQQC. If ISPF is available, the converted query is also placed in the ISPF dialog manager variable pool. (ISPF is not available in CICS.) The query in your temporary storage area is not changed. Only the global variables and the ISPF variable pool are changed.

When you specify the TARGET=VARS option, the converted query cannot exceed 32,768 bytes in length. If the converted query is larger than this value, use the TARGET=QUERY option or shorten the query before running the command.

CONFIRM
Indicates whether a confirmation panel is displayed when this command will replace an existing object in the database.

SUBSTITUTE
Indicates whether to replace substitution variables in your query with values.

YES
If you have variables in your query, QMF attempts to substitute values for them. If all the variables are defined, no prompt panel is displayed. If QMF cannot resolve all the variables, it prompts you to enter values.QMF first looks at the command for a variable definition before looking at existing global variables.

NO
No variable names in your query are resolved.

&variable
Identifies a substitution variable for the CONVERT command. Variables can be assigned values up to 55 single-byte characters long with this option. Up to 10 substitution variables can be specified in a single command.

Variable names that do not match the variable names in your query are ignored. If you defined your variables with the SET GLOBAL command, you do not have to specify them on the CONVERT command. A value specified on the CONVERT command overrides a value set with SET GLOBAL. If you have variables in your query and do not specify substitution values for all of them on your CONVERT command, a prompt panel is displayed. All supplied parameter values appear on the prompt panel. Any variable names included in your query that aren't assigned values are listed and a message is displayed.

The variable name must be prefaced with an ampersand. Use two ampersands if you issue the CONVERT command from within a linear procedure.

value
The character string that makes up the content of the substitution variable.

A value that contains blank characters must be surrounded with delimiters. Valid delimiters for a substitution variable value are single quotation marks, double quotation marks, and parentheses. When the delimiters are quotation marks, the quotation marks are included as part of the value. When the delimiters are parentheses, the parentheses are not included as part of the value. Do not enter a query comment as a variable value. Query comments are preceded by two dashes (--), which the database interprets as minus signs.

Usage notes

- Queries cannot have three-part names.
- If you provide values for substitution variables and also specify SUBSTITUTE=NO, an error message is issued.
- QMF updates the Last Used field for the object when you use this command. This field appears on object list panels displayed by the LIST command. You can change the list of commands that cause the field to be updated by setting the DSQEC_LAST_RUN global variable.

Examples

1. To convert a query in QMF temporary storage into an SQL query and substitute a value of 38 for the variable DEPT in the converted query:

```
CONVERT QUERY (&DEPT=38
```

2. The following example shows how to use the CONVERT command to improve the organization of an existing SQL query. For example, suppose the SQL query in temporary storage is:

```
SELECT 'JOB',JOB,'SERIAL',ID FROM Q.STAFF
    WHERE ID<99 ORDER BY 2
```

   The converted query after running the CONVERT command is the following:

```
SELECT 'JOB', JOB, 'SERIAL', ID
FROM Q.STAFF
WHERE ID < 99
ORDER BY 2
```

3. To convert a saved query named QBEQUERY into an SQL query in QMF temporary storage:

```
CONVERT QUERY QBEQUERY
```

4. To convert a saved query named MYQUERY into an SQL query, and put it into the ISPF dialog manager pool and the global variable pool:

```
CONVERT QUERY MYQUERY (TARGET=VARS
```

Related reference

Global variables that control how commands and procedures are executed
DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

Global variables that store results of CONVERT QUERY
DSQQC global variables reflect the results of a CONVERT QUERY command. None of these global variables can be modified by the SET GLOBAL command.

DELETE

The DELETE command removes different elements, depending on the panel on which it is used.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

The DELETE command removes any of the following elements:

- A line from an SQL query or a procedure
- A line from a panel in Prompted Query
- A line of column information on FORM.MAIN or FORM.COLUMNS
- A calculation line from a FORM.CALC panel
- A condition from FORM.CONDITIONS
- A text line on FORM.BREAK, FORM.DETAIL, FORM.FINAL, or FORM.PAGE
- An error message that is displayed below a query
- A row from a table in the database when you use the Table Editor

Usage notes

- To delete a line, position the cursor on the line to be deleted and press the Delete key.

  You can delete a specification from a prompted query in one of two ways:

  - In the echo area, position the cursor on the underscore character that appears to the left of the specification to be deleted. If the specification is longer than one line, place the cursor on the first line of the specification. Then, press the Delete function key.
  - Type DELETE on the command line, and then position the cursor on the underscore character that appears to the left of the specification to be deleted. If the specification is longer than one line, place the cursor on the first line of the specification. Then, press Enter.
- When you use DELETE in the Table Editor, the transaction is saved immediately or when you end your Table Editor session. You can specify which method you want used with the SAVE option on the EDIT TABLE command.
- If a table or table join is deleted from a prompted query, QMF reevaluates the remaining joins to determine whether remaining tables are still connected (or joined):
  - If so, all remaining joins are left in the query.
  - If not, the joins left are only for the tables that are connected to the first table selected for the query. The **Join Tables** panel is displayed to prompt you to build any remaining joins for the other tables.

DESCRIBE

Use the DESCRIBE command to display information about tables, views, columns of tables or views, or objects that are saved to the QMF catalog (QUERY, PROC, FORM, FOLDER, or ANALYTIC objects). To issue the command, press the Describe key from an object list panel or a Prompted Query panel. You cannot enter the DESCRIBE command on the command line.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

The amount of information that is shown is based on the type of object.

When you press the Describe key for a table, the information on the Table Description panel includes the subtype of the table. The subtype can be alias, history table, table, or view.

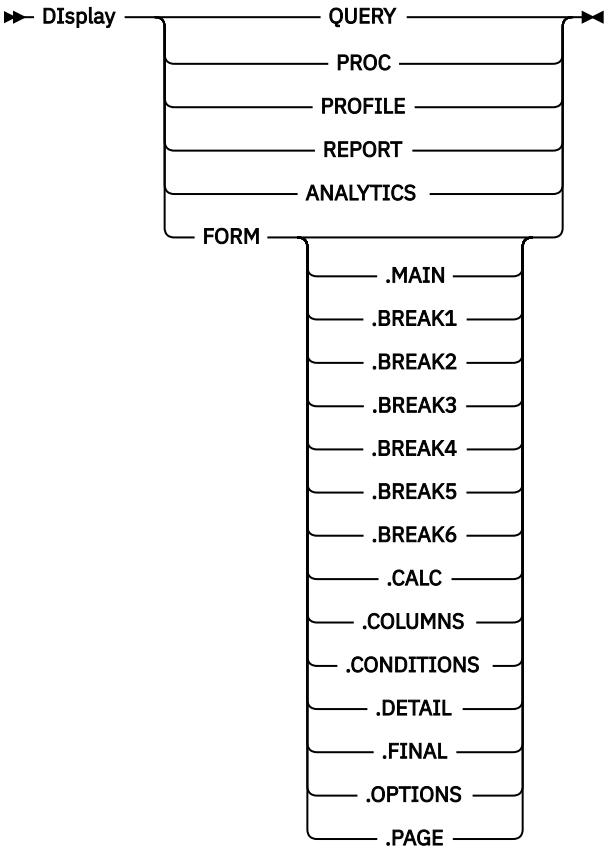
DISPLAY

The DISPLAY command displays an object from QMF temporary storage or an object from the database.

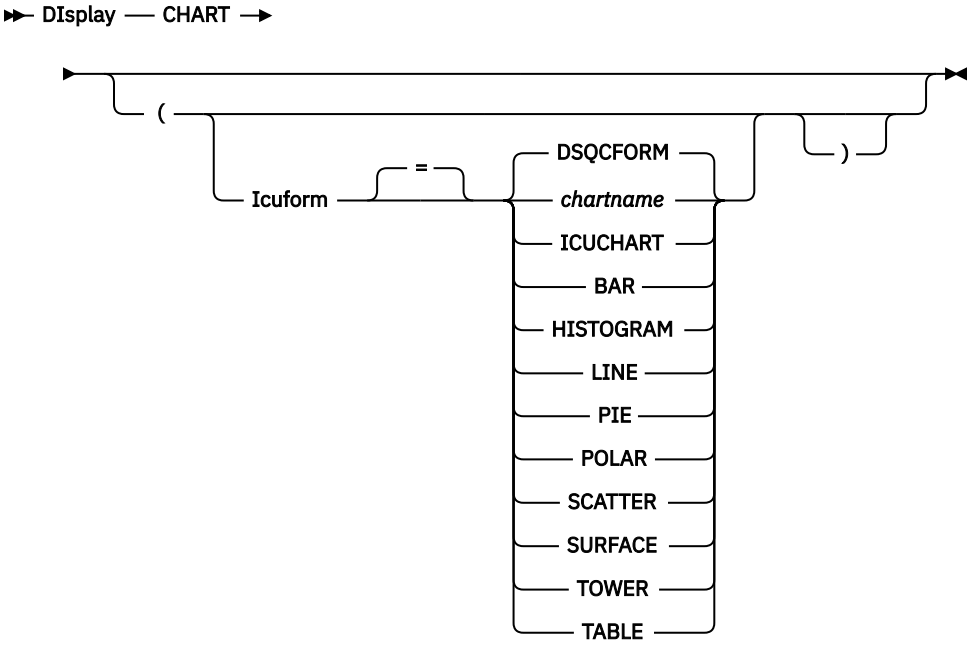
TSO with ISPF	TSO without ISPF	CICS
X	X	X

Syntax

Display a QMF object in temporary storage



Display a CHART



Display an object from the database

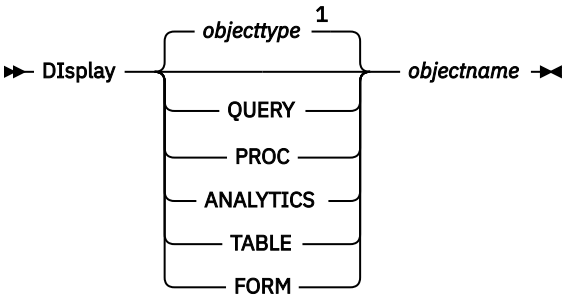
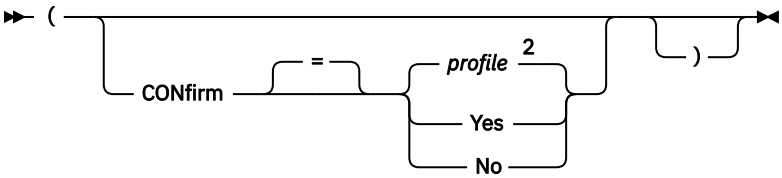


TABLE options



Notes:

[1] The type of the named object, if appropriate, is used. QMF objects have priority over other types of objects (such as database objects).
[2] The value set in your profile is used.

Description

objectname
The name of an object in the database. Valid objects include:

- QMF objects (PROC, QUERY, FORM, ANALYTICS)
- Table objects (TABLE, VIEW, SYNONYM, ALIAS)

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

ICUFORM
Indicates the chart format to use with the GDDM Interactive Chart Utility (ICU). QMF provides several ready-to-use chart styles.

DSQCFORM
The name of the default chart format that is provided by QMF. Unless customized by your administrator, this option provides a bar-style chart.

ICUCHART
The name of the default chart format that is provided by the ICU.

chartname
Indicates the name of a saved chart format that is previously saved in the ICU.

TABLE options:

CONFIRM
Indicates whether a confirmation panel is displayed when the estimated resource to complete the command exceeds the allocated resource that is defined in the Db2 resource limit facility, which provides governing functions. There is also a CONFIRM option in the SET PROFILE command.

If the DISPLAY TABLE command is directed to a Unicode database and the table contains columns that have graphic data types, QMF casts the data to other types to avoid errors.

Usage notes

- A QMF administrator can display any QMF object that is saved in the database.
- If the named object is not a table, it replaces the contents of the same object in the QMF temporary storage area.

  If the named object is a table, it replaces the contents of the QMF data object and the QMF form object in temporary storage. A new FORM is created to match the data in the table. This form provides default formatting for the displayed report.

  You can override the default formatting by setting the following global variables:

  DSQDC_EC_DATE
  DSQDC_EC_TIME
  DSQDC_EC_CHAR
  DSQDC_EC_NUM
  DSQDC_EC_DEC
- You can display the QMF Analytics for TSO Home panel by running DISPLAY ANALYTICS. The DISPLAY command for ANALYTIC objects is supported in TSO only. It is not supported in CICS.
- Running the DISPLAY command with the ANALYTICS option runs the saved ANALYTIC object (the specification) with the current QMF DATA. The resulting chart or statistical graph is displayed within QMF Analytics for TSO. To display the Parameter Selection panel that you used to define the parameters for the chart or statistical analysis, press the Parameters key.
- You can display tables that are owned by other users if you are authorized to do so. Use the owner qualifier to display tables that are owned by another user.
- If your current database location is a Db2 for z/OS server, you can display a table from a remote location by specifying a three-part name for the table.

  If your database administrator set up QMF to use the multirow fetch feature, both databases you are working with (local and remote) must be Db2 for z/OS; otherwise, your command fails. Your database administrator can turn off multirow fetch.

  QMF commands with three-part names cannot be directed to DB2 for VSE and VM databases. Additionally, you cannot access data at a remote location if you start QMF as a stored procedure.

  By default, three-part names cannot be used to access remote tables that contain LOB data. However, you can set the DSQEC_LOB_RETRV global variable to 2 or 3 to access LOB metadata or data with a three-part name. Or, you can use the CONNECT command to connect to the database, and then run the query to access the remote table.
- You can use the DISPLAY TABLE command to display data from a QMF Data Service server. Use a three part name format and ensure that the DSQEC_DS_SUPPORT global variable is set.
- The SHOW command is similar to the DISPLAY command. The difference is:

  SHOW
  Shows object panels, global variables, and certain parts of panels in QMF temporary storage.

  DISPLAY
  Displays both QMF objects and database objects.
- You can modify a displayed SQL query, form, or procedure with the Insert and Delete function keys. You can also type over a form's text or data. Save the changed object with the SAVE command.
- If you previously viewed a form panel, DISPLAY FORM displays the last form panel that you viewed. If you did not display any part of the current form, DISPLAY FORM displays FORM.MAIN.
- When you use DISPLAY CHART, the contents of DATA as formatted by FORM are displayed. The data can be further formatted by the Interactive Chart Utility (ICU) to represent report data graphically. To display a chart, you must have a graphics display device.
- After you work on a chart in the ICU and exit, the QMF panel on which you entered the DISPLAY CHART command is displayed again. If you want to return to a form panel, enter the DISPLAY CHART command from that form panel.

- If you enter CHART on the DISPLAY command prompt, the DISPLAY CHART command prompt appears so that you can specify the parameters that are needed to display your chart.
- If you are displaying a report or chart and the form is incompatible with the data or contains errors, the first form panel that contains an error is displayed with the error highlighted. You must correct the first error that is displayed, and then reissue the CHECK command, or try to redisplay the report or chart to see the next error.
- QMF formats the data in the resulting report according to options specified in QMF forms. Edit codes control how data of different types is displayed. The M edit code is used for metadata and displays the data type and length of the data instead of the data itself.

  If your hardware does not support decimal floating-point instructions, QMF assigns an edit code of M by default to any columns containing decimal floating-point data. You cannot change this edit code.

  QMF also assigns an edit code of M by default to any columns containing XML, binary (BINARY or VARBINARY), or LOB (BLOB, CLOB, or DBCLOB) data. Depending on the data type, you can change the default edit code from M to another edit code to display the actual data. The ability to change the edit code for LOB data is controlled by the value of the DSQEC_LOB_RETRV global variable. This global variable can also be set to display LOB data instead of metadata by default.

  To display XML or LOB data that is longer than the column width, specify edit codes that allow column wrapping, as follows:

  – For XML or CLOB data, set the column width on FORM.MAIN or FORM.COLUMNS to a value of up to 32767 and specify the CW edit code.
  – For BLOB data, set the column width on FORM.MAIN or FORM.COLUMNS to a value of up to 32767 and specify the BW or XW edit code.
  – For DBCLOB data, set the column width on FORM.MAIN or FORM.COLUMNS to a value of up to 16383 and specify the GW edit code.

  If you are working with XML or LOB data and you receive out-of-storage errors while using an edit code other than M, you can change the edit code to M to clear the error and display the report.
- You can display XML data only when you are connected to a database release that supports the XML data type.
- The maximum length of a data row that can be displayed in a QMF report depends on how the DSQEC_TWO_GB_ROW global variable is set:

  – When the global variable is set to 1, the maximum length of a data row in the report is 2 GB.
  – When the global variable is set to 0, row length is limited to 32 KB unless the report contains an XML or LOB column.

    Regardless of the DSQEC_TWO_GB_ROW setting, up to 2 GB of XML, CLOB, and BLOB data, and up to 1 GB of DBCLOB data can be displayed. However, the maximum length of a LOB row can be restricted by the DSQEC_LOB_COLMAX global variable.

    When the table contains LOB or XML columns, the LOB or XML data is not stored as part of the record.

  Regardless of how the DSQEC_TWO_GB_ROW global variable is set, a single table cannot have a maximum record size that is greater than the page size. Because Db2 stores records within pages that are 4 KB, 8 KB, 16 KB, or 32 KB in size, the maximum length of a data row that can be displayed remains at 32 KB when you display a single table. If you display a view that joins two or more tables, the row length can be up to 2 GB.
- You cannot use the DISPLAY CHART command to chart data or tables that contain columns that are defined as BINARY, VARBINARY, or XML. To use this command to display tables that contain DECFLOAT data, the processor on which QMF is running must support decimal floating-point instructions.
- When you issue a DISPLAY TABLE command that references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. You set the value of this register using the SET CURRENT SCHEMA statement.

- QMF updates the **Last Used** field for the object when you use this command. This field appears on object list panels that are displayed by the LIST command. You can change the list of commands that cause the field to be updated by setting the DSQEC_LAST_RUN global variable.

Examples

1. To present a prompt panel for the QMF DISPLAY command:

```
DISPLAY ?
```

2. To display the current QMF procedure object:

```
DISPLAY PROC
```

3. To display a shared QMF query that is called MONTHLY owned by a user named JANET:

```
DISPLAY QUERY JANET.MONTHLY
```

4. If your current location is a Db2 for z/OS server, and you want to display a table that is called VISION owned by a user named JOHNSON at a remote database named BOISE, enter the following command:

```
DISPLAY TABLE BOISE.JOHNSON.VISION
```

   QMF commands with three-part names cannot be directed to DB2 for VM or VSE databases, nor can data be accessed remotely if you start QMF as a stored procedure.

5. This example shows how to enter a DISPLAY command in a QMF procedure when the table named on the DISPLAY command must span multiple lines:

```
PROC                                                                    MODIFIED LINE
1

DISPLAY TABLE
+"LOCATION12345678"."LONGOWNERID12345678911234567892134567893123456789412 3
+456712345678911234567892123456789312345678941234678951234567896123456789 7
+12345"."LONGNAME12345678911234567892123456789312345678941234567895123456 7
+896123456789712345678911234567892123456789312345678941234 5"
```

Figure 5. Entering an object name that spans multiple lines in a linear procedure

Related concepts

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.

How QMF recasts certain data types when displaying data
When a DISPLAY TABLE command is directed to a Unicode database and the table referenced in the command contains columns with graphic data types, QMF converts the graphic data types to character data types.

Related reference

Data types for which QMF displays column metadata
If the column is not null, you can use the M edit code to display the metadata for the column (its data type and length) rather than the actual data.

SET PROFILE
The SET PROFILE command changes values in your QMF profile. These values influence the behavior of your QMF session.

SET special register
You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

Global variables that control how commands and procedures are executed

DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

DPRE

DPRE is a command synonym that provides a print preview so that you can see how a report will appear when it is printed.

TSO with ISPF	TSO without ISPF	CICS
X		

When you issue the DPRE command synonym, QMF runs a REXX exec named DSQA*n*R1C (where *n* is a one-character national language identifier that depends on the language in which QMF is running). DSQA*n*R1C calls an associated exec named DSQABR1C. These applications are shared for use by everyone. When you issue the DPRE command, QMF completes the report and prints the report to a data set that is allocated to DSQPRINT. The ISPF browser is then called to view this data set. In the ISPF browser, you can use the FIND command to quickly navigate to a specific character string in the report.

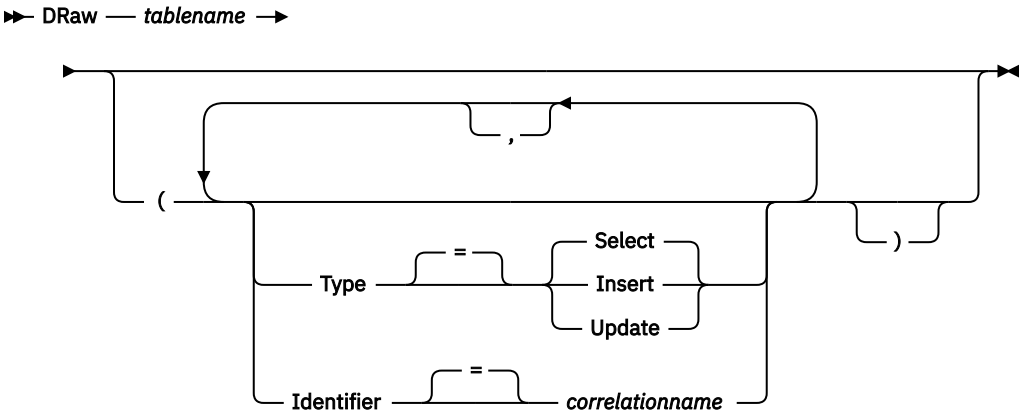
If you are using an NLF, issue the translated command synonym for DPRE. For example, the German command synonym for DPRE is AGB. For the translated command synonym for DPRE in the other language environments, see the Q.COMMAND_SYNONYM_*n* control table, where *n* is the one-character language identifier for the language in which you are using QMF.

DRAW

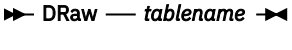
The DRAW command helps you compose a basic SQL query or QBE query.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

DRaw an SQL query



DRaw a QBE query



Description

tablename

The name of a table in the database.

This can be the name of a TABLE, VIEW, SYNONYM, or ALIAS.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.To display the panel, issue the following command:

```
DRAW ?
```

TYPE
The type of query you want to compose.

SELECT
Composes a basic query for selecting data from the columns of a table or view. Type the other clauses you need when the query is displayed. To select more than one table, use the DRAW command for each table. This is the default query type.

INSERT
Composes a basic query for inserting data into a table or view. When the query is displayed, type the new data to the left of the column names.

UPDATE
Composes a basic query to change the values of specified rows of a table or view. When the query is displayed, type your changes to the right of the column names and delete the lines you do not need.

IDENTIFIER
Specifies an identifier to uniquely designate the table in the composed query. This option is ignored when TYPE=INSERT.

When you join tables, use this option to identify which columns in the composed query come from each of the joined tables. The query will not run if the tables have common column names that are not identified.

correlationname
A user-defined name that becomes a correlation name for the table in the composed query. This name is used to qualify columns in the query to avoid ambiguity or to establish a correlated reference for subqueries. It can also be used merely as a better name for the table to improve readability of the query.

If you do not specify this option, no correlation name is added to the composed query.

Usage notes

- The DRAW command is valid only on an SQL query or a QBE query panel.
- If you issue the DRAW command when a SELECT statement is already on the query panel, QMF joins the newly specified tables to the first. Use the IDENTIFIER option whenever adding another table to an existing SQL SELECT query.
- Some queries require additional information before they can be run.
- You can draw a table or view at another location by including a location qualifier for the table name.
- You can use the DRAW command to draw a table from a QMF Data Service server. Use a three part name format and ensure that the DSQEC_DS_SUPPORT global variable is set.
- If you issue the DRAW command with the UPDATE option on an SQL query panel, columns that were defined with the AS ROW BEGIN, AS ROW END, or AS TRANSACTION START ID attributes are excluded from the column list. If you issue the DRAW command with the INSERT option on an SQL query panel, a value of DEFAULT is generated for columns that were defined with the AS ROW BEGIN, AS ROW END, or AS TRANSACTION START ID attributes.
- When you issue a DRAW command that references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. QMF allows you to set the value of this register using the SET CURRENT SCHEMA statement.

Examples

1. To draw a SELECT query for the table Q.STAFF uniquely identified by S:

```
DRAW Q.STAFF (TYPE=SELECT IDENTIFIER=S
```

   Here is the result:

```
SELECT S.ID, S."NAME", S.DEPT, S.JOB, S."YEARS"
     , S.SALARY, S.COMM
FROM Q.STAFF S
```

2. If your table names or column names contain any of the following, the DRAW command surrounds the names with double quotation marks:

   - Special characters
   - QMF reserved words
   - IBM SQL reserved words
   - Db2 reserved words

   For example, suppose a table called MYTABLE contains special characters or reserved words and you issue the command DRAW MYTABLE.

   Here is the result:

```
SELECT NORMALNAME, KEYWORDFOLLOWS, "UNION"
     , "HAS BLANKS IN IT", "SPECIAL+CHARS_IN!"
     ,  "Mixed_Case_%S" FROM USER.MYTABLE
```

3. To join two tables, issue the DRAW command twice in succession, once for each table. Be sure to use the IDENTIFIER parameter to identify which columns in the composed query are associated with each table.

```
DRAW Q.ORG (I=ORG
DRAW Q.STAFF (I=STAFF
```

   QMF displays the query, which joins the Q.ORG and Q.STAFF tables:

```
SELECT ORG.DEPTNUMB, ORG.DEPTNAME, ORG.MANAGER
      , ORG.DIVISION, ORG.LOCATION

      , STAFF.ID, STAFF."NAME", STAFF.DEPT, STAFF.JOB
      , STAFF."YEARS", STAFF.SALARY, STAFF.COMM
 FROM Q.ORG ORG

    , Q.STAFF STAFF
```

Related reference

SET special register
You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

EDIT *object*

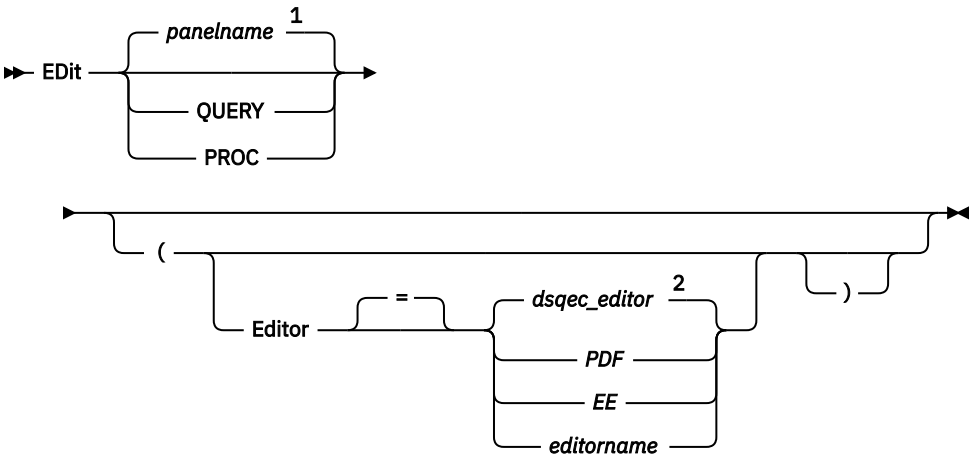
Use the EDIT *object* command to modify QMF objects by using an external editor.

You can use the EDIT command to modify the following types of QMF objects:

- A QMF procedure currently in temporary storage
- A SQL query currently in temporary storage
- A QMF procedure currently in the database
- A SQL query currently in the database

TSO with ISPF	TSO without ISPF	CICS
X	*	

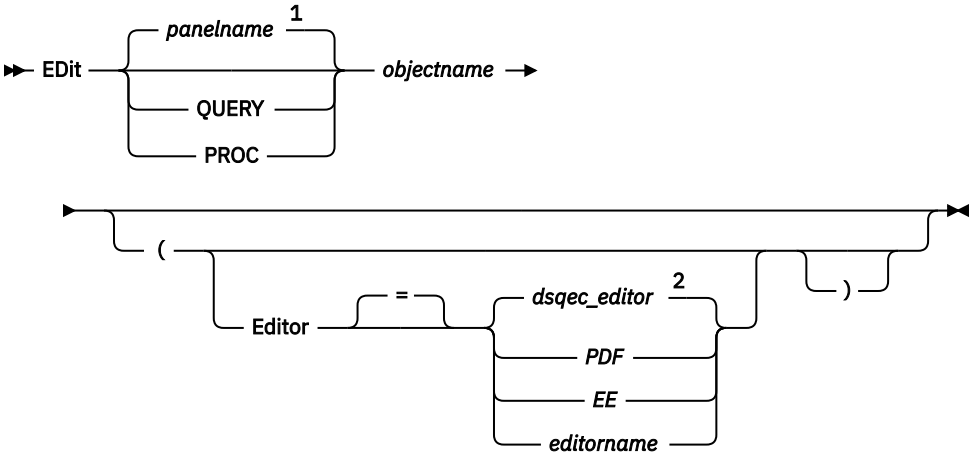
EDIT a QMF SQL query or procedure from temporary storage



Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The value set in this global variable is used.

EDIT a QMF SQL query or procedure from the database



Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The value set in this global variable is used.

Description

objectname* and *panelname
If you are running an object directly from the database, *objectname* names a query or procedure in the database. If you are running an object from temporary storage, *panelname* names a QMF object panel.

EDITOR
Specifies the name of the editor used to edit your QMF procedure or SQL query.

dsqec_editor
Specifies the value of the DSQEC_EDITOR global variable, which you use to set the default editor. Initially, DSQEC_EDITOR is set to blank, which defaults to PDF.

PDF
Specifies that the ISPF/PDF editor is used to edit the procedure or query. To use the PDF editor to edit a query or procedure, start QMF as an ISPF dialog.

EE
Specifies that the QMF Enhanced Editor is used to edit the procedure or query.

editorname
The name of any other editor available to you. It can also be the name of a CLIST that starts an editor. For more information about available editors, see your administrator.

Usage notes

- If you want to build a new query or procedure using the EDIT command, reset the query or procedure first to clear the QMF temporary storage area. Do this by issuing the RESET command with the QUERY or PROC parameter.
- Use the following methods to modify an existing query or procedure:
  - First, display the query or procedure to bring it into the QMF temporary storage area. Then, use the EDIT command to modify the query or procedure.
  - Use the EDIT command, and specify the name of the query or procedure to modify.
- After editing your query or procedure, you can save it, replacing whatever was in QMF temporary storage. If your query or procedure is too large to fit in QMF's temporary storage area, it is stored in a data set. If this happens, a message is displayed telling you the name of the data set that your procedure or query is in.
- The SAVE command within the editor is not the same as the QMF SAVE command. The editor only saves to the QMF temporary storage area. If you want the query or procedure to be saved in the database, you must use the QMF SAVE command.
- Although you cannot use the EDIT command in CICS to edit a QMF query or procedure, you can use the QMF DISPLAY or SHOW command to display such an object and then modify it using QMF.

Examples

1. To display the EDIT command prompt panel:

   ```
   EDIT ?
   ```

2. To export the current query and place it in the ISPF/PDF editor:

   ```
   EDIT QUERY
   ```

   When the edit session ends, the edited data set is imported to the QUERY temporary storage area.

   To use the PDF editor, start QMF as an ISPF dialog.

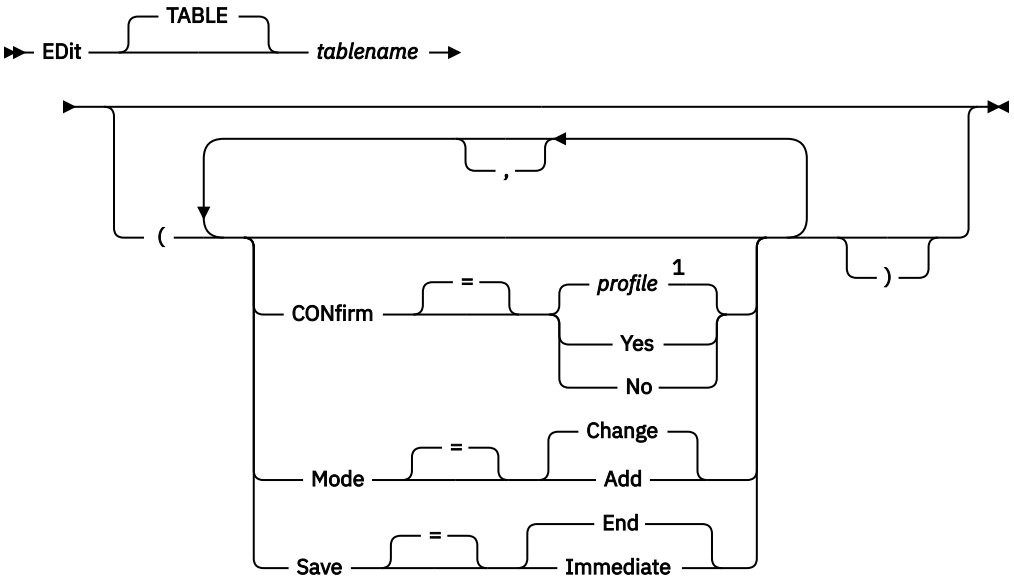
EDIT TABLE

The EDIT TABLE command invokes the QMF Table Editor. During a Table Editor session, you can make additions, changes, or deletions to records in your table using fields on the panels provided.

TSO with ISPF	TSO without ISPF	CICS
X	X	*

Issue the END command to exit a Table Editor session.

EDIT a table



Notes:

[1] The value set in your profile is used.

Description

tablename
The name of a table in the database.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

MODE
The type of Table Editor session to run.

CHANGE
Operates the Table Editor in a mode permitting rows in the table to be changed. Change mode includes the ability to:

- Search for rows
- View data in a row
- Update columns in a row
- Delete a row
- Advance through a set of rows

ADD
Operates the Table Editor in a mode permitting new records to be added to the table.

SAVE
Specifies when to commit changes and deletions made during a Table Editor session.

IMMEDIATE
Changes made during the edit session are processed individually for each row. This choice increases the availability of the table to other users while your edit session is active.

END
All changes made during the edit session are held until the session ends. You have an opportunity to cancel all changes at any time. This choice decreases the availability of the table to other users as your edit session progresses.

CONFIRM
Indicates whether confirmation panels are displayed during the Table Editor session.

There are confirmation panels for these session events:

- Adding a row
- Changing a row
- Deleting a row
- Typed entries about to be lost
- Session end

Usage notes

- You cannot use the Table Editor to edit a table containing BINARY, VARBINARY, BLOB, or XML data. If the table contains columns with CLOB or DBCLOB data types, those columns cannot be changed, but other columns in the table can be edited. To edit a table that contains DECFLOAT data, the processor on which QMF is running must support decimal floating-point instructions.
- If global variable DSQCP_RMV_BLANKS is set to 1, the Table Editor strips out trailing blanks in CHANGE mode for VARCHAR or VARGRAPHIC columns. If a VARCHAR or VARGRAPHIC column contains only blanks after updating, the length of this column will be zero.
- QMF provides a set of global variables to individually control the activation of the various edit session confirmation panels.
- The Table Editor supports null and default values with specially reserved characters. You can alter the definition of these reserved characters prior to the edit session by changing the values of global variables.
- When you issue an EDIT TABLE command that references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. QMF allows you to set the value of this register using the SET CURRENT SCHEMA statement.

Examples

1. To display a prompt panel for the QMF EDIT TABLE command:

```
EDIT TABLE ?
```

2. To add new rows to a table named TABTWO owned by user BILL:

```
EDIT TABLE BILL.TABTWO (MODE=ADD
```

Related reference

SET special register
You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

Global variables associated with the Table Editor
DSQCP global variables are associated with the operations of the Table Editor. All of these global variables can be modified by the SET GLOBAL command.

END

The END command ends the current operation and returns to an earlier state.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

The result of the END command varies depending on what panel you are using and whether an initial procedure is executing:

- If you enter END (or press the End function key) from the QMF home panel, yourQMF session ends.
- If you enter END (or press the End function key) from any of the following QMF panels, the QMF home panel is displayed:

```
QUERY       FORM.MAIN      FORM.COLUMNS
PROC        FORM.CALC      FORM.OPTIONS
PROFILE     FORM.DETAIL    FORM.BREAK.n
REPORT      FORM.FINAL     FORM.CONDITIONS
            FORM.PAGE      Global variable list
```

- From a prompt panel, the panel on which you issued the command that caused the prompt is displayed. (This could be the QMF home panel or the panel for FORM, PROC, PROFILE, QUERY, or REPORT.)

  If you press the End function key after making an entry on the prompt panel and before pressing Enter, the entry you made is not processed.
- From a Table Editor panel, your changes are committed and the panel from which you called the Table Editor is displayed.

  When you press the End function key from a Table Editor panel, a confirmation panel is displayed so you can decide whether to end (commit your changes to the database) or not (return to the Table Editor panels).

The END command does not work as described above in the following situations:

- If QMF was started with an initial procedure, END reruns the initial procedure without displaying the QMF home panel.
- If the current panel is the QMF home panel and END is issued through the QMF command or callable interface, the QMF session is not terminated immediately. Instead, the CLIST or program containing the END command regains control. In this case, the QMF session is not terminated until the CLIST or program ends.
- If END is issued from a new interactive session that was started by the INTERACT command, control is returned to the application or procedure from which the INTERACT command was issued. In this case, END does not terminate the session or display the QMF home panel.
- If the END command is issued from a new interactive session that was started as the result of issuing a command on the database object list panel, the database object list is displayed. In this case, END does not terminate the session or display the QMF home panel.

ENLARGE

The ENLARGE command in QMF increases the size of a QBE example table.

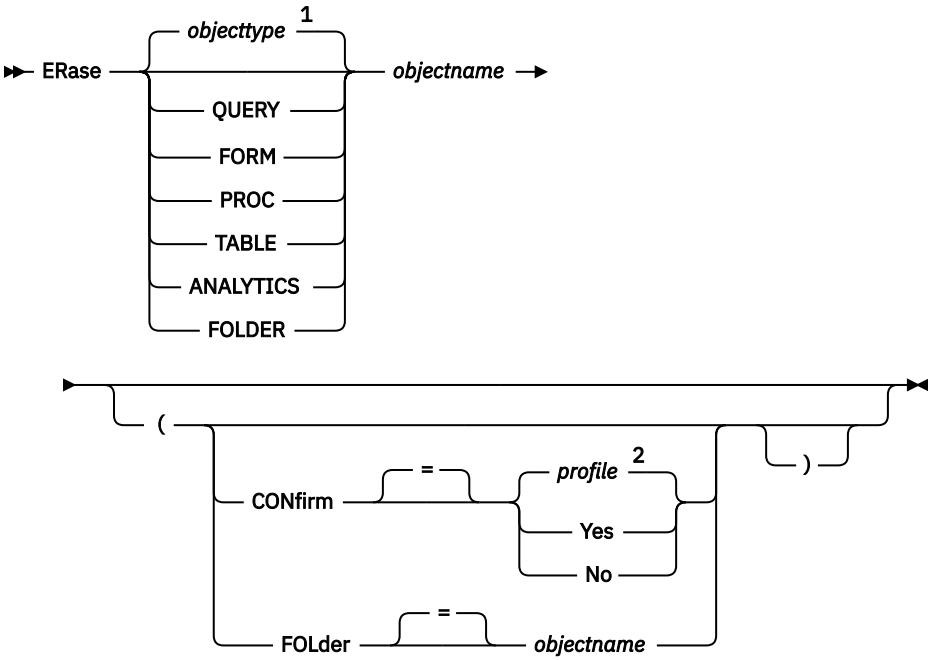
TSO with ISPF	TSO without ISPF	CICS
X	X	X

ERASE

The ERASE command removes an object from the database.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

ERASE an object from the database



Notes:

[1] The type of the named object, if appropriate, is used. QMF objects have priority over other types of objects (such as database objects).

[2] The value set in your profile is used.

Description

objectname

The name of the QMF object in the database.

When you specify the name of a FORM object, all parts of the form are erased.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

CONFIRM

Whether or not a confirmation panel is displayed.

YES

Displays a confirmation panel if an object in the database will be removed by this command.

NO

No confirmation panel is displayed.

FOLDER

The name of the QMF folder object to use with the ERASE command.

You can erase a QMF object from a folder by using the FOLDER keyword with the ERASE command. When you erase a QMF object from a folder, the QMF object is erased from the folder only; the QMF object itself is not erased.

You can specify a folder name either by including the FOLDER keyword in the ERASE command or by setting the DSQEC_CURR_FOLDER global variable:

- If the FOLDER keyword is specified with the ERASE command, that folder name overrides the folder name that is set in DSQEC_CURR_FOLDER.
- If the FOLDER keyword is not specified with the ERASE command and DSQEC_CURR_FOLDER is set to a folder name, the object is removed from the folder that is specified by DSQEC_CURR_FOLDER.

- If the FOLDER keyword is not specified and DSQEC_CURR_FOLDER is not set, the object itself is erased.

The folder name must be a valid QMF object name. The folder name cannot be a QMF object type, such as QUERY, PROC, FORM, ANALYTIC, or FOLDER. Wildcards '%' and '_' are not valid in a folder name. If the folder name includes a blank, the folder name must be enclosed in double quotation marks.

The FOLDER keyword is not valid when you are connected to a DB2 Server for VSE and VM database.

Usage notes

- Objects can be erased only from the current database location. You cannot erase a remote table by using a three-part name. Instead, first connect to the location where the table is located, then issue the ERASE command. You cannot connect to a remote location if you have started QMF as a stored procedure.
- If you specify an object name that does not exist, no warning message is issued from a linear procedure.
- When you issue an ERASE TABLE command that references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. QMF allows you to set the value of this register using the SET CURRENT SCHEMA statement.
- When a QMF query, procedure, form, or analytics object is erased, that object is also removed from any folder object that references it.

Examples

1. To display a command prompt panel:

```
ERASE ?
```

2. To erase the table PATTI.TABLEONE:

```
ERASE TABLE PATTI.TABLEONE
```

3. To erase a query named JBQUERY and display a confirmation panel:

```
ERASE JBQUERY (CONFIRM=YES
```

4. To erase the table PATTI.TABLETWO at the DALLAS location while your local location is BOISE, you must first connect to DALLAS:

```
CONNECT TO DALLAS
```

Then issue the ERASE command:

```
ERASE TABLE PATTI.TABLETWO
```

5. To erase a query named MYQUERY from a folder named SALES but not erase the query itself:

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

6. When using the ERASE command in a QMF procedure, you must use double quotation marks to continue an authorization ID across more than one line within a QMF linear procedure. All continuation lines must have a plus (+) sign in column one, as shown in the following figure:

```
PROC                                                          MODIFIED LINE   1
 ERASE QUERY
 +"LOCATION12345678”.”LONGOWNERID12345678901234567890123456789012345678901234567
 +90123456789012345678901234567890123456789012345678901234567890123456789012345678”.”LONGN
 +AME01234567890123456789012345678901234567890123456789012345678901234567890123456789012345
 +56789012345678901234567890123456789012345678”
```

Figure 6. Continuing an authorization ID across more than one line using the ERASE command

Related reference

SET special register
You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

EXIT

The EXIT command stops your QMF session.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

EXIt

You can issue the command on the QMF home panel, on the QUERY, REPORT, FORM, PROFILE, or global variable list panel, or you can put it in a procedure.

You can also enter the EXIT command from the QMF command area of any object on the QMF database object list panel. You cannot enter the EXIT command on a command prompt, confirmation, or Help panel.

Restriction: If you issue EXIT through the QMF command interface or in a procedure that is run through the command interface, your session is not terminated immediately. Instead, the CLIST or application program that is running from the command interface regains control. Your session is not terminated until the CLIST or application completes.

Related reference

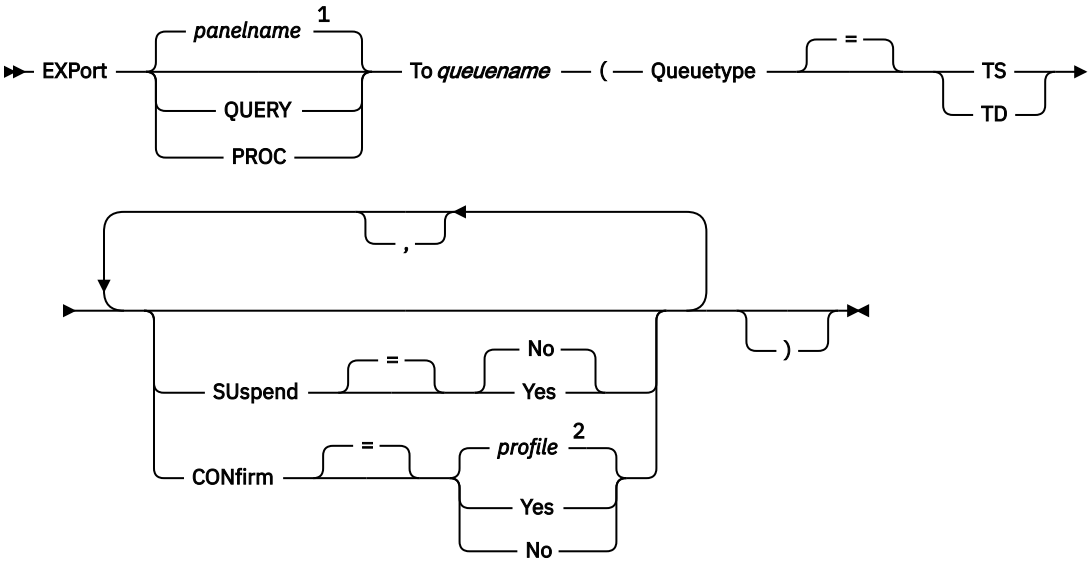
LIST
Use the LIST command to display lists of QMF objects and database tables stored in the database. When you first issue the LIST command in a QMF session, ensure that you use one of these parameters: Queries, Forms, Procs, Analytics, Folders, QMF, Tables, or All.

EXPORT in CICS

Use the EXPORT command in CICS to send queries, forms, procedures, reports, and data from QMF temporary storage to a CICS data queue. You can also use the EXPORT command to send queries, forms, procedures, and tables from the database to a CICS data queue, or to send charts from QMF to a GDDM library that contains GDF files.

The syntax for exporting objects from QMF temporary storage is different from the syntax for exporting objects from the database.

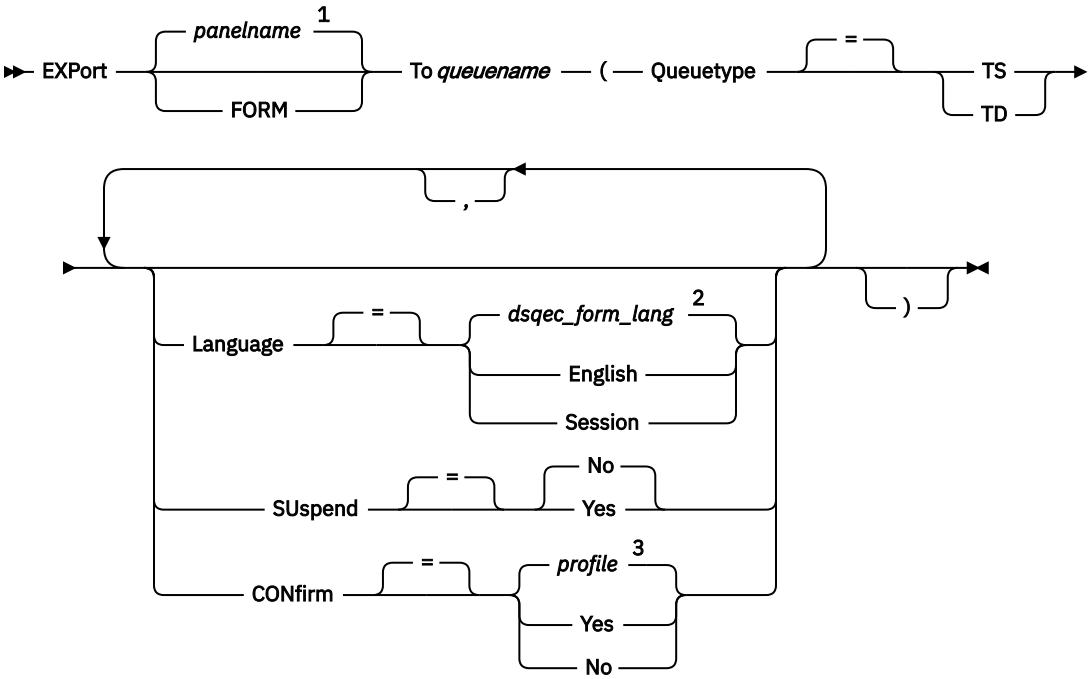
EXPORT a QMF query or procedure from temporary storage



Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The value set in your profile is used.

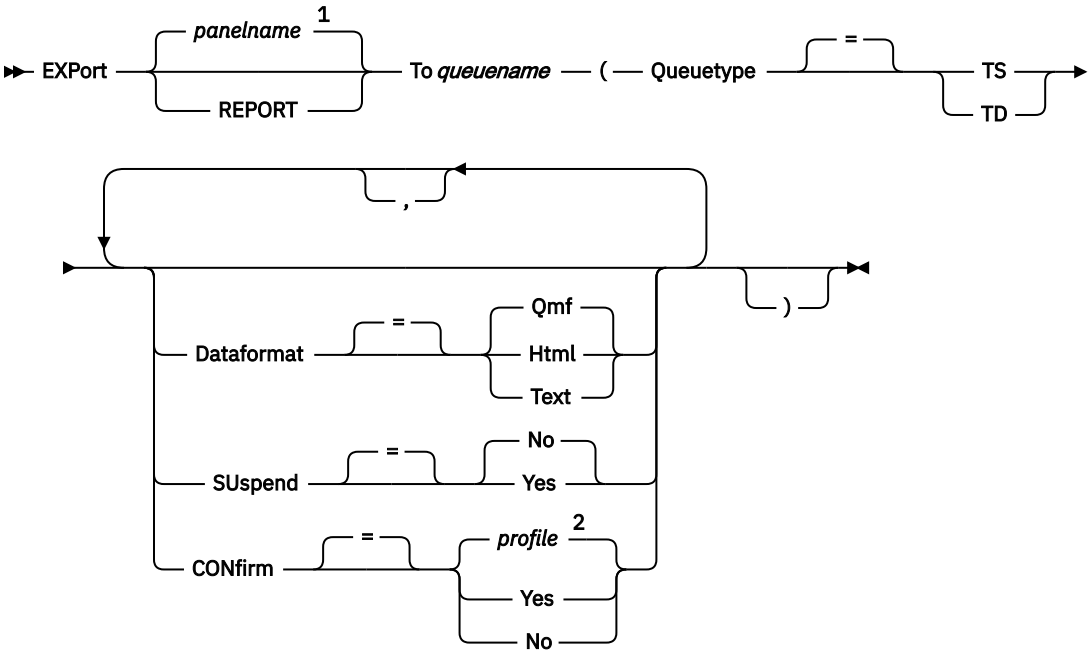
EXPORT a QMF form from temporary storage



Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The value set in this global variable is used.
[3] The value set in your profile is used.

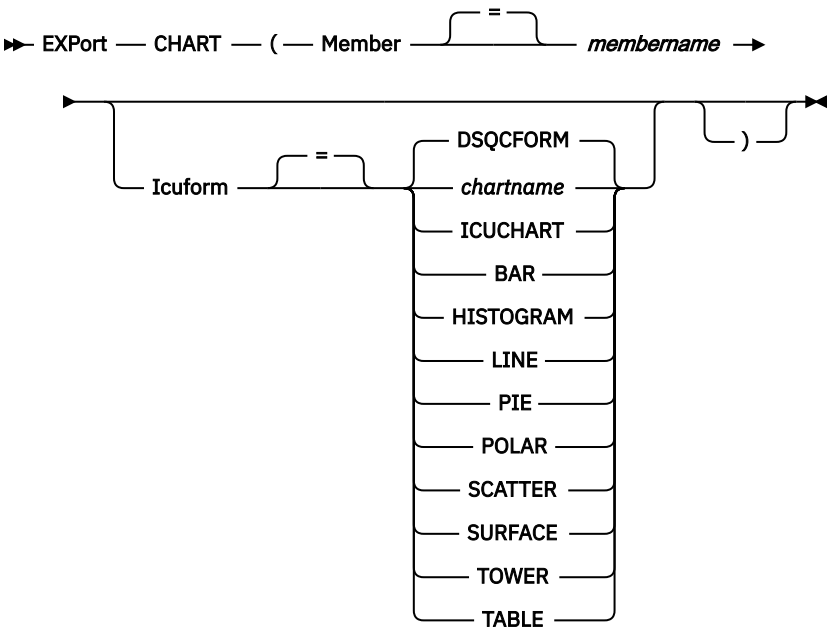
EXPORT a QMF report



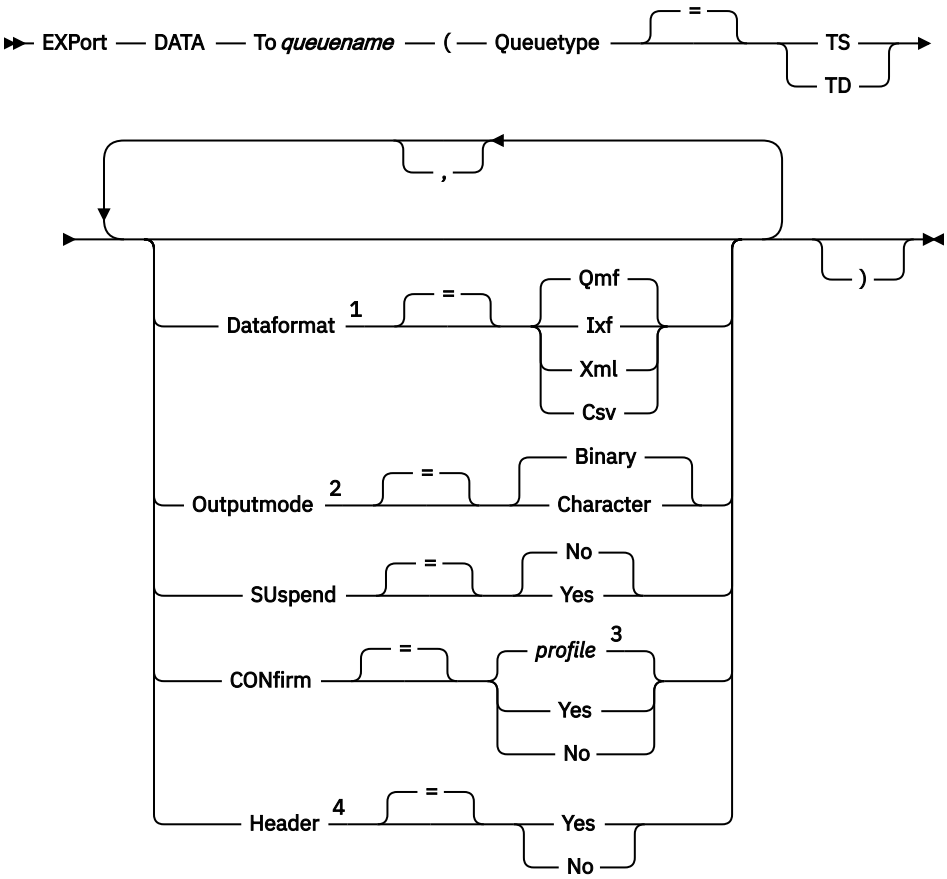
Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The value set in your profile is used.

EXPORT a QMF chart



EXPORT QMF data



Notes:

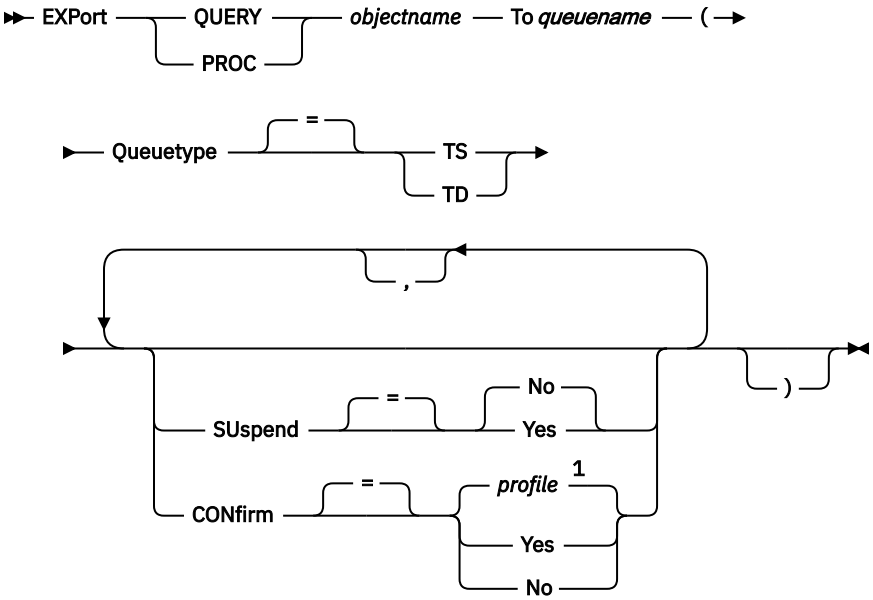
[1] If your data or table contains an XML column or LOB data, you must use the DATAFORMAT=XML clause on the command. This format can also be used when the data or table to be exported does not contain an XML column. Before you can export QMF data in XML format, you must set up the z/OS conversion environment for Unicode support.

[2] Accepted only when DATAFORMAT=IXF.

[3] The value set in your profile is used.

[4] Valid only when DATAFORMAT=CSV.

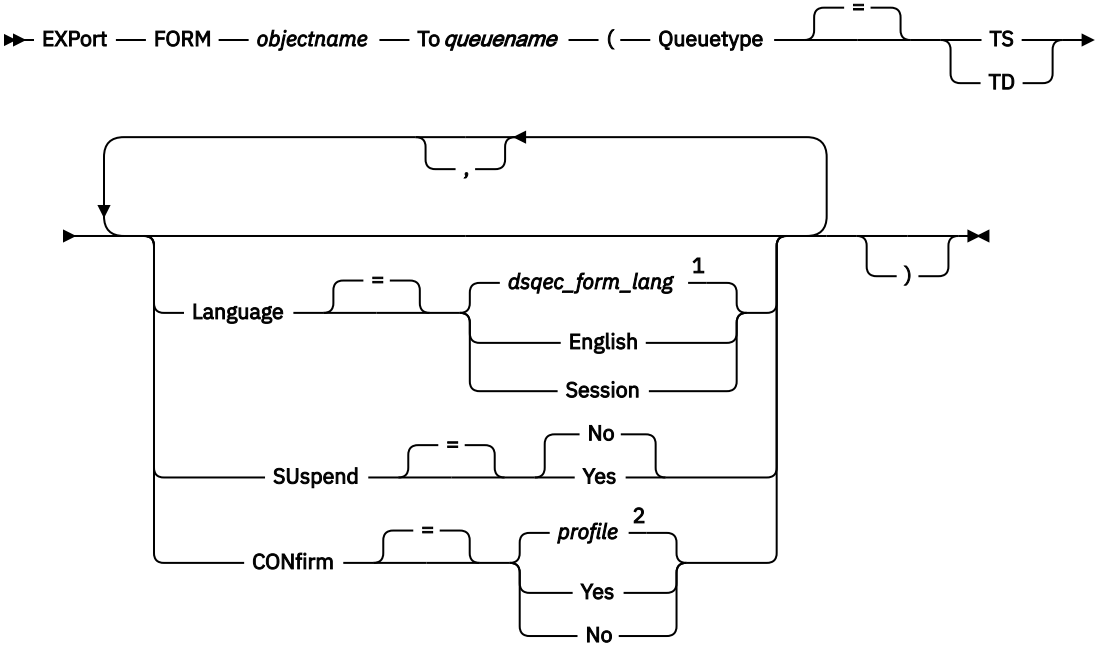
EXPORT a QMF query or procedure from the database



Notes:

[1] The value set in your profile is used.

EXPORT a QMF form from the database

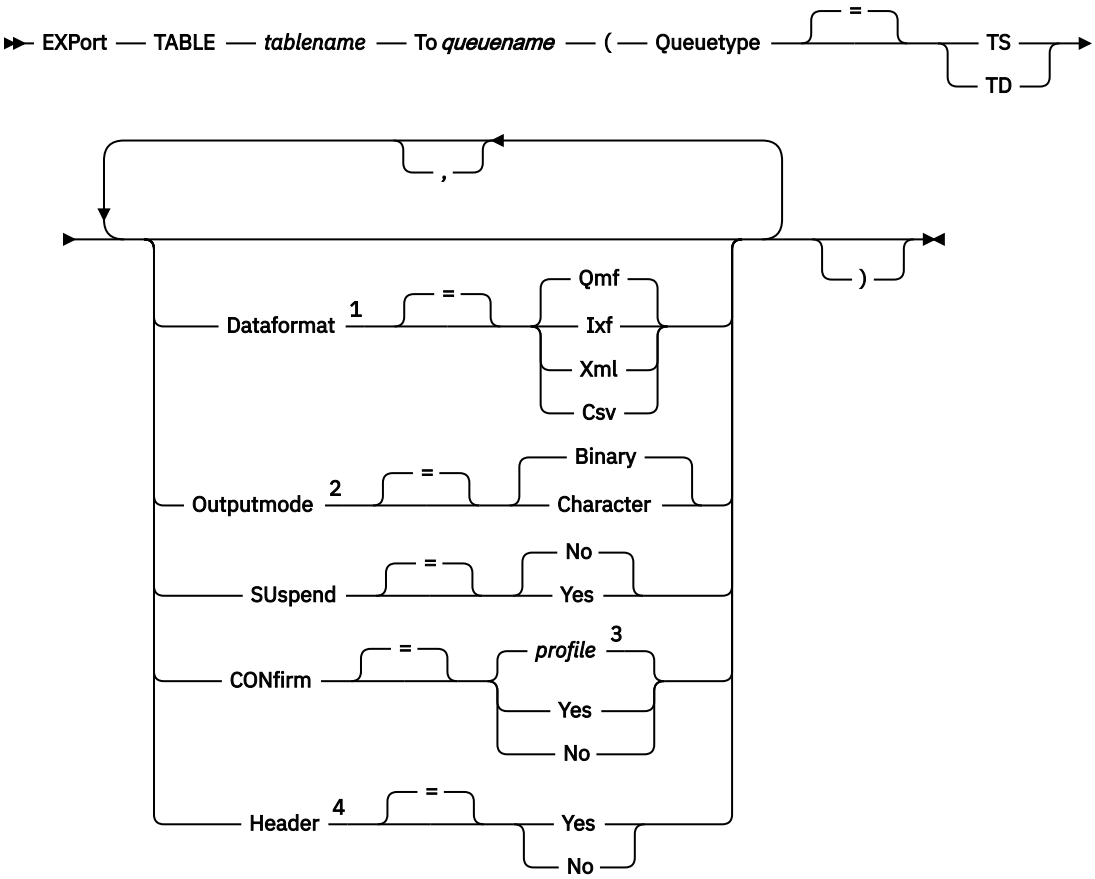


Notes:

[1] The value set in this global variable is used.

[2] The value set in your profile is used.

EXPORT a TABLE from the database



Notes:

[1] If your data or table contains an XML column or LOB data, you must use the DATAFORMAT=XML clause on the command. This format can also be used when the data or table to be exported does not contain an XML column. Before you can export QMF data in XML format, you must set up the z/OS conversion environment for Unicode support.

[2] Accepted only when DATAFORMAT=IXF.

[3] The value set in your profile is used.

[4] Valid only when DATAFORMAT=CSV.

Description

objectname
The name of a QMF object in the database.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

tablename
The name of a table, view, synonym or alias.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

queuename
Names the CICS data queue to receive the exported object. The maximum length of the name is:

- 4 characters when QUEUETYPE is TD.

- 8 characters when QUEUETYPE is TS.

  For a TS queue, surround the name in single quotation marks if it contains special characters, such as a period.

The type of storage for the queue must match the type specified with the QUEUETYPE parameter.

QUEUETYPE
Indicates the type of CICS storage used for the data queue receiving the object. There is no default for QUEUETYPE; it must be specified.

TS
A CICS temporary storage queue

TD
A CICS transient data queue

SUSPEND
Specifies the action to take when the data queue is busy and unavailable.

NO
Cancels the export request.

YES
Waits until the data queue is available.

MEMBER
Indicates that the exported object will be a member in the VSAM data set defined by your QMF environment for GDDM GDF (graphics data format) data. If the member already exists, it will be replaced.

membername
Names the member that receives the exported object. Member names are limited to 8 characters.

CONFIRM
Indicates whether a confirmation panel is displayed when this command will change or replace the data queue. This option is only valid for CICS temporary storage queues (QUEUETYPE=TS).

LANGUAGE
Indicates whether QMF keywords contained within the exported form are recorded in English or in the current NLF session language.

A QMF form containing QMF keywords in English can be used in any QMF session. A QMF form containing QMF keywords in any other national language supported by QMF can be used only in a session of that same national language.

DATAFORMAT
Specifies the file format to be used for the exported object.

QMF
Uses QMF format. This is the default format for exporting a report, the DATA object, or a table. The maximum length of a data row to be exported is 32 KB for this format. You can use 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.

The DSQDC_SHORT_EXPT global variable controls the length of all column name fields in the header records of data or tables exported with a value of QMF on the DATAFORMAT parameter.

HTML
Uses HTML format. This can be used only when exporting a report. The maximum length of a data row to be exported is 32 KB for this format. You can use 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.

TEXT
Exports reports without control information. This option can be used only when you export a report.

IXF
Uses Integrated Exchange Format. This can be used only when exporting the DATA object or a table. The maximum length of a data row to be exported is 32 KB for this format. You can use 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.

XML
Uses 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 format only when exporting a DATA object or a table, and it is the only option when exporting data or tables to a UNIX file.

You must specify this format to export data that is defined as an XML data type. XML data type data can only be exported when you are connected to a database release that supports the XML data type.

You must also specify this format to export data or a table that contains LOB data. Note that the ability to export LOB data might be restricted by the DSQEC_LOB_RETRV and DSQEC_LOB_COLMAX global variables.

The maximum length of a data row to be exported is 2 GB for this format. Ensure that all characters in the XML data to be exported are supported by the XML parser.

Some sample XML files are supplied with QMF. These files allow you to display data in a browser.

CSV
Specifies CSV format. You can use this option only when exporting a data object or table. 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.

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.

ICUFORM
Specifies the name of a chart format. A chart format contains the specifications required to turn data into a chart. Different formats are used to produce different types of charts.

DSQCFORM
The name of the default chart format provided by QMF.

This format can be customized by your administrator. It provides a bar chart if not customized.

chartname
The name of a saved chart format

ICUCHART
Specifies the default chart format for the GDDM Interactive Chart Utility.

BAR
HISTOGRAM
LINE
PIE
POLAR
SCATTER
SURFACE
TOWER
TABLE
The name of a chart format provided by QMF.

HEADER
Specifies whether to include column headings with the exported data. You can specify this option only when the DATAFORMAT=CSV.

YES
Column headings are exported. This is the default setting. If you use this setting, the value of the DSQDC_COL_LABELS global variable controls whether column labels or column names are exported. The default of DSQDC_COL_LABELS is 1, which means that column labels are exported.

NO
Column headings are not exported.

Usage notes

- Use of TSO data sets in CICS is not recommended. However, if you do choose to use TSO data sets, there are additional customization steps required to support both IMPORT and EXPORT commands. TSO data sets referenced by the EXPORT command under CICS must be defined as either partitioned (with a data set organization, or DSORG, value of PO) or physical sequential (DSORG=PS).
- If you export to a transient data queue, the queue must be open, enabled and empty before you issue the EXPORT command.
- If the specified CICS data queue already exists, its contents are replaced with the exported object.
- An empty or partial CICS data queue can result if there is an error in the execution of the EXPORT command.
- In some cases, if the object is exported to the same data queue from which the current data was imported, you might receive an Incomplete Data prompt. At the prompt, choose NO and export the object to a different data queue.
- When a form is exported, all parts of the form are exported.

  However, QMF will drop any FORM.DETAIL panel variations that were not modified from their default values. In this manner, unwanted FORM.DETAIL variations can be dropped by exporting and then importing the same form.
- If you are exporting a report or chart and the form is incompatible with the data or contains errors, the first form panel containing an error is displayed with the error highlighted. To see other errors, correct the currently displayed error and press the Check function key.
- To use this command with columns that contain DECFLOAT data, the processor on which QMF is running must support decimal floating-point instructions.
- If you are exporting a table or data to a database using a three-part-name and your database administrator has set up QMF to use the multirow fetch feature, both databases you are working with (local and remote) must be Db2 for z/OS; otherwise, your command will fail. Your database administrator can turn off multirow fetch. QMF commands with three-part names cannot be directed to DB2 for VSE and VM databases.
- You can use the EXPORT TABLE command to export a table from a QMF Data Service server. Use a three part name format and ensure that the DSQEC_DS_SUPPORT global variable is set.
- QMF updates the Last Used field for the object when you use this command. This field appears on object list panels displayed by the LIST command. You can change the list of commands that cause the field to be updated by setting the DSQEC_LAST_RUN global variable.
- If an EXPORT TABLE command is directed to a Unicode database and the table contains columns that have graphic data types, QMF casts the data to other types to avoid errors.
- When you issue an EXPORT TABLE command that references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. QMF allows you to set the value of this register using the SET CURRENT SCHEMA statement.
- When you issue the EXPORT DATA or EXPORT TABLE command, QMF exports either column labels or column names, depending on the value of the DATAFORMAT parameter:

  - When DATAFORMAT=QMF, column names are exported even for columns that have database labels defined.
  - When DATAFORMAT=IXF, labels are exported for any columns that have labels defined. Column names are exported for all other columns.
  - When DATAFORMAT=XML, QMF exports labels for all columns. If a column does not have a label already defined, QMF creates a label from the column name and exports that label.
  - When DATAFORMAT=CSV, column labels or column names are exported in column headings unless HEADER=NO is specified. The DSQDC_COL_LABELS global variable controls whether column labels or column names are exported.
- Although you can export data from temporal tables, you cannot export a temporal table. The data that you export from a temporal table is not associated with history data.

Examples

1. To display a command prompt panel for exporting an object:

```
EXPORT ?
```

2. To export a query from QMF temporary storage to a transient data queue:

```
EXPORT QUERY TO queuename (QUEUETYPE = TD)
```

3. To export data to a transient data queue with a data format of IXF:

```
EXPORT DATA TO queuename (QUEUETYPE=TD
CONFIRM=NO DATAFORMAT=IXF
```

   You can abbreviate the command keywords. For example:

```
EXP DATA TO queuename (QUEUET=TD CONF=N DATA=IXF
```

4. If your current location is Db2 for z/OS you can export a table from a remote Db2 location by including the location qualifier in the object name:

```
EXPORT TABLE VENICE.LARA.STATSTAB
    TO queuename (QUEUETYPE = TS
```

   The table is exported from the database to which you are currently connected.

   QMF commands with three-part names cannot be directed to DB2 for VSE and VM databases.

5. To export a table to a TS queue in IXF character format:

```
EXPORT TABLE KMMTABLE TO MYQUEUE
 (QUEUETYPE=TS DATAFORMAT=IXF OUTPUTMODE=CHARACTER
```

6. To export data in CSV format without column headings:

```
EXPORT DATA TO MYDATA
 (DATAFORMAT=CSV HEADER=NO
```

Related concepts

How QMF recasts certain data types when displaying data
When a DISPLAY TABLE command is directed to a Unicode database and the table referenced in the command contains columns with graphic data types, QMF converts the graphic data types to character data types.

Related reference

SET special register
You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

Global variables that control how commands and procedures are executed

DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

Global variables that control various displays
DSQDC global variables control the display of certain kinds of information. All of these global variables can be modified by the SET GLOBAL command.

Related information

Search for information about unsupported characters by referring to the XML Toolkit for z/OS User's Guide.

EXPORT in TSO

The EXPORT command sends queries, forms, procedures, reports, tables, data, and charts to certain data sets and files.

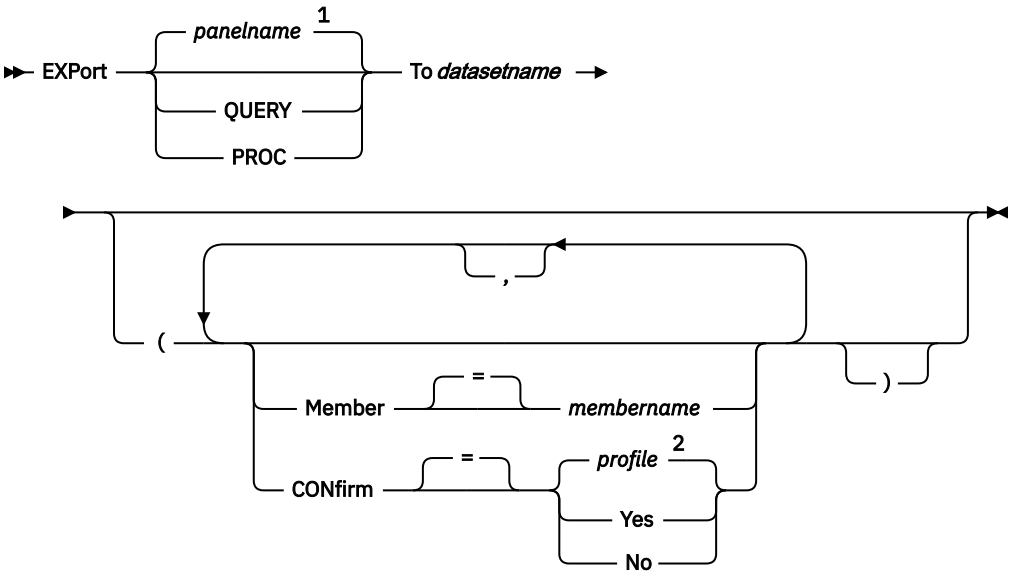
TSO with ISPF	TSO without ISPF
X	X

Specifically, the EXPORT command sends:

- Queries, forms, procedures, reports, and data from QMF temporary storage to a TSO data set
- Queries, forms, procedures, and tables from the database to a TSO data set
- Charts from QMF to a GDDM-partitioned data set that contains GDF files
- HTML-formatted reports and XML-formatted data and tables to UNIX files

The syntax for exporting objects from QMF temporary storage is different from the syntax for exporting objects from the database.

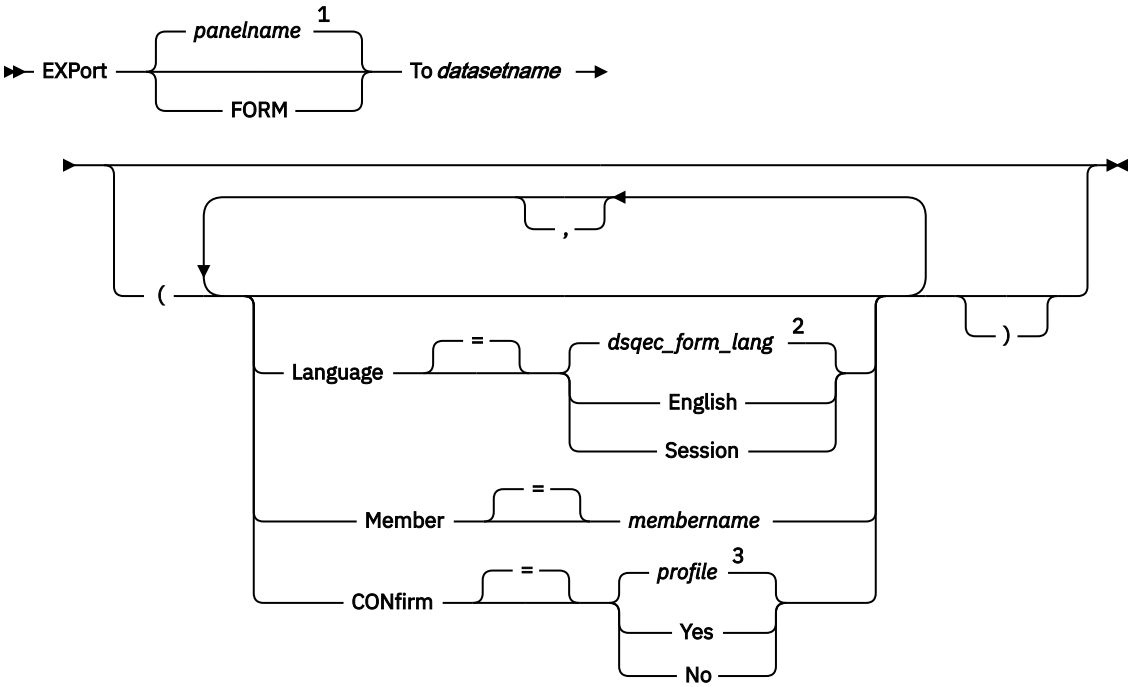
EXPORT a QMF query or procedure from temporary storage



Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The value set in your profile is used.

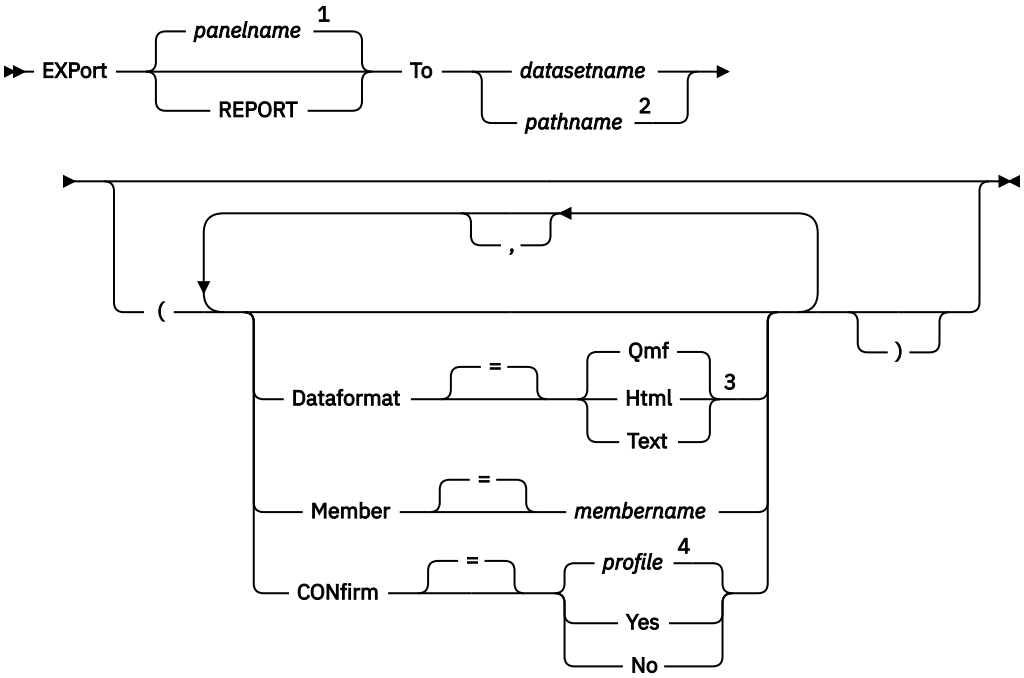
EXPORT a QMF form from temporary storage



Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The value set in this global variable is used.
[3] The value set in your profile is used.

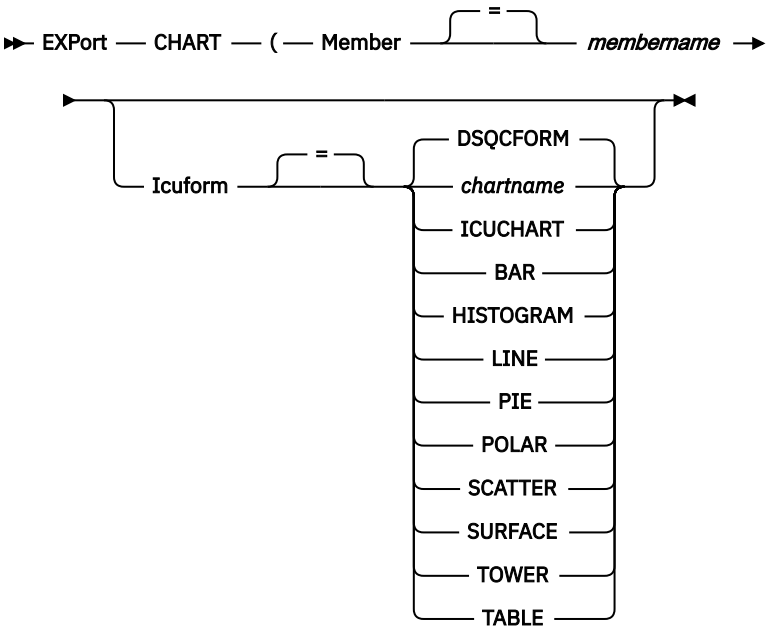
EXPORT a QMF report from temporary storage to a TSO data set or UNIX file



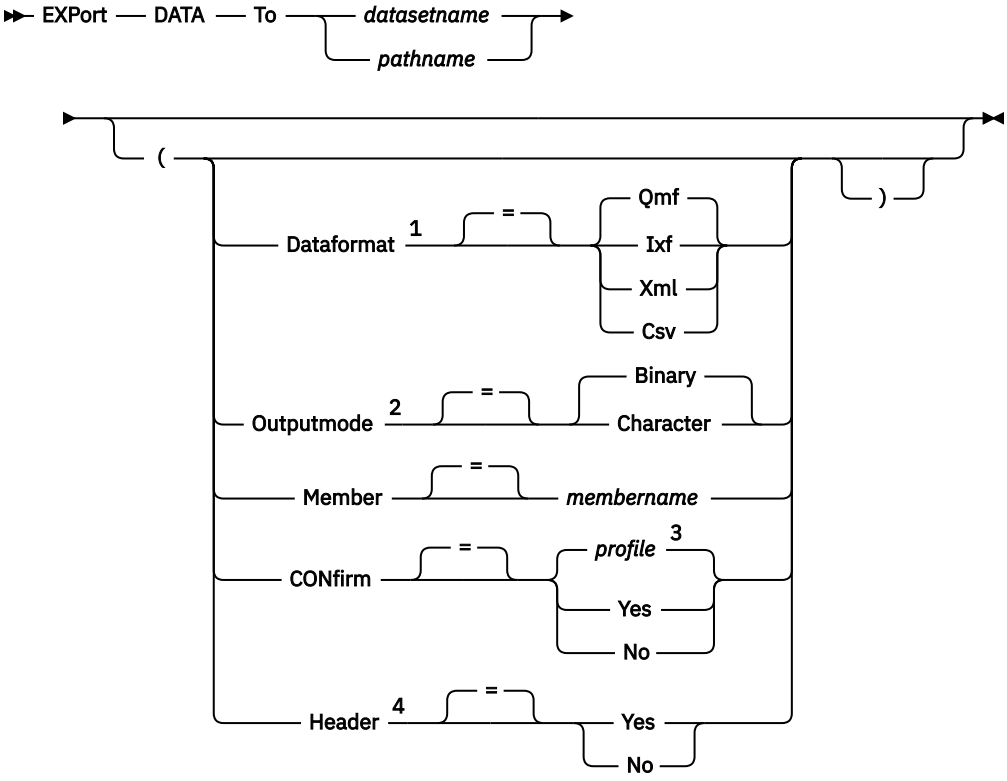
Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] Valid only when DATAFORMAT=HTML.
[3] Specify a value of HTML for the DATAFORMAT keyword when exporting a report to a UNIX file.
[4] The value set in your profile is used.

EXPORT a QMF chart from temporary storage



EXPORT QMF data from temporary storage to a TSO data set or UNIX file



Notes:

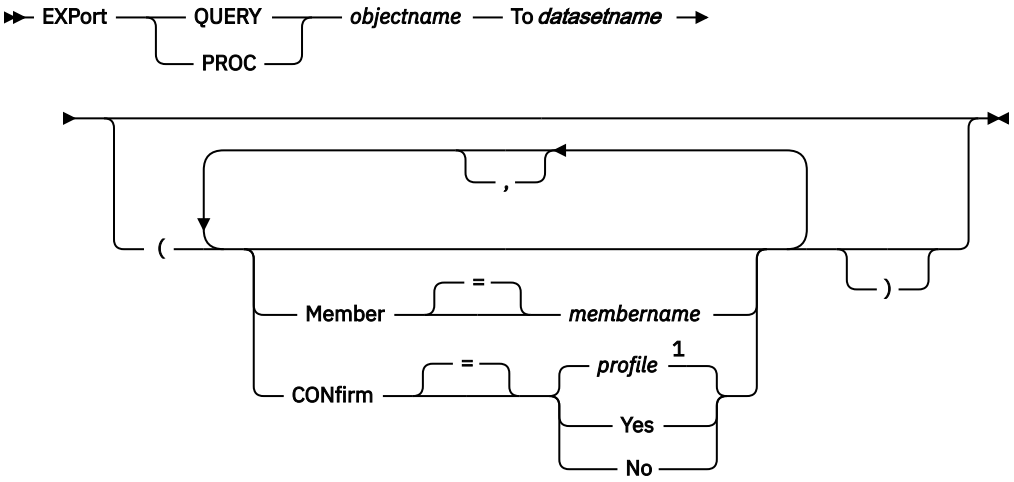
[1] If your data or table contains an XML column or LOB data, you must use the DATAFORMAT=XML clause on the command. This format can also be used when the data or table to be exported does not contain an XML column. Before you can export QMF data in XML format, you must set up the z/OS conversion environment for Unicode support.

[2] Accepted only when DATAFORMAT=IXF.

[3] The value set in your profile is used.

[4] Valid only when DATAFORMAT=CSV.

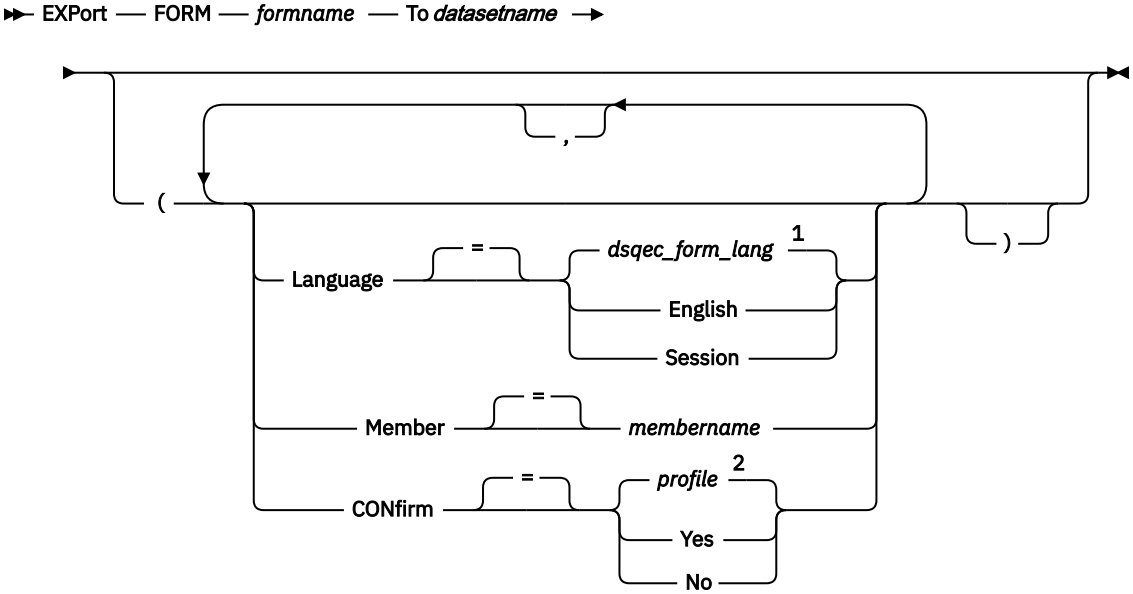
EXPORT a QMF query or procedure from the database



Notes:

[1] The value set in your profile is used.

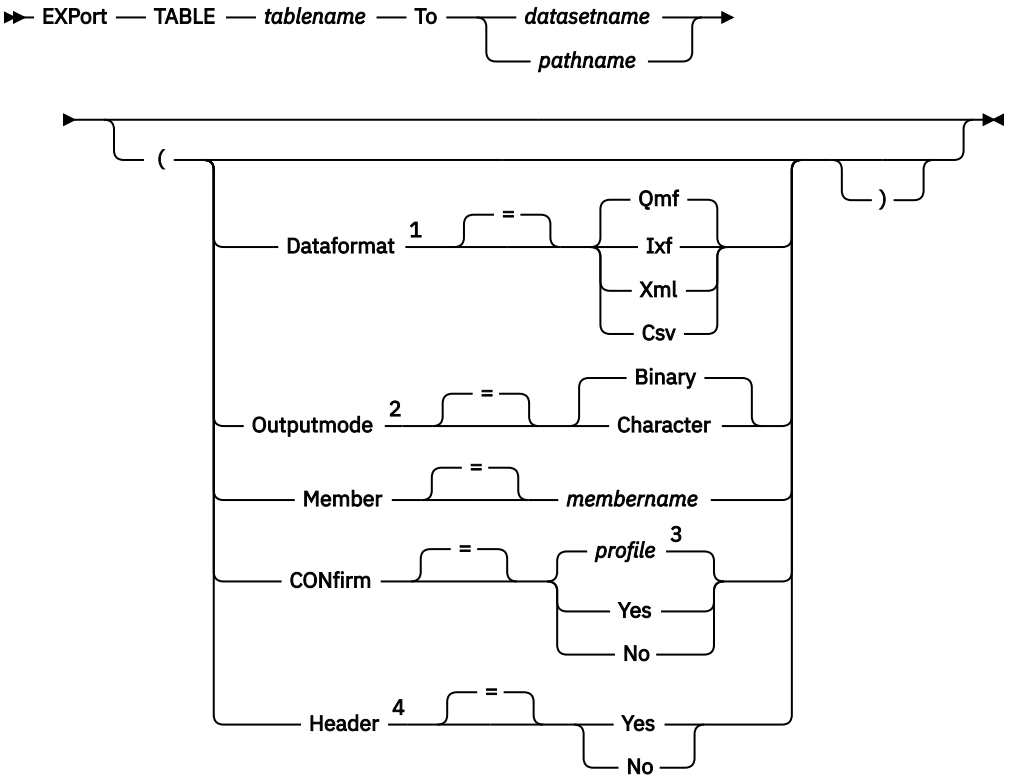
EXPORT a QMF form from the database



Notes:

[1] The value set in this global variable is used.
[2] The value set in your profile is used.

EXPORT a table from the database to a TSO data set or UNIX file



Notes:

[1] If your data or table contains an XML column or LOB data, you must use the DATAFORMAT=XML clause on the command. This format can also be used when the data or table to be exported does not contain an XML column. Before you can export QMF data in XML format, you must set up the z/OS conversion environment for Unicode support.

[2] Accepted only when DATAFORMAT=IXF.

[3] The value set in your profile is used.

[4] Valid only when DATAFORMAT=CSV.

Description

objectname

The name of a QMF object in the database.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

You cannot export an ANALYTIC object.

tablename

The name of a table, view, synonym, or alias in the database.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

datasetname

The name of the TSO data set for the exported object. The data set name is specified in either of the following ways:

- A partial TSO name without single quotation marks.

If the name is not surrounded in single quotes, QMF generates a fully qualified name by using your TSO prefix as the first qualifier and appending the object type as the last qualifier.

- A fully-qualified TSO data set name, where the entire name is enclosed in single quotation marks.

  This form must be used when the data set name has a prefix that is not your own.

  If you are using standard DASD devices, be sure that your storage management software is configured to handle dynamic allocation of extended data sets. When configuring these data sets, specify the default storage classes. The data sets must be defined as partitioned (with a data set organization, or DSORG, value of PO) or physical sequential (DSORG=PS). When your storage management system is configured in this manner, QMF dynamically allocates a data set with the name specified on the EXPORT command if it does not already exist. If you are exporting data in XML format, you could receive dynamic allocation errors if you have not properly configured your data sets. See the information provided with your storage management software for more information about how to configure dynamic allocation of extended data sets.

  If you are not using standard DASD devices, you must pre-allocate your data sets before using the EXPORT command. You can use global variables to specify the type and size of new data sets that will contain exported objects:

  – Use the DSQEC_PO global variable to specify the type of partitioned data set to create when you export an object to a member of a new data set. Specify a value of zero to use your site's default type, a value of 1 to use a PDS data set, or a value of 2 for a PDSE data set.
  – Use the DSQEC_DSALLOC_DIR global variable to specify the number of directory blocks when you export a member of a new PDS data set. The default is 20.
  – Use the DSQEC_DSALLOC_PRI global variable to specify the primary space allocation in tracks. The default is 15 tracks.
  – Use the DSQEC_DSALLOC_SEC global variable to specify the secondary space allocation in tracks. The default is 105 tracks.
  – Use the DSQEC_DSLRECL1 global variable to specify the logical record length (LRECL) of a new data set when you export an SQL query or procedure object. The LRECL for new data sets can be from 79 through 32,760 bytes. The default is 79.

If the specified data set name or file name already exists, the object is exported to a member of the existing data set, replacing its contents with the exported object provided that the file or data set attributes are suitable (for example, the record format and logical record length must be large enough to hold the exported data).

pathname
Names the UNIX file to receive the exported object. Surround UNIX path names in quotes and ensure that they are 250 or fewer characters. If you do not surround the path name in quotes, QMF appends the QMF object type to the end of the path name and surrounds the entire path name in quotes.

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

For charts, the exported object will be a member in the partitioned data set defined by your QMF environment for GDDM GDF (graphics data format) data. If the member already exists, it will be replaced.

membername
Names 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.

The MEMBER command option is ignored when you specify a UNIX path and file name.

CONFIRM
Indicates whether a confirmation panel is displayed when this command will replace an existing TSO data set or partitioned data set member. The CONFIRM parameter is ignored when you specify a UNIX path and file name. The UNIX file is replaced if it exists.

LANGUAGE
Indicates whether QMF keywords contained within the exported form are recorded in English or in the current NLF session language.

A QMF form containing QMF keywords in English can be used in any QMF session. A QMF form containing QMF keywords in any other national language supported by QMF can be used only in a session of that same national language.

DATAFORMAT
Specifies the file format to be used for the exported object.

QMF
This is the default format for exporting a report, a DATA object, or a table. The maximum length of a data row to be exported is 32 KB for this format. You can use 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.

The DSQDC_SHORT_EXPT global variable controls the length of all column name fields in the header records of data or tables exported with a value of QMF on the DATAFORMAT parameter.

HTML
You can specify HTML only when exporting a report. This is the default when exporting to a UNIX file. The TSO data set or UNIX file can be transferred to a Web server for viewing by a Web browser. The maximum length of a data row to be exported is 32 KB for this format. You can use 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.

TEXT
Exports reports without control information. This option can be used only when you export a report.

IXF
The Integrated Exchange Format. This can be used only when exporting a DATA object or a table. The maximum length of a data row to be exported is 32 KB for this format. You can use 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.

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 a DATA object or a table, and this is the only option when exporting data or tables to a UNIX file.

You must specify this format to export data that is defined as an XML data type. XML data type data can only be exported when you are connected to a database release that supports the XML data type.

You must also specify this format to export data or a table that contains LOB data. Note that the ability to export LOB data might be restricted by the DSQEC_LOB_RETRV and DSQEC_LOB_COLMAX global variables.

The maximum length of a data row to be exported is 2 GB for this format. Ensure that all characters in the XML data to be exported are supported by the XML parser.

Some sample XML files are supplied with QMF that allow you to display data in a browser.

CSV
Specifies CSV format. You can use this option only when exporting a data object or table. 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.

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.

ICUFORM
Specifies the name of a chart format. A chart format contains the specifications required to turn data into a chart. Different formats are used to produce different types of charts.

DSQCFORM
The name of the default chart format provided by QMF.

This format can be customized by your administrator. It provides a bar chart if not customized.

chartname
The name of a saved chart format

ICUCHART
Specifies the default chart format for the GDDM Interactive Chart Utility.

BAR
HISTOGRAM
LINE
PIE
POLAR
SCATTER
SURFACE
TOWER
TABLE
The name of a chart format provided by QMF.

HEADER
Specifies whether to include column headings with the exported data. You can specify this option only when the DATAFORMAT=CSV.

YES
Column headings are exported. This is the default setting. If you use this setting, the value of the DSQDC_COL_LABELS global variable controls whether column labels or column names are exported. The default of DSQDC_COL_LABELS is 1, which means that column labels are exported.

NO
Column headings are not exported.

Usage notes

- An empty or partial data set (or member of a partitioned data set) can result if there is an error in the execution of the EXPORT command.
- In some cases, if the object is exported to the same data set from which the current data was imported, you might receive an Incomplete Data prompt. At the prompt, choose NO and export the object to a different data set.
- When a form is exported, all parts of the form are exported.

  However, QMF will drop any FORM.DETAIL panel variations that were not modified from their default values. In this manner, unwanted FORM.DETAIL variations can be dropped by exporting and then importing the same form.
- If you are exporting a report or chart, and the form is incompatible with the data or contains errors, the first form panel containing an error is displayed with the error highlighted. To see other errors, correct the currently displayed error and press the Check function key.
- If the object is exported to a pre-existing PDS or PDSE data set that contains SQL query and procedure objects as members, you might get the following error message: `Record format of F should be V.` This message indicates that the record format should be defined as fixed rather than variable.

- If the UNIX file does not exist, QMF creates a new file for you. The file is created for reading and writing (PATHOPTS(ORDWR,OCREAT)). File permissions are set for the file owner to read, write, and execute (PATHMOD(SIRWXU)). If your file requires different attributes, allocate the file using the TSO ALLOCATE command and then export the object. If the UNIX file named in the path name exists, QMF erases and re-creates the file for reading and writing (PATHOPTS(ORDWR,OCREAT,OTRUNC)).
- To preserve the case of the path name, use a value of MIXED or STRING for the CASE option in the QMF profile.
- If you are exporting a table or data to a database using a three-part name and your database administrator has set up QMF to use the multirow fetch feature, both databases you are working with (local and remote) must be Db2 for z/OS; otherwise, your command will fail. Your database administrator can turn off multirow fetch. QMF commands with three-part names cannot be directed to DB2 for VSE and VM databases, nor can data be accessed remotely if you have started QMF as a stored procedure.
- You can use the EXPORT TABLE command to export a table from a QMF Data Service server. Use a three part name format and ensure that the DSQEC_DS_SUPPORT global variable is set.
- To use this command with columns that contain DECFLOAT data, the processor on which QMF is running must support decimal floating-point instructions.
- QMF updates the Last Used field for the object when you use this command. This field appears on object list panels displayed by the LIST command. You can change the list of commands that cause the field to be updated by setting the DSQEC_LAST_RUN global variable.
- If an EXPORT TABLE command is directed to a Unicode database and the table contains columns that have graphic data types, QMF casts the data to other types to avoid errors.
- When you issue an EXPORT TABLE command that references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. QMF allows you to set the value of this register using the SET CURRENT SCHEMA statement.
- When you issue the EXPORT DATA or EXPORT TABLE command, QMF exports either column labels or column names, depending on the value of the DATAFORMAT parameter:
  - When DATAFORMAT=QMF, column names are exported even for columns that have database labels defined.
  - When DATAFORMAT=IXF, labels are exported for any columns that have labels defined. Column names are exported for all other columns.
  - When DATAFORMAT=XML, QMF exports labels for all columns. If a column does not have a label already defined, QMF creates a label from the column name and exports that label.
  - When DATAFORMAT=CSV, column labels or column names are exported in column headings unless HEADER=NO is specified. The DSQDC_COL_LABELS global variable controls whether column labels or column names are exported.
- Although you can export data from temporal tables, you cannot export a temporal table. The data that you export from a temporal table is not associated with history data.

Examples

1. To display a command prompt panel for exporting an object:

   ```
   EXPORT ?
   ```

2. You can export an object (table, form, procedure, query, or report) from the database to which you are currently connected to a data set at the system on which QMF is executing. First use the CONNECT command to connect to the system where the object resides. Then issue an EXPORT command like this one:

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

   You cannot connect to a remote database if you have started QMF as a stored procedure.

3. If your current location is Db2 for z/OS, you can export a table from a remote Db2 location by including the location qualifier in the object name:

   ```
   EXPORT TABLE VENICE.LARA.STATSTAB TO dataset
   ```

   The table is exported from the database to which you are currently connected.

   QMF commands with three-part names cannot be directed to DB2 for VSE and VM databases, nor can data be accessed remotely if you have started QMF as a stored procedure.
4. If your TSO prefix is TOM and you use the TSO data set 'TOM.LOREN.QUERY(GAMMA)', specify the member name as follows:

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

   If you have no TSO prefix, your TSO user ID is used.

   If your prefix is set to blank, nothing is prefixed to the TSO name.
5. To export data in IXF character format:

   ```
   EXPORT DATA TO JBLP
    (CONFIRM=NO DATAFORMAT=IXF OUTPUTMODE=CHARACTER
   ```
6. To export a form using the national language of the current QMF session:

   ```
   EXPORT FORM TO MYFORM (LANGUAGE=SESSION
   ```
7. To copy a form named FORMA at the current location to a data set named FORMS at the system where QMF is executing:

   ```
   EXPORT FORM FORMA TO FORMS
   ```
8. To copy the table OKAMOTO.STATUS that resides in a database called TOKYO to a data set called YOURDATA on the system where QMF is running, issue the following command:

   ```
   EXPORT TABLE TOKYO.OKAMOTO.STATUS TO YOURDATA
   ```
9. To export the Q.STAFF table to the UNIX file '/u/DEPTJ49/pernal/mystaff.personnel', set the CASE option of your profile to MIXED or STRING and issue the following command:

   ```
   EXPORT TABLE Q.STAFF TO '/u/DEPTJ49/pernal/mystaff.personnel' (DATAFORMAT=XML
   ```
10. To export a report to a UNIX file specified by the unquoted UNIX path name `/u/QMFDEV/Robin/reports/test`, set the CASE option of your QMF profile to MIXED or STRING and issue the following command:

    ```
    EXPORT REPORT TO /u/QMFDEV/Robin/reports/test (DATAFORMAT=HTML
    ```

    This command exports the report to a file with the following absolute path name:

    /u/QMFDEV/Robin/reports/test.REPORT
11. To export data in CSV format without column headings:

    ```
    EXPORT DATA TO MYDATA
     (DATAFORMAT=CSV HEADER=NO
    ```

Related concepts

How QMF recasts certain data types when displaying data

When a DISPLAY TABLE command is directed to a Unicode database and the table referenced in the command contains columns with graphic data types, QMF converts the graphic data types to character data types.

Related reference

SET special register

You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

Global variables that control how commands and procedures are executed
DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

Global variables that control various displays
DSQDC global variables control the display of certain kinds of information. All of these global variables can be modified by the SET GLOBAL command.

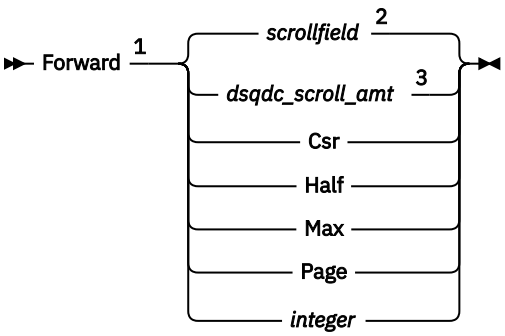
Related information

Search for information about unsupported characters by referring to the XML Toolkit for z/OS User's Guide.

FORWARD

The FORWARD command scrolls toward the bottom of a scrollable area. You can scroll until the last line is at the top of your screen.

TSO with ISPF	TSO without ISPF	CICS
X	X	X



Notes:

[1] Specify scroll amounts only when there is a SCROLL field on the active panel. PAGE is assumed in all other situations.
[2] The value shown in the SCROLL field is used. This value is also maintained in the global variable DSQDC_SCROLL_AMT.
[3] The value set in this global variable is used.

Description

CSR
Scrolls the line where the cursor is positioned to the top of the scrollable area.

HALF
Scrolls forward half the depth of the scrollable area or to the bottom (if that is nearer).

MAX
Scrolls to the bottom of the scrollable area. FORWARD MAX is equivalent to BOTTOM.

PAGE
Scrolls forward the depth of the scrollable area or to the bottom (if that is nearer).

integer
Scrolls forward this number of lines on the panel (a whole number ranging from 1 through 9999).

Usage notes

- MAX is in effect only for the current command. This value will not remain in the SCROLL field after the command completes. The global variable DSQDC_SCROLL_AMT cannot be set to this value.
- To scroll forward in the footing text on form panels, position the cursor on the portion of the panel where the footing text is located and enter the FORWARD command.

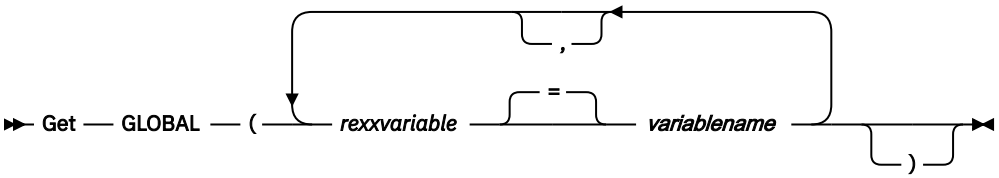
GET GLOBAL

The linear syntax of the GET GLOBAL command assigns values of QMF global variables to REXX variables in applications and procedures written in REXX.

The extended syntax of the GET GLOBAL command allows application programs (written in languages other than REXX) to use the callable interface to access data from the QMF global variable pool.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Linear syntax (used with REXX only)



Description

rexxvariable
The name of a REXX variable in your procedure with logic or REXX application.

variablename
The name of a QMF global variable.

Usage notes

This command is not valid on the QMF command line.

When accessing multiple variables with the GET GLOBAL command, the following rules apply:

- Equal signs are optional between *rexxvariable* and *variablename*.
- Commas are optional between global variable/value pairs.
- Delimiters between *rexxvariable* and *variablename* must be one or more blanks or an equal sign with or without blanks.
- Delimiters between global variable/value pairs (both *rexxvariable* and *variablename*) must be one or more blanks or a comma (with or without blanks).
- Each REXX variable can have only one variable name associated with it.

The GET GLOBAL command does not have an associated command prompt panel. Command prompting is not available for this command.

Although not required by QMF, it is recommended that uppercase be used for all variable names.

Unless there is a synonym specified, QMF considers "get global" (in lowercase) to be an error. For consistency across systems, specify this and all other QMF commands in uppercase (whether in QMF or REXX procedures or in the callable interface).

- In a QMF application written in REXX, this example assigns the value of the QMF global variable DSQAITEM to the REXX variable ITEM:

ADDRESS QRW "GET GLOBAL (ITEM = DSQAITEM"

- In a QMF procedure written in REXX, this example assigns the value of the QMF global variable DSQCIQMG to the REXX variable MSG:

  "GET GLOBAL (MSG = DSQCIQMG"

GETQMF macro

GETQMF is an edit macro, not a QMF command. It inserts a QMF report into a document.

TSO With ISPF	TSO Without ISPF	CICS
X	X	

From an edit session, you can use the GETQMF macro, as shown in the command below, to insert a QMF report into the document being edited without leaving the session. The QMF report to be inserted must be printed within a QMF session before it can be inserted into a document.

```
GETQMF type option name
```

type
Whether SCRIPT/VS control words are inserted.

DCF
For a SCRIPT/VS document. Document Composition Facility (DCF) places the SCRIPT/VS control words before and after the QMF report. In addition, each printer page eject is replaced by a SCRIPT/VS page eject, and SCRIPT/VS control words are placed at the heading and footing of each page.

ASIS
For a QMF report as it is. If TYPE is not specified, ASIS is assumed.

option name
Whether you are creating a new report or inserting an existing one.

USEQMF
Creates a QMF report dynamically using a procedure that prints a report, where *name* is the name of the saved procedure.

DSN
Inserts an existing report from a TSO data set, where *name* is the name of the TSO data set containing the report.

HELP

The HELP command displays information about QMF. Two forms of help information are available.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Topic help

Message help

Description

messageid
A QMF message identifier. QMF attempts to find message help associated with the message ID you specify. If found, it is displayed. If not, an error message is displayed.

A message ID must begin with "DSQ" followed by a five-digit number or "DYQ" followed by a four-digit number (for example: DSQ20114 or DYQ0008).

When your trace settings specify tracing of messages and commands (such as when you are running QMF in batch mode or when you start QMF as a stored procedure and specify the L2 trace option), you can search the trace output for message numbers to diagnose problems.

Usage notes

The information you see when you issue the HELP command without the *messageid* parameter depends on what is on your screen at the time.

From the QMF home panel:
Issuing HELP displays a list of topics about QMF features and functions, such as commands, charts, procedures, reports, and forms.

From a panel with an error message on it:
Issuing HELP displays information about the error message.

From other help panels:
Issuing HELP displays more information about the displayed panel. There are separate sequences of help for the following panels:

- QUERY
- PROC
- PROFILE
- REPORT
- All form panels
- Database object list
- Global variable list
- Prompted Query
- Table Editor

When you specify a message ID with HELP, information about the message is displayed. For example, if you want to display information about error message DSQ20047, issue the following command:

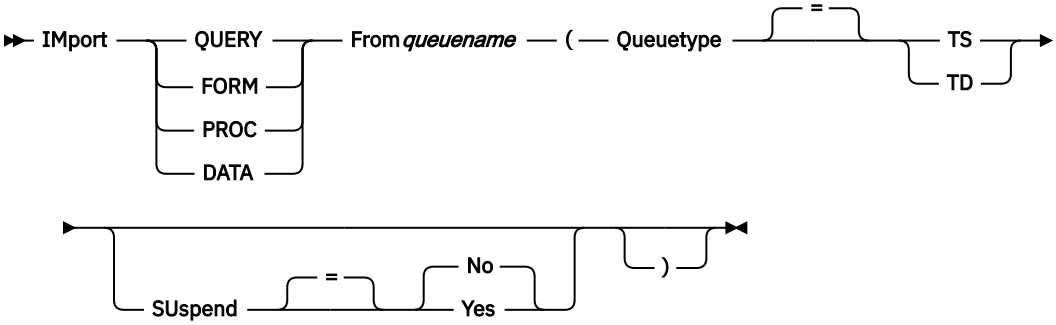
```
HELP DSQ20047
```

Related information
How to read QMF messagesWhen an error occurs, QMF normally displays a message number.

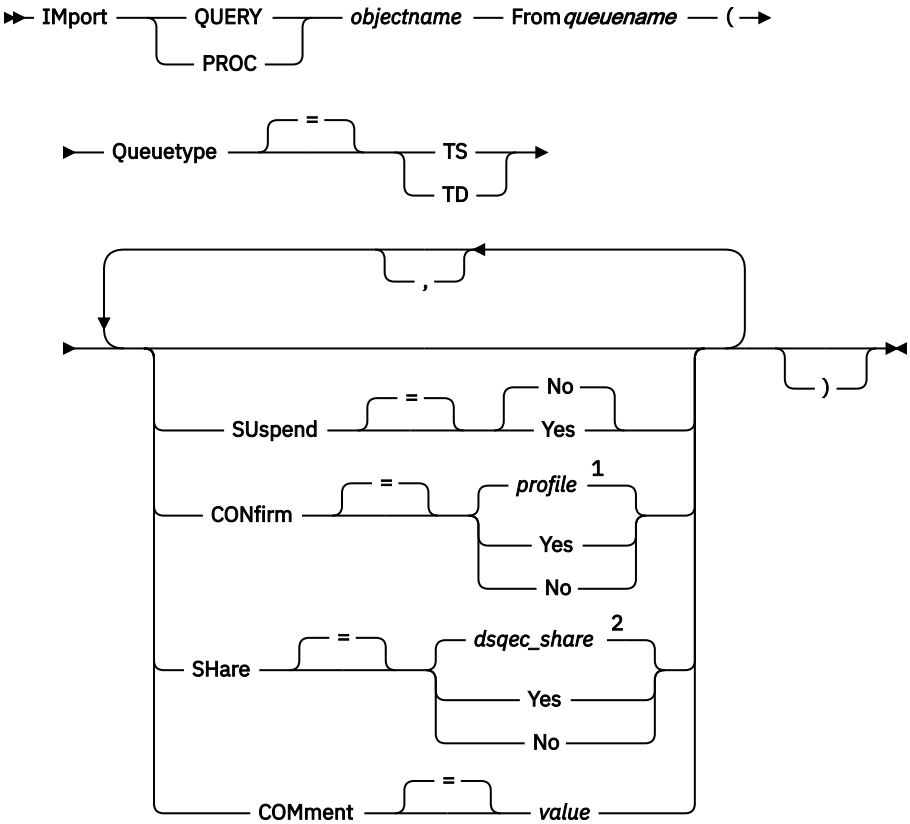
IMPORT in CICS

The IMPORT command copies the contents of a CICS data queue into QMF temporary storage or into the database.

IMPORT a QMF object into temporary storage



IMPORT a QMF query or procedure into the database

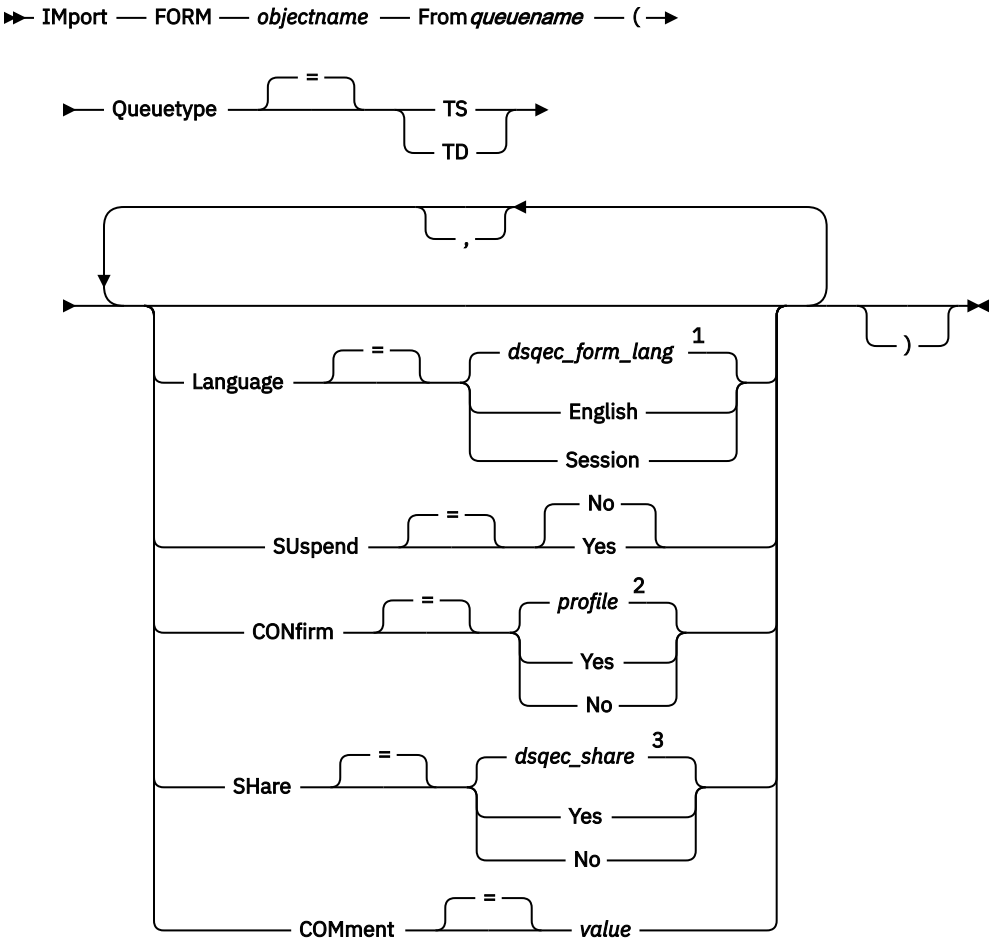


Notes:

[1] The value set in your profile is used.

[2] For an object being replaced, the current value is left unchanged. Otherwise, the value set in this global variable is used.

IMPORT a QMF form into the database



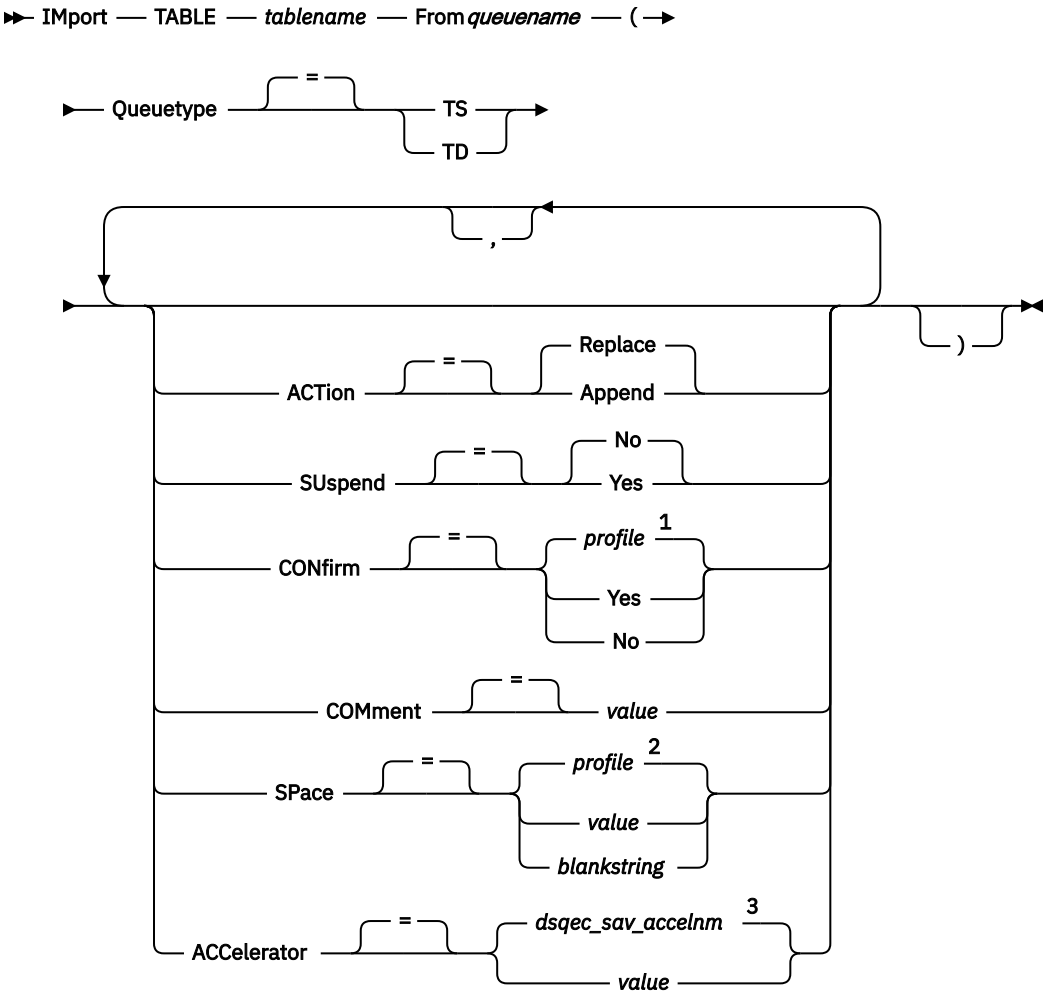
Notes:

[1] The value set in this global variable is used.

[2] The value set in your profile is used.

[3] For an object being replaced, the current value is left unchanged. Otherwise, the value set in this global variable is used.

IMPORT a table into the database



Notes:

[1] The value set in your profile is used.
[2] The value set in your profile is used.
[3] The value set in this global variable is used.

Description

objectname
The name for the QMF object in the database.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

tablename
The name of a table, view, synonym, or alias.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

queuename
The name of a CICS data queue containing the QMF object. The maximum length of the name is:

- 4 characters when QUEUETYPE is TD.
- 8 characters when QUEUETYPE is TS.

For a TS queue, surround the name in single quotation marks if it contains special characters, such as a period.

QUEUETYPE
Indicates the type of data queue containing the QMF object. There is no default for QUEUETYPE; it must be specified.

TS
A CICS temporary storage queue

TD
A CICS transient data queue

ACTION
Indicates whether to replace the entire database table with the imported data or to append the imported data to the existing table.

LANGUAGE
Indicates whether QMF keywords contained within the imported form are recorded in English or in the current NLF session language.

A QMF form containing QMF keywords in English can be used in any QMF session. A QMF form containing QMF keywords in any other national language supported by QMF can be used only in a session of that same national language.

SUSPEND
Specifies the action to take when the data queue is busy or unavailable.

NO
Cancels the import request.

YES
Waits until the data queue is available.

CONFIRM
Indicates whether a confirmation panel is displayed when this command will replace an existing object in the database.

SHARE
Determines whether other QMF users can access the imported object.

COMMENT
Stores a comment with the imported object. A comment is a remark or note that you can create when you import the object. The purpose of creating a comment is to provide descriptive information about the object. Users with whom the object is shared can then view this information by pressing the Comments key when the object is displayed in a list.

You cannot replace a comment on a table you do not own or on a remote table using a three-part name.

value
The character string that makes up the content of the comment.

A value that contains blank characters must be surrounded with delimiters. Valid delimiters for a comment value are single quotation marks, parentheses, and double quotation marks. If you are using the IMPORT command from the QMF command line or in a procedure to store a comment with the object, the comment text can be up to 78 single-byte characters. If you are using the IMPORT Command Prompt panel to enter the comment, the comment can be up to 57 single-byte characters.

When the comment itself contains a delimiter character (a single quotation mark, double quotation mark, or parentheses), surround the entire comment with one of the other types of delimiters so that QMF saves the entire comment.

SPACE
Names a storage space to hold any tables that are created by the SAVE DATA command. A blank value specifies that you will use the space that is chosen by the database manager program.

ACCELERATOR
Specifies the name of the accelerator in which the table will be created.

Usage notes

- Use of TSO data sets in CICS is not recommended. However, if you do choose to use TSO data sets, there are additional customization steps required to support both IMPORT and EXPORT commands. TSO data sets referenced by the IMPORT command under CICS must be defined as either partitioned (with a data set organization, or DSORG, value of PO) or physical sequential (DSORG=PS).
- A QMF administrator can import a QMF object for another user.
- The queue must contain a single, complete QMF object before the IMPORT command is issued.
- When data is imported, a new form is created. Any existing form in temporary storage is replaced.
- You cannot import reports, charts, or CSV data.
- If you are connected to a remote location, the tables at the server are read-only. Objects cannot be imported into that database.
- When you import into the database and an object already exists with the same name you specify, QMF replaces or appends the object (according to the value of the ACTION parameter), subject to these conditions:
  - A form can replace only a form.
  - A procedure can replace only a procedure.
  - A query can replace only a query.
  - A table can replace or append only a similar table object.

    A similar table is one with the same number of columns, with corresponding columns each having 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.

    Column names and labels do not have to match.

    If the data to be imported contains XML columns, the data to be imported and the existing table:

    - Must have the same number of XML columns in the same positions
    - Must have the same null characteristics defined for the XML columns
- If you are importing a table containing an XML column, ensure that the column contains well-formed XML documents. Ensure that all characters in the XML columns to be imported are supported by the XML parser. The data you are importing must conform to the QMF XML format. XML data can only be imported when you are connected to a database release that supports the XML data type.
- When the data is in XML format, the maximum length of a data row to be imported is 2 GB.
- When you import into an existing table, the column names and labels remain unchanged. If the table does not exist, a new table is created using the column names and labels in the imported object.
- If your current location is a Db2 for z/OS server, you can import to an existing table at a remote location by specifying a three-part name for the table. (However, you cannot import a new table or any QMF objects this way.) If your database administrator has set up QMF to use the multirow fetch feature, both databases you are working with (local and remote) must be Db2 for z/OS if you are using three-part names; otherwise, your command will fail. QMF commands with three-part names cannot be directed to DB2 for VM or VSE databases.
- Use the IMPORT command sparingly in CICS because it can negatively affect QMF performance for other users.
- The contents of a CICS TD queue are discarded when errors occur during an import. Be sure to use the correct object type for the object currently in the queue. A mismatch will result in an empty queue and no object imported.
- QMF handles CICS TD queues differently than CICS TS queues:

Transient data queues
QMF imports the entire transient data queue, possibly creating a long delay prior to displaying the object. The entire object must fit into your storage or spill file.

- An intrapartition TD queue can hold up to 32 KB rows of data.
- An extrapartition TD queue can be as large as needed to hold the object.

Temporary storage queues
A temporary storage queue can hold up to 32 KB rows of data. When importing DATA from a CICS TS queue, QMF pauses to display the report after retrieving the number of rows indicated by the DSQSIROW parameter. You can complete the import by issuing a BOTTOM command. If there is not enough storage to complete the report, use the QMF RESET command to reset the data.

- The ability to import a table that contains LOB data is controlled by the DSQEC_LOB_SAVE global variable.
- To use this command with columns that contain DECFLOAT data, the processor on which QMF is running must support decimal floating-point instructions.
- QMF updates the Last Used field for the object when you use this command. This field appears on object list panels displayed by the LIST command. You can change the list of commands that cause the field to be updated by setting the DSQEC_LAST_RUN global variable.
- When you issue an IMPORT TABLE command that references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. QMF allows you to set the value of this register using the SET CURRENT SCHEMA statement.
- When you issue the IMPORT TABLE command with the ACTION=REPLACE parameter and the data to be imported contains column label information, QMF creates labels on the new table if the database supports the LABEL ON statement. If the database does not support the LABEL ON statement, the new table is created without column labels.
- When you import into an existing table, the column names and labels remain unchanged. If you issue the IMPORT TABLE command with the ACTION=REPLACE or ACTION=APPEND parameter, and the existing table is a temporal table, the table remains temporal. When you import a table, new values are created for columns that were defined with the GENERATED ALWAYS attribute.
- If you issue the IMPORT TABLE command and the specified table does not exist, a new table is created using the column names and labels in the imported object. You cannot import a table into a new temporal table. When you import into a new table, the table is created with new values for columns that were defined with the GENERATED ALWAYS attribute.
- You cannot specify both the SPACE and the ACCELERATOR parameter in the same command.
- If the SPACE or ACCELERATOR parameter is used in the command and the table already exists, SPACE or ACCELERATOR is ignored. The table is re-created at the original location.
- The value of the DSQEC_SAV_ALLOWED global variable determines the default behavior of the SPACE and ACCELERATOR parameters:
  - When the global variable is set to 0, the SAVE DATA command cannot not be used.
  - When the global variable is set to 1, tables are saved only to the database, and only the SPACE parameter is allowed. If the SPACE parameter is not specified, the value is taken from the QMF profile.
  - When the global variable is set to 2, tables are saved only to the accelerator, and only the ACCELERATOR parameter is allowed. If the ACCELERATOR parameter is not specified, the accelerator name that is specified in the DSQEC_SAV_ACCELNM global variable is used.
  - When the global variable is set to 3, tables are saved by default to the database and are saved to the accelerator only when the ACCELERATOR parameter is specified. If neither the SPACE parameter nor the ACCELERATOR parameter is specified, the value of the SPACE setting from the QMF profile is used.
  - When the global variable is set to 4, tables are saved by default to the accelerator and are saved to the database only when the SPACE parameter is specified. If neither the SPACE parameter

nor the ACCELERATOR parameter is specified, the accelerator name that is specified in the DSQEC_SAV_ACCELNM global variable is used.

- Accelerator-only tables are created in the database defined in the DSQEC_SAV_ACCELDB global variable. The Q.PROFILES.SPACE value is not used when defining accelerator-only tables.

Examples

1. To display a prompt panel for the QMF IMPORT command:

```
IMPORT ?
```

2. To copy the data queue VTAB to the table REYNOLDS.VISIONS:

```
IMPORT TABLE REYNOLDS.VISIONS FROM VTAB (QUEUETYPE=TD
```

3. To copy the data queue QUERY.A to the query REYNOLDS.QUERYA:

```
IMPORT QUERY REYNOLDS.QUERYA
    FROM 'QUERY.A' (QUEUETYPE=TS
```

Related reference

RESET object
The RESET command restores an object in temporary storage to its initial state. This command does not apply to ANALYTIC objects.

SET special register
You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

Global variables that control how commands and procedures are executed
DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

Related information

Exporting and importing objectsYou can write applications that issue QMF™ EXPORT and IMPORT commands to place objects outside of the QMF environment.

Exporting data or tables in XML formatIf your data or table contains an XML column or LOB data, you must use the DATAFORMAT=XML clause on the EXPORT DATA or EXPORT TABLE command.

Search for information about unsupported characters in the XML Toolkit for z/OS User's Guide, and about support for implicit casting with your database.

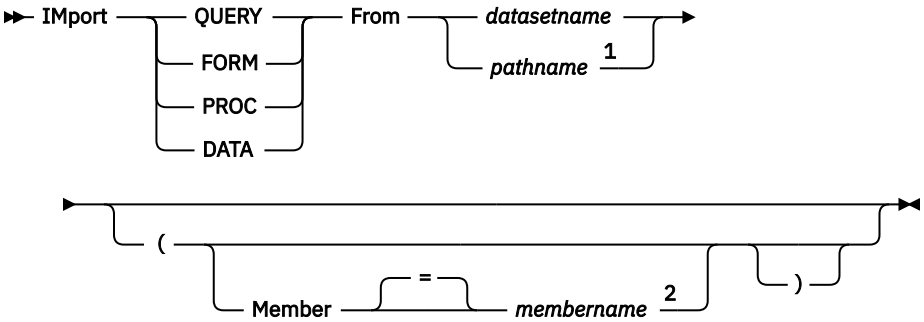
IMPORT in TSO

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

TSO with ISPF	TSO without ISPF
X	X

Syntax

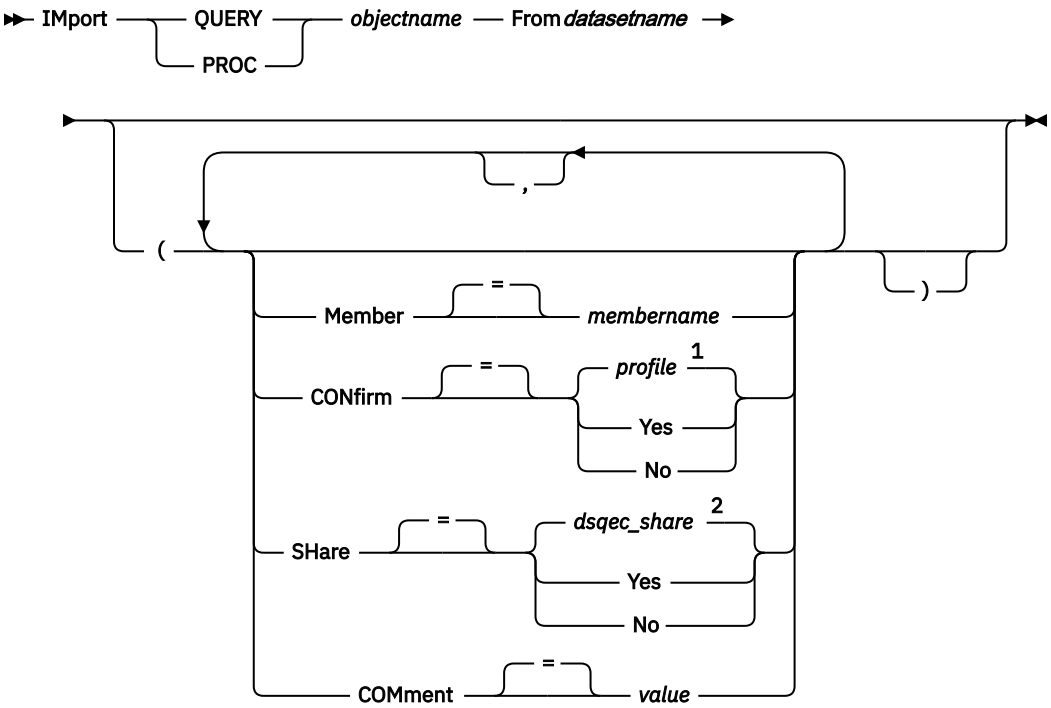
IMPORT a QMF object into temporary storage



Notes:

[1] QMF accepts a path name only when the object is DATA and the data is in XML format.
[2] Accepted only when you import from a TSO data set.

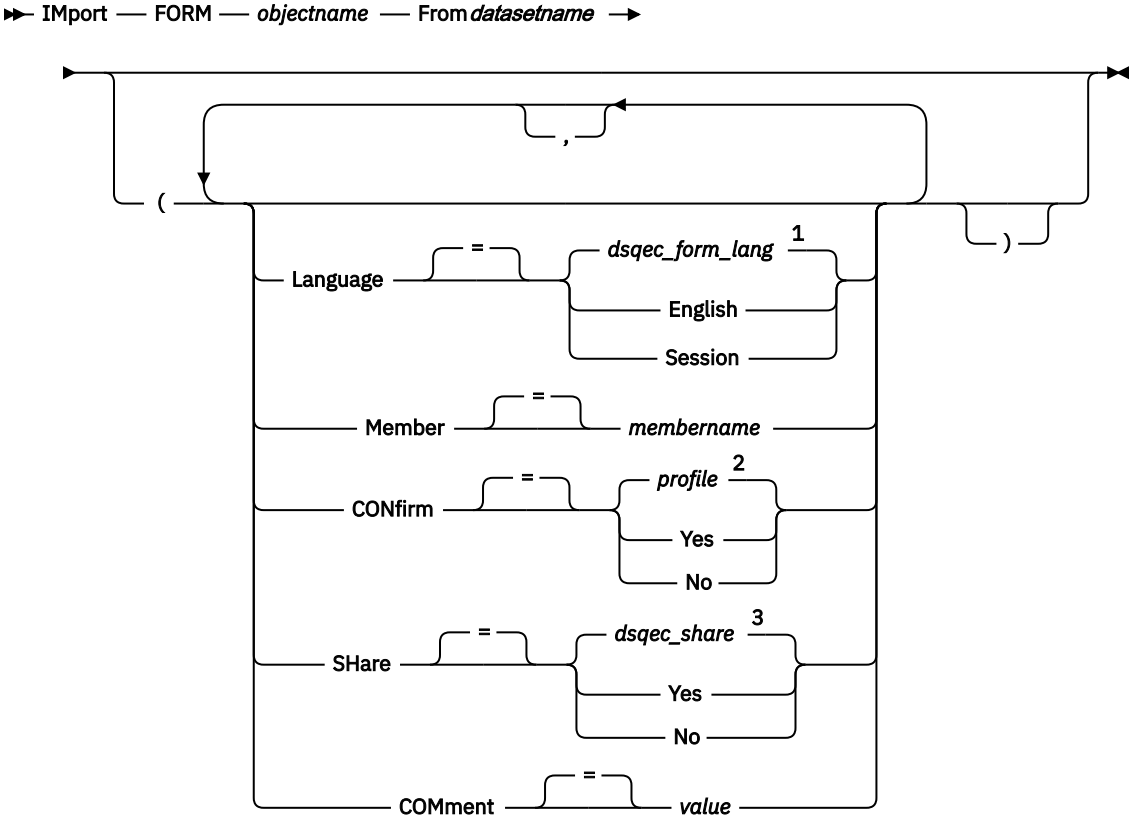
IMPORT a QMF query or procedure into the database



Notes:

[1] The value set in your profile is used.
[2] For an object being replaced, the current value is left unchanged. Otherwise, the value set in this global variable is used.

IMPORT a QMF form into the database



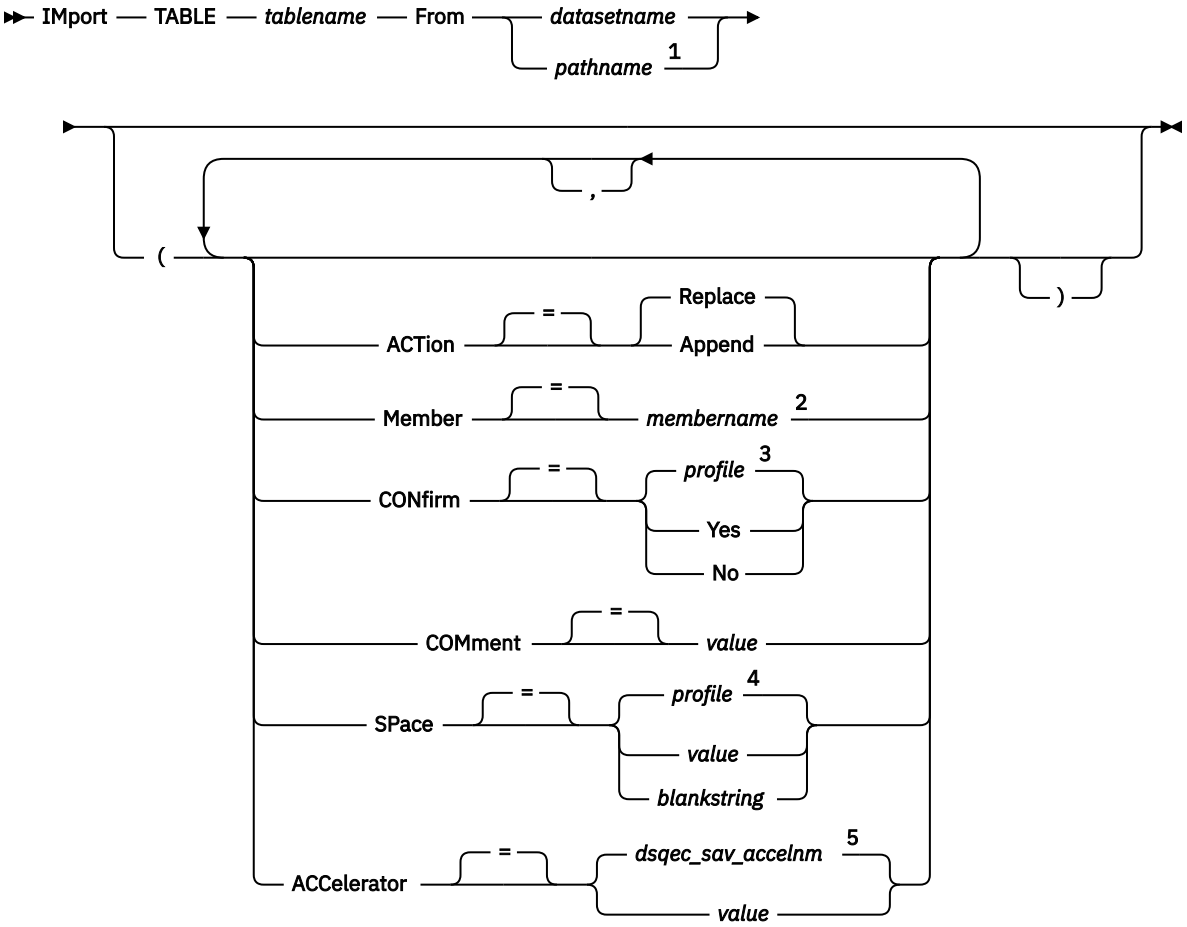
Notes:

[1] The value set in this global variable is used.

[2] The value set in your profile is used.

[3] For an object being replaced, the current value is left unchanged. Otherwise, the value set in this global variable is used.

IMPORT a table into the database



Notes:

[1] QMF accepts a path name only when the table is in XML format.
[2] Accepted only when you import from a TSO data set.
[3] The value set in your profile is used.
[4] The value set in your profile is used.
[5] The value set in this global variable is used.

Description

datasetname
The TSO data set to copy. The data set name is specified in either of the following ways:

- A partial TSO name without single quotation marks.

  A fully qualified data set name is generated by using your TSO prefix as the first qualifier and appending the object type as the last qualifier.
- A fully qualified TSO data set name, where the entire name is enclosed in single quotation marks.

  Quotation marks must be used when the data set name has a prefix that is not your own.

pathname
Names the UNIX file from which to retrieve the object. Surround UNIX path names in quotes and ensure that they are 250 or fewer characters. If you do not surround the path name in quotes, QMF appends the QMF object type to the end of the path name and surrounds the path name in quotes.

objectname
The name for the QMF object in the database.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

tablename
The name of a table, view, synonym, or alias.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

ACTION
Indicates whether to replace the entire database table with the imported data or to append the imported data to the existing table.

LANGUAGE
Indicates whether QMF keywords contained within the imported form are recorded in English or in the current NLF session language.

A QMF form that contains QMF keywords in English can be used in any QMF session. A QMF form that contains QMF keywords in any other national language that is supported by QMF can be used only in a session of that same national language.

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

membername
The name of the member 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
Indicates whether a confirmation panel is displayed when this command will replace an existing object in the database.

SHARE
Determines whether other QMF users can access the imported object.

COMMENT
Stores a comment with the imported object. A comment is a remark or note that you can create when you import the object. The purpose of creating a comment is to provide descriptive information about the object. Users with whom the object is shared can then view this information by pressing the Comments key when the object is displayed in a list.

You cannot replace a comment on a table you do not own or on a remote table using a three-part name.

value
The character string that makes up the content of the comment.

A value that contains blank characters must be surrounded with delimiters. Valid delimiters for a comment value are single quotation marks, parentheses, and double quotation marks. If you are using the IMPORT command from the QMF command line or in a procedure to store a comment with the object, the comment text can be up to 78 single-byte characters. If you are using the **IMPORT Command Prompt** panel to enter the comment, the comment can be up to 57 single-byte characters.

When the comment itself contains a delimiter character (a single quotation mark, double quotation mark, or parentheses), surround the entire comment with one of the other types of delimiters so that QMF saves the entire comment.

SPACE
Names a storage space to hold any tables that are created by the SAVE DATA command. A blank value specifies that you will use the space that is chosen by the database manager program.

ACCELERATOR
Specifies the name of the accelerator in which the table will be created.

Usage notes

- If you import a QBE query that was exported from a QMF Version 11.1 or earlier system, the query is converted in temporary storage with long-name characteristics and cannot be used if you connect to a QMF Version 11.1 or earlier system. Furthermore, if you save the imported query, it cannot be used in QMF Version 11.1 or earlier systems.
- Data sets referenced by the IMPORT command must be either partitioned (with a data set organization, or DSORG, value of PO) or physical sequential (DSORG=PS).
- The IMPORT command fails if the object or the database into which the object is being imported is read-only.
- The data set must contain a single, complete QMF object before the IMPORT command is issued.
- A QMF administrator can import a QMF object for another user.
- When data is imported, a new form is created. Any existing form in temporary storage is replaced.
- You cannot import reports, charts, or CSV data.
- When you import into the database and an object exists with the same name you specify, QMF replaces or appends the object (according to the value of the ACTION parameter), subject to these conditions:
  - A form can replace only a form.
  - A procedure can replace only a procedure.
  - A query can replace only a query.
  - A table can replace or append only a similar table object.

    A similar table is one with the same number of columns, with corresponding columns each having 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.

    Column names and labels do not have to match.

    If the data to be imported contains XML columns, the data to be imported and the existing table:

    - Must have the same number of XML columns in the same positions
    - Must have the same null characteristics that are defined for the XML columns
- When you import into an existing table, the column names and labels remain unchanged. If the table does not exist, a new table is created that uses the column names and labels in the imported object.
- Objects can be imported to a remote location. Use the QMF CONNECT command to make the remote location your current location first, followed by the IMPORT command.
- If your current location is a Db2 for z/OS server, you can import to an existing table at a remote location by specifying a three-part name for the table. (However, you cannot import a new table or any QMF objects this way.) If your database administrator set up QMF to use the multirow fetch feature, both databases you are working with (local and remote) must be Db2 for z/OS if you are using three-part names; otherwise, your command fails. Your database administrator can turn off multirow fetch. QMF commands with three-part names cannot be directed to DB2 for VSE and VM databases, nor can data be accessed remotely if you start QMF as a stored procedure.
- If you are importing a table that contains an XML column, ensure that the column contains well-formed XML documents. Ensure that all characters in the XML columns to be imported are supported by the XML parser. XML data can only be imported when you are connected to a database release that supports the XML data type.
- When the data is in XML format, the maximum length of a data row to be imported is 2 GB.
- The ability to import a table that contains LOB data is controlled by the DSQEC_LOB_SAVE global variable.
- You cannot import ANALYTIC objects.
- To use this command with columns that contain DECFLOAT data, the processor on which QMF is running must support decimal floating-point instructions.

- QMF updates the **Last Used** field for the object when you use this command. This field appears on object list panels that are displayed by the LIST command. You can change the list of commands that cause the field to be updated by setting the DSQEC_LAST_RUN global variable.
- When you issue an IMPORT TABLE command that references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. QMF allows you to set the value of this register using the SET CURRENT SCHEMA statement.
- When you issue the IMPORT TABLE command with the ACTION=REPLACE parameter and the data to be imported contains column label information, QMF creates labels on the new table if the database supports the LABEL ON statement. If the database does not support the LABEL ON statement, the new table is created without column labels.
- When you import into an existing table, the column names and labels remain unchanged. If you issue the IMPORT TABLE command with the ACTION=REPLACE or ACTION=APPEND parameter, and the existing table is a temporal table, the table remains temporal. When you import a table, new values are created for columns that were defined with the GENERATED ALWAYS attribute.
- If you issue the IMPORT TABLE command and the specified table does not exist, a new table is created using the column names and labels in the imported object. You cannot import a table into a new temporal table. When you import into a new table, the table is created with new values for columns that were defined with the GENERATED ALWAYS attribute.
- You cannot specify both the SPACE and the ACCELERATOR parameter in the same command.
- If the SPACE or ACCELERATOR parameter is used in the command and the table already exists, SPACE or ACCELERATOR is ignored. The table is re-created at the original location.
- The value of the DSQEC_SAV_ALLOWED global variable determines the default behavior of the SPACE and ACCELERATOR parameters:
  - When the global variable is set to 0, the SAVE DATA command cannot not be used.
  - When the global variable is set to 1, tables are saved only to the database, and only the SPACE parameter is allowed. If the SPACE parameter is not specified, the value is taken from the QMF profile.
  - When the global variable is set to 2, tables are saved only to the accelerator, and only the ACCELERATOR parameter is allowed. If the ACCELERATOR parameter is not specified, the accelerator name that is specified in the DSQEC_SAV_ACCELNM global variable is used.
  - When the global variable is set to 3, tables are saved by default to the database and are saved to the accelerator only when the ACCELERATOR parameter is specified. If neither the SPACE parameter nor the ACCELERATOR parameter is specified, the value of the SPACE setting from the QMF profile is used.
  - When the global variable is set to 4, tables are saved by default to the accelerator and are saved to the database only when the SPACE parameter is specified. If neither the SPACE parameter nor the ACCELERATOR parameter is specified, the accelerator name that is specified in the DSQEC_SAV_ACCELNM global variable is used.
- Accelerator-only tables are created in the database defined in the DSQEC_SAV_ACCELDB global variable. The Q.PROFILES.SPACE value is not used when defining accelerator-only tables.

Examples

1. To display a prompt panel for the QMF IMPORT command:

   ```
   IMPORT ?
   ```

2. If your TSO prefix is JULIA and you want to copy a member of your partitioned data set ('JULIA.LOREN.QUERY(GAMMA)') into the database and give it the name FIRSTQ:

   ```
   IMPORT QUERY FIRSTQ FROM LOREN (MEMBER=GAMMA
   ```

3. To add data (NEW.ROWS) to a table (MYTABLE):

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

4. To import a table to a remote database server (VENICE), first connect to that location:

   ```
   CONNECT TO VENICE
   ```

   Then import the table:

   ```
   IMPORT TABLE LARA.STATSTAB FROM YOURDATA
   ```

   You cannot connect to a remote database if you start QMF as a stored procedure.
5. If your current location is a Db2 for z/OS server, and you want to copy the data set 'G7.STATS.TABLE' from the system where QMF is executing to an existing table (OKAMOTO.STATUS) at a remote database location (TOKYO):

   ```
   IMPORT TABLE TOKYO.OKAMOTO.STATUS FROM 'G7.STATS.TABLE'
   ```

   QMF commands with three-part names cannot be directed to DB2 for VSE and VM databases, nor can data be accessed remotely if you start QMF as a stored procedure.
6. To import a form for another user (JEAN) if you are the QMF administrator (QMFADM), issue the following command:

   ```
   IMPORT FORM JEAN.REPORT12 FROM FORMTEST (COMMENT='12 MONTH FORMAT')
   ```
7. To import data from the UNIX file `/u/DEPTJ49/pernal/mystaff.personnel`, issue the following command:

   ```
   IMPORT DATA FROM '/u/DEPTJ49/pernal/mystaff.personnel'
   ```

   Be sure that the CASE option of your QMF profile is set to STRING or MIXED to maintain the lowercase characters.

Related reference

CONNECT in TSO
You can use the CONNECT command from within a QMF session to connect to any database server that is part of the distributed network.

SET special register
You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

Global variables that control how commands and procedures are executed
DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

Related information

Exporting and importing objectsYou can write applications that issue QMF™ EXPORT and IMPORT commands to place objects outside of the QMF environment.

Exporting data or tables in XML formatIf your data or table contains an XML column or LOB data, you must use the DATAFORMAT=XML clause on the EXPORT DATA or EXPORT TABLE command.

Search for information about unsupported characters in the XML Toolkit for z/OS User's Guide, and about support for implicit casting with your database.

INSERT

The INSERT command inserts lines onto specific panels.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Specifically, the INSERT command inserts:

- A text line into a FORM.PAGE, FORM.FINAL, FORM.BREAKn, or FORM.DETAIL panel
- A column description line on a FORM.MAIN or FORM.COLUMNS panel
- A line for a report calculation expression on a FORM.CALC or FORM.CONDITIONS panel
- A line on an SQL query, Prompted Query, or PROC panel

Usage notes

- To insert a line at the top of the scrollable area, position the cursor directly above the first line and press the Insert key.
- To insert a calculation line into a FORM.CALC panel, position the cursor on the line above where you want to add the line, and press the Insert key. An alternative method is to type INSERT on the command line, position the cursor on the line above, then press Enter.
- You can insert a specification in a prompted query in one of the following ways:
  - In the echo area, position the cursor on the underscore character that appears to the left of the specification that is above where you want the new specification to appear and press Insert.
  - Type INSERT on the command line, then position the cursor on the underscore character that appears to the left of the specification that is above where you want the new specification to appear. Then press Enter.

INTERACT

The INTERACT command enables user interaction while a procedure or application is running.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Two forms of interaction are available:

Session
Begins an interactive dialog within the current QMF session

Command
Runs a single command in an interactive dialog

Session form of INTERACT

Notes:

[1] This form is valid for QMF procedures or callable interface applications.

Command form of INTERACT

Notes:

[1] Use with the command interface (DSQCCI). Has no effect when issued from the callable interface.

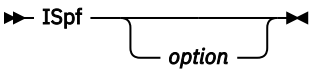
qmfcommand
The QMF command to be run.

ISPF

ISPF is a command synonym that is supplied by QMF. ISPF calls the Interactive System Product Facility (ISPF).

TSO with ISPF	TSO without ISPF	CICS
X		

Call ISPF from QMF



option
The initial option to pass to ISPF/PDF. For example, if you enter 3, the third ISPF panel option is selected directly.

If you do not specify an option, the ISPF/PDF primary option menu is displayed.

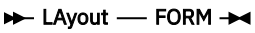
LAYOUT

The LAYOUT command generates a sample QMF report using just a QMF form as input. This can assist in the development of a QMF form by providing a visual rendering of a representative report.

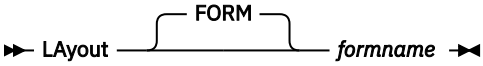
TSO with ISPF	TSO without ISPF	CICS
X		

LAYOUT is a command synonym for an ISPF application that is supplied by QMF. It analyzes the form and creates sufficient generic data to exercise the basic report features specified in the QMF form. No query is needed.

Lay out a QMF report using the form in temporary storage



Lay out a QMF report using a form from the database



Description

formname
The name of a QMF form in the database.

Usage notes

- You can use your sample form to display a report with various characters representing the data. If there are no breaks in the report, the following characters are displayed:

  X
  Character data

  0
  Numeric data

  If the report contains breaks, the break levels are shown using the following characters:

  A
  Character data in the first break

1
Numeric data in the first break

B
Character data in the second break

2
Numeric data in the second break

After you have seen what your form will look like, you can make changes to it without running a query.

- The LAYOUT command creates its data in QMF (binary) data format.
- The LAYOUT command is implemented as an ISPF application using the QMF command interface. The command prompt panel is defined using ISPF services and is allocated to ISPF as an ISPF panel.
- QMF updates the Last Used field for the object when you use this command. This field appears on object list panels displayed by the LIST command. You can change the list of commands that cause the field to be updated by setting the DSQEC_LAST_RUN global variable.

Examples

1. To display a prompt panel:

```
LAYOUT ?
```

2. To create a sample report using an existing form (MYFORM) in the database:

```
LAYOUT MYFORM
```

or

```
LAYOUT FORM MYFORM
```

3. To run the LAYOUT command using the form in temporary storage:

```
LAYOUT FORM
```

4. To enter the LAYOUT command from a QMF procedure, you must use delimited identifiers (double quotation marks) to continue a form object name across two or more lines in a QMF linear procedure. All continuation lines must have a plus sign (+) in column one, as shown in the following figure:

```
PROC                                                          MODIFIED LINE

LAYOUT TABLE
+"AUTHID_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
+XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"."OBJECT_NAMEXXXXXXXXXXXXXXX
+XXXXXXXXXXXXXXXXXXXXXX"
```

Figure 7. Entering the LAYOUT command from a QMF procedure

Related reference

Global variables that control how commands and procedures are executed
DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

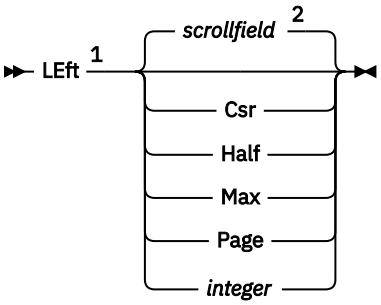
Related information

Exporting data or tables in QMF formatThe data file that you export by using the EXPORT command with the DATAFORMAT=QMF clause consists of two parts: header records, which describe the data in the records, and the data records, which contain the data.

LEFT

The LEFT command scrolls toward the left boundary of a report panel or a QBE query.

TSO with ISPF	TSO without ISPF	CICS
X	X	X



Notes:

[1] Specify scroll amounts only when there is a SCROLL field on the active panel. PAGE is assumed in all other situations.
[2] The value shown in the SCROLL field is used. This value is also maintained in the global variable DSQDC_SCROLL_AMT.

Description

CSR
Scrolls toward the left, repositioning the column in which the cursor lies to the right edge of the panel. If the cursor is at the left edge of the panel, LEFT CSR has the same effect as LEFT PAGE.

HALF
Scrolls toward the left half the width of the panel or to the left boundary (if that is nearer).

MAX
Scrolls to the left boundary of the panel.

PAGE
Scrolls toward the left the width of the panel or to the left boundary (if that is nearer).

integer
Scrolls toward the left this number of columns (a whole number ranging from 1 through 9999).

Usage notes

- MAX is in effect only for the current command. This value will not remain in the SCROLL field after the command completes. You cannot set the global variable DSQDC_SCROLL_AMT to this value.
- Use the LEFT function key to scroll left in a report. To specify a scroll amount, type the number of columns you want to scroll on the command line and then press the LEFT function key.

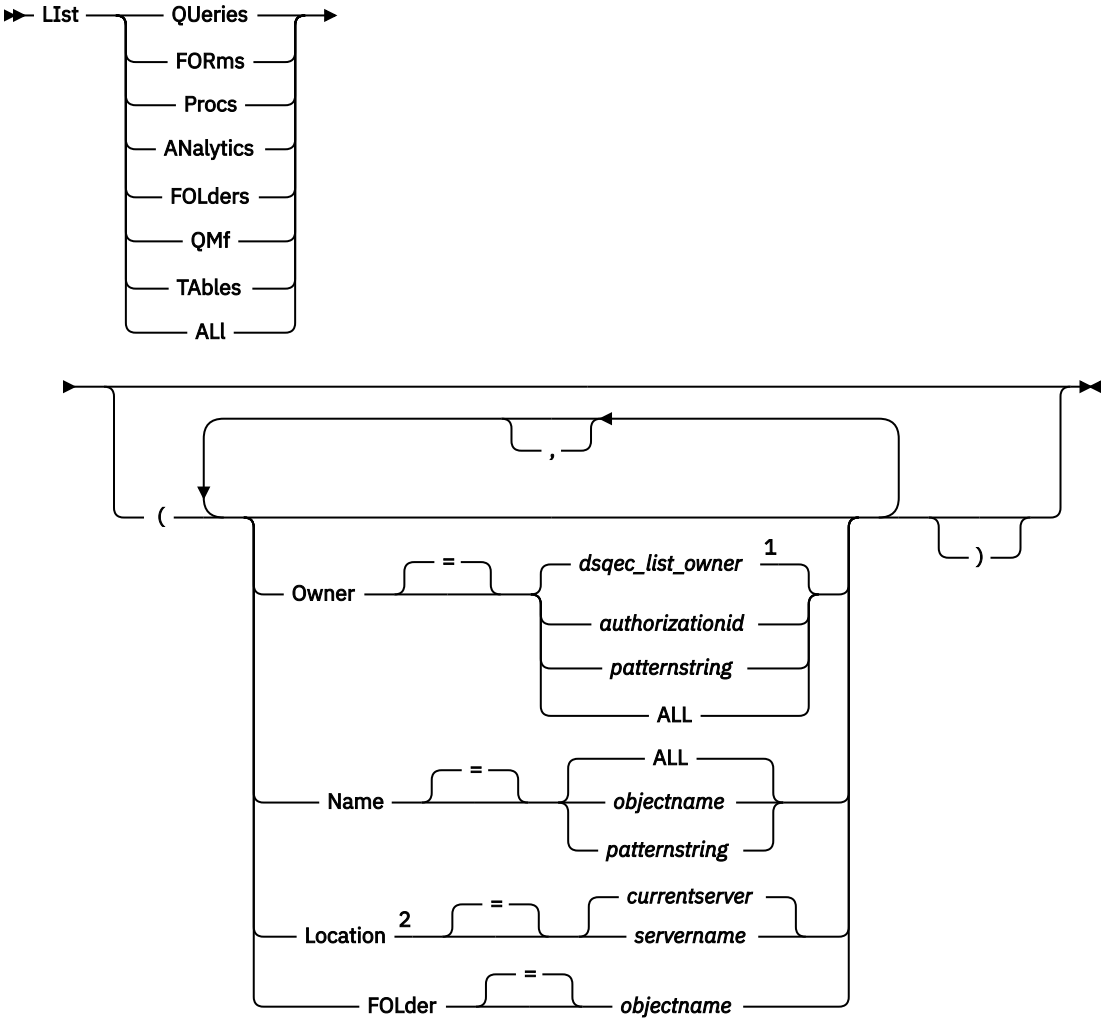
LIST

Use the LIST command to display lists of QMF objects and database tables stored in the database. When you first issue the LIST command in a QMF session, ensure that you use one of these parameters: Queries, Forms, Procs, Analytics, Folders, QMF, Tables, or All.

When you issue the LIST command without parameters, QMF displays the most recent list you requested.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Create a list of objects from the database



Notes:

[1] The value set in this global variable is used.
[2] Usage is limited to tables.

Display the current list of objects

Description

QUERIES
Lists only QMF queries.

FORMS
Lists only QMF forms.

PROCS
Lists only QMF procedures.

ANALYTICS
Lists only QMF analytics objects.

FOLDERS
Lists only QMF folders.

QMF
Lists only QMF objects: queries, forms, procedures, folders, and analytics objects.

TABLES
Lists only database table objects: aliases, history tables, tables, and views.

ALL
Lists all objects – QMF objects and database tables.

OWNER
Specifies the ownership qualifier for objects to list. The default is provided by the DSQEC_LIST_OWNER global variable.

authorizationid
The name of a user, a schema, or a database collection.

patternstring
Searches for owner names that have a certain pattern. The pattern is specified by a string in which the underscore and percent sign characters have special meanings.

ALL
Lists all objects that can be accessed by the current authorization ID, regardless of owner.

If the enhanced list function has been installed, privileges need be granted only to a user's primary or secondary authorization ID, instead of to PUBLIC, to be seen in the list when OWNER=ALL is specified. RACF group names can be used as secondary authorization IDs.

NAME
Specifies the name of an object to list.

ALL
Lists all objects regardless of name.

objectname
The name of a QMF object or a database table.

patternstring
Searches for object names that have a certain pattern. The pattern is specified by a string in which the underscore and percent sign characters have special meanings.

LOCATION
Specifies the location of objects to list. The current database server is the default.

servername
The name of a database application server in the distributed network.

LIST commands that include the LOCATION option can be initiated from and directed to Db2 for z/OS databases only. The QMF session is connected to a Db2 for z/OS database when the global variable DSQAO_DB_MANAGER has a value of 2.

FOLDER
Specifies the name of the folder to use with the LIST command. When the FOLDER keyword is specified, only the requested object types that exist in that folder are listed.

You can specify a folder name in the LIST command either by setting the DSQEC_CURR_FOLDER global variable or by specifying the FOLDER keyword in the command. A folder name that is specified with the FOLDER keyword overrides the folder name that is set in DSQEC_CURR_FOLDER.

The wildcard characters '%' and '_' are not allowed for OWNER, NAME, or FOLDER keywords when LIST folder content is requested. If the folder name contains a blank, the folder name must be surrounded in double quotes.

Usage notes

- QMF objects you do not own are listed only if they were saved with the option SHARE=YES.
- The pattern string used with the OWNER and NAME parameters can be specified as follows:
  - The % symbol represents a string of zero or more characters.
  - The _ symbol represents any single character.

For example, to list all QMF objects with owners that contain the character D in the second position, enter:

```
LIST QMF (OWNER=_D%
```

- The wildcard characters '%' and '_' are not allowed for OWNER, NAME, or FOLDER keywords when LIST folder content is requested.
- When you request a list of objects, QMF displays them in the default order: owner first, then name. To change the default list order, you change the DSQDC_LIST_ORDER global variable.

  The DSQDC_LIST_ORDER global variable is a two-character value. The first character specifies the sort characteristic and the second specifies whether the sort is ascending or descending. Changing the value of DSQDC_LIST_ORDER applies only to the current session. The default value is 1A.

  The values for the first character are shown in the following table:

Table 4. Sort sequences for the LIST command

Value	Characteristic (primary key)	Sort sequence
1	Default	Owner (current owner first) then name
2	Owner	Owner then name
3	Name	Name then owner
4	Type	Type, name, owner
5	Modified	Modified, last used, owner, name, type
6	Last used	Last used, modified, owner, name, type

  The second character can have the following specifications:

  A Ascending order
  D Descending order

  For example, to create a new list with the most recently modified objects at the top of the list, enter this SET GLOBAL command:

```
SET GLOBAL (DSQDC_LIST_ORDER=5D
```

  To create a new list with the current owner's objects at the top of the list, enter this SET GLOBAL command:

```
SET GLOBAL (DSQDC_LIST_ORDER=1A
```

  These examples do not change the order of an existing list.
- If you connected to a new location since you created the object list being displayed, your list is now obsolete. You must refresh the list or cancel it and create a new one. Commands issued in the `Action` column of an obsolete list are not executed.
- You cannot list queries, procedures, forms, folders, or analytics objects at a remote location using the Location parameter. To list these objects at a remote location, first connect to that location, then use the LIST command.
- When you request a list of tables, QMF uses views to retrieve the information:
  - If your current location is Db2 for z/OS and you request a list from that location (if LOCATION is not specified or is specified to be the current location), QMF uses the views named in the global variables DSQEC_ALIASES and DSQEC_TABS_LDB2.
  - If your current location is Db2 for z/OS and you request a list from a different Db2 for z/OS location, QMF uses the views named in the global variables DSQEC_ALIASES and DSQEC_TABS_RDB2.

  - If your current location is DB2 Server for VSE and VM, QMF uses the view named in the global variable DSQEC_TABS_SQL.
- Issuing the LIST command when operating in QMF batch mode or when QMF has been started as a stored procedure results in an error. Also, to issue LIST through the QMF command interface, LIST must be preceded by INTERACT (unless it is used from a Prompted Query dialog panel).
- You can use single quotation marks with the LIST command anywhere the command is valid. You cannot use double quotation marks with the LIST command.

Examples

1. To get a list of all the forms that you own, enter:

```
LIST FORMS
```

2. To get a list of all the queries that begin with the name APP1, regardless of owner, enter:

```
LIST QUERIES (OWNER=ALL NAME=APP1%
```

3. To get a list of all the tables at a database server named DALLASDB, regardless of owner, with the characters CUST in the name, enter:

```
LIST TABLES (LOCATION=DALLASDB OWNER=ALL NAME=%CUST%
```

4. To get a list of all QMF procedures in a folder named YE2014, enter:

```
LIST PROCS (FOLDER=YE2014
```

5. When using the LIST command in a QMF linear procedure, you must use single quotation marks to continue an authorization ID across more than one line. All continuation lines must have a plus sign (+) in column one, as shown in the following figure:

```
PROC                                          MODIFIED  LINE     1
 LIST ALL (
 + OWNER='LONGOWNERID99999999999999999999999999999999999999999999999999999999999
 +99999999999999999999999999999999999999999999999999%'
 + NAME='LONGNAME99999999999999999999999999999999999999999999999999999999999999
 +99999999999999999999999999999999999999999999999999'
```

Figure 8. Continuing an authorization ID across more than one line within a QMF linear procedure

Related reference

Global variables that control how commands and procedures are executed

DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

MESSAGE

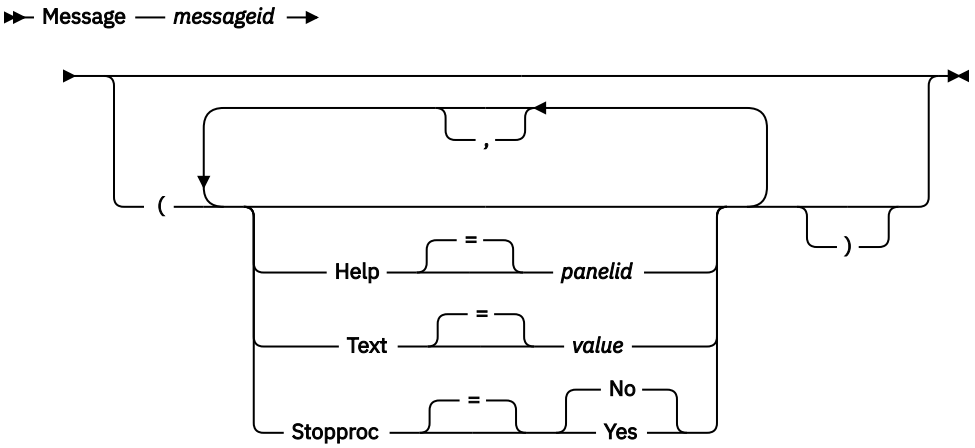
Use the MESSAGE command from user applications (procedures, programs, execs, and CLISTs) to pass a message to the QMF message area.

TSO with ISPF	TSO without ISPF	CICS
X	*	*

With the MESSAGE command you can:

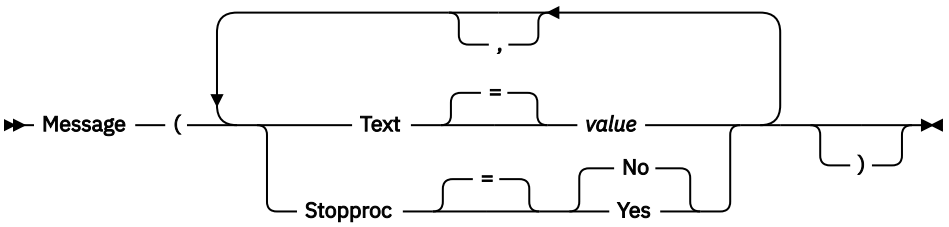
- Display a message from the ISPF library
- Assign a help panel for an ISPF message
- Generate a QMF-like message
- Suppress execution of QMF linear procedures

Display a message defined to ISPF



Notes:

Generate a QMF-like message



Description

messageid
The identification number of a message definition in an ISPF message library. The designated library must be concatenated to your ISPMLIB data set.

HELP
Specifies the help panel to accompany the message. This option will override the tutorial help panel specified in the ISPF message definition.

panelid
The name of a panel in an ISPF panel library. The designated library must be concatenated to your ISPPLIB data set.

TEXT
Defines the message text. Message text up to 360 single-byte characters long can be issued with this option.

When used with an ISPF message ID, this option will override the long message specified in the ISPF message definition.

value
The character string that makes up the content of the message.

A value that contains blank characters must be surrounded with delimiters. Valid delimiters for a message value are single quotation marks, parentheses, and double quotation marks.

If the text spans multiple lines:

- In a linear procedure, place a + character at the beginning of each line to indicate continuation.
- In a procedure with logic, place a comma at the end of each line except the last.

STOPPROC
Sets a termination switch for QMF linear procedures. The setting remains active until the current application ends or the setting is changed again by the application.

YES
Sets the procedure termination switch on. Any QMF linear procedure receiving control ends its execution immediately.

NO
Sets the procedure termination switch off. QMF linear procedure execution is not suppressed.

Usage notes

- The MESSAGE command cannot be issued from the QMF command line. It can only be issued from a QMF procedure or an application using the QMF command or callable interface.
- The STOPPROC option has limited usage within a linear procedure. Once the procedure termination switch is set on, the procedure will end immediately.

Examples

1. To display ISPF message ISPG053 with your own help panel (called CMDHELP):

```
MESSAGE ISPG053 (HELP=CMDHELP
```

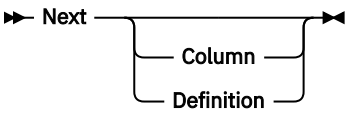
2. To issue a QMF-like message:

```
MESSAGE (TEXT=(Sales report for YE '05 is complete.)
```

NEXT

Use the NEXT command to navigate forward through the set of variations associated with the FORM.DETAIL panel. You can also use the NEXT command to display the next column or the next definition from the Column Definition or Column Alignment panel, or to display the next row in the set of accessed rows in the Table Editor.

TSO with ISPF	TSO without ISPF	CICS
X	X	X



Description

COLUMN
Displays the next column from the Column Definition or Column Alignment panel.

DEFINITION
Displays the next column with a non-blank definition expression from the Column Definition panel.

Usage notes

- Column definition requires REXX facilities and is not supported in CICS.
- The COLUMN and DEFINITION parameters:
  - Provide direct panel navigation on active column definition or alignment panels.
  - Are not normally entered on the command line or from an application, although they can be.
- On a FORM.DETAIL panel, the NEXT command:
  - Displays the next panel variation (unless it would result in an error).
  - Can be entered from the command line, by pressing a function key, or from an application.

- In the Table Editor, the NEXT command can only be entered using a function key.

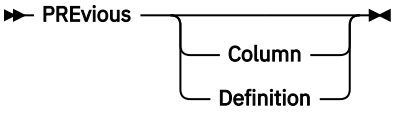
PREVIOUS

Use the PREVIOUS command to navigate backward.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

The PREVIOUS command:

- Navigates backward through the set of variations that are associated with the FORM.DETAIL panel.
- Displays the previous column or the previous definition when the form definition is displayed.
- Displays the row just added (if you are in Add mode) or the most recent successful search criteria (if you are in Search mode) in a Table Editor session.



Description

COLUMN
The previous column is displayed from the **Column Definition** or **Column Alignment** panel.

DEFINITION
The most recent column with a non-blank definition expression is displayed when in the **Definition** panel.

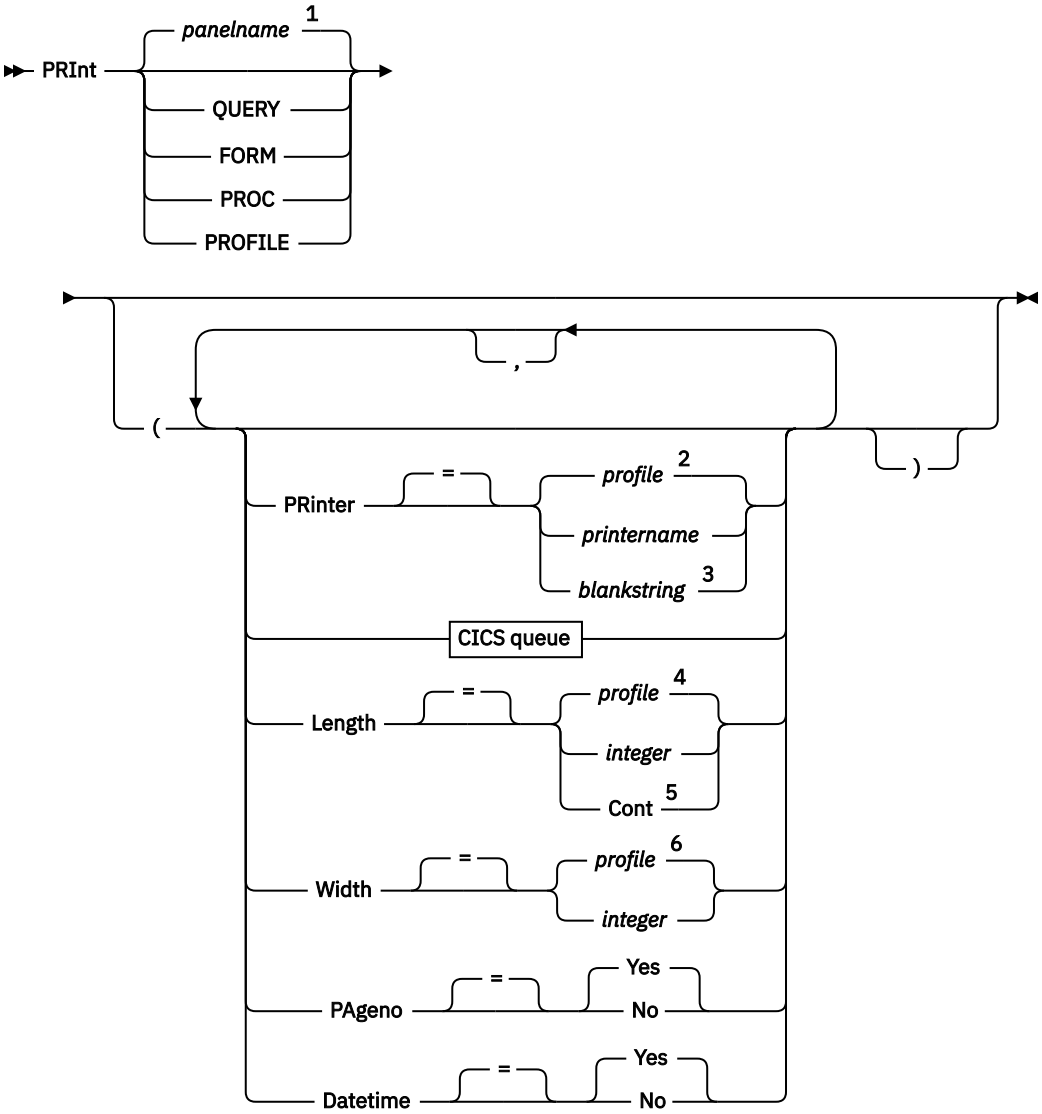
Usage notes

- Column definition requires REXX facilities and is not supported in CICS.
- The Column and Definition parameters provide direct panel navigation on active column definition or alignment panels.
- On a FORM.DETAIL panel, the PREVIOUS command:
  - Displays the previous panel variation (unless it would result in an error).
  - Can be entered from the command line, by pressing a function key, or from an application.
- In the Table Editor, the PREVIOUS command can be entered only by using a function key.

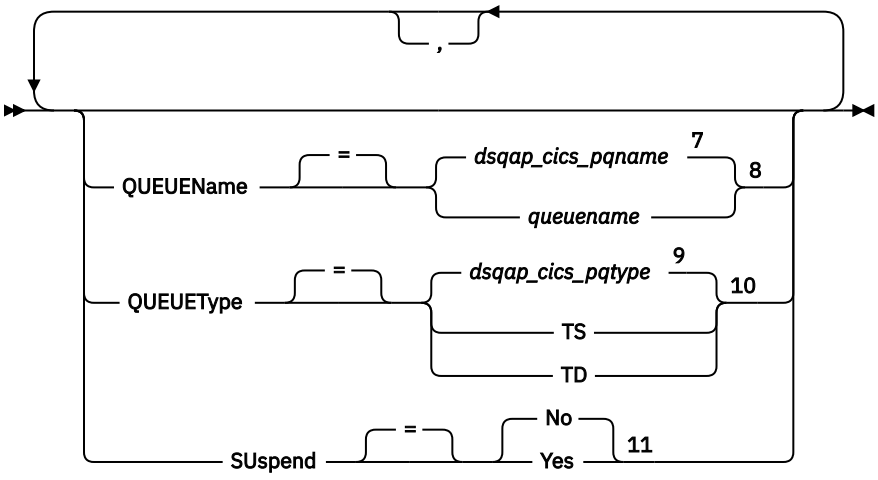
PRINT in CICS

The PRINT command in CICS prints a copy of an object in the QMF temporary storage area or an object that is stored in the database.

PRINT a QMF object from temporary storage



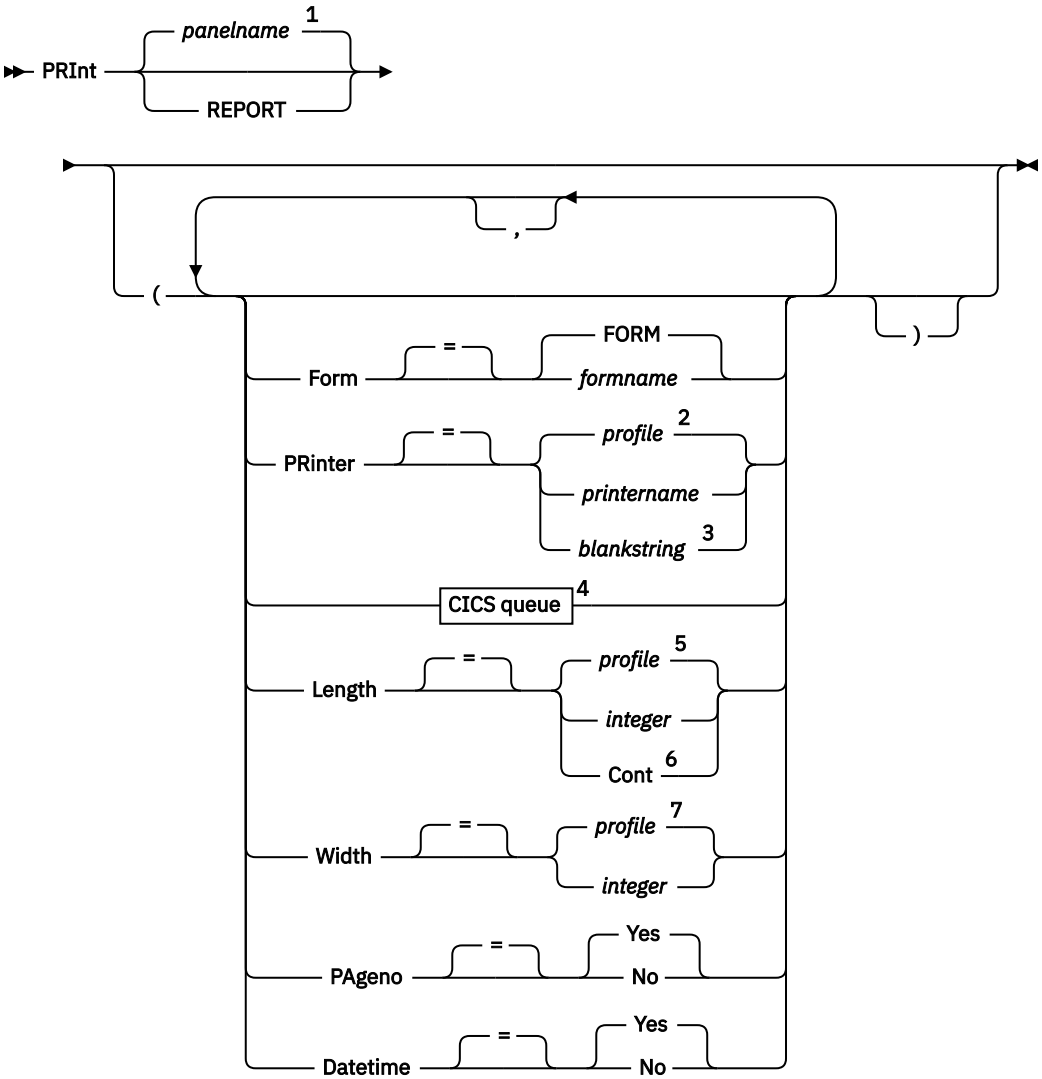
CICS queue



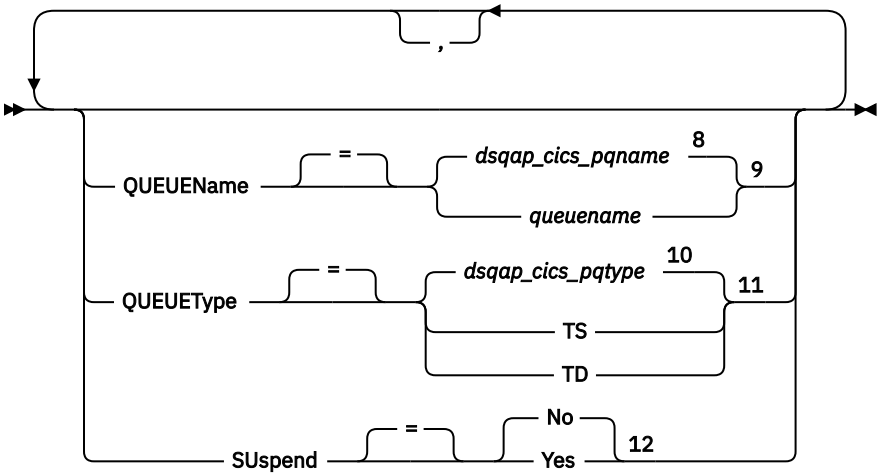
Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The value set in your profile is used.
[3] Use of this option is limited. Refer to the description that follows.
[4] The value set in your profile is used.
[5] Use of this option is limited. Refer to the description that follows.
[6] The value set in your profile is used.
[7] The value set in this global variable is used.
[8] The value set in this global variable is used.
[9] The value set in this global variable is used.
[10] The value set in this global variable is used.
[11] Use of this option is limited. Refer to the description that follows.

PRINT a QMF report from temporary storage



CICS queue

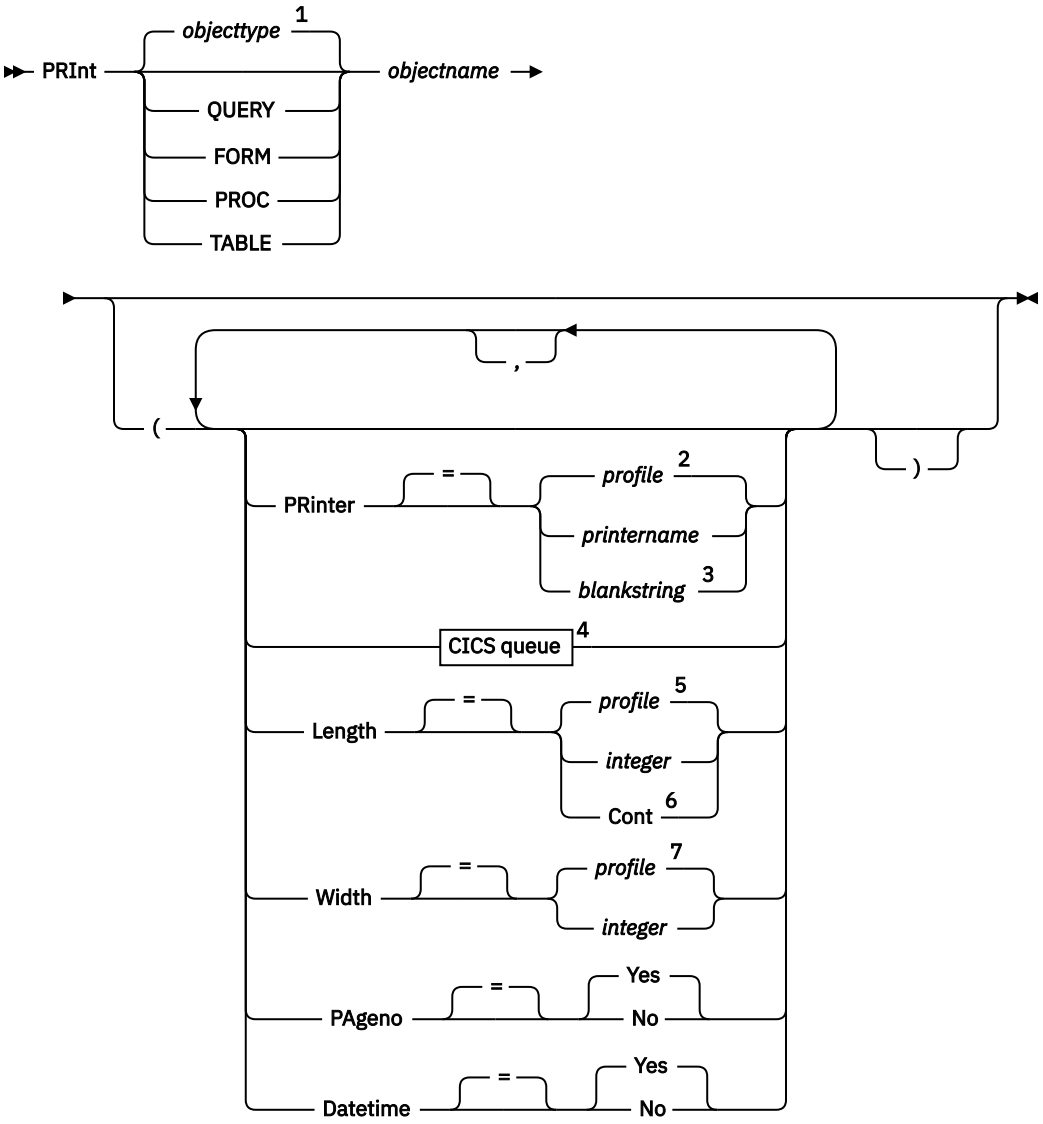


Notes:

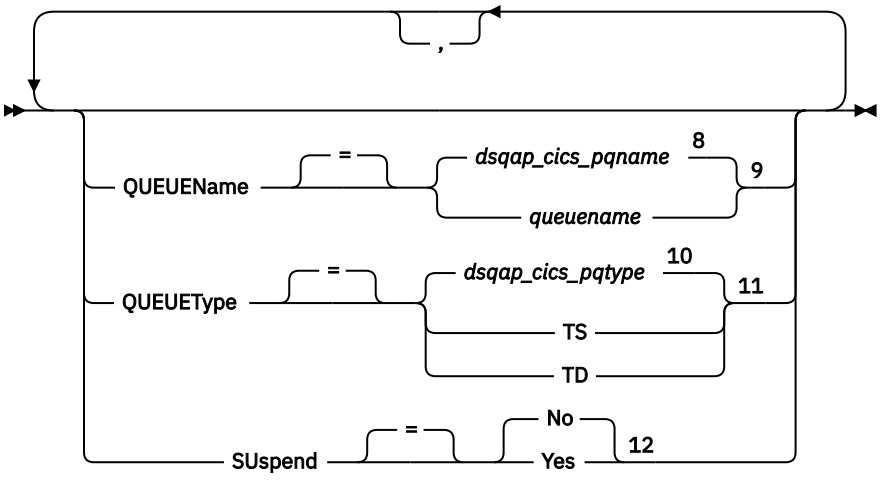
[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The value set in your profile is used.
[3] Use of this option is limited. Refer to the description that follows.

[4] Use of this option is limited. Refer to the description that follows.
[5] The value set in your profile is used.
[6] Use of this option is limited. Refer to the description that follows.
[7] The value set in your profile is used.
[8] The value set in this global variable is used.
[9] Use of this option is limited. Refer to the description that follows.
[10] The value set in this global variable is used.
[11] Use of this option is limited. Refer to the description that follows.
[12] Use of this option is limited. Refer to the description that follows.

PRINT an object from the database



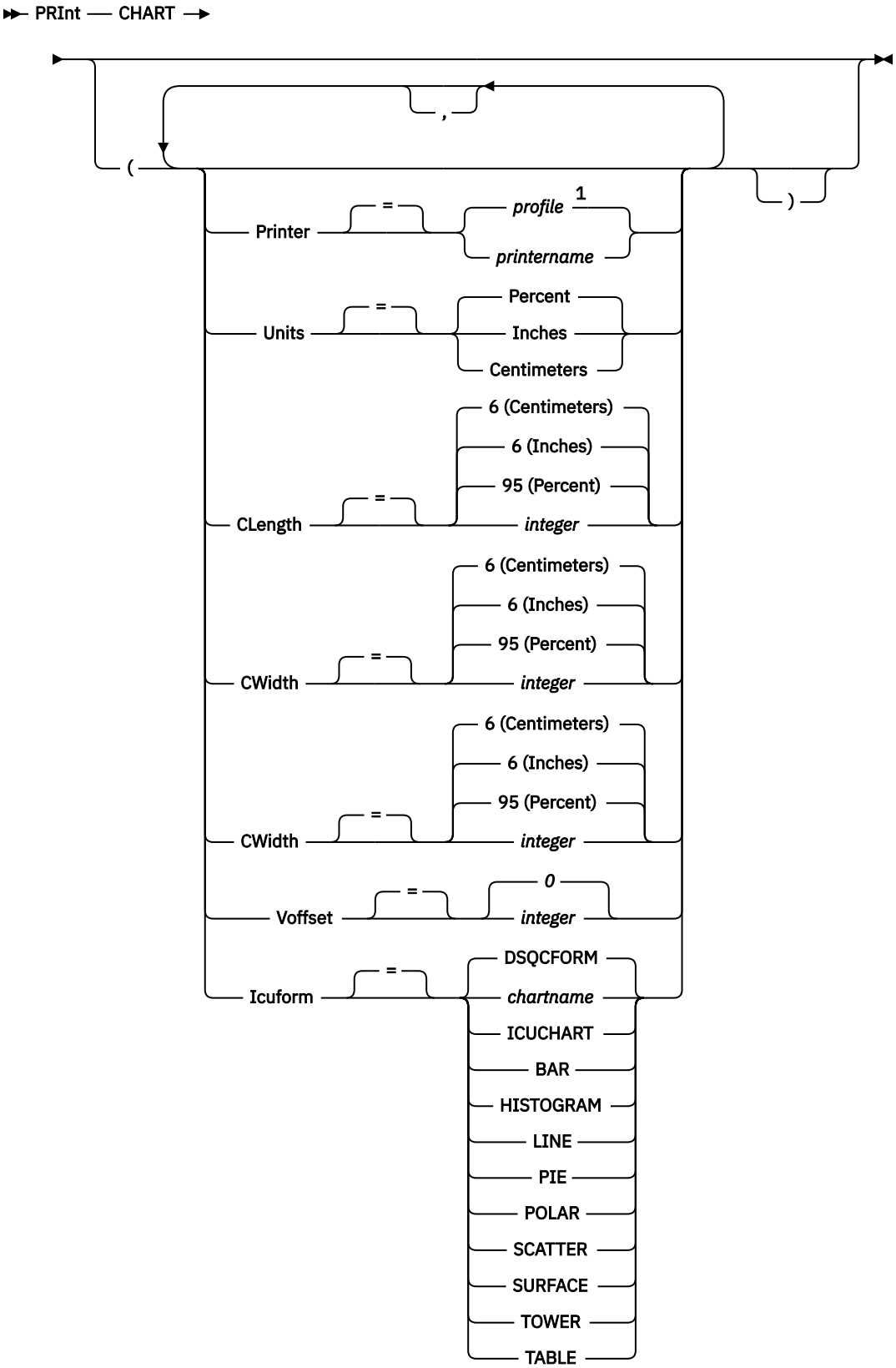
CICS queue



Notes:

[1] The type of the named object, if appropriate, is used. QMF objects have priority over other types of objects (such as database objects).
[2] The value set in your profile is used.
[3] Use of this option is limited. Refer to the description that follows.
[4] Use of this option is limited. Refer to the description that follows.
[5] The value set in your profile is used.
[6] Use of this option is limited. Refer to the description that follows.
[7] The value set in your profile is used.
[8] The value set in this global variable is used.
[9] Use of this option is limited. Refer to the description that follows.
[10] The value set in this global variable is used.
[11] Use of this option is limited. Refer to the description that follows.
[12] Use of this option is limited. Refer to the description that follows.

PRINT a chart



Notes:

[1] The value set in your profile is used.

Description

objectname
The name of an object in the database. Valid objects include:

- QMF objects (PROC, QUERY, FORM)
- Table objects (TABLE, VIEW, SYNONYM, ALIAS)

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

PRINTER
Specifies the output destination for the PRINT command.

printername
Specifies a printer destination. The value must be the nickname of a GDDM printer.

blankstring
Specifies a queue destination. This value must be indicated by a string of 0 to 8 blanks that are enclosed in single quotation marks (' ').

This option is not valid for chart, form, or prompted query objects.

These options are valid only when you print to a queue destination (when option PRINTER=blankstring is specified).

QUEUENAME
Specifies the CICS data queue to receive the printed object. The default is the current value of the QMF global variable DSQAP_CICS_PQNAME.

queuename
The name of a CICS data queue. The type of storage for the queue must match the type that is specified with the QUEUETYPE parameter.

QUEUETYPE
Identifies the type of CICS storage that is used for the CICS data queue that is specified by the QUEUENAME parameter. The default is the current value of the QMF global variable DSQAP_CICS_PQTYPE.

TS
Specifies a CICS temporary storage queue on an auxiliary device.

TD
Specifies a CICS transient data queue.

SUSPEND
Specifies the action to take when the data queue is busy and unavailable.

NO
Cancels the print request.

YES
Waits until the data queue is available.

LENGTH
Specifies the length of a printed page. The unit of length is one line.

integer
Specifies the maximum number of lines between page breaks. The number must be an integer from 1 to 999.

Minimum lengths apply to the objects shown in the following table:

Table 5. Objects and their minimum lengths when printing

Object	Minimum Length
Form	25

Table 5. Objects and their minimum lengths when printing (continued)

Object	Minimum Length
SQL query	25
Procedure	25
Prompted query	25
Table	8
QBE query	7 (5 when you print to a data set)
Profile	7 (5 when you print to a data set)

The minimum length for a report varies with the form used and the value of the command options DATETIME and PAGENO.

The maximum length of a printed form is 66.

CONT
Specifies continuous printing, without page breaks.

This option is not valid for chart, form, or prompted query objects, or whenever a printer name is specified.

WIDTH
Specifies the width of a printed page. The unit of width is one single-byte character.

integer
Specifies the maximum number of characters to be printed on any line. The number must be an integer from 22 to 999.

Lines wider than the value specified are cut off on the right, unless the object you are printing is a report. In that case, lines longer than the value specified are formatted on a subsequent page, unless you specified line wrapping on the FORM.OPTIONS panel.

PAGENO
Specifies the inclusion of page numbers with the printed object.

This option is ignored when you print a report and the form contains the variable &PAGE.

YES
Page numbers are included at the bottom of the page.

NO
Page numbers are suppressed.

DATETIME
Specifies the inclusion of the system date and time on each page of the printed object.

This option is ignored when you print a report and the form contains the variable &DATE or &TIME.

YES
Date and time are included at the bottom of the page.

NO
Data and time are not included.

FORM
Specifies the form to use when you print a report.

FORM
The current form object in temporary storage. This value is the default.

formname
The name of a QMF form in the database. This form replaces the current form in temporary storage.

UNITS
Specifies the unit of measure for chart dimension parameters CLENGTH, CWIDTH, HOFFSET, and VOFFSET.

PERCENT
Chart dimensions are relative to the screen size (100 percent).

CENTIMETERS
Chart dimensions are expressed in centimeters.

INCHES
Chart dimensions are expressed in inches.

CLENGTH
The length of the chart area expressed as a number. The unit of measure is determined by the UNITS parameter. The default varies with the unit of measure.

CWIDTH
The width of the chart area expressed as a number. The unit of measure is determined by the UNITS parameter. The default varies with the unit of measure.

HOFFSET
The horizontal offset of the chart from the left side of the page, expressed as a number. The unit of measure is determined by the UNITS parameter.

VOFFSET
The vertical offset of the chart from the top of the page, expressed as a number. The unit of measure is determined by the UNITS parameter.

ICUFORM
Specifies the name of a chart format. A chart format contains the specifications that are required to turn data into a chart. Different formats are used to produce different types of charts.

DSQCFORM
The name of the default chart format that is provided by QMF.

This format can be customized by your administrator. It provides a bar chart if not customized.

chartname
The name of a chart format.

ICUCHART
Specifies the default chart format for the GDDM Interactive Chart Utility.

BAR
HISTOGRAM
LINE
PIE
POLAR
SCATTER
SURFACE
TOWER
TABLE
The name of a chart format that is provided by QMF.

Usage notes

- When you print a form, all parts of the form are printed.
- When you print a report, the report is printed according to the form specifications.
- When you print a table, the table is formatted using a default form.

  To print a table that is formatted with a form other than the default form, display the table, display the form that you want, and then issue the PRINT REPORT command.

However, if the form requires that the rows of data be in sorted order (for example, the form uses breaks), you must first run a query that selects data from the table in sorted order rather than display the table.

- When you print a chart, the form specifications are applied to the data and the chart is formatted by the GDDM Interactive Chart Utility.
- To print to a data queue, use the QUEUENAME parameter to name a CICS extrapartition transient data queue (QUEUETYPE=TD). The CICS DCT (destination control table) must first have a definition for the data queue that routes the output to a data queue.
- When you print a report or chart, if the form contains errors, the form panel on which the first error was found is displayed, and the error is highlighted. To see other errors, you must correct the first error displayed.

  Some errors are not detected until you create a report.
- With a DBCS printer, you can print reports that contain DBCS data even if you do not have a terminal that displays DBCS data. Start QMF with the program parameter DSQSDBCS set to YES. Contact your administrator for details on customizing your QMF start procedure.
- If you are using DBCS data and QMF splits the page, printing resumes on the second and subsequent pages of the report at the fourth-byte position from the left side of the page.
- The page number, date, and time can be included in the chart title by specifying &PAGE, &DATE, and &TIME, respectively, on the FORM.PAGE panel.
- If you are using a three-part name to print a table and your database administrator set up QMF to use the multirow fetch feature, both databases you are working with (local and remote) must be Db2 for z/OS; otherwise, your command fails. Your database administrator can turn off multirow fetch.

  QMF commands with three-part names cannot be directed to DB2 for VSE and VM databases.

  By default, three-part names cannot be used to access remote tables that contain LOB data. However, you can set the DSQEC_LOB_RETRV global variable to 2 or 3 to access LOB metadata or data with a three-part name. Or, you can use the CONNECT command to connect to the database, and then run the query to access the remote table.
- You can use the PRINT TABLE command to print from a QMF Data Service server. Use a three part name format and ensure that the DSQEC_DS_SUPPORT global variable is set.
- The maximum length of a row that can be printed depends on the type of object:
  - Printing a table or printing a view that is based on a single table

    Db2 stores records within pages that are 4 KB, 8 KB, 16 KB, or 32 KB in size. Because you cannot create a table with a maximum record size that is greater than the page size, the maximum length of a data row that can be printed is 32 KB when you print a single table. For tables that contain LOB or XML columns, each data row contains a locator or pointer that references the location of the data. The data itself is not stored as part of the record and metadata is printed for these types of columns when you issue the PRINT TABLE command.
  - Printing a report or printing a view that is based on two or more tables

    The maximum length of a data row that can be printed from a view that joins two or more tables or from a QMF report in temporary storage is 2 GB when the DSQEC_TWO_GB_ROW global variable is set to 1. When the variable is set to 0, all rows except those that contain LOB or XML columns are truncated at 32 KB. Up to 2 GB of XML, CLOB, and BLOB data and up to 1 GB of DBCLOB data can be printed regardless of the DSQEC_TWO_GB_ROW setting.

  Operations with XML data typically require larger amounts of storage, so printing reports or tables that contain XML data might be limited by the amount of storage you have available.
- The PRINT TABLE command prints XML metadata rather than XML data. By default, the PRINT REPORT command prints XML metadata rather than XML data unless you change the M edit code.
- The behavior of the PRINT REPORT and PRINT TABLE commands for LOB data is controlled by the DSQEC_LOB_RETRV global variable as follows:

  - When the DSQEC_LOB_RETRV global variable is set to 1, LOB metadata is printed by default. You can print LOB data by changing the default M edit code.
  - When the DSQEC_LOB_RETRV global variable is set to 2, LOB metadata is printed and the default M edit code cannot be changed.
  - hen the DSQEC_LOB_RETRV global variable is set to 3, LOB data is printed instead of LOB metadata.
- QMF updates the **Last Used** field for the object when you use this command. This field appears on object list panels that are displayed by the LIST command. You can change the list of commands that cause the field to be updated by setting the DSQEC_LAST_RUN global variable.
- If a PRINT TABLE command is directed to a Unicode database and the table contains columns that have graphic data types, QMF casts the data to other types to avoid errors.
- When you issue a PRINT TABLE command that references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. You set the value of this register by using the SET CURRENT SCHEMA statement.

A printed report differs from a report that is displayed on a screen in the ways that are shown in the following table:

Table 6. Differences between displayed and printed reports

Part of report	Displayed report	Printed report
Number of pages	One page that can be scrolled	One or more pages
Page headings and footings	Appear only once	Appear at the top and bottom of each page
Detail headings	Before the first detail line at the beginning of a report and on every screen that follows	Before the first detail line at the beginning of a report and on every page that follows
Fixed column	Remain in place when report is scrolled horizontally	Repeated on the left side of each page

Examples

1. To display a prompt panel for the QMF PRINT command:

```
PRINT ?
```

2. To print a table that is formatted with a form other than the default form:

```
DISPLAY tablename
DISPLAY formname
PRINT REPORT
```

Related concepts

How QMF recasts certain data types when displaying data

When a DISPLAY TABLE command is directed to a Unicode database and the table referenced in the command contains columns with graphic data types, QMF converts the graphic data types to character data types.

Related reference

SET special register

You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

Global variables that control how commands and procedures are executed

DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

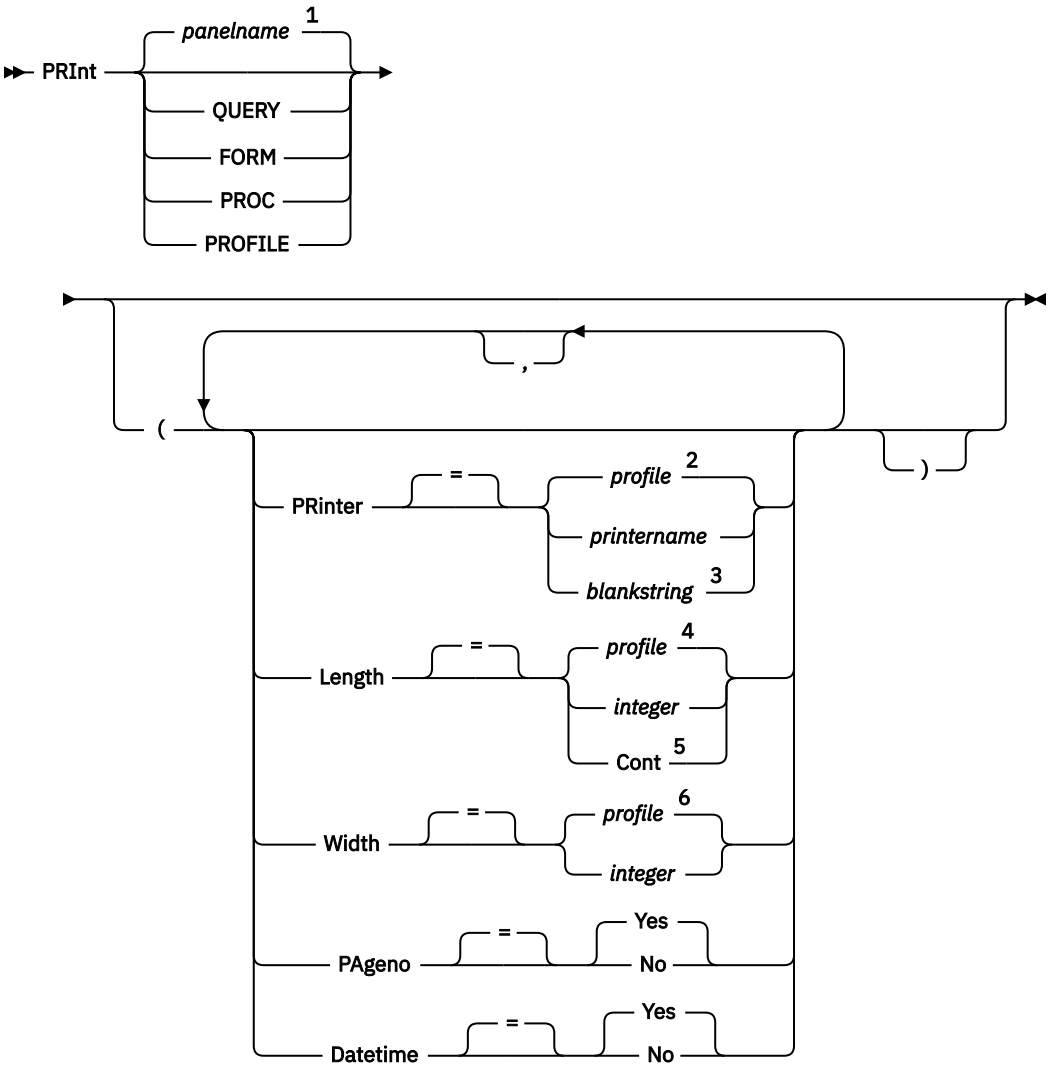
PRINT in TSO

The PRINT command prints a copy of an object in the QMF temporary storage area or an object that is stored in the database.

TSO with ISPF	TSO without ISPF
X	X

Syntax

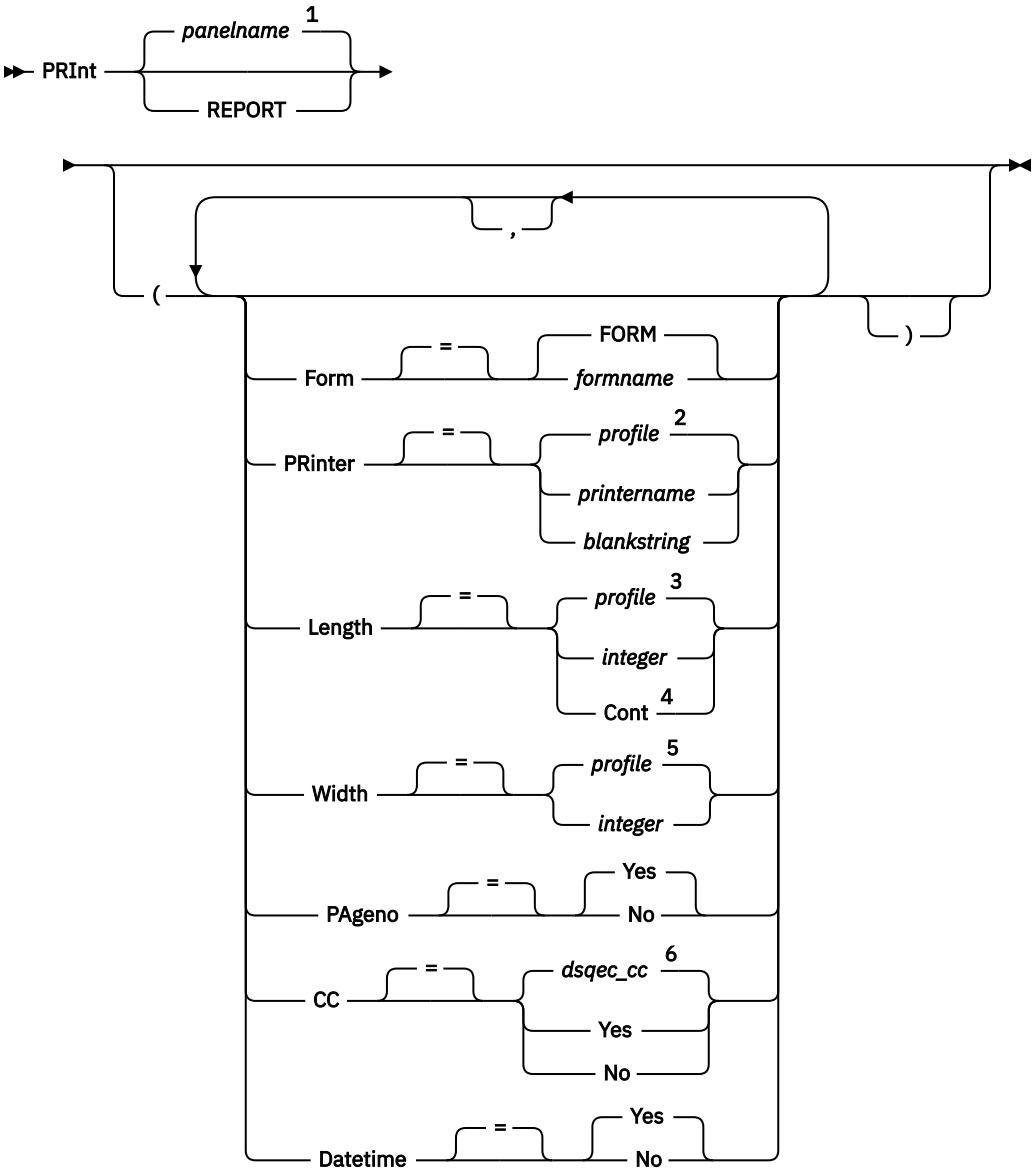
PRINT a QMF object from temporary storage



Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The value set in your profile is used.
[3] Use of this option is limited. Refer to the description that follows.
[4] The value set in your profile is used.
[5] Use of this option is limited. Refer to the description that follows.
[6] The value set in your profile is used.

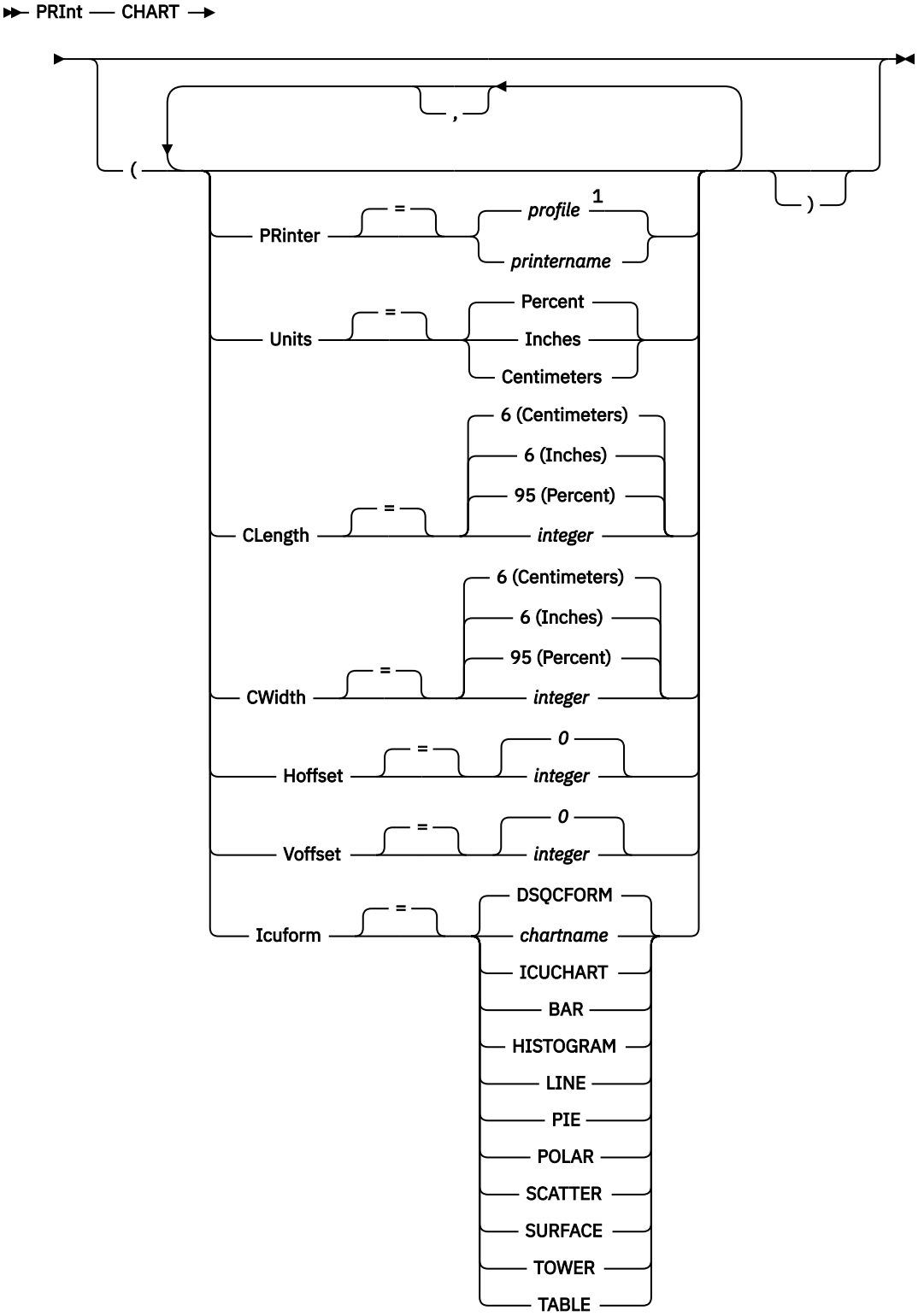
PRINT a QMF report from temporary storage



Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The value set in your profile is used.
[3] The value set in your profile is used.
[4] Use of this option is limited. Refer to the description that follows.
[5] The value set in your profile is used.
[6] Set *dsqec_cc* to 1 (where cc is in effect) to obtain a carriage control character in column 1 of the report; set this global variable to 0 for no carriage control character.

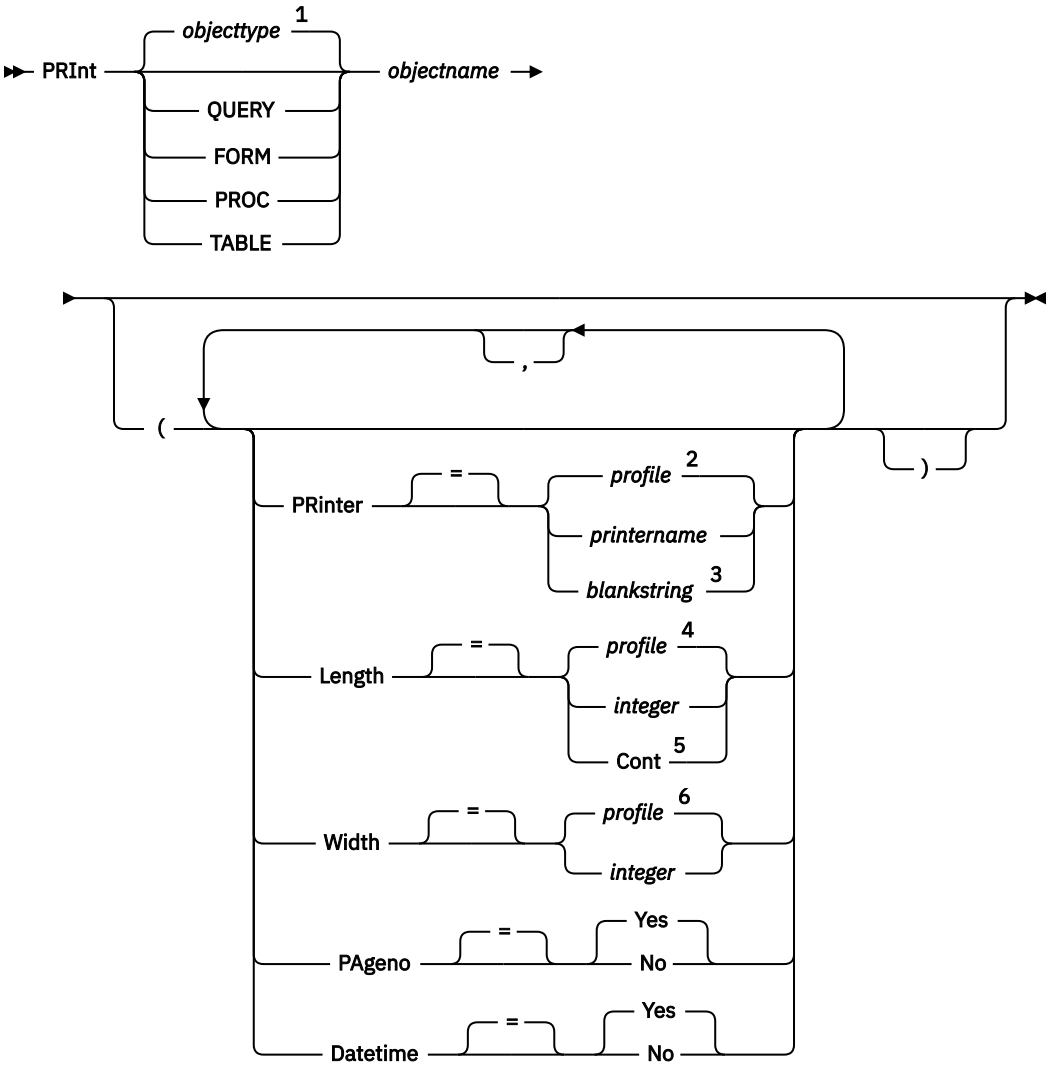
PRINT a chart



Notes:

[1] The value set in your profile is used.

PRINT an object from the database



Notes:

[1] The type of the named object, if appropriate, is used. QMF objects have priority over other types of objects (such as database objects).
[2] The value set in your profile is used.
[3] Use of this option is limited. Refer to the description that follows.
[4] The value set in your profile is used.
[5] Use of this option is limited. Refer to the description that follows.
[6] The value set in your profile is used.

Description

objectname
The name of an object in the database. Valid objects include:

- QMF objects (PROC, QUERY, FORM)
- Table objects (TABLE, VIEW, SYNONYM, ALIAS)

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

PRINTER
Specifies the output destination for the PRINT command.

printername
Specifies a printer destination. This value must be the nickname of a GDDM printer.

blankstring
Specifies a file destination. This value must be indicated by a string of 0 to 8 blanks that are enclosed in single quotation marks (' ').

The physical destination for the print output is a data set or a device that is allocated to the QMF DSQPRINT data set. Contact your administrator for details specific to your QMF environment.

Use a string of blanks for the PRINTER option when you start QMF for TSO as a Db2 for z/OS stored procedure and you want to receive output back in a result set.

This option is not valid for chart, form, or prompted query objects.

LENGTH
Specifies the length of a printed page. The unit of length is one line.

integer
Specifies the maximum number of lines between page breaks. The number must be an integer from 1 to 999.

Minimum lengths apply to the objects shown in the following table:

Table 7. Objects and their minimum lengths when printing

Object	**Minimum Length**
Form	25
SQL query	25
Procedure	25
Prompted query	25
Table	8
QBE query	7 (5 when you print to a data set)
Profile	7 (5 when you print to a data set)

The minimum length for a report varies with the form used and the value of the command options DATETIME and PAGENO.

The maximum length of a printed form is 66.

CONT
Specifies continuous printing, without page breaks.

This option is not valid for chart, form, or prompted query objects, or whenever a printer name is specified.

WIDTH
Specifies the width of a printed page. The unit of width is one single-byte character.

integer
Specifies the maximum number of characters to be printed on any line. The number must be an integer from 22 to 999.

Lines wider than the value specified are cut off on the right, unless the object you are printing is a report. In that case, lines longer than the value specified are formatted on a subsequent page, unless you specified line wrapping on the FORM.OPTIONS panel.

PAGENO
Specifies the inclusion of page numbers with the printed object.

This option is ignored when you print a report and the form contains the variable &PAGE.

YES
Page numbers are included at the bottom of the page.

NO
Page numbers are suppressed.

DATETIME
Specifies the inclusion of the system date and time on each page of the printed object.

This option is ignored when you print a report and the form contains the variable &DATE or &TIME.

YES
Date and time are included at the bottom of the page.

NO
Data and time are not included.

FORM
Specifies the form to use when you print a report.

FORM
The current FORM object in temporary storage. This is the default.

formname
The name of a QMF form in the database. This form will replace the current form in temporary storage.

UNITS
Specifies the unit of measure for chart dimension parameters CLENGTH, CWIDTH, HOFFSET, and VOFFSET.

PERCENT
Chart dimensions are relative to the screen size (100 percent).

CENTIMETERS
Chart dimensions are expressed in centimeters.

INCHES
Chart dimensions are expressed in inches.

CLENGTH
The length of the chart area, expressed as a number. The unit of measure is determined by the UNITS parameter. The default varies with the unit of measure.

CWIDTH
The width of the chart area expressed as a number. The unit of measure is determined by the UNITS parameter. The default varies with the unit of measure.

HOFFSET
The horizontal offset of the chart from the left side of the page, expressed as a number. The unit of measure is determined by the UNITS parameter.

VOFFSET
The vertical offset of the chart from the top of the page, expressed as a number. The unit of measure is determined by the UNITS parameter.

ICUFORM
Specifies the name of a chart format. A chart format contains the specifications that are required to turn data into a chart. Different formats are used to produce different types of charts.

DSQCFORM
The name of the default chart format that is provided by QMF.

This format can be customized by your administrator. It provides a bar chart if not customized.

chartname
The name of a chart format.

ICUCHART
Specifies the default chart format for the GDDM Interactive Chart Utility.

BAR
HISTOGRAM
LINE
PIE
POLAR
SCATTER
SURFACE
TOWER
TABLE
The name of a chart format that is provided by QMF.

Usage notes

- This command does not apply to QMF Analytics for TSO. To print a QMF Analytics for TSO chart or statistical analysis, first generate or display the chart or analysis, and then use the Print function key to print it. For more information, press the Help key from the Print panel in QMF Analytics for TSO.
- When you print a form, all parts of the form are printed.
- When you print a report, the report is printed according to the form specifications.
- When you print a table, the table is formatted using a default form.

  You can override the default formatting by setting the following global variables:

  DSQDC_EC_DATE
  DSQDC_EC_TIME
  DSQDC_EC_CHAR
  DSQDC_EC_NUM
  DSQDC_EC_DEC

  To print a table that is formatted with any form other than the default form, display the table, display the form that you want, and then issue the PRINT REPORT command.

  However, if the form requires that the rows of data be in sorted order (for example, the form uses breaks), you must first run a query that selects data from the table in sorted order rather than display the table.
- When you print a chart, the form specifications are applied to the data and the chart is formatted by the GDDM Interactive Chart Utility.
- When you print a report or chart and the form contains errors, the form panel on which the first error was found is displayed, and the error is highlighted. To see other errors, you must correct the first error displayed.

  Some errors are not detected until you create a report.
- With a DBCS printer, you can print reports that contain DBCS data even if you do not have a terminal that displays DBCS data. Start QMF with the program parameter DSQSDBCS set to YES. Contact your administrator for details on customizing your QMF start procedure.
- If you are using DBCS data and QMF splits the page, printing resumes on the second and subsequent pages of the report at the fourth-byte position from the left side of the page.
- The page number, date, and time can be included in the chart title by specifying &PAGE, &DATE, and &TIME, respectively, on the FORM.PAGE panel.
- If you are using a three-part name to print a table and your database administrator set up QMF to use the multirow fetch feature, both databases you are working with (local and remote) must be Db2 for z/OS; otherwise, your command fails. Your database administrator can turn off multirow fetch.

  QMF commands with three-part names cannot be directed to DB2 for VSE and VM databases, nor can data be accessed remotely if you start QMF as a stored procedure.

  By default, three-part names cannot be used to access remote tables that contain LOB data. However, you can set the DSQEC_LOB_RETRV global variable to 2 or 3 to access LOB metadata or data with a

three-part name. Or, you can use the CONNECT command to connect to the database, and then run the query to access the remote table.

- You can use the PRINT TABLE command to print from a QMF Data Service server. Use a three part name format and ensure that the DSQEC_DS_SUPPORT global variable is set.
- The maximum length of a data row that can be printed depends on the type of object:
  - Printing a table or printing a view that is based on a single table

    Db2 stores records within pages that are 4 KB, 8 KB, 16 KB, or 32 KB in size. Because you cannot create a table with a maximum record size that is greater than the page size, the maximum length of a data row that can be printed is 32 KB when you print a single table. For tables containing LOB or XML columns, each data row contains a locator or pointer that references the location of the data. The data itself is not stored as part of the record and metadata is printed for these types of columns when you issue the PRINT TABLE command.
  - Printing a report or printing a view that is based on two or more tables

    The maximum length of a data row that can be printed from a view that joins two or more tables or from a QMF report in temporary storage is 2 GB when the DSQEC_TWO_GB_ROW global variable is set to 1. When the variable is set to 0, all rows except those that contain LOB or XML columns are truncated at 32 KB. Up to 2 GB of XML, CLOB, and BLOB data and up to 1 GB of DBCLOB data can be printed regardless of the DSQEC_TWO_GB_ROW setting.

  Operations with XML data typically require larger amounts of storage, so printing reports, tables, or views that contain XML data might be limited by the amount of storage you have available. You can use the DSQSPILL and DSQSPTYP parameters to specify the use of extended storage for data no longer needed in active storage.
- The PRINT TABLE command prints XML metadata rather than XML data. By default, the PRINT REPORT command prints XML metadata rather than XML data unless you change the M edit code.
- The behavior of the PRINT REPORT and PRINT TABLE commands for LOB data is controlled by the DSQEC_LOB_RETRV global variable as follows:
  - When the DSQEC_LOB_RETRV global variable is set to 1, LOB metadata is printed by default. You can print LOB data by changing the default M edit code.
  - When the DSQEC_LOB_RETRV global variable is set to 2, LOB metadata is printed and the default M edit code cannot be changed.
  - When the DSQEC_LOB_RETRV global variable is set to 3, LOB data is printed instead of LOB metadata.
- QMF updates the **Last Used** field for the object when you use this command. This field appears on object list panels that are displayed by the LIST command. You can change the list of commands that cause the field to be updated by setting the DSQEC_LAST_RUN global variable.
- If a PRINT TABLE command is directed to a Unicode database and the table contains columns that have graphic data types, QMF casts the data to other types to avoid errors.
- When you issue a PRINT TABLE command that references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. You set the value of this register using the SET CURRENT SCHEMA statement.

A printed report differs from a report that is displayed on a screen in the ways that are shown in the following table:

Table 8. Differences between displayed and printed reports

Part of report	**Displayed report**	**Printed report**
Number of pages	One page that can be scrolled	One or more pages
Page headings and footings	Appear only once	Appear at the top and bottom of each page

Table 8. Differences between displayed and printed reports (continued)		
Part of report	**Displayed report**	**Printed report**
Detail headings	Before the first detail line at the beginning of a report and on every screen that follows	Before the first detail line at the beginning of a report and on every page that follows
Fixed column	Remain in place when report is scrolled horizontally	Repeated on the left side of each page

Examples

1. To display a prompt panel for the QMF PRINT command:

```
PRINT ?
```

2. To print a table that is formatted with a form other than the default form:

```
DISPLAY tablename
DISPLAY formname
PRINT REPORT
```

Related concepts

How QMF recasts certain data types when displaying data
When a DISPLAY TABLE command is directed to a Unicode database and the table referenced in the command contains columns with graphic data types, QMF converts the graphic data types to character data types.

Related reference

SET special register
You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

Global variables that control how commands and procedures are executed
DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

QMF

Use the QMF command to issue a base QMF command, bypassing command synonym recognition. This avoids ambiguity with any site-defined commands that have the same names as base QMF commands.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Issue a base command

Description

qmfcommand
The QMF command to be run.

Usage notes

You can issue the QMF command from the command line, from a procedure, from a database object list panel, or from an application.

Examples

To display the QMF database object list when your site has defined the LIST command to have a different function, enter:

```
QMF LIST
```

REDUCE

The REDUCE command is used in reports and in QBE.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

REFRESH

Use the Refresh function key to issue the REFRESH command. You cannot enter the REFRESH command on the command line.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

The Refresh function key can be used in the following ways:

- On the database object list to re-create the list.
- On the Table Editor CHANGE panel to discard keyed entries before pressing the Change key. The panel is refreshed with the unchanged values for the row still in the database.

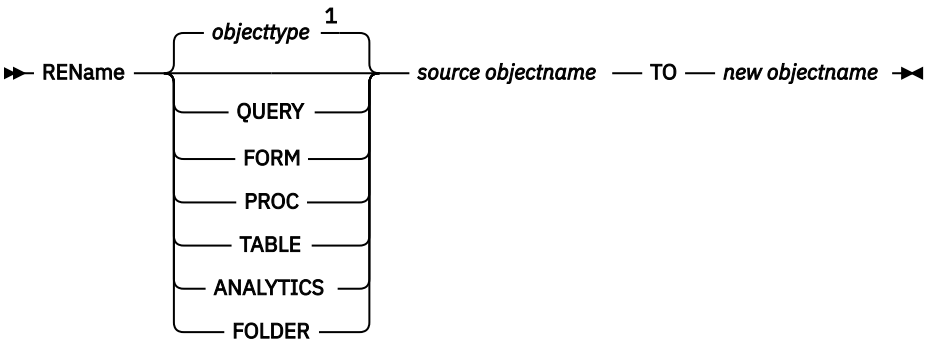
In the Table Editor, a confirmation panel can be displayed before any keyed entries would be lost by the REFRESH command. This confirmation panel is enabled by using the option CONFIRM=YES for the EDIT TABLE command in conjunction with the setting for the global variable DSQCP_TEMOD.

RENAME

The RENAME command renames an object in the database.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

RENAME an object in the database



Notes:

[1] The type of the named object, if appropriate, is used. QMF objects have priority over other types of objects (such as database objects).

Description

objecttype
The type of object to be renamed. Specifying the object type is optional. If you do not specify the object type, QMF determines the type based on the source objectname found in the database. The QMF catalog is searched first, followed by the Db2 database.

source objectname
The name of the object to be renamed. You must specify the object name. Optionally, you can also specify the owner identifier and the location name.

- If you do not specify the owner identifier, the current authorization ID is used.
- If you specify the location name, it must match the current database location.

new objectname
Specifies the new name of the object. The new object name must not already exist in the Db2 database when the source is a table or in the QMF catalog when the source is a QMF object.

Do not specify an owner identifier or location name for the new object name. The source object owner and location are used for the new object name.

The new object name retains all of the physical and metadata attributes of the source object. For example, for a QMF object, the date created, modified, and last used, shared, and object level are retained in the new object name.

Usage notes

- Objects can be renamed only from the current database location. You cannot rename a remote table by using a three-part name. Instead, first connect to the location where the table is located, then issue the RENAME command.
- When you issue a RENAME TABLE command that references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. QMF allows you to set the value of this register by using the SET CURRENT SCHEMA statement.
- When a QMF object is renamed, that object is also renamed in any folder object that references it.

Examples

1. To rename a QMF query from MYAUTHID.MYQUERY1 to MYAUTHID.MYQUERY2:

```
RENAME QUERY MYAUTHID.MYQUERY1 TO MYQUERY2
```

2. To rename a Db2 table from MYAUTHID.MYTABLE1 to MYAUTHID.MYTABLE2:

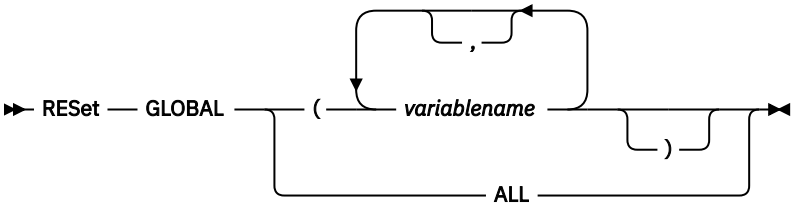
```
RENAME TABLE MYAUTHID.MYTABLE1 TO MYTABLE2
```

RESET GLOBAL

The RESET GLOBAL command deletes the names and values of variables that were created by an administrator or user. These are global variables with names that do not begin with "DSQ."

TSO with ISPF	TSO without ISPF	CICS
X	X	X

RESET global variables



Description

variablename
Names of specific variables to be deleted. You can name up to 10 variables that were created by an administrator or user.

ALL
Deletes the names and values of all variables that were created by an administrator or user. If you have multiple global variables defined, or you do not remember the names of your global variables, you can use this parameter to delete all global variables at one time.

Usage notes

- You can use global variables in queries, procedures, and forms, but not in the Table Editor.
- When you issue RESET GLOBAL ?, a prompt panel is displayed on which you can enter the names of the variables that you want to delete.
- On the Global Variable List panel, you can delete a variable by positioning your cursor on the line you want to delete and pressing the Delete key.

Examples

1. To delete the names and values for all global variables that were previously set by an administrator or user:

```
RESET GLOBAL ALL
```

2. To delete the names and values for only the variables named DEPT and LOCATION:

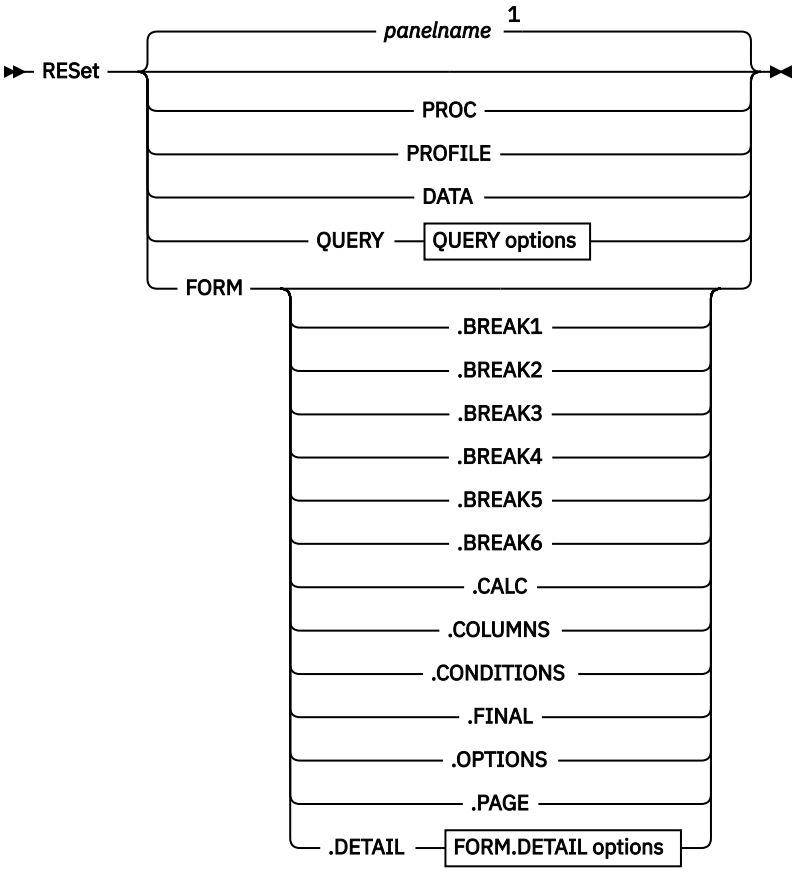
```
RESET GLOBAL (DEPT LOCATION
```

RESET *object*

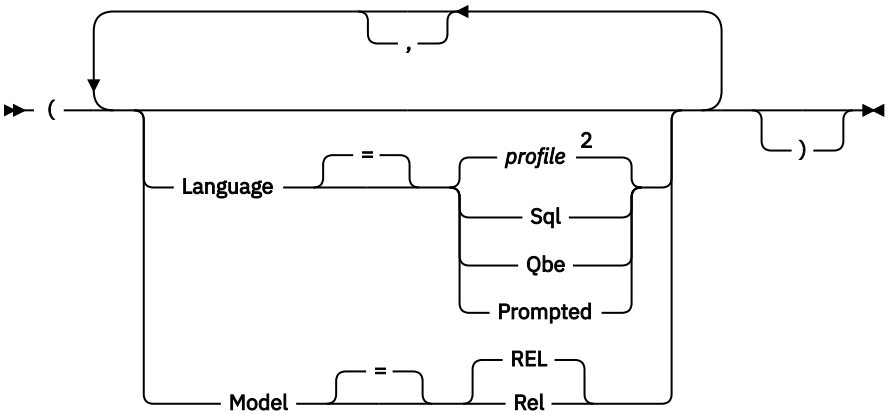
The RESET command restores an object in temporary storage to its initial state. This command does not apply to ANALYTIC objects.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

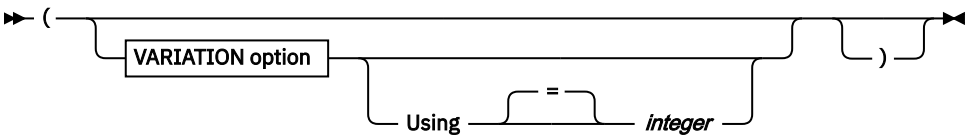
RESET a QMF object in temporary storage



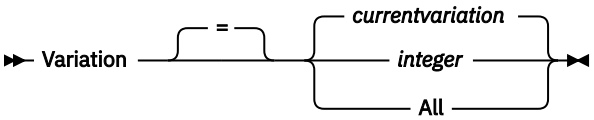
QUERY options



FORM.DETAIL options



VARIATION option



Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The value set in your profile is used.

Description

PROC
Displays an empty procedure panel.

PROFILE
Displays your profile with the values reset to those saved in the database at the current location.

DATA
Purges all data in the DATA temporary storage area and closes the database cursor. The REPORT object in temporary storage is discarded. The QMF home panel is displayed if the RESET command was issued from the REPORT panel.

QUERY
Displays an empty query panel.

QUERY options

LANGUAGE
Specifies which query language to initialize in the query panel.

SQL
Displays a blank SQL Query panel.

QBE
Displays a blank QBE Query panel.

PROMPTED
Displays a blank Prompted Query panel and starts a new Prompted Query dialog.

MODEL
Specifies the data model used for queries. Relational data is the only supported value (REL).

FORM
Displays the FORM.MAIN panel with all parts of the form reset to their default values. The defaults are set to match the column information in the DATA object. If the DATA object is empty, there will be no column information in the form.

If the current panel is FORM.MAIN, the default object for the RESET command is FORM.

FORM.COLUMNS
Displays the FORM.COLUMNS panel with just that part of the form reset to match the column information in the DATA object. If the DATA object is empty, there will be no column information.

FORM.BREAK1
FORM.BREAK2
FORM.BREAK3
FORM.BREAK4
FORM.BREAK5
FORM.BREAK6
FORM.CALC
FORM.CONDITIONS
FORM.FINAL
FORM.OPTIONS
FORM.PAGE
FORM.DETAIL
Displays the specified form panel with just that part of the form reset to its default values.

FORM.DETAIL options

VARIATION
Specifies a detail variation to display with its fields reset.

If this option is omitted, the current detail variation is reset. An exception to this is when more than one detail variation exists and the current panel is not FORM.DETAIL. In this situation, you must specify this option.

integer
The number for a detail variation. The number must be an integer 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.

ALL
Reset 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 variation with similar changes.

integer
The number for an existing detail variation. The number must be an integer from 1 to 99.

Examples

1. To display a prompt panel for the QMF RESET command:

```
RESET ?
```

2. To display an empty SQL Query panel:

```
RESET QUERY (LANGUAGE=SQL
```

3. To erase the data in QMF temporary storage:

```
RESET DATA
```

4. To display the FORM.BREAK6 panel set to the default values for your data:

```
RESET FORM.BREAK6
```

5. To reset only FORM.DETAIL variation 1:

```
RESET FORM.DETAIL (VARIATION=1
```

6. To reset detail variation 2 using detail variation 1 as a template:

```
RESET FORM.DETAIL (VARIATION=2 USING=1
```

7. To reset all detail variations:

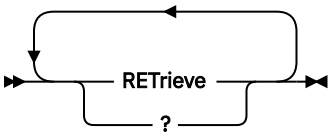
```
RESET FORM.DETAIL (VARIATION=ALL
```

RETRIEVE

The RETRIEVE command displays your most recent command line entry. Using RETRIEVE repeatedly displays command line input in reverse order.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

RETRIEVE a previous command line entry



Description

Use the RETRIEVE command or a ? character on the QMF command line to display the most recently entered command. You can enter multiple ? characters at once to go as far back in the command history as necessary. For example, entering ??? displays the third previous command line entry. The confirmation message that you receive after issuing the RETRIEVE command indicates how far back the retrieved input was entered relative to the input that was most recently entered. When the oldest entry is retrieved, and the RETRIEVE command or ? is entered again, the most recent entry is again displayed.

Usage notes

- When a function key was used to execute a command, only the text that was entered on the command line at that time is redisplayed. The function key must be pressed again to execute the command.
- After the command is retrieved, you can press Enter to reissue the command. If the command is not complete, be sure to modify it before pressing Enter, or press a function key that is compatible with the text on the command line. Characters in retrieved text are converted (or not converted) into uppercase according to the CASE parameter specified in your profile.
- When you type the RETRIEVE command or ? character over existing text on the command line:
  - No space is necessary between the last ? that you type and the existing text. For example, if the command DISPLAY QUERY is already on the command line, ??SPLAY QUERY retrieves the second previous command line entry.
  - RET can be entered, but there must be at least one blank space between RET and the rest of the text. For example, the following is accepted:

    ```
    RET LAY QUERY
    ```

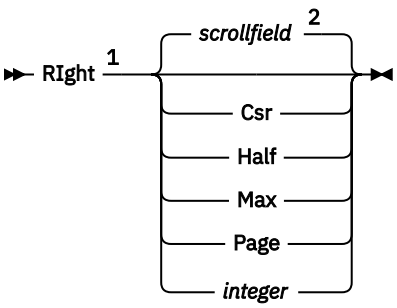
    The following is not accepted:

    ```
    RETPLAY QUERY
    ```

RIGHT

The RIGHT command scrolls toward the right boundary of a QBE query or report panel.

TSO with ISPF	TSO without ISPF	CICS
X	X	X



Notes:

[1] Specify scroll amounts only when there is a SCROLL field on the active panel. PAGE is assumed in all other situations.
[2] The value shown in the SCROLL field is used. This value is also maintained in the global variable DSQDC_SCROLL_AMT.

Description

CSR
Scrolls toward the right, repositioning the column in which the cursor lies to the left edge of the panel. If the cursor is at the right edge of the panel, RIGHT CSR has the same effect as RIGHT PAGE.

HALF
Scrolls toward the right half the width of the panel or to the right boundary (if that is nearer).

MAX
Scrolls to the right boundary of the panel.

PAGE
Scrolls toward the right the width of the panel or to the right boundary (if that is nearer).

integer
Scrolls toward the right this number of columns (a whole number ranging from 1 through 9999).

Usage notes

- MAX is in effect only for the current command. This value will not remain in the SCROLL field after the command completes. You cannot set the global variable DSQDC_SCROLL_AMT to this value.
- Use the RIGHT function key to scroll right in a report. To specify a scroll amount, type the number of columns you want to scroll on the command line and then press the RIGHT function key.

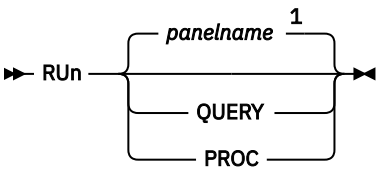
RUN

The RUN command runs queries or procedures from QMF temporary storage or from the database at the current location.

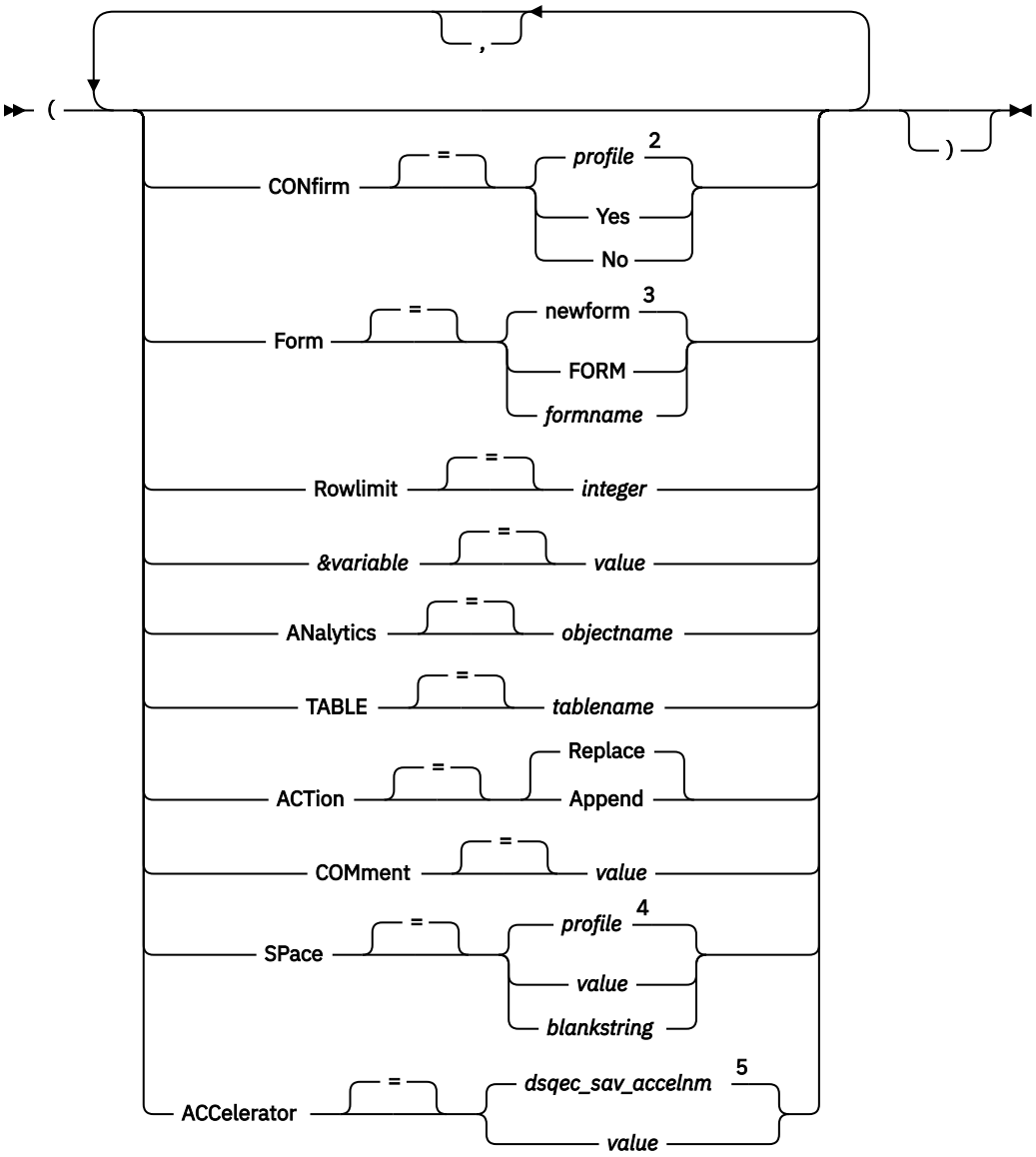
TSO with ISPF	TSO without ISPF	CICS
X	X	*

Syntax

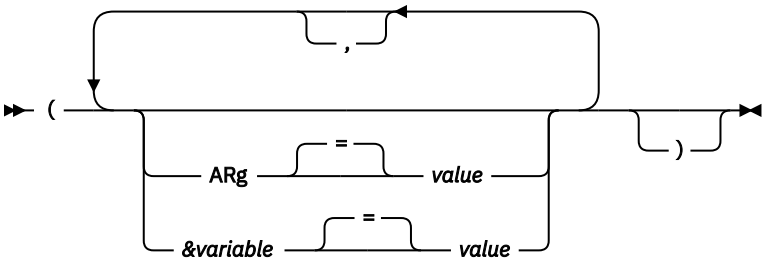
RUN a QMF query or procedure from temporary storage



QUERY options



PROC options



Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.

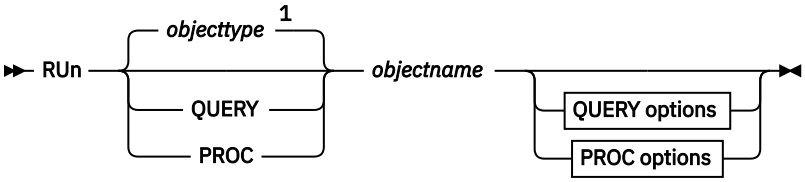
[2] The value set in your profile is used.

[3] "newform" is not an option that can be specified. Rather, when the FORM parameter is not specified on the command, a new form is created by default, with initial values based on the selected data. If you do not specify a form, you can override the default formatting options by setting the following global variables: DSQDC_EC_DATE, DSQDC_EC_TIME, DSQDC_EC_CHAR, DSQDC_EC_NUM, and DSQDC_EC_DEC.

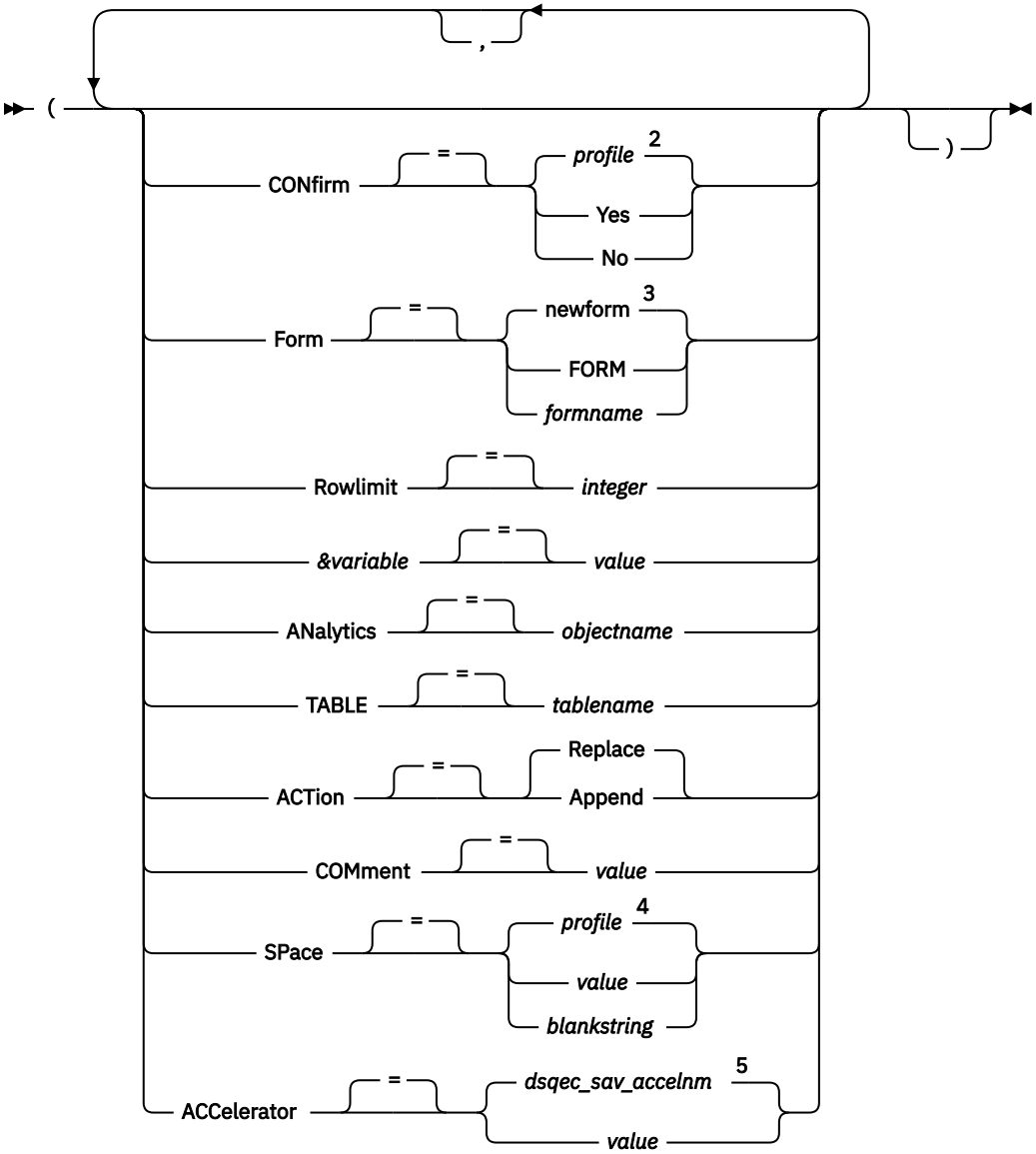
[4] The value set in your profile is used.

[5] The value set in this global variable is used.

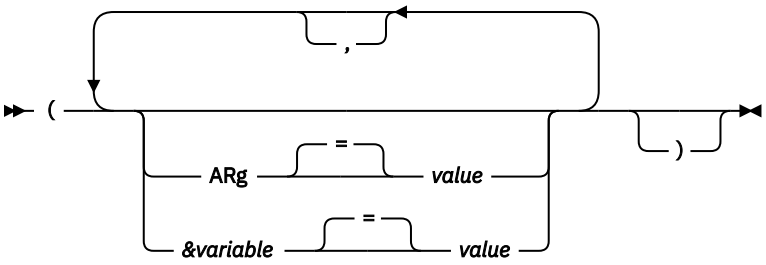
RUN a QMF query or procedure from the database



QUERY options



PROC options



Notes:

[1] The type of the named object, if appropriate, is used. QMF objects have priority over other types of objects (such as database objects).

[2] The value set in your profile is used.

[3] "newform" is not an option that can be specified. Rather, when the FORM parameter is not specified on the command, a new form is created by default, with initial values based on the selected data. If you do not specify a form, you can override the default formatting options by setting the following global variables: DSQDC_EC_DATE, DSQDC_EC_TIME, DSQDC_EC_CHAR, DSQDC_EC_NUM, and DSQDC_EC_DEC.

[4] The value set in your profile is used.

[5] The value set in this global variable is used.

Description

objectname

The name of a QMF object in the database. An object that is owned by another user must be qualified with the owner's name.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.

&variable

Identifies a substitution variable for the RUN command. Variables can be assigned values up to 55 single-byte characters long with this option. Up to 10 substitution variables can be specified in a single command.

The variable name must be prefaced with an ampersand. Use two ampersands if you issue the RUN command from within a linear procedure.

When your query contains substitution variables, QMF first checks whether the variable values are specified on the command itself and, if not, checks for global variables that have the referenced names. QMF prompts you for any variables that do not yet have a value assigned.

If you are running a query that contains multiple SQL statements, the value that is specified for a particular variable name applies to all variables with that name in the entire query. This case applies whether you specify the values at the time you run the query or set global variables before you run the query.

value

The character string that makes up the content of the substitution variable.

A value that contains blank characters must be surrounded with delimiters. Valid delimiters for a substitution variable value are single quotation marks, double quotation marks, and parentheses. When the delimiters are quotation marks, the quotation marks are included as part of the value. When the delimiters are parentheses, the parentheses are not included as part of the value.

Do not enter a query comment as a variable value. Query comments are preceded by two dashes (--), which the database interprets as minus signs.

Query options:

CONFIRM

Indicates whether a confirmation panel is displayed when the query will do either of the following:

- Change an existing object in the database.
- Exceed a cost-estimate limit specified in the resource limit facility (Db2 governor).

There is also a CONFIRM option in the SET PROFILE command.

If the query contains multiple SQL statements, your response to the confirmation panel applies to all statements in the query unless the query contains multiple COMMIT statements.

If the query does not contain multiple COMMIT statements, the answer that you provide in response to the single prompt applies to all changes that are made by all SQL statements in the query. If the query contains multiple statements that change the database and these statements are of different types, the confirmation prompt asks about only one type of statement. For example, if the query contains a DROP statement and an UPDATE statement, the confirmation prompt refers to the UPDATE statement only; however, your response to the prompt applies to both the DROP and UPDATE statements in this case.

If the query contains multiple SQL statements and multiple COMMIT statements, a confirmation panel is displayed for every COMMIT statement. However, if a COMMIT statement follows SQL statements that change only a database catalog, a confirmation panel is not displayed for that COMMIT statement.

FORM
Indicates which QMF FORM to use when formatting the selected data.

FORM
The QMF FORM currently in temporary storage is used. A FORM must be in temporary storage to use this choice.

The report can be displayed if the current FORM is appropriate for the selected data.

formname
The name of a QMF FORM in the database. A form that is owned by another user must be qualified with the owner's name. Additional requirements are:

- The FORM must exist in the database at the current location.
- You must be authorized to use a form that is owned by another user.

The FORM specified becomes the current FORM in temporary storage. The report can be displayed if this FORM is appropriate for the selected data.

ROWLIMIT
Sets a limit for the number of data rows that are returned by a query. Use this option only when you want to restrict how many rows of data are available for the report, from 1 to 99999999 rows.

integer
An integer from 1 - 99999999.

ANALYTICS
When you run a query to display analytics, use this option to indicate which ANALYTIC object name is to be used to format the query results.

objectname
The name of a QMF ANALYTIC object in the database server to which you are currently connected. The name of the current server is shown on the QMF home panel. An ANALYTIC object that is owned by another user must be qualified with the owner's name, and it either must be shared or you must have administrator authority.

TABLE
Specifies that the query results are to be inserted into a table instead of returned to QMF.

tablename
The name of the table in which to insert the data. If the table does not exist, a new table is created in the Q.PROFILES.SPACE table space. You can specify a table with a three-part name only if the table already exists.

If you specify this option, you can also specify the ACTION and COMMENT options.

You cannot specify this option if you also specify the FORM option.

ACTION
Indicates whether to replace the entire database table with the data that is 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.

COMMENT
Stores a comment 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. A comment is a remark or note that you can create when you run the query. The purpose of creating a comment is to provide descriptive information about the data. Users with whom the table is shared can then view this information by pressing the Comments key when the table is displayed in a list.

You cannot replace a comment on a table you do not own or on a remote table that uses a three-part name.

value
The character string that makes up the content of the comment.

A value that contains blank characters must be surrounded with delimiters. Valid delimiters for a comment value are single quotation marks, parentheses, and double quotation marks. If you are using the RUN command from the QMF command line or in a procedure to store a comment with the object, the comment text can be up to 78 single-byte characters. If you are using the **RUN Command Prompt** panel to enter the comment, the comment can be up to 57 single-byte characters.

When the comment itself contains a delimiter character (a single quotation mark, double quotation mark, or parentheses), surround the entire comment with one of the other types of delimiters so that QMF saves the entire comment.

SPACE
Names a storage space to hold any tables that are created by the SAVE DATA command. A blank value specifies that you will use the space that is chosen by the database manager program.

ACCELERATOR
Specifies the name of the accelerator in which the table will be created.

PROC options:

ARG
The argument string to pass to a QMF procedure with logic (REXX procedure). One argument up to 80 characters long can be passed with this option.

The argument string is received by the REXX procedure by using the REXX command PARSE ARG or the REXX function ARG(1).

value
The character string that makes up the content of the argument.

A value that contains blank characters must be surrounded with delimiters. Valid delimiters for an argument value are single quotation marks, parentheses, and double quotation marks. When the delimiters are double quotation marks, the quotation marks are included as part of the value.

Usage notes

- The maximum allowed length of an SQL query that can be run by a RUN QUERY command is determined by the database to which you are connected when you issue the command:
  - In Db2 for z/OS, SQL queries can be up to 2 MB long when the DSQEC_SQLQRYSZ_2M global variable is set to 1. When the variable is set to 0, the maximum size is 32 KB.
  - In DB2 for iSeries and Db2 for Linux, UNIX, and Windows, SQL queries can be up to 65 KB when the DSQEC_SQLQRYSZ_2M global variable is set to 1. When the variable is set to 0, the maximum size is 32 KB.
  - In DB2 for VM and VSE, SQL queries are limited to 8 KB regardless of how the DSQEC_SQLQRYSZ_2M global variable is set.

  QMF supports a query size of 32 KB for prompted and QBE queries unless the database to which you are connected does not support SQL statements of this size.

- You cannot combine a CALL or CREATE PROCEDURE statement with any other SQL statement; each of these statements must be used by itself in an SQL query.
- No more than one SELECT statement can be used in a query that includes other SQL statements.
- If the query contains multiple statements and one of the statements fails, processing stops and no subsequent statements are run. If statements before the failing statement changed the database, these changes are rolled back (not applied to the database) unless the query contains a COMMIT statement. If the query contains one or more COMMIT statements, all database changes that occurred before the SQL error and after the last successful COMMIT statement are rolled back. Some statements, such as SET, apply to the QMF session or environment and therefore are not rolled back in error situations.

  The DSQEC_RUN_MQ global variable controls whether queries with multiple SQL statements are allowed. To run a query with multiple statements, ensure that each statement is separated by a semicolon; then set the DSQEC_RUN_MQ global variable to 1 and run the query. When the variable is set to zero, all statements after the first semicolon are ignored.
- QMF objects can be shared with other users by saving them in the database with the SHARE=YES option of the QMF SAVE command.
- QMF administrative authority does not extend to the RUN command. QMF objects that are saved in the database with the SHARE=NO option cannot be run directly by a QMF administrator. However, QMF administrator can use the DISPLAY command to bring any of these objects into temporary storage and then issue the RUN command.
- Any variables that are used within a QMF query or procedure object must have their values provided before the RUN command executes. A prompt panel displays to gather values for any variables that are not already specified by either:
  - A *&variable* option as part of the command
  - A previously set global variable
- A QMF procedure that contains QMF commands in English can be run in any QMF session when the global variable DSQEC_NLFCMD_LANG is set to 1. However, if it was saved in any other QMF national language, it can be run only in a session of that same national language.
- QMF procedure or query object comments cannot be processed as variables. Do not use two consecutive dashes (--) in variable values. They are treated as part of the command or query to be run, not as comments.
- QMF procedures with logic (REXX procedures) are not supported in a CICS environment.
- If you are running a query that references a three-part table or view name, and your database administrator set up QMF to use the multirow fetch feature, both databases you are working with (local and remote) must be Db2 for z/OS; otherwise, your command fails. Your database administrator can turn off multirow fetch.

  QMF commands that reference three-part names cannot be directed to DB2 for VM or VSE databases, nor can data be accessed remotely if you started QMF as a stored procedure.

  By default, three-part names cannot be used to access remote tables that contain LOB data. However, you can set the DSQEC_LOB_RETRV global variable to 2 or 3 to access LOB metadata or data with a three-part name. Or, you can use the CONNECT command to connect to the database, and then run the query to access the remote table.
- You can use the RUN QUERY command to retrieve data from a QMF Data Service server. Use a three part name format and ensure that the DSQEC_DS_SUPPORT global variable is set.
- QMF formats the data in the resulting report according to options specified in QMF forms. Edit codes control how data of different types is displayed. The M edit code is used for metadata and displays the data type and length of the data instead of the data itself.

  If your hardware does not support decimal floating-point instructions, QMF assigns an edit code of M by default to any columns containing decimal floating-point data. You cannot change this edit code.

  QMF also assigns an edit code of M by default to any columns containing XML, binary (BINARY or VARBINARY), or LOB (BLOB, CLOB, or DBCLOB) data. Depending on the data type, you can change the default edit code from M to another edit code to display the actual data. The ability to change the

edit code for LOB data is controlled by the value of the DSQEC_LOB_RETRV global variable. This global variable can also be set to display LOB data instead of metadata by default.

To display XML or LOB data that is longer than the column width, specify edit codes that allow column wrapping, as follows:

- For XML or CLOB data, set the column width on FORM.MAIN or FORM.COLUMNS to a value of up to 32767 and specify the CW edit code.
- For BLOB data, set the column width on FORM.MAIN or FORM.COLUMNS to a value of up to 32767 and specify the BW or XW edit code.
- For DBCLOB data, set the column width on FORM.MAIN or FORM.COLUMNS to a value of up to 16383 and specify the GW edit code.

Display of database objects that contain XML data might be limited by the amount of storage available to you. If you are using QMF for TSO, your QMF administrator can set the DSQSPILL and DSQSPTYP parameters to specify the use of extended storage for data no longer needed in active storage.

If you are working with XML or LOB data and you receive out-of-storage errors while using an edit code other than M, you can change the edit code to M to clear the error and display the report.

- To run a query or procedure that involves XML data, you must be connected to a database release that supports the XML data type.
- The maximum length of a data row that can be returned from a RUN QUERY command depends on how the DSQEC_TWO_GB_ROW global variable is set:
  - When the global variable is set to 1, the maximum length of a data row in the report is 2 GB.
  - When the global variable is set to 0, row length is limited to 32 KB unless the report contains an XML or LOB column.

    Regardless of the DSQEC_TWO_GB_ROW setting, up to 2 GB of XML, CLOB, and BLOB data, and up to 1 GB of DBCLOB data can be displayed. However, the maximum length of a LOB row can be restricted by the DSQEC_LOB_COLMAX global variable.

    When the table contains LOB or XML columns, the LOB or XML data is not stored as part of the record.

  Regardless of how the DSQEC_TWO_GB_ROW global variable is set, a single table cannot have a maximum record size that is greater than the page size in Db2. Because Db2 stores records within pages that are 4 KB, 8 KB, 16 KB, or 32 KB in size, the maximum length of a data row that can be returned in the report remains at 32 KB when you are selecting columns from a single table. If your SELECT statement references a view that joins two or more tables, the row length of the returned data can be up to 2 GB.
- QMF updates the **Last Used** field for the object when you use this command. This field appears on object list panels that are displayed by the LIST command. You can change the list of commands that cause the field to be updated by setting the DSQEC_LAST_RUN global variable.
- If the RUN QUERY command retrieves data from a table that is stored in a Unicode database and the table contains columns that have graphic data types, QMF casts the data to other types to avoid errors.
- You cannot specify both the SPACE and the ACCELERATOR parameter in the same command.
- If the SPACE or ACCELERATOR parameter is used in the command and the table already exists, SPACE or ACCELERATOR is ignored. The table is re-created at the original location.
- The value of the DSQEC_SAV_ALLOWED global variable determines the default behavior of the SPACE and ACCELERATOR parameters:
  - When the global variable is set to 0, the SAVE DATA command cannot not be used.
  - When the global variable is set to 1, tables are saved only to the database, and only the SPACE parameter is allowed. If the SPACE parameter is not specified, the value is taken from the QMF profile.
  - When the global variable is set to 2, tables are saved only to the accelerator, and only the ACCELERATOR parameter is allowed. If the ACCELERATOR parameter is not specified, the accelerator name that is specified in the DSQEC_SAV_ACCELNM global variable is used.

  - When the global variable is set to 3, tables are saved by default to the database and are saved to the accelerator only when the ACCELERATOR parameter is specified. If neither the SPACE parameter nor the ACCELERATOR parameter is specified, the value of the SPACE setting from the QMF profile is used.
  - When the global variable is set to 4, tables are saved by default to the accelerator and are saved to the database only when the SPACE parameter is specified. If neither the SPACE parameter nor the ACCELERATOR parameter is specified, the accelerator name that is specified in the DSQEC_SAV_ACCELNM global variable is used.
- QMF users can use the QMF Data Service feature to access non-Db2 data such as VSAM, IMS, sequential files, SMF data, SYSLOG data, and more. Through QMF for TSO/CICS SQL queries, you can access QMF Data Service defined data sources using three part names. For example, you might access a VSAM data set defined to a QMF Data Service server named CQDR by issuing the following query: `SELECT * FROM CQDR.CQDSQL.VSAM_IMITMTRN`. Additionally, support for the DISPLAY, DRAW, EXPORT, and PRINT commands in Prompted queries and Query-by-Example (QBE) queries has been added through APAR PI94894.
- If you do not specify a form, you can override the default formatting options by setting the following global variables: DSQDC_EC_DATE, DSQDC_EC_TIME, DSQDC_EC_CHAR, DSQDC_EC_NUM, and DSQDC_EC_DEC.
- Accelerator-only tables are created in the database defined in the DSQEC_SAV_ACCELDB global variable. The Q.PROFILES.SPACE value is not used when defining accelerator-only tables.
- Use the Db2 LOAD Utility cross-loader feature by setting the DSQEC_SAV_LOADER global variable to 1 and using the TABLE keyword on the RUN command. Note that you cannot use the cross-loader and an accelerator table at the same time.

Variable values for the RUN command

QMF assumes that it is at the end of a value for a variable that is specified on the RUN command when it finds a blank, comma, left or right parenthesis, single quotation mark, double quotation mark, or an equal sign. If the value is enclosed in quotation marks, they are included in the value. If the value is enclosed in parentheses, the parentheses are not included in the value. To include parentheses in your final value, you must double them. For example, in processing from the command line, if QMF encounters a single or a double quotation mark, it tries to find a match for it. End strings that start with a quotation mark with a similar quotation mark. If QMF does not find another quotation mark to pair with the first one, it takes the rest of the command specification and includes it with the beginning quotation mark as part of the value.

To include characters like a blank, comma, right or left parentheses, single quotation mark, double quotation mark, or equal sign in your variable, you can enclose the *value* specification in parentheses. For example, in the following RUN command, the value specification for the variable &X ends at the first command and QMF does not accept NAME as a RUN keyword:

```
RUN QUERY (&X=DEPT,NAME,SALARY
```

The same query can be specified on the command line and is properly processed by adding parentheses:

```
RUN QUERY (&X=(DEPT,NAME,SALARY)
```

When the RUN command within a procedure runs a query, the variable parameter can pass a value to a variable within the query. For example, suppose the query uses a variable named &DEPARTMENT. Specifying `&&DEPARTMENT = 66` assigns the value 66 to the variable &DEPARTMENT in the query without making &DEPARTMENT a variable of the procedure. Specifying `&&DEPARTMENT = &DEPT` makes &DEPT a variable of the procedure, and assigns its value to &DEPARTMENT in the query. Values for variables can be set on the SET GLOBAL command before you execute the RUN command. However, a value that is specified on the RUN command overrides the same value that is set with SET GLOBAL.

If you do not set values for your variables before you run your query or procedure, QMF displays a prompt panel so you can enter the values. Be sure the value that is assigned to the variable is no longer than 55 single-byte characters (or the equivalent in double-byte characters).

You can specify values for up to 100 variables in a query or procedure. You can specify up to 10 variables on the RUN command; others must be set by using SET GLOBAL. QMF first looks on the command for a value, then it looks for a global value. If the limit is exceeded, the command is rejected with an error message. Variable names that do not match parameters in your query are ignored.

If your linear procedure sets a variable with SET GLOBAL, that value is not available to commands in that same procedure. However, it would be available to queries and procedures that are called by that procedure.

If you omit the *&variable* parameter, and the object to be run is a query that uses variables and no global variables are set for those variables, a prompt panel is displayed on which you can enter variable values. Variables cannot be replaced by other variables on the RUN command.

System considerations

Any commands that are contained in the procedure that is specified in a RUN PROC command are executed on the system where QMF is running. Thus, if the procedure contains commands not valid for the system where QMF is running (for example, it is a TSO procedure that contains CICS commands or vice versa), those commands fail when you run the procedure.

Examples

1. To display a prompt panel for the QMF RUN command:

   ```
   RUN ?
   ```

2. To run the query currently in QMF temporary storage and format the report with a form from the database (REPORT3) owned by another user (MARIA):

   ```
   RUN QUERY (FORM=MARIA.REPORT3
   ```

3. To run your query from the database (SALESQ) and provide a value for the substitution variable YR:

   ```
   RUN QUERY SALESQ (&YR=1999
   ```

   The same command in a QMF linear procedure is written as:

   ```
   RUN QUERY SALESQ (&&YR=1999
   ```

4. When you issue a RUN QUERY command, it runs a query that is stored at the current location (optionally using a form found at the current location). For example, suppose that the query STATSCHK contains the following statement:

   ```
   SELECT * FROM JOHNSON.STATUS
   ```

   The following command retrieves the query, form, and data from the current location:

   ```
   RUN QUERY STATSCHK (FORM=FORMCHK
   ```

   However, suppose that the query is as follows:

   ```
   SELECT * FROM BILLINGS.JOHNSON.STATUS
   ```

   In this case, the following command retrieves the data from the BILLINGS location and the query and form from the current location:

   ```
   RUN QUERY STATSCHK (FORM=FORMCHK
   ```

Related concepts
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.

How QMF recasts certain data types when displaying data
When a DISPLAY TABLE command is directed to a Unicode database and the table referenced in the command contains columns with graphic data types, QMF converts the graphic data types to character data types.

Related reference

SET PROFILE
The SET PROFILE command changes values in your QMF profile. These values influence the behavior of your QMF session.

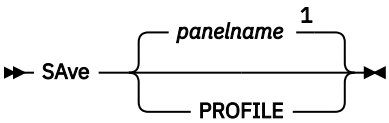
Global variables that control how commands and procedures are executed
DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

SAVE

The SAVE command saves in the database at the current location objects that are currently in QMF temporary storage.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

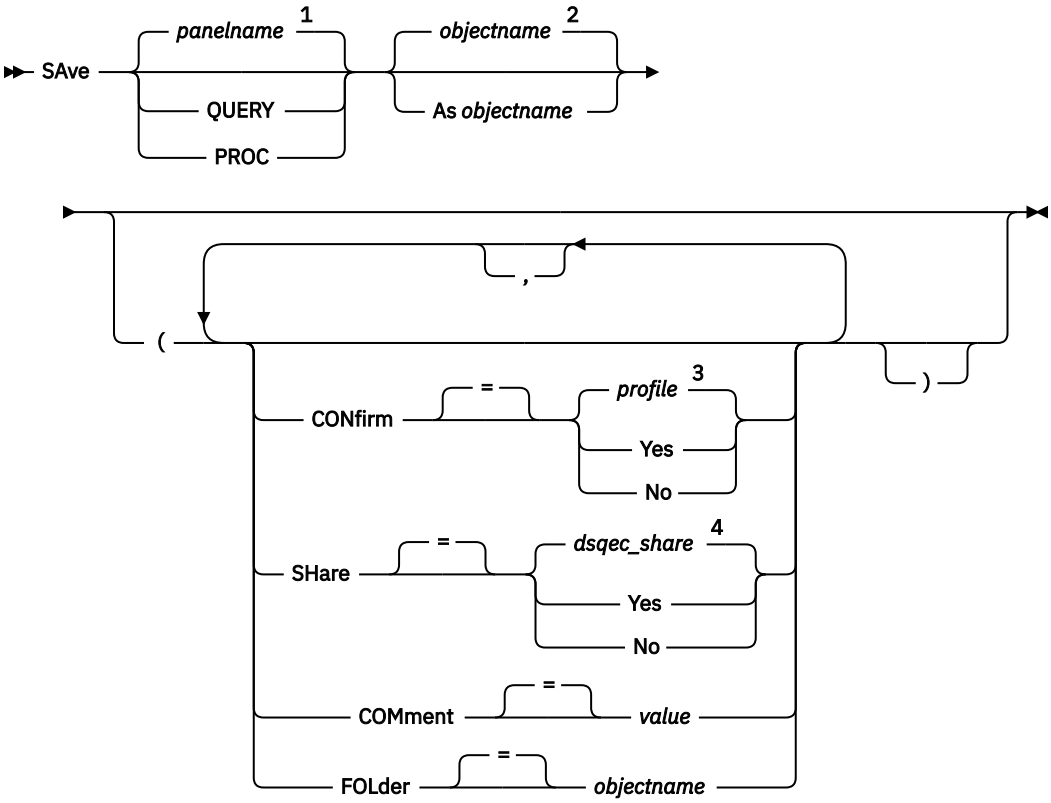
SAVE a QMF profile in the database



Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.

SAVE a QMF query or procedure in the database



Notes:

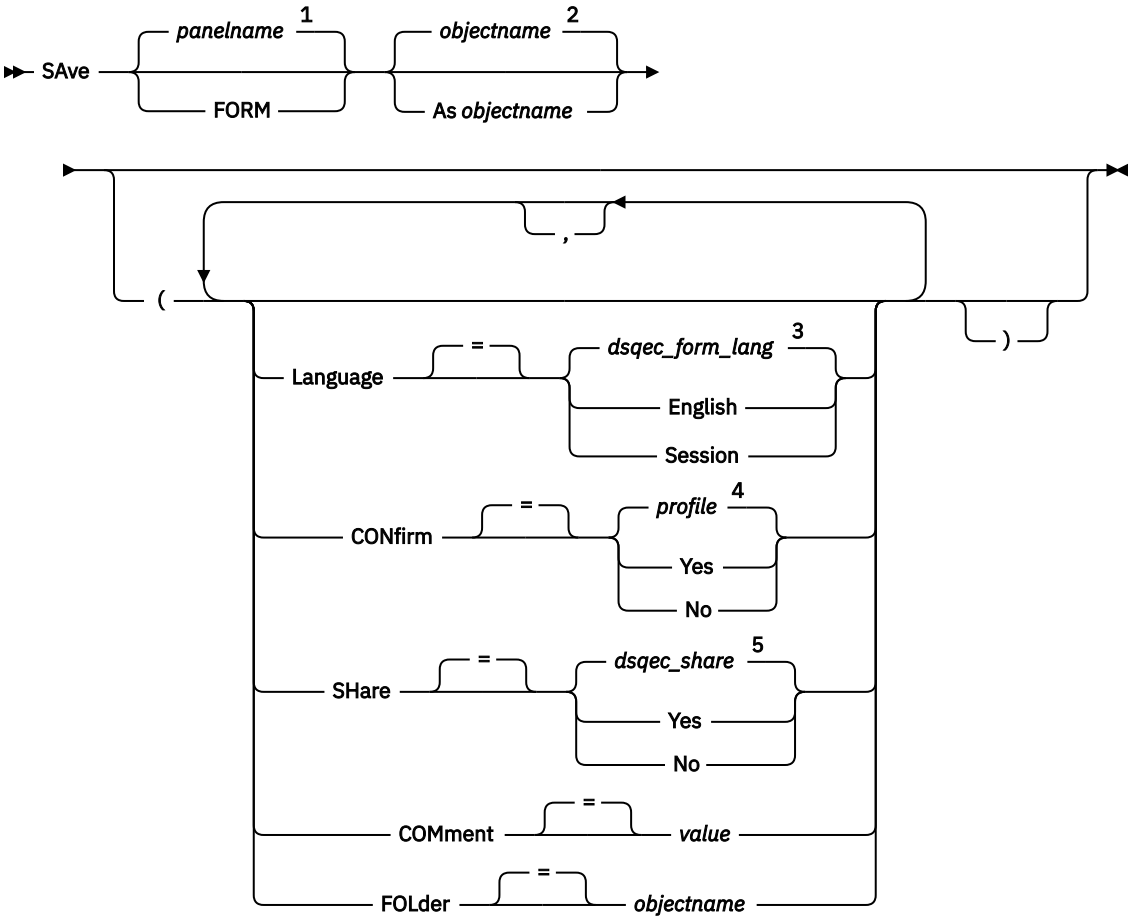
[1] The name of the QMF object panel currently displayed, if appropriate, is used.

[2] The name of the object currently in QMF temporary storage, if any, is used.

[3] The value set in your profile is used.

[4] For an object being replaced, the current value is left unchanged. Otherwise, the value set in this global variable is used.

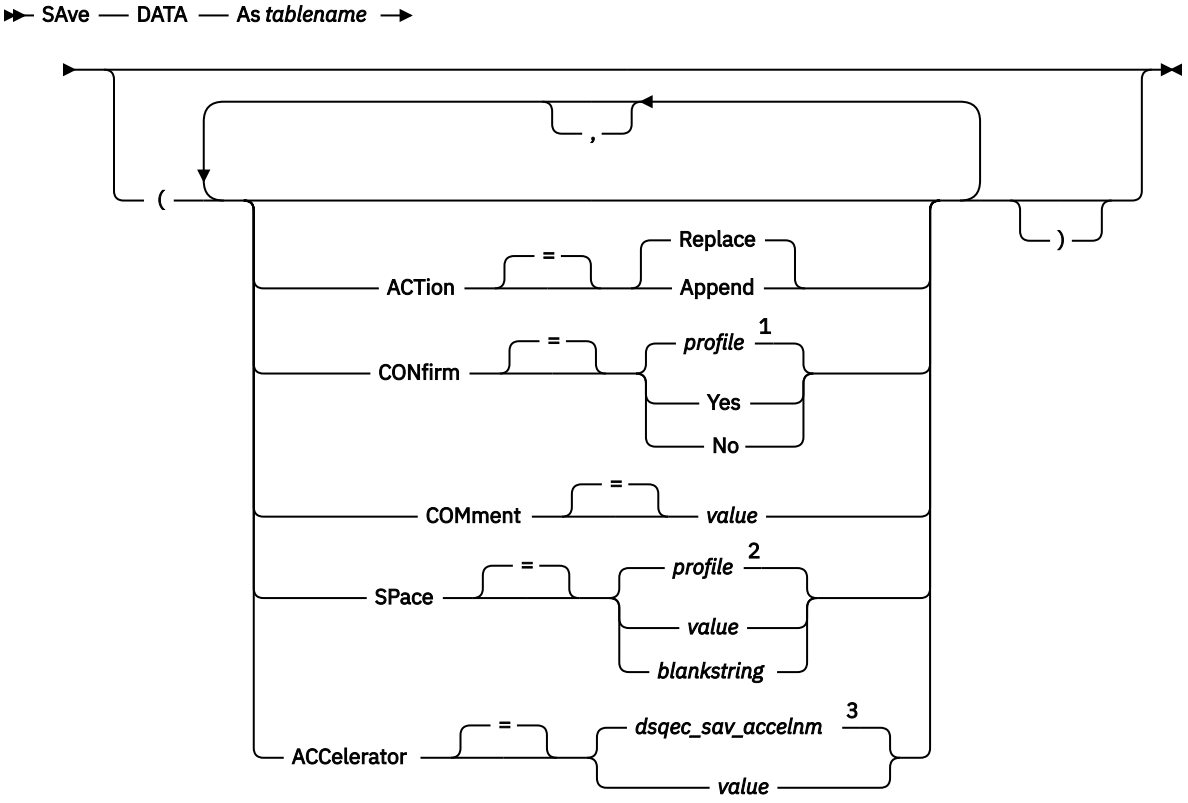
SAVE a QMF form in the database



Notes:

[1] The name of the QMF object panel currently displayed, if appropriate, is used.
[2] The name of the object currently in QMF temporary storage, if any, is used.
[3] The value set in this global variable is used.
[4] The value set in your profile is used.
[5] For an object being replaced, the current value is left unchanged. Otherwise, the value set in this global variable is used.

SAVE QMF data in the database



Notes:

[1] The value set in your profile is used.
[2] The value set in your profile is used.
[3] The value set in this global variable is used.

Description

objectname
The name for the QMF object in the database. The maximum length of the object name is dependent on the database to which you are currently connected.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.To display the panel, issue the following command:

```
SAVE objecttype AS ?
```

where *objecttype* is the type of object you want to save. For example, to display a prompt panel for saving a query, enter:

```
SAVE QUERY AS ?
```

tablename
The name of a table, view, synonym, or alias.

If the object name is too long to fit on the QMF command line, issue the command from a command prompt panel. The name does not need to be delimited by quotation marks when continued on multiple lines on the panel.To display the panel, issue the following command:

```
SAVE TABLE AS ?
```

ACTION
Indicates whether to replace the entire database table with the saved data or to append the saved data to the existing table.

LANGUAGE
Indicates whether QMF keywords contained within the saved form are recorded in English or in the current NLF session language.

A QMF form that contains QMF keywords in English can be used in any QMF session. A QMF form that contains QMF keywords in any other national language that is supported by QMF can be used only in a session of that same QMF national language.

CONFIRM
Indicates whether a confirmation panel is displayed when this command will replace an existing object in the database.

SHARE
Determines whether other QMF users can access the saved object.

SPACE
Names a storage space to hold tables created by the SAVE DATA command. A blank value specifies that the default storage space will be will be determined by the database at the current location.

ACCELERATOR
Specifies the name of the accelerator in which the table will be created.

COMMENT
Stores a comment with the saved object. A comment is a remark or note that you can create when you save the object. The purpose of creating a comment is to provide descriptive information about the object. Users with whom the object is shared can then view this information by pressing the Comments key when the object is displayed in a list.

You cannot replace a comment on a table you do not own or on a remote table that uses a three-part name.

value
The character string that makes up the content of the comment.

A value that contains blank characters must be surrounded with delimiters. Valid delimiters for a comment value are single quotation marks, parentheses, and double quotation marks. If you are using the SAVE command from the QMF command line or in a procedure to store a comment with the object, the comment text can be up to 78 single-byte characters. If you are using the **SAVE Command Prompt** panel to enter the comment, the comment can be up to 57 single-byte characters.

When the comment itself contains a delimiter character (a single quotation mark, double quotation mark, or parentheses), surround the entire comment with one of the other types of delimiters so that QMF saves the entire comment.

FOLDER
The name of the QMF folder object to use with the SAVE command.

You can add a QMF object to a folder by using the FOLDER keyword with the SAVE command. When a folder name is specified with the SAVE command, the QMF object is saved and is also included in the folder.

You can specify a folder name either by including the FOLDER keyword in the SAVE command or by setting the DSQEC_CURR_FOLDER global variable:

- If the FOLDER keyword is specified with the SAVE command, that folder name overrides the folder name that is set in DSQEC_CURR_FOLDER.
- If the FOLDER keyword is not specified with the SAVE command and DSQEC_CURR_FOLDER is set to a folder name, the object is saved and the object is added to the folder name that is specified by DSQEC_CURR_FOLDER.

- If the FOLDER keyword is not specified and DSQEC_CURR_FOLDER is not set, the object is saved but is not added to a folder.

The folder does not need to exist in the database when the SAVE command is run.

The folder name must be a valid QMF object name. The folder name cannot be a QMF object type, such as QUERY, PROC, FORM, ANALYTIC, or FOLDER. Wildcards '%' and '_' are not valid in a folder name. If the folder name includes a blank, the folder name must be enclosed in double quotation marks.

The FOLDER keyword in not valid with the SAVE DATA or SAVE PROFILE commands. The FOLDER keyword is not valid when you are connected to a DB2 Server for VSE and VM database.

Usage notes

- If you save a QBE query that was created in a QMF Version 11.1 or earlier system, the query is saved with long-name characteristics and is longer be usable in QMF Version 11.1 or earlier systems. If you want to avoid converting and replacing an old query, rename the query when you save it.
- You cannot issue the SAVE command to save an ANALYTICS object. To save a QMF Analytics for TSO chart or statistics specification, use the Save function key in QMF Analytics for TSO.
- A QMF administrator can save a QMF object for another user.
- When you save an object and an object exists with the same name, QMF replaces or appends the object (according to the value of the ACTION parameter), subject to these conditions:
  - A query can replace only a query.
  - A procedure can replace only a procedure.
  - A form can replace only a form.
  - Data can replace or append only a similar table object.

    A similar table is one with the same number of columns, with corresponding columns each having 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.

    Column names and labels do not have to match.

    If the data that is to be saved contains XML columns, the data that is to be saved and the existing table must have:

    - The same number of XML columns in the same positions.
    - The same null characteristics defined for the XML columns.
- When you save into an existing table, the column names and labels remain unchanged. If the table does not exist, a new table is created that uses the column names and labels that are recorded within the QMF data object.
- Objects can be saved to a remote location. Use the QMF CONNECT command to make the remote location your current location first, followed by the SAVE command.

  If your current location is a Db2 for z/OS database, you can save to an existing table at a remote location by specifying a three-part name for the table. You cannot save a new table or any QMF objects this way. If your database administrator set up QMF to use the multirow fetch feature, both databases you are working with (local and remote) must be Db2 for z/OS if you are using three-part names; otherwise, your command fails. Your database administrator can turn off multirow fetch.

  QMF commands with three-part names cannot be directed to DB2 for VSE and VM databases, nor can data be saved remotely if you started QMF as a stored procedure.
- To use the SAVE DATA command with columns that contain DECFLOAT data, the processor on which QMF is running must support decimal floating-point instructions.
- Db2 stores records within pages that are 4 KB, 8 KB, 16 KB, or 32 KB in size. Because you cannot create a table with a maximum record size that is greater than the page size, the maximum length of a data row

that can be saved with the SAVE DATA command is limited to 32 KB even when the QMF report displays rows longer than this limit. When you save data that contains an XML column, each data row contains a pointer that references the location of the data; the data itself is not stored as part of the record and therefore does not count toward the 32 KB limit.

- Operations with XML or LOB data typically require larger amounts of storage. Therefore, saving data or tables that contain XML or LOB data might be limited by the amount of storage that you have available.
  - To save an object with XML data, you must be connected to a database release that supports the XML data type.
  - The ability to save a table that contains LOB data is controlled by the DSQEC_LOB_SAVE global variable. If saving of LOB data is enabled and the DSQEC_LOB_RETRV global variable is set to 1 or 3, the SAVE DATA command saves all LOB data in the table. If the DSQEC_LOB_RETRV global variable is set to 2, LOB data cannot be saved regardless of the DSQEC_LOB_SAVE global variable setting because LOB data is not retrieved.
- When you save into an existing table, the column names and labels remain unchanged. If you replace or append data in an existing temporal table, the table remains temporal. However, you cannot save data into a new temporal table. If the specified table does not exist, a new table is created using the column names and labels in the QMF data object. When you save data into a new or existing table, new values are created for columns that were defined with the GENERATED ALWAYS attribute.
- QMF updates the **Last Used** field for the object when you use this command. This field appears on object list panels that are displayed by the LIST command. You can change the list of commands that cause the field to be updated by setting the DSQEC_LAST_RUN global variable.
- When you issue a SAVE DATA command that references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. You can set the value of this register by using the SET CURRENT SCHEMA statement.
- When you issue the SAVE DATA command with the ACTION=REPLACE parameter and the data to be saved contains column label information, QMF creates labels on the new table if the database supports the LABEL ON statement. If the database does not support the LABEL ON statement, the new table is created without column labels.
- The maximum length of a query that can be run by the RUN QUERY command depends on the type of database to which the command is directed.
- You cannot specify both the SPACE and the ACCELERATOR parameter in the same command.
- If the SPACE or ACCELERATOR parameter is used in the command and the table already exists, SPACE or ACCELERATOR is ignored. The table is re-created at the original location.
- The value of the DSQEC_SAV_ALLOWED global variable determines the default behavior of the SPACE and ACCELERATOR parameters:
  - When the global variable is set to 0, the SAVE DATA command cannot not be used.
  - When the global variable is set to 1, tables are saved only to the database, and only the SPACE parameter is allowed. If the SPACE parameter is not specified, the value is taken from the QMF profile.
  - When the global variable is set to 2, tables are saved only to the accelerator, and only the ACCELERATOR parameter is allowed. If the ACCELERATOR parameter is not specified, the accelerator name that is specified in the DSQEC_SAV_ACCELNM global variable is used.
  - When the global variable is set to 3, tables are saved by default to the database and are saved to the accelerator only when the ACCELERATOR parameter is specified. If neither the SPACE parameter nor the ACCELERATOR parameter is specified, the value of the SPACE setting from the QMF profile is used.
  - When the global variable is set to 4, tables are saved by default to the accelerator and are saved to the database only when the SPACE parameter is specified. If neither the SPACE parameter nor the ACCELERATOR parameter is specified, the accelerator name that is specified in the DSQEC_SAV_ACCELNM global variable is used.

- Accelerator-only tables are created in the database defined in the DSQEC_SAV_ACCELDB global variable. The Q.PROFILES.SPACE value is not used when defining accelerator-only tables.

Examples

1. To display a prompt panel for saving a form:

```
SAVE FORM ?
```

2. To include a comment with a saved query:

```
SAVE QUERY AS STAFFQ2 (COMMENT=(Staff report for departments))
```

3. To save a query in QMF temporary storage into the database at the current location:

```
SAVE QUERY AS HAZEL.QUERY3
```

4. To save a QMF object to a remote database server (MADRID), first connect to that location:

```
CONNECT TO MADRID
```

   Then save the object:

```
SAVE FORM AS FORMAT2
```

   You cannot connect to a remote database if you started QMF as a stored procedure.

5. If your current location is Db2 for z/OS and you want to save your data to an existing table (HAZEL.STATUS) at a remote database location (BILLINGS):

```
SAVE DATA AS BILLINGS.HAZEL.STATUS
```

   QMF commands with three-part names cannot be directed to DB2 for VSE and VM databases, nor can data be accessed remotely if you started QMF as a stored procedure.

6. QMF administrator (QMFADM) saving a procedure for another user (HAZEL):

```
SAVE PROC AS HAZEL.MONTHLY (COMMENT=(MONTHLY PROCESS)
```

7. To save a QMF query object called YR2014 and include that object in a FOLDER named SALES:

```
AVE QUERY AS YR2014 (FOLDER = SALES
```

Related reference

RUN
The RUN command runs queries or procedures from QMF temporary storage or from the database at the current location.

SET special register
You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

Global variables that control how commands and procedures are executed
DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

Related information

Search for information about support for implicit casting with your database.

SEARCH

In the Table Editor, the SEARCH command locates specified information in a database table.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

SEARCH for information using the Table Editor

Usage notes

- When searching for data with a specific ending, be aware of the data type of the column you are searching. If the column has a fixed width and the data in the column varies in width, use a trailing percent sign to represent any blanks that might follow your search criteria.
- When you are in SEARCH mode, enter your search criteria and press the SEARCH function key to retrieve rows whose columns match your search criteria.
- To search for data when you know only part of a value, use either or both of the following symbols in your search criteria as wildcards for locating patterns:

  % (percent)
  Use as a placeholder for any number and combination of characters, including no characters at all.

  _ (underscore)
  Use as a placeholder for exactly one character.

  You can use both % and _ in the same value. Each can be used multiple times. For example, using a pattern of _OS% as your search criteria might find a match with the column values of ROSS, DOS or BOSLEY.

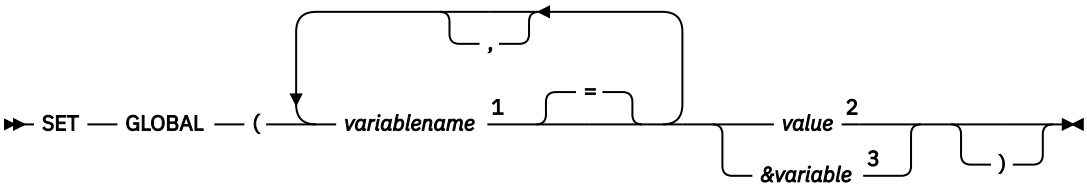
SET GLOBAL

The SET GLOBAL command assigns values to global variables from the QMF command line, from a procedure, or through the callable interface. You cannot change the value of a global variable that is defined as read-only.

You can define up to 10 substitution variables from the QMF command line or in a procedure. In the callable interface, the number of variables is limited only by your environment, and the exact syntax of the command depends on the language used. Use this linear syntax of the command with QMF procedures and REXX applications. Use the extended syntax to change the values of variables in callable interface languages other than REXX.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Set a global variable in a QMF procedure or REXX application



Notes:

[1] Identifies the global variable to which a value is assigned.
[2] The character string that makes up the content of the global variable.
[3] A global variable name that contains the content of the global variable.

Description

variablename
Identifies the global variable to which a value is assigned.

value
The character string that makes up the content of the global variable.

A value that contains blank characters must be surrounded with delimiters. Valid delimiters for a global variable value are single quotation marks, parentheses, and double quotation marks. When the delimiters are double quotation marks, the quotation marks are included as part of the global variable.

When a SET GLOBAL command is entered from a linear procedure and the variable value spans multiple lines, the value must be enclosed in quotation marks and a continuation character (+) must be used in the first position of each line. Parentheses cannot be used as a delimiter when spanning multiple lines.

Usage notes

- Global variables can be used in queries, procedures, and forms. Preface a variable with one or more ampersands (&) when you use it in a QMF object.
- A global variable name can contain a numeric character, but the first character of a global variable name cannot be numeric.
- The first character of a global variable name must be an alphabetic character (A through Z) or one of these special characters:

```
¢  !  $   ~   {  }  ? @  #  %  \
```

  On the SET GLOBAL command, variable names are not preceded with an ampersand like they are on the RUN and CONVERT commands.

  Global variable names cannot begin with DSQ, because QMF reserves these letters for QMF predefined global variables.
- A global variable name cannot contain blanks or any of the following characters:

```
.  ,  ;  :  <  >  (  )  |  +  -  *  / =  &   ¬  '  "
```

- Variable names are limited to 18 single-byte characters (or the equivalent in double-byte characters) unless the variable is to be used as a substitution variable. Substitution variable names are limited to 17 characters. Character constants do not need to be enclosed in single quotation marks.
- You can assign a variable value of 55 or fewer bytes with the SET GLOBAL command. To define variable values over 55 bytes, use the SHOW GLOBALS command to display the GLOBALS panel.
- Global variable names with question marks are not recognized in QMF forms.
- Global variables set to form variable names or aggregation variable names are not recognized by the QMF form.
- Trailing blanks are not recognized in global variable names.
- If a variable is a character string that is a name (such as the name of a column, a table, or an operator):
  - Double all embedded quotation marks.
  - Enclose the complete string in a set of single quotation marks. (These quotation marks are not considered part of the value.)

  For example, suppose that the SELECT statement is the following:

```
SELECT DEPT, &COL FROM &TABLE
```

  The SET GLOBAL command that sets the variables for this SELECT statement might be something like the following:

```
SET GLOBAL (COL='NAME', TABLE='Q.STAFF'
```

- If the variable value that you are setting is a character string that contains quotation marks, you can use quotation marks or parentheses to delimit the value. For example, consider the following query:

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

To set the STAFF_NAME variable to a value of 'JAMES' using quotation marks, issue the following command:

```
SET GLOBAL (STAFF_NAME = '''JAMES''')
```

To set the STAFF_NAME variable using parentheses, issue the following command:

```
SET GLOBAL  (STAFF_NAME=('JAMES'))
```

- If the variable contains a blank, comma, single quotation mark, double quotation mark, or an equal sign, the entire value must be enclosed in a set of parentheses. However, if the value includes an unmatched set of left or right parentheses or begins or ends with a left or right parenthesis, respectively, you must use quotation marks instead.

  For example, consider the following SELECT statement:

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

  To specify more than one column name in this SELECT statement, you need to include commas and, optionally, blanks to separate the values. Therefore, the SET GLOBAL command must be surrounded by parentheses, as shown in the following example:

  ```
  SET GLOBAL (COLS=(NAME, JOB, SALARY)
  ```

- At least one variable must be specified.
- If a quotation mark is required within a variable value, use two single quotation marks.
- Do not use a query comment as a variable value. A query comment is preceded by two dashes (--), which the database interprets as minus signs.
- When you are setting many variables, it is easier to keep track of them if you use a procedure.
- If the variable is a numeric string, you do not need to use quotation marks.
- If the variable name is not found in the QMF product global variable pool, a new variable is created. If the variable name is found, the new value replaces the old value.

Examples

1. To display a prompt panel where you can enter the variables and values that you want to set, issue the following command:

   ```
   SET GLOBAL ?
   ```

2. To assign a value of 38 to the variable DEPT and a value of 'SALES' to the variable JOB, enter the following command:

   ```
   SET GLOBAL (DEPT = 38, JOB = '''SALES'''
   ```

3. One way to assign the value of 'O''BRIEN' to the variable NAME is to use the following command:

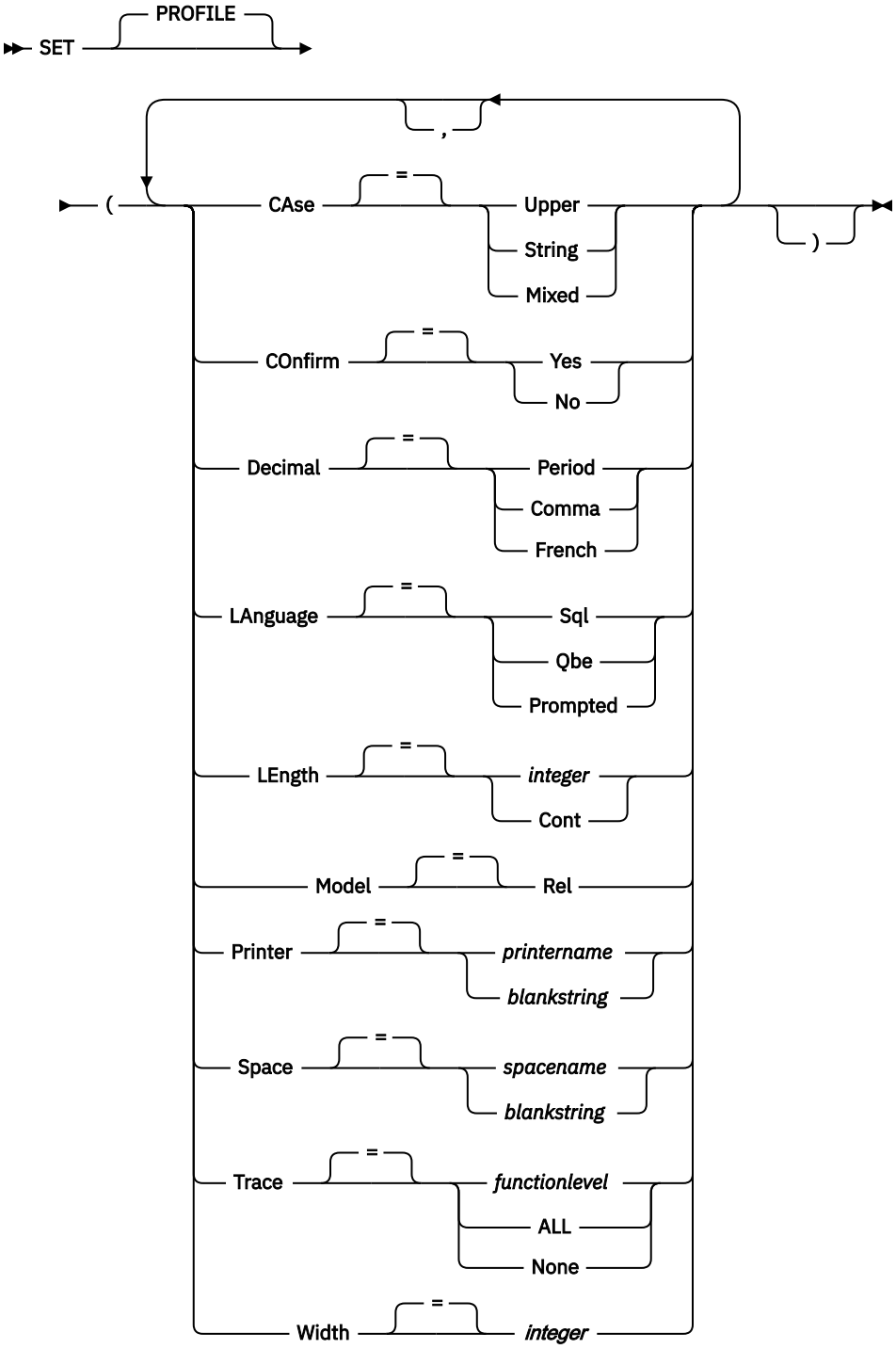
   ```
   SET GLOBAL (NAME ='''O''''BRIEN'''
   ```

SET PROFILE

The SET PROFILE command changes values in your QMF profile. These values influence the behavior of your QMF session.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Change the QMF profile in temporary storage



Description

CASE
Specifies whether commands and other input are converted to uppercase.

UPPER
Converts all input to uppercase.

STRING
Converts input to uppercase, except for the following:

- Characters enclosed in single or double quotation marks
- Comments in SQL or QBE queries and procedures
- Column headings, page headings and footings, break headings, or detail headings
- Data entered in the Table Editor
- All text in procedures with logic (which use REXX)

MIXED
Does not convert input to uppercase. Input is used just as it is typed. When this value is used, all operators in QBE queries, all reserved words, and all QMF commands must be entered in uppercase. Column names in QBE queries must be entered in uppercase unless they are written using lowercase in the database.

Use this option if you use the QMF CONNECT command in TSO and your site uses mixed-case passwords for RACF. Otherwise, QMF converts the password to uppercase, causing the CONNECT command to fail.

CONFIRM
Specifies the default action for confirmation prompting with QMF commands that support the CONFIRM option. This default applies when the commands do not specify the CONFIRM option.

Confirmation prompting provides an opportunity to cancel an irrevocable command action before it takes place. Irrevocable command actions include changing, replacing or purging an object, such as a data set or something in the database.

YES
Enables the display of confirmation panels, which provide an opportunity to cancel the command before it runs.

When you run a query that contains multiple SQL statements that change the database, a single confirmation panel is displayed. The answer you provide in response to this prompt applies to changes that will be made by all SQL statements in the query.

NO
Disables the display of confirmation panels.

DECIMAL
Specifies how to punctuate decimal numbers in a report. This option controls the formatting characteristics of the decimal point and the thousands separators for numeric values formatted with the decimal edit codes.

PERIOD
Uses a period (.) for the decimal point and a comma (,) for the thousands separators.

COMMA
Uses a comma (,) for the decimal point and a period (.) for the thousands separators.

FRENCH
Uses a comma (,) for the decimal point and a space for the thousands separators.

The following examples show the results of using the DECIMAL option when formatting the value 7654321 with two decimal places:

PERIOD
7,654,321.00

COMMA
7.654.321,00

FRENCH
7 654 321,00

LANGUAGE
Specifies the default query language for the query panel.

SQL
Structured Query Language

QBE
Query-by-Example

PROMPTED
Prompted Query

LENGTH
Specifies the default length of a printed page. The unit of length is one line.

integer
Specifies the maximum number of lines between page breaks. The number must be an integer from 1 to 999.

CONT
Specifies continuous printing, without page breaks.

MODEL
MODEL is left for compatibility with previous releases. MODEL may not be updated by the SET PROFILE command.

PRINTER
Specifies the default output destination for the QMF PRINT command and the Print function key in QMF Analytics for TSO.

printername
Specifies a printer destination. This must be the nickname of a GDDM printer.

blankstring
Specifies a file destination. This value must be indicated by a string of 0 to 8 blanks enclosed in single quotation marks (' ').

The physical destination for the print output is determined by your QMF environment and tailoring by your administrator:

- In TSO, the output goes to the data set or device allocated to the QMF file DSQPRINT.
- In CICS, the output goes to a CICS queue specified by the QUEUENAME option of the PRINT command or its default.

Use a string of blanks for the printer option when you have started QMF for TSO as a stored procedure and you want to receive output back in a result set.

SPACE
Specifies the default storage space in the database, where tables created with the SAVE DATA or IMPORT TABLE command will be placed.

Note that when creating accelerator-only tables, the SPACE profile is not used to define the default database definition. Accelerator-only tables are created in the database defined in the DSQEC_SAV_ACCELDB global variable. The Q.PROFILES.SPACE value is not used when defining accelerator-only tables. Accelerator-only tables are created through the SAVE DATA, IMPORT TABLE and RUN QUERY with the TABLE keyword.

spacename
The name of a valid storage structure for the current database location. This could be a dbspace name, a database name, a table space name, or a combination of a database and a table space name.

To specify implicit creation of table spaces, specify the DATABASE keyword followed by a database name in double quotation marks. For example, when you specify the following value for the SPACE option, the database manager implicitly creates a table space exclusively for each table within the DATABASEA database:

```
DATABASE "DATABASEA"
```

blankstring
Specifies the storage structure default, which depends on the database to which you are currently connected. This value must be indicated by a string of 0 to 50 blanks enclosed in single quotation marks.

TRACE
Turns the QMF trace facility on or off.

functionlevel
Enables trace activity for individual functions and allows you to specify the level of trace detail you want for each function you specify.

Specify *functionlevel* as a list of alternating letters (codes that indicate the functions you want to trace) and numbers (level of trace detail you want for each function specified). Codes and levels are shown in following table:

Table 9. Codes for tracing specific QMF functions

Code	**Function traced**
A	Applications
C	Common services
D	Driver modules
E	Front-end processor
F	Formatter
G	Graphic translator
I	Database interface
L	Messages and commands
P	Graphics plotter
R	Radix partition tree
U	User exits

Specify the level of trace detail you want for each function by entering one of the following numbers after the code shown in the previous table.

- 0 = No tracing
- 1 = Tracing occurs at entry and exit points as well as for input and output parameters
- 2 = Traces internal data as well as Level-1 data.

For example, a trace code of A2 traces applications at the highest level of detail.

Use one of the following trace codes to trace messages, commands, or both:

- Messages only (L1)
- Messages and QMF commands (L2)

The L trace code can help you find errors in batch-mode procedures.

ALL
Enables trace activity for all functions and all levels.

NONE
Disables trace activity.

When you start QMF for TSO as a Db2 for z/OS stored procedure, you set the level of trace detail by passing a parameter value on the CALL statement that starts QMF. When QMF has been started in this manner and the trace output has been set to go to any destination other than the default trace data set (DSQDEBUG), trace settings cannot be changed.

WIDTH
Specifies the default width of a printed page. The unit of width is one single-byte character.

integer
Specifies the maximum number of characters to be printed on any line. The number must be an integer from 22 to 999.

Lines wider than the value specified are cut off on the right, unless the object you are printing is a report. In that case, lines longer than the value specified are formatted on a subsequent page, unless you specified line wrapping on the FORM.OPTIONS panel.

Usage notes

- The changes in effect as a result of the SET PROFILE command remain in effect for the current QMF session. To save these changes in your profile so that they persist from one QMF session to another, use the SAVE PROFILE command after you enter SET PROFILE.
- To change values in the QMF profile without using the SET PROFILE command, enter SHOW PROFILE and change any options on the profile panel.

SHOW

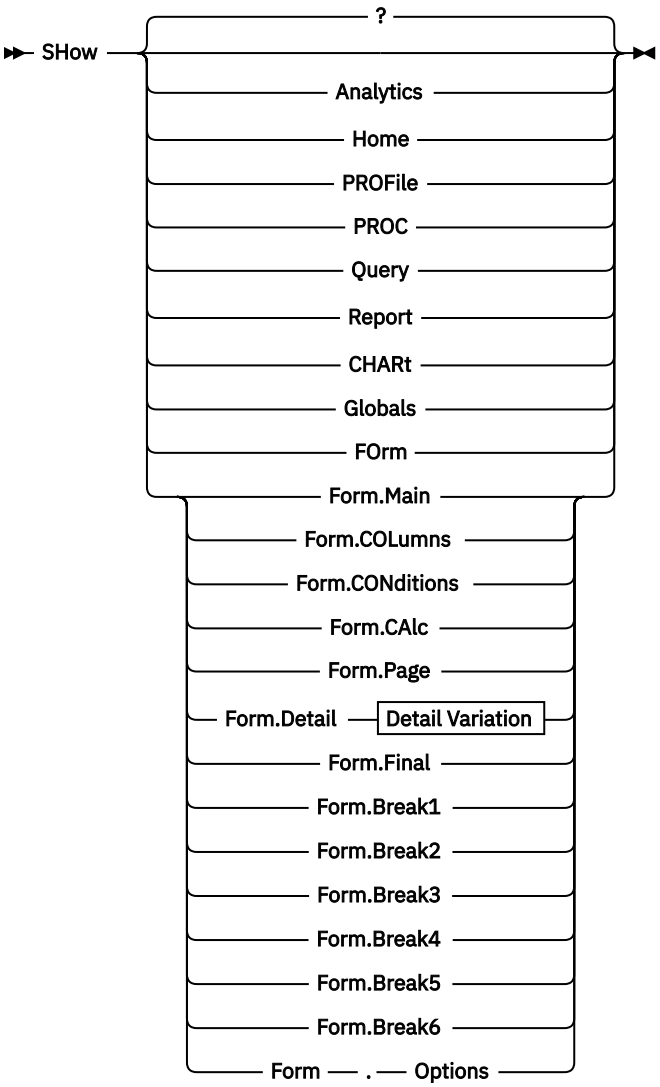
The SHOW command has many uses. For example, you can use the SHOW command to navigate among object panels and show a variation of the FORM.DETAIL panel.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

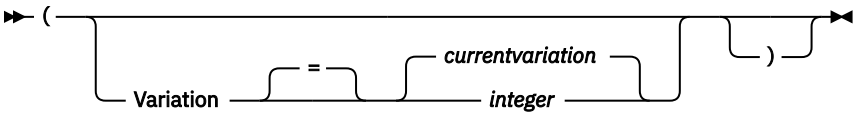
Specifically, the SHOW command is used to:

- Show the QMF Analytics for TSO Home panel.
- Navigate among object panels
- Show a list of global variables
- Show fields that are too long to fit on the panel
- Show the SQL equivalent of a prompted query
- Show a command panel from the database object list that lets you specify any QMF command or synonym
- Show a variation of a FORM.DETAIL panel
- Show the object name and authorization ID of the current object
- Show the service information for a module

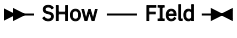
SHOW an object panel



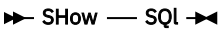
Detail Variation options



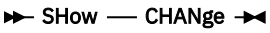
SHOW more information for fields on certain panels



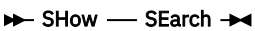
SHOW the SQL equivalent for a prompted query



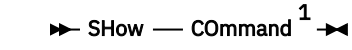
SHOW the Table Editor Change panel



SHOW the Table Editor Search panel



SHOW a command entry panel



Notes:

[1] Valid only from a database object list panel with an Action column.

SHOW the name or authorization ID of the current object

SHOW service information

Description

ANALYTICS
HOME
PROFILE
PROC
QUERY
REPORT
CHART
GLOBALS
FORM.MAIN
FORM.COLUMNS
FORM.CONDITIONS
FORM.CALC
FORM.PAGE
FORM.DETAIL
FORM.FINAL
FORM.BREAK1
FORM.BREAK2
FORM.BREAK3
FORM.BREAK4
FORM.BREAK5
FORM.BREAK6
FORM.OPTIONS
NAME
The specified object panel is shown as the current panel.

FORM
The current form object panel is shown as the current panel. This could be any one of the various form parts that was previously shown or displayed.

FIELD
Show additional information for a field on a base panel. This command option is used only with function keys from panels in the following situations:

- To show the characteristics of a column or to enlarge the input area for a long character field when using the Table Editor
- To enlarge the input area when providing comparison values in Prompted Query
- To enlarge the input area when changing or viewing a global variable value on the global variable list panel

SQL
Show the SQL statement equivalent of a prompted query. The SQL statement can be viewed but not modified.

CHANGE
SEARCH
Show the specified Table Editor panel during a Change mode edit session. This is used alternately to toggle between the two panels.

This command option is available only through function keys provided with the Table Editor.

COMMAND
Show a QMF command entry panel when using the database object list panel. A QMF command or command synonym can be independently executed without first leaving the object list.

This command option is available only through a function key provided with the database object list.

NAME
Show the complete name of the object that is currently being displayed. SHOW NAME provides a view of the complete object name in a pop-up panel when the object name has been truncated. In some cases, the report object may not be an object name associated with the report. In these cases, the SHOW NAME command shows a blank authorization ID and object name.

SERVICE.*modulename*
Show the service information for the specified module. The information is returned in a message.

Detail variation

VARIATION
Specifies a detail variation to show.

If this option is omitted, the current detail variation is shown.

This option does not appear in the SHOW command prompt panel because the number is typed directly on the FORM.DETAIL panel.

integer
The number for a detail variation. The number must be an integer 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.

Usage notes

- The SHOW command is similar to the DISPLAY command. Here are the differences:
  - The SHOW command shows object panels, global variables, and certain parts of panels in QMF temporary storage.
  - The DISPLAY command displays objects from the database or objects currently in QMF temporary storage.
- SHOW ANALYTICS is available in QMF for TSO only.
- SHOW CHART applies to QMF form-based charts. It does not apply to QMF Analytics for TSO charts.
- The SHOW GLOBALS command displays the GLOBALS panel. On the GLOBALS panel, you can set or change any variable that has an entry field in the Value column enclosed by brackets or parentheses. Otherwise, the variable is read-only. Change existing values by typing over the value that is shown or by pressing the Show Field key to display the Show Global Variable screen. You can also press the Add key to define a new variable on the Add Global Variable screen. The maximum length of a variable value that is defined on the Show Global Variable screen or Add Global Variable screen is 32,768 bytes.
- By default, values for global variables persist for the duration of the QMF session or until you reset them. However, the DSQEC_USERGLV_SAV global variable can be set to save global variable values from one session to another.
- SHOW REPORT and SHOW CHART can fail if the form is incompatible with the data, or if the form contains errors. QMF displays the form panel on which the first error occurs, highlighting the entry area containing the error. To see any remaining errors, correct the first error displayed and press Enter.

Examples

1. To display a prompt panel for the QMF SHOW command, enter one of the following commands:

```
SHOW
SHOW ?
```

2. To show the QMF Analytics for TSO Home panel:

```
SHOW ANALYTICS
```

3. To show the name of the current QMF object:

```
SHOW NAME
```

4. To directly navigate to the QMF home panel:

```
SHOW HOME
```

5. To show variation 2 of FORM.DETAIL:

```
SHOW FORM.DETAIL (VARIATION=2
```

6. To create a new variation of FORM.DETAIL:

```
SHOW FORM.DETAIL (VARIATION=99
```

SORT

The SORT command sorts items in a database object list. You can issue this command only by pressing the Sort function key. When you request sorting, a panel is displayed that lets you select the order of the object names.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

You can set the DSQDC_LIST_ORDER global variable to change the default sort order.

Related reference

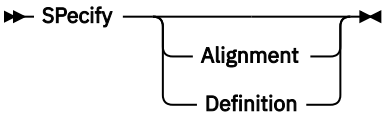
Global variables that control various displays
DSQDC global variables control the display of certain kinds of information. All of these global variables can be modified by the SET GLOBAL command.

SPECIFY

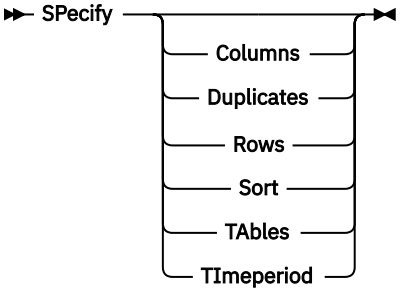
The SPECIFY command can be used in Prompted Query and on FORM.COLUMNS.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

SPECIFY with FORM.COLUMNS



SPECIFY with Prompted Query



Description

On the FORM.COLUMNS panel, SPECIFY displays a panel from which you can provide additional information about columns in the form or define new columns in the form.

ALIGNMENT
Displays the column number, column heading, heading alignment, and data alignment values. Only the heading and data alignment values can be modified.

DEFINITION
Displays the column number, column heading, and the definition for the column (if any). Only the definition value can be modified.

In Prompted Query, the Specify key displays the Specify panel, which allows you to specify the following parts of a prompted query:

COLUMNS
Specifies the columns you want in the query.

DUPLICATES
Specifies whether or not duplicate entries are to be shown.

ROWS
Allows you to specify which rows of data you want returned.

SORT
Allows you to specify how you want to sort the rows.

TABLES
Allows you to name the tables to be used in the query.

TIMEPERIOD
Include data from a specific period of time.

Usage notes

- To define a column, issue SPECIFY with the cursor on the column information line.
  - For column alignment, the cursor position (when issuing the SPECIFY command) determines which column appears in the alignment panel.
  - For column definition, the cursor position (when issuing the SPECIFY command) determines which column appears in the definition panel.
- If the cursor is not on the column information line, a panel is displayed beginning with the first column.
- On a FORM.COLUMNS panel with column definition, you can:
  - Define a column based on other columns
  - Group results based on ranges of values
  - Define user functions against individual data values
  - Display partial columns
  - Set control breaks for partial columns

  - Apply multiple usages to a single column
- SPECIFY alone displays a list of items from which to select. SPECIFY with an object displays the specified object panel.

START

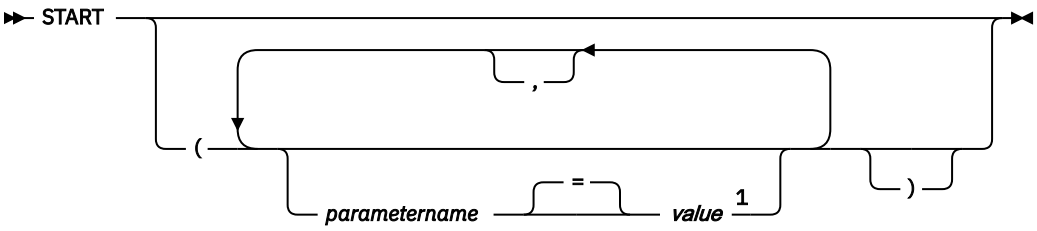
The START command begins a new QMF session.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Syntax

The syntax of the START command depends on the language you are using. The linear syntax, which is used with REXX applications, is shown here. Languages other than REXX (C, COBOL, FORTRAN, PL/I, or Assembler) use the extended syntax of the START command.

Starting a QMF session from REXX

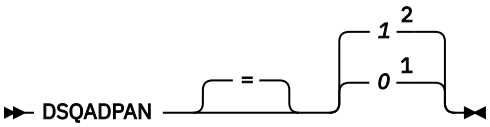


Notes:

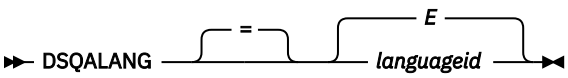
[1] For any parameter, the value NULL may be specified to explicitly indicate the default.

QMF program parameters that can be used on the START command

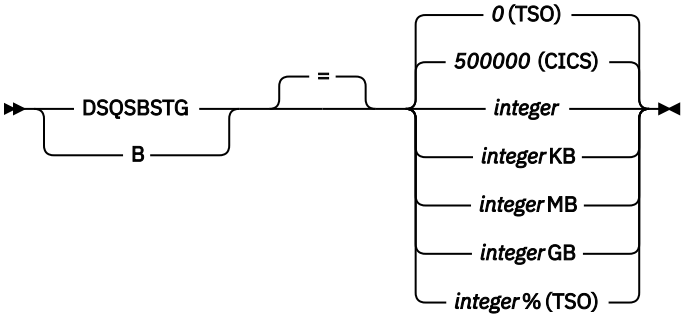
Auto report display



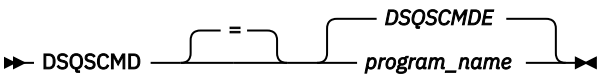
Presiding language



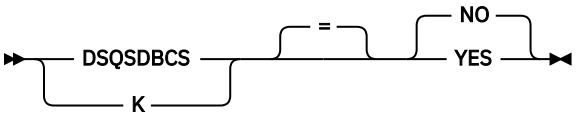
Amount of virtual storage for reports



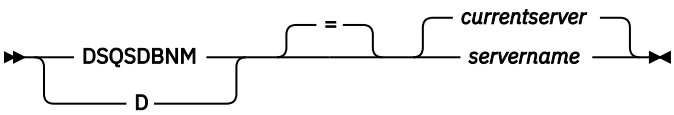
Program to pass startup parameters (TSO)



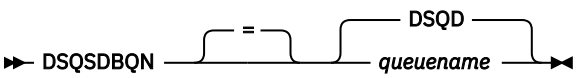
DBCS support



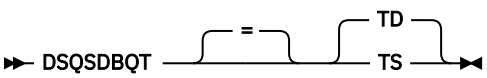
Initial database location



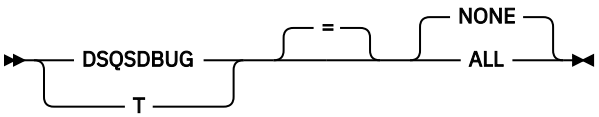
Trace data storage name (CICS)



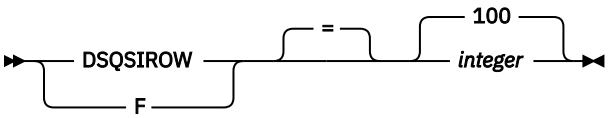
Trace data storage type (CICS)



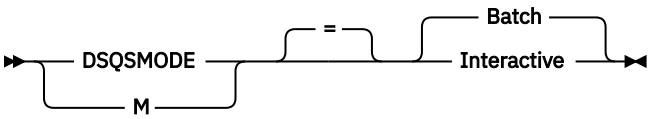
Initial trace



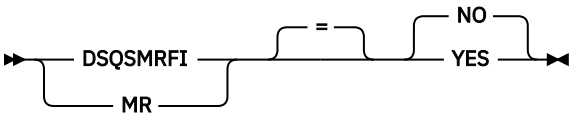
Rows fetched before display



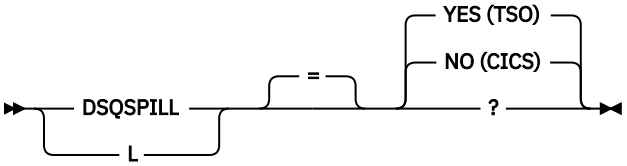
Mode of operation



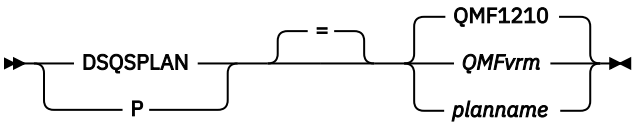
Multirow fetch and insert



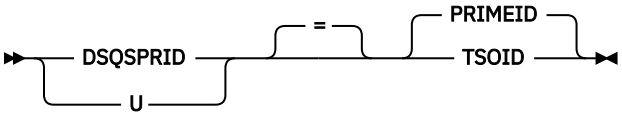
Use of auxiliary storage to hold data no longer needed in active storage



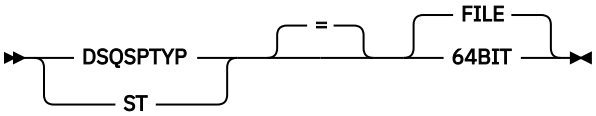
QMF application plan name (TSO)



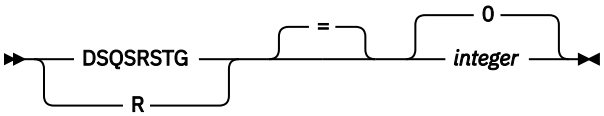
QMF profile key (TSO)



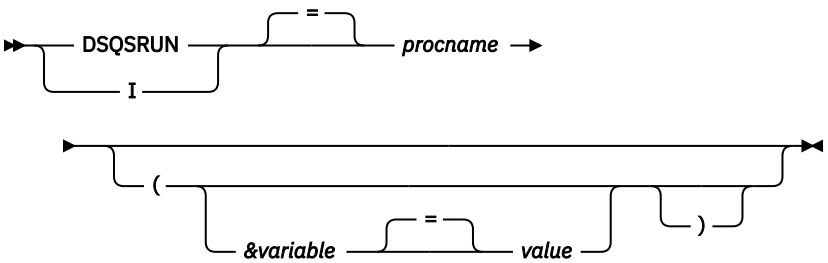
Use of extended storage to hold data no longer needed in active storage (TSO)



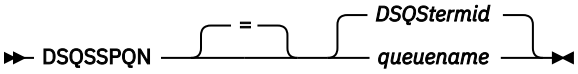
Dynamic allocation of virtual storage for reports (TSO)



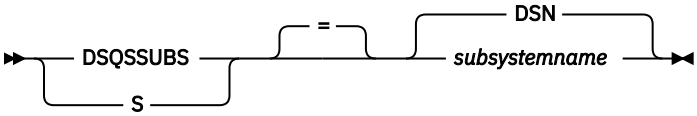
Initial QMF procedure



Spill data storage name (CICS)



Db2 subsystem ID (TSO)



Notes:

[1] When started through the callable interface
[2] When started by using a method other than the callable interface

Description

QMF *vrm*
The format for distinguishing the level of QMF, where *vrm* represents the combination of version, release, and modification-level identifiers.

DSQS*termid*
The default name for the spill data queue in a CICS environment, where *termid* represents the 4-character CICS terminal ID.

STATE

STATE is an application-support command and can be run only through the QMF command interface.

TSO with ISPF	TSO without ISPF	CICS
X		

The STATE command saves the following set of QMF global variable values in the ISPF variable pool:

- DSQALANG
- DSQAMODL
- DSQAMODP
- DSQAPCAS
- DSQAPDEC
- DSQAPGRP
- DSQAPLEN
- DSQAPLNG
- DSQAPPFK
- DSQAPPRT
- DSQAPRMP
- DSQAPSPC
- DSQAPSYN
- DSQAPTRC
- DSQAPWID
- DSQAQMF
- DSQAREVN
- DSQAROWS
- DSQASUBI
- DSQASUBP
- DSQASYST
- DSQATRAC
- DSQAVARN
- DSQCATTN

Related reference

Global variables for state information not related to the profile
DSQAO global variables contain status information or settings of parameters or flags. None of these global variables can be modified by the SET GLOBAL command.

TOP

The TOP command scrolls to the beginning of queries, procedures, reports, global variable lists, and scrollable form panels.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

Usage notes

- TOP is equivalent to BACKWARD MAX.
- To scroll to the top of footing text on form panels, position the cursor on the portion of the panel where the footing text is located and enter the TOP command.

TRACE

The TRACE command allows application programs that are written in C, COBOL, FORTRAN, PL/I, or ASSEMBLER to use the callable interface to request a service trace. This command can be issued only from within a QMF application.

Output from the TRACE command is written to the QMF DSQDEBUG data set.

TSO with ISPF	TSO without ISPF	CICS
X	X	X

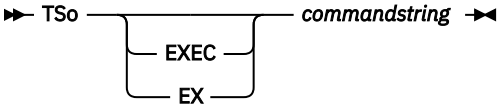
To turn on the QMF trace facility from within QMF, you can use the TRACE keyword of the SET PROFILE command or the DSQSDBUG startup parameter.

TSO

The TSO command lets you issue a command in the TSO environment without terminating your use of QMF.

TSO with ISPF	TSO without ISPF
X	X

Issuing a TSO command



Description

EXEC or EX
Indicates that the value for *commandstring* is the data set name of a CLIST or REXX program rather than a TSO command.

commandstring
A character string that constitutes a valid command or exec in the TSO environment.

Usage notes

Everything after the TSO command is sent to TSO and interpreted there.

- If execution is successful, you return to the same panel in QMF from which you entered the TSO command.
- If execution is not successful, you receive the same error message from TSO as you would if you were not going through QMF.

Examples

1. To send user ID PEGGY5 a message with the TSO SEND command:

```
TSO SEND 'I RECEIVED YOUR PROC2. THANK YOU.' USER(PEGGY5)
```

2. To run the REXX program SAMPLE in data set KELLY1.EXEC:

```
TSO EXEC 'KELLY1.EXEC(SAMPLE)'
```

Chapter 3. Basic SQL statements and functions used in QMF queries

You can issue SQL statements directly to the database from the QMF SQL Query panel. The SQL Query panel supports all SQL statements that can be run dynamically.

Selected SQL statements and keywords that are used in QMF SQL queries are described in this topic.

When you type a query on the SQL Query panel, remember to:

- Enclose reserved words in double quotation marks.

  In many cases, words that are keywords in database management systems cannot be used as the name of a table, view, column, or index in a QMF SQL query unless they are enclosed in double quotation marks.
- Enclose in double quotation marks any part of an object name that spans two or more lines.

  When any part of an object name (the location, authorization ID, or the object name itself) is continued on a new line, that part of the name must be delimited by double quotation marks. The following figure shows an example of a long object name that spans two lines. The name is qualified with an authorization ID that also spans two lines.

```
SQL QUERY                                          MODIFIED  LINE    1
SELECT * FROM "VERY_LONG_AUTHID_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX"."VERY_LONG_TABLE_NAME_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXX"
```

  Figure 9. Delimiting qualified object names that span multiple lines.
- Activate support for multiple statements and use proper syntax if the query contains more than one SQL statement.

  To include multiple SQL statements in a QMF SQL query, set the DSQEC_RUN_MQ global variable to 1 and place a semicolon at the end of every statement except the last.

  CREATE PROCEDURE and CALL statements must be used alone in a query.

  No more than one SELECT statement can be used in a query that includes other SQL statements.
- Both simple and bracketed comments can be used in an SQL query. Simple comments are introduced with two consecutive hyphens (--) and end with the end of a line. Simple comments cannot be continued to the next line. Bracketed comments are introduced with /* and end with */. Bracketed comments can be continued to the next line.

When your SQL query references an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name. QMF allows you to issue SET CURRENT SCHEMA statements to change the value of this register.

Related reference

SET special register

You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

Global variables that control how commands and procedures are executed

DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

Related information

Search for the complete SQL reference information for your database server.

ADD

You can add columns to a table only if you created the table or are specifically authorized to do so.

The following example adds one column to the description of a table named PERS:

```
ALTER TABLE PERS
ADD PHONENO SMALLINT
```

The new column is initially generated with null values. Use the UPDATE statement to provide actual values for the new column.

In Db2 for Linux, UNIX, and Windows, you can define a column as NOT NULL WITH DEFAULT, but you cannot define an added column to be NOT NULL.

NOT NULL WITH DEFAULT is invalid when your current location is a DB2 Server for VSE and VM.

ALL

A subquery generally returns only one value. However, it is possible for a query to return a set of values. With ALL, each value in the returned set must be satisfied.

To permit a query to return a set of values, rather than an individual value, use the ALL keyword with one of the following comparison operators:

```
          =    ¬=    >    >=    <    <=
```

The symbol ¬= is an alternative symbol for < > (not equal to). It is an American National Standards Institute (ANSI) SQL operator. (If you are using remote data access, the preferred symbol is < >.)

The following query produces a report that lists the department with the highest average salary. Use the ALL keyword to specify that the department selected by the main SELECT statement must have an average salary equal to or greater than all average salaries of other departments.

```
SELECT DEPT, AVG(SALARY) FROM Q.STAFF
GROUP BY DEPT
HAVING AVG(SALARY) >= ALL
      (SELECT AVG(SALARY) FROM Q.STAFF
       GROUP BY DEPT)
```

Operators other than the equal sign (=) can be used with the ALL keyword. If any of the results produced by the subquery are null, the result of the condition with ALL is unknown.

ALTER TABLE

You can alter a table only if you created the table or are specifically authorized to do so. The ALTER TABLE statement specifies which existing table to change.

For example, following ALTER TABLE, you can use the ADD statement to add a new column on the right side of a table.

Related reference

ADD

Use the ADD command to add rows to a table in the Table Editor or add global variables to the global variable list.

AND

You can select rows based on multiple conditions connected by AND or OR.

Two conditions connected by AND select only rows that satisfy both conditions. An example is shown below.

This query:

```
SELECT ID, NAME, YEARS, SALARY
FROM Q.STAFF
WHERE YEARS = 10 AND SALARY > 20000
```

Produces this report:

```
   ID  NAME       YEARS     SALARY
-----  ---------  -----  ---------
   50  HANES         10   20659.80
  210  LU            10   20010.00
```

If you use both AND and OR, use parentheses to specify the order in which AND and OR conditions are evaluated. The following examples show how using parentheses affects the order of evaluation in clauses that include the AND keyword.

- With parentheses:

  The following clause selects employees that satisfy at least one of these conditions:

  – The employee's job is sales and the employee's commission is more than $1,200.
  – The employee has more than 10 years of service.

  ```
  WHERE (JOB='SALES' AND COMM > 1200) OR YEARS > 10
  ```

  The query in which this clause appears returns information for the following employee IDs: 90, 260, 310, 340.

  You can use more than one level of parentheses. The condition is evaluated from the innermost level of nested parentheses outward, as in algebraic expressions.

- Without parentheses:

  If you do not use parentheses, all conditions connected by AND are evaluated and connected first, then conditions connected by OR are evaluated. That is, if A, B, and C are conditions, these two phrases produce the same results:

  ```
      A AND B OR C
    (A AND B) OR C
  ```

Related reference

OR

You can select rows based on multiple conditions connected by OR. Conditions connected by OR select every row that satisfies any of the conditions.

ANY

A subquery generally returns only one value. However, it is possible for a query to return a set of values.

To permit a query to return a set of values rather than an individual value, the ANY keyword can be used with the following comparison operators:

```
=   ¬=   >   >=   <   <=
```

With ANY, at least one value in the set returned must be satisfied.

IN can be used in a subquery in place of = ANY, and SOME is a synonym for ANY.

The symbol ¬= is an alternative symbol for < > (not equal to). It is an ANSI SQL operator. (If you are using remote data access, the preferred symbol is < >.)

The following query produces a list of employees who work in the Eastern division. First, the subquery finds the department numbers in the Eastern division. Then, the main query finds the employees who work in any of these departments.

The following query produces a list of names and IDs of employees who work in the Eastern division:

```
SELECT NAME, ID
FROM Q.STAFF
WHERE DEPT = ANY
 (SELECT DEPTNUMB FROM Q.ORG WHERE DIVISION='EASTERN')
```

The keyword ANY was used in this query because there are multiple departments in the Eastern division. If ALL is used instead of ANY, the result is an empty set. (No employee works in all the departments of the Eastern division.)

AS

You can use an AS clause in a SELECT statement to name or rename a result column in a query. The name must not be qualified and does not have to be unique.

The following example shows the use of an AS clause in a query submitted in Db2 for z/OS:

```
SELECT NAME, SALARY*0.05 AS "RAISE"
  FROM Q.STAFF
```

If the AS clause is not specified and the result column is derived from a column name, the result column name is the unqualified name of that column.

AVG

AVG is a column function valid only on columns containing numeric data.

The following example includes more than one column function in the SELECT statement. For Department 10, it calculates and displays the following: sum of employee salaries; the minimum, average, and maximum salary; and the number of employees (COUNT) in the department.

This query:

```
SELECT SUM(SALARY), MIN(SALARY), AVG(SALARY),
   MAX(SALARY), COUNT(*)
FROM Q.STAFF
WHERE DEPT = 10
```

Produces this report:

```
       COL1        COL2             COL3        COL4              COL5
----------- ----------- ---------------- ----------- -----------------
   83463.45    19260.25 20865.8625000000    22959.20                 4
```

Write the AVG column function like this:

```
AVG(expression)
```

The parentheses are required. In the above syntax, *expression* is most often a column name, but can also be:

- An arithmetic expression containing at least one column name
- The DISTINCT keyword, followed by a column name

A column name in a function must not refer to a long-string column or a column derived from a column function (a column of a view can be derived from a function). Column functions cannot be nested within other column functions. Null values are not included in the calculation made by a column function.

You cannot use the AVG function on a column if the sum of the data in the column would cause an overflow condition.

BETWEEN x AND y

You can retrieve data from each row whose column, named in a WHERE clause, has a value within two limits. Use BETWEEN in place of an AND condition when using greater than or equal to (>=) and less than or equal to (<=) operators.

The limits you specify are inclusive. Enter the lower boundary (smaller value) of the BETWEEN condition first, then the upper boundary (larger value). The following example selects employees who have a salary between $20,000 and $21,000. GRAHAM has a salary of exactly $21,000.

This query:

```
SELECT ID, NAME, SALARY
FROM Q.STAFF
WHERE SALARY BETWEEN 20000 AND 21000
```

Produces this report:

```
  ID  NAME          SALARY
----- ---------  ---------
  50  HANES       20659.80
 210  LU          20010.00
 310  GRAHAM      21000.00
```

Examples:

- To select everyone whose name is alphabetically between HANES and MOLINARE:

  ```
  SELECT ID, NAME
  FROM Q.STAFF
  WHERE NAME BETWEEN 'HANES' AND 'MOLINARE'
  ```

- To select everyone who has between 10 and 12 years of service (inclusive):

  ```
  SELECT ID, NAME, YEARS
  FROM Q.STAFF
  WHERE YEARS BETWEEN 10 AND 12
  ```

- To select employees whose salary is not in the range of $19,000 to $21,000:

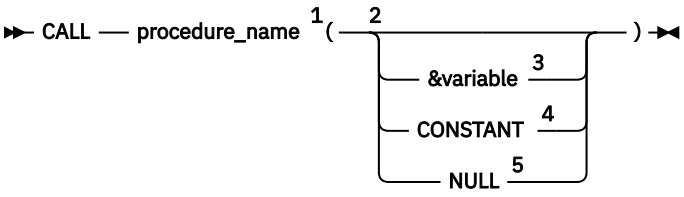
  ```
  SELECT ID, NAME, YEARS
  FROM Q.STAFF
  WHERE SALARY NOT BETWEEN 19000 AND 21000
  ```

  Each employee whose salary is less than $19,000 or more than $21,000 is included in the report. Employees with salaries between and including $19,000 and $21,000 are not included.

CALL

To run a stored procedure from within a QMF session, you must issue a CALL statement from the **SQL Query** panel. The database to which the CALL statement is directed must support the ability to call a stored procedure.

After you enter a CALL statement, a RUN command is issued to run the stored procedure.



Notes:

[1] This identifies the stored procedure to call.
[2] Parameter values can be in, out, or inout parameters.
[3] This identifies a QMF substitution variable to be used as input or output to the stored procedure.
[4] This identifies a CONSTANT to be used as input or output to the stored procedure.
[5] The parameter is a NULL value. The corresponding parameter of a stored procedure must be defined as IN, and the description of the stored procedure must allow for NULL parameters.

The CALL statement must be used alone in a SQL query. It cannot be combined with other statements.

How parameters are used

QMF supports up to 63 parameters on the CALL statement. Parameters on the CALL statement are used in the following way:

- Input parameters (IN)

  Input values passed to the stored procedure.
- Output parameters (OUT)

  The names of user-defined QMF substitution variables receive the values of the output variables that are returned from the stored procedure. Before you use the CALL statement, these names must be set by the user with the QMF SET GLOBAL command.

  You must use a QMF global variable to specify output parameters for a stored procedure if you want to view the output. The output parameters can then be displayed using the SHOW GLOBALS command. A maximum of 10 QMF global variables can be entered from the **SQL Query** panel. The maximum size of a QMF substitution variable is 32 KB.

  The CALL statement fails when OUT parameters defined for the stored procedure are not initialized correctly. QMF global variables whose values are copied into output parameters for the stored procedure have special initialization requirements:

  – An output parameter with a numeric data type must be initialized to 0.
  – An output parameter with a data type of CHAR must be initialized to blank or NULL.
- Input/output parameters (INOUT)

  Can be used as input or output, and can have the behavior of either input or output parameters.

Guidelines for using the CALL statement

- CALL statements in QMF can be directed only to Db2 for z/OS databases.
- QMF does not process three-part names that are referenced in CALL statements. Only stored procedures at the current location (the location to which QMF is connected) are run. If a three-part name is entered, QMF accepts it, but an error message is issued if the location entered does not match the current location.
- If a schema name is not provided for the stored procedure name, QMF uses the value of the CURRENT SQLID register as the schema name.
- Authorization checking is done by the database. The current SQLID must be authorized to run the stored procedure that is specified in the CALL statement.

- Parameters that are defined with data types of DATE, TIME, TIMESTAMP, or TIMESTAMP WITH TIME ZONE must have their values enclosed in single quotation marks. QMF handles these data types as character strings.
- Data of the following types cannot be passed in a parameter on the CALL statement: BINARY, VARBINARY, VARGRAPHIC, GRAPHIC, LONG VARGRAPHIC, CLOB, BLOB, DBCLOB, ROWID, and XML. DECFLOAT data can be passed if the processor on which QMF is running supports decimal floating-point instructions.
- QMF supports the return of the first 63 result sets when a stored procedure that returns result sets is run. Select one by setting the global variable DSQEC_SP_RS_NUM.
- The maximum data size of a LOB column that is to be returned from a stored procedure is determined by the DSQEC_LOB_COLMAX global variable.

How to write a CALL statement with long identifiers

A single SQL query line is limited to 79 bytes on the QMF **SQL Query** panel. An identifier that spans more than one line in a CALL statement that is entered on the **SQL Query** panel must be a delimited identifier. Here are some examples that show how to code long CALL statements:

- A long parameter as a delimited identifier that spans more than one line:

```
CALL USERID.PROC (’THIS IS THE FIRST PARM’, 4, 1954, "THIS IS ANOTHER
 PARM THAT WILL SPAN TWO LINES ON THIS PANEL", 14, 99)
```

- A long stored procedure name as a delimited identifier that spans more than one line:

```
CALL USERID.’THIS_IS_A_REALLY_LONG_STORED_PROCEDURE_NAME_THAT_EXCEEDS_
MORE_THAN_ONE_LINE_ON_THE_QUERY_PANEL' (’PARM1’, ’ ’, 0, ’PARM4’)
```

- Break the lines between identifiers:

```
CALL USERID.PROC ('THIS IS THE FIRST PARM’, 4, 1964,
’THIS IS ANOTHER PARM THAT WILL NOW FIT ON THIS LINE’,
14, 99)

CALL USERID.PROC (’THIS IS THE FIRST PARM’, 333333,
12345678901234567890123456789O, 200305,
’THIS IS THE LAST PARM’)
```

- Use a delimiter (in this case, double quotation marks) when the text spans more than two lines:

```
CALL USERID.PROC ("THIS IS THE FIRST PARM AND IT WILL NOT ONLY EXTEND
PAST THE FIRST LINE, IT WILL ALSO EXTEND BEYOND THE SECOND LINE BECAUSE
THERE ARE TOO MANY WORDS TO FIT IN TWO LINES ALONE").
```

Specifying a QMF form for data returned in a result set

If you do not specify a form on the RUN QUERY command that issues the CALL statement, a default form is created based on the returned result set. If the stored procedure returns more than one result set, you can display one of the result sets by specifying its number in the global variable DSQEC_SP_RS_NUM; the rest of the result sets are ignored.

If the RUN QUERY command that calls the stored procedure includes the FORM parameter, ensure that the specified form matches the data returned in the result set, or QMF issues an error message. In this case, you can load the proper form using the DISPLAY FORM command, or modify or reset the current form to match the returned data.

Related reference

Global variables that control how commands and procedures are executed

DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

COMMIT

The COMMIT statement applies all database changes that were made during the unit of work that contains the commit statement.

If confirmation prompts are enabled and a query contains one or more COMMIT statements, a confirmation panel is displayed for every COMMIT statement. Your responses to the confirmation prompts apply to all changes to the database that occurred since the beginning of the query or after the last COMMIT statement. However, if a COMMIT statement follows SQL statements that change only a database catalog, a confirmation panel is not displayed for that COMMIT statement.

If a query contains multiple statements and one of the statements fails, processing stops and no subsequent statements are run. If a multistatement query contains one or more COMMIT statements and an error occurs, processing stops and no subsequent statements are run. All database changes that occurred before the SQL error and after the last successful COMMIT statement are rolled back. Some statements, such as SET, apply to the QMF session or environment and therefore are not rolled back in error situations.

Example

In the following example, if confirmation prompts are enabled, a confirmation prompt is issued after the first two COMMIT statements. However, no prompt panel displays for the third COMMIT statement because it follows an ALTER statement, which changes only a database catalog.

```
CREATE TABLE MYSTAFF2 LIKE Q.STAFF;
INSERT INTO MYSTAFF2 SELECT * FROM Q.STAFF;
COMMIT;

INSERT INTO W397754.MYSTAFF2
(ID, "NAME", DEPT, JOB, "YEARS", SALARY, COMM)
   VALUES (99,'WILLY',22,'SUB',2,1.00,0.0);
INSERT INTO W397754.MYSTAFF2
(ID, "NAME", DEPT, JOB, "YEARS", SALARY, COMM)
   VALUES (99,'WILLY2',22,'SUB',2,1.00,0.0);
COMMIT;

ALTER TABLE MYSTAFF2
ADD COMMENT CHAR(30);
COMMIT;

UPDATE MYSTAFF2
SET COMMENT = 'UPDATE FOR WILLIAMS'
WHERE NAME = 'WILLIAMS';
SELECT * FROM MYSTAFF2;
```

COUNT

The COUNT function counts only non-null values. Therefore, the data type of the result of the COUNT function always has the NOT NULL attribute.

There are two uses of COUNT:

- COUNT with the DISTINCT keyword, which has two forms:
  - COUNT(DISTINCT *colname*)

    Counts rows returned in which there is a non-null value in a named column. It eliminates duplicates from the count.

    This format must be used with a column name; it cannot be used with an expression. An example of this form of the COUNT function is shown below:

    ```
    SELECT COUNT(DISTINCT DIVISION)
    FROM Q.ORG
    ```

The result is 4.

– COUNT(DISTINCT *expression*)

Returns distinct values for columns within a group. For example, the following query returns the number of distinct education levels of the job applicants in the Q.APPLICANT table as well as the average number of years of education the applicants have.

```
SELECT COUNT(DISTINCT EDLEVEL), AVG(EDLEVEL)
FROM Q.APPLICANT
```

- COUNT(*)

Counts rows returned regardless of the value of any column. This format is not used with a column name. For example:

```
SELECT SUM(SALARY), MIN(SALARY), AVG(SALARY),
   MAX(SALARY), COUNT(*)
FROM Q.STAFF WHERE DEPT = 10
```

This example includes more than one column function in the SELECT statement. It calculates and displays, for Department 10, the sum of employee salaries; the minimum, average, and maximum salary; and the number of employees (COUNT) in the department. It produces the following report:

```
               COL1        COL2                 COL3        COL4         COL5
------------------  ----------  -------------------  ----------  -----------
          83463.45    19260.25    20865.8625000000    22959.20            4
```

Related reference

DISTINCT

Use the DISTINCT keyword before the column names in an SQL statement to prevent duplicate rows from being selected.

CREATE SYNONYM

The CREATE SYNONYM statement defines an alternative name for a table or view. This lets you refer to a table owned by another user without having to enter the fully qualified name.

You can also create synonyms for your own tables and views. The synonym remains defined until it is dropped from the database.

The following example creates a new name for the table Q.APPLICANT.

```
CREATE SYNONYM APPLS FOR Q.APPLICANT
```

After executing this statement, you can use APPLS in all commands and statements where you previously used Q.APPLICANT.

A synonym is only of value when it is shorter than the fully-qualified table name. It can also be a valuable protection for your queries if you are using tables created by someone else. For example, suppose that the Q.APPLICANT table is dropped and re-created by user BDJ1385L. All your queries were written using the synonym APPLS. Your first step is to drop the synonym by using this command:

```
DROP SYNONYM APPLS
```

Then make this change:

```
CREATE SYNONYM APPLS FOR BDJ1385L.APPLICANT
```

If you share a query that uses a synonym, it will not work for the users with whom you have shared it until those users create the same synonym. You cannot share synonyms you define under your authorization ID. However, other users can define the same synonyms with the same meanings.

If your site uses DBCS data, do not create a synonym that contains double-byte characters that are internally represented as double quotation marks unless your database specifically supports double-byte characters in table names.

Related reference

Naming conventions
Ensure that names of your objects adhere to the naming conventions for QMF.

CREATE TABLE

The CREATE TABLE statement defines a table. You provide the name of the table and the names and attributes of its columns. You can grant or revoke authorization for other people to use a table you created.

Syntax

The syntax of the CREATE TABLE statement is:

```
CREATE TABLE tablename (column1 type1 NOT NULL,
column2 type2 . . .)
  IN space-name
```

tablename
The name that you assign to the table.

If your site uses DBCS data, names of tables cannot contain double-byte characters that are internally represented as double quotation marks unless your database specifically supports double-byte characters in table names.

column1 type1
The name that you assign to the first column and the data type describing it. If the data type is CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, or DECIMAL, you must specify the maximum length of a data element in parentheses. For DECIMAL, you must also specify the number of places after the assumed decimal point.

column2 type2
The name that you assign to the second column and the data type describing it.

NOT NULL
Optional for any column you define. If you use NOT NULL in the table definition, then any attempt to have no value in the corresponding column of the table produces an error message. Omitting NOT NULL allows null values in the column.

IN *space-name*
Refers to a table space or a dbspace in which the table is to be created. This clause is needed only if your site does not provide a space to be used by default.

To find the space name that is used when QMF creates tables for SAVE DATA or IMPORT TABLE commands, issue the QMF command DISPLAY PROFILE and look at the value of the SPACE option.

Examples

The following CREATE statement defines a table called PERS. The columns in PERS have the same characteristics as Q.STAFF, but contain no data.

```
CREATE TABLE PERS
(ID SMALLINT NOT NULL,
NAME VARCHAR(9),
DEPT SMALLINT,
JOB CHAR(5),
YEARS SMALLINT,
SALARY DECIMAL(7,2),
COMM DECIMAL(7,2))
IN space-name
```

ID
The employee number is a small integer and null cannot be specified for it.

NAME
The maximum length of the name is nine characters.

DEPT
The data type of the department number column is small integer.

JOB
The name of the job has five characters.

YEARS
The number of years is small integer.

SALARY
A seven-digit number with two decimal positions.

COMM
A seven-digit number with two decimal positions. (Remember the final parenthesis.)

You can use NOT NULL with any set of columns in the CREATE TABLE statement; in the example, it appears with the ID column. It means that any row entered into PERS must have, at the very least, an employee number.

This statement defines the Q.APPLICANT table:

```
CREATE TABLE APPLICANT
(TEMPID  SMALLINT  NOT NULL,
NAME     VARCHAR(9),
ADDRESS  VARCHAR(17),
EDLEVEL  SMALLINT,
COMMENTS VARCHAR(29))
IN space-name
```

This statement defines the Q.INTERVIEW table:

```
CREATE TABLE INTERVIEW
(TEMPID   SMALLINT,
INTDATE   DATE,
STARTTIME TIME,
ENDTIME   TIME,
MANAGER   SMALLINT,
DISP      VARCHAR(6),
LASTNAME  VARCHAR(9),
FIRSTNAME VARCHAR(9))
IN space-name
```

Defining the table does not put data into it.

Related reference
GRANT
The GRANT statement gives users authorization to perform one or more operations on a table.

INSERT
INSERT is an SQL statement that adds data to a table.

Naming conventions
Ensure that names of your objects adhere to the naming conventions for QMF.

REVOKE

The REVOKE statement removes authorization granted by a GRANT statement.

CREATE VIEW

A view is a logical table that contains data selected from existing tables. The view can rename and rearrange columns, omit unwanted columns or rows, define columns by expressions, group results, and combine more than one table.

Views make it possible to view data that exists in parts of one or more tables. No data actually exists in a view.

Any SELECT statement that does not contain an ORDER BY clause can be used as the basis of a view; the selected columns and rows become the columns and rows of the view. In the following example, the NAME, ID, and JOB columns from the Q.STAFF table become the columns of the D42 view. The column names for D42 are LAST NAME, EMP. ID, and JOB.

```
CREATE VIEW D42
 ("LAST NAME", "EMP. ID", JOB)
   AS SELECT NAME, ID, JOB
FROM Q.STAFF
WHERE DEPT = 42
```

Issue the command `DISPLAY TABLE D42` to display this view:

```
LAST NAME EMP. ID  JOB
--------- -------  -----
KOONITZ        90  SALES
PLOTZ         100  MGR
YAMAGUCHI     130  CLERK
SCOUTTEN      200  CLERK
```

There are two main reasons for using a view:

- To simplify writing a query.
- To prevent access to data. Anyone using the view D42 in the above example cannot see salary data.

Use a view by its name, like you use a table name. You can select from it, writing the same kind of SELECT statement as if it were a table. For example, you can run the following query for the D42 view:

```
SELECT * FROM D42
WHERE JOB='CLERK'
```

With a few restrictions, you can insert, update, and delete rows in a view. Corresponding changes are made to the tables the view is based on.

There are a few things that you cannot do with a view:

- You cannot insert, update, or delete data using a view if the view contains:
  - Data from more than one table
  - A column defined by a column function; for example, SUM(SALARY)
  - Data selected by the DISTINCT or GROUP BY keywords
- You cannot update or insert data if the view contains a column defined by an expression (like SALARY/12). However, you can delete data in this case.
- You cannot use the UNION keyword when creating a view.
- You cannot join a view created using a GROUP BY clause to another table or view.

DELETE

You can delete rows from a table only if you created the table or are specifically authorized to do so. You can delete information from a table by row. Individual fields in a row or complete columns of information cannot be deleted.

The DELETE statement consists of two parts:

DELETE FROM
The table from which rows are to be deleted.

WHERE
Criteria that determines which rows will be deleted.

If DELETE is entered with no WHERE clause specified, all rows of the table are deleted. The table still exists, but it no longer contains any rows.

The following statement deletes employee number 140 from the PERS table.

```
DELETE FROM PERS
WHERE ID = 140
```

In this example, ID, rather than employee name, is used to avoid deleting more rows than anticipated, because there could be more than one employee with the same name.

You can delete more than one row with one DELETE statement by including a condition to show which rows to delete. The following example deletes everyone in Department 10:

```
DELETE FROM PERS
WHERE DEPT = 10
```

Related reference
GRANT
The GRANT statement gives users authorization to perform one or more operations on a table.

DISTINCT

Use the DISTINCT keyword before the column names in an SQL statement to prevent duplicate rows from being selected.

Examples

The following example lists only the unique divisions that exist in the Q.ORG table:

This query:

```
SELECT DISTINCT DIVISION
FROM Q.ORG
```

Produces this report:

```
DIVISION
---------
CORPORATE
EASTERN
MIDWEST
WESTERN
```

Compare this result with the following example:

This query:

```
SELECT DIVISION
FROM Q.ORG
```

Produces this report:

```
DIVISION
----------
CORPORATE
EASTERN
EASTERN
EASTERN
MIDWEST
MIDWEST
WESTERN
WESTERN
```

DISTINCT can also select distinct combinations of data. For example:

```
SELECT DISTINCT DEPT, JOB
 FROM Q.STAFF
 ORDER BY DEPT
```

The report that is produced from this example shows the jobs that are represented in every department.

Remember these properties when you use DISTINCT:

- DISTINCT comes after SELECT.
- DISTINCT comes before the first column name and is not separated from the column name with a comma.
- DISTINCT applies to all the columns that are selected.

DISTINCT can be used with COUNT.

Use DISTINCT with other column functions when you want only the distinct values for the columns within a group to be used. For example, AVG(DISTINCT PRICE) ignores duplicate prices in the column and averages a list in which each price appears once. AVG(PRICE) averages all the prices in the column without regard to the fact that some prices are duplicates.

To list the different values that appear for YEARS, use a query like the following query:

```
SELECT DISTINCT YEARS
FROM Q.STAFF
ORDER BY YEARS
```

To list the department numbers for departments in which at least one employee has 10 or more years of service, use a query like the following query:

```
SELECT DISTINCT DEPT
FROM Q.STAFF
WHERE YEARS >= 10
```

Related reference

COUNT
The COUNT function counts only non-null values. Therefore, the data type of the result of the COUNT function always has the NOT NULL attribute.

DROP

The DROP statement erases tables, views, synonyms, aliases, and other objects (like indexes) from the database.

You must have the authority to drop tables or views from the database. To drop a synonym, you must be its owner. To drop an alias, you must be the owner or have SYSADM or SYSCTRL authority.

The syntax of the DROP statement is:

```
DROP object object-name
```

object
TABLE, VIEW, SYNONYM, or ALIAS

object-name
The name by which the object is known in the database.

The following table shows some examples of effects of using the DROP statement.

Table 10. Examples of using the DROP statement

This statement:	**Erases this object:**
`DROP TABLE PERS`	The table PERS
`DROP VIEW D42`	The view D42
`DROP SYNONYM APPLS`	The synonym APPLS
`DROP ALIAS PROJECTIONS1`	The alias PROJECTIONS1

Attention: Use DROP TABLE with extreme caution. Dropping a table destroys the data in it, and destroys any views that are based on the table. If you rebuild the table after you drop it, you will need to regrant any privileges that were granted on the table or any of its views.

Issuing the QMF command ERASE TABLE *name* is equivalent to running any of the following commands:

```
DROP TABLE name
DROP VIEW name
DROP SYNONYM name
DROP ALIAS name
```

DROP VIEW does not affect any tables that the view is based on, and does not destroy tables in the database. A view that was dropped can easily be re-created. However, DROP VIEW revokes any privileges that are granted on the view.

DROP SYNONYM removes the synonym from a dictionary of synonyms, so it no longer refers to anything in the database. It has no effect on the tables or views the synonym accessed. For example, if APPLS is in the synonym table for Q.APPLICANT, executing the statement DROP SYNONYM APPLS does not affect Q.APPLICANT.

EXISTS

The EXISTS statement determines whether a row exists that satisfies a given condition.

This is shown in the subquery of the following query:

```
SELECT ID, NAME, DEPT
FROM Q.STAFF CORRVAR
WHERE EXISTS
  (SELECT * FROM Q.ORG WHERE MANAGER = CORRVAR.ID)
```

Related reference

IN
Use the IN statement to retrieve rows that match at least one value in a group of values you specify.

GRANT

The GRANT statement gives users authorization to perform one or more operations on a table.

You must be authorized to insert, update, delete, alter, or select rows in a table you do not own. Authorization must be granted by the creator of the table or by someone to whom the creator granted such authorization.

The syntax of the GRANT statement is:

```
GRANT operation-list ON tablename
TO user-list WITH GRANT OPTION
```

operation-list
One or more of the following privileges that are separated by commas: ALTER, DELETE, INSERT, SELECT, UPDATE (*column-list*).

ALL grants authorization to perform all operations for which the grantor is authorized.

tablename
Names a table or view for which the authorization is granted.

user-list
Lists each user ID with commas between them. PUBLIC can be specified in place of *user-list* to grant authorization to all users.

WITH GRANT OPTION clause
Authorizes another user to use the GRANT keyword to grant the same privileges to other users. This clause is optional.

The following statement grants authorization to all other users to display the PERS table or issue SELECT statements that select data from it:

```
GRANT SELECT ON PERS TO PUBLIC
```

The following statement grants authorization to user HSAM4419 to insert and delete rows in PERS:

```
GRANT INSERT, DELETE ON PERS TO HSAM4419
```

The following statement grants authorization to SMITH to update PERS and to grant this authorization to other users:

```
GRANT UPDATE ON PERS TO SMITH WITH GRANT OPTION
```

Related reference
REVOKE
The REVOKE statement removes authorization granted by a GRANT statement.

Related information
To find information about granting authorization, see the SQL reference information for the database you are using.

GROUP BY

The GROUP BY statement identifies a selected column to use for grouping results. It divides the data into groups by the values in the column specified, and returns one row of results for each group.

You can use GROUP BY with more than one column name (separate column names with commas). Always place GROUP BY after FROM and WHERE in a query, and before HAVING and ORDER BY.

All selected columns without an associated aggregation must appear in the GROUP BY clause.

GROUP BY accumulates the results by group, but does not necessarily order the groups; you need an ORDER BY statement to do that. When you retrieve multiple rows from a table, the GROUP BY, HAVING, and ORDER BY clauses can be used to indicate:

- How you want the rows grouped (GROUP BY)
- A condition that the rows, as a group, must meet (HAVING)
- The order in which you want the rows returned to you (ORDER BY)

For example, the following query selects the average salary for each department:

```
SELECT DEPT, AVG(SALARY)
FROM Q.STAFF
GROUP BY DEPT
```

This query produces the following report:

```
    DEPT                COL1
  ------  ------------------
      10    20865.8625000000
      15    15482.3325000000
      20    16071.5250000000
      38    15457.1100000000
      42    14592.2625000000
      51    17218.1600000000
      66    17215.2400000000
      84    16536.7500000000
```

In the above example, GROUP BY divides the table into groups of rows with the same department number, and returns one row of results for each group. The DEPT column can be selected without a built-in function because it is used with GROUP BY, and because every member of each group has the same value in the DEPT column. All column names included in a SELECT clause must either have an associated built-in function or must appear in the GROUP BY clause. For example, if DEPT is not used in the GROUP BY clause, the list of average salaries has little meaning.

The following query is correct:

```
SELECT DEPT, AVG(SALARY), JOB
FROM Q.STAFF
GROUP BY DEPT, JOB
```

The following query is incorrect:

```
SELECT DEPT, AVG(SALARY), JOB
FROM Q.STAFF
GROUP BY DEPT
```

Generally, GROUP BY produces one row of a report for each different value of the column specified in the GROUP BY clause. When there are several columns named in the GROUP BY clause, a new row is produced in the report each time a value in one of those columns changes. However, if there are null values in the column, each null value is treated as a separate group, consisting of one member.

Using GROUP BY in SQL is an alternative to using the usage code GROUP on the form. GROUP BY provides an extension to the grouping that can be specified on the form and it allows conditional selection of data, which cannot be done on the form. For example, to see the least, greatest, and average of total department salaries:

1. Write and run this query:

   ```
   SELECT DEPT, SUM(SALARY), SUM(SALARY), SUM(SALARY)
   FROM Q.STAFF
   GROUP BY DEPT
   ```

2. Use these usage codes on the form:

   ```
   NUM COLUMN HEADING                      USAGE
   --- ----------------------------------  -------
    1  DEPT
    2  SUM(SALARY)                         MINIMUM
    3  SUM(SALARY)1                        AVERAGE
    4  SUM(SALARY)2                        MAXIMUM
   ```

The report contains four columns, of which the last three are almost identical. All three show the total salary for each department; but the final row shows the minimum, average, and maximum of the totals.

Additional examples:

- To list the largest and smallest salary by job for each department, excluding managers, use a query like the following:

```
SELECT DEPT, JOB, MIN(SALARY), MAX(SALARY)
FROM Q.STAFF
WHERE JOB < >'MGR'
GROUP BY DEPT, JOB
```

- To list, for each number of years of service, the number of employees with that number of years and their average salaries, use a query like the following:

```
SELECT YEARS, COUNT(*), AVG(SALARY)
FROM Q.STAFF
GROUP BY YEARS
```

  The HAVING keyword must be used with grouped data. When the HAVING statement and the GROUP BY statement are both used, the HAVING statement must follow the GROUP BY statement.

- To list the least, greatest, and average salary in each department, excluding managers, for departments with an average salary greater than $12,000, use a query like the following:

```
SELECT DEPT, MIN(SALARY), MAX(SALARY), AVG(SALARY)
FROM Q.STAFF
WHERE JOB < > 'MGR'
GROUP BY DEPT
HAVING AVG(SALARY) > 12000
```

- To list, for each number of years of service, the number of employees with that number of years and their average salaries, but only for groups with more than two employees, use a query like the following:

```
SELECT YEARS, COUNT(*), AVG(SALARY)
FROM Q.STAFF
GROUP BY YEARS
HAVING COUNT(*) > 2
```

Related reference

GROUP usage code

The GROUP usage code identifies a column by which to group data for summaries. For example, you can group data from an employee table by department.

HAVING

The HAVING clause filters results obtained by the GROUP BY clause. In the following example, the clause HAVING COUNT(*) > 4 eliminates all departments with four or fewer members from the final result.

```
SELECT DEPT, AVG(SALARY)
FROM Q.STAFF
GROUP BY DEPT
HAVING COUNT(*) > 4
```

The query produces this report:

```
  DEPT                COL1
------  ------------------
    38    15457.1100000000
    51    17218.1600000000
    66    17215.2400000000
```

Both WHERE and HAVING eliminate unwanted data from your report. The WHERE condition is used with column selection. It determines whether an individual row is included. The HAVING condition is used with built-in functions. It determines whether an entire group is included.

HAVING is always followed by a column function (such as SUM, AVG, MAX, MIN, or COUNT). HAVING can also be followed by a subquery that finds a grouped value to complete the HAVING condition. Use WHERE to eliminate unwanted row data and HAVING to eliminate unwanted grouped data.

For example:

- This is correct: `HAVING MIN(YEARS) > 6`
- This is incorrect: `HAVING YEARS > 6`

Additional examples:

- To list the least, greatest, and average salary in each department, excluding managers, for departments with an average salary greater than $12,000, use a query like the following:

```
SELECT DEPT, MIN(SALARY), MAX(SALARY), AVG(SALARY)
FROM Q.STAFF
WHERE JOB <> 'MGR'
GROUP BY DEPT
HAVING AVG(SALARY) > 12000
```

  This query produces the following report:

```
  DEPT        COL1        COL2                COL3
------  ----------  ----------  ------------------
    15    12258.50    16502.83    13756.5100000000
    20    13504.60    18171.25    15309.5333333333
    38    12009.75    18006.00    14944.7000000000
    42    10505.90    18001.75    13338.7500000000
    51    13369.80    19456.50    16235.2000000000
    66    10988.00    21000.00    16880.1750000000
    84    13030.50    17844.00    15443.0000000000
```

  The HAVING keyword can only be used with grouped data. When HAVING and GROUP BY clauses are both used, the HAVING clause must follow the GROUP BY clause.

- To list, for each number of years of service, the number of employees with that number of years and their average salaries, but only for groups with more than two employees, use a query like the following:

```
SELECT YEARS, COUNT(*), AVG(SALARY)
FROM Q.STAFF
GROUP BY YEARS
HAVING COUNT(*) > 2
```

  This query produces the following report:

```
 YEARS         COL1                COL2
------  -----------  ------------------
     5            5    15552.0400000000
     6            6    16930.0250000000
     7            6    18611.8050000000
    10            3    20162.6000000000
     -            4    13694.0625000000
```

IN

Use the IN statement to retrieve rows that match at least one value in a group of values you specify.

Using the IN statement is equivalent to using multiple OR statements to join conditions; when applying search conditions to a column, sometimes it is easier to use the IN statement instead of multiple OR statements. When IN is used, at least two values must be specified within the parentheses. Enclose the list of values (excluding NULL, which cannot be used with IN) in parentheses. Separate one value from the next with a comma; a blank between values is optional.

The order of the objects in the list is not important; you receive the same rows in any case. The order of objects in the list does not affect the ordering of the result. To order the result, use an ORDER BY clause.

This query:

```
SELECT DEPTNUMB, DEPTNAME
FROM Q.ORG
WHERE DEPTNUMB IN (20, 38, 42)
```

Produces this report:

```
DEPTNUMB DEPTNAME
-------- --------------
      20  MID ATLANTIC
```

```
      38  SOUTH ATLANTIC
      42  GREAT LAKES
```

In the query above, `IN (20, 38, 42)` is equivalent to `(DEPTNUMB = 20 OR DEPTNUMB = 38 OR DEPTNUMB = 42)`.

Additional examples:

- To select every department in the Eastern and Midwestern divisions:

```
SELECT DEPTNAME, DIVISION, LOCATION
FROM Q.ORG
WHERE DIVISION IN ('EASTERN', 'MIDWEST')
```

- To select every salesperson and clerk in Departments 15, 20, and 38:

```
SELECT ID, NAME, JOB, DEPT
FROM Q.STAFF
WHERE JOB IN ('CLERK', 'SALES')
AND DEPT IN (15, 20, 38)
```

- To select everyone with 1, 2, or 3 years of service, or whose YEARS value is null:

```
SELECT ID, NAME, YEARS
FROM Q.STAFF
WHERE YEARS IN (1, 2, 3) OR YEARS IS NULL
```

INSERT

INSERT is an SQL statement that adds data to a table.

The INSERT statement has the following format:

```
INSERT INTO tablename
VALUES (value1, value2, ...)
```

In this syntax, *tablename* is the name of the table or view into which you want to insert data, and *value1, value2* (and so on), are the values you insert.

The list of data values after VALUES must correspond with the list of columns in the table into which they are inserted. There must be the same number of values as columns, and each value must have a data type that agrees with its column. As shown in the following example, null values can be inserted by specifying NULL.

This statement:

```
INSERT INTO PERS
VALUES (400, 'HARRISON', 20, 'SALES', NULL, 18000.66, 0)
```

Inserts this line into the PERS table:

```
   ID  NAME         DEPT  JOB     YEARS     SALARY       COMM
------ ---------  ------  -----  ------  ---------  ---------
  400  HARRISON       20  SALES       -   18000.66       0.00
```

The PERS table is a copy of the Q.STAFF sample table. If you do not want to use the CREATE TABLE statement, you can also create PERS with these two commands:

```
DISPLAY Q.STAFF
SAVE DATA AS PERS
```

Insert column values in a row

If you want to insert a row without providing values for all of the columns in a row, you can use a list of columns with the INSERT statement.

Specify the values you want to insert into the columns, as in this example:

```
INSERT INTO PERS (ID, NAME, JOB, SALARY)
VALUES (510, 'BUCHANAN', 'CLERK', 11500.75)
```

An easy way to create an INSERT statement is by using the QMF DRAW command as follows:

```
DRAW tablename (TYPE=INSERT
```

Columns for which values are not specified are given no value (NULL). If a column is defined as NOT NULL, you must specify values for it.

Copy rows from one table to another

Rows can be inserted into a table by copying data from another table and using a subquery to identify columns to be inserted instead of using the VALUES clause with INSERT. The information retrieved by the subquery is placed into the table as if multiple INSERT commands had been entered.

The following statement copies the ID, NAME, JOB, and YEARS columns for members of Department 38 from Q.STAFF into PERS:

```
INSERT INTO PERS (ID, NAME, JOB, YEARS)
SELECT ID, NAME, JOB, YEARS
FROM Q.STAFF
WHERE DEPT = 38
```

Values must be specified for all columns that are defined as NOT NULL.

A one-to-one correspondence does not have to exist between columns being selected and columns being inserted; however, there should not be more columns selected than inserted. If fewer columns are selected than are being inserted, the remaining columns are inserted with nulls. Rows cannot be selected for insertion into the same table.

Related reference

CREATE TABLE
The CREATE TABLE statement defines a table. You provide the name of the table and the names and attributes of its columns. You can grant or revoke authorization for other people to use a table you created.

GRANT
The GRANT statement gives users authorization to perform one or more operations on a table.

IS

The IS keyword is used only with NULL and NOT NULL.

Related reference

NULL
If a table is created and only partially filled with data, the fields that contain no data are considered to be null, meaning their values are unknown.

LIKE

LIKE can be used only with character, graphic, and binary data in SQL queries and only with character and graphic data in QBE queries.

To select character, graphic, or binary data when you know only part of a value, use LIKE in a WHERE clause, plus a symbol for the unknown data:

- A percent sign (%) is the symbol for any number of characters (or none).
- An underscore (_) is the symbol for any single character. Use more than one underscore in succession to represent an exact number of unknown characters.

You can also use % and _ together. For example, to select every name with AN or ON as the second and third letters:

```
SELECT ID, NAME
FROM Q.STAFF
WHERE NAME LIKE '_AN%' OR NAME LIKE '_ON%'
```

For character data, the value after LIKE must always be enclosed in single quotation marks. If you are using graphic data, the value after LIKE must be preceded by the single-byte character 'G' enclosed in single quotation marks. The percent sign and the underscore must be double-byte characters.

Selecting a string of characters

You can select rows containing a string of characters that might be part of a word or number you know exists in the data. In the following example, `WHERE ADDRESS LIKE '%NY'` selects any address that contains the characters NY at the end. The percent sign (%) can stand for anything – any number of characters (or none).

This query:

```
SELECT NAME, ADDRESS
FROM Q.APPLICANT
WHERE ADDRESS LIKE '%NY'
```

Produces this report:

```
NAME      ADDRESS
--------- -----------------
JACOBS    POUGHKEEPSIE, NY
REID      ENDICOTT, NY
LEEDS     EAST FISHKILL, NY
```

When using LIKE to search for data with a specific ending, be aware of the data type of the column you are searching. If the column has a fixed width and the data in the column varies in width, add blanks to the character string to match the blanks in the column data.

For example, if the ADDRESS column in the example has a data type of CHAR(17), the width of the column is fixed, with blanks filling the space where the data is not as wide as the column. Searching with an ending character string requires that you anticipate (and search for) the string with every possible number of trailing blanks that could be encountered in the data.

For example, to select everyone whose name begins with W, use a query like the following:

```
SELECT ID, NAME
FROM Q.STAFF
WHERE NAME LIKE 'W%'
```

Ignoring specific characters

You can use an underscore (_) to specify a character string that ignores a given number of characters. Use a specific number of underscores to specify the same number of characters that are to be ignored in the search. For example, the following clause is used to search a column of 8-character part numbers for the character string "G2044" occurring in positions 2 through 6. The first character and the last two characters are ignored. Single quotes are required around an all-digit value in Db2 for z/OS. (Note that there are two underscores after the value in quotation marks.)

```
WHERE PARTNO LIKE '_G2044__'
```

Examples

- To select every name that has an S in any position after the first position, use a query like the following:

```
SELECT ID, NAME
FROM Q.STAFF
WHERE NAME LIKE '_%S%'
```

- To select every name that ends in SON, use a query like the following:

```
SELECT ID, NAME
FROM Q.STAFF
WHERE NAME LIKE '%SON'
```

  This example works because the NAME column has a data type of VARCHAR, which has no blanks following it in the database. If a column has a data type of CHAR, with a fixed width, the query has to anticipate all lengths of names ending in SON, and has to include those combinations in the search value.

MAX and MIN

MAX and MIN operate on columns that contain character, graphic, numeric, date/time, and binary data (except for binary large object, or BLOB, data).

Write the MIN and MAX functions as follows:

```
MAX(expression)
MIN(expression)
```

The parentheses are required. In this syntax, *expression* is most often a column name, but can be:

- An arithmetic expression containing at least one column name
- The DISTINCT keyword, followed by a column name

A column name in a function must not refer to a long-string column or a column derived from a column function. (A column of a view can be derived from a function.) Column functions cannot be nested within other column functions.

The data type of the result of the MAX or MIN function always allows nulls even if the operand of these functions is NOT NULL. Null values are not included in the calculation made by a built-in function.

The following example includes more than one column function in the SELECT statement. For Department 10, it calculates and displays the sum of employee salaries; the minimum, average, and maximum salary; and the number of employees (COUNT) in the department.

```
SELECT SUM(SALARY), MIN(SALARY), AVG(SALARY),
   MAX(SALARY), COUNT(*)
FROM Q.STAFF
WHERE DEPT = 10
```

If you use MAX or MIN with character data, a binary collating sequence is applied when comparing data.

NOT

You can exclude data by using the NOT keyword in the WHERE clause of the query.

Examples

The following example selects all divisions that are not EASTERN or WESTERN.

This query:

```
SELECT DEPTNUMB, LOCATION,
 DIVISION FROM Q.ORG
WHERE NOT
 (DIVISION = 'EASTERN' OR DIVISION = 'WESTERN')
```

Produces this report:

```
DEPTNUMB LOCATION       DIVISION
-------- -------------  ---------
      10  NEW YORK       CORPORATE
      42  CHICAGO        MIDWEST
      51  DALLAS         MIDWEST
```

To make it clear what the NOT condition applies to, use parentheses. If you use NOT with AND or OR and you do not use parentheses, conditions that are preceded by NOT are negated before they are connected by AND or OR. For example, if A, B, and C are conditions, these two phrases are equivalent:

```
NOT A AND B OR C
((NOT A) AND B) OR C
```

With greater than, less than, or equals, NOT must precede the entire condition, as in WHERE NOT YEARS = 10. You can also negate the equal sign with the not symbol (¬).

These statements are correct:

- `WHERE YEARS ¬ > 10`
- `WHERE NOT YEARS = 10`

This statement is incorrect:

```
WHERE YEARS NOT = 10
```

The symbol ¬= is an alternative operator for < > (not equal to). It is an ANSI SQL operator. (If you are using remote data access, the preferred symbol is < >.)

You can use NOT NULL, NOT LIKE, NOT IN, or NOT BETWEEN; only in these cases can NOT follow the first part of the condition. For example:

```
WHERE YEARS IS NOT NULL
```

To select everyone whose salary is not between $17,000 and $21,000, use a query like the following query:

```
SELECT ID, NAME, SALARY
FROM Q.STAFF
WHERE SALARY NOT BETWEEN 17000 AND 21000
```

To select everyone who does not earn a salary less than $18,000 and also earns a commission of less than $500, use a query like the following query:

```
SELECT ID, NAME, SALARY, COMM
FROM Q.STAFF
WHERE NOT (SALARY < 18000 AND COMM < 500)
```

To select only managers in Q.STAFF who are not managers of departments in the Q.ORG table, use a query like the following query:

```
SELECT ID, NAME, DEPT
FROM Q.STAFF
WHERE JOB = 'MGR'
AND ID NOT IN (SELECT MANAGER FROM Q.ORG)
```

NULL

If a table is created and only partially filled with data, the fields that contain no data are considered to be null, meaning their values are unknown.

A null value is not the same as any of these values:

- A numerical value of zero

- A character string of all blanks
- A character string of length zero
- The character string NULL (of length 4)

Each of these values can be entered in a row and column of a table. A null value occurs where no value was entered, or where the value was set to null. It prints and displays as a single hyphen (-) by default.

- This clause is correct: `WHERE columnname IS NULL`
- This clause is incorrect: `WHERE columnname = ' '`

The VALUE scalar function can be used to change how a null value is printed and displayed.

To select rows that have a null value in a column, enter:

```
WHERE columnname IS NULL
```

Examples

To select everyone who does not receive a commission, use a query like the following query:

```
SELECT ID, NAME
FROM Q.STAFF
WHERE COMM IS NULL
```

To select everyone whose commission is zero, use a query like the following query:

```
SELECT ID, NAME
FROM Q.STAFF
WHERE COMM = 0
```

To select everyone who does get a commission, use a query like the following query:

```
SELECT ID, NAME
FROM Q.STAFF
WHERE COMM IS NOT NULL
```

Related reference

SQL scalar functions
Three types of SQL scalar functions are date/time functions, conversion functions, and string functions.

OR

You can select rows based on multiple conditions connected by OR. Conditions connected by OR select every row that satisfies any of the conditions.

This query:

```
SELECT ID, NAME, YEARS, SALARY
FROM Q.STAFF
WHERE YEARS = 10 OR SALARY > 20000
```

Produces this report:

```
  ID  NAME       YEARS     SALARY
----- ---------  -----  ---------
  50  HANES         10   20659.80
 140  FRAYE          6   21150.00
 160  MOLINARE       7   22959.20
 210  LU            10   20010.00
 260  JONES         12   21234.00
 290  QUILL         10   19818.00
 310  GRAHAM        13   21000.00
```

Related reference

AND

You can select rows based on multiple conditions connected by AND or OR.

ORDER BY

As part of the SQL SELECT statement, you can specify the sequence in which selected rows are displayed. You can also eliminate duplicate rows in a selection.

ORDER BY specifies the order in which rows appear in a report. If you use ORDER BY, it must be the last clause in the entire statement. Any columns named after ORDER BY must also be named after SELECT.

The format of the ORDER BY clause is:

```
  ORDER BY columnname ASC|DESC
```

The ASC keyword specifies that you want the data to appear in ascending order; this is the default if no sequence is specified. The DESC keyword specifies that you want the data to appear in descending order.

The following query produces a report with rows in ascending order.

```
SELECT NAME, JOB, YEARS
FROM Q.STAFF
WHERE DEPT = 84
ORDER BY JOB
```

Here is the report:

```
NAME      JOB    YEARS
--------- -----  -----
GAFNEY    CLERK      5
QUILL     MGR       10
DAVIS     SALES      5
EDWARDS   SALES      7
```

Instead of naming a column for *columnname*, you can refer to the column by its position in the SELECT statement, which you express as a number.

Sorting sequence

The sequence for sorting character data in numeric order is:

1. Special characters, including blanks
2. Lowercase letters in alphabetical order
3. Uppercase letters in alphabetical order
4. Numbers
5. Null values

The default sequence for sorting numbers is ascending order. The default sequence for sorting DATE, TIME, TIMESTAMP, and TIMESTAMP WITH TIME ZONE values is chronological. The sequence for sorting DBCS data is determined by the internal value of the data and generally is not meaningful.

Examples:

- To list employees in descending order by salary, use a query like the following:

```
SELECT ID, NAME, SALARY
FROM Q.STAFF
ORDER BY SALARY DESC
```

- To list employees in ascending order by last name, use a query like the following:

```
SELECT ID, NAME, SALARY
FROM Q.STAFF
ORDER BY NAME
```

Ordering by more than one column

To order by more than one column, put the column name or the column number in a list after ORDER BY. You can mix column names and column numbers in the same list. If you want to order by a defined column, you must use its column number.

A column name in an ORDER BY clause, possibly followed by ASC or DESC, is a sort specification. Sort specifications in a list are separated by commas. The first column that follows the ORDER BY clause is put in order first, the second column is ordered within the limits of the first ORDER BY column, and so on.

Examples:

- To order by years within job, use a query like the following:

```
SELECT NAME, JOB, YEARS
FROM Q.STAFF
WHERE DEPT=84
ORDER BY JOB, YEARS DESC
```

  This query produces the following report:

```
NAME      JOB    YEARS
--------- -----  -----
GAFNEY    CLERK      5
QUILL     MGR       10
EDWARDS   SALES      7
DAVIS     SALES      5
```

- To order by job within years, use a query like the following:

```
SELECT NAME, JOB, YEARS
FROM Q.STAFF
WHERE DEPT=84
ORDER BY YEARS DESC, JOB
```

  This query produces the following report:

```
NAME      JOB    YEARS
--------- -----  -----
QUILL     MGR       10
EDWARDS   SALES      7
GAFNEY    CLERK      5
DAVIS     SALES      5
```

- To list employees in descending order by years of service and, within each year, in descending order by salary, use a query like the following:

```
SELECT YEARS, ID, NAME, SALARY
FROM Q.STAFF
ORDER BY YEARS DESC, SALARY DESC
```

- To list employees in ascending order by salary within department, use a query like the following:

```
SELECT DEPT, ID, NAME, SALARY
FROM Q.STAFF
ORDER BY DEPT, SALARY
```

Ordering columns by column number

You cannot use an expression like SALARY+COMM after an ORDER BY statement. To order by a column defined by an expression, use a number that specifies the column's position in the SELECT statement of the query. For example, consider the following query:

```
SELECT ID, NAME, SALARY+COMM
FROM Q.STAFF
WHERE COMM IS NOT NULL
ORDER BY 3
```

In the query above, SALARY+COMM is column 3 in the SELECT statement, so ORDER BY 3 specifies to order by that column.

You can use more than one column number in a list after ORDER BY, and you can use column names and column numbers in the same list. For example, to list employees in descending order by salary within a department, use a query like the following:

```
SELECT DEPT, ID, NAME, SALARY
FROM Q.STAFF
ORDER BY 1, 4 DESC
```

REVOKE

The REVOKE statement removes authorization granted by a GRANT statement.

The syntax of the REVOKE statement is:

```
REVOKE operation-list ON tablename FROM user-list
```

operation-list
Lists one or more of the following, separated by commas: ALTER, DELETE, INSERT, SELECT, UPDATE. Use ALL to revoke all privileges at once.

tablename
Names the table or view for which the authorization is revoked.

user-list
Lists each user ID with commas between them. PUBLIC can be specified in place of *user-list*. The use of PUBLIC does not revoke a privilege from any user ID for which authorization was specifically granted; such a privilege must also be specifically revoked.

REVOKE and GRANT are similar, with the following exceptions:

- With REVOKE, you cannot specify a column list after UPDATE. UPDATE revokes the authorization to update any column. To revoke authorization to update specific columns and allow it to remain for others:
  1. Revoke the authorization to update any column.
  2. Grant the authorization to update a specific list of columns.
- If you grant a privilege to JONES, who then grants it to JACOBS, and you revoke the privilege from JONES, that privilege is also revoked from JACOBS.

The following statement revokes from JACOBS the authorization to write SELECT queries using the PERS table:

```
REVOKE SELECT ON PERS FROM JACOBS
```

The following statement revokes from user HSAM4419 the privilege to update any column in the PERS table:

```
REVOKE UPDATE ON PERS FROM HSAM4419
```

SELECT

With the SELECT statement, you can specify the name of each column you want to retrieve from a table. You can name one or more columns from a table or view, or you can select all the columns. Each SELECT statement can select information from several tables.

You can use the DISTINCT keyword to eliminate duplicate information if you are selecting data from multiple tables.

QMF displays selected data according to the default edit code for the data type.

No more than one SELECT statement can be used in a query that includes other SQL statements.

Selecting every column from a table

To retrieve all the columns from a table, use an asterisk (*) instead of naming the columns. The format of a SELECT statement used for this selection is:

```
SELECT * FROM tablename
```

In this statement, *tablename* is the name of the table or view you are searching. For example, this statement returns all the columns in Q.ORG:

```
SELECT * FROM Q.ORG
```

This query returns all the columns but only displays rows where the department number is 10:

```
SELECT *
FROM Q.STAFF
WHERE DEPT = 10
```

Selecting columns from a table

To select columns from a table, enter SELECT, followed by the exact names of the columns in the order (left to right) in which you want them in your report. Separate column names by a comma.

The following statement produces a report with the department names on the left and the department numbers on the right:

```
SELECT DEPTNAME, DEPTNUMB
FROM Q.ORG
```

You can change the order of columns in the report by changing the form. The default order of the columns on the form is the same order in which they are named in the query.

You can select a column more than once; this allows you to use multiple aggregation functions on the form.

You can select up to 750 column names (or expressions) in Db2 for z/OS and up to 255 when connected to DB2 for VSE and VM databases.

You can use a column name in a WHERE clause without using the column name in the SELECT clause.

Examples:

- To select only the ID and NAME columns from the Q.STAFF table, use a query like the following:

  ```
  SELECT ID, NAME
  FROM Q.STAFF
  ```

- To select the NAME and ID columns from the Q.STAFF table, and list NAME first, use a query like the following:

  ```
  SELECT NAME, ID
  FROM Q.STAFF
  ```

Add descriptive columns

You can add a column of descriptive information to your report by putting a quoted constant in the column list of your SELECT statement. The length of a constant is determined by the database. Constants can contain alphabetic characters, numeric characters, or a combination of the two. The following example lists the names and addresses of people in the Q.APPLICANT table with 14 years of education, and identifies each as an applicant.

This query:

```
SELECT NAME, ADDRESS, 'APPLICANT'
FROM Q.APPLICANT
```

```
WHERE EDLEVEL = 14
ORDER BY NAME
```

Produces this report:

```
NAME       ADDRESS            COL1
---------  -----------------  ---------
CASALS     PALO ALTO,CA       APPLICANT
REID       ENDICOTT,NY        APPLICANT
RICHOWSKI  TUCSON,AZ          APPLICANT
```

The report includes three columns: one containing names, one containing addresses, and a newly created column containing the word APPLICANT for each row selected. The database manager adds a column name to the newly created column. This name varies, depending on the database manager used at your site. You can change this column name using the form panels.

Using subqueries

Subqueries select data from a table. The data is then used to test a condition in the WHERE clause of the main query. For example, this query has a subquery (beginning with the SELECT DEPTNUMB statement) that produces a list of employees who work in the Eastern division:

```
   SELECT NAME, ID
   FROM Q.STAFF
   WHERE DEPT = SOME
     (SELECT DEPTNUMB
     FROM Q.ORG
     WHERE DIVISION='EASTERN')
```

First, the subquery finds the department numbers in the Eastern division. Then, the main query finds employees who work in any of these departments.

When there are several subqueries, the last one is executed first; the first one is executed last.

Examples

Each of the following examples includes one subquery, which is highlighted.

```
SELECT DEPT, NAME, SALARY
FROM Q.STAFF CORRVAR
WHERE SALARY =
  (SELECT MAX(SALARY)
   FROM Q.STAFF
   WHERE DEPT = CORRVAR.DEPT)
```

```
SELECT ID, NAME
FROM Q.STAFF
WHERE DEPT IN
   (SELECT DISTINCT DEPTNUMB
    FROM Q.ORG
    WHERE DIVISION = 'MIDWEST')
ORDER BY ID
```

```
   SELECT DEPT, AVG(SALARY)
   FROM Q.STAFF
   GROUP BY DEPT
   HAVING AVG(SALARY) >
      (SELECT AVG(SALARY) FROM Q.STAFF)
```

Accessing QMF Data Service (QDS) data

QMF users can use the QMF Data Service feature to access non Db2 data such as VSAM, IMS, sequential files, SMF data, SYSLOG data and more.

To access QDS data, the QMF for TSO and CICS global variable DSQEC_DS_SUPPORT must be set to a value of '1'; for more information about setting the DSQEC_DS_SUPPORT global variable, see “Global variables that control how commands and procedures are executed” on page 311.

QMF Data Service data sources are accessed through three part table names in the SQL queries, Prompted queries, or Query-by-Example queries. QMF Data Service might join one or more sources that exist at the server. SQL accepted by QMF Data Service is a subset of SQL accepted by Db2 for z/OS. Refer to the QMF Data Service SQL guide for accepted SQL syntax.

Related concepts

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.

Related reference

DISTINCT
Use the DISTINCT keyword before the column names in an SQL statement to prevent duplicate rows from being selected.

Related information

Search for the SQL reference information for your database manager for details about the limits for tables, views, and columns in a SELECT statement.

SET *Db2 global variable*

You can use the SET statement in a QMF SQL query to set Db2 for z/OS or Db2 for Linux, UNIX, and Windows global variables.

In most cases, the QMF global variable DSQEC_KEEP_THREAD must be set to 1 before you can use the SET *Db2 global variable* statement. However, you can use the SET *Db2 global variable* statement without setting DSQEC_KEEP_THREAD to 1 if any of the following conditions are true:

- The SET *Db2 global variable* statement is included in a procedure that is run in batch mode. The Db2 global variable is reset to its default value after the procedure completes.
- The QMF CONNECT command was issued to connect to a remote database and the SET *Db2 global variable* statement is run on the remote database.
- The SET *Db2 global variable* statement is included in a multistatement query and the QMF global variable DSQEC_RUN_MQ is set to 1. The Db2 global variable is reset to its default value after the query completes.

SET *special register*

You can use the SET statement in a QMF SQL query to set the special registers listed in this topic.

Db2 for z/OS special registers you can set in a QMF SQL query:

- CURRENT ACCELERATOR
- CURRENT APPLICATION COMPATIBILITY
- CURRENT DEGREE

  The value of the CURRENT DEGREE register persists for the entire QMF session, regardless of the remote servers to which you connect during the session.
- CURRENT GET_ACCEL_ARCHIVE
- CURRENT LOCALE LC_CTYPE
- CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
- CURRENT OPTIMIZATION HINT

  QMF allows you to set this register as long as the Db2 for z/OS subsystem allows it.
- CURRENT PATH

  You can set this register by issuing a SET PATH or SET FUNCTION PATH statement.
- CURRENT PRECISION

- CURRENT QUERY ACCELERATION
- CURRENT REFRESH AGE

  A value of ANY for this register is not supported in QMF.
- CURRENT SCHEMA

  The value of CURRENT SCHEMA is the same as the value of CURRENT SQLID unless a SET SCHEMA statement has been issued specifying a different value.

  The following QMF commands, as well as all QMF query interfaces, use the default schema ID to provide access to unqualified Db2 tables and views:

  – DISPLAY TABLE
  – DRAW
  – EDIT TABLE
  – ERASE TABLE
  – EXPORT TABLE
  – IMPORT TABLE
  – PRINT TABLE
  – SAVE DATA

  If queries or any of the above commands reference an unqualified table or view name, QMF sends the unqualified name to Db2 for resolution. Db2 uses the value in the CURRENT SCHEMA register to qualify the table or view name.

  For example, suppose that you issue the following statement to set the CURRENT SCHEMA register to a value of SALES:

  ```
  SET SCHEMA = SALES
  ```

  After this statement is issued, a command such as `DISPLAY EMPLOYEES` causes QMF to send the unqualified name to Db2 for resolution, and the table SALES.EMPLOYEES is displayed.
- CURRENT SQLID

  The value of this register is reflected in the DSQAO_CONNECT_ID global variable.

  The value of the CURRENT SQLID register persists for the entire QMF session, regardless of the remote servers to which you connect during the session.
- CURRENT TEMPORAL BUSINESS_TIME
- CURRENT TEMPORAL SYSTEM_TIME

Db2 for Linux, UNIX, and Windows special registers that you can set in a QMF SQL query

- CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
- CURRENT PATH
- CURRENT REFRESH AGE
- CURRENT SCHEMA

  See above for a description of how QMF handles default schema IDs as set in this register.

  CURRENT SQLID can be specified in place of CURRENT SCHEMA.

  The value of the CURRENT SQLID register persists for the entire QMF session, regardless of the remote servers to which you connect during the session.
- CURRENT TEMPORAL BUSINESS_TIME
- CURRENT TEMPORAL SYSTEM_TIME

DB2 for iSeries special registers that you can set in a QMF SQL query

- CURRENT PATH

You cannot use the SET statement in QMF SQL queries to set registers in DB2 for VSE and VM databases.

Tip: Although you cannot set the CURRENT EXPLAIN MODE special register through the SET statement, you can use the DSQEC_EXPL_MODE global variable to set the value of CURRENT EXPLAIN MODE. For more information, see "Global variables that control how commands and procedures are executed" on page 311.

You can display the value currently assigned to any special register by using a SELECT statement in a QMF SQL query. For example, to display the value of the CURRENT PRECISION special register in Db2 for z/OS, run the following SQL query:

```
SELECT CURRENT PRECISION FROM SYSIBM.SYSDUMMY1
```

Related reference

Global variables for state information not related to the profile
DSQAO global variables contain status information or settings of parameters or flags. None of these global variables can be modified by the SET GLOBAL command.

Related information

Search for information about how to set each special register in your database information.

SOME

Use the SOME keyword with comparison operators to permit a query to return a set of values rather than a single value.

You can use SOME with the following comparison operators:

```
     =   ¬=   >   >=   <   <=   < >
```

The symbol ¬= is an alternative symbol for < > (not equal to). It is an ANSI SQL operator. (If you are using remote data access, the preferred symbol is < >.)

ALL, ANY, and IN can also be used to return a set of values:

- When ALL is used, all values in the set returned satisfy the condition.
- When ANY or SOME is used, at least one value in the set returned satisfies the condition.
- IN can be used in a subquery in place of either of the following:

```
= SOME
= ANY
```

The following query produces a list of employees who work in the Eastern division. First, the subquery finds the department numbers in the Eastern division. Then, the main query finds the employees who work in these departments.

```
SELECT NAME, ID
FROM Q.STAFF
WHERE DEPT = SOME
   (SELECT DEPTNUMB FROM Q.ORG WHERE DIVISION='EASTERN')
```

The keyword SOME is used in this query because there are multiple departments in the Eastern division. If ALL is used instead of SOME (or ANY), the result is an empty set. No employee works in all the departments of the Eastern division.

SUM

SUM is valid only on columns that contain numeric values.

The data type of the result of the sum always allows nulls, even if the operand of these functions is NOT NULL. Null values are not included in the calculation made by a built-in function.

The following example includes more than one column function in the SELECT statement. For Department 10, it calculates and displays the sum of employee salaries; the minimum, average, and maximum salary; and the number of employees (COUNT) in the department.

This query:

```
SELECT SUM(SALARY), MIN(SALARY), AVG(SALARY),
   MAX(SALARY), COUNT(*)
FROM Q.STAFF
WHERE DEPT = 10
```

Produces this report:

```
              COL1        COL2                COL3        COL4         COL5
------------------  ----------  ------------------  ----------  -----------
          83463.45    19260.25    20865.8625000000    22959.20            4
```

You can write the SUM column function like this:

```
SUM(expression)
```

The parentheses are required. In the above syntax, *expression* is most often a column name, but can also be:

- An arithmetic expression containing at least one column name.
- DISTINCT followed by a column name.

A column name in a function must not refer to a long-string column or a column derived from a column function (a column of a view can be derived from a function). Column functions cannot be nested within other column functions.

UNION

UNION merges the rows of two or more tables into one report. To make sense, these rows should relate to one another, have the same width, and have the same data type.

Using UNION, you can merge values from two or more tables into the same columns (but different rows) of the same report. You can use UNION more than once in a query.

Examples in this topic that use UNION ALL require enhanced UNION support.

The following example selects the name and employee columns from Q.STAFF and the name and applicant columns from Q.APPLICANT.

```
SELECT NAME,'EMPLOYEE'
FROM Q.STAFF
WHERE YEARS < 3
UNION
SELECT NAME,'APPLICANT'
FROM Q.APPLICANT
WHERE EDLEVEL > 14
```

The query produces this report:

```
NAME       COL1
---------  ---------
BURKE      EMPLOYEE
GASPARD    APPLICANT
JACOBS     APPLICANT
```

The portion of the query that selects from Q.STAFF also creates a column in the report with the constant EMPLOYEE in it. The portion of the query that selects from Q.APPLICANT does the same with the constant APPLICANT. A default column name is assigned to that column, but can easily be changed on the form panels.

In any query, the lengths of the columns are matched. In the previous query, EMPLOYEE is padded with a blank to match the length of APPLICANT.

The following example selects from Q.STAFF and Q.INTERVIEW all the managers and the people they interviewed.

```
SELECT NAME, ' '
FROM Q.STAFF, Q.INTERVIEW
WHERE MANAGER = ID
UNION
SELECT NAME, 'NO INTERVIEWS'
FROM Q.STAFF
WHERE JOB = 'MGR'
  AND ID NOT IN (SELECT MANAGER FROM Q.INTERVIEW)
```

The query produces this report:

```
NAME       COL1
---------  -------------
DANIELS    NO INTERVIEWS
FRAYE
HANES
JONES      NO INTERVIEWS
LEA
LU         NO INTERVIEWS
MARENGHI   NO INTERVIEWS
MOLINARE
PLOTZ
QUILL
SANDERS
```

Retaining duplicate rows with UNION

UNION implies that only DISTINCT rows are selected from the columns named in both SELECT statements.

If you want to keep duplicates in the result of a UNION operation, specify the optional keyword ALL after UNION. When UNION ALL is specified, duplicate rows are not eliminated from the result.

The following example selects all salespeople in Q.STAFF who have been employed for more than five years, or who earn a commission greater than $850. The salespeople who meet both conditions appear twice in the resulting report.

This query:

```
SELECT * FROM Q.STAFF
WHERE JOB = 'SALES' AND YEARS > 5
UNION ALL
SELECT * FROM Q.STAFF
WHERE JOB = 'SALES' AND COMM > 850
ORDER BY 2
```

Produces this report:

```
   ID  NAME         DEPT  JOB     YEARS      SALARY        COMM
------ ---------  ------  -----  ------  ----------  ----------
  340  EDWARDS        84  SALES       7    17844.00     1285.00
  340  EDWARDS        84  SALES       7    17844.00     1285.00
  310  GRAHAM         66  SALES      13    21000.00      200.30
   90  KOONITZ        42  SALES       6    18001.75     1386.70
   90  KOONITZ        42  SALES       6    18001.75     1386.70
   40  O'BRIEN        38  SALES       6    18006.00      846.55
   20  PERNAL         20  SALES       8    18171.25      612.45
   70  ROTHMAN        15  SALES       7    16502.83     1152.00
   70  ROTHMAN        15  SALES       7    16502.83     1152.00
  220  SMITH          51  SALES       7    17654.50      992.80
```

```
  220  SMITH          51  SALES       7    17654.50      992.80
  150  WILLIAMS       51  SALES       6    19456.50      637.65
  280  WILSON         66  SALES       9    18674.50      811.50
```

If UNION rather than UNION ALL is specified, determining which salespeople satisfied both conditions requires closer inspection, as shown in the report in the following figure:

```
   ID  NAME         DEPT  JOB     YEARS      SALARY        COMM
------ ---------  ------  -----  ------  ----------  ----------
  340  EDWARDS        84  SALES       7    17844.00     1285.00
  310  GRAHAM         66  SALES      13    21000.00      200.30
   90  KOONITZ        42  SALES       6    18001.75     1386.70
   40  O'BRIEN        38  SALES       6    18006.00      846.55
   20  PERNAL         20  SALES       8    18171.25      612.45
   70  ROTHMAN        15  SALES       7    16502.83     1152.00
  220  SMITH          51  SALES       7    17654.50      992.80
  150  WILLIAMS       51  SALES       6    19456.50      637.65
  280  WILSON         66  SALES       9    18674.50      811.50
```

Figure 10. An example of the results of the UNION statement

The order of evaluation of each subquery has no effect on the result of the operation. However, when you use UNION ALL or UNION to combine two SELECT queries, the result of the operation depends on the order of evaluation. Parentheses are resolved first, starting with the innermost set. Then each clause is resolved from left to right.

For example, the following queries yield different results:

- In this example, all rows of TABLE1 are merged with all rows of TABLE2 to form an intermediate table, which is merged with TABLE3 with the elimination of duplicates.

  ```
  (TABLE1 UNION ALL TABLE2) UNION TABLE3
  ```

- In this example, all rows of TABLE2 are merged with TABLE3 with the elimination of duplicates, to form an intermediate table that is merged with all rows of TABLE1.

  ```
  TABLE1 UNION ALL (TABLE2 UNION TABLE3)
  ```

Rules for using UNION

- You can put UNION between two SELECT statements only if the two statements select the same number of columns and the corresponding columns are compatible data types (for example, numeric to numeric).
- Corresponding columns in select statements merged by UNION do not need to have the same name. Because the names of the interleaved columns are likely to be different, do not use a column name after an ORDER BY. Instead, always use a column number, such as ORDER BY 1.
- The lengths and data types of the columns named in the SELECT statements only need to be comparable. The columns must both have numeric, character, graphic, date, time, or timestamp values. They cannot be a combination of these data types.

  For example:

  ```
  SELECT ID
  ⋮
  UNION
  SELECT DEPT
  ⋮
  ```

  If ID is CHAR(6) and DEPT is CHAR(3), the column in the result table is CHAR(6). The values in the resulting table that are derived from DEPT are padded on the right with blanks.

When to use UNION versus when to join tables

When to use UNION to merge tables and when to join tables depends on what kind of results you want in your report:

- UNION interleaves rows from two queries into one report.
- Joining tables does not interleave the rows, but joins each row from one table horizontally to each row from another table. When joining, it is essential that you use a condition (a WHERE clause) to limit the number of combinations so that every row is not joined to every other row.

The following query does not produce a report that is as readable or meaningful as the UNION query. Because no common column was used in the WHERE condition in this query to join the two tables, the report contains duplicates.

This query:

```
SELECT S.NAME, 'EMPLOYEE', A.NAME, 'APPLICANT'
FROM Q.STAFF S, Q.APPLICANT A
WHERE YEARS < 3 AND EDLEVEL > 14
```

Produces this report:

```
NAME       COL1      NAME2      COL3
---------  --------  ---------  ---------
BURKE      EMPLOYEE  JACOBS     APPLICANT
BURKE      EMPLOYEE  GASPARD    APPLICANT
```

You can also use UNION between two SELECT statements that refer to the same table. For example, to list all employees by number within department, and identify those with ten years of service, use a query like the following:

```
SELECT DEPT, ID, NAME, YEARS, 'TEN YEARS'
FROM Q.STAFF
WHERE YEARS = 10
 UNION
SELECT DEPT, ID, NAME, YEARS, ' '
FROM Q.STAFF
WHERE NOT YEARS = 10
ORDER BY 1, 2
```

Related reference

QMF functions that require specific support
Support for these functions varies with the database or environment.

UPDATE

The UPDATE statement changes the values of specified existing columns in rows of a table. You can update a table only if you created the table or are authorized to update the table.

The UPDATE statement consists of the following parts:

- UPDATE specifies the table to update.
- SET specifies the column to update and the new value to place in the table.
- WHERE specifies which row to update.

An easy way to create an UPDATE statement is by specifying a query type of UPDATE when you issue the DRAW command.

A single UPDATE statement can update one row in a table, more than one row, as shown in the first 2 examples. The statement can also update all rows for a column when the WHERE clause is omitted.

Examples

The following example updates the PERS table for employee 250. It changes JOB to SALES and increases SALARY by 15%.

```
UPDATE PERS
SET JOB='SALES', SALARY=SALARY * 1.15
WHERE ID = 250
```

To give every clerk in PERS a $300 increase, use an UPDATE statement like the following statement:

```
UPDATE PERS
SET SALARY = SALARY+300
WHERE JOB = 'CLERK'
```

To increase everyone's years of service by 1 in the PERS table, use an UPDATE statement like the following statement:

```
UPDATE PERS
SET YEARS = YEARS + 1
```

Related reference

DRAW
The DRAW command helps you compose a basic SQL query or QBE query.

GRANT
The GRANT statement gives users authorization to perform one or more operations on a table.

WHERE

Use a WHERE clause in your SELECT statement to specify a condition (one or more selection criteria) that identifies the row or rows you want to retrieve, update, or delete. Only the rows that satisfy the search condition are affected.

Both WHERE and HAVING eliminate data that you do not want in your report:

- The WHERE condition is used with column selection. It determines whether an individual row is included.

  Use WHERE to eliminate unwanted rows.
- The HAVING condition is used with built-in functions. It determines whether a whole group is included.

  HAVING is always followed by a column function (such as SUM, AVG, MAX, MIN, or COUNT). HAVING can also be followed by a subquery that finds a grouped value to complete the HAVING condition.

  Use HAVING to eliminate unwanted grouped data.

You can compare column values by using any of the operators that are shown in the following table. The condition that is defined in the first column is specified by entering the corresponding words or symbols in the second column.

Table 11. Comparison operators

Comparison	**Comparison operator to use**
Equal to	=
Not equal to	<> or ¬=
Greater than	>
Greater than or equal to	>=
Not greater than (Db2 for z/OS only)	¬>

Table 11. Comparison operators (continued)	
Comparison	**Comparison operator to use**
Less than	<
Less than or equal to	<=
Not less than (Db2 for z/OS only)	¬<
Multiple conditions	AND OR
Values within a range	BETWEEN x AND y
Values matching any in a list	IN (x, y, z)
Selects a string of characters	% (example: LIKE '%abc%')
Ignores certain characters	_ (example: LIKE '_a_')
Negative conditions	NOT

A not sign (¬) can cause parsing errors in statements that are passed from one database management system to another. To avoid this possible problem in statements to be executed at a remote location, substitute an equivalent for any operation in which the not sign appears. For example, substitute <> for ¬=, <= for ¬>, and >= for ¬<.

Values to be compared with columns of character data must be enclosed in single quotation marks (as in WHERE NAME = 'JONES'). Numeric data is not enclosed in quotes.

If you are using graphic data, the value after WHERE must be preceded by the single-byte character 'G' and be enclosed in single quotation marks. The percent sign and the underscore must be double-byte characters.

Examples

Here are some examples of how to use a WHERE clause in a query:

In the following example, the search condition specifies that the value in the DEPT column must be 20. This query:

```
SELECT DEPT, NAME, JOB
FROM Q.STAFF
WHERE DEPT = 20
```

Produces this report:

```
DEPT NAME        JOB
---- ----------  -----
 20  SANDERS     MGR
 20  PERNAL      SALES
 20  JAMES       CLERK
 20  SNEIDER     CLERK
```

To list the least, greatest, and average salary in each department, excluding managers, for departments with an average salary greater than $12,000, use the following query. This query:

```
SELECT DEPT, MIN(SALARY),
 MAX(SALARY), AVG(SALARY)
FROM Q.STAFF
WHERE JOB <> 'MGR'
GROUP BY DEPT
HAVING AVG(SALARY) > 12000
```

Produces this report:

```
  DEPT        COL1        COL2                COL3
------  ----------  ----------  ------------------
    15    12258.50    16502.83    13756.5100000000
    20    13504.60    18171.25    15309.5333333333
    38    12009.75    18006.00    14944.7000000000
    42    10505.90    18001.75    13338.7500000000
    51    13369.80    19456.50    16235.2000000000
    66    10988.00    21000.00    16880.1750000000
    84    13030.50    17844.00    15443.0000000000
```

You can write a WHERE search condition that uses any of the comparison operators. For example, to select only employees who made commissions of $1,000 or more, use a query like the following one. This query:

```
SELECT ID, COMM
FROM Q.STAFF
WHERE COMM >= 1000
```

Produces this report:

```
  ID     COMM
----- -------
  70  1152.00
  90  1386.70
 340  1285.00
```

To select everyone with 10 years of service or more:

```
SELECT ID, NAME, YEARS
FROM Q.STAFF
WHERE YEARS >= 10
```

To select everyone with more than 10 years of service:

```
SELECT ID, NAME, YEARS
FROM Q.STAFF
WHERE YEARS > 10
```

To select every manager:

```
SELECT ID, NAME, YEARS
FROM Q.STAFF
WHERE JOB = 'MGR'
```

To select everyone whose name occurs later in the alphabet than SMITH:

```
SELECT NAME, ID
FROM Q.STAFF
WHERE NAME > 'SMITH'
```

To select every employee name in Q.STAFF that is not in Department 10:

```
SELECT NAME, ID
FROM Q.STAFF
WHERE DEPT < > 10
```

Calculated results

You can use calculated values as part of a search condition. You can also display them for selected rows just as you display column values.

You can use an arithmetic expression in the SELECT clause or in the WHERE clause of the query:

- When the expression is part of the SELECT clause, the new calculated column resulting from the expression appears in the report.

- When the expression is part of the WHERE clause, it is part of the search condition; no new column appears in the report.

The following two queries illustrate the use of an arithmetic expression in a SELECT clause.

This query:

```
SELECT DEPT, NAME, SALARY
FROM Q.STAFF
WHERE DEPT = 38
```

Produces this report:

```
DEPT NAME            SALARY
---- ----------  ----------
 38  MARENGHI      17506.75
 38  O'BRIEN       18006.00
 38  QUIGLEY       16808.30
 38  NAUGHTON      12954.75
 38  ABRAHAMS      12009.75
```

This query:

```
SELECT DEPT, NAME, SALARY/12
FROM Q.STAFF
WHERE DEPT = 38
```

Produces this report:

```
  DEPT  NAME                    COL1
------  ---------  ------------------
    38  MARENGHI      1458.8958333333
    38  O'BRIEN       1500.5000000000
    38  QUIGLEY       1400.6916666666
    38  NAUGHTON      1079.5625000000
    38  ABRAHAMS      1000.8125000000
```

You can use the following arithmetic operators in calculations:

+ Add
- Subtract
* Multiply
/ Divide

Within expressions, you can use column names (as in RATE*HOURS), columns and constants (as in RATE*1.07), and built-in functions (as in AVG(SALARY)/2). An expression can consist of numeric constants (such as 3*7) or character constants (such as SALARY + COMM).

When a table is created, each column in it is defined to hold a certain type of data. Arithmetic operations can be performed only on numeric data types, and the results of an operation can depend on the data types of the operands.

Examples:

- To select the name and total earnings (salary plus commission) of every employee who earns more than $20,000 a year, use a query like the following:

  ```
  SELECT NAME, SALARY + COMM
  FROM Q.STAFF
  WHERE SALARY + COMM > 20000
  ```

  The above query does not list anyone whose salary alone is greater than $20,000 when the amount of the commission is null, because the result of operating on an unknown is unknown.

- To list anyone whose commission is 5% or more of their total earnings, use a query like the following:

  ```
  SELECT NAME, SALARY, COMM
  FROM Q.STAFF
  WHERE COMM >= 0.05 * (SALARY + COMM)
  ```

SQL scalar functions

Three types of SQL scalar functions are date/time functions, conversion functions, and string functions.

Date/time functions

Date/time functions calculate or change the following items:

- DATE, TIME, TIMESTAMP, and TIMESTAMP_TZ change the data type of their argument to the data type specified by the function.
- CHAR changes the data type of its argument (a DATE or TIME value) to the CHAR data type.
- DAYS calculates the number of days between one date and another.
- YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, and MICROSECOND select parts of DATE, TIME, TIMESTAMP, or TIMESTAMP WITH TIME ZONE values.

Each date/time function is followed by an argument that is enclosed in parentheses. The following example lists, by number, each project that is scheduled to begin in 1998 by applying the YEAR function to the STARTD column of the Q.PROJECT table.

This query:

```
SELECT PROJNO, STARTD, ENDD, TIMESTAMP
FROM Q.PROJECT
WHERE YEAR(STARTD) = 1998
```

Produces this report:

```
PROJNO  STARTD      ENDD        TIMESTAMP
------  ----------  ----------  --------------------------
1409    1998-06-15  1999-12-31  1996-03-13-09.12.57.149572
1410    1998-09-29  2000-03-31  1996-03-13-12.18.23.402917
```

Date/time functions (see the following table) can be used wherever an expression can be used. The first or only argument of each of these functions is an expression that passes the value to be manipulated.

Table 12. Date/time functions

Function	Argument	Result
DATE	Date, timestamp, timestamp with time zone, or string representation of a date	Date
TIME	Time, timestamp, timestamp with time zone, or string representation of a time	Time
TIMESTAMP	Timestamp, timestamp with time zone, string representation of a timestamp or timestamp with time zone, or a date (or string representation of a date) and a time (or string representation of a time)	Timestamp
TIMESTAMP_TZ	Timestamp or timestamp with time zone, string representation of a timestamp or timestamp with time zone, or a date (or string representation of a date) and a time (or string representation of a time)	Timestamp
DAY, MONTH, or YEAR	Date, timestamp, timestamp with time zone, or a date duration	Day, month, or year part

Table 12. Date/time functions (continued)

Function	Argument	Result
HOUR, MINUTE, or SECOND	Time, timestamp, timestamp with time zone, or a time duration	Hour, minute, or second part
MICROSECOND	Timestamp or timestamp with time zone	Fractional seconds
DAYS	Date, timestamp, timestamp with time zone, or a string representation of a date	Days since January 1, 0001
CHAR	Date or time and the specified date/time output format	String representation in specified date/time format. By default, or if the DSQSFISO program parameter is set to YES, and the format is not specified, ISO format is returned. If the DSQSFISO program parameter is set to NO, the result is returned in the format that is specified in the DATE FORMAT and TIME FORMAT fields on Db2 installation panel DSNTIP4. This behavior may be changed within a QMF session by setting the DSQEC_DSQSFISO global variable.

Other conversion functions

Scalar functions (see the following table) allow the conversion of a value from one data type to another.

Table 13. Conversion functions

Function and syntax	Argument	Result
BIGINT(V)	V = A number expression or a string expression	A big-integer representation of V or a string representation of V
BINARY(V, length)	V = A string expression *length* = an integer that specifies the length of the resulting string	A fixed-length binary string
DECFLOAT(V,P)	V = A number expression or a string expression P = Digits of precision for the result (16 or 34; the default is 34)	A decimal floating-point representation of a number or a string representation of a number

Table 13. Conversion functions (continued)

Function and syntax	Argument	Result
DECIMAL(V,P,S)	V = A number P = Precision of the result, in the range 1 to 31 S = Scale of the result	Decimal representation of V
DIGITS(argument)	A binary integer or decimal number	A character string that represents the digits of the argument
FLOAT(argument)	A number	Floating-point number that represents the argument
HEX(argument)	Any data type other than a long character or long graphic string	A character string that represents actual hex digits of the argument
INTEGER(argument)	A number within the range of binary integers	Fullword representation of the argument
VARBINARY(V, length)	V = A string expression *length* = integer that specifies the length of the resulting string	A varying-length binary string
VARGRAPHIC(argument)	Short character string	Graphic string that is the DBCS representation of the argument

The following query produces results for some of the functions explained in the previous table.

This query:

```
SELECT SALARY,          --SALARY
DECIMAL(SALARY,9,3),    --COL1
DIGITS(SALARY),         --COL2
FLOAT(SALARY),          --COL3
HEX(NAME),              --COL4
FLOAT(YEARS)            --COL5
FROM Q.STAFF
WHERE DEPT = 10
```

Produces this report:

```
    SALARY          COL1  COL2          COL3  COL4                COL5
---------  ------------  -------  ----------  ------------------  ----------
  22959.20     22959.200  2295920   2.296E+04  D4D6D3C9D5C1D9C5    7.000E+00
  20010.00     20010.000  2001000   2.001E+04  D3E4                1.000E+01
  19260.25     19260.250  1926025   1.926E+04  C4C1D5C9C5D3E2      5.000E+00
  21234.00     21234.000  2123400   2.123E+04  D1D6D5C5E2          1.200E+01
```

String functions

The functions that are shown in the following table enable the manipulation and retrieval of string segments.

Table 14. String functions

Function and syntax	Argument	Result
LENGTH(argument)	Any data type	Integer represents the length of the argument

Table 14. String functions (continued)

Function and syntax	Argument	Result
SUBSTR(S,N,L)	S: Character or graphic string to be evaluated. N: Binary integer; represents the starting position of the substring in S. L: Binary integer; represents the length of the substring.	Substring of S
VALUE(*arg1*, *arg2*...)	Arguments must have compatible data types.	The first non-null value of the provided arguments.

The LENGTH function returns the actual variable length of the data if the data type is VARCHAR; it returns the fixed length if the data type is CHAR.

The VALUE function takes two or more arguments and returns the first argument in the series that resolves to a non-null value. For example, the following statement retrieves the commission of every employee by querying the COMM column of Q.STAFF. If the COMM column contains a null value for any row, the result for that row is "0," the second argument of the VALUE function.

```
SELECT VALUE(COMM,0) FROM Q.STAFF
```

The first or only argument of each of these functions is an expression that passes the value to be manipulated or retrieved. For LENGTH, the value of this expression can be any data type. For SUBSTR, the value must be a character string or a graphic string. For VALUE, two or more values must be specified, and their data types must be compatible. For example, you cannot specify an INTEGER string for the first argument and a CHARACTER string for the second argument in a VALUE function.

For example, this query finds the first initial and last name of an applicant with the temporary ID number 400:

```
SELECT SUBSTR(FIRSTNAME,1,1)||LASTNAME
FROM Q.INTERVIEW
WHERE TEMPID = 400
```

Related information

To find information about available conversion functions and about the compatibility of data types, see the SQL reference information for the database you are using.

Concatenation

The concatenation operator (CONCAT) joins two values of an expression into a single string. The alternate operator for CONCAT is ||. Because vertical bars can cause parsing errors in statements passed from one database management system to another, CONCAT is the preferred operator for statements executed at remote locations.

To use the concatenation operator, include "CONCAT" between the strings that you want to combine. For example, the following query lists all last names in Q.INTERVIEW that begin with letters the occur later than M in the alphabet, and combines those last names with their respective first names.

```
SELECT LASTNAME CONCAT ', ' CONCAT FIRSTNAME
FROM Q.INTERVIEW
WHERE LASTNAME > 'M'
```

The following rules apply to the CONCAT operator:

- The operands of a concatenation operator must both be character strings or both be graphic strings.
- The length of the result is the sum of the lengths of the operands.

- The data type of the result is:
  - VARCHAR when one or more operands is VARCHAR
  - CHAR when both operands are CHAR
  - VARGRAPHIC when one or more operands is VARGRAPHIC
  - GRAPHIC when both operands are GRAPHIC
- Concatenation cannot be specified in a LIKE clause or in the SET clause of an UPDATE statement.
- If either operand is a null value, the result is a null value. To avoid a null value result, use the VALUE string function in combination with the CONCAT operator.

Related reference

SQL scalar functions

Three types of SQL scalar functions are date/time functions, conversion functions, and string functions.

Chapter 4. Forms, reports, and charts

QMF creates reports from data stored in your database. A QMF form consists of a number of panels used to control report formatting.

When you select data (by running a query, importing data, or displaying a table or view), you can use QMF form panels to format the data into a report or chart. You can also use form panels to perform specific calculations on report data, such as adding columns or calculating percentages.

Using QMF forms

QMF automatically generates form panels when a table is displayed or a SELECT query is run without specifying a form. The resulting report is based on default formatting provided by QMF.

You can see the default form by typing `DISPLAY FORM.MAIN` (or `DISPLAY FORM`) after you run a query without specifying a form name on the RUN command.

Each form panel has entry areas in which information is added or changed. In the instructions in this information, a letter is assigned to each entry area on each form panel (such as **C**) and corresponds to the description following the panel. If there is a default value, it is shown in the entry area on the panel. Each entry area is described in terms of its effect on reports. If an entry area affects charts, that description follows.

Creating reports in QMF

Reports are initially created by applying a default form to the data retrieved from your query.

To alter a report's default format (for example, to change the column widths, add page headings, or change the spacing between lines of a report), you change the default choices displayed on the form panels. Data entered into an entry area can be converted to uppercase, depending on the setting of the CASE option of your profile.

Displaying a report without any data

With the LAYOUT command, you can view a report with generic data so that you can test a form you are creating.

Variable data is displayed using the letters A, B, C, D, E, F, and X, and the numbers 0, 1, 2, 3, 4, 5, and 6. All other text (including headings) is displayed as entered. You can tailor the different form panels to produce a representative report independent of the data. Combined with the LAYOUT command, forms with complex variables can be used repeatedly.

Related reference

LAYOUT

The LAYOUT command generates a sample QMF report using just a QMF form as input. This can assist in the development of a QMF form by providing a visual rendering of a representative report.

Symbols used in reports to indicate errors

When QMF cannot display a value in a report, it displays a special symbol in place of the value. The symbol that is displayed depends on the underlying cause.

Refer to the following table for a list of the symbols and their meanings.

Table 15. Error symbols that can appear on QMF forms	
Symbol displayed	**Cause of error**
********	The column is not wide enough to display the formatted value. Only numeric columns display this symbol. (Character columns are truncated instead.)
>>>>>>>>	The value exceeds the maximum value allowed by the data type for that column. This is called an overflow condition and is usually detected by QMF.
????????	The value is undefined. The following conditions will result in an undefined value in the report: • Numeric underflow • Numeric overflow detected by the database • Dividing a value by zero (in a query, calculation, or column definition) • Expressions that REXX is unable to evaluate • REXX expressions that evaluate to a nonnumeric value • Aggregations calculated using undefined values (except FIRST and LAST)
' ' (blanks)	The data has no instance (DSQNOINS) or no relationship (DSQNOREL).

Common report format changes

You are likely to make certain changes to alter the format of a report more often than other changes. You make these changes in specific form panels.

The following table lists some common additions or changes that alter the format of a report, and lists the appropriate form panel (or panels) you should normally use.

Table 16. Report formatting quick reference		
Aspect of report you need to add or change	**Specific element you need to add or change**	**Form panel to use**
Breaks in the report	Default break text	MAIN, OPTIONS
	Break text width	OPTIONS
	Break heading text	BREAK*n*
	Break footing text	MAIN, BREAK*n*
	Break summary	BREAK*n*
	Placement on page	BREAK*n*
	Outlining	MAIN, OPTIONS
Calculations	(No specific element)	CALC

Table 16. Report formatting quick reference (continued)		
Aspect of report you need to add or change	**Specific element you need to add or change**	**Form panel to use**
Specifications for report columns	Alignment	COLUMNS (Specify panel)
	Definition	COLUMNS (Specify panel)
	Heading	MAIN, COLUMNS
	Usage	MAIN, COLUMNS
	Indent	MAIN, COLUMNS
	Width	MAIN, COLUMNS
	Editing	MAIN, COLUMNS
	Sequencing	MAIN, COLUMNS
	Automatic ordering	OPTIONS
	Headings repeated at breaks	BREAK*n*
	Headings repeated at detail blocks	DETAIL
Conditional formatting	(No specific element)	CONDITIONS
Detail block text	Remove tabular information	DETAIL
	Specify placement of tabular information	DETAIL
	Include text with column values	DETAIL
Detail heading text	(No specific element)	DETAIL
Final text on the report	Placement on page	FINAL
	Width	OPTIONS
	Final summary	FINAL
Fixed columns	(No specific element)	OPTIONS
Whether a new page is started	For breaks	MAIN, BREAK*n*
	For detail block text	DETAIL
	For final text	FINAL
Page heading and footing	(No specific element)	MAIN, PAGE
Associate a panel variation with a condition	(No specific element)	DETAIL
Separator lines	(No specific element)	OPTIONS
Spacing between detail blocks	(No specific element)	OPTIONS, DETAIL

Creating charts in QMF

Certain entry areas on the form panels determine what appears on a chart, such as chart headings, legends, axis labels, and data plotted on the X and Y axes. However, not all entry areas on all panels affect charts.

The descriptions of the form panels point out both the panels and panel entry areas that affect charts and how these panels can be modified.

The following table lists some common additions or changes that alter your chart within QMF, and lists the appropriate form panel (or panels) you can use to make these changes.

Table 17. Chart function quick reference

To add or change:	Use this form panel:
Legend labels (Y-data column headings)	MAIN, COLUMNS
X-axis data labels (BREAK or GROUP columns)	MAIN, COLUMNS
Y-axis data (numeric data columns)	MAIN, COLUMNS
Chart heading (page heading)	MAIN, PAGE
Vertical position of chart heading	PAGE
Function name in legend label	OPTIONS

You cannot chart data or tables containing columns defined as BINARY, VARBINARY, or XML.

FORM.MAIN

Use FORM.MAIN to make simple changes to a report or chart.

Other panels (see the following table) work with FORM.MAIN to modify the appearance of reports or charts.

Table 18. Report/chart appearance change guide

Form name	Function	Additional information
FORM.MAIN	Basic format of a report or chart	“FORM.MAIN” on page 200
FORM.BREAK*n* (*n* = 1 to 6)	Text before and after breaks in a report	“FORM.BREAKn” on page 203
FORM.CALC	Expressions for calculations in a report	“FORM.CALC” on page 210
FORM.COLUMNS	Use of columns in a report or chart	“FORM.COLUMNS” on page 213
FORM.CONDITIONS	Expressions for conditional formatting	“FORM.CONDITIONS” on page 222
FORM.DETAIL	Text included with column values or headings of a report	“FORM.DETAIL” on page 224
FORM.FINAL	Content and placement of final text in a report	“FORM.FINAL” on page 229
FORM.OPTIONS	Miscellaneous adjustments to a report	“FORM.OPTIONS” on page 233
FORM.PAGE	Content and placement of page headings and footings in a report or chart	“FORM.PAGE” on page 238

Everything entered on FORM.MAIN is automatically reflected in a corresponding entry area on one of the other form panels. However, not all of the entry areas on the other panels are reflected on FORM.MAIN.

The previous table shows the entry areas on the FORM.MAIN panel. There are two areas on the FORM.MAIN and FORM.COLUMNS panels that are not entry areas: `Total Width of Report Columns` and `NUM`.

```
FORM.MAIN

COLUMNS:               Total Width of Report Columns: 42

      A                                                B      C      D      E      F
 NUM COLUMN HEADING                                  USAGE   INDENT WIDTH EDIT  SEQ
 --- ------------------------------------------- ------- ------ ----- ----- ---
   1 ID                                                      2      6     L     1
   2 NAME                                                    2      9     C     2
   3 DEPT                                                    2      6     L     3
   4 JOB                                                     2      5     C     4
   5 YEARS                                                   2      6     L     5

PAGE:    HEADING  ===> G
         FOOTING  ===>
FINAL:   TEXT     ===> H
BREAK1:  NEW PAGE FOR BREAK? ===> NO
         FOOTING  ===>                I
BREAK2:  NEW PAGE FOR BREAK? ===> NO
         FOOTING  ===>
OPTIONS: OUTLINE? ===> YES         J         DEFAULT BREAK TEXT? ===> YES

1=Help     2=Check    3=End          4=Show          5=Chart         6=Query
7=Backward 8=Forward  9=            10=Insert       11=Delete       12=Report
OK, cursor positioned.
COMMAND ===>                                                 SCROLL ===> PAGE
```

Figure 11. Entry areas on FORM.MAIN

Entry areas A through F correspond to identical entry areas on the FORM.COLUMNS panel. If all the columns in the form are not visible on the FORM.MAIN panel, you can scroll forward and backward to see them.

With these entry areas you can:

A
Assign column headings.

The DSQDC_COL_LABELS global variable controls whether the column heading defaults to the database label assigned to the column or the name of the column in the table from which it was selected.

B
Choose how to process columns.

C
Adjust indentation of columns.

D
Adjust width of columns.

The "Default widths of data types" table in FORM.COLUMNS shows the default width for each data type.

E
Specify formatting of columns. You can use certain edit codes in this field.

F
Change the sequence of columns.

Reports
The order of columns in the form is determined by the order in which they are specified in the SELECT statement of the query. Change the order of columns in the report by using the automatic reorder option or by changing the sequence (SEQ) column (F) on the FORM.MAIN panel.

Charts
Of these entry areas, COLUMN HEADING, USAGE, WIDTH, and EDIT apply to charts. The codes that appear in the USAGE entry area affect processing.

Entry areas G through J have corresponding form panels.

G PAGE

Reports
Enter one line of page heading and footing text for the report. QMF determines the horizontal and vertical placement of the heading and footing lines. The PAGE entry area corresponds to two entry areas on the FORM.PAGE panel.

Charts
Whatever appears in the PAGE entry area for a report heading also appears on a chart as its heading. Footing text cannot be specified for a chart.

H FINAL

Reports
Enter one line of final text for the report. The default placement of the line can be changed on the FORM.FINAL panel. The FINAL entry corresponds to one entry on the FORM.FINAL panel.

I BREAK1 and BREAK2

Reports
Enter footing text for up to two levels of breaks, and specify whether to start a new page each time the value in the specified break column changes. QMF determines the horizontal and vertical placement of the break footings. The BREAK1 and BREAK2 entry areas correspond to entry areas on the FORM.BREAK1 and the FORM.BREAK2 panels.

J OPTIONS

Reports
For reports with breaks, use the OUTLINE option to determine whether QMF displays the value of the break column on each tabular data line of the report. YES displays the value in the BREAK column only when the value itself changes.

For reports with breaks, use the DEFAULT BREAK TEXT option to determine whether to generate default break footing text to mark the BREAK aggregation line. When you do not enter any break footing text, YES displays a default break footing of asterisks.

This entry area corresponds to two entry areas on the FORM.OPTIONS panel.

Total width of report columns

Reports
This area shows the character width of the columns of the report.

You cannot change this area directly, but when you change INDENT, WIDTH, or edit codes for a column (or use a usage code of OMIT or ACROSS), the new total width of the report columns (in characters) appears after the colon.

If you use an edit code of G with DBCS data, each double-byte character counts as two positions.

If you use the usage code ACROSS, the width appears as an algebraic expression of the form: $a + (N \times b)$.

a
A constant value.

N
An unknown that stands for the number of sets of columns that are duplicated across the page, one set for each distinct value in the ACROSS column.

b
The width of each group of columns.

NUM

Reports
This area shows the number of each column in the order in which it was selected by the query that was run. You cannot change this area, but you can change the order of your columns by using the SEQ entry area.

You can indicate which column you want to use as a substitution variable by using its column number. For example, &6 refers to the sixth column selected by the query, even though it might not appear in the sixth position of the report.

Usually, columns appear on the report from left to right in order by their sequence numbers. However, when you use BREAK, GROUP, or an aggregation function on FORM.MAIN or FORM.COLUMNS and specify YES for `Automatic reordering of report columns?` on FORM.OPTIONS, QMF automatically reorders the columns in the report.

With automatic column reordering, if you use one or more of the BREAK codes as a usage, the control columns are moved to the left of the report. They appear there in order by their BREAK code numbers.

Also, columns whose usage is one of the aggregating usage codes (AVERAGE, COUNT, FIRST, LAST, CALC*id*, MAXIMUM, MINIMUM, STDEV, SUM, CPCT, CSUM, PCT, TPCT, or TCPCT) are moved to the right of the report and appear there in order by their column numbers.

The Report text line width column (Area C) and Automatic reordering of report column (Area J) in FORM.OPTIONS provides more information about width and order of columns.

Related concepts

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.

Usage codes
QMF usage codes can be entered in the USAGE field on QMF FORM.MAIN or FORM.COLUMNS to define how to use column data to produce reports and charts.

Related reference

Global variables that control various displays
DSQDC global variables control the display of certain kinds of information. All of these global variables can be modified by the SET GLOBAL command.

FORM.BREAKn

Use the FORM.BREAK*n* panels (where *n* is a number from 1 through 6) to make choices about the text and its placement for up to six breaks in a report. QMF places the text you specify on each break panel after its associated break in the report.

FORM.BREAK*n* panels do not affect charts.

Specify a break usage code in the USAGE entry area (B) on FORM.MAIN or FORM.COLUMNS opposite one of the column names. That column then becomes the *control column* and a break occurs in the report whenever the value in this control column changes.

When it evaluates values in VARCHAR columns, QMF differentiates between a value padded with blanks or hexadecimal zeros and the same values without these trailing characters. Using FORM.BREAK*n* panels in such cases creates a break.

You can use the same level of break on multiple columns. In this case, a break occurs when a value changes in any one of those columns.

Area I on FORM.MAIN specifies footing text for BREAK1 and BREAK2 in a report and whether to start a new page each time the value in the control column changes. Whatever you specify in area I of FORM.MAIN is reflected on FORM.BREAK1 and FORM.BREAK2. What you specify on areas H and N on FORM.BREAK1 and FORM.BREAK2 is reflected on FORM.MAIN.

There are six FORM.BREAK panels – one for each possible level of break. They are all the same, except for the panel title. The following figure shows the entry fields on the FORM.BREAK panels.

```
FORM.BREAK1

 A New Page for Break?        ===> NO   B Repeat Detail Heading?   ===> NO
 C Blank Lines Before Heading ===> 0    D Blank Lines After Heading ===> 0
 E LINE F ALIGN G BREAK1 HEADING TEXT
    ----    ------   ----+----1----+----2----+----3----+----4----+----5----+
    1       LEFT
    2       LEFT
    3       LEFT
            *** END ***

 H New Page for Footing?      ===> NO   I Put Break Summary at Line ===> 1
 J Blank Lines Before Footing ===> 0    K Blank Lines After Footing ===> 1
 L LINE M ALIGN N BREAK1 FOOTING TEXT
    ----    ------   ----+----1----+----2----+----3----+----4----+----5----+
    1       RIGHT
    2       RIGHT
    3       RIGHT
            *** END ***

 1=Help        2=Check     3=End     4=Show        5=Chart       6=Query
 7=Backward    8=Forward   9=       10=Insert     11=Delete     12=Report
 OK, FORM.BREAK1 is displayed.
 COMMAND ===>                                          SCROLL ===> PAGE
```

Figure 12. Entry fields on the FORM.BREAK panels

A New page for break?
Specify whether to begin a new page whenever the value in the control column for the break changes. This value affects printed and exported reports. It does not affect displayed reports. A new page is started if the report is not already at the top of the page.

Specifying YES for more than one break level can produce more pages than expected in your printed or exported report. Extra pages can occur when multiple breaks occur at the same time.

If you specify two or more breaks and also specify YES for `New page for break?` on each break, a page is generated for each specified break whenever the highest break level occurs. Multiple breaks frequently occur together, since the highest break level forces all lower break levels to occur. All breaks occur for the first row of data in a report.

B Repeat detail heading?
Specify whether the detail heading is to be repeated at the beginning of each new break level that follows the break heading text and before the detail block text.

In printed reports, if a break begins at the top of a page and you specify YES here, only one set of detail headings appears.

Detail headings consist of the detail heading text that is specified on the FORM.DETAIL panel, plus column headings (unless you suppress column headings on the FORM.DETAIL panel).

Specifying YES for `Repeat Detail Heading?` on FORM.DETAIL overrides the specifications that are given here.

C Blank lines before heading
Enter the number of blank lines before the first line of the break heading text, if specified, or before the first break member line if there is no break heading text. The value can be any number from 0 through 999.

D Blank lines after heading
Enter the number of blank lines after the last line of the break heading text, if specified. This entry can be any number from 0 through 999.

E LINE
Identify the lines of break heading text and specify their positions relative to themselves and to the line at which the break heading starts (as indicated in the `Blank Lines Before Heading` entry area). You can specify any number from 1 through 999 or a blank. If blank, QMF ignores any associated text.

The numbers that you choose need not start with 1 or be consecutive.

For example, consider the following values on FORM.BREAK1:

```
LINE  ALIGN  BREAK1 HEADING TEXT
----  -----  ------------------
3     LEFT   DEPARTMENT &4
2     LEFT   BEGINNING OF LISTING
```

These values display as follows:

```
BEGINNING OF LISTING
DEPARTMENT 35
```

Notice that a blank line appears before the first line of text.

F ALIGN

Specify where each line of the break heading text is to be placed horizontally in the report. You can place the lines anywhere in the width of the report. For an online report, the width is the width of the displayed report; for a printed report, the width is the page width.

Left
Left-justifies the break heading text.

Right
Right-justifies the break heading text.

Center
Centers the break heading text.

n
Begins the break heading text in the *n*th position of the line, where *n* can be any number from 1 through 999999.

Append
Attaches the line to the end of the previous line of break heading text. If APPEND is used on the first line of break heading text, the line of text is left-aligned.

The appended line of text must have the same LINE value as the line of text it is being appended to. If the report is not wide enough to accommodate the appended line of text, some of the text might be truncated.

For example, consider the following entries on FORM.BREAK1:

```
Blank Lines Before Heading ===> 0
LINE  ALIGN   BREAK1 HEADING TEXT
----  ------  ----------------------
1     LEFT    DEPARTMENT
1     APPEND   &4
3     LEFT
```

These values align the columns in the resulting report as shown:

```
              DEPT        COMM  JOB        SALARY
            ------  ----------  -----  ----------
          DEPARTMENT 66
                66       55.50  CLERK    10988.00
                            -   MGR      18555.50
                       844.00  SALES    16858.20
                       200.30  SALES    21000.00
                       811.50  SALES    18674.50
                                        ----------
                                    *    86076.20

          DEPARTMENT 84
                84      188.00  CLERK    13030.50
                            -   MGR      19818.00
```

G BREAK1 HEADING TEXT

Enter the heading text that you want associated with the break. Every time the value in the break column changes, the text that is specified in this entry is displayed in the report. You can add up to 999 lines of break heading text by using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

By default, break heading text extends from the left to the right margin of a report. However, you can choose the width of break heading text on the `Report text line width` entry on FORM.OPTIONS.

To make the break heading text appear in a report in uppercase and lowercase, specify in your profile a CASE value of either STRING or MIXED.

STRING
Displays break heading text as entered, but converts any other input to uppercase.

MIXED
Displays all input exactly as entered.

Break heading text can contain the following variables:

Global variables
Use SET GLOBAL to set variables for use in break heading text.

&*n*
n is a number that represents the current row in column *n* on the form that is used for this report. Column *n* is not necessarily the *n*th column that you see in a report. It is the *n*th column that is listed on FORM.MAIN and FORM.COLUMNS. For example, the break heading text `BEGINNING OF DEPARTMENT &3` might display the following line on a report:

```
BEGINNING OF DEPARTMENT 38
```

The following variables can also be used with date, time, timestamp, and timestamp with time zone values in break heading text:

&DATE
The current date is formatted according to the default at your site, which reflects one of the following date formats:

- USA (United States of America)
- EUR (European)
- ISO (International Standards Organization)
- JIS (Japanese Industrial Standard)
- An alternative date format that is supplied by your site

&TIME
The current time is formatted according to the default at your site, which reflects one of the formats that are listed under &DATE.

&PAGE
The page number is printed on each page when the report is formatted.

If a page in a report is wider than either the printer width or the default printing width that is specified in your profile, QMF splits the page. It gives all parts of the split page the same page number, but with subscripts. (If you are using DBCS data and QMF splits the page, printing resumes on the second and subsequent pages of the report at the fourth byte position from the left side of the page.)

&ROW
The number of the first data row within the current break level is printed or displayed in your report.

H New page for footing?
Specify whether to begin a new page (if the report is printed) before displaying any break footing text specified. A new page is started if the report is not already at the top of the page.

I Put break summary at line
Specify whether the break summary is to be formatted, and, if so, where it is to be placed in relation to the lines of break footing text. The value for this entry can be any number from 1 through 999 or the word NONE (for no break summary).

J Blank lines before footing
Specify the number of blank lines before the first line of break footing text. This entry can be any number from 0 through 999 or the word BOTTOM.

K Blank lines after footing
Specify the number of blank lines after the last line of the break footing text. The value for this entry can be any number from 0 through 999.

If you specify a break and you have a column-wrapped column with a usage code of FIRST, LAST, MIN, or MAX, you might need to increase the value in this field to see all the wrapped lines in the break summary. You can use the CW edit code to wrap data in columns.

L LINE
Identify the lines of break footing text and specify their positions relative to themselves and to the line at which the break footing starts (as indicated in the `Blank Lines Before Footing` entry area). You can specify any number from 1 through 999 or a blank. A blank ignores any associated text.

The numbers that you choose need not start with 1 or be consecutive.

For example, consider the following values on FORM.BREAK1:

```
LINE  ALIGN  BREAK1 FOOTING TEXT
----  -----  ------------------
3     LEFT   DEPARTMENT &4
2     LEFT   END OF LISTING
```

These values display as follows in the report:

```
END OF LISTING
DEPARTMENT 35
```

M ALIGN
Specify where each line of the break footing text is to be placed horizontally in the report. For breaks without break summaries, you can place the lines of break footing text anywhere in the width of the report. The width of the report is shown at the top of FORM.MAIN.

For breaks with break summaries created with usage codes (except OMIT, BREAKn, GROUP, or ACROSS), QMF places the lines of break footing text anywhere from the left margin to the beginning of the indent area that is associated with the left-most column of summary data.

Left
Left-justifies the break footing text.

Right
Right-justifies the break footing text.

Center
Centers the break footing text.

n
Begins the break footing text in the *n*th position of the line, where *n* can be any number from 1 through 999999.

Append
Positions the line at the end of the previous line of break footing text. If APPEND is used for a line of text that is not appended to another line, the line of text is left-aligned.

The appended line of text must have the same LINE value as the line of text it is being appended to.

For example, consider the following entries on FORM.BREAK1:

```
LINE  ALIGN   BREAK1 FOOTING TEXT
----  ------  ----------------------
1     RIGHT   TOTAL
1     APPEND   SALARIES--DEPT. &4;
3     RIGHT
4     RIGHT
5     RIGHT
```

These values align columns as follows in the resulting report.

```
       DEPT        COMM  JOB        SALARY
      ------  ----------  -----  ----------
         66       55.50  CLERK    10988.00
                      -  MGR      18555.50
                 844.00  SALES    16858.20
                 200.30  SALES    21000.00
                 811.50  SALES    18674.50
                                 ----------
      TOTAL SALARIES--DEPT. 66    86076.20

         84      188.00  CLERK    13030.50
                      -  MGR      19818.00
                 806.10  SALES    15454.50
                1285.00  SALES    17844.00
                                 ----------
      TOTAL SALARIES--DEPT. 84    66147.00
```

If the report is not wide enough to accommodate the appended line of text, some of the text might be truncated.

N BREAK1 FOOTING TEXT

Enter the footing text that you want associated with the break. Every time the value in the break column changes, the text that is specified in this entry is displayed in the report. You can add up to 999 lines of break footing text by using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

By default, break footing text extends from the left margin of a report either to the beginning of the break summary data (if any), or to the right margin of a report. However, you can choose the width of break footing text on the `Report text line width` entry on the FORM.OPTIONS panel.

To make the break footing text appear in a report in uppercase and lowercase, specify in your profile a CASE value of either STRING or MIXED.

STRING
Displays break footing text as entered, but converts any other input to uppercase.

MIXED
Displays all input exactly as entered.

Break footing text can contain the following variables:

Global variables
Use SET GLOBAL to set variables for use in break footing text.

&n
n is a number that stands for the most current value in column *n* on the form that is used for this report. Column *n* is not necessarily the *n*th column that you see in a report. It is the *n*th column that is selected from the database, or the *n*th column that is listed on FORM.MAIN and FORM.COLUMNS.

For example, the break footing text `END OF DEPARTMENT &3` might display as follows on a report:

```
END OF DEPARTMENT 38
```

&COUNT
The number of rows that are retrieved or printed since the last break at the same level. This value increases from data row to data row.

&ROW
The number of the last data row is printed or displayed in your report.

id
Calculated value.

&CALC*id* is described in FORM.CALC.

&DATE
The current date.

&TIME
The current time.

&PAGE
The current page number.

For more information about those variables, see the earlier descriptions of &DATE, &TIME, and &PAGE.

&an
n is a valid column number and *a* is one of the following QMF aggregation functions: AVG, COUNT, CPCT, CSUM, FIRST, LAST, MAX, MIN, PCT, STDEV, SUM, TCPCT, TPCT. The values of the aggregations are based on running values within the current break level.

For example, assume that the fourth column of the report contains salaries and you want to summarize the salaries in each group in break footing text. Type the following BREAK1 FOOTING TEXT:

```
TOTAL SALARY FOR DEPARTMENT &3 IS &SUM4
```

The resulting line of break footing text in the report would be:

```
TOTAL SALARY FOR DEPARTMENT 38 IS $77,285.55
```

If you specify the aggregation variable in break footing text, you need not specify that same aggregation as the usage for that column. However, the aggregation must be compatible with the edit code and data type of the column. For example, you cannot specify &SUM3 in your final text if the data in column 3 has a character edit code.

If you use a percent aggregation variable (PCT, TPCT, or TCPCT) in break footing text and you associate it with a column that has a D edit code, QMF formats the percent value as if it had an L edit code. Likewise, if you use the STDEV (standard deviation) aggregation variable and associate it with a column that has a P or a D edit code, QMF formats the standard deviation as if it had an L edit code.

Related concepts

Variables used in forms
You can use global variables (both those defined by users and those supplied by QMF) and form variables in QMF forms. A variable can replace a string of text or a numeric value. You can assign different values to the variable to produce different reports without changing the form.

Related reference

Edit codes for character data
You can use several edit codes to format character data.

Edit codes for numeric data
You can use several edit codes to format numeric data.

FORM.CALC

On the FORM.CALC panel you can enter expressions for report calculations. This panel initially contains only one row – a place for one expression. However, up to 998 additional rows can be inserted.

FORM.DETAIL
FORM.DETAIL consists of detail variations that you define. You can create up to 99 variations, and each variation can correspond to conditions entered on FORM.CONDITIONS. Unless each condition is mutually exclusive, different detail variations can be displayed for the same data row.

FORM.MAIN
Use FORM.MAIN to make simple changes to a report or chart.

FORM.OPTIONS
Use FORM.OPTIONS to adjust the appearance of your report.

SET GLOBAL
The SET GLOBAL command assigns values to global variables from the QMF command line, from a procedure, or through the callable interface. You cannot change the value of a global variable that is defined as read-only.

FORM.CALC

On the FORM.CALC panel you can enter expressions for report calculations. This panel initially contains only one row – a place for one expression. However, up to 998 additional rows can be inserted.

Restriction: FORM.CALC uses expressions written in REXX, which is not available in CICS.

The following figure shows the entry fields on the FORM.CALC panel. Each entry area is described below in terms of its effect on reports. FORM.CALC does not affect charts.

```
FORM.CALC
                                                      C       D       E
 A              B                                     PASS    For &CALCid
 ID   CALCULATION EXPRESSION                          NULLS?  WIDTH   EDIT
 ---  ---------------------------------------------  ------  -----   -----
                                                      NO      10      C

      *** END ***

 1=Help        2=Check     3=End     4=Show        5=Chart       6=Query
 7=Backward    8=Forward   9=       10=Insert     11=Delete     12=Report
 OK, Cursor positioned.
 COMMAND ===>                                              SCROLL ===> PAGE
```

Figure 13. Entry fields on the FORM.CALC panel

A ID
Enter a one- to three-character identifier for the corresponding calculation expression. The identifier is any number from 1 through 999. When appended to the CALC usage code or the &CALC variable, it identifies which expression on FORM.CALC is to be used in a calculation.

The *id* variable can be used only in detail block text, final text, and break footing text. The CALC*id* usage code and *id* variable activate the evaluation of the calculation expression on FORM.CALC whose ID equals *id*.

For a &CALC variable, the evaluated result is edited according to the width and edit code specified for the expression in the FORM.CALC panel. For a CALC*id* usage code, the evaluated result is edited according to the width of the columns and the edit code of the CALC.

B CALCULATION EXPRESSION
Enter an expression. It can contain up to 50 characters. You cannot execute QMF commands (using the callable or command interfaces) from within a REXX program used in FORM.CALC.

Other than *id*, any valid form variable can be used in the expressions. The following variables are valid:

Global variables
Use SET GLOBAL to set variables for use in calculation expressions.

Column variables: &n
n is a column number.

Aggregation variables: &an
n is a valid column number, and *a* is one of the following QMF aggregation functions: AVG, COUNT, CPCT, CSUM, FIRST, LAST, MAX, MIN, PCT, STDEV, SUM, TCPCT, TPCT.

&ROW
Print the number of the data row at the time the calculation is evaluated. The &ROW variable is replaced just before the &CALC*id* variable or CALC usage code is evaluated.

&COUNT
Row count.

&DATE
The current date.

&TIME
The current time.

&PAGE
The current page (always 1 for displayed reports).

You can find more detailed descriptions of &COUNT, &DATE, &TIME, and &PAGE in FORM.BREAKn.

When an expression is entered, its variables are validated. Column variables are checked for valid column numbers and for compatible usages or edit codes (or both). Be sure to use substitution variables that are compatible with the expression, because QMF does not check for nonnumeric substitution variables in an arithmetic expression. For example, if the sixth column has an edit code of C and the expression uses &SUM6, an error exists and a message is issued.

If you encounter a syntax error on the expression, you must correct it either in the REXX program itself or in the REXX expression. Be sure to follow the REXX coding rules.

For example, suppose that you include in the expression a program name that does not exist. After you correct the program name or create the program, enter `SHOW F.CALC` and make any necessary modifications. If you do not need to make any other changes, retype one of the characters in the expression. Doing this causes QMF to validate the variables again to ensure you have built your form correctly. If you do not revalidate your form, you might get unpredictable results.

C PASS NULLS
Enter YES or NO.

YES
Allows you to use the values provided by QMF, which are shown in the following table, to change the handling of the value depending on the situation:

Table 19. Default handling of database null values

Situation	Character string that replaces the value
Data is null	DSQNULL
Data is undefined	DSQUNDEF
Data has numeric overflow	DSQOFLOW
Data has no instance	DSQNOINS
Data has no relationship	DSQNOREL

NO
Returns a null for the values listed above. Nothing is passed to REXX for evaluation.

For example, any database variable that is null (a database null) is replaced with the character string DSQNULL before the expression is passed to REXX for evaluation. You can provide a REXX expression or program that checks for the string and substitutes 0 (or whatever is appropriate for your purpose) for the database null value.

If a null value is returned by the REXX expression, you can pass it to your report.

If the expression contains a substitution value that is null, undefined, overflows, or has no instance or no relationship, then the entire expression will be set to the value that represents that condition. This expression reduction is performed only on expressions, not comparisons.

If the expression contains more than one substitution value that is null, undefined, overflows, or has no instance or no relationship, then the following order of precedence will be used for expression reduction:

1. Undefined
2. Overflow
3. Null
4. No instance
5. No relationship

D WIDTH
Enter the width (in single-byte characters) to which the evaluated result of the corresponding expression is edited in report text. It is applicable only to results obtained for &CALC*id* variables. If the CALC*id* usage cannot be edited according to the edit code for the column, the edit code of the CALC*id* is used.

WIDTH is a 5-character entry field. It must contain a number from 1 through 32,767. The default is 10.

E EDIT
Enter the edit code to be used when the evaluated result of the corresponding expression is edited in report text. It is applicable only to results obtained for &CALC*id* variables. Results of CALC*id* usages are edited using the edit code specified for the column on FORM.MAIN or FORM.COLUMNS.

EDIT is a 5-character field. The default is C (for character data) when a line is inserted in FORM.COLUMNS. Only the edit codes shown in the following table are accepted.

Table 20. Edit codes accepted on FORM.CALC

Data type to be formatted	Edit codes accepted on FORM.CALC	Effect or usage
Numeric	D, E, I, J, K, L, P	You can use optional suffixes with these numeric edit codes. Z is an optional suffix for all numeric edit codes and can be used to suppress zero values. C is an optional suffix for the D edit code and causes QMF to use the currency symbol specified in the global variable DSQDC_CURRENCY instead of the default currency symbol. You can add a decimal scale value from 0 to 99 to any numeric edit code except E.
Character	C	Character editing (default).
User-defined	U*xxxx*, V*xxxx*	User edit codes for numeric or character editing.

The following figure summarizes the results returned when an edit code is applied to an expression.

Table 21. Results returned when an edit code is applied to an expression

Result from user expression	Applicable edit code	Edited result	
Numeric	Numeric	Edited according to edit code.	
	Nonnumeric	Character representation of result edited according to edit code.	
	U*xxxx*, V*xxxx*	Edited by user edit routine (expression result for U*xxxx* is passed to routine as extended floating-point data).	
Nonnumeric	Numeric	Edited as if C (character).	
	Nonnumeric	C*xx*	Character
		U*xxxx*, V*xxxx*	As edited by user edit routine.

Related concepts

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.

Usage codes
QMF usage codes can be entered in the USAGE field on QMF FORM.MAIN or FORM.COLUMNS to define how to use column data to produce reports and charts.

Related reference

FORM.BREAKn
Use the FORM.BREAK*n* panels (where *n* is a number from 1 through 6) to make choices about the text and its placement for up to six breaks in a report. QMF places the text you specify on each break panel after its associated break in the report.

FORM.FINAL
Use FORM.FINAL to make detailed choices about the content and placement of final text in a report. QMF places the text at the end of the report, and you can use it, for example, to identify the final summary data of a report.

SET GLOBAL
The SET GLOBAL command assigns values to global variables from the QMF command line, from a procedure, or through the callable interface. You cannot change the value of a global variable that is defined as read-only.

FORM.COLUMNS

Use FORM.COLUMNS to make choices about the uses of the columns. What you specify on FORM.COLUMNS is reflected on FORM.MAIN.

Conversely, what you specify on FORM.MAIN (areas A through F) is reflected on FORM.COLUMNS.

The following figure shows the entry fields on the FORM.COLUMNS panel.

```
FORM.COLUMNS

COLUMNS:          Total Width of Report Columns: 66
          A                               B       C      D       E       F
 NUM    COLUMN HEADING                  USAGE    INDENT  WIDTH   EDIT   SEQ
 ---    ------------------------------  ------   -----   -----   -----  ---
   1    ID                                       2       6       L      1
   2    NAME                                     2       9       C      2
   3    DEPT                                     2       6       L      3
   4    JOB                                      2       5       C      4
   5    YEARS                                    2       6       L      5
   6    SALARY                                   2       10      L2     6
   7    COMM                                     2       10      L2     7
   8    Total Earnings                           2       12      L2     8
        *** END ***

 1=Help        2=Check     3=End       4=Show        5=Chart     6=Query
 7=Backward    8=Forward   9=Specify  10=Insert     11=Delete   12=Report
 OK, FORM.COLUMNS is displayed.
 COMMAND ===>                                            SCROLL ===> PAGE
```

Figure 14. Entry fields of the FORM.COLUMNS panel

A COLUMN HEADING

Reports

Assign column headings. On the default form, column headings can be any of the following:

- The database label assigned to the column or the name of the column in the table from which it was selected

  The DSQDC_COL_LABELS global variable controls whether the column heading defaults to the database label or the column name.
- A generated heading constructed by QMF for columns that contain constants or calculated values

You can enter any new heading of up to 40 characters over a heading shown in the COLUMN HEADING area. The heading, like the original column name, can contain blanks or special characters. To create multiple-line headings, use an underscore in a column heading to specify a break between lines. For example, EMPLOYEE_NAME displays as follows in the report:

```
EMPLOYEE
NAME
```

A single underscore before or after an entire column heading has no effect. For example, _EMPLOYEE NAME does not add a blank line. However, consecutive underscores within the text of a column heading produce one or more blank lines in a column title. You can have up to nine lines in a column heading.

For example, consider these two column names:

```
1  ONE_TWO_THREE_FOUR_FIVE_SIX_SEVEN
2  SIX__LINE___TITLE
```

There is one blank line for each underscore entered, so these values display as follows in the report:

```
      ONE         SIX
      TWO
    THREE       LINE
    FOUR
    FIVE
     SIX        TITLE
    SEVEN
```

If you are using double-byte characters in column headings, you can specify a break between lines if the underscore you use is a single-byte character.

To create column headings in uppercase and lowercase, specify in your profile a CASE value of either STRING or MIXED.

Headings are left-justified over columns of character data, and right-justified over columns of numeric data. If there is more than one line in the heading, the longest line is justified, and shorter lines are centered within the longest line. You can override these defaults by entering a new alignment value.

If any line of a heading is longer than the width of the column, it fills the whole width of the column and is cut off on the right.

You cannot use a global variable in a column heading; QMF will not substitute a value for the variable.

Charts

Column headings for data plotted on the Y axis appear in the legend of a chart. Therefore, you probably want these column headings to be as concise as possible, or the legend will take up too much space on the chart.

B USAGE

Reports

Specify how you want a column processed for a report. If the usage code for a column is blank, the values in the column are listed with no other processing unless one or more columns in the report has a usage of GROUP and at least one column has an aggregation usage. In that case, columns with blank usages are omitted. A number of aggregation functions, listed in the following table, can be entered in this area.

Table 22. Aggregation functions

Aggregation	Usage code	Minimum abbreviation	Additional information
Across	ACROSS	AC	“ACROSS usage code” on page 252
Average	AVERAGE (or AVG)	AV	“Aggregation usage codes” on page 252
Break1	BREAK, BREAK1	B, B1	“FORM.BREAKn” on page 203
Break1x	BREAKX, BREAK1X	BX, B1X	“FORM.BREAKn” on page 203
Break2	BREAK2	B2	“FORM.BREAKn” on page 203
Break2x	BREAK2X	B2X	“FORM.BREAKn” on page 203
Break3	BREAK3	B3	“FORM.BREAKn” on page 203
Break3x	BREAK3X	B3X	“FORM.BREAKn” on page 203
Break4	BREAK4	B4	“FORM.BREAKn” on page 203
Break4x	BREAK4X	B4X	“FORM.BREAKn” on page 203
Break5	BREAK5	B5	“FORM.BREAKn” on page 203
Break5x	BREAK5X	B5X	“FORM.BREAKn” on page 203
Break6	BREAK6	B6	“FORM.BREAKn” on page 203
Break6x	BREAK6X	B6X	“FORM.BREAKn” on page 203
Calculate	CALC*id*	CA	“FORM.CALC” on page 210
Count	COUNT	CO	“Aggregation usage codes” on page 252

Table 22. Aggregation functions (continued)

Aggregation	Usage code	Minimum abbreviation	Additional information
Cumulative percent	CPCT	CP	“Aggregation usage codes” on page 252
Cumulative sum	CSUM	CS	“Aggregation usage codes” on page 252
First	FIRST	F	“Aggregation usage codes” on page 252
Group	GROUP	G	“GROUP usage code” on page 257
Last	LAST	L	“Aggregation usage codes” on page 252
Maximum	MAXIMUM	MA	“Aggregation usage codes” on page 252
Minimum	MINIMUM	MI	“Aggregation usage codes” on page 252
Omit	OMIT	O	“OMIT usage code” on page 258
Percent	PCT	P	“Aggregation usage codes” on page 252
Standard deviation	STDEV	ST	“Aggregation usage codes” on page 252
Sum	SUM	SU	“Aggregation usage codes” on page 252
Total cumulative percent	TCPCT	TC	“Aggregation usage codes” on page 252
Total percent	TPCT	TP	“Aggregation usage codes” on page 252

C INDENT

Reports

Specify the number of blank spaces to the left of a column. The blank spaces separate the column from the previous column or from the left margin. INDENT can be any number from 0 through 999. For columns using a graphic edit code, the minimum indent is 1. The default INDENT for each column is 2.

INDENT is always specified as a number of single-byte characters.

D WIDTH

Reports

Specify the number of character positions reserved for displaying data from a column. This width applies to the column heading as well and can be any number from 1 through 32,767.

If the column you are displaying uses a graphic edit code, the WIDTH value can be any number from 1 through 16,383. The width required to display or print the data is twice the width defined for the column in the database plus one character space.

When assigning a width for numeric data, ensure that the value you specify accounts for space for the following characters as well as for digits:

- A minus sign (except with edit code J)
- A decimal point (when edit codes specify them)
- Separators for groups of thousands (with edit codes D, K, and P)
- A currency symbol (with edit code D)
- A percent sign (with edit code P)

If the length of a value to be displayed exceeds the width of the column (for example, when you attempt to display a column containing XML data):

- If it is numeric data, it is replaced with a row of asterisks (********).

In some cases, you can avoid a numeric overflow by using a different data type. For example, in an arithmetic operation, if all operands are decimal numbers and an overflow occurs, you can change at least one operand to a floating-point number. In this example, the operand can be a floating-point constant or a floating-point table column.

- If it is character, date, time, or timestamp data, it is cut off on the right or left (depending on the alignment specified for the data).

Resolve column width problems by changing the WIDTH value for the column and displaying the report again. Alternatively, you can specify that you want to keep the column width the same, but wrap data that will not fit on a line to the next line in the column. Column wrapping applies only to nonnumeric data.

The width of a column on the default form is at least as great as the longest line in the column heading. Otherwise, the assigned width depends on the data type of the column, as shown in the following table.

Table 23. Default widths of data types

Data type	**Width on default form**
SMALLINT	6
INTEGER	11
BIGINT	20
DECIMAL	The width of the column in the database, plus 3 character spaces.
FLOAT	10
DECFLOAT(16)	12 if decimal floating-point data is supported by the operating system; otherwise, metadata is displayed with a default width of 8.
DECFLOAT(34)	12 if decimal floating-point data is supported by the operating system; otherwise, metadata is displayed with a default width of 8.
CHAR	The width of the column in the database.
VARCHAR	The maximum width of the column in the database.
LONG VARCHAR	The smaller of: • The column width. • A width determined by QMF, based on the quantity and type of other columns in the report.
GRAPHIC	The width of the column in the database.
VARGRAPHIC	The width of the column in the database.
LONG VARGRAPHIC	The smaller of: • The column width. • A width determined by QMF, based on the quantity and type of other columns in the report.
DATE	10 or, if your date format is locally defined by your site, the larger of: • The width of the column heading. • The width of the locally defined date format.

Table 23. Default widths of data types (continued)

Data type	Width on default form
TIME	8 or, if your time format is locally defined by your site, the larger of: • The width of the column heading. • The width of the locally defined time format.
TIMESTAMP(0)	19
TIMESTAMP(n)	20 + n (where n = 1 to 12)
TIMESTAMP (0) WITH TIME ZONE	25
TIMESTAMP (n) WITH TIME ZONE	26 + n (where n = 1 to 12)
BINARY(*n*)	Metadata is displayed by default with a default width of 8 + *n*, where *n* is 1 to 255.
VARBINARY(*n*)	Metadata is displayed by default with a default width of 11 + *n*, where *n* is 1 to 32704.
XML	Metadata is displayed by default. If the column name is fewer than 3 characters, the default width is 3. If the column name is greater than 3 characters, the default width is the same as the width of the column name, up to 10.
CLOB	Metadata is displayed by default. The default width is locally defined, up to 10.
BLOB	Metadata is displayed by default. The default width is locally defined, up to 10.
DBCLOB	Metadata is displayed by default. The default width is locally defined, up to 10.

When inserting a line on FORM.COLUMNS, the default width is 10.

For single-precision floating point data, values with data types of FLOAT are treated the same for single-precision or double-precision numbers.

To work with DECFLOAT data in QMF, the processor on which QMF is running must support decimal floating-point instructions.

You can override default formatting behavior for character, numeric, decimal, date, and time data types by setting the following global variables:

DSQDC_EC_CHAR
DSQDC_EC_NUM
DSQEC_DEC
DSQDC_EC_DATE
DSQDC_EC_TIME

Values in these global variables override the default formatting rules that are shown in the previous table.

Charts

Specify the number of character positions for labels on the X axis of a chart.

If the width exceeds the allotted space, the labels might be omitted. Truncating the width of column headings is one way to handle the problem of omitted labels. When labels are truncated, more labels fit in the allotted space.

Values from columns with date, time, timestamp, and timestamp with time zone data types (treated as character strings) cannot appear on the Y axis.

For single-precision floating point data, values with data types of FLOAT are treated the same for single-precision or double-precision numbers.

To work with DECFLOAT data in QMF, the processor on which QMF is running must support decimal floating-point instructions.

E EDIT

Reports
Specify how QMF formats data for display. The default is C when inserting a line in FORM.COLUMNS.

Charts
The X-axis labels come from columns using GROUP or BREAK (or from the left-most column of the report when there is no GROUP or BREAK). The effect that edit codes have on the data in those columns appears in the X-axis labels. For example, if data selected for the X axis is column wrapped, only the first line is incorporated into the labels.

Numeric columns that are edited with U*xxxx* or V*xxxx* cannot be used for Y-axis data.

When column substitution values (&*n*) are used in the page heading (and, therefore, in the chart heading), they are edited according to the edit code for that column in the form.

You can use character edit codes with date, time, timestamp, and timestamp with time zone data to allow wrapping of those columns.

F SEQ

Reports
Enter numbers in this column to change the sequence of the columns in your report. Initial settings are taken from the NUM column. Any numbers from 1 through 999 are allowed. If two numbers are the same, those columns appear in the same order they are listed on the form. The `Automatic reordering of report columns` option on the FORM.OPTIONS panel must be set to NO (the default) for SEQ to have an effect on column reordering.

When variables are resolved, the column number is taken from NUM, not SEQ.

SEQ numbers are ignored in ACROSS reports.

Related concepts

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.

Related reference

SET PROFILE
The SET PROFILE command changes values in your QMF profile. These values influence the behavior of your QMF session.

Global variables that control various displays
DSQDC global variables control the display of certain kinds of information. All of these global variables can be modified by the SET GLOBAL command.

Related information

DBCS data and QMF objects

Specifying column attributes

Using the SPECIFY command, you can change the alignment of a column heading, change the data within a column, or define a column.

There are two ways to access the alignment and definition panels.

- Press the Specify function key to display the **Specify** panel, then choose `Alignment` or `Definition`.

- Enter SPECIFY ALIGNMENT or SPECIFY DEFINITION (or a valid abbreviation) on the command line, then move the cursor to the column you want and press Enter. This bypasses the **Specify** panel and takes you directly to the **Alignment** or **Definition** window.

Column alignment

When you specify alignment, the panel that is shown in the following figure overlays the FORM.COLUMNS panel and shows the alignment specifications for the column you chose.

```
                        Alignment
 Column number   :    3
 Column Heading  :  DEPT_HEADING_CAN_BE UP TO_40 CHARS LONG!

 Heading alignment  :  [DEFAULT  ]
 Data alignment     :  [LEFT     ]


 F1=Help  F5=Previous Column  F6=Next Column  F12=Cancel
```

Figure 15. The FORM.COLUMNS Alignment panel

Choices for heading and data alignment are LEFT, RIGHT, CENTER, and DEFAULT. The default for the heading and data of a column that contains character data is right-justified, while the default for the heading and data of a column that contains numeric data is left-justified.

To change an alignment value, type the new value over the current value. Use the tab key to move between the heading and data alignment entry fields and from one column alignment specification to another.

Column alignment applies mainly to tabular data. However, if you use _B with a substitution variable, the data is aligned as follows:

- The data is edited according to the edit code and width of the column.
- If the alignment is not DEFAULT, leading and trailing blanks are removed.
- The value is aligned according to the specified alignment value.
  - If the data is character, trailing blanks are removed.
  - If the data is numeric, leading blanks are removed.
  - If &_B is used, no blanks are removed.

In tabular reports, leading and trailing blanks are removed if the value for data alignment is LEFT, RIGHT, or CENTER. The blanks are not removed if the data alignment value is DEFAULT.

If you are using edited character data with leading blanks, or edited numeric data with trailing blanks, the blanks are not removed regardless of the alignment value.

Column definition

Restriction: Column definition is not available in CICS because its function depends on REXX.

Column definition allows you to define a new column of data using an expression. There are some differences between columns that are retrieved by a query and columns that you define. The main difference is in the data type and length that is assigned to user-defined columns.

When you define a column, you are prompted to enter an expression to define the column and whether null values are included when REXX evaluates the expression. QMF determines the data type and column length based on the edit code and column width specified for that column on FORM.COLUMNS. However, if you use a usage code for the defined column that does not agree with the edit code for the column, the usage code determines the data type.

Another difference between user-defined columns and columns retrieved from the database is that values for user-defined columns are not retained when the data is saved or exported.

Column wrapping can also appear to work differently for defined columns.

- If the data for a defined column is less than 254 bytes, there is no apparent difference in how column wrapping works.
- If the data for a defined column is greater than 254 bytes and the column width is 254 or less, the data is wrapped up to and including the 254th byte, but the remainder of the data is truncated.
- If the data for a defined column is greater than 254 bytes and the column width is 255 or more, the data is wrapped at the width of the column.

When you specify Definition from FORM.COLUMNS, the following panel is displayed, where you can enter an expression (up to 50 characters) defining your new column.

```
                        Definition

   Column number :     8
   Column Heading:   Total Earnings

   Type an expression to define this column.
   Expression  [ totearn(&6 &7)                              ]
   Pass Nulls? [ YES    ]
 ____________________________________________________________
   F1=Help  F5=Previous Column  F6=Next Column
   F10=Previous Definition  F11=Next Definition  F12=Cancel
```

Figure 16. The FORM.COLUMNS Definition panel

You can define the new column in terms of:

- A character or numeric constant
- These form variables (which are described in FORM.BREAKn):
  - &*n* (where *n* is a number that indicates the column's position in the SELECT statement of the query)
  - &DATE
  - &TIME
  - &ROW
- A valid global variable
- A valid REXX expression or function
- An expression that involves any of these items

If you include a REXX expression in your column definition, you might receive unexpected results if the value returned by REXX is longer than 32,767 characters.

Use the Previous and Next function keys to move from one column definition panel to another.

PASS NULLS

If the PASS NULLS question is answered YES, you can use the values that are provided by QMF shown in the following table to change the handling of the value depending on the situation:

Table 24. Default handling of database null values

Situation	**Character string that replaces the value**
Data is null	DSQNULL
Data is undefined	DSQUNDEF
Data has numeric overflow	DSQOFLOW
Data has no instance	DSQNOINS
Data has no relationship	DSQNOREL

For example, any database variable that is null (a database null value) is replaced with the character string DSQNULL before the expression is passed to REXX for evaluation. You can provide a REXX expression or program that checks for the string and substitutes 0 (or whatever is appropriate for your purpose) for the database null value.

If a null value is returned by the REXX expression, you can pass it to your report.

If PASS NULLS is set to YES and the expression contains a substitution variable that is null, undefined, overflows, or has no instance or no relationship, then the entire expression is set to the value that represents that condition. This expression reduction is performed only on expressions, not comparisons.

If the PASS NULLS answer is NO, a null is returned for the preceding list of values. Nothing is passed to REXX for evaluation.

Related reference

FORM.BREAKn
Use the FORM.BREAK*n* panels (where *n* is a number from 1 through 6) to make choices about the text and its placement for up to six breaks in a report. QMF places the text you specify on each break panel after its associated break in the report.

SET GLOBAL
The SET GLOBAL command assigns values to global variables from the QMF command line, from a procedure, or through the callable interface. You cannot change the value of a global variable that is defined as read-only.

Printing considerations

When you print a FORM, the column definition and alignment information are printed on a page following FORM.COLUMNS instead of the `Specify Alignment` and `Specify Definition` windows that appear on your screen. The NUM field is repeated with the column definition and alignments.

The following figure shows an example:

```
1                                        FORM:
FORM.COLUMNS

        HEADING   DATA                                              PASS
 NUM    ALIGN     ALIGN      DEFINITION                              NULLS?
 ---    --------  ---------  --------------------------------------  ------
   1    DEFAULT   DEFAULT                                           NO
   2    CENTER    CENTER                                            NO
   3    DEFAULT   DEFAULT                                           NO
   4    LEFT      DEFAULT                                           NO
   5    DEFAULT   DEFAULT                                           NO
   6    DEFAULT   DEFAULT                                           NO
   7    DEFAULT   DEFAULT                                           NO
   8    RIGHT     RIGHT       &6 + &7                               NO
   9    DEFAULT   DEFAULT      (&6 + &7)  *  &5                     NO
        *** END ***

   05/05/91   11:10 AM                                  PAGE   3
```

Figure 17. Column definition and alignment information that result from printing a form

FORM.CONDITIONS

Use FORM.CONDITIONS to enter expressions for conditional formatting. Conditional formatting allows you to create expressions that determine when the formatting variations specified in FORM.DETAIL appear.

Restriction: FORM.CONDITIONS uses expressions written in REXX, which is not supported in CICS.

You can use conditional formatting to specify detail text for grouped data. The condition is evaluated using data from the first row of the group. If the condition evaluates to true, the detail text for that variation is printed. If the condition evaluates to false, the detail text for that variation is not printed for that group.

The following figure shows the entry fields on the FORM.CONDITIONS panel.

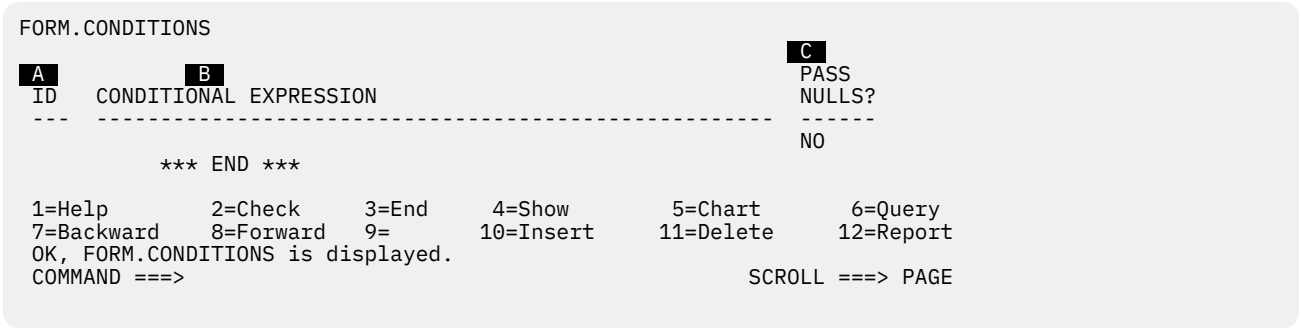


Figure 18. Entry fields on the FORM.CONDITIONS panel

A ID
Enter a one- to three-character identifier for the conditional expression. The identifier is any number from 1 through 999. When appended to the C selection code in the `Select Panel Variation?` field of the FORM.DETAIL panel, it identifies which expression in FORM.CONDITIONS determines whether the detail variation gets formatted.

B CONDITIONAL EXPRESSION
Enter a valid REXX expression. The difference between an expression in FORM.CALC and in FORM.CONDITIONS is that a condition results in a value of either true or false. An expression that evaluates to 1 is true; an expression that evaluates to anything else is false. Nonnumeric data, including blanks and nulls, are assumed to be false. You can use any valid global variables in conditional expressions. However, the only QMF form variables you can use in conditional expressions are &ROW, &DATE, &TIME, and &*n*, where *n* specifies the column's position in the SELECT statement of the query.

C PASS NULLS
Enter YES or NO.

YES
Allows you to use the values that are provided by QMF, which are shown in the following table, to change the handling of the value depending on the situation:

Table 25. Default handling of database null values

Situation	Character string that replaces the value
Data is null	DSQNULL
Data is undefined	DSQUNDEF
Data has numeric overflow	DSQOFLOW
Data has no instance	DSQNOINS
Data has no relationship	DSQNOREL

NO
Returns a null for the values listed above. Nothing is passed to REXX for evaluation.

Related concepts
Using REXX with QMF forms
Expressions used in FORM.CALC, FORM.CONDITIONS, and FORM.COLUMNS (Column Definition) can consist of terms (*strings*, *symbols*, and *functions*) interspersed with operators and parentheses. Do not execute QMF commands (using the callable or command interfaces) from within a REXX expression or program.

Related reference
FORM.DETAIL

FORM.DETAIL consists of detail variations that you define. You can create up to 99 variations, and each variation can correspond to conditions entered on FORM.CONDITIONS. Unless each condition is mutually exclusive, different detail variations can be displayed for the same data row.

FORM.DETAIL

FORM.DETAIL consists of detail variations that you define. You can create up to 99 variations, and each variation can correspond to conditions entered on FORM.CONDITIONS. Unless each condition is mutually exclusive, different detail variations can be displayed for the same data row.

Use FORM.DETAIL to:

- Specify text to precede column headings.
- Combine tabular data with text.
- Omit tabular data and show data values entirely as text.

FORM.DETAIL does not affect charts.

The following figure shows the entry fields of the FORM.DETAIL panel.

```
FORM.DETAIL                                                A  VAR   1 of  1

 B Include Column Headings with Detail Heading? ===> YES
 C LINE D ALIGN E DETAIL HEADING TEXT
 ----    ------   ----+----1----+----2----+----3----+----4----+----5----+
 1       LEFT
 2       LEFT
         *** END ***

 F New Page for Detail Block? ===> NO    G Repeat Detail Heading?   ===> NO
 H Keep Block on Page?        ===> NO    I Blank Lines After Block  ===> 0
 J Put Tabular Data at Line (Enter 1-999 or NONE) ===> 1
 K LINE L ALIGN M DETAIL BLOCK TEXT
 ----    ------   ----+----1----+----2----+----3----+----4----+----5----+
 1       LEFT
 2       LEFT

         *** END ***
 N Select Panel Variation? ===> YES

 1=Help        2=Check     3=End      4=Show        5=Chart       6=Query
 7=Backward    8=Forward   9=        10=Insert     11=Delete     12=Report
 OK, FORM.DETAIL is displayed.
 COMMAND ===>                                            SCROLL ===> PAGE
```

Figure 19. Entry fields of the FORM.DETAIL panel

A VAR 1 of 1
The first number represents the current panel variation, and the second represents the total number of variations you created (the maximum is 99). The default form displays `VAR 1 of 1`.

You can create a detail variation by entering a value one greater than the total number of variation panels over the current panel variation value. New panels must be added sequentially.

You can navigate to existing panel variations by entering the identifying value over the current panel variation value. You can also display different panel variations by entering the NEXT and PREVIOUS commands on the command line.

Sections B through E specify text to be followed in a report by column headings that are specified on FORM.COLUMNS.

B Include column headings with detail heading?

YES
Column headings become part of the detail headings. The resulting detail heading is repeated whenever requested on BREAK panels or in G *Repeat Detail Heading*.

NO
Column headings are suppressed.

C LINE
Identify lines of detail heading text and their relative positions. Any number of lines can be specified. The line numbers can be any number from 1 through 999 or blank.

If you use the same LINE value for more than one line, those lines are joined according to the ALIGN value for the additional line or lines. Lines with the same LINE value overlay each other if they are longer than the report width, or if their ALIGN values conflict.

D ALIGN
Specify where each line of detail heading text is to be placed horizontally in the report. You can place the lines anywhere within the width of the report.

Left
Left-justifies the detail heading text.

Right
Right-justifies the detail heading text.

Center
Centers the detail heading text.

n
Begins the detail heading text in the *n*th position of the line, where *n* can be any number from 1 through 999999.

Append
If APPEND is used for a line of text that is not appended to another line, the line of text is left-aligned.

The previous line of text and the appended line of text must have the same LINE value if they are to be placed on the same line. If the report is not wide enough to accommodate the appended line of text, some of the text might be truncated.

E DETAIL HEADING TEXT
Specify the detail heading text. You can add up to 999 lines of text using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

Detail heading text always precedes column headings in a report. Detail headings consist of detail heading text, column headings, or both. Unless omitted, detail heading text and column headings constitute detail headings.

By default, a detail heading can extend from the left margin to the right margin of the report. Any text that extends beyond the right margin is not displayed or printed. You can alter the width by changing the report text width on the FORM.OPTIONS panel. If you do not explicitly specify a width, the right margin is determined by the width of the tabular data.

When you print a report, all the detail headings that are selected for the current row of data when the page heading is formatted are printed. If the number of lines for the detail heading exceeds the number of available lines on the page, the excess detail heading lines are lost.

Detail headings can contain the following variable values:

Global variables
Use SET GLOBAL to set variables for use in detail heading text.

&n
The value in the *n*th column on the form that is used for this report. For example, consider the following detail heading:

```
ID NUMBER:  &1   EMPLOYEE NAME:  &2
```

This detail heading can produce the following heading in a report:

```
ID NUMBER:  50    EMPLOYEE NAME:  HANES
```

The &*n* value is the value of column *n* from the current row at the start of the new page. Detail headings for unconditionally selected variations are shown at the top of each screen in displayed reports. However, the value for &*n* appears only on the first screen of a displayed report. If you want to display the report online with page breaks, issue the DPRE command.

With this special syntax, the width of the substitution value is determined by the width that is specified by the associated column on the FORM.COLUMNS or FORM.MAIN panel.

&ROW
The number of the current data row when the detail heading is formatted.

&DATE
The date the PRINT command was executed (in printed reports) or the current date (in displayed reports).

&TIME
The time the PRINT command was executed (in printed reports) or the current time (in displayed reports).

&PAGE
The current page number.

The form variables &DATE, &TIME, and &PAGE are described in FORM.BREAKn.

Sections **F** through **M** specify report data that can be repeated in a report for each data row. This data, called a detail block, is the tabular data (if selected) and text that is associated with a single data line or a single detail line (for example, a row from a table).

F New page for detail block?
Specify whether to start each occurrence of the detail block on a new page in a printed report. A new page is started if the report is not already at the top of the page.

G Repeat detail heading?
Specify whether to repeat the detail heading before each occurrence of the detail block text. The detail heading includes any detail heading text that is specified on the FORM.DETAIL panel, followed by column headings (if not suppressed) listed on the FORM.COLUMNS panel.

NO
The detail heading is formatted at the beginning of each screen for online reports or each page for printed reports.

YES
The detail heading is formatted before each occurrence of detail block text.

H Keep block on page?
Specify whether to keep each detail block together on one page of your printed report.

NO
Allows detail blocks to be split across two or more pages of your printed report.

YES
Prevents detail blocks from being split across pages. If a detail block is too long to be printed on one page, it is started on a new page.

I Blank lines after block
Specify how many blank lines you want after the detail block text.

The detail spacing option on the FORM.OPTIONS panel also affects the number of blank lines after detail block text.

J Put tabular data at line (Enter 1-999 or NONE)
Specify whether to generate the tabular data (in the tabular format that is specified on FORM.COLUMNS or FORM.MAIN) and where to place this tabular data. The number corresponds to the number of the detail block text line on which the tabular data is placed. NONE (or N) indicates not to format the tabular data. NONE does not affect break text or aggregation values.

This option can be used to mix text with tabular data. When a number is specified, tabular data overlays or combines with any detail block text on the same line.

If NONE is specified, tabular data is not formatted, but the column values can be included in the detail block text by using column substitution values.

K LINE
Identify the lines of detail block text and specify their relative positions. Any number of tabular data lines can be specified. You can specify any number from 1 through 999 or a blank. For additional information, see **C** *LINE*.

L ALIGN
Specify where each line of detail block text is to be placed horizontally in the report. You can place the lines anywhere within the width of the report. Valid values are LEFT, RIGHT, CENTER, APPEND, or any number from 1 through 999,999.

The ALIGN values do not affect the horizontal placement of tabular data. To change the placement of tabular data, modify the column widths or indents on FORM.COLUMNS or FORM.MAIN. For more information, see **D** *ALIGN*.

M DETAIL BLOCK TEXT
Specify the detail block text. You can add up to 999 lines of detail block text using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

By default, detail block text extends from the left margin to the right margin of the report. Any text that extends beyond the right margin is not displayed or printed. You can alter the width by changing the report text width on the FORM.OPTIONS panel. If you do not specify a width, the right margin is determined by the width of the tabular data.

Detail block text can contain literal text along with the following variable values:

Global variables
Use SET GLOBAL to set variables for use in detail block text.

&n
The value in the *n*th column on the form that is used for this report. For example, consider the following detail block text:

```
DEPARTMENT:  &3   EMPLOYEE NAME:  &2
```

This detail block text produces a line like the following in the report:

```
DEPARTMENT:  20    EMPLOYEE NAME:  SANDERS
```

&COUNT
The number of rows that are displayed or printed since the last break. This value is a running count and increases from data row to data row.

&ROW
The number of the data row for the detail block is printed or displayed in your report.

In detail block text with a group summary report, the number of the data row for the last row in the group is printed.

&CALCid
Calculated value.

&DATE
The current date.

&TIME
The current time.

&PAGE
The current page number.

The form variables &DATE, &TIME, and &PAGE are described in FORM.BREAKn.

&an

n is a valid column number, and *a* is one of the following QMF aggregation functions: AVG, COUNT, CPCT, CSUM, FIRST, LAST, MAX, MIN, PCT, STDEV, SUM, TCPCT, TPCT. The values of the aggregations are based on running values within the current break level.

In detail block text, the values for aggregations are based on the data values since the last break through the current row. Calculated values such as AVG and STDEV are also based on data values since the last break. For example, &AVG6 is the sum of column six (through the current row) divided by COUNT.

At the detail level, &SUM and &CSUM produce the same result. &SUM6 and &CSUM6 in the detail block text each produces the total value of column 6 through the current row.

If you use a percent aggregation variable (PCT, TPCT, or TCPCT) in detail block text, and if you associate it with a column that has a D edit code, QMF formats the percent value in the detail block text as if it had an L edit code. Likewise, if you use the STDEV aggregation variable in detail block text and associate it with a column that has a P or a D edit code, QMF formats the standard deviation in the detail block text as if it had an L edit code.

N Select panel variation

Specify when to select a panel variation. You must enter one of the following allowable values; blanks are not allowed:

YES

Always selected for formatting in the report. It is the default when the variation number is 1.

NO

Never selected for formatting. It is the default when the variation number is from 2 through 99. This value can be used to temporarily inhibit the formatting of a variation in a report.

The following two choices allow you to selectively format your report. You can associate an entire panel of detail text and formatting options with a specific condition on the FORM.CONDITIONS panel (conditional formatting), or a specific data column that corresponds to a branch of tree data.

C1-C999

Can be selected to identify a condition on FORM.CONDITIONS. If the condition is true, the associated FORM.DETAIL variation is formatted.

E1-E999

Can be selected for formatting when data exists for the indicated column. The column is identified by the number that follows E. This number corresponds to the NUM value for a column on FORM.MAIN or FORM.COLUMNS.

Related concepts

Variables used in forms

You can use global variables (both those defined by users and those supplied by QMF) and form variables in QMF forms. A variable can replace a string of text or a numeric value. You can assign different values to the variable to produce different reports without changing the form.

Related reference

Edit codes for numeric data

You can use several edit codes to format numeric data.

FORM.BREAKn

Use the FORM.BREAK*n* panels (where *n* is a number from 1 through 6) to make choices about the text and its placement for up to six breaks in a report. QMF places the text you specify on each break panel after its associated break in the report.

FORM.CALC

On the FORM.CALC panel you can enter expressions for report calculations. This panel initially contains only one row – a place for one expression. However, up to 998 additional rows can be inserted.

DPRE

DPRE is a command synonym that provides a print preview so that you can see how a report will appear when it is printed.

NEXT
Use the NEXT command to navigate forward through the set of variations associated with the FORM.DETAIL panel. You can also use the NEXT command to display the next column or the next definition from the Column Definition or Column Alignment panel, or to display the next row in the set of accessed rows in the Table Editor.

SET GLOBAL
The SET GLOBAL command assigns values to global variables from the QMF command line, from a procedure, or through the callable interface. You cannot change the value of a global variable that is defined as read-only.

FORM.FINAL

Use FORM.FINAL to make detailed choices about the content and placement of final text in a report. QMF places the text at the end of the report, and you can use it, for example, to identify the final summary data of a report.

Area **H** on FORM.MAIN specifies the final text for a report. Whatever you specify in this area of FORM.MAIN is reflected on FORM.FINAL. Similarly, the first line of final text is reflected on FORM.MAIN.

The following figure shows the entry fields on the FORM.FINAL panel.

```
FORM.FINAL

 A New Page for Final Text?===> NO     B Put Final Summary at Line ===> 1
 C Blank Lines Before Text ===> 0
 D LINE E ALIGN F FINAL TEXT
  ----    ------   ----+----1----+----2----+----3----+----4----+----5----+
  1       RIGHT
  2       RIGHT
  3       RIGHT

          *** END ***

 1=Help         2=Check     3=End      4=Show         5=Chart       6=Query
 7=Backward     8=Forward   9=        10=Insert      11=Delete     12=Report
 OK, FORM.FINAL is displayed.
 COMMAND ===>                                              SCROLL ===> PAGE
```

Figure 20. Entry fields on the FORM.FINAL panel

A New page for final text?

Reports
Specify whether to place the final text on a page separate from the body in a printed report. A new page is started if the report is not already at the top of the page.

B Put final summary at line

Reports
Specify whether to generate the final summary of a report and, if so, where to place it in relation to the final text. The value for this entry can be any number from 1 through 999 or the word NONE. The number is the number of the line of final text next to which you want to place the final summary. NONE (or N) omits the final summary.

If you expect the final summary value of a wrapped column to be greater than one line long, include final text on the line corresponding to the last line you expect for your wrapped final summary value. Including final text is only necessary if the wrapped column has a usage code of MAX, MIN, FIRST, or LAST.

For example, if the column NAME (from Q.STAFF) is set to a width of 2, has an edit code of CW, and a usage code of MAX, you must place some final text (perhaps a period) on the fifth line of FORM.FINAL to see the entire final summary value for that column (YAMAGUCHI).

Two data lines per summary in an across report can appear only if the across summary column and final summary are both present. This occurs when a column in the form has a usage of CSUM, CPCT, PCT, TPCT, or TCPCT.

When the across summary column is omitted on FORM.OPTIONS, the ACROSS-across values are also omitted and only one line is formatted per group (with ACROSS-down values).

When the final summary is omitted on FORM.FINAL, the ACROSS-down values are omitted and only one line is formatted per group (with the ACROSS-across values).

Charts

When there are two summary lines, but only one is charted by the Interactive Chart Utility (ICU), the second summary data line contains values only in columns for which PCT, CPCT, or CSUM is specified. In these columns:

- The value in the first line is the summary value for that category relative to the ACROSS-across (group) total.
- The value in the second line is the summary value for that category relative to the ACROSS-down (category) total.

C Blank lines before text

Reports

Specify the number of blank lines between the body of the report and the first line of final text. The value for this entry can be any number from 1 through 999 or the word BOTTOM. The default is 0.

For example, if you want one blank line between the body of the report and the first line of final text, type 1 in this field. If you want the final text to be separated from the body by two blank lines, type 2 in this field.

If you want the final text to be displayed at the bottom of the current page (regardless of where the body of the report ends) type BOTTOM (or B) in this field.

D LINE

Reports

Identify the lines of final text and specify their positions relative to themselves and to the line at which the final text starts (as indicated in the `Blank Lines Before Text` field).

The numbers that you choose need not start with 1 or be consecutive. You can choose spacing between the lines of the final text and between the body of the report and the first line of final text. For example, if you have three lines of final text, and you choose LINE values of 1, 3, and 5 for the text, QMF starts the final text at the line you indicated in the `Blank Lines Before Text` field and places one blank line between lines of text. If you do not use 1 as one of your LINE values, QMF does not begin the final text at the line you specified in the `Blank Lines Before Text` field. It leaves extra blank lines, up to the first specified line number. A blank LINE value tells QMF to ignore any associated text.

For example, consider the following values on FORM.FINAL:

```
LINE  ALIGN  FINAL TEXT
----  -----  -----------------
2     LEFT   GRAND TOTALS FOR
3     LEFT   ALL DEPARTMENTS
```

These values display on the resulting report as shown in the following figure:

```
GRAND TOTALS FOR
ALL DEPARTMENTS
```

Figure 21. Adding a blank line before final text in a report

Notice that a blank line appears before the first line of text.

In the example, if you indicated a value of 0 in the `Blank Lines Before Text` field, you might expect the text `GRAND TOTALS FOR` on the line immediately following the body of the report. But because the first line of text has a LINE value of 2, QMF skips one blank line (for the missing first line of the final text), and then prints the first line from FORM.FINAL on the second line of the final text in the report.

If you use the same LINE value for more than one line, those lines are joined according to the ALIGN value for the additional line or lines. Lines with the same LINE value overlay each other if their ALIGN values are the same or otherwise conflict. For example, you can specify the same LINE value for two lines of final text, with an ALIGN value of LEFT for the first line and an ALIGN value of CENTER for the second line. If the text on the first line extends past the center of the report, the second line overlays part of the first line.

E ALIGN

Reports

Specify where each line of final text is placed horizontally in a report. If a report contains final summary data, the line length for the final text is from the left margin to the beginning of the summary data.

However, if a report does not contain final summary data, the line length for the final text is the complete length of the line (from the left to the right margin). For an online report, the line length is the width of the displayed report; for a printed report, the line length is the width of the printed report.

Left
Left-justifies the line of final text.

Right
Right-justifies the line of final text. This setting is the default.

Center
Centers the line of final text.

n
Begins the line of final text in the *n*th position of the line, where *n* can be any number from 1 through 999999.

Append
Positions the line at the end of the previous line of final text. If APPEND is used on the first line of final text (that is, on the line of text with the lowest LINE value), the line of text is left-aligned.

The appended line of text must have the same LINE value as the line of text it is being appended to. If the report is not wide enough to accommodate the appended line of text, some of the text might be truncated.

For example, consider the following entries on FORM.FINAL:

```
Blank Lines Before Text ===> 0
LINE  ALIGN   FINAL TEXT
----  ------  -----------------
1     RIGHT   TOTAL
1     APPEND   SALARIES
3     RIGHT
```

These values produce a report like this:

```
       DEPT        COMM  JOB        SALARY
      ------  ----------  -----  ----------
         66       55.50  CLERK    10988.00
                         .
                         .
                         .
                 1285.00  SALES    17844.00
                                  ----------
                              *    66147.00
```

```
                                ==========
                  TOTAL SALARIES   152223.20
```

F FINAL TEXT

Reports

You can add up to 999 lines of final text using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value or by specifying a specific horizontal position.

By default, final text extends from the left margin of a report to the beginning of the summary data (if a report has summary data) or to the right margin of a report. However, you can specifically choose the width of final text by changing the `Report text line width` entry on FORM.OPTIONS.

To make the final text appear in a report in uppercase and lowercase, specify a CASE value of either STRING or MIXED in your profile.

Final text can contain the following variable values:

Global variables
Use SET GLOBAL to set variables for use in final text.

&n
The last value in the *n*th column on the form that is used for this report.

&COUNT
The number of rows that are displayed or printed since the last break. This value is a running count and increases from data row to data row.

&ROW
The number of the last data row of the entire report is printed or displayed in your report.

&CALCid
Calculated value.

&DATE
The current date.

&TIME
The current time.

&PAGE
The current page number.

&an
n is a valid column number, and *a* is one of the following QMF aggregation functions: AVG, COUNT, CPCT, CSUM, FIRST, LAST, MAX, MIN, PCT, STDEV, SUM, TCPCT, TPCT. The values of the aggregations are based on running values within the current break level.

If you use a percent aggregation variable (PCT, TPCT, or TCPCT) in detail block text, and if you associate it with a column that has a D edit code, QMF formats the percent value in the detail block text as if it had an L edit code. Likewise, if you use the STDEV aggregation variable in detail block text and associate it with a column that has a P or a D edit code, QMF formats the standard deviation in the detail block text as if it had an L edit code.

Related reference

Edit codes for numeric data
You can use several edit codes to format numeric data.

FORM.BREAKn
Use the FORM.BREAK*n* panels (where *n* is a number from 1 through 6) to make choices about the text and its placement for up to six breaks in a report. QMF places the text you specify on each break panel after its associated break in the report.

FORM.MAIN
Use FORM.MAIN to make simple changes to a report or chart.

FORM.OPTIONS

Use FORM.OPTIONS to adjust the appearance of your report.

SET GLOBAL
The SET GLOBAL command assigns values to global variables from the QMF command line, from a procedure, or through the callable interface. You cannot change the value of a global variable that is defined as read-only.

SET PROFILE
The SET PROFILE command changes values in your QMF profile. These values influence the behavior of your QMF session.

FORM.OPTIONS

Use FORM.OPTIONS to adjust the appearance of your report.

Area **J** on FORM.MAIN (OUTLINE and DEFAULT BREAK TEXT) specifies two options that affect the overall appearance of a report. What you specify in that area of FORM.MAIN is reflected on FORM.OPTIONS. Similarly, some of what you specify on FORM.OPTIONS is reflected on FORM.MAIN.

The following figure shows the entry fields on the FORM.OPTIONS panel.

```
FORM.OPTIONS

    What do you want for
 A  Detail spacing?                              ===> 1
 B  Line wrapping width?                         ===> NONE
 C  Report text line width?                      ===> DEFAULT
 D  Number of fixed columns in report?           ===> NONE

    Do you want
 E  Outlining for break columns?                 ===> YES
 F  Default break text (*)?                      ===> YES
 G  Function name in column heading when grouping? ===> YES
 H  Column wrapped lines kept on a page?         ===> YES
 I  Across summary column?                       ===> YES
 J  Automatic reordering of report columns?      ===> NO
 K  Page renumbering at the highest break level? ===> NO

    Do you want separators for
 L  Column heading? ===> YES      M  Break summary? ===> YES
 N  Across heading? ===> YES      O  Final summary? ===> YES

 1=Help        2=Check     3=End     4=Show        5=Chart       6=Query
 7=            8=          9=       10=           11=           12=Report
 OK, FORM.OPTIONS is displayed.
 COMMAND ===>                                          SCROLL ===> PAGE
```

Figure 22. Entry fields on the FORM.OPTIONS panel

A Detail spacing?

Reports
Select spacing between tabular data lines or detail blocks. The spacing within the detail block text is not affected. The value can be any number from 1 through 999. The default is single spacing with no blank line between each block of text.

The `Blank Lines after Block` option on the FORM.DETAIL panel also affects the spacing between detail blocks.

B Line wrapping width?

Reports
Specify whether the columns in a report are to be wrapped and, if so, at what width. The value for this entry can be any number from 1 through 999 or the word NONE. The default is NONE, indicating that the lines in a report are not to be wrapped.

Lines cannot be wrapped in ACROSS reports or reports with column wrapping. Detail heading text and detail block text are not wrapped. They are truncated at the report text line width. However,

if the value for report text width is DEFAULT, and the line-wrapping width is not NONE, the detail heading text and detail block text are truncated at the line-wrapping width.

If the value in this entry area is greater than the print width, the data in the columns of a report is truncated on the right.

If you want line wrapping (that is, the detail lines in a report begin on one line and continue on one or more subsequent lines), type a number in this entry area to indicate the maximum width of the lines of data you want in the report. As many whole columns as possible are positioned across the report. Any remaining columns are placed on one or more subsequent lines of the report. All wrapped lines begin with the column indent, then include the tabular data.

If a column and its indent are too wide to fit within the line-wrapping width specified, a new line does not begin for the column and the column is cut off on the right.

Only column headings, tabular data, and column summaries are wrapped when you specify a width. All other data in the report is formatted as usual.

The following figure shows part of a report with line wrapping (at a width of 35) and tabular data line spacing of 2.

```
    ID  NAME         DEPT  JOB
------  ---------  ------  -----
 YEARS      SALARY        COMM
------  ----------  ----------
   160  MOLINARE       10  MGR
     7    22959.20           -

   210  LU             10  MGR
    10    20010.00           -

   240  DANIELS        10  MGR
     5    19260.25           -
```

Figure 23. Line wrapping in a report

C Report text line width?

Reports
Specify the width of the final text, detail heading text, detail block text, and break text in a report. The values in this entry area can be DEFAULT, COLUMNS, or any number from 1 through 999999.

DEFAULT
Break footing text and final footing text use the full width of all columns up to the first summary column, as indicated in FORM.COLUMNS and FORM.MAIN.

COLUMNS
All text areas use the full width of all columns as indicated in FORM.COLUMNS and FORM.MAIN. (This option is the same as DEFAULT for detail heading text and detail block text.)

A number from 0 through 999999
The width in characters for all text types. 0 indicates that no text is formatted.

D Number of fixed columns in report?

Reports
Specify the number of columns that remain in place when you scroll reports horizontally on the screen. When fixed columns are specified, the report is divided into a fixed area and a scrollable area. For printed reports of more than one page, fixed columns are repeated on the left side of each page. The scrollable area of a printed report refers to the area that changes during page splitting.

The value can be any number from 1 through 999 or NONE (the default).

If the number specified is greater than the number of columns in the report, all columns are fixed. Columns with OMIT usages are not counted as fixed columns.

Fixed columns can be used with column reordering (SEQ). If the columns were reordered and you select a number of columns (*n*) as fixed columns, the first *n* columns of the new order are the fixed columns. This applies to automatic reordering and user reordering.

The fixed-column area of a report can affect the text of the report. The portions of break, detail, and final text that are within the fixed area are repeated on the left side of any printed pages of the report. The portions of break, detail, and final text that are within the scrollable area appear on the first page of a printed report, but do not appear on subsequent pages when page splitting occurs.

Page heading and footing text are not affected by fixed column settings in either displayed or printed reports.

Fixed columns can conflict with other report options. You cannot use line wrapping with fixed columns (see **B** *Line wrapping width?* earlier in this topic). Also, if the total width of all fixed columns in a report is greater than the displayable screen width, both the displayed and printed versions of the report are affected. For displayed reports, you can scroll the report up and down, but you cannot scroll it to the left or right. For printed reports, this message is displayed:

```
The report cannot be printed; the fixed area is too wide.
```

E Outlining for break columns?

Reports

If you assigned a usage code of BREAK to one of your columns, use this entry area to determine whether the value in the BREAK column is to be displayed only when the value changes or on every line in a report.

YES

Displays the value in the BREAK column only when the value changes.

NO

Displays the value in the BREAK column on every tabular data line in the report.

Outlining begins at the top of a page. The value is printed at the top of a page even if it has not changed from the bottom line of the previous page.

F Default break text (*)?

Reports

If a report contains breaks for which you did not indicate break footing text, use this entry area to specify whether to generate break footing text to mark the BREAK aggregation line.

The default break text consists of one asterisk for the highest-numbered break level text, two asterisks for the next-highest numbered break level text, and so on.

G Function name in column heading when grouping?

Reports

If a report has combined data (for example, as a result of summing a column) and you use the usage code GROUP to suppress the tabular data lines, this entry area determines the heading of the aggregated column.

YES

Displays a word indicating the type of aggregation as part of the column heading.

NO

Suppresses the aggregation name in the column heading.

Charts

If you use YES for charts, the function name appears in the legend on the chart. NO is recommended.

H Column wrapped lines kept on a page?

Reports

If you specified column wrapping for one or more columns in a report, this entry area determines whether the wrapped columns can be split between two pages.

YES
Keeps the column-wrapped lines on the same page unless the wrapped column is longer than the page depth.

NO
Allows wrapped columns to be split between pages if necessary.

Across summary column?

Reports

Specify whether to display the automatically generated "across summary" column. The `Across summary column` field produces additional columns that show totals across the specified columns.

In the ACROSS report shown in the following figure, you can read the lines for Departments 10 through 84 across to see the average salary for each job and the department average in the last column. The job salary averages are under the final summary separators at the bottom of each column.

```
            <---------------- JOB ----------------->
            <- CLERK -->  <-- MGR --->  <- SALES -->  <- TOTAL -->
              AVERAGE       AVERAGE       AVERAGE       AVERAGE
  DEPT        SALARY        SALARY        SALARY        SALARY
------    ----------    ----------    ----------    ----------
    10                    20865.86                    20865.86
    15      12383.35      20659.80      16502.83      15482.33
    20      13878.68      18357.50      18171.25      16071.53
    38      12482.25      17506.75      17407.15      15457.11
    42      11007.25      18352.80      18001.75      14592.26
    51      13914.90      21150.00      18555.50      17218.16
    66      10988.00      18555.50      18844.23      17215.24
    84      13030.50      19818.00      16649.25      16536.75
          ==========    ==========    ==========    ==========
            12612.61      19805.80      17869.36      16675.64
```

Figure 24. A report showing averages across the columns

The across summary column is displayed to the right of the columns in a report.

It is possible to get two data lines per summary in any across report for which at least one column has a usage of PCT, CPCT, or CSUM. However, this only happens if the across summary column and final summary are both present or both absent in the report.

When two data lines per summary are returned, the second summary data line contains values only in those columns for which PCT, CPCT, or CSUM is specified. In such columns, the value in the first line is the summary value for that subcategory relative to the ACROSS-across (group) total. The value in the second line is the summary value for that subcategory relative to the ACROSS-down (subcategory) total.

When the across summary column is omitted (on FORM.OPTIONS), the ACROSS-across values are also omitted and only one line is formatted per group (with the one line containing the ACROSS-down values).

When the final summary is omitted (on FORM.FINAL), the ACROSS-down values are omitted and only one line is formatted per group (with the one line containing the ACROSS-across values).

Charts

Only one of the two possible "across" summary lines of data can be transferred to the ICU. Charts cannot display both lines of data. If two values exist for a column in each group, the value on the second line (ACROSS-down) is the value that is passed to the ICU and shows on the chart.

You can force the ACROSS-across values to be charted if the final summary is omitted. This causes the ACROSS-down values to be omitted.

J Automatic reordering of report columns?

Reports

Specify whether the columns in a report are automatically reordered when you specify a usage of BREAK*n*, GROUP, or one of the aggregating functions (such as AVERAGE, COUNT, FIRST, LAST, MAXIMUM, MINIMUM, STDEV, SUM, CPCT, CSUM, PCT, TPCT, or TCPCT).

The default is NO. (The columns are not automatically reordered; they appear in the report in the order in which they are shown on FORM.MAIN or FORM.COLUMNS, even if you use a usage code of BREAK*n*, GROUP, or one of the aggregating functions.)

If you specify YES, the columns are reordered according to the following rules:

- BREAK*n* columns to the far left
- GROUP columns to the left after BREAK*n* columns
- All nonaggregated columns to the left after BREAK*n* and GROUP columns
- All aggregated columns to the far right

If you use ACROSS as a usage, the value in this entry area is ignored because the purpose of an ACROSS report is defeated if the columns cannot be reordered.

Charts

If automatic reordering of report columns is set to YES, it can have an effect on which Y-data column is selected for the X axis in a chart. The following conditions must be met for automatic column reordering to have an effect:

- No GROUP or BREAK*n* usage codes are used on the form to select Y data columns for the X axis of the chart.
- An aggregation function (such as AVERAGE, SUM, or COUNT) is used on the form with one of the columns.

If these conditions are met, the aggregated columns are moved from the left side of the report to the far right. For example, suppose that YEARS originally appeared on the left side of your report; therefore, the YEARS column was plotted on the X axis when you displayed your chart. (You did not specify GROUP or BREAK to select data columns for the X axis.) Additionally, suppose you decide to use the aggregation function of AVERAGE with YEARS; the YEARS column now moves to the far right of the report. Because it is no longer the left-most column, it is not plotted on the X axis of your chart. The column that now appears at the left of your report is plotted on the X axis.

K Page renumbering at the highest break level?

Reports

Specify whether a printed report begins a new page beginning with the number 1 whenever the value in the control column with the highest break level changes. The highest break level is the one with the lowest number. This option affects only printed reports, because QMF treats online reports as one long page.

Use the default for this option (NO) to indicate that you do not want to restart the page numbering of the report whenever the value in the highest-level break column changes; enter YES in this entry area to start page renumbering. If you indicate YES, that value is ignored unless you use at least one BREAK usage on the form and enter YES in the `New Page for Break` entry area on the corresponding FORM.BREAK*n* panel.

L Column heading?

Reports

Specify whether the dashed lines that separate the column headings from the tabular data lines in the report are to be displayed.

M Break summary?

Reports

Specify whether the equal signs that separate the break summary from the break member lines are to be displayed.

N Across heading?

Reports
Specify whether the dashed lines and arrows that mark columns in across reports are to be displayed.

O Final summary?

Reports
Specify whether the equal signs that separate the final summary from the body of the report are to be displayed.

Related reference

FORM.DETAIL
FORM.DETAIL consists of detail variations that you define. You can create up to 99 variations, and each variation can correspond to conditions entered on FORM.CONDITIONS. Unless each condition is mutually exclusive, different detail variations can be displayed for the same data row.

FORM.MAIN
Use FORM.MAIN to make simple changes to a report or chart.

FORM.PAGE

Use FORM.PAGE to make detailed choices about the content and placement of the page headings and footings in a report.

For both online and printed reports, QMF places headings at the top of an online report and footings at the bottom. Headings and footings appear at the top and bottom of each page of a printed report.

Area G on the FORM.MAIN panel specifies page headings and footings for a report. Whatever you specify in area G of FORM.MAIN is shown on FORM.PAGE. Similarly, the first line of the page heading and footing that you specify on FORM.PAGE is shown on FORM.MAIN.

The following figure shows the entry fields on the FORM.PAGE panel.

```
FORM.PAGE

 A Blank Lines Before Heading ===> 0      B Blank Lines After Heading ===> 2
 C LINE D ALIGN E PAGE HEADING TEXT
     ----   ------   ----+----1----+----2----+----3----+----4----+----5----+
     1      CENTER
     2      CENTER
     3      CENTER
     4      CENTER

 F Blank Lines Before Footing ===> 2      G Blank Lines After Footing ===> 0
 H LINE I ALIGN J PAGE FOOTING TEXT
     ----   ------   ----+----1----+----2----+----3----+----4----+----5----+
     1      CENTER
     2      CENTER
     3      CENTER
     4      CENTER
            *** END ***

 1=Help        2=Check     3=End     4=Show        5=Chart       6=Query
 7=Backward    8=Forward   9=       10=Insert     11=Delete     12=Report
 OK, FORM.PAGE is displayed.
 COMMAND ===>                                          SCROLL ===> PAGE
```

Figure 25. Entry fields on the FORM.PAGE panel

A Blank lines before heading

Reports
Specify the number of blank lines between the top of a page and the first line of the page heading. The value can be any number from 1 through 999.

Charts
An entry in this area determines vertical placement of the heading on the chart. However, too many blank lines can change the labels on the Y axis.

B Blank lines after heading

Reports
Specify the number of blank lines between the last line of the page heading and the body of the report. The value can be any number from 1 through 999. The default is 2.

C LINE

Reports
Identify the lines of page heading text and specify their positions relative to themselves and to the line at which the page heading starts (as indicated in the `Blank Lines Before Heading` entry area).

The numbers that you choose need not start with 1 or be consecutive. You can choose spacing between the lines of the page heading and between the top of the page and the first line of page heading text. A blank ignores any associated text.

For example, consider the following values on FORM.PAGE:

```
LINE  ALIGN   PAGE HEADING TEXT
----  ------  ----+----1----+----2----
4     LEFT    MONTHLY INVENTORY
4     RIGHT   PAGE &PAGE
2     CENTER  ABC COMPANY
```

These values display as follows in the resulting report:

```
            ABC COMPANY

 MONTHLY INVENTORY       PAGE 1
```

Charts
Use LINE to position the lines of heading text vertically relative to themselves and to the line at which the chart (page) heading starts.

D ALIGN

Reports
Specify where each line of the page heading text is placed horizontally in the report. You can place the lines anywhere in the width of the report. For an online report, the width is the width of the displayed report; for a printed report, the width is the page width.

Left
Left-justifies the line of page heading text.

Right
Right-justifies the line of page heading text.

Center
Centers the line of page heading text.

n
Begins the line of page heading text in the *n*th position of the line, where *n* can be any number from 1 through 999999.

Append
Positions the line at the end of the previous line of page heading text. If APPEND is used on the first line of page heading text, the line of text is left-aligned.

The appended line of text must have the same LINE value as the line of text it is being appended to. If the report is not wide enough to accommodate the appended line of text, some of the text might be truncated. For example, consider the following entries on FORM.PAGE:

```
LINE  ALIGN    PAGE HEADING TEXT
----  ------  ----+----1----+----2----
1     CENTER   ABC COMPANY MANAGERS --
1     APPEND    &DATE, &TIME
3     CENTER
4     CENTER
5     CENTER
```

These entries align the columns as shown in the following figure:

```
                ABC COMPANY MANAGERS -- 98/08/04, 14:20


    ID  NAME         DEPT  JOB     YEARS      SALARY        COMM
------  ---------  ------  -----  ------  ----------  ----------
    10  SANDERS        20  MGR         7    18357.50           -
    30  MARENGHI       38  MGR         5    17506.75           -
```

Figure 26. Appending one line to another in a report

Charts

ALIGN does not affect a chart heading, except when LINE is used to place more than one line of text on the same line of the heading.

E PAGE HEADING TEXT

Reports

Enter the text that you want to appear either at the top of each page of a printed report or before the first line of a displayed report. You can add up to 999 lines of page heading text using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

To make the page heading text appear in a report in uppercase and lowercase, specify in your PROFILE a CASE value of either STRING or MIXED.

Page headings can contain the following variable values:

&*n*
n is a number that stands for the first value in column *n* on the current page of the report. Column *n* is the *n*th column that is selected from the database, or the *n*th column that is listed on FORM.MAIN and FORM.COLUMNS.

&ROW
The number of the first data row on the current page is printed or displayed in your report.

&DATE
The current date.

&TIME
The current time.

&PAGE
The current page number.

When &DATE, &TIME, or &PAGE are entered in page heading text, the system date, time, or page number do not appear at the bottom of printed reports. This applies only to these three variables that are entered on FORM.PAGE.

Charts

This information about PAGE HEADING TEXT also applies to charts, except for the description of ALIGN. The only time that the value specified for ALIGN affects a chart heading is when LINE is used to place one or more lines of text that is entered on FORM.PAGE on the same line in

the formatted report. If you are not using the LINE function, the chart heading is automatically centered.

F Blank lines before footing

Reports

Specify the number of blank lines between the body of the report and the first line of page footing text. The value for this entry can be any number from 1 through 999. The default is 2.

G Blank lines after footing

Reports

Specify the number of blank lines between the last line of page footing text and the bottom of the page. The value for this entry can be any number from 1 through 999.

If a report contains break summary data and one or more wrapped columns, you might need to increase the value in this entry area to see all the lines of summary data. The CW edit code wraps data in columns.

H LINE

Reports

Identify the lines of page footing text and specify their positions relative to themselves and to the line at which the page footing starts (as indicated in the `Blank Lines Before Footing` entry area). You can specify any number from 1 through 999 or a blank.

For example, consider the following values on FORM.PAGE:

```
LINE  ALIGN   PAGE FOOTING TEXT
----  ------  ----+----1----+----2----
3     LEFT    MONTHLY INVENTORY
3     RIGHT   PAGE &PAGE
2     LEFT    ABC COMPANY
```

These values display as follows in the resulting report:

```
ABC COMPANY
MONTHLY INVENTORY                          PAGE 1
```

Notice that a blank line appears before the first line of text.

I ALIGN

Reports

Specify where each line of the page footing text is to be placed horizontally in the report. You can place the lines of text anywhere between the left and right margin. For an online report, the width is the width of the displayed report; for a printed report, the width is the page width.

Left
Left-justifies the line of page footing text.

Right
Right-justifies the line of page footing text.

Center
Centers the line of page footing text.

n
Begins the line of page footing text in the *n*th position of the line, where *n* can be any number from 1 through 999999.

Append
Positions the line at the end of the previous line of page footing text. If APPEND is used on the first line of page footing text (the line of text with the lowest LINE value), the line of text is left-aligned.

The appended line of text must have the same LINE value as the line of text it is being appended to.

For example, consider the following entries on FORM.PAGE:

```
LINE  ALIGN   PAGE FOOTING TEXT
----  ------  ----+----1----+----2----
1     CENTER  ABC COMPANY MANAGERS --
1     APPEND   &DATE, &TIME
```

These changes align the columns:

```
    ID  NAME         DEPT  JOB     YEARS      SALARY        COMM
------  ---------  ------  -----  ------  ----------  ----------
                              .
                              .
                              .

    10  SANDERS        20  MGR         7    18357.50           -
    30  MARENGHI       38  MGR         5    17506.75           -


                  ABC COMPANY MANAGERS -- 98/08/04, 16:20
```

If the report is not wide enough to accommodate the appended line of text, some of the text might be truncated.

J PAGE FOOTING TEXT

Reports

Enter the text that you want to appear either at the bottom of each page of a printed report or before the last line of a displayed report. You can add up to 999 lines of page footing text using the INSERT command. Each line of text can be up to 55 characters long. You can add text to the line by using APPEND as the ALIGN value, or by specifying a specific horizontal position.

To make the page footing text appear in a report in uppercase and lowercase, specify in your profile a CASE value of either STRING or MIXED.

Page footings can contain the following variable values:

Global variables
Use SET GLOBAL to set variables for use in page footing text.

&*n*
n is a number that represents the last row in column *n* processed for the current page of this report. Column *n* is the *n*th column that is selected from the database, or the *n*th column that is listed on FORM.MAIN and FORM.COLUMNS.

&ROW
The number of the last data row on the current page is printed or displayed in your report.

&DATE
The current date.

&TIME
The current time.

&PAGE
The current page number.

When &DATE, &TIME, or &PAGE are entered in page footing text, they appear (instead of the system date, time, or page number) at the bottom of the printed report. This applies only to these three variables that are entered on FORM.PAGE.

Related reference

Edit codes for numeric data
You can use several edit codes to format numeric data.

FORM.BREAKn

Use the FORM.BREAK*n* panels (where *n* is a number from 1 through 6) to make choices about the text and its placement for up to six breaks in a report. QMF places the text you specify on each break panel after its associated break in the report.

FORM.MAIN
Use FORM.MAIN to make simple changes to a report or chart.

SET GLOBAL
The SET GLOBAL command assigns values to global variables from the QMF command line, from a procedure, or through the callable interface. You cannot change the value of a global variable that is defined as read-only.

SET PROFILE
The SET PROFILE command changes values in your QMF profile. These values influence the behavior of your QMF session.

How QMF evaluates forms for errors

QMF distinguishes between two types of errors on form panels.

- *Error conditions* – errors that require correction before the form can be used
- *Warning conditions* – errors that do not require correction before the form can be used

Error conditions

An error condition results from entering an invalid value in an entry area. For example, typing Y0 in the OUTLINE field on FORM.OPTIONS results in an error because Y0 is not an allowed value for the entry area.

An error can also occur if there is a conflict that prevents the report from being displayed. For example, SUM is a valid entry for USAGE on a numeric column. However, SUM produces an error if entered for a column with character data.

You must correct errors before the report can be displayed using the form. However, you can save, import, export, display, and print forms that contain errors.

After you correct errors, QMF identifies any warning conditions.

Warning conditions

A warning condition results when the values in two or more entry areas conflict. Unlike an error, a warning condition need not be corrected before you use the form. Instead, QMF warns you of the conflict and interprets the condition to format the report or chart.

You can either accept the report or chart as-is, or change one or more of the conflicting entries to correct the form.

The following table lists some common warning conditions and how QMF formats the report in each case. These warning conditions can also affect the chart that is created from the report.

Table 26. Warning conditions that indicate formatting problems

Condition	**QMF action**
More than one ACROSS usage	Accepts first ACROSS; omits remaining ACROSS columns from report
ACROSS usage without GROUP usage	Omits ACROSS column from report
GROUP usage without aggregating usage	Omits GROUP column from report
ACROSS and GROUP usages with one or more blank usages	If aggregation used, omits columns with blank usages from report; otherwise, omits ACROSS and GROUP columns from report

Table 26. Warning conditions that indicate formatting problems (continued)	
Condition	**QMF action**
GROUP usage with at least one aggregation usage and one or more blank usages	Omits columns with blank usages from report
Line wrapping with ACROSS usage or with column wrapping edit code	Ignores line wrapping
ACROSS usage without automatic column reordering	Ignores value of column reordering option; produces standard ACROSS report

Checking for and correcting errors

Normally, pressing Enter while displaying a form panel positions the cursor on the command line. However, if you press Enter immediately after entering one or more erroneous values on a form, QMF highlights any errors and displays a message describing the first one. Pressing Enter does not identify any errors made during a previous interaction.

If you press Enter again (with or without correcting the first error), QMF positions the cursor on the command line. To display a message about the next error in the form, use the CHECK command.

QMF checks a form for errors whenever you issue a command that uses a form (for example, DISPLAY REPORT, PRINT CHART, PRINT REPORT, EXPORT REPORT, EXPORT CHART, or RUN QUERY with the FORM option). You can issue the command either by entering it on the command line or by using a function key. QMF also checks for errors when you display the form.

If a form contains an expression with an error, this error is not detected until QMF passes the values to REXX for evaluation. If you enter a QMF command (other than CHECK, DISPLAY REPORT, DISPLAY CHART, PRINT REPORT, PRINT CHART, or RUN QUERY with the FORM option) while displaying a form, QMF processes your command whether or not the form contains errors. The displayed message pertains to the command you entered. Therefore, you can display, save, import, or export a form even if the form contains errors or warning conditions.

Related reference

CHECK
The CHECK command checks form panels for errors and conflicting entries.

Form and data incompatibility

There might be times when you modify a form in such a way that the form is inconsistent with the data. This situation is treated differently from error and warning conditions.

In this situation, there is no error message at the top of the screen and the CHECK command does not identify the problem. Instead, when you try to display the report, a message is displayed and the form panel containing the incompatibility is displayed.

To avoid incompatibilities, follow these guidelines:

- The number of columns in the form (excluding defined columns) and in the data must be equal.
- Edit codes in the form must match the data type for each column in the data.
- Every LONG VARCHAR and LONG VARGRAPHIC column must have a blank or OMIT usage code in the form.

Using REXX with QMF forms

Expressions used in FORM.CALC, FORM.CONDITIONS, and FORM.COLUMNS (Column Definition) can consist of terms (*strings*, *symbols*, and *functions*) interspersed with operators and parentheses. Do not

execute QMF commands (using the callable or command interfaces) from within a REXX expression or program.

Restriction: FORM.CALC, FORM.CONDITIONS, and FORM.COLUMNS (Column Definition) use expressions written in REXX, which QMF does not support in CICS.

Strings
Literal constants enclosed in single or double quotation marks. For example, `'High'` and `"Low"`.

Symbols
Numeric literals (numbers), variables, or nonnumeric literals without quotation marks:

- *Numeric literals* can be expressed in integer, decimal, or exponential notation. For example:

  ```
  123
   25.45
     .432
    1.7E4     (equivalent to 17000)
   7.6e-3   (equivalent to .0076)
  ```

  Commas are not allowed, except as decimal points. (QMF allows commas for decimal points only when they are defined as such to the database manager.)
- *Variables* are restricted by how the expression is used.
- *Nonnumeric literals* are symbols that are neither numbers nor variables. These are handled like strings in the evaluation of expressions.

Functions

Functions have the following syntax:

```
function-name([[expression][,][expression][,] ...])
```

In this syntax, 0 to *n* `expression` arguments can exist (where *n* is the maximum number of comma-separated expressions allowed by REXX).

In the above syntax, *function-name* must identify either a built-in function or an external function (for example, a REXX program). Evaluation of an expression is left to right, modified by parentheses and by operator precedence in the usual algebraic manner (with the exception of the minus prefix). The expression must be 1000 or fewer bytes, including variable values.

Related concepts
Variables used in forms
You can use global variables (both those defined by users and those supplied by QMF) and form variables in QMF forms. A variable can replace a string of text or a numeric value. You can assign different values to the variable to produce different reports without changing the form.

Related reference
REXX operators
There are several types of operators allowed in QMF expressions: arithmetic, comparison, concatenation, and logical (or Boolean). Each operator (except the prefix operator) acts on two terms. These terms can be symbols, functions, or subexpressions in parentheses. Each prefix operator acts on the term or subexpression that follows it.

Using calculated values in reports

You can use several methods to include calculated values in a QMF report. These methods are: including calculations in the query with SQL statements, defining a new column based on an expression, and specifying and using expressions defined on the FORM.CALC panel.

The first method of including calculations in a report is handled by the database, and the other two are handled by QMF from specifications made on the form. When calculations are specified on the form, they are evaluated using REXX.

QMF verifies conditions, column definitions, and expressions whenever a form is loaded, imported, displayed, or run with a query. When you modify a condition, column definition, or expression, QMF verifies it again. A REXX error can result if QMF passes unexpected data during verification. To avoid this kind of REXX error, include your calculation, along with validation statements, in a REXX program.

When using FORM.CONDITIONS or FORM.COLUMNS (Column Definition), make sure the expression or program returns the same value if invoked multiple times with the same parameters. If the program does not return the same value, breaks might not resolve as expected, and summary values might not match printed results.

There can be a significant difference in performance, capability, and flexibility of calculations performed by the database and those evaluated using REXX. A REXX program can return values dependent upon complex logic or the values processed by REXX functions. However, although REXX offers more function and programming options, there can be some drawbacks to relying on REXX for all of the calculations in a report.

REXX requires a certain amount of resources to evaluate expressions. If REXX is called repeatedly for completion of a report, you might notice an impact on performance. Because of this, you might choose to specify some calculations in the query. For example, suppose that you need to create a new column in a report based on the following:

```
((Column A - Column B) * 100) / Column B
```

To create the column, you can enter the expression in SQL and rerun the query, or enter the expression as the definition for a new column in the form and display the report. Because the column defined in the form requires a call to REXX for every detail row processed for the report, you might decide to define the new column in the query.

How QMF and REXX interact

QMF interprets REXX expressions by invoking the DSQCXPR program as a REXX function.

The following sequence of events occurs to interpret the expression:

1. PASS NULLS literals are substituted where applicable.
2. All global variables and substitution variables are replaced in the expression and enclosed in double quotation marks.
3. The expression is concatenated to "DSQ$#VAL=".
4. REXX is invoked and the program name (DSQCXPR) and argument list (expression) are passed.
5. DSQCXPR invokes the REXX interpreter instruction for the expression.
6. Any syntax errors are captured.
7. The results from the expression via the DSQ$#VAL symbol or the error results are returned.

The @IF routine can be used to test for specific values within a REXX expression and then interpret the associated REXX expressions and return the results. The @IF routine will:

- Verify that at least three arguments are passed.
- Verify that an odd number of arguments is passed.
- Interpret odd-numbered arguments (comparisons). If the first expression evaluates to true, the next expression is interpreted and the results returned, and so on.

  If no odd-numbered arguments are true, the last argument is interpreted and returned.

Because QMF does not place double quotation marks around numeric values in REXX expressions, any negative values in your expression might not be treated as such. To avoid having negative signs treated as the subtraction arithmetic operator, you can separate the variables that get passed to REXX with commas (instead of spaces) or enclose any negative values (including substitution variables that might result in negative values) with double quotation marks. For example, `myexec(A -1)` results in an evaluation error, but `myexec(A,-1)` and `myexec("A" "-1")` do not. However, if you use commas, be aware that:

- There are limits on the number of commas that are allowed in an expression.
- You might need to modify your parse statement to include commas.

REXX limits the maximum length of a single string. As QMF adds characters to strings, a string can exceed the limit after it is processed by QMF. If REXX passes a string longer than 32,767 bytes to QMF, the string is truncated to 32,767 bytes.

To improve performance, start QMF using the REXX callable interface.

Related information

For information about limits on commas and string length in expressions, see the procedure-language information for TSO.

When expressions are evaluated by REXX

Expressions specified on the FORM.CALC panel and used as substitution variables (&CALC*id*) in text areas of the form are passed to REXX for evaluation at the certain times, depending on where they are placed in the form.

- Calculations are processed when they are formatted:
  - References on FORM.DETAIL panels with the `Select Panel Variation` field set to `NO` or to `C`*n* (where the *n* condition is false) are not evaluated.
  - If the calculation is listed on separate lines in one variation, it might be evaluated multiple times.
  - If the calculation is referenced on several selected FORM.DETAIL variations (in which the `Select Panel Variation` field is YES or `C`n, where condition n is true), the calculation might be evaluated multiple times.
- Expressions specified on the FORM.CALC panel and used as usage codes on the FORM.COLUMNS panel are evaluated by REXX whenever the value is needed for formatting.
- Expressions specified on the FORM.COLUMNS Definition panel to define a new column are evaluated by REXX each time a row is fetched from the database. Rows can be fetched more than once (for example, to support printing a report in which page-splitting is required or to support a usage code, such as TCPCT, which requires all the data to be retrieved first).
- Expressions specified on the FORM.CONDITIONS panel and referred to on a FORM.DETAIL panel variation are evaluated by REXX at least once for every detail row formatted in a report.

REXX operators

There are several types of operators allowed in QMF expressions: arithmetic, comparison, concatenation, and logical (or Boolean). Each operator (except the prefix operator) acts on two terms. These terms can be symbols, functions, or subexpressions in parentheses. Each prefix operator acts on the term or subexpression that follows it.

Restriction: FORM.CALC, FORM.CONDITIONS, and Column Definition use expressions written in REXX, which QMF does not support in CICS.

Arithmetic operators

+
Add

-
Subtract

*****
Multiply

/
Divide

%
Divide and return only the integer part of the quotient

//
Divide and return only the remainder (not modulo because the result can be negative)

******
Raise the number to a whole-number power (exponentiation)

Prefix -
Negate the next term

Prefix +
Take the next term as-is

Comparative operators

==
Exactly equal (identical)

=
Equal (numerically or when padded)

¬==, /==
Not exactly equal (inverse of ==)

¬=, /=
Not equal (inverse of =)

>
Greater than

<
Less than

< >
Not equal

>=
Greater than or equal

¬<
Not less than

<=
Less than or equal

¬>
Not greater than

Concatenation operator

||
Concatenate terms (you can use no blanks or one blank)

REXX provides other concatenation operators.

Logical (Boolean) operators

&
AND (returns 1 if both terms are true)

|
Inclusive OR (returns 1 if either term is true)

&&
Exclusive OR (returns 1 if either term is true, but not both)

Prefix ¬
Logical NOT (negates; 1 becomes 0 and vice-versa)

Operator priorities

Expression evaluation is from left to right. You can modify this order by using parentheses and operator priority.

Use parentheses to clarify the meaning when the priority of operators is not obvious. An expression in parentheses is evaluated first.

When the following sequence is encountered, and `operator2` has a higher priority than `operator1`, the expression (`term2 operator2 term3 ...`) is evaluated first, applying the same rule repeatedly, as necessary:

```
term1 operator1 term2 operator2 term3 ...
```

For example, * (multiply) has a higher priority than + (add), so 3+2*5 evaluates to 13, rather than 25, which results if strict left-to-right evaluation occurred.

The order of priority of the operators (from highest to lowest) is as follows:

+ - ¬
: Prefix operators

******
: Exponentiation

*** / % //**
: Multiply and divide

+ -
: Add and subtract

||
: Concatenation (with or without blank)

=, >, ...
: All comparison operators

&
: And

|, &&
: Or, exclusive or

The & and && operators must be followed by a blank in calculation expressions to differentiate them from substitution variables.

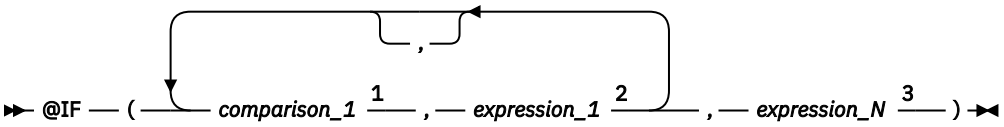
For operators of equal priority (the multiply and divide operators, for example), the left-to-right rule prevails.

The only difference between these priorities and conventional algebra is that the prefix minus operator has a higher priority than the exponential operator. Thus -3**2 evaluates to 9, not -9.

Testing for specific values within a REXX expression

The REXX @IF function is used to test for specific values within a REXX expression and then interpret the associated REXX expressions and return the results.

You can use the @IF function anywhere you normally use a REXX expression. REXX expressions can be used in FORM.CALC, FORM.CONDITIONS and FORM.COLUMNS (Column Definition).



Notes:

[1] A valid REXX expression that can be reduced to a 0 or 1. Typically contains a REXX comparative operator. The @IF function tests the comparison and if the result is 1, the expression following the function is evaluated and the results are returned. The @IF function evaluates the comparisons left to right until a true comparison is found. If no comparisons are found to be true, then the last expression is interpreted and the results are returned.

[2] A valid REXX expression consisting of terms (strings, symbols, and functions) interspersed with operators and parentheses. If the comparison preceding the expression is true, the expression is interpreted and the results are returned.

[3] A valid REXX expression. If no comparisons are true, then *expression_N* is interpreted and the results are returned.

Guidelines for using the @IF function:

- There must be an odd number of arguments.
- The minimum number of arguments is 3; the maximum is 19.
- The first token must be @IF and it must be immediately followed by a left parenthesis.
- Arguments must be delimited by commas.
- The argument list must end with a right parenthesis.
- The last argument serves as an "otherwise", or default, expression.
- If an odd-numbered argument is not the last, then it is a comparison.
- If PASS NULLS is set to YES and the expression contains a substitution variable that is null, undefined, overflows, has no instance, or no relationship, then the entire expression will be set to the value that represents that condition. This reduction is performed only on expressions, not comparisons.
- If PASS NULLS is set to YES and the expression contains more than one substitution variable that is null, undefined, overflows, has no instance, or no relationship, then the following order of precedence will be used for expression reduction:

  1. Undefined
  2. Overflow
  3. Null
  4. No instance
  5. No relationship

  The use of multiple arguments (comparisons and expressions) passed to the @IF function will eliminate the need to nest @IF functions (nested @IF functions are not supported for expression reduction).

Given SELECT ID, NAME, DEPT, SALARY, COMM FROM Q.STAFF, a new column is defined with the following expression and PASS NULLS is set to YES:

```
@If(&3=10,'MGMT',&5=DSQNULL,'N/A',&5/&4*100)
```

This expression can be logically restated as:

```
Select
  When &3 = 10    Return MGMT       /* All Department 10 employees are managers */
  When &5 is NULL Return N/A        /* Comission is NULL, mark N/A    */
  Otherwise       Return &5/&4*100  /* For all others, calculate commission % */
```

The result would be displayed as:

```
 ID  NAME      DEPT  SALARY      COMM  COL1
---  --------  ----  --------  ------  ----
 10  SANDERS    20   18357.50       -  N/A
 20  PERNAL     20   18171.25  612.45  3.37
 30  MARENGHI   38   17506.75       -  N/A
110  NGAN       15   12508.20  206.60  1.65
120  NAUGHTON   38   12954.75  180.00  1.38
160  MOLINARE   10   22959.20       -  MGMT
```

Related information

For information about other concatenation operators that REXX provides, see the procedure-language information for TSO.

Report calculation expression examples

This example illustrates the use of operators in QMF report calculations.

In the examples shown in the following table, assume that:

- &SUM1 has the value 1600
- &SUM2 has the value 400
- &DATE has the value "87/12/15"

Table 27. Examples of how to use calculation expressions

Expression	Result
&SUM2/25	16
&SUM2-&SUM1*.25	0
&SUM1+&SUM2 < 4000	1 (true)
' ' = ''	1 (true)
' ' == ''	0 (false)
&SUM1+(&DATE<'88')&SUM2	2000
date(u) (built-in function)	"12/15/87"

The following expression produces the same result as the date(u) function:

```
substr(&DATE,4,5) || "/" ||
substr(&DATE,7,8) || "/" ||
substr(&DATE,1,2)
```

Usage codes

QMF usage codes can be entered in the USAGE field on QMF FORM.MAIN or FORM.COLUMNS to define how to use column data to produce reports and charts.

This topic contains brief descriptions of each of the QMF usage codes. It contains usage code exercises and examples of how reports and charts can be changed with usage codes.

If you leave the USAGE field blank, the column data is displayed according to the edit code for the column. Some columns contain data types that QMF cannot display, such as LONG VARCHAR, LONG VARGRAPHIC, and DECFLOAT data (when the processor on which QMF is running does not support decimal floating-point instructions). In these cases, QMF displays the column metadata instead of the actual data. You can omit these columns from your report by using the OMIT usage code. This usage code can be used to omit any column from a report.

Related concepts

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.

ACROSS usage code

Additional columns of data are created, grouped, and summarized according to the values in the column that is assigned the ACROSS usage code.

Reports:

A column can have a usage of ACROSS only if one or more columns has a usage of GROUP and one or more columns use aggregations. The summary line for each group value can contain several sets of results from the columns that use aggregations. There is one set for each group of values in the column that uses ACROSS. The heading for a column that uses ACROSS has three levels:

1. The column heading as entered on the form
2. The set of values within the column
3. For each value in the set, the column headings for columns with aggregations

If more than one column has a usage of ACROSS, QMF accepts the first ACROSS and omits the remaining ACROSS columns from the report. If one column has a usage of ACROSS, no other column should have a blank usage. If you leave a column usage blank in an across report, QMF runs the report but omits all columns with blank usages.

The Across summary column (Area I) in FORM.OPTIONS shows an example across summary report with averages across columns.

Charts:

The information about reports also applies to charts. ACROSS on charts displays a category of data (such as JOB) broken down into subcategories (such as SALES and CLERK) within a larger category (such as DEPARTMENT). The data for these subcategories is displayed in a bar chart. Color display devices show the bars in different colors for different subcategories.

Aggregation usage codes

You use aggregation usage codes to summarize data in a column or replace data with a calculation.

The following table shows which aggregation usage codes are valid when used with different data types.

Table 28. Valid usage codes for data types

Data type	Valid usage codes
Numeric	AVG, COUNT, CPCT, CSUM, FIRST, LAST, MAX, MIN, PCT, STDEV, SUM, TCPCT, TPCT
Character, Date, Time, Timestamp, Timestamp with time zone	COUNT, FIRST, LAST, MAX, MIN

Restriction: LONG VARCHAR and LONG VARGRAPHIC columns cannot be aggregated. The only valid usage code for these data types is OMIT; you can also leave the USAGE field blank.

Summarizing data in a column

Reports:

Aggregation usage codes summarize the data in a column. The results of an aggregation can appear in the middle of the report as subtotals or at the end of the report as totals.

AVERAGE
Average of the values in the column

COUNT
Count of the values in the column

FIRST
First value in the column

LAST
Last value in the column

MAXIMUM
Maximum value in the column

MINIMUM
Minimum value in the column

STDEV
Standard deviation of the values in the column

SUM
Sum of the values in the column

When you use MAXIMUM and MINIMUM on character, date, time, timestamp, or graphic data, QMF uses an EBCDIC collating sequence to compare the data. To determine the maximum and minimum for numeric data, QMF uses algebraic comparisons. Nulls can be included in the result for MAX, MIN, FIRST, and LAST.

A date/time function applied to a DATE, TIME, TIMESTAMP, or TIMESTAMP WITH TIME ZONE value changes the data type of that value to numeric. Therefore, the resulting value can be aggregated.

The format of the result is determined by the edit code of the column, except for COUNT, STDEV, and percentage aggregations. COUNT can be applied to data of any type, but always produces an integer result; thus, its result is formatted with edit code K. STDEV, PCT, CPCT, TPCT, and TCPCT are formatted with edit code L.

Charts:

The information on reports for these usage codes is also true for charts.

AVERAGE, MAXIMUM, MINIMUM, STDEV, and SUM can all be useful in charting QMF data. Entries such as FIRST and LAST might not be useful in a chart format.

The following values are sent as null values to the ICU when you display a chart of the report:

- Null values in a report
- Data values too long for the width of the column
- Undefined values
- Arithmetic overflow values

Replacing a data value with a calculation

Reports:

The following codes refer to aggregations that replace each detail line value in a column with a calculation and show a final result of the aggregation at the end of the report. They can also appear in the middle of the report as subtotals.

CSUM
The cumulative sum for each value in a column.

PCT
The percentage each value is of the total:

- In reports with BREAK or ACROSS usages, PCT shows what percentage each value in the break or across group is of the break or across total.
- In all other reports, PCT shows the percentage each value in the column is of the column total.

CPCT
The cumulative percentage for each value in a column:

- In reports with BREAK or ACROSS usages, CPCT shows the cumulative percentage of the break or across total for each value in the break or across group.
- In all other reports, CPCT shows the cumulative percentage each value in the column is of the column total.

TPCT
The total percentage each value is of the column total:

- In reports with BREAK or ACROSS usages, TPCT shows what percentage each value in the column is of the column total.
- In all other reports, TPCT displays the column total.

TCPCT
The total cumulative percentage for each value in a column:

- In reports with BREAK or ACROSS usages, TCPCT shows the cumulative percentage each value in the column is of the column total.
- In all other reports, TCPCT displays the column total.

These aggregations work only on numeric data. Nulls in the column are not included in the result, but undefined values and numeric overflows are evaluated. The format of the result is determined by the edit code of the column.

Four versions of a report follow. The only difference is a result of the aggregation specified on the form for the salary column.

Report 1: SUM SALARY (total)

```
                          SUM
NAME       JOB         SALARY
---------  -----  ----------
MOLINARE   MGR       22959.20
LU         MGR       20010.00
DANIELS    MGR       19260.25
JONES      MGR       21234.00
                   ==========
                     83463.45
```

Report 2: CSUM SALARY (cumulative total)

```
                          CSUM
 NAME       JOB         SALARY
 ---------  -----  ----------
 MOLINARE   MGR       22959.20
 LU         MGR       42969.20
 DANIELS    MGR       62229.45
 JONES      MGR       83463.45
                    ==========
                      83463.45
```

Report 3: PCT SALARY (percentage)

```
                          PCT
NAME       JOB         SALARY
---------  -----  ----------
MOLINARE   MGR          27.51
LU         MGR          23.97
DANIELS    MGR          23.08
JONES      MGR          25.44
                   ==========
                      100.00
```

Report 4: CPCT SALARY (cumulative percentage)

```
                          CPCT
   NAME       JOB        SALARY
   ---------  -----  ----------
   MOLINARE   MGR         27.51
   LU         MGR         51.48
   DANIELS    MGR         74.56
   JONES      MGR        100.00
                      ==========
                         100.00
```

Two versions of the same report with a break follow. The first report uses PCT to show:

- The percentage each salary is of its break group total
- The percentage each break group is of the column total

```
                          PCT
JOB    NAME           SALARY
-----  ---------  ----------
CLERK  JAMES           25.71
       KERMISCH        23.34
       NGAN            23.81
       SNEIDER         27.14
                  ----------
               *       41.61

MGR    HANES           52.95
       SANDERS         47.05
                  ----------
               *       30.91

SALES  PERNAL          52.41
       ROTHMAN         47.59
                  ----------
               *       27.47

                  ==========
                      100.00
```

This report uses TPCT to show:

- The percentage each salary is of the column total
- Subtotals at the breaks

```
                          TPCT
JOB    NAME           SALARY
-----  ---------  ----------
CLERK  JAMES           10.70
       KERMISCH         9.71
       NGAN             9.91
       SNEIDER         11.29
                  ----------
               *       41.61

MGR    HANES           16.37
       SANDERS         14.54
                  ----------
               *       30.91

SALES  PERNAL          14.40
       ROTHMAN         13.08
                  ----------
               *       27.47

                  ==========
                      100.00
```

Whenever you use a percentage usage code (PCT, CPCT, TPCT, and TCPCT), QMF shows the total percentage as 100. However, occasionally the individual percentages add up to a number slightly higher or lower than 100. That happens because QMF sometimes rounds off the individual percentages when it calculates them.

Charts:

The above information on how usage codes affect reports is also true for charts. Some of these codes might not be as meaningful in a chart as in a report for the following reasons:

- Cumulative percentages or sums can be difficult to express in a meaningful way graphically.
- Errors that cause undefined data values are considered null values. These values appear as question marks in a report.
- If any of the following symbols are contained in a report to be charted, they are considered null values:
  - Hyphens represent null values in a report.
  - Asterisks represent data values too long for the width of the column.
  - Greater-than signs (>) represent arithmetic overflow.
  - Question marks (?) represent undefined values.

Related reference

Edit codes for numeric data
You can use several edit codes to format numeric data.

BREAK usage codes

The BREAK usage codes provide six levels of breaks (or groupings) in a report.

Reports:

When a column's usage is BREAK1, the column is a control column for level-1 breaks. Any change in the value of the column causes a break. Subtotals are displayed for columns whose usage is one of the aggregation usages, and the level-1 break text is displayed.

When you use a BREAK usage code, be aware of the following:

- To show a break in your report for each change of value in a column, your query must use an ORDER BY clause. The report then shows exactly as many breaks as there are different values in the column. Without ORDER BY, the report could show as many breaks as there are lines in the report.
- If the answer set for the query is large, QMF might perform multiple retrievals of data from the database. To ensure that the data is returned in the same order each time, be sure to include an ORDER BY clause in the query. Similarly, if BREAK is used on a defined column, ensure that multiple evaluations of the column will result in the same results each time.
- More than one column can have a usage of BREAK. The columns are then considered together for the purpose of determining breaks. For example, if a table contains columns for YEAR, MONTH, and DAY, giving each a usage code of BREAK1 causes a level-1 break at every change in date.
- A usage code of BREAK2 controls the column for level-2 breaks. The column is displayed just to the right of a control column for level-1 breaks (if the automatic column reordering option on FORM.OPTIONS is set to YES). The sequence of break numbers might have gaps. For example, you can use BREAK2, BREAK3, and BREAK5 in a form without using BREAK1 or BREAK4.

The BREAK, GROUP, and aggregation usage codes can change the order of the columns on the report. If you choose to automatically reorder the columns in a report, control columns are moved to the left of the report, and columns using aggregations are moved to the right. By default, columns are not reordered.

You can use BREAK*n*X (where *n*=1 to 6) to omit the control column from a report.

Charts:

The BREAK1 usage code can be used to modify the chart. The values in a column with a BREAK usage code are selected for the X axis. The remaining numeric columns are plotted as Y-axis data, and remaining nonnumeric columns are ignored.

You can use BREAK*n*X (where *n*=1 to 6) to omit the control column from a chart. You can also use it to get evenly spaced X-axis points for numeric data.

The chart formats provided by QMF, which are tailored to handle discrete versus continuous data.

Related reference

FORM.OPTIONS
Use FORM.OPTIONS to adjust the appearance of your report.

CALCid usage code

The CALC*id* usage code activates the evaluation of the calculation expression in FORM.CALC whose ID equals *id* for group, break, or final column summaries in the report. The result is edited according to the edit code specified on FORM.CALC and the width given on FORM.COLUMNS.

When CALC*id* is used as a usage code, the calculation is applied to the last row of data. If the column value is used in the calculation, only the last row of data is evaluated. This differs from other usage codes in which every row of data is evaluated.

GROUP usage code

The GROUP usage code identifies a column by which to group data for summaries. For example, you can group data from an employee table by department.

Reports:

The GROUP usage code displays only one line of summary data for each set of values in the column. The summary line can display only values that are the same for each member of the group, such as the value in a control column, or the results of columns that have been summarized through the use of an aggregation usage code.

When you want a report to show a summary line for each group of values in a column, use a query that includes the GROUP BY and ORDER BY clauses. GROUP BY accumulates the results of the query by group; ORDER BY orders the groups. The report then shows exactly as many summary lines as there are different values in the column. Without ORDER BY in the query, the report could show as many summary lines as there are lines in the report.

Using GROUP BY and ORDER BY can also improve the performance of a query.

When you use the GROUP usage code, be aware of the following:

- The query that selects the data must include an ORDER BY clause. Without an ORDER BY clause, the report can produce unexpected results.
- More than one column can have a GROUP usage code. If this is the case, a change in value in any column starts a new group. With two usage codes of GROUP, the report could have many more lines of grouped values.
- The report runs but omits all columns with blank usages if all of the following are true:
  - One or more columns in a report has a GROUP usage code
  - Any other column has an aggregation usage
  - Any remaining columns have blank usages
- If any column has a GROUP usage code and all other columns do not have a usage code assigned, the report omits the column containing the GROUP usage.
- Both GROUP and ACROSS columns are omitted if no columns contain aggregating usage codes.

Charts:

The effect of GROUP as it is used to format a report is similar to its effect on a chart.

Date and time usage codes

Arithmetic functions cannot be specified for DATE, TIME, TIMESTAMP, or TIMESTAMP WITH TIME ZONE values.

The following usage codes are allowed with these data types:

ACROSS
GROUP
BREAKn (n=1,2,...,6)
BREAK*n*X (*n*=1,2,...6)
FIRST
LAST
COUNT
MINIMUM
MAXIMUM
OMIT

The following usage codes are not allowed with DATE, TIME, TIMESTAMP, and TIMESTAMP WITH TIME ZONE values:

AVERAGE
STDEV
PCT
CPCT
TPCT
TCPCT
SUM
CSUM

OMIT usage code

If the usage code is OMIT, the column and its values are excluded from the tabular report or chart.

The values in the column can still appear in the report if you use form variables (such as &*n*, which represents the position of the column in the SELECT statement of the query).

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. You specify edit codes for the data you are working with on the FORM.MAIN, FORM.COLUMNS, or FORM.CALC panels.

The following table displays a summary of QMF edit codes.

Table 29. Summary of QMF edit codes

Type of data	Edit codes that you can use for this type of data	Description	More information
Character data	C	Does not change the display of the data	"Edit codes for character data" on page 261
	CW	Wraps the data at the column width boundary. To enable wrapping for CLOB or XML data, you can use this edit code.	
	CT	Wraps the data at the column width boundary, breaking the line at the nearest blank space	
	CDx	Wraps the column data according to a delimiter you specify For example, the edit code CDx wraps the column data each time an x is encountered (if the data cannot fit on one line).	
	U*xxxx*	User-defined formatting Data passed to the edit routine has the internal database representation of the source data unless the field on which the user edit code is used is the result of an expression. Replace *xxxx* with 0 - 4 characters (letters, digits, or special characters).	These codes require a custom-developed formatting routine. For more information about how to create these routines, see .
	V*xxxx*	User-defined formatting Replace *xxxx* with 0 - 4 characters (letters, digits, or special characters).	
Character or binary data	B	Binary formatting	"Edit codes for either character or binary data" on page 262
	BW	Binary formatting with column wrapping at the column width boundary. To enable wrapping for BLOB data, you can use this edit code.	
	X	Hexadecimal formatting	
	XW	Hexadecimal formatting with column wrapping at the column width boundary. To enable wrapping for BLOB data, you can use this edit code.	
	C	Binary formatting	
	CW	Binary formatting	

Table 29. Summary of QMF edit codes (continued)

Type of data	Edit codes that you can use for this type of data	Description	More information
Numeric data	E or EZ	Scientific notation	"Edit codes for numeric data" on page 263
	D, DC, DZ, DZC I, IZ J, JZ K, KZ L, LZ P, PZ	Decimal notation with different combinations of leading zeros, minus signs for negative numbers, thousands separators, currency symbols, and percent signs	
	U*xxxx* V*xxxx*	See previous description. Though V codes can be used for either character or numeric data, numeric data is converted to a character string and this character string is passed to the edit program.	"Edit codes for either character or binary data" on page 262
Graphic (double-byte character) data	G	Does not change the display of the data	"Edit codes for graphic data" on page 263
	GW	Wraps the data at the column width boundary. To enable wrapping for DBCLOB data, you can use this edit code.	
Date data	TDY*x* TDM*x* TDD*x*	Four-digit year*	"Edit codes for date data" on page 264
	TDYA*x* TDMA*x* TDDA*x*	Abbreviated two-digit year*	
	TDL	Database-defined format	
Time data	TTS*x*	24-hour clock format (including seconds)**	"Edit codes for time data" on page 265
	TTC*x*	12-hour clock format (including seconds)**	
	TTA*x*	Abbreviated clock format (no seconds)**	
	TTAN	Abbreviated clock format (no seconds, no delimiter)	
	TTU*x*	USA format**	
	TTL	Database-defined format	
Timestamp data	TSI	Formats timestamp data	"Edit codes for timestamp data" on page 266
	TSZ	Formats timestamp with time zone data	
All data types	M	Displays metadata (data type and length) instead of the actual data	"Data types for which QMF displays column metadata" on page 267

* *x* represents the character that you specify to serve as the delimiter between parts of the date.

** *x* represents the character that you specify to serve as the delimiter between parts of the time.

Edit codes for character data

You can use several edit codes to format character data.

C

Makes no change in the display of a value. You can override this edit code by setting the DSQDC_EC_CHAR global variable.

CW

Makes no change in the display of a value, but if the value cannot fit on one line in the column, this code wraps the text according to the width of the column. Instead of cutting off the data at the end of the column, QMF puts as much data as it can on a line in the column, and then continues wrapping the data on the next line in the column.

To enable wrapping for XML or CLOB data, you can use this edit code.

Data in column-wrapped columns (CW, CT, CD, XW, and BW edit codes) is always aligned using default alignment. (Alignment for headings in column-wrapped columns can be modified.) LEFT, CENTER, and RIGHT alignment are ignored for these edit codes.

If your site uses DBCS data, you can use the CW edit code on columns of mixed double-byte and single-byte character data. The minimum width of such a column is 4.

The following examples show a report before and after the width of the LOCATION column is reduced and its edit code changed to CW.

- Before column wrapping:

```
DEPTNAME        LOCATION
--------------  -------------
HEAD OFFICE     NEW YORK
PACIFIC         SAN FRANCISCO
```

- After column wrapping:

```
DEPTNAME        LOCAT
--------------  -----
HEAD OFFICE     NEW Y
                ORK
PACIFIC         SAN F
                RANCI
                SCO
```

CT

Makes no change in the display of a value, but if the value cannot fit on one line in the column, tells QMF to wrap the column according to the text in the column. Instead of cutting off the data at the end of the column, QMF fits as much data as possible on a line, interrupts the line when it finds a blank, and continues wrapping the data on the next line. If a string of data is too long to fit in the column and does not contain a blank, QMF wraps the data by width until it finds a blank and can continue wrapping by text.

If your site uses DBCS data, you can use the CT edit code on columns of mixed double-byte and single-byte character data. QMF interrupts the line when it finds an SBCS blank. The minimum width of such a column is 4.

The following examples show a report before and after the width of the LOCATION column is reduced and its edit code changed to CT.

- Before column wrapping:

```
DEPTNAME        LOCATION
--------------  -------------
HEAD OFFICE     NEW YORK
PACIFIC         SAN FRANCISCO
```

- After column wrapping:

```
DEPTNAME        LOCAT
--------------  -----
HEAD OFFICE     NEW
                YORK
PACIFIC         SAN
                FRANC
                ISCO
```

CDx
Tells QMF to wrap the column according to a delimiter in the text. QMF begins a new line in the column each time it sees a special delimiter in the text. For this edit code, replace the "x" with a special delimiter of your choice. The delimiter can be any character, including a blank, and does not appear in the output.

If your site uses DBCS data, you can use the CDx edit code on columns of mixed double-byte and single-byte character data. The minimum width of such a column is 4, and the delimiter must be outside of the DBCS string.

If a string of data is too long to fit in the column and does not contain a delimiter, QMF wraps the data by width until it finds a delimiter and can continue wrapping by that delimiter. If a string of data contains multiple successive delimiters, QMF shows a blank line for each one after the first. For example, if the data contains two delimiters, QMF begins a new line when it gets to the first delimiter, skips a line when it gets to the second delimiter, and then continues wrapping the output.

The following example shows how the text `THE GOLDEN RULE` would be formatted with an edit code of CDE (E being the delimiter character). QMF does not display or print the delimiter character.

```
TH
GOLD
N RUL
```

To allow column wrapping with date, time, and timestamp values, use CW, CT, and CDx edit codes.

When you use these edit codes (on any data type), column wrapping is only performed when tabular data is displayed or printed. A reference to &*n* in a text line only displays the first line of the wrapped data.

Edit codes for either character or binary data

You can use several edit codes to format either binary or character data.

X
Formats data as a series of hexadecimal characters.

XW
Formats data as a series of hexadecimal characters; wraps the data, breaking lines at the column boundary.

B
Formats data in binary format (as a series of zeroes and ones).

BW
Formats data in binary format; wraps the data, breaking lines at the column boundary.

C
Formats data in binary format.

CW
Formats data in binary format.

When you use edit codes XW or BW, column wrapping is only performed when tabular data is displayed or printed. A reference to &*n* in a text line only displays the first line of the wrapped data.

To enable wrapping for BLOB data, you can use the XW or BW edit code.

Edit codes for graphic data

You can use the certain edit codes to format graphic data.

G
Makes no change in the display of a value.

GW
Makes no change in the display of a value, but if the value cannot fit on one line in the column, tells QMF to wrap the text according to the width of the column. Instead of cutting off the data at the end of the column, QMF puts as much data as it can on a line in the column, and then continues wrapping the data on the next line in the column.

To enable wrapping for DBCLOB data, you can use the GW edit code.

Edit codes for numeric data

You can use several edit codes to format numeric data.

A Z in the second position of the edit code suppresses zero values.

E or EZ
Displays numbers in scientific notation. For example, with this code, the number -1234.56789 would display as -1.234E+03. The E edit code is the default for columns defined with FLOAT or DECFLOAT data types.

QMF shows up to 17 significant digits when editing floating-point data, or up to 31 significant digits when editing extended floating-point data, even if the width of the column can accommodate more. Decimal floating-point numbers show 16 significant digits for long-format values and 34 significant digits for extended-format values. To work with decimal floating-point data in QMF, the processor on which QMF is running must support decimal floating-point instructions.

D, DC, DZ, DZC, I, IZ, J, JZ, K, KZ, L, LZ, P, and PZ
These edit codes display numbers in decimal notation, with different combinations of leading zeroes, minus signs for negative numbers, thousands separators, currency symbols, and percent signs, as shown in the table later in this topic.

Each code can be followed by a number (from 0 to 99) that tells how many places to allow after the decimal point. Numbers with more places after the decimal are rounded; numbers with fewer places are padded with zeroes. A C in the second or third position of the D edit code displays a user-defined currency symbol instead of the standard currency symbol.

On the default form, the L edit code is used for all columns with numeric data types other than FLOAT or DECFLOAT. The number of decimal places used is the same as in the column definition.

You can override the default edit code for integer, small integer, and big integer by setting the DSQDC_EC_NUM global variable. You can override the default edit code for decimal data by setting the DSQDC_EC_DEC global variable.

You might notice small variances for a value when different edit codes are applied to it. For example, the value 0.068124999 displays as 0.068125 when you use an edit code of L6. However, using an edit code of L5 results in 0.06812. In this case, the digit 2 is not rounded to 3 because the next digit in the original number is less than five.

You can define a currency symbol by using the global variable DSQDC_CURRENCY.

Edit codes D, I, J, K, L, and P will format decimal floating-point numbers in decimal notation only if exponent values are less than E+100 or greater than E-100.

The following table shows what edit codes D, DC, I, J, K, L, and P provide, and how each formats the number -1234567.885. The display assumes that:

- WIDTH is 15.
- The value of DECIMAL in the QMF profile is PERIOD. (The characters used for the thousands separators and the decimal point depend on that value.)

Table 30. Attributes and examples of decimal edit codes

Edit code	Leading zeroes	Negative sign	Thousands separators	Currency symbol	Percent sign	Example
D2	N	Y	Y	Y	N	-$1,234,567.89
DC2	N	Y	Y	Y	N	-€1,234,567.89
I2	Y	Y	N	N	N	-00001234567.89
J2	Y	N	N	N	N	000001234567.89
K2	N	Y	Y	N	N	-1,234,567.89
L2	N	Y	N	N	N	-1234567.89
P2	N	Y	Y	N	Y	-1,234,567.89%

Related reference

Global variables that control various displays
DSQDC global variables control the display of certain kinds of information. All of these global variables can be modified by the SET GLOBAL command.

Edit codes for date data

The default date edit code, TD, displays dates in the format that is specified at the database requester. You can change the default date edit code by setting the DSQDC_EC_DATE global variable.

In the edit codes explained in this topic, *x* represents the character to be used as a delimiter between date values. You can choose any special character for this delimiter, including blank, but not letters or numbers.

Four-digit year

The following table shows edit codes you can use to format dates with a four-digit year.

Table 31. Edit codes to use to obtain a four-digit year in the formatted results

Edit code	Result	Format
TDY*x*	Year first	YYYY*x*MM*x*DD
TDM*x*	Month first	MM*x*DD*x*YYYY
TDD*x*	Day first	DD*x*MM*x*YYYY

Abbreviated two-digit year

The following table shows edit codes you can use to format dates with a two-digit year.

Table 32. Edit codes to use to obtain a two-digit year in the formatted results

Edit code	Result	Format
TDYA*x*	Year first	YY*x*MM*x*DD
TDMA*x*	Month first	MM*x*DD*x*YY
TDDA*x*	Day first	DD*x*MM*x*YY

Alternative date format

TDL
Locally defined. See your administrator for format information.

Date edit code examples

The examples in the following table show the date July 17, 2010, formatted with various date edit codes.

Table 33. Examples of formatting with date edit codes

Edit code	Format	Notes
TDD.	17.07.2010	European format
TDY-	2010-07-17	International Standards Organization (ISO) and Japanese Industrial Standard (JIS) formats
TDM/	07/17/2010	USA format
TDD-	17-07-2010	Four-digit year with the day first and a dash (-) for a delimiter
TDDA/	17/07/10	Two-digit year with the day first and a slash (/) for a delimiter
TDDA.	17.07.10	Two-digit year with the day first and a period (.) for the delimiter
TDDA-	17-07-10	Two-digit year with the day first and a dash (-) for the delimiter
TDDA	17 07 10	Two-digit year with the day first and a blank for the delimiter
TDMA/	07/17/10	Two-digit year with the month first and a slash (/) for the delimiter
TDMA-	07-17-10	Two-digit year with the month first and a dash (-) for the delimiter
TDYA/	10/07/17	Two-digit year with the year first and a slash (/) for the delimiter

Edit codes for time data

You can use several edit codes to format time data.

In the following table, *x* represents the character to be used as a delimiter between time values. You can choose any special character for this delimiter, including blank, but not letters or numbers.

Table 34. Time edit codes

Edit code	Format	Notes
TTS*x*	HH*x*MM*x*SS	24-hour clock, including seconds
TTC*x*	HH*x*MM*x*SS	12-hour clock, including seconds
TTA*x*	HH*x*MM	Abbreviated (no seconds)
TTAN	HHMM	Abbreviated (no seconds, no delimiter)
TTU*x*	HH*x*MM AM HH*x*MM PM	USA format
TTL	Locally defined	See your administrator for format information

Default time format

The default time edit code, TT, displays time in the format specified at the database requester. You can change the default time edit code by setting the DSQDC_EC_TIME global variable.

Time edit code examples

The examples in the following table show how the time 1:25:10 PM is formatted with various time edit codes.

Table 35. Examples of formatting with time edit codes

Edit code	Format	Notes
TTS.	13.25.10	ISO, European formats
TTS:	13:25:10	JIS format
TTU:	01:25 PM	USA format
TTS,	13,25,10	Hours, minutes, and seconds (24-hour) with a comma (,) delimiter
TTC:	01:25:10	Hours, minutes, and seconds (12-hour) with a colon (:) delimiter
TTA.	13.25	Hours and minutes (24-hour) with a period (.) delimiter
TTA,	13,25	Hours and minutes (24-hour) with a comma (,) delimiter
TTAN	1325	Hours and minutes (24-hour) with no delimiter

Edit codes for timestamp data

QMF provides the TSI and TSZ edit codes for formatting timestamp data.

TSI

The TSI edit code can be used only with columns that have a TIMESTAMP data type. The format of timestamp data edited with the TSI edit code is:

```
yyyy-mo-dd-hh.mm.ss.nnnnnnnnnnnn
```

The characters in this format have the following meanings:

yyyy
Four-digit value representing the year

mo
Two-digit value representing the month

dd
Two-digit value representing the day

hh
Two-digit value representing the hour

mm
Two-digit value representing the minutes

ss
Two-digit value representing the seconds

nnnnnnnnnnnn
Twelve-digit value representing the number of fractional seconds

For example, 2010-09-30-13.08.36.123456654321 is 1:08 P.M. and 36.123456654321 seconds on September 30, 2010, in the notation commonly used in the United States.

TSZ

The TSZ edit code can be used only with columns that have a TIMESTAMP WITH TIME ZONE data type. The time zone is the difference, in hours and minutes, between the local time and Coordinated Universal Time (UTC), formerly known as Greenwich Mean Time (GMT). The format of timestamp data formatted with the TSZ edit code is:

```
yyyy-mo-dd-hh.mm.ss.nnnnnnnnnnnnzth:tm
```

The characters in this format have the same meanings as for the TSI format with the exception of the following:

z
A plus (+) or minus (-) sign that indicates the time zone offset relative to Coordinated Universal Time (UTC)

th
A two-digit value representing the time zone hours

tm
A two-digit value representing the time zone minutes

The valid range for the time zone portion of the format is from -24:00 to +24:00. To specify UTC, you can either specify a time zone of -0:00 or +0:00 or replace the time zone offset and its sign with an uppercase Z.

For example, 2010-09-30-13.08.36.123456654321-08:00 indicates a time of 1:08 P.M. and 36.123456654321 seconds on September 30, 2010, in San Jose, California, in the United States. The timestamp 2010-09-30-13.08.36.123456654321Z indicates a time of 1:08 P.M. and 36.123456654321 seconds wherever UTC is in effect.

Data types for which QMF displays column metadata

If the column is not null, you can use the M edit code to display the metadata for the column (its data type and length) rather than the actual data.

QMF automatically assigns the M edit code to the following data types:

- BINARY
- VARBINARY
- BLOB, CLOB, or DBCLOB
- DECFLOAT (in cases where the processor on which QMF is running does not support decimal floating-point instructions)
- XML

The metadata might be truncated if the column is not wide enough to display it.

Depending on the data type, you can change the M edit code to another code using FORM.MAIN or FORM.COLUMNS, as follows:

Table 36. Data types whose edit codes default to M

Data type	Valid edit codes other than M
XML	Any edit code valid for character data. If the data in the XML column is longer than 32,767 characters, specify a value of 32767 in the WIDTH field on FORM.MAIN or FORM.COLUMNS and use the CW (column wrapping by width) edit code so that the data is not truncated. If you are working with XML data and you receive out-of-storage errors while using an edit code other than M, you can change the edit code to M to clear the error and display the report.
BINARY, VARBINARY	Any edit code valid for binary data.
DECFLOAT	If the processor on which QMF is running does not support decimal floating-point instructions, the M edit code cannot be changed. On processors that support decimal floating-point instructions, the default edit code for DECFLOAT data is E.

Table 36. Data types whose edit codes default to M (continued)

Data type	Valid edit codes other than M
LOB data types (CLOB, BLOB, DBCLOB)	• For BLOB: B, BW, X, or XW • For DBCLOB: G or GW • For CLOB: Any edit code that can be used for character data The ability to change the edit code for LOB data is controlled by the value of the DSQEC_LOB_RETRV global variable. This global variable can also be set to display LOB data instead of metadata by default. To display LOB data that is longer than the column width, specify edit codes that allow column wrapping, as follows: • For CLOB data, set the column width on FORM.MAIN or FORM.COLUMNS to a value of up to 32767 and specify the CW edit code. • For BLOB data, set the column width on FORM.MAIN or FORM.COLUMNS to a value of up to 32767 and specify the BW or XW edit code. • For DBCLOB data, set the column width on FORM.MAIN or FORM.COLUMNS to a value of up to 16383 and specify the GW edit code. If you are working with LOB data and you receive out-of-storage errors while using an edit code other than M, you can change the edit code to M to clear the error and display the report.

User-defined edit codes

Additional edit codes – U*xxxx* and V*xxxx* – are available for special purposes to format data of all types except BLOB, CLOB, DBCLOB, and XML.

The *xxxx* characters can be any 4-character combination, excluding embedded blanks or null values. To use user-defined edit codes to edit data in columns that contain DECFLOAT data, the processor on which QMF is running must support decimal floating-point instructions.

A custom-developed formatting routine is required to support these codes. See your administrator for the user edit codes available to you and the type of data supported by each code.

Considerations for aggregation functions and edit codes

QMF calculates the result of an aggregation function based on the actual values stored in the database table, not on the values resulting from the edit code for a column.

To obtain the aggregation result using the values resulting from the edit code for a column, you must use an alternative method such as defining a new column and then using a REXX function.

For example:

1. Create and save the following query, naming it Q1:

```
SELECT 10.5 from Q.ORG
```

2. Issue the command `RUN Q1 (ROW 2`. The report appears as follows:

```
  COL1
------
  10.5
  10.5
```

3. Issue the command `SH F. COL`.
4. Position the cursor under `COL1`, and press the Insert function key.

5. Type COLNEW under COLUMN HEADING, SUM under USAGE for both COL1 and COLNEW, and change the edit code for COLNEW to L as shown in the following figure:

```
FORM.COLUMNS                                          MODIFIED

                       Total Width of Report Columns: 20
 NUM COLUMN HEADING              USAGE   INDENT WIDTH EDIT  SEQ
 --- --------------------------- ------- ------ ----- ----- ---
   1 COL1                        SUM     2      6     L1    1
   2 COLNEW                      SUM     2      10    L     1
     *** END ***
```

Figure 27. Obtaining an aggregation result using values from the edit code for a column

6. Position the cursor under COLNEW, and press the Specify function key.
7. Choose **Definition**, and then press Enter.
8. Type the following REXX expression, and then press Enter:

```
format(&1,5,0)
```

9. Press the Cancel function key to close the **Specify** window.
10. Press the Report function key to display the following report:

```
  COL1      COLNEW
------  ----------
  10.5          11
  10.5          11
======  ==========
  21.0          22
```

Note that COLNEW has rounded values for each row and that the sum is the sum of the rounded values.

Variables used in forms

You can use global variables (both those defined by users and those supplied by QMF) and form variables in QMF forms. A variable can replace a string of text or a numeric value. You can assign different values to the variable to produce different reports without changing the form.

Global variables in forms make it possible for multiple queries to share the same form. For example, using the SET GLOBAL command, you can assign a string of text such as "Annual Report for 2005" to a variable such as &*ann* and use it in a form. You can use the SHOW GLOBALS command to display some or all of the available global variables. On the GLOBALS panel, you can set or change any variable that has an entry field in the Value column enclosed by brackets or parentheses. Otherwise, the variable is read-only. Change existing values by typing over the value shown.

By default, values for global variables persist for the duration of the QMF session or until you reset them. However, the DSQEC_USERGLV_SAV global variable can be set to save global variable values from one session to another.

Normally, QMF removes trailing blanks from character values for substitution variables. For numeric values, leading blanks are removed. To retain leading or trailing blanks in substitution variable values in the report, append _B to any variable on a form panel (for example: &3_B). This special syntax is meaningful only for substitution variables in the form panels. It does not apply to substitution variables used in queries or procedures, or to the variables &ROW, &DATE, &TIME, and &PAGE.

QMF supplies variables called *form variables* that return system information or information about your report. The form variables are the following:

- &ROW
- &COUNT
- &DATE
- &TIME

- &PAGE
- &CALC*id*
- &*n*
- &*an*

These variables are defined in the context of the form panel they are entered on and where they appear in the report. They are explained (if applicable) in the individual sections for each form panel.

The following table shows which variables are allowed on the various form panels.

Table 37. Variables allowed on form panels

	F.PAGE		F.BREAK		F.CALC	F.COLUMNS (Column Definition)	F.CONDITIONS	F.DETAIL		F.FINAL
	Head	Foot	Head	Foot				Head	Block	
&ROW	x	x	x	x	x	x	x	x	x	x
&DATE	x	x	x	x	x	x	x	x	x	x
&TIME	x	x	x	x	x	x	x	x	x	x
&PAGE	x	x	x	x	x			x	x	x
&COUNT				x	x				x	x
&CALC*id*				x					x	x
&*n*	x	x	x	x	x	x	x	x	x	x
&*an*				x	x				x	x
Global variables	x	x	x	x	x	x	x	x	x	x

Single or double quotation marks do not affect variables used in the form.

Related reference

SET GLOBAL

The SET GLOBAL command assigns values to global variables from the QMF command line, from a procedure, or through the callable interface. You cannot change the value of a global variable that is defined as read-only.

Chapter 5. General topics

Reference information not covered in other areas.

Naming conventions

Ensure that names of your objects adhere to the naming conventions for QMF.

Names with single-byte characters

The following naming rules apply when saving objects in the database.

- Names for queries, forms, procedures, tables, and views must be unique. (You cannot have a query and a form with the same name.)
- Names cannot start with a number.
- A name enclosed in double quotation marks can start with any character except a double quotation mark or a blank.

  Attention: Although Db2 allows double quotation marks in database object names, names of this type are not supported by QMF. QMF commands that reference object names that include a double quotation mark will result in an error, even if the entire object name is enclosed in double quotation marks. To delete an object that has a double quotation mark in its name, use the Db2 DROP statement from the SQL Query panel.
- You can use any character in a QMF object name except the following special characters:

  ```
  . , ; : < > ( ) | + - * /  = & ¬ ' "
  ```

  In some non-English single-byte character sets, the not sign (¬) displays as a circumflex (^); the vertical bar (|) displays as an exclamation point (!).
- Avoid using the special characters listed above in the name of a table, view, or other database object. If you use any of the special characters in SQL names, you must enclose the entire name in double quotation marks.
- A fully-qualified name (of the form *location.owner.name*) cannot be longer than 280 characters. The *location* qualifier can be up to 16 bytes; the *owner* qualifier can be up to 128 bytes; and the *name* of the object can be up to 128 bytes. For example, the following is a fully-qualified name:

  ```
  NEW_YORK.Q.STAFF
  ```
- Do not use QMF reserved words for names because, when used in a QMF command, they will never refer to something in the database. The QMF reserved words are:

  ```
  CHART  FORM  QUERY  DATA  TABLE  PROC  REPORT  FORM  PROFILE
  ```
- Do not use SQL reserved words for names.

Names with double-byte characters

If your site supports double-byte character set (DBCS) data, you can use double-byte characters alone or mixed with single-byte character set (SBCS) data in your names.

The following rules apply when using double-byte characters:

- Names with both double-byte and single-byte characters can contain the same single-byte characters described earlier.
- You can specify column headings in a form with mixed double-byte and single-byte characters. A heading consisting of double-byte characters only can be up to 19 double-byte characters long.

- Object names containing only double-byte characters can be no more than 63 double-byte characters. A name can be qualified by a user ID. The user ID can contain either all single-byte or all double byte characters. User IDs can be up to 128 single-byte characters or 63 double-byte characters on all databases except DB2 for VSE and VM, where user IDs must be up to eight single-byte characters or three double-byte characters.
- If your database specifically supports double-byte characters in table names, all names can contain any double-byte characters.
- If your database does not specifically support DBCS data in table names, all names can contain any double-byte characters except those that are represented internally as a double quotation mark (X'7F').

Related information

Search the SQL reference information for a list of SQL reserved words and for rules about using special characters in SQL names.

Formatting decimals with commas instead of decimal points

If you use commas instead of decimal points to indicate decimals and a number ends in a comma, the number is interpreted as an integer.

For example, consider the following command, which ends in a comma:

```
RUN PROC (&1=3,
```

This command is interpreted as follows:

```
RUN PROC (&1=3
```

Commas used as separators must have a blank after them to distinguish them from decimal indicators.

QMF temporary storage areas

Objects in QMF are held in specific temporary storage areas while you are developing them or working with them.

QUERY
Holds queries of all types. There is one temporary storage area for all query types (prompted queries, SQL queries, and QBE queries). To display the contents of the QUERY temporary storage area, enter `SHOW QUERY`.

PROC
Holds QMF procedures. There is one temporary storage area for both types of procedures (linear procedures and procedures with logic). To display the contents of the PROC temporary storage area, enter `SHOW PROC`.

FORM
Holds formatting specifications for a report. You can display formatting specifications for different parts of the report by entering `SHOW FORM` *`formname`*.

DATA
Holds data that results from IMPORT, RUN, or DISPLAY commands. The contents of the DATA area are formatted by the specifications in the FORM area to yield a report.

To display the contents of DATA, enter `SHOW REPORT`. This command does not show DATA directly (nothing does); it shows the contents of the DATA temporary storage area as formatted by the form in the FORM temporary storage area.

To display DATA in chart form with the Interactive Chart Utility (ICU), enter `SHOW CHART`.

REPORT
Holds the contents of the DATA object as formatted by the form currently in the FORM temporary storage area.

To display the contents of a report, enter `SHOW REPORT`.

CHART
Holds the CHART object, which consists of your report specifications that are displayed in graphical format by the GDDM Interactive Chart Utility.

PROFILE
Holds your QMF profile. To display the contents of the PROFILE temporary storage area, enter `SHOW PROFILE`.

To save the contents of any of these temporary storage areas, use the SAVE command.

If you did not save an object that you are working on, it is deleted when you exit QMF. It also is overwritten when you issue commands, such as the following commands, that bring a new object of the same type into the same temporary storage area:

- IMPORT
- RUN QUERY or RUN PROC
- DISPLAY *objectname*, where *objectname* is an object that is stored in the database that is different from the object of the same type that is in the temporary storage area

Examples

For example, if you are working on an SQL query that you did not save and you issue the command `DISPLAY QUERY MYQUERY`, MYQUERY overwrites the unsaved SQL query currently on the **SQL Query** panel.

Related reference

DISPLAY
The DISPLAY command displays an object from QMF temporary storage or an object from the database.

IMPORT in CICS
The IMPORT command copies the contents of a CICS data queue into QMF temporary storage or into the database.

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

RUN
The RUN command runs queries or procedures from QMF temporary storage or from the database at the current location.

SAVE
The SAVE command saves in the database at the current location objects that are currently in QMF temporary storage.

SHOW
The SHOW command has many uses. For example, you can use the SHOW command to navigate among object panels and show a variation of the FORM.DETAIL panel.

Report completion and the incomplete data prompt

When you run a query or display a table or view, QMF retrieves only enough rows from the database to display the report. This allows QMF to display the report as soon as possible, although QMF might need to retrieve more rows to finish the report.

If you do not complete the report (by either resetting the DATA object or scrolling to the bottom of the report), QMF completes it when you request the next operation that involves the database. The following commands require that QMF completes the report before the next command runs:

- CONNECT
- DISPLAY *tablename*
- DPRE

- DRAW *tablename*
- EDIT TABLE
- ERASE
- EXPORT (from the database)
- IMPORT (to the database)
- LIST
- PRINT (from the database)
- REFRESH (of a database object list)
- RUN (an object in the database)
- SAVE (DATA, FORM, PROC, QUERY, or PROFILE)

If the QMF temporary storage area becomes full while QMF completes your report, QMF displays the Incomplete Data Object prompt panel shown in the following figure.

```
                    INCOMPLETE DATA OBJECT

The temporary storage area does not contain all of the rows of
DATA.  Because there is not enough storage for QMF to capture
all the rows and columns of data, DATA must be RESET or the
current command must be withdrawn.

Do you want to RESET the DATA object?

_ 1.  YES - RESET the DATA object.
  2.  NO  - Do not RESET the DATA object.


______________________________________________________________
  F1=Help  F12=Cancel
```

Figure 28. The Incomplete Data Object prompt panel

You can respond to this prompt in one of two ways:

YES
Removes all the data in QMF temporary storage, so that none of it is available to you. If you are finished with the contents of the DATA object, choose YES.

NO
Cancels the command and leaves the DATA object as-is.

Changing QMF's response to long-running queries

Some QMF commands will not run until all the rows resulting from a query are stored in the temporary storage area. If a query is in the process of running, and you issue a new command, QMF's default response is to finish the query, and then run the new command.

You can change QMF's response to this condition by setting the DSQEC_RESET_RPT global variable as follows:

```
SET GLOBAL DSQEC_RESET_RPT=n
```

In this command, *n* can be:

0
The Reset Report prompt panel is not displayed and QMF runs the query.

1
The Reset Report prompt panel is displayed. This panel prompts you to stop or continue the query.

2
The Reset Report prompt panel is not displayed and the query is stopped.

Avoiding using nulls as data when editing a QMF object

QMF uses GDDM to display its panels, and null values (which have an internal hexadecimal representation of X'00') are subject to GDDM screen presentation. Therefore, avoid using nulls on QMF panels, such as the Edit Query panel. Instead, use an alternative, such as a constant hex representation or the database HEX function in an SQL query.

For example, to change a byte to a null value (binary zero) in a table named TEST that has a column named FLD1 with a hex value of 03C1549F, run this update statement:

```
UPDATE TEST SET FLD1=X'0300549F' WHERE FLD=X'03C1549F'
```

Now this field can be displayed using the database HEX function:

```
SELECT HEX(FLD1) FROM TEST
```

Methods of writing queries

You can write queries in Structured Query Language (SQL), or you can use assisted methods of writing queries that are called Prompted Query and Query-by-Example (QBE).

SQL

If you are familiar with SQL, you can issue SQL statements and queries directly to the database by using the **SQL Query** panel.

You can use multiple SQL statements in a query except for CALL or CREATE PROCEDURE. Each of these statements must be used alone in an SQL query. To use multiple statements, set the DSQEC_RUN_MQ global variable to 1 and place a semicolon at the end of every SQL statement except the last. No more than one SELECT statement can be used in a query that includes other SQL statements.

When any part of an object name (the location, authorization ID, or the object name itself) is continued on a new line in an SQL query, that part of the name must be delimited by double quotation marks.

Prompted Query

Prompted Query prompts you step by step to build a query. To start Prompted Query, issue the following command:

```
RESET QUERY (LANGUAGE=PROMPTED
```

You do not need the LANGUAGE parameter on the command if the query language in your profile is already set to PROMPTED.

When you begin working with a new prompted query, QMF displays a dialog panel on the right side of the screen to guide you through creating the query. As you work with the dialog panels, the prompted query is built in the echo area on the left side of the screen.

Query-by-Example (QBE)

QBE is a graphic alternative to writing queries in SQL.

Related concepts

Basic SQL statements and functions used in QMF queries
You can issue SQL statements directly to the database from the QMF SQL Query panel. The SQL Query panel supports all SQL statements that can be run dynamically.

Related reference

RUN

The RUN command runs queries or procedures from QMF temporary storage or from the database at the current location.

SET PROFILE
The SET PROFILE command changes values in your QMF profile. These values influence the behavior of your QMF session.

Global variables that control how commands and procedures are executed
DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

Procedures

When you start QMF, the system initialization procedure runs to configure the QMF session.

You can create a procedure that contains a series of QMF commands and run it with a single RUN command. This is helpful when you are using commands that are too long to enter on the command line. Avoid using system-specific commands in your procedure, if possible, because you might need to run the procedure on a system other than the system for which it was written.

When you run a procedure, the contents of QMF temporary storage areas DATA, FORM, and QUERY change just as they do with commands entered on the command line.

Because minimum unique abbreviations might change in future QMF releases, you should use the full names for commands, options, and values in procedures (rather than abbreviated names).

You can create two types of procedures: procedures with logic or linear procedures. If the first statement of a procedure is a REXX comment, QMF assumes it is a *procedure with logic*. Otherwise QMF assumes it is a *linear procedure*.

A procedure with logic can run a linear procedure and vice-versa. There is no limit on the length of any procedure.

Procedures with logic

Procedures with logic include REXX instructions that perform conditional logic and calculations, build strings, and pass commands back to the environment in which QMF is running.

Restriction: Procedures with logic are not available in CICS, as their function depends on REXX.

Procedures with logic have their own REXX variable pool. You can use procedures with logic to get and set QMF global variables. QMF commands in procedures with logic can contain substitution variables.

QMF commands in procedures with logic must be in uppercase regardless of your profile setting.

Substitution variables
The values of substitution variables are resolved at the time each command is executed.

The variable can be a private procedure variable that exists for the duration of the procedure or it can refer to a QMF global variable.

Global variables
The values of global variables are immediately available to the procedure.

Use the GET GLOBAL command to copy the value of a global variable into a procedure variable, or use the SET GLOBAL command to set new global variable values.

Return codes and procedure termination
Success or failure of a command is indicated by a return code. Your procedure must test the return code and take appropriate action to handle error conditions.

The procedure can move to the `error` label whenever a nonzero return code occurs by using the `signal on error` statement.

Continuation lines
Span multiple lines by adding a comma at the end of the previous line. Command keywords and substitution variables cannot span lines.

Comments
Create a comment by surrounding the comment text with asterisks, then slashes, as follows:

```
/*comment*/
```

Linear procedures

Linear procedures can contain comment lines, blank lines, substitution variables, any QMF command, and RUN commands that run other procedures or queries.

When a variable is set using the SET GLOBAL command in a linear procedure, the value is unavailable to commands in that same procedure because all substitution variables in a linear procedure must be resolved before the procedure is run. You are prompted for any unresolved variables in your procedure. However, the variable is available to any queries or procedures called by the procedure in which it was set.

Substitution variables
QMF scans the entire procedure for substitution variables, and the values are resolved before the procedure is run.

Global variables
Access global variable values in linear procedures by using substitution variables.

After the global variables are set, if you need to reset them you must code a RESET GLOBAL statement at the end of your procedure. Otherwise, the previous set of substitution values will continue to be used.

Return codes and procedure termination
Success or failure of a command is indicated by a return code. If a command is not successful, the procedure ends and the incorrect command is displayed at the top of the procedure area.

Continuation lines
Indicated by a plus sign (+) in column one of the continued line. Command keywords, substitution variables, and comments cannot span lines.

Comments
Comments are prefaced by two dashes, as follows:

```
  --comment
```

Printing objects

The rules for printing QMF objects vary depending on the type of object you are printing and the operating system you are using.

To print reports, tables, profiles, procedures, SQL queries, and QBE queries, use these guidelines:

- No printer nickname is required for non-GDDM printing.
- To print without GDDM, enter:

```
PRINTER=' '
```

  GDDM gets control only if the nickname is supplied on the PRINT command or in your profile.

  If no nickname is supplied (meaning you specify a blank for the printer name, as in PRINTER=' '), output goes to DSQPRINT unless you started QMF as a stored procedure, in which case output goes to a result set. If a nickname is used, output goes to GDDM.

To print charts, use these guidelines:

- A valid GDDM printer nickname is required.

- The default printer name in your profile is used if no printer name is supplied.
- The device token must be a valid print device.
- The GDDM Interactive Chart Utility always gets control when the PRINT command is issued.

To print prompted queries and forms, use these guidelines:

- A valid GDDM printer nickname is required.
- GDDM always gets control when the PRINT command is issued.
- Output goes to the ddname associated with the printer nickname.

The Table Editor

The Table Editor provides a convenient method of adding or changing rows in tables. Without writing a query, you can make changes to columns you are authorized to update.

You can add rows to a table, delete rows from a table, or search for and change existing rows in a table.

To access the Table Editor, enter one of the following commands, depending on whether you want to change existing rows or add rows to your table:

```
EDIT tablename (MODE=CHANGE
EDIT tablename (MODE=ADD
```

You use function keys to issue Table Editor commands. A different set of function keys is displayed depending on whether you are in ADD or CHANGE mode. Additionally, in those modes, when you edit columnar data having a type of VARCHAR, VARGRAPHIC, or LONG VARGRAPHIC, the Table Editor strips trailing blanks if global variable DSQCP_RMV_BLANKS is set to 1.

When performing a search, ensure that the length of your search string equals the column length, or the database will not find a match. If the length of your data is shorter than the column length, you must pad the search string with wildcards to find a match. You can use the underscore (_) wildcard to represent one character, or the percent sign (%) wildcard to represent multiple characters. For example, suppose that FLD1 is defined as a 5-character field. Its value is AB_D, which is four characters long and contains the underscore wildcard character (_). When doing a search, enter a value that represents all five character positions of the column width; for example AB_D_, AB_D%, AB_% or AB%. If you enter the actual 4-character value AB_D, QMF issues the following SELECT statement on your behalf:

```
SELECT FLD1 FROM tablename WHERE FLD1 LIKE 'AB_D'
```

The database will not find the match in this case, since FLD1 is a 5-character field. For example, with AB_D_, QMF generates the following statement:

```
SELECT FLD1 FROM tablename WHERE FLD1 LIKE 'AB_D_'
```

With AB%, QMF generates the following:

```
SELECT FLD1 FROM tablename WHERE FLD1 LIKE 'AB%'
```

The database finds the correct row in both of the last two cases because the wildcards account for all five character positions required by the database for FLD1.

Different sets of function keys appear in the Table Editor depending on which mode you are in. For example, you can press a function key labeled SEARCH while in CHANGE mode to look for the rows you want to change. SEARCH mode displays another set of function keys.

The following table lists function keys that are displayed on the various panels of the modes indicated.

Table 38. Table Editor function keys by mode

CHANGE mode	ADD mode	SEARCH mode
BACKWARD	ADD	BACKWARD

Table 38. Table Editor function keys by mode (continued)		
CHANGE mode	**ADD mode**	**SEARCH mode**
CANCEL	BACKWARD	CANCEL
CHANGE	CANCEL	CLEAR
DELETE	CLEAR	END
END	END	FORWARD
FORWARD	FORWARD	HELP
HELP	HELP	PREVIOUS
NEXT	PREVIOUS	SEARCH
REFRESH	SHOW FIELD	SHOW CHANGE
SHOW FIELD		SHOW FIELD
SHOW SEARCH		

On the SHOW FIELD panel, the Enter key closes the panel and saves the information; the Cancel key closes the panel without saving the information.

Related reference

EDIT TABLE

The EDIT TABLE command invokes the QMF Table Editor. During a Table Editor session, you can make additions, changes, or deletions to records in your table using fields on the panels provided.

Online help

Topic help, message help, and field-sensitive help are available in QMF.

Topic help

You can press the HELP function key for information any time you are viewing a QMF panel that is not displaying an error message. For example, pressing the Help function key when the QMF home panel is displayed lets you select topics of general interest and specific information about commands, forms, and all other parts of QMF.

Message help

If QMF encounters an error, a message appears just above the command line. For example, if you make a typing error in the RUN command, a message similar to the following appears:

```
RNU is not a command.
```

You can correct the command on the command line and press Enter.

If the error isn't clear from the message, press the Help function key or enter the HELP command for more information. If you need even more information, press the More Help function key. Press the Cancel function key when you want to return to the original panel.

Field-sensitive help

Field-sensitive help provides direct access to online help information for the entry fields on all form panels. To obtain field-sensitive help, position your cursor in an entry area and press the Help function key.

Related reference

HELP

The HELP command displays information about QMF. Two forms of help information are available.

Remote data access

There are two ways to access data at remote locations: using *distributed unit of work* or *remote unit of work*.

Distributed unit of work (three-part names in QMF commands)

Distributed unit of work allows you to access data at a remote location by including a three-part table or view name in a QMF command. The three parts of the name specify the location, owner, and name of the object and are separated by periods. For example, the following QMF command displays a table named STAMPS, which is owned by user ID JBP and is in a remote database named NEW_YORK:

```
DISPLAY TABLE NEW_YORK.JBP.STAMPS
```

An alias is a locally-defined name used to refer to a table or view at the same or a remote Db2 for z/OS database. You can define an alias for a remote table or view, making it easier to specify the name in QMF commands. You can list aliases that are owned by your primary and current Db2 authorization IDs. Authorization to use the table or the view to which the alias refers is checked when you use the alias in queries or QMF commands.

QMF commands with three-part names can only be initiated from Db2 for z/OS databases. They cannot be directed to DB2 for VSE and VM servers. No remote access is allowed when QMF has been started as a stored procedure.

By default, three-part names cannot be used to access remote tables that contain LOB data. However, you can set the DSQEC_LOB_RETRV global variable to 2 or 3 to access LOB metadata or data with a three-part name. Or, you can use the CONNECT command to connect to the database, and then run the query to access the remote table.

You cannot use QMF commands with three-part names to access QMF queries, procedures, forms, folders, or analytics objects at a remote server. Instead, use the CONNECT command to connect to the remote server, then issue the QMF command to access the objects you need. Also, QMF supports operations with XML data only when you are connected to a database release that supports the XML data type.

The following QMF commands support three-part table or view names:

- DISPLAY TABLE
- DRAW TABLE
- EDIT TABLE
- EXPORT TABLE
- PRINT
- SAVE DATA
- IMPORT DATA
- IMPORT TABLE

Additionally, you can use the RUN QUERY command to run SQL statements that use three-part names to refer to tables or views in remote databases.

Remote unit of work (QMF CONNECT command)

Remote unit of work lets you connect to a remote location using the QMF CONNECT command and access and use data at that location. Additionally, when you make a connection with remote unit of work, you can access data from yet another location and use it at the location to which you are currently connected.

You cannot use the CONNECT command when QMF has been started as a stored procedure.

Related reference

CONNECT in CICS
With the CONNECT command, you can connect to any database server that is part of the distributed network from within a QMF session. If you are connected to a DB2 Server for VSE and VM database, you can also use the CONNECT command to change the database user.

CONNECT in TSO
You can use the CONNECT command from within a QMF session to connect to any database server that is part of the distributed network.

The governor interrupt

Your site can set database resource limits on queries or procedures that you run.

If your query or procedure exceeds a time limit or retrieves more rows from the database than the limit set by your site, processing is interrupted. A panel is displayed that lets you specify whether you want to continue or cancel the query or procedure. In TSO, the elapsed CPU time is shown in seconds.

You can cancel or continue with or without prompting. However, if you continue, the query or procedure can still be canceled by the QMF governor.

The governor interrupt display comes from the QMF for TSO/CICS governor. If your site uses a different governor facility, your choices might be different. Your QMF administrator can provide more information on the limits set by your site.

Related information

Control QMF resource usageThe Governor provides the necessary functions for QMF and Db2 database administrators to effectively manage, control, and restrict QMF resource usage.

How QMF recasts certain data types when displaying data

When a DISPLAY TABLE command is directed to a Unicode database and the table referenced in the command contains columns with graphic data types, QMF converts the graphic data types to character data types.

- Columns defined as GRAPHIC are cast as CHAR.
- Columns defined as VARGRAPHIC or LONG VARGRAPHIC are cast as VARCHAR.
- Columns defined as DBCLOB are cast as CLOB.

QMF casts the data in this manner to avoid incompatibilities between coded character set identifiers (CCSIDs). A coded character set identifier (CCSID) contains all of the information necessary to assign and preserve the meaning and rendering of characters through various stages of processing and interchange. QMF uses EBCDIC graphic CCSIDs to display the requested data, while Unicode databases use Unicode graphic CCSIDs to retrieve the data. Incompatibilities in CCSIDs can occur for Unicode databases where the MIXED parameter has been assigned a value of NO for the DSNHDECP module.

When the following commands reference tables that contain columns with any of the graphic data types above, the data cannot be cast to prevent the incompatibility and a -332 SQL code is issued:

- EDIT TABLE (MODE=CHANGE
- EDIT TABLE (MODE=ADD

  The SQL code is issued for this command only when the command references a table that contains graphic data types and you are using QMF on a device that does not support double-byte character set (DBCS) data.
- IMPORT TABLE

  The SQL code is issued when the data to be imported contains columns with graphic data types and the data was created on a different system than the one into which the data is being imported.

Appendix A. QMF sample tables

QMF provides sample tables that you can use to help you learn and test product functions. These tables contain data about applicants, interviews, parts, products, employees, and suppliers of a fictitious electrical parts manufacturer, J & H Supply Company.

Additionally, QMF Analytics for TSO provides the following sample tables that you can use to learn about QMF Analytics for TSO functions:

- Q.CASHFLOW
- Q.CLIMATE_10YR
- Q.CLIMATE_USA
- Q.WORLDINFO

Q.APPLICANT

This table provides information about people who have applied for jobs with the company. Each row represents an applicant.

The columns are as follows:

TEMPID
Temporary identification of the applicant

NAME
Last name of the applicant

ADDRESS
City and state in which the applicant lives

EDLEVEL
Education level of the applicant

COMMENTS
Notes made by the interviewer

The Q.APPLICANT table is shown in the following figure:

```
TEMPID  NAME       ADDRESS            EDLEVEL  COMMENTS
------  ---------  -----------------  -------  ----------------------------
   400  FROMMHERZ  SAN JOSE,CA             12  NO SALES EXPERIENCE
   410  JACOBS     POUGHKEEPSIE,NY         16  GOOD CANDIDATE FOR WASHINGTON
   420  MONTEZ     DALLAS,TX               13  OFFER SALES POSITION
   430  RICHOWSKI  TUCSON,AZ               14  CAN'T START WORK UNTIL 12/96
   440  REID       ENDICOTT,NY             14  1 YEAR SALES EXPERIENCE
   450  JEFFREYS   PHILADELPHIA,PA         12  GOOD CLERICAL BACKGROUND
   460  STANLEY    CHICAGO,IL              11  WANTS PARTIME JOB
   470  CASALS     PALO ALTO,CA            14  EXPERIENCED SALESMAN
   480  LEEDS      EAST FISHKILL,NY        12  NEEDS INTERVIEW WITH BROWN
   490  GASPARD    PARIS,TX                16  WORKED HERE FROM 1/94 TO 6/94
```

Figure 29. The Q.APPLICANT table

Q.INTERVIEW

This table is for sites that support date/time data. It shows dates and times in ISO format. The format of DATE, TIME, and TIMESTAMP data in your reports depends on the format chosen as your site's default. It can be modified with edit codes that can be used with date, time, and timestamp data.

The columns in the Q.INTERVIEW table are as follows:

TEMPID
Temporary identification of the applicant

INTDATE
Date of the interview

STARTTIME
Time the interview started

ENDTIME
Time the interview ended

MANAGER
Employee number of the manager who interviewed the applicant

DISP
Whether or not the applicant will be hired

LASTNAME
Last name of the applicant

FIRSTNAME
First name of the applicant

The Q.INTERVIEW table is shown in the following table:

```
TEMPID  INTDATE     STARTTIME  ENDTIME   MANAGER  DISP    LASTNAME   FIRSTNAME
------  ----------  ---------  --------  -------  ------  ---------  ---------
   400  1996-02-05  13.00.00   15.12.00      270  NOHIRE  FROMMHERZ  RICHARD
   410  1996-02-11  15.00.00   16.18.00       10  HIRE    JACOBS     SUSAN
   420  1996-04-07  09.00.00   09.58.00      140  HIRE    MONTEZ     RITA
   430  1996-04-24  10.30.00   11.30.00      290  NOHIRE  RICHOWSKI  JOHN
   440  1996-03-13  10.15.00   11.23.00      160  HIRE    REID       CATHY
   450  1996-09-19  09.45.00   11.00.00       50  HIRE    JEFFREYS   PAUL
   460  1996-10-06  14.45.00   16.22.00      100  HIRE    STANLEY    JOHN
   470  1996-02-05  16.30.00   18.00.00      270  HIRE    CASALS     DAVID
   480  1996-03-13  13.30.00   14.45.00      160  NOHIRE  LEEDS      DIANE
   490  1996-09-30  15.00.00   15.44.00      140  NOHIRE  GASPARD    PIERRE
```

Figure 30. The Q.INTERVIEW table

Q.ORG

This table provides information on the organization of the company.

Each row represents a department. The columns are as follows:

DEPTNUMB
Number of the department (must be unique)

DEPTNAME
Descriptive name of the department

MANAGER
Employee number of the manager of the department

DIVISION
Division to which the department belongs

LOCATION
Name of the city in which the department is located

The Q.ORG table is shown in the following figure:

```
DEPTNUMB  DEPTNAME        MANAGER  DIVISION    LOCATION
--------  --------------  -------  ----------  -------------
      10  HEAD OFFICE         160  CORPORATE   NEW YORK
      15  NEW ENGLAND          50  EASTERN     BOSTON
      20  MID ATLANTIC         10  EASTERN     WASHINGTON
      38  SOUTH ATLANTIC       30  EASTERN     ATLANTA
      42  GREAT LAKES         100  MIDWEST     CHICAGO
      51  PLAINS              140  MIDWEST     DALLAS
      66  PACIFIC             270  WESTERN     SAN FRANCISCO
      84  MOUNTAIN            290  WESTERN     DENVER
```

Figure 31. The Q.ORG table

Q.PARTS

This table provides information about parts.

The columns are as follows:

SUPPNO
Number of the supplier

PARTNAME
Name of the part

PRODUCT
Product for which the part is needed

PRODNO
Number of the product

PROJNO
Number of the project

The Q.PARTS table is shown in the following figure:

```
SUPPNO  PARTNAME    PRODUCT      PRODNO  PROJNO
------  ----------  -----------  ------  ------
1100P   PLASTIC     RELAY            30  1501
1100P   STEEL       WRENCHSET       509  1520
1200S   WIRE        GENERATOR        10  1401
1200S   BEARINGS    MOTOR            50  1402
1300S   COPPER      RELAY            30  1501
1300S   BLADES      SAW             205  1510
1400P   MAGNETS     GENERATOR        10  1409
1400P   VALVES      MOTOR            50  1407
1400P   OIL         GEAR            160  1405
```

Figure 32. The Q.PARTS table

Q.PRODUCTS

This table provides information about products and their prices.

The columns are as follows:

PRODNUM
Number of the product

PRODNAME
Descriptive name of the product

PRODGRP
General type of product

PRODPRICE
Price of the product

The Q.PRODUCTS table is shown in the following table:

```
PRODNUM  PRODNAME     PRODGRP     PRODPRICE
-------  -----------  ----------  ---------
     10  GENERATOR    ELECTRICAL      45.75
    505  SCREWDRIVER  TOOL             3.70
    101  SHAFT        MECHANICAL       8.65
     20  SWITCH       ELECTRICAL       2.60
     30  RELAY        ELECTRICAL       7.55
     40  SOCKET       ELECTRICAL       1.40
     50  MOTOR        ELECTRICAL      35.80
    150  CAM          MECHANICAL       1.15
    160  GEAR         MECHANICAL       9.65
    190  BUSHING      MECHANICAL       5.90
    205  SAW          TOOL            18.90
    330  HAMMER       TOOL             9.35
    450  CHISEL       TOOL             7.75
    509  WRENCHSET    TOOL            25.90
```

Figure 33. The Q.PRODUCTS table

Q.PROJECT

This table provides information about project schedules.

The columns are as follows:

PROJNO
Number of the project (must be unique)

PRODNUM
Number of the product

DEPT
Number of the department responsible for the project

STARTD
Date the project is to start

ENDD
Date the project is to end

TIMESTAMP
Year, month, day, and time of the report

The Q.PROJECT table includes date/time data, showing dates and times in ISO format. This format is an arbitrary choice. The table you see depends on the choice made by your administrator. The Q.PROJECT table is shown in the following figure:

```
PROJNO  PRODNUM    DEPT  STARTD      ENDD        TIMESTAMP
------  -------  ------  ----------  ----------  --------------------------
1401         10      20  1996-01-01  1998-03-31  1994-12-18-10.14.44.000001
1402         50      66  1996-01-30  1997-06-30  1994-12-18-10.15.01.999998
1403        150      51  1996-02-02  1999-05-29  1994-12-18-10.22.23.000001
1404        190      38  1997-01-04  1999-06-30  1994-12-18-10.25.43.999999
1405        160      15  1997-04-29  1999-10-30  1995-12-31-14.23.00.999999
1406         20      20  1997-07-11  1998-12-31  1996-01-05-13.31.18.009999
1407         50      42  1997-12-12  2000-06-15  1996-01-05-13.42.27.000000
1408         30      42  1999-03-13  2000-09-30  1996-01-05-13.44.16.999999
1409         10      66  1998-06-15  1999-12-31  1996-03-13-09.12.57.149572
1410        190      10  1998-09-29  2000-03-31  1996-03-13-12.18.23.402917
1501         30      51  1999-01-04  1999-12-31  1996-03-13-12.22.14.201966
1502        150      38  1999-03-01  2000-07-17  1996-03-13-13.17.48.948276
```

Figure 34. The Q.PROJECT table

Q.SALES

This table provides data on sales orders made by sales representatives.

Its columns are as follows:

ORDERNO
The unique number of the parts order taken by the representative

SALESREPNO
Unique employee serial number of the sales representative who made the sale

PRODNO
The unique product number of the product sold

QUANTITY
The number of products ordered by the customer in the CUSTNO column

CUSTNO
Unique numeric identifier for each customer

The Q.SALES table is shown in the following figure:

ORDERNO	SALESREPNO	PRODNO	QUANTITY	CUSTNO
3456	20	10	50	1200
6667	20	160	120	4400
1991	40	150	600	4500
7777	60	30	150	8500
1020	60	30	150	8500
3333	70	50	240	9600
1115	70	101	120	8300
3580	20	190	360	4900
2345	90	450	360	2500
5770	70	205	100	8300
6432	40	150	120	8900
4432	90	505	150	2550
3455	150	190	360	8800
4477	220	330	480	5600
6540	150	150	200	8850
6688	280	150	300	6600
4080	300	101	500	5900
5456	300	20	60	6300
3360	310	101	120	3600
4596	310	160	100	2000
4321	340	330	200	3000
4567	40	450	100	4100
7010	20	505	150	3500
1550	90	160	200	4000
2888	90	50	240	5000
5432	220	20	100	6000
6677	40	10	150	9111
5521	60	50	150	9666
4010	150	205	225	4297
3968	220	509	200	7329
5832	280	509	300	7299
4491	300	50	100	5581
3962	340	10	240	3681

Figure 35. The Q.SALES table

Q.STAFF

This table provides data on the employees of the J&H Supply Company.

The columns are as follows:

ID
Employee serial number (must be unique)

NAME
Name of the employee

DEPT
Department number of the employee

JOB
Classification of the employee's job

YEARS
Number of years the employee has worked for the company

SALARY
Employee's annual salary in dollars and cents

COMM
Employee's commission in dollars and cents

The Q.STAFF table is shown in the following figure:

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
10	SANDERS	20	MGR	7	18357.50	-
20	PERNAL	20	SALES	8	18171.25	612.45
30	MARENGHI	38	MGR	5	17506.75	-
40	O'BRIEN	38	SALES	6	18006.00	846.55
50	HANES	15	MGR	10	20659.80	-
60	QUIGLEY	38	SALES	-	16808.30	650.25
70	ROTHMAN	15	SALES	7	16502.83	1152.00
80	JAMES	20	CLERK	-	13504.60	128.20
90	KOONITZ	42	SALES	6	18001.75	1386.70
100	PLOTZ	42	MGR	7	18352.80	-
110	NGAN	15	CLERK	5	12508.20	206.60
120	NAUGHTON	38	CLERK	-	12954.75	180.00
130	YAMAGUCHI	42	CLERK	6	10505.90	75.60
140	FRAYE	51	MGR	6	21150.00	-
150	WILLIAMS	51	SALES	6	19456.50	637.65
160	MOLINARE	10	MGR	7	22959.20	-
170	KERMISCH	15	CLERK	4	12258.50	110.10
180	ABRAHAMS	38	CLERK	3	12009.75	236.50
190	SNEIDER	20	CLERK	8	14252.75	126.50
200	SCOUTTEN	42	CLERK	-	11508.60	84.20
210	LU	10	MGR	10	20010.00	-
220	SMITH	51	SALES	7	17654.50	992.80
230	LUNDQUIST	51	CLERK	3	13369.80	189.65
240	DANIELS	10	MGR	5	19260.25	-
250	WHEELER	51	CLERK	6	14460.00	513.30
260	JONES	10	MGR	12	21234.00	-
270	LEA	66	MGR	9	18555.50	-
280	WILSON	66	SALES	9	18674.50	811.50
290	QUILL	84	MGR	10	19818.00	-
300	DAVIS	84	SALES	5	15454.50	806.10
310	GRAHAM	66	SALES	13	21000.00	200.30
320	GONZALES	66	SALES	4	16858.20	844.00
330	BURKE	66	CLERK	1	10988.00	55.50
340	EDWARDS	84	SALES	7	17844.00	1285.00
350	GAFNEY	84	CLERK	5	13030.50	188.00

Figure 36. The Q.STAFF table

Q.SUPPLIER

This table provides data on the suppliers of the J&H Supply Company.

The columns are as follows:

ACCTNO
The account number of the company

COMPANY
The name of the company

STREET
The street address of the company

CITY
The city in which the company is located

STATE
The state in which the company is located

ZIP
The company's zip code

NOTES
Other information about the supplier

The Q.SUPPLIER table is shown in the following figure:

ACCTNO	COMPANY	STREET	CITY	STATE	ZIP	NOTES
1100P	WESTCO, INC.	1900 115TH ST.	EMERYVILLE	CA	16600	THIS COMPANY HAS A STRONG HISTORY OF ON-TIME DELIVERY. WESTCO IS GROWING QUICKLY.
1200S	MAJOR ELECTRICS	4250 BENSON ST.	DALLAS	TX	87050	MAJOR ELECTRICS DECLARED BANKRUPTCY IN 1987, BUT HAS RECOVERED. FORESEE NO FURTHER PROBLEMS.
1300S	FRANKLIN, INC.	40025 EASTLAND	DOVER	DE	99000	DUE TO ITS LOCATION ON EASTERN SEABOARD, FRANKLIN HAS EXCELLENT TRANSPORTATION FACILITIES.
1400P	MOTORWORKS, INC.	19503 BESWICK	JOLIET	IL	12000	PROXIMITY TO CHICAGO ENSURES GOOD TRANSPORTATION, BOTH BY RAIL AND TRUCK. A RELIABLE SUPPLIER.

Figure 37. The Q.SUPPLIER table

You might have to adjust the column width on either FORM.MAIN or FORM.COLUMNS to see all the information in the NOTES column.

Q.CASHFLOW

This sample table provides data about cost and revenue that can be used with QMF Analytics for TSO discounted cash flow analysis.

The columns are as follows:

PERIOD
The accounting period

COSTS
Costs for the period

REVENUE
Revenue for the period

CASHFLOW
The calculated cashflow for the period

An excerpt of the Q.CASHFLOW table is shown in the following figure:

PERIOD	COSTS	REVENUE	CASHFLOW
1	-800	0	-800
2	-600	0	-600
3	-100	200	100
4	0	400	400
5	0	500	500
6	-800	300	-500

Figure 38. Excerpt from the Q.CASHFLOW table

Q.CLIMATE_10YR

This QMF Analytics for TSO sample table provides data about climate over a 10-year period.

The columns are as follows:

YEAR
The year that the climate data pertains to

MONTH
The month that the climate data pertains to

TEMPMIN
The lowest temperature (in Fahrenheit) for the month

TEMPMAX
The highest temperature (in Fahrenheit) for the month

RAINFALL
The amount of rainfall for the month in inches

SUNSHINE
Number of hours of sunshine for the month

An excerpt of the Q.CLIMATE_10YR table is shown in the following figure:

YEAR	MONTH	TEMPMIN	TEMPMAX	RAINFALL	SUNSHINE
2001	1	9	70	3	234
2001	2	18	72	7	205
2001	3	16	77	12	180
2001	4	32	91	3	230
2001	5	32	95	4	234
2001	6	41	115	3	230
2001	7	43	111	1	227
2001	8	39	115	3	238
2001	9	32	93	6	226
2001	10	27	88	6	221
2001	11	14	79	12	183
2001	12	19	73	9	204
2002	1	27	77	11	185
2002	2	25	75	45	25
2002	3	25	88	17	161
2002	4	32	90	3	226
2002	5	32	100	2	225
2002	6	43	108	4	241
2002	7	46	111	4	228
2002	8	45	113	2	238
2002	9	39	102	9	197
2002	10	32	90	3	227
2002	11	27	79	5	221
2002	12	23	66	8	219

Figure 39. Excerpt from the Q.CLIMATE_10YR table

Q.CLIMATE_USA

This QMF Analytics for TSO sample table provides data about climate in the United States, including data about rainfall and sunshine.

The columns are as follows:

MONTH
The month that the climate data pertains to

STATE
The two-letter abbreviation for the state that the climate data pertains to

TEMPMIN
The lowest temperature (in Fahrenheit) for the month

TEMPMAX
The highest temperature (in Fahrenheit) for the month

RAINFALL
The amount of rainfall for the month in inches

SUNSHINE
Number of hours of sunshine for the month

An excerpt of the Q.CLIMATE_USA table is shown in the following figure:

MONTH	STATE	TEMPMIN	TEMPMAX	RAINFALL	SUNSHINE
1	AK	9	25	1	1
1	AL	23	61	11	34
1	AR	19	50	12	1
1	AZ	12	68	7	80
1	CA	19	72	18	41
1	CO	-13	50	1	51
1	CT	18	39	3	1
1	DE	25	37	3	11
1	FL	34	70	12	83
1	GA	25	55	9	45
1	HI	63	82	7	133
1	IA	-6	27	4	1
1	ID	12	50	4	29
1	IL	9	36	2	10
1	IN	9	32	4	1
1	KS	9	54	3	50
1	KY	18	36	6	1
1	LA	27	63	10	51
1	MA	12	39	5	1
1	MD	19	39	9	1
1	ME	5	28	4	1
1	MI	9	34	3	1
1	MN	-2	19	3	1
1	MO	9	37	10	1
1	MS	28	57	6	44
1	MT	1	39	5	1
1	NC	14	50	8	23
1	ND	-6	23	3	1
1	NE	3	41	3	10
1	NH	10	32	1	1
1	NJ	21	41	2	11
1	NM	5	54	4	48
1	NV	10	57	2	58
1	NY	5	41	6	1
1	OH	18	39	2	1
1	OK	18	55	3	41
1	OR	23	52	15	9
1	PA	12	37	7	1
1	RI	16	34	3	1
1	SC	27	55	11	34
1	SD	0	34	1	3
1	TN	18	46	8	2
1	TX	19	66	4	97
1	UT	-2	54	3	41
1	VA	21	43	12	1
1	VT	12	34	3	1
1	WA	19	50	28	1
1	WI	1	25	3	1
1	WV	18	39	4	1
1	WY	0	37	1	16

Figure 40. Excerpt from the Q.CLIMATE_USA table

Q.WORLDINFO

This QMF Analytics for TSO sample table provides data about geographic regions in which the J & H Supply Company conducts business.

The columns are as follows:

COUNTRY ID
The three-digit numeric (ISO 3166-1) country code for the country

ALPHA 3
The three-letter (ISO 3166-1) country code for the country

ALPHA2
The two-letter (ISO 3166-1) country code for the country

COUNTRY
The country name

An excerpt of the Q.WORLDINFO table is shown in the following figure:

COUNTRY ID	ALPHA3	ALPHA2	COUNTRY
4	AFG	AF	AFGHANISTAN
8	ALB	AL	ALBANIA
10	ATA	AQ	ANTARCTICA
12	DZA	DZ	ALGERIA
16	ASM	AS	AMERICAN SAMOA
20	AND	AD	ANDORRA
24	AGO	AO	ANGOLA
28	ATG	AG	ANTIGUA AND BARBUDA
31	AZE	AZ	AZERBAIJAN
32	ARG	AR	ARGENTINA
36	AUS	AU	AUSTRALIA
40	AUT	AT	AUSTRIA
44	BHS	BS	BAHAMAS
48	BHR	BH	BAHRAIN
50	BGD	BD	BANGLADESH
51	ARM	AM	ARMENIA
52	BRB	BB	BARBADOS
56	BEL	BE	BELGIUM
60	BMU	BM	BERMUDA
64	BTN	BT	BHUTAN
68	BOL	BO	BOLIVIA
70	BIH	BA	BOSNIA AND HERZEGOVINA
72	BWA	BW	BOTSWANA
74	BVT	BV	BOUVET ISLAND
76	BRA	BR	BRAZIL
84	BLZ	BZ	BELIZE
86	IOT	IO	BRITISH INDIAN OCEAN TERRITORY
90	SLB	SB	SOLOMON ISLANDS
92	VGB	VG	VIRGIN ISLANDS, BRITISH
96	BRN	BN	BRUNEI DARUSSALAM
100	BGR	BG	BULGARIA
104	MMR	MM	MYANMAR
108	BDI	BI	BURUNDI
112	BLR	BY	BELARUS
116	KHM	KH	CAMBODIA
120	CMR	CM	CAMEROON
124	CAN	CA	CANADA
132	CPV	CV	CAPE VERDE
136	CYM	KY	CAYMAN ISLANDS
140	CAF	CF	CENTRAL AFRICAN REPUBLIC
144	LKA	LK	SRI LANKA
148	TCD	TD	CHAD
152	CHL	CL	CHILE
156	CHN	CN	CHINA

Figure 41. Excerpt from the Q.WORLDINFO table

Appendix B. QMF global variables

QMF provides many global variables that help you control aspects of your QMF session, QMF commands, and panel display. The global variables also help you control behavior of QMF functions in procedures and applications.

Naming convention for QMF global variables

The naming convention for most global variables that are provided with QMF is DSQ*cc_xxxxxxxxxxxx*. *cc* identifies the category of variable, and *xxxxxxxxxxxx* is a descriptive name up to 12 characters long. An underscore character (_) is included after *cc*.

cc can be any one of the following identifiers:

AP
Variables for profile-related state information

AO
Variables for other (not profile-related) state information

CM
Variables for information about the message produced by the previous command

CP
Variables for information about the Table Editor

DC
Variables that control how QMF displays information displayed on the screen

EC
Variables that control how QMF executes commands and procedures

QC
Variables whose values are produced by a CONVERT QUERY option

QM
Variables that contain RUN QUERY error message information

QW
Variables unique to QMF for Workstation.

Session variables

Session variables follow a different naming convention. Session variables are global variables that store the values that users enter in some fields on some panels if the DSQEC_SESSGLV_SAV global variable is set to 1 or 2. The naming convention for session variables is as follows:

DXY*npppp_ln_dd*

where:

- *n* is the national language identifier
- *pppp* is the last four letters of the panel ID
- *ln* is an ID that is associated with the field
- *dd* is an ID that is associated with the field and is used only if the field is dependent on another field

Setting and displaying values for global variables

If the value you want to assign to a global variable is 55 or fewer bytes, use the SET GLOBAL command to assign the value. If the variable is over 55 bytes, use the SHOW GLOBALS command.

About this task

By default, a global variable value is retained until you reset it or end the QMF session. However, the DSQEC_USERGLV_SAV global variable can be set to save global variable values from one session to another.

To customize global variables at initialization, see the information in Installing and Managing Db2 QMF for TSO and CICS about initializing global variables and QMF session behavior when QMF starts.

Procedure

To assign a value that is over 55 bytes to a global variable:

1. Use the SHOW GLOBALS command to display the **GLOBALS** panel.
2. Press the **Show Field** key to display the entire entry field.

   The maximum length for a global variable on the Show Global Variable screen is 32,768 bytes.
3. Type the value for the variable on the lines provided.

Related reference

SET GLOBAL
The SET GLOBAL command assigns values to global variables from the QMF command line, from a procedure, or through the callable interface. You cannot change the value of a global variable that is defined as read-only.

SHOW
The SHOW command has many uses. For example, you can use the SHOW command to navigate among object panels and show a variation of the FORM.DETAIL panel.

Global variables for state information not related to the profile

DSQAO global variables contain status information or settings of parameters or flags. None of these global variables can be modified by the SET GLOBAL command.

Table 39. Global variables for state information not related to the profile

Callable interface variable name	Command interface variable name	Length	Description
DSQAO_APPL_TRACE	DSQATRAC	01	**0** for level A0 **1** for level A1 **2** for level A2
DSQAO_ATTENTION	DSQCATTN	01	User attention flag.
DSQAO_BATCH	DSQABATC	01	Batch or interactive mode; values can be: **1** for an interactive session **2** for a batch-mode session

Table 39. Global variables for state information not related to the profile (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQAO_CONNECT_ID	DSQAAUTH	128	The user ID used to connect to the database and under which work is done. The value of this variable changes when you issue the following command or statement: • Issue a QMF CONNECT command to reconnect to the database under a different authorization ID • Issue a SET CURRENT SQLID statement on a Db2 for z/OS database.
DSQAO_CONNECT_LOC	None	18	The location name of the database to which you are currently connected; the name is 16 characters (padded to the right with blanks, if necessary).
DSQAO_CURSOR_OPEN	DSQACRSR	01`	Database cursor status; values can be: **1** if the cursor is open **2** if the cursor is closed
DSQAO_DATE_FORMAT	None	05	Contains the value that is specified in SYSIBM.DATE_FORMAT. Values can be ISO, USA, EUR, JIS, or LOCAL.
DSQAO_DB_MANAGER	DSQADBMG	01	Database manager, indicated by one of the following values: **1** DB2 for VSE and VM **2** Db2 for z/OS **3** Db2 for Linux, UNIX, and Windows **4** DB2 for iSeries
DSQAO_DBCS	DSQADBCS	01	DBCS support status; values can be: **1** for DBCS support **2** for no DBCS support
DSQAO_DSQSBSTG	None	10	Contains the value specified by the DSQSBSTG parameter or the default if the parameter was not specified.
DSQAO_DSQSFISO	None	01	Contains the value that is specified by the DSQSFISO parameter or the default if the parameter was not specified. The following values are used: **0** QMF is not precompiled with DATE(ISO) and TIME(ISO). **1** QMF is precompiled with DATE(ISO) and TIME(ISO). This is the default.

Table 39. Global variables for state information not related to the profile (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQAO_DSQSMRFI	None	01	This field reflects the value that was specified for the DSQSMRFI program parameter when QMF was started. **0** NO was specified for the DSQSMRFI program parameter, meaning that Db2 single-row fetch and insert is used. **1** YES was specified for the DSQSMRFI program parameter, meaning that Db2 multirow fetch and insert is used. Multirow fetch uses a rowset cursor.
DSQAO_DSQSMTHD	None	01	Contains the value specified by the DSQSMTHD program parameter or the default if the parameter was not specified. The following values are used: **0** NO was specified; QMF runs with one thread. This is the default. **1** YES was specified; QMF will run with a second thread that will be used for commands (RUN QUERY, DISPLAY TABLE) and subsequent scrolling (BOTTOM, TOP, FORWARD, BACKWARD, RIGHT and LEFT) of reports with open cursors.
DSQAO_DSQSPILL	None	01	Contains the value specified by the DSQSPILL parameter or the default if the parameter was not specified. The following values are used: **0** for not using spill storage. This value corresponds with a DSQSPILL parameter value of NO. **1** for using spill storage. This value corresponds with a DSQSPILL parameter value of YES.
DSQAO_DSQSPTYP	None	5	Contains the value specified by the DSQSPTYP parameter or the default if the parameter was not specified. The following values are used: **FILE** for spilling data to a file. **64BIT** for spilling data to extended virtual storage.
DSQAO_DSQSRSTG	None	8	Contains the value specified by the DSQSRSTG parameter or the default if the parameter was not specified.

Table 39. Global variables for state information not related to the profile (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQAO_FORM_PANEL	DSQASUBP	02	Current form panel; values can be: **1** for FORM.MAIN **2** for FORM.COLUMNS **3** for FORM.PAGE **4** for FORM.FINAL **5** for FORM.BREAK1 **6** for FORM.BREAK2 **7** for FORM.BREAK3 **8** for FORM.BREAK4 **9** for FORM.BREAK5 **10** for FORM.BREAK6 **11** for FORM.OPTIONS **12** for FORM.CALC **13** for FORM.DETAIL **14** for FORM.CONDITIONS A blank value means that the form does not exist in QMF temporary storage.
DSQAO_INTERACT	DSQAIACT	01	Setting of the interact flag; values can be: **0** for no interactive execution **1** when interactive execution is allowed
DSQAO_LOCAL_DB2	None	18	The location name of the local Db2 for z/OS database. This value is the location name for the subsystem named in the variable DSQAO_SUBSYS_ID. In a remote unit of work environment, DSQ_LOCAL_DB2 is the name of the application requester. The name is 16 characters (padded to the right with blanks, if necessary).
DSQAO_LOCATION	DSQAITLO	18	Location name of the current object, if any. This value is applicable only if a three-part name was used. The name is 16 characters (padded to the right with blanks, if necessary).
DSQAO_NLF_LANG	DSQALANG	01	National language of user; for the English language environment, this value is ‘E’.
DSQAO_NUM_FETCHED	DSQAROWS	16	Fetched data rows; contains '0' when the DATA object is empty.

Table 39. Global variables for state information not related to the profile (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQAO_OBJ_NAME	DSQAITMN	128	The name of the table (contained in a report), query, procedure, or form shown on the currently displayed panel. If the current panel does not display an object, or if the displayed object has no name, this variable contains blanks.
DSQAO_OBJ_OWNER	DSQAITMO	128	The owner of the table (contained in a report), query, procedure, or form shown on the currently displayed panel. If the current panel does not display an object, or if the displayed object has no owner, this variable contains blanks
DSQAO_OTC_LICENSE	None	01	QMF product identifier **0** Indicates that no product identifier was found. **1** Indicates that the product identifier for DB2 for QMF for z/OS standalone product, 5697-QM2, was found. **2** Indicates that the product identifier for DB2 QMF Classic Edition (5650-DB2 or 5615 DB2) was found. **3** Indicates that the product identifier for DB2 QMF Enterprise Edition (5650-DB2 or 5615 DB2) was found.
DSQAO_PANEL_TYPE	DSQAITEM	01	Type of current panel; values can be: **1** for HOME **2** for QUERY **3** for REPORT **4** for FORM **5** for PROC **6** for PROFILE **7** for CHART **8** for LIST **9** for Table Editor **A** for GLOBALS
DSQAO_QMF_RELEASE	DSQAREVN	02	Numeric release number of QMF, which is displayed in header records for exported forms, reports, and prompted queries. For QMF Version 12 Release 2, this value is '19'.

Table 39. Global variables for state information not related to the profile (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQAO_QMF_VER_RLS	DSQAQMF	10	Version and release of QMF. For QMF Version 12 Release 2, this value is 'QMFV12R1.0'.
DSQAO_QMFADM	None	01	QMF administrator authority: **0** Current authorization ID does not have QMF administrator authority. **1** Current authorization ID has administrator authority.
DSQAO_QRY_SUBTYPE	DSQASUBI	01	Query subtype; values can be: **1** for a subtype of SQL **2** for a subtype of QBE **3** for a subtype of PROMPTED Blank means that the current panel is not QUERY.
DSQAO_QUERY_MODEL	DSQAMODL	01	Model for data access. Value is always '1' for relational.
DSQAO_ROW_LENGTH	DSQAROWW	05	Contains a value indicating the length in bytes of each data row returned from the last processed query (if the report is still in storage). If the report is no longer in storage, the value is reset to 0 (zero). Values can be: **0** No report currently in storage. ***n*** Indicates the number of bytes in the row.
DSQAO_SAME_CMD	DSQACMDM	01	Values can be: **0** if the two commands are not the same **1** if the two commands are the same
DSQAO_STO_PROC_INT	None	01	Shows whether QMF for TSO was started as a Db2 for z/OS stored procedure. Possible values are: **0** QMF was not started as a stored procedure. **1** QMF was started as a stored procedure.
DSQAO_SUBSYS_ID	None	04	If QMF is running in TSO, this value is the ID of the local Db2 subsystem to which QMF is attached. If you specify a value for the DSQSSUBS program parameter in CICS, this global variable contains that value. The parameter is tolerated and the value is not processed. The value is placed in the global variable field and nothing is done with it. This logic permits the same program to be used in multiple environments.

Table 39. Global variables for state information not related to the profile (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQAO_SYSTEM_ID	DSQASYST	01	Current operating system; values can be: **2** TSO under z/OS **3** TSO or native z/OS **5** CICS
DSQAO_TERMINATE	DSQCSESC	01	QMF termination flag; values can be: **0** if the session was not marked for termination **1** if the session was marked for termination
DSQAO_TIME_FORMAT	None	05	Contains the value that is specified in SYSIBM.TIME_FORMAT. Values can be ISO, USA, EUR, JIS, or LOCAL.
DSQAO_VARIATION	DSQAVARN	02	Form panel variation number; blank means FORM.DETAIL is not the current panel.

Global variables for profile-related state information

DSQAP global variables store information related to QMF profile settings. None of these global variables can be modified by the SET GLOBAL command.

Table 40. Global variables for profile-related state information

Callable interface variable name	Command interface variable name	Length	Description
DSQAP_CASE	DSQAPCAS	01	CASE parameter; values can be: **1** for UPPER **2** for MIXED **3** for STRING If your site uses RACF support for mixed-case passwords under TSO, set this value to 2. Without this setting, all input (including passwords) is converted to uppercase, causing the CONNECT command to fail. When you set CASE to MIXED, ensure that you enter all input in uppercase, because QMF recognizes commands only in uppercase.
DSQAP_CONFIRM	DSQAPRMP	01	CONFIRM parameter; values can be: **0** for NO **1** for YES

Table 40. Global variables for profile-related state information (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQAP_DECIMAL	DSQAPDEC	01	DECIMAL parameter; values can be: **1** for PERIOD **2** for COMMA **3** for FRENCH
DSQAP_LENGTH	DSQAPLEN	18	LENGTH parameter; its value is that of the parameter ('1' through '999' or 'CONT').
DSQAP_MODEL	None	08	MODEL parameter; its value is that of the parameter.
DSQAP_PFKEY_TABLE	DSQAPPFK	31	Name of the function keys table.
DSQAP_PRINTER	DSQAPPRT	08	PRINTER parameter; values can be: • A nickname for a GDDM printer. • Blanks for the printer associated with DSQPRINT.
DSQAP_QUERY_LANG	DSQAPLNG	01	LANGUAGE parameter; values can be: **1** for SQL **2** for QBE **3** for PROMPTED
DSQAP_QUERY_MODEL	DSQAMODP	01	Model for data access. Value is always '1' for relational.
DSQAP_RESOURC_GRP	DSQAPGRP	16	RESOURCE GROUP parameter.
DSQAP_SPACE	DSQAPSPC	50	SPACE parameter; its value is that of the parameter.
DSQAP_SYNONYM_TBL	DSQAPSYN	31	Name of the synonyms table used for the current QMF session. When a user enters a command synonym, the synonym definition must be stored in the table named here or the command fails.
DSQAP_TRACE	DSQAPTRC	18	TRACE parameter; values can be: **ALL** (maximum tracing) **NONE** (minimum tracing) You can also specify a series of letters and numbers that specifies which components are to be traced at which levels of detail (for example, A2L2C1).
DSQAP_WIDTH	DSQAPWID	18	WIDTH parameter; its value is that of the parameter ('22' through '999').

Global variables associated with CICS

DSQAP global variables are associated with CICS environments. Only DSQAP_CICS_PQNAME and DSQAP_CICS_PQTYPE can be modified by the SET GLOBAL command.

When the queue type is transient data (TD), the maximum length of the corresponding queue name is 4. For example, if DSQAO_CICS_SQTYPE is TD, the maximum length of DSQAO_CICS_SQNAME is 4.

Table 41. Global variables associated with the CICS environment

Callable interface variable name	Command interface variable name	Length	Description
DSQAP_CICS_PQNAME	None	08	Names the CICS data queue to contain the QMF print output.
DSQAP_CICS_PQTYPE			Type of CICS storage used to contain the QMF print output: **TS** Writes the QMF print to a CICS temporary storage queue on an auxiliary storage device. This value is the default. **TD** Writes the QMF print to a CICS transient data queue.
DSQAO_CICS_SQNAME	None	08	Names the CICS data queue to be used as the spill file.
DSQAO_CICS_SQTYPE	None	02	Type of CICS storage used to contain the QMF spill file: **TS** Writes the QMF spill data to a CICS temporary storage queue on an auxiliary storage device. This value is the default. **TD** Writes the QMF spill data to a CICS transient data queue.
DSQAO_CICS_TQNAME	None	08	Names the CICS data queue to contain the QMF trace data.
DSQAO_CICS_TQTYPE	None	02	Type of CICS storage used to contain the QMF trace data: **TS** Writes the QMF trace to a CICS temporary storage queue on an auxiliary storage device. **TD** Writes the QMF trace to a CICS transient data queue. This value is the default.

Global variables related to a message produced by the most recent command

DSQCM global variables contain information about the most recent QMF command that was issued. None of these global variables can be modified by the SET GLOBAL command.

Table 42. Global variables that capture information about the most recently issued command

Callable interface variable name	Command interface variable name	Length	Description
DSQCM_MESSAGE	DSQCM_MESSAGE	80	Message text.
DSQCM_MESSAGE_ALL	DSQCIMSA	360	Complete message text.

Table 42. Global variables that capture information about the most recently issued command (continued)			
Callable interface variable name	**Command interface variable name**	**Length**	**Description**
DSQCM_MSG_HELP	DSQCIMID	08	ID of message help panel.
DSQCM_MSG_NUMBER	DSQCIMNO	08	Message number.
DSQCM_SUB_TXT_*nn*	DSQCIM*nn*	20	Substitution value *nn*.

Global variables associated with the Table Editor

DSQCP global variables are associated with the operations of the Table Editor. All of these global variables can be modified by the SET GLOBAL command.

The following table shows global variables that are associated with the operations of the Table Editor. All of these global variables can be modified by the SET GLOBAL command.

If the CONFIRM option of the EDIT TABLE command is NO, the Table Editor suppresses the display of all confirmation panels. If the CONFIRM option is YES, the Table Editor determines which categories of confirmation are enabled by checking the values of the global variables that are shown in this table.

The Table Editor defaults depend on the SAVE keyword from the EDIT TABLE command:

- When SAVE=IMMEDIATE, the default for each category is to enable.
- When SAVE=END, the default for the DELETE, MODIFY, and END/CANCEL categories is to enable; the default for the ADD and CHANGE categories is to disable.

Table 43. Global variables associated with the Table Editor			
Callable interface variable name	**Command interface variable name**	**Length**	**Description**
DSQCP_RMV_BLANKS	None	01	Retains or removes trailing blanks of VARCHAR columns. This variable affects only the Table Editor in Change mode. Values can be: **0** Trailing blanks of VARCHAR columns are not removed. **1** Trailing blanks of VARCHAR columns are removed. This value is the default.
DSQCP_TEADD	None	01	Displays a confirmation panel after an ADD subcommand; values can be: **0** Panel is disabled. **1** Panel is enabled. **2** Panel is enabled or disabled depending on the Table Editor defaults. This value is the default.
DSQCP_TECHG	None	01	Displays a confirmation panel after a CHANGE subcommand; values can be: **0** Panel is disabled. **1** Panel is enabled. **2** Panel is enabled or disabled depending on the Table Editor defaults. This value is the default.

Table 43. Global variables associated with the Table Editor (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQCP_TEDEL	None	01	Displays a confirmation panel after a DELETE subcommand; values can be: **0** Panel is disabled. **1** Panel is enabled. **2** Panel is enabled or disabled depending on the Table Editor defaults. This value is the default.
DSQCP_TEDFLT	None	01	The reserved character used to indicate the default value for a column in the Table Editor; initially set to a plus sign (+) character.
DSQCP_TEDFLT_DBCS	None	04	The reserved DBCS character used to indicate the default value for a graphic string column in the Table Editor. The value must be a 4-byte mixed string, composed of one DBCS character, preceded by the shift-out character, and followed by the shift-in character. It is initially set to a DBCS plus sign (+) character. This global variable is used only in a DBCS environment.
DSQCP_TEEND	None	01	Displays a confirmation panel when you issue an END subcommand or a CANCEL subcommand to terminate a Table Editor subsession. The panel can be displayed in several variations: • If END or CANCEL is issued • If modifications are made to the database • If the screen contains modified data when END or CANCEL is issued Values can be: **0** Panel is disabled. **1** Panel is enabled. **2** Panel is enabled or disabled depending on the Table Editor defaults. This value is the default.
DSQCP_TEMOD	None	01	Displays a confirmation panel when displayed data is modified and a PREVIOUS, CLEAR, SHOW CHANGE, SHOW SEARCH, REFRESH, or NEXT subcommand is issued. The resulting panel includes the name of the subcommand as part of the panel text. Values can be: **0** Panel is disabled. **1** Panel is enabled. **2** Panel is enabled or disabled depending on the Table Editor defaults.
DSQCP_TENULL	None	01	The reserved character used to indicate the null value for a column in the Table Editor; initially set to a hyphen (-) character.

Table 43. Global variables associated with the Table Editor (continued)			
Callable interface variable name	**Command interface variable name**	**Length**	**Description**
DSQCP_TENULL_DBCS	None	04	The reserved DBCS character used to indicate the null value for a graphic-string column in the Table Editor. The character is also used to indicate ignore in the context of search criteria. The value must be a 4-byte mixed string composed of one DBCS character, preceded by the shift-out character, and followed by the shift-in character. It is initially set to a DBCS hyphen (-) character. This global variable is used only in a DBCS environment.

Global variables that control various displays

DSQDC global variables control the display of certain kinds of information. All of these global variables can be modified by the SET GLOBAL command.

Table 44. Global variables that control the display of certain types of information			
Callable interface variable name	**Command interface variable name**	**Length**	**Description**
DSQDC_COL_LABELS	None	01	Controls whether the column heading shown in FORM.MAIN and FORM.COLUMNS defaults to the database label assigned to the column or the name of the column in the table from which it was selected. **0** Column names are used as column headings in default QMF forms. **1** Database labels are used as column headings in default QMF forms. This value is the default value.
DSQDC_COST_EST	None	01	Controls the display of the database cost estimate; values can be: **0** Does not display the cost estimate. **1** Displays the cost estimate. This value is the default. **2** Does not display the database status and cost estimate panels.

Table 44. Global variables that control the display of certain types of information (continued)

Callable interface variable name	Command interface variable name	Length	Description

DSQDC_CURRENCY	None	18	The currency symbol used when the DC edit code is specified.
			
			The value can be a string with a length from 1 to 18 bytes. For English, the default is the euro currency symbol. The default varies for other languages. In a DBCS environment, this value can be a mixed string of SBCS and DBCS characters. The total length of the mixed string, including the shift-out and shift-in characters, cannot exceed 18 bytes.
			
			If you require a currency symbol that is not represented on the keyboard, you can still specify the symbol. Set the DSQDC_CURRENCY variable in a procedure with logic to the hex value that is equivalent to the correct symbol. For example, the following procedure sets the currency symbol to HEX '9F', which specifies the euro currency symbol in English QMF:
			
			`/*  */`
			`"SET GLOBAL (DSQDC_CURRENCY =" '9F'X`
			
			If trailing blanks are needed for the currency symbol, put the currency symbol in single quotation marks. This example shows the blanks for French QMF:
			
			`SET GLOBAL (DSQDC_CURRENCY = 'FR '`
			
			You can issue this command from the command line or in a linear procedure.

DSQDC_DISPLAY_RPT	DSQADPAN	01	Displays a report after RUN QUERY; values can be: **0** QMF does not display the resulting report from a RUN QUERY command. This value is the default if QMF is started interactively with DSQQMF*n* (where *n* is a where n is a National Language Feature identifier). This value is also the default if QMF is started in batch mode. Changing this variable when QMF is started in batch mode does not cause any QMF screen to display. **1** QMF automatically displays the report. This value is the default if QMF is started with the callable interface. The value can be overridden with the DSQADPAN program parameter on the START command. When setting this global variable to 1, you can review the displayed report and choose whether to commit or roll back changes. To do this, press F3 (END) when you have finished reviewing your changes. You will then be prompted to either commit or roll back changes. Select 1 to commit or 2 to roll back your changes and then press Enter.

Table 44. Global variables that control the display of certain types of information (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQDC_EC_CHAR	None	05	User-defined default edit code for character data (fixed character, varying character, and very long character). **C** Does not change the display of the data. This is the default. **CW** Wraps the data at the column width boundary. **CT** Wraps the data at the column boundary, breaking the line at the nearest blank space. **CD*x*** Wraps the column data according to a delimiter (x) you specify if the data cannot fit on one line. The delimiter can be any character, including a blank and does not appear in the output. **U*xxxx*** User-defined formatting. Replace xxxx with 0 - 4 characters (letters, digits, or special characters). **V*xxxx*** User-defined formatting. Replace xxxx with 0 - 4 characters (letters, digits, or special characters). **B** Binary formatting. **BW** Binary formatting with column wrapping at the column width boundary. **X** Hexadecimal formatting.. **XW** Hexadecimal formatting with column wrapping at the column width boundary. **M** Displays metadata (data type and length) instead of the actual data.
DSQDC_EC_DATE	None	05	Default edit code for DATE data. Values can be: **TDY*x*** Four-digit year with year first. **TDM*x*** Four-digit year with month first. **TDD*x*** Four-digit year with day first. **TDYA*x*** Abbreviated two-digit year with year first. **TDYM*x*** Abbreviated two-digit year with month first. **TDDA*x*** Abbreviated two-digit year with day first. **TDL** Locally defined date format. **TD** Default date format of the database system. This is the default value for this global variable. *x* represents the character that you specify to serve as the delimiter between parts of the date.

Table 44. Global variables that control the display of certain types of information (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQDC_EC_DEC	None	05	User defined default edit code for decimal data. **E or EZ** Scientific notation. A Z in the second position of the edit code suppresses zero values. **D, DC, DZ, DZC, I, IZ, J, JZ, K, KZ, L, LZ, P, PZ** Decimal notation with different combinations of leading zeros, minus signs for negative numbers, thousands separators, currency symbols, and percent signs. Each code can be followed by a ' ' (blank), a number (from 0 to 99) or an * (astersik). Specifying a blank is the same as specifying a 0. A value of K invokes the same behavior as K0. For example, K, K0, K3 or K* are all valid settings. When the code is followed by a number (from 0 to 99) or blank, that tells how many places to allow after the decimal point. A C in the second or third position of the D edit code displays a user-defined currency symbol instead of the standard currency symbol. A Z in the second position of the edit code suppresses zero values. When the code is followed by an *, QMF will format decimal data based on the column definition of the database. The default value is L*. When L* is specified, QMF will format decimal data based on the column definition of the database. This behavior is consistent with previous releases of QMF. A C in the second or third position of the D edit code displays a user-defined currency symbol instead of the standard currency symbol. A Z in the second position of the edit code suppresses zero values. The default value is L. When L* is specified, QMF will format decimal data based on the column definition of the database. This behavior is consistent with previous releases of QMF. **U*xxxx*** User-defined formatting. Replace *xxxx* with 0 - 4 characters (letters, digits, or special characters). **V*xxxx*** User-defined formatting. Replace *xxxx* with 0 - 4 characters (letters, digits, or special characters). **M** Displays metadata (data type and length) instead of the actual data.

Table 44. Global variables that control the display of certain types of information (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQDC_EC_NUM	None	05	User-defined default edit code for numeric data (integer, small integer, and big integer.) **E or EZ** Scientific notation. A Z in the second position of the edit code suppresses zero values. **D, DC, DZ, DZC, I, IZ, J, JZ, K, KZ, L, LZ, P, PZ** Decimal notation with different combinations of leading zeros, minus signs for negative numbers, thousands separators, currency symbols, and percent signs. A C in the second or third position of the D edit code displays a user-defined currency symbol instead of the standard currency symbol. A Z in the second position of the edit code suppresses zero values. The default value is L. **U*xxxx*** User-defined formatting. Replace *xxxx* with 0 - 4 characters (letters, digits, or special characters). **V*xxxx*** User-defined formatting. Replace *xxxx* with 0 - 4 characters (letters, digits, or special characters). **M** Displays metadata (data type and length) instead of the actual data.
DSQDC_EC_TIME	None	05	Default edit code for TIME data. Values can be: **TTS*x*** 24-hour clock format (including seconds). **TTC*x*** 12-hour clock format (including seconds). **TTA*x*** Abbreviated clock format (no seconds). **TTAN** Abbreviated clock format (no seconds, no delimiter). **TTU*x*** USA format. **TTL** Locally defined time format. **TT** Default time format of the database system. This is the default value for this global variable. *x* represents the character that you specify to serve as the delimiter between parts of the time.

Table 44. Global variables that control the display of certain types of information (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQDC_LIST_ORDER	None	02	Sets the default sort order for objects in a list of database objects. Values for the first character can be: **1** The list uses 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 date modified. **6** The list is sorted by date last used. The list of commands that cause this date to be updated is set by the DSQEC_LAST_RUN global variable. Values for the second character can be: **A** The list is sorted in ascending order. **D** The list is sorted in descending order. This variable applies only to objects that are listed as a result of the LIST command. It does not apply to lists produced in other contexts, such as from a Display Prompt panel, and it does not apply to lists of tables.
DSQDC_POS_SQLCODE	None	01	Sets the action QMF takes when a positive SQL code is returned from the database. Possible values are: **0** Does not log the message to the trace data file (DSQDEBUG) and no help text is provided. **1** Logs the QMF message associated with the SQL code to the trace data file (DSQDEBUG). **2** QMF message help is available for the positive SQL code. This global variable does not apply to SQL codes +495 and +100.
DSQDC_SCROLL_AMT	None	04	Sets the scroll amount for QMF panels; values can be: **Csr** Sets the scroll amount to `cursor`. QMF scrolls the line or column where the cursor is positioned to the bottom of the scrollable area when you scroll backward. Likewise, QMF scrolls to the top when you scroll forward, and to the far left and far right when you scroll left or right. **Half** Sets scroll amount to half the scrollable area. **Page** Sets scroll amount to a full page. This value is the default. ***n*** Sets scroll amount to *n* number of lines or columns. You can specify any number from 1 to 9999 for *n*.

Table 44. Global variables that control the display of certain types of information (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQDC_SHORT_EXPT	None	01	Applies to data or tables exported with a value of QMF on the DATAFORMAT parameter of the EXPORT command. Controls the length of all column name fields in the header records. Possible values are: **0** QMF sets the length of column fields in the header records to 30 bytes. This length is the default length for: • Db2 for z/OS Version 8.1.5, or later • DB2 for iSeries Version 5.2, or later • Db2 for Linux, UNIX, and Windows, Version 8.1, or later **1** QMF sets the length of column fields in the header records to 18 bytes. This length is the default length for: • Db2 for z/OS, Version 8.1.5, or earlier • DB2 for iSeries, Version 5.2, or earlier • Db2 for Linux, UNIX, and Windows, Version 8.1, or earlier • All DB2 Server for VSE and VM databases
DSQDC_SHOW_PANID	DSQCPDSP	01	Displays panel IDs of QMF panels; values can be: **0** Suppresses panel identifiers. This value is the default. **1** Displays panel identifiers.

Related reference

Global variables that control how commands and procedures are executed

DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

Global variables that control how commands and procedures are executed

DSQEC global variables control how commands and procedures are executed. All of these global variables can be modified by the SET GLOBAL command.

Table 45. Global variables that control how commands and procedures are executed

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_ALIASES	None	31	View for retrieving lists of table and view aliases when you request a list of tables from a Db2 for z/OS location. Also applies if the current server is Db2 for z/OS or Db2 for Linux, UNIX, and Windows.
DSQEC_BUFFER_SIZE	None	03	Sets the length of the data buffer used to fetch data from the database. Valid values range from 4 - 256 (each integer is 1KB; for example, 4 equals 4K, 256 equals 256K, etc.). The default value is 4 (4KB).
DSQEC_CC	None	01	Suppresses the carriage control characters in the report output format; values can be: **0** No carriage control character in column 1. **1** Carriage control is in effect; the report has a carriage control character in column 1.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_COLS_LDB2	None	31	View for retrieving column information for a table at the current location, if that location is Db2 for z/OS.
DSQEC_COLS_RDB2	None	31	View for retrieving column information for a table at a remote Db2 for z/OS location (if it is not the current location).
DSQEC_COLS_SQL	None	31	View for retrieving column information for a table in a DB2 for VSE and VM database.
DSQEC_CON_ACC_RES	None	01	Applies to executable SELECT queries that QMF submits to Db2 for z/OS. Use this variable 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, QMF 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 QMF queries (such as SQL SELECT queries, prompted queries, or QBE P. queries), but also from other QMF operations such as DISPLAY TABLE. Possible values are: **0** QMF specifies no concurrent access resolution options on the PREPARE statement associated with the pending SQL SELECT statement. This value is the default. **1** SKIP LOCKED DATA This value can be specified for executable SELECT statements directed to Db2 for z/OS Version 9 (New Function Mode), or later. **2** USE CURRENTLY COMMITTED This value can be specified for executable SELECT statements directed to Db2 for z/OS Version 10 (New Function Mode), or later. **3** WAIT FOR OUTCOME This value can be specified for executable SELECT statements directed to Db2 for z/OS Version 10 (New Function Mode), or later.
DSQEC_CON_CSWL	None	01	This global variable enables the use of the DB2 for z/OS statement concentration with literals feature. It applies to dynamic SQL SELECT statements submitted to DB2 for z/OS through QMF commands such as RUN QUERY and DISPLAY, EXPORT and PRINT TABLE. When you set this variable, QMF specifies support through the DB2 for z/OS CONCENTRATE STATEMENTS WITH LITERALS prepare attribute: 0 = Do not enable DB2 for z/OS statement concentration with literals. This is the default. 1 = Enable DB2 for z/OS statement concentration with literals.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_CURR_FOLDER	None	128	Specifies the name of the current folder to be used for QMF commands that allow folder processing (SAVE, LIST, and ERASE). The default is blank. When a folder name is identified in this global variable, that folder is used when any QMF command that uses QMF folder objects is processed. For example, when DSQEC_CURR_FOLDER is set and the SAVE QUERY AS Q1 command is executed, the query will be saved and the query object will be included in the folder that is identified in the global variable. You can override this global variable by specifying a folder name with the FOLDER keyword with the QMF command. In this case, the folder name that is specified with the FOLDER keyword overrides the folder name that is specified in the DSQEC_CURR_FOLDER global variable. If this global variable is blank and the FOLDER keyword is not specified, folder processing is not used. **Restriction:** This global variable is not supported when QMF is connected to DB2 Server for VSE and VM.
DSQEC_DISABLEADM	None	01	Suppression of QMF administrator authority. When the value of this global variable is changed, the effect is immediate. Possible values can be: **0** QMF administrator authority is available (if the authorization ID has QMF administrator authority). **1** QMF administrator authority is suppressed (regardless of the authority of the authorization ID). The initial default value for this global variable can be overridden by the DSQUOPTS initialization exit routine. Alternately, QMF administrator authority can be controlled by the user's profile MODEL setting.
DSQEC_DSALLOC_DIR	None	03	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	None	08	QMF allocates data sets in tracks. This global variable 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. PS, PDS, and PDSE data sets can have a maximum value of 16777215 tracks.
DSQEC_DSALLOC_SEC	None	08	QMF allocates data sets in tracks. This global variable 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. PS and PDS data sets can have a maximum value of 65535 tracks; PDSE data sets can have a maximum value of 16777215 tracks.
DSQEC_DSLRECL1	None	05	Specifies the logical record length (LRECL) that is to be used when an SQL query or QMF procedure is exported to a new data set. Valid values are 79 - 32760. The default value is 79.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_DSQSFISO	None	01	Specifies the format of CHAR(*datetime-expression*) data within a QMF report. The following values are used: **0** The result of CHAR(*datetime-expression*) data is in the format specified in the DATE FORMAT and TIME FORMAT fields on Db2 installation panel DSNTIP4. The current Db2 DATE and TIME format values can be found by referencing global variables DSQAO_DATE_FORMAT and DSQAO_TIME_FORMAT. **1** The result of CHAR(*datetime-expression*) data is in ISO format. DSQEC_DSQSFISO takes its default value from the value of program parameter DSQSFISO. The DSQSFISO program parameter setting may be seen in state global variable DSQAO_DSQSFISO. Note that if DSQEC_DSQSFISO is modified, the value of DSQAO_DSQSFISO will not change. DSQEC_DSQSFISO should be referenced for the current behavior settings.
DSQEC_DS_SUPPORT	None	01	Provides support for QMF Data Service (QDS) **0** Do not allow access to QMF Data Service (default). **1** Allow access to QMF Data Service. This global variable controls whether RUN QUERY (SQL, PQ or QBE), DISPLAY TABLE, DRAW, EXPORT and PRINT TABLE commands should be analyzed by the QMF Data Service component. If an object that is referenced in the command is defined to the QMF Data Service component, then the entire command is executed by QMF Data Service. If none of the objects referenced in the command access an object defined to QMF Data Service, then the command is executed by the current Db2 connection. If the QDS service could not be loaded or is not available, then this value is ignored and all requests are routed to Db2.
DSQEC_DS_NOPAR	None	01	Indicates whether Parallelism is currently in use. **0** Parallelism is currently in use (default). **1** Parallelism is not currently in use.
DSQEC_DS_PAR	None	02	The valid values: **-1** No restrictions are placed on QDS (default). **0** QDS will advise DVS that Map Reduce may be used, but Map Reduce Client may not be used. **1** Neither Map Reduce nor Map Reduce Client are allowed. **2-10** Both Map Reduce and Map Reduce Client can be used, but the degree of Map Reduce Client parallelism is limited to the number specified. (For example, 2 means that 2 parallel paths can be used, 3 means 3 can be used, and so on.) **Note:** If DSQEC_DS_NOPAR is set to 1 then the value of DSQEC_DS_PAR is ignored and no parallelism is in use.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_EDITOR	None	18	Specifies the value to use for the EDITOR keyword on the EDIT command when the EDITOR keyword is not specified. The valid values for this global variable are: **PDF** The ISPF/PDF editor is used to edit the procedure or query. To use the PDF editor to edit a query or procedure, start QMF as an ISPF dialog. **EE** The SQL QUERY or PROC enhanced editor is used to edit the procedure or query. **editorname** The name of any other editor that is available to you. You can also specify the name of a CLIST that starts an editor. For more information about available editors, see your QMF administrator. The default value is blank.
DSQEC_EDITOR_PVIEW	None	1	Controls the QMF Enhanced Editor preview command. The preview command is available when you edit an SQL query to preview the results of a SELECT query. **0** Do not allow the preview command to run. Message DYQE069 is issued to warn the user that the command is inactive. **1** Allow the preview command to run. This is the default.
DSQEC_EXPL_MODE	None	07	Specifies the setting that is to be used for the Db2 special register CURRENT EXPLAIN MODE when the RUN QUERY command is issued. The special register controls the behavior of the EXPLAIN facility for eligible dynamic SQL statements. Before a query is run, QMF sets the CURRENT EXPLAIN MODE special register to the value that is specified by this global variable. The valid values for this global variable are: **NO** The EXPLAIN facility is disabled and no EXPLAIN information is captured when explainable dynamic statements are run. This is the default value. **YES** The EXPLAIN facility is enabled and EXPLAIN information is inserted into the EXPLAIN tables for eligible dynamic SQL statements after the statement is prepared and run. All dynamic SQL statements are compiled and run. **EXPLAIN** The EXPLAIN facility is enabled and EXPLAIN information is inserted into the EXPLAIN tables for eligible dynamic SQL statements after the statement is prepared. Dynamic statements, except for SET statements, are not run. For servers other than Db2 for Linux, UNIX, and Windows or DB2 10 for z/OS (New Function Mode) or later, the only valid value is NO.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_EXTND_STG	None	31	Specifies the number of megabytes of extended storage that QMF acquires on each request to the extended storage manager when the DSQSPTYP program parameter is set to 64BIT. This program parameter is available in QMF for TSO only. When an operation requires extended storage, QMF requests 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 QMF users work. If the average size is large and you set the value low, QMF issues many calls to the extended storage manager to complete the DATA object. These repeated calls might affect performance. Values can be from 1 to 1000. The default value is 25, indicating that QMF requests 25 MB of storage on each request.
DSQEC_FORM_LANG	None	01	Establishes the default NLF language in a saved, exported, or imported form; values can be: **0** The form uses the presiding NLF language. **1** The form uses English. This value is the default.
DSQEC_ISOLATION	None	01	Default query isolation level. Values can be: **0** Isolation level UR (uncommitted read) Uncommitted read can be useful in a distributed environment. However, if you are using uncommitted read, any reports that users view might contain data that was deleted from the database after the report was displayed. **1** Isolation level CS (cursor stability) This value is the default. When using cursor stability, QMF does not display the report until all database commands that affects the data in the report are complete.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_KEEP_THREAD	None	01	Specifies whether a thread is released or kept active at the end of a query. This global variable does not affect threads that are created for procedures that run in batch mode or threads that are created when QMF is connected to a remote database through the CONNECT command. When procedures are run in batch mode, threads persist until the procedure completes. When QMF is connected to a remote database, threads persist until the connection ends. The valid values for this global variable are: **0** The thread is released at the end of the query. This is the default value. If this setting is used, the SET *DB2 global variable* statement fails unless it is run in one of the following situations: • The statement is included in a procedure that is run in batch mode. The Db2 global variable is reset to its default value after the procedure completes. • The QMF CONNECT command is issued to connect to a remote database and the SET *DB2 global variable* statement is run on the remote database. • The SET *DB2 global variable* statement is included in a multistatement query and the QMF DSQEC_RUN_MQ global variable is set to 1. The Db2 global variable is reset to its default value after the query completes. **1** The thread is kept active until the end of the QMF session or the DSQEC_KEEP_THREAD global variable is set to 0. This setting allows users to run the SET *DB2 global variable* statement to set Db2 global variables. If you set any Db2 global variables while DSQEC_KEEP_THREAD is set to 1 and then change DSQEC_KEEP_THREAD to 0, those Db2 global variables revert to their default values.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_LAST_RUN	None	01	Specifies the set of commands that cause the LAST_USED field on QMF object lists to be updated. This field is based on the LAST_USED column of the Q.OBJECT_DIRECTORY control table. The value in the LAST_USED column is updated regardless of whether the issued command is successful. However, in some cases, the LAST_USED column is not updated immediately, and if QMF is terminated abnormally, the column might not be updated. Possible values are: **0** QMF updates the LAST_USED timestamp whenever any of the following commands is issued: • CONVERT • DISPLAY • EXPORT • IMPORT • LAYOUT • PRINT • RUN • SAVE This value is the default. **1** QMF restricts updates of the LAST_USED timestamp to RUN, SAVE, and IMPORT commands only. **2** QMF restricts updates of the LAST_USED timestamp to the RUN command only.

DSQEC_LIST_OWNER	None	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, the following command followed by a LIST command instructs QMF to list 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_%`
			The value you set with this global variable does not apply to lists displayed when you press the List key on QMF panels other than the home panel.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_LOB_COLMAX	None	10	Specifies the maximum data size of a LOB column that is to be retrieved, in bytes, up to the maximum LOB size of 2147483637, or 2 GB. A value of 0 specifies no maximum. 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, an error is issued and no report data 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 issued and the command is terminated. The default is 32767.
DSQEC_LOB_RETRV	None	01	Specifies how LOB data or metadata is retrieved. The valid values are: **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. 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.
DSQEC_LOB_SAVE	None	01	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.
DSQEC_NLFCMD_LANG	None	01	Sets expected NLF language for commands. Values can be: **0** Commands must be in the presiding NLF language. This value is the default. **1** Commands must be in English.
DSQEC_PO	None	01	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.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_PRO_ENABLE	None	01	Controls whether a confirmation panel is displayed before QMF overwrites or discards the contents of the QUERY, FORM, PROC, or PROFILE temporary storage areas. Possible values are: **0** No confirmation panel is displayed before the contents of the supported temporary storage areas are overwritten. This value is the default. **1** A confirmation panel is displayed if the global variable that corresponds to the temporary storage area in question is also set to 1. The following global variables individually control overwrites in each of the supported temporary storage areas: • DSQEC_PRO_FORM controls overwrites of the FORM temporary storage area, which stores current QMF report formatting specifications. • DSQEC_PRO_PROC controls overwrites of the PROC temporary storage area, which stores current QMF procedures. • DSQEC_PRO_PROF controls overwrites of the PROFILE temporary storage area, which stores current QMF profile settings. • DSQEC_PRO_QUERY controls overwrites of the QUERY temporary storage area, which stores the current QMF query.
DSQEC_PRO_FORM	None	01	This variable controls whether a confirmation panel is displayed before QMF overwrites or discards the contents of the FORM temporary storage area. The DSQEC_PRO_ENABLE global variable must be set to 1. Possible values are: **0** No confirmation panel is displayed before the contents of the temporary storage area are discarded. **1** A confirmation panel is displayed, giving the user the opportunity to proceed or cancel the command that caused the pending discard. The contents of the temporary storage area can then be saved with the SAVE command.
DSQEC_PRO_PROC	None	01	This variable controls whether a confirmation panel is displayed before QMF overwrites or discards the contents of the PROC temporary storage area. The DSQEC_PRO_ENABLE global variable must be set to 1. Possible values are: **0** No confirmation panel is displayed before the contents of the temporary storage area are discarded. **1** A confirmation panel is displayed before the contents of the temporary storage area are discarded. The user can proceed or cancel the command that caused the pending discard. The contents of the temporary storage area can then be saved with the SAVE command.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_PRO_PROF	None	01	This variable controls whether a confirmation panel is displayed before QMF overwrites or discards the contents of the PROFILE temporary storage area. The DSQEC_PRO_ENABLE global variable must be set to 1. Possible values are: **0** No confirmation panel is displayed before the contents of the temporary storage area are discarded. **1** A confirmation panel is displayed before the contents of the temporary storage area are discarded. The user can proceed or cancel the command that caused the pending discard. The contents of the temporary storage area can then be saved with the SAVE command.
DSQEC_PRO_QUERY	None	01	This variable controls whether a confirmation panel is displayed before QMF overwrites or discards the contents of the QUERY temporary storage area. The DSQEC_PRO_ENABLE global variable must be set to 1. Possible values are: **0** No confirmation panel is displayed before the contents of the temporary storage area are discarded. **1** A confirmation panel is displayed before the contents of the temporary storage area are discarded. The user can proceed or cancel the command that caused the pending discard. The contents of the temporary storage area can then be saved with the SAVE command.
DSQEC_RERUN_IPROC	None	01	Reruns the invocation procedure after the END command; values can be: **0** Suppresses rerun of the invocation procedure after the END command. **1** Reruns the invocation procedure after the END command. This value is the default. If you start QMF with an invocation procedure, set this variable to '0'; QMF terminates instead of rerunning the procedure.
DSQEC_RESET_RPT	None	31	Determines whether QMF prompts you when an incomplete DATA object in temporary storage might be affecting performance; possible values are: **0** Reset Report prompt panel is not displayed and QMF completes the running report. This value is the default value. **1** Reset Report prompt panel is displayed; this panel prompts you to complete or reset the currently running report before starting the new command. **2** Reset Report prompt panel is not displayed and QMF resets the currently running report.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_RUN_MQ	None	01	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, QMF ignores all statements after encountering the first semicolon. This value is the default. **1** Multiple SQL statements are supported. A semicolon must be placed at the end of each statement except the last. **Restrictions:** Although a SELECT statement can be included with other statements in a query, only one SELECT statement can be included per query. CALL and CREATE PROCEDURE statements must be used alone in an SQL query.
DSQEC_SAV_ACCELNM	None	128	Specifies the name of the default accelerator to be used when creating accelerator-only tables from SAVE DATA, IMPORT TABLE and RUN QUERY to TABLE commands. This variable is referenced only if the ACCELERATOR keyword is not specified. Although you can set this global variable to a blank, do not set it to blank if the DSQEC_SAV_ALLOWED global variable is set to '4'.
DSQEC_SAV_ACCELDB	None	08	Contains a data base name to be used on creation of new accelerator only tables in Db2 for z/OS data bases. This variable is referenced only when an accelerator only table is being created from the SAVE DATA, IMPORT TABLE and RUN QUERY with TABLE keyword commands. When this variable is not blank, the IN DATABASE clause will be specified on the CREATE TABLE statement and accelerator only tables will be created in the specified database. The default for this variable is blank. Note that when creating accelerator only tables via the SAVE DATA, IMPORT TABLE and RUN QUERY with TABLE keyword commands, QMF does not reference the user's profile SPACE value.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_SAV_ALLOWED	None	01	Controls whether users save data to a new table in the database or in an accelerator using the QMF SAVE DATA, RUN QUERY to TABLE, or IMPORT TABLE commands. Except for option 0, this field does not influence the location of existing tables that the replaced data is in or teh data is appended to. Existing tables are replaced or appended to in the database or accelerator regardless of the setting of this variable. Valid values for this global variable are: **0 - Disable Save Data** Users cannot issue the QMF SAVE DATA, RUN QUERY to TABLE, or IMPORT TABLE commands to save data to a table in the database or accelerator. An error message will be displayed and no data will be saved. **1 - Enable Save Data to database tables only** Users can save data to a table in the database by using the QMF SAVE DATA, RUN QUERY to TABLE, or IMPORT TABLE commands. Users cannot save data to accelerator-only tables. This setting is the default. **2 - Enable Save Data to accelerator only tables only** Users can save data to an accelerator-only table by using the QMF SAVE DATA, RUN QUERY to TABLE, or IMPORT TABLE commands. Users cannot save data to database tables. The DSQEC_SAV_ACCELNM global variable contains the default name of the accelerator but can be overridden by the ACCELERATOR keyword. **3 - Enable Save Data to either database or accelerator only tables (database default)** Users can save data either to a table in the database or to an accelerator-only table by using the QMF SAVE DATA, RUN QUERY to TABLE, or IMPORT TABLE commands. If no command keyword overrides are present, such as SPACE or ACCEL, tables are saved in the database. **4 - Enable Save Data to either database or accelerator only tables (accelerator default)** Users can save data either to a table in the database or to an accelerator-only table by using the QMF SAVE DATA, RUN QUERY to TABLE, or IMPORT TABLE commands. If no command keyword overrides are present, such as SPACE or ACCELERATOR, tables are saved in the accelerator. When this option is chosen, the DSQEC_SAV_ACCELNM global variable must contain the name of the accelerator.
DSQEC_SAV_LOADER	None	01	Allows the Db2 LOAD utility (cross-loader feature) to be used when using the RUN QUERY with TABLE keyword. **0** Run Query with TABLE keyword will not use the Db2 LOAD utility (cross-loader feature) to save the data. (Default) **1** Run Query with TABLE keyword will use the Db2 LOAD utility (cross-loader feature) to save the data.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_SAV_LOGCTL	None	02	The Db2 Load Utility cross-loader feature returns errors in a result set. DSQEC_SAV_LOGCTL controls the amount of output returned from the cross-loader that is saved by QMF. **-1** QMF will not save any results. **0** QMF will save all results. **1-16** QMF will save results with a return code greater than what you entered or higher. Examples: • DSQEC_SAV_LOGCTL is set to 4 and the LOADER returns a RC of 8. The result set will be saved. • DSQEC_SAV_LOGCTL is set to 8 and the LOADER returns a RC of 4. The result set will not be saved.
DSQEC_SAV_LOGTABLE	None		The name of the table to which QMF saves result sets returned from the cross-loader. *Q.ERROR_LOG* is the default name and should be created when QMF is installed. This is the QMF message error log. The name can be a one or two-part name in the form of: *USERID.TABLENAME* If left blank the result set will not be saved. If the user enters a name other than Q.ERROR_LOG, the table must exist. QMF will not create the table. No save will be done. An entry will be made in the QMF trace indicating the result set was not saved. It is also recommended that the error log be in a different table space than the one the data is being saved it otherwise QMF may not be able to have the result set if the utility is terminated.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_SESSGLV_SAV	None	01	Controls whether user input in some data entry fields on some panels is saved within and across QMF sessions. User input is saved as session variables that are stored in the Q.GLOBAL_VARS table as global variables that are named with a DXY prefix. The DSQEC_SESSGLV_SAV global variable is checked throughout the session, as well as when QMF starts and exits. The valid values are: **0** If this setting is specified when QMF starts, all session variables are deleted from the Q.GLOBAL_VARS table. If this setting is specified during a QMF session, all session variables are deleted from storage. No session variables are saved for the remainder of the current session unless this setting is changed to 1 or 2. If this setting is specified when QMF exits, all session variables are deleted from the Q.GLOBAL_VARS table, which means that no user input persists to the next QMF session. This is the default value. **1** If this setting is specified when QMF starts, all session variables for the user are restored from the Q.GLOBAL_VARS table. If this setting is specified during a QMF session, session variables are saved for the remainder of the current session. For example, if you enter values in the LIST Command Prompt panel, exit the LIST panel, and return to that panel within the same session, those fields are populated with the values that you previously entered. If this setting is specified when QMF exits, all session variables that were created or changed by the user during the current session are discarded and not saved to the Q.GLOBAL_VARS table. All session variable values that existed in the Q.GLOBAL_VARS table before the current session remain unchanged. You can use this option, for example, to reinitialize the same session variable values at the start of each QMF session. When the next QMF session is started, the value reverts to 0 unless it is overridden by an initial global variable that set by an administrator. **2** If this setting is specified when QMF starts, all session variables for the user are restored from the Q.GLOBAL_VARS table. If this setting is specified during a QMF session, session variables are saved for the remainder of the current session unless this setting is changed to 0. For example, if you enter values in the LIST Command Prompt panel, exit the LIST panel, and return to that panel within the same session, those fields are populated with the values that you previously entered. If this setting is specified when QMF exits, all session variables are saved to the Q.GLOBAL_VARS table, which means that any user input that was saved during the session also persists to the next QMF session. This parameter applies to most fields on command prompt panels that are accessed through the following commands: CONNECT, CONVERT, DISPLAY, DRAW, EDIT, ERASE, EXPORT, IMPORT, LIST, PRINT, RESET, RUN, SAVE, SET, and SHOW.
DSQEC_SHARE	None	31	Specifies the default value for the SHARE parameter; possible values are: **0** Do not share data with other users. **1** Share data with other users.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_SP_RS_NUM	None	04	Indicates which result set returned by a stored procedure is used to create the report. Possible values are: **0** Ignores result sets. **1** Returns the first result set. **2** Returns the second result set. ***n*** Returns the *n*th result set. The maximum value for *n* is 32.
DSQEC_SPAC_OVRIDE	None	01	Specifies whether users can override the default table space that is specified in the QMF profile. Valid values for this global variable are: **0 - Disable Space Keyword Option** Users cannot issue the SAVE DATA, RUN QUERY to TABLE, or IMPORT TABLE commands with the SPACE keyword option. **1 - Enable Space Keyword Option** Users can issue the SAVE DATA, RUN QUERY to TABLE, or IMPORT TABLE commands with the space keyword option. This setting is the default.
DSQEC_SQLQRYSZ_2M	None	01	Controls whether SQL queries greater than 32,767 bytes (32 KB) in length are supported by the RUN QUERY command. **0** SQL queries directed to Db2 for z/OS, DB2 for iSeries, and Db2 for Linux, UNIX, and Windows databases are limited to 32,767 bytes (32 KB). This value is the default. **1** SQL queries can be greater than 32 KB. 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 VSE and VM are limited to 8 KB. Additional customization might be required to run queries larger than 32 KB from QMF for CICS.
DSQEC_TABS_LDB2	None	31	View for retrieving lists of tables and views at the current server, if it is Db2 for z/OS or Db2 for Linux, UNIX, and Windows
DSQEC_TABS_RDB2	None	31	View for retrieving lists of tables and views at remote Db2 subsystems.
DSQEC_TABS_SQL	None	31	View for retrieving lists of tables and views for a DB2 for VSE and VM database.
DSQEC_TRACE_LIMIT	None	31	Limits the amount of trace output to the specified number of bytes. The valid range is 0 - 2147483647. This variable can be used to reduce the size of QMF trace output. This global variable is typically set as directed by IBM Software Support.

Table 45. Global variables that control how commands and procedures are executed (continued)

Callable interface variable name	Command interface variable name	Length	Description
DSQEC_TRACE_MODULE	None	54	Contains the names of QMF modules to be traced. Up to 6 modules can be specified, separated by commas. After module names are specified in the global variable, initiate the trace by issuing the SET PROFILE command with the TRACE keyword to set to ALL. Example: SET PROFILE (TRACE=ALL **Note:** If modules are specified via the SET GLOBAL command from the command line, the module names must be enclosed in single quotes.
DSQEC_TWO_GB_ROW	None	01	Controls the length of rows returned in QMF reports. Use one of the following values: **0** Limits the length of a data row in a QMF report to 32 KB, unless the report contains an XML or LOB column. **1** Allows the length of a data row to be greater than 32 KB, up to a maximum length of 2 GB. **Important:** • Regardless of the DSQEC_TWO_GB_ROW global variable setting, up to 2 GB of XML, CLOB, or BLOB data, and up to 1 GB of DBCLOB data can be displayed by default. However, the maximum length of a LOB row can be restricted by the DSQEC_LOB_COLMAX global variable. • Regardless of the DSQEC_TWO_GB_ROW global variable setting, a single table cannot have a maximum record size that is greater than the page size. Db2 stores records within pages that are 4 KB, 8 KB, 16 KB, or 32 KB in size. So, the maximum length of a data row that can be displayed remains at 32 KB when you display or select data from a single table. If you display or select data from a view that joins two or more tables, the row length can be up to 2 GB. Because of these page size considerations, the length of a data row in a QMF report that can be saved with the SAVE DATA command is also limited to 32 KB. The ability to save LOB data is controlled by the DSQEC_LOB_SAVE global variable.

Table 45. Global variables that control how commands and procedures are executed (continued)			
Callable interface variable name	**Command interface variable name**	**Length**	**Description**
DSQEC_USERGLV_SAV	None	01	Determines whether global variables that were created or changed by the user, including those that start with "DSQ," are saved when the QMF session ends. Values that are to be saved are stored in the Q.GLOBAL_VARS table and associated with the user ID of the session. If the values are saved, they are restored at the start of the user's next QMF session. The valid values are: **0** When QMF exits, all global variables are deleted from the Q.GLOBAL_VARS table, and no global variables from the current session are saved to the Q.GLOBAL_VARS table. This is the default value. **1** When QMF exits, all global variables that were created or changed by the user during the current session are discarded and not saved to the Q.GLOBAL_VARS table. All global variable values that were already in the Q.GLOBAL_VARS table remain as they were prior to the current QMF session. You can use this option, for example, to re-initialize the same global variable values at the start of each QMF session. When the next QMF session is started, the value reverts to 0 unless it is overridden by an initial global variable that set by an administrator. **2** When QMF exits, all global variables that were created or changed by the user are saved to the Q.GLOBAL_VARS table. When the user starts QMF again, global variables that were saved from the user's previous session are restored. Any values that were defined by an administrator in the Q.GLOBAL_VARS table are superseded by the user's values unless the variable was defined as read-only.

Related reference

RUN
The RUN command runs queries or procedures from QMF temporary storage or from the database at the current location.

SAVE
The SAVE command saves in the database at the current location objects that are currently in QMF temporary storage.

PREPARE statement for Db2See the information about the concurrent-access-resolution attribute of the PREPARE statement.

Global variables that store results of CONVERT QUERY

DSQQC global variables reflect the results of a CONVERT QUERY command. None of these global variables can be modified by the SET GLOBAL command.

Table 46. Global variables that reflect the results of a CONVERT QUERY command			
Callable interface variable name	**Command interface variable name**	**Length**	**Description**
DSQQC_LENGTH_*nnn*	DSQCL*nnn*	05	Length of converted result *nnn*.
DSQQC_QRY_COUNT	DSQCQCNT	03	Number of queries in converted result; value must always be '1' unless the original query is a QBE I. or U. query.

Table 46. Global variables that reflect the results of a CONVERT QUERY command (continued)			
Callable interface variable name	**Command interface variable name**	**Length**	**Description**
DSQQC_QRY_LANG	DSQCQLNG	01	Language of converted query; values can be: **1** for SQL **2** for QBE **3** for Prompted
DSQQC_QRY_TYPE	DSQCQTYP	Not specified	First word in converted results.
DSQQC_RESULT_*nnn*	DSQCQ*nnn*	Not specified	*nnn*

Global variables that show RUN QUERY error message information

DSQQM global variables store the results of a RUN QUERY command. None of these global variables can be modified by the SET GLOBAL command.

Table 47. Global variables that store the results of a RUN QUERY command			
Callable interface variable name	**Command interface variable name**	**Length**	**Description**
DSQQM_MESSAGE	DSQCIQMG	80	Text of query message.
DSQQM_MESSAGE_ALL	DSQCIQMA	360	Complete query message text.
DSQQM_MSG_HELP	DSQCIQID	08	ID of message help panel.
DSQQM_MSG_NUMBER	DSQCIQNO	08	Message number.
DSQQM_SQL_RC	DSQCISQL	16	The SQLCODE from the last command or query.
DSQQM_SQL_STATE	None	05	The SQLSTATE associated with the SQLCODE in DSQQM_SQL_RC, if SQLSTATE is returned by the database manager.
DSQQM_SUB_TXT_*nn*	DSQCIQ*nn*	20	Substitution value *nn*.
DSQQM_SUBST_VARS	DSQCIQ00	04	Number of substitution variables.

Global variables that store panel input values

DXY global variables store the values that users enter in data entry fields if the DSQEC_SESSGLV_SAV global variable is set to 1 or 2. Input in only some data entry fields on some panels is saved. User input for fields that are not listed in the following table are not saved, regardless of the DSQEC_SESSGLV_SAV global variable setting.

All of these global variables can be modified by the SET GLOBAL command. However, use caution when changing or deleting these variables because doing so changes the values that are generated on command prompt panels.

Table 48. Mapping between DXY global variables and panel field names			
Global variable name (where *n* is a national language identifier and *ln* is an ID associated with a line of a multiline field)	**Range of *ln* values**	**Command**	**Field name**
DXY*n*PCO1_*ln*	01 - 03	CONNECT	User
DXY*n*PCO1_05	–	CONNECT	Location

Table 48. Mapping between DXY global variables and panel field names (continued)

Global variable name (where *n* is a national language identifier and *ln* is an ID associated with a line of a multiline field)	**Range of *ln* values**	**Command**	**Field name**
DXY*n*PC03_01	–	CONNECT (CICS)	Location
DXY*n*PCNV_*ln*	02 - 07	CONVERT	Name
DXY*n*PDSP_*ln*	02 - 07	DISPLAY	Name
DXY*n*PDSP_*ln*_01	02 - 07	DISPLAY QUERY	Name
DXY*n*PDSP_*ln*_02	02 - 07	DISPLAY PROC	Name
DXY*n*PDSP_*ln*_03	02 - 07	DISPLAY FORM	Name
DXY*n*PDSP_*ln*_05	02 - 07	DISPLAY REPORT	Name
DXY*n*PDSP_*ln*_07	02 - 07	DISPLAY CHART	Name
DXY*n*PDSP_*ln*_08	02 - 07	DISPLAY TABLE	Name
DXY*n*PDRS_*ln*	01 - 06	DRAW	Name
DXY*n*PDRS_07	–	DRAW	Type
DXY*n*PDRS_08	–	DRAW	Identifier
DXY*n*PEDT_01	–	EDIT	Type
DXY*n*PED1_*ln*	01 - 06	EDIT (QUERY or PROC)	Name
DXY*n*PED2_*ln*	01 - 06	EDIT TABLE	Name
DXY*n*PED2_07	–	EDIT TABLE	Mode
DXY*n*PED3_*ln*	01 - 06	EDIT (QUERY or PROC), then make changes and exit without saving.	Name
DXY*n*PED3_09	–	EDIT (QUERY or PROC), then make changes and exit without saving.	Comment
DXY*n*PED3_*ln*	10 - 12	EDIT (QUERY or PROC), then make changes and exit without saving.	Folder
DXY*n*PERA_*ln*	02 - 07	ERASE	Name
DXY*n*PERA_*ln*_01	02 - 07	ERASE QUERY	Name
DXY*n*PERA_*ln*_02	02 - 07	ERASE PROC	Name
DXY*n*PERA_*ln*_03	02 - 07	ERASE FORM	Name
DXY*n*PERA_*ln*_08	02 - 07	ERASE TABLE	Name
DXY*n*PEXM_*ln*	02 - 07	EXPORT	Name
DXY*n*PEXM_*ln*_01	02 - 07	EXPORT QUERY	Name
DXY*n*PEXM_*ln*_02	02 - 07	EXPORT PROC	Name

Table 48. Mapping between DXY global variables and panel field names (continued)

Global variable name (where *n* is a national language identifier and *ln* is an ID associated with a line of a multiline field)	**Range of *ln* values**	**Command**	**Field name**
DXY*n*PEXM_*ln*_03	02 - 07	EXPORT FORM	Name
DXY*n*PEXM_*ln*_05	02 - 07	EXPORT REPORT	Name
DXY*n*PEXM_*ln*_06	02 - 07	EXPORT DATA	Name
DXY*n*PEXM_*ln*_07	02 - 07	EXPORT CHART	Name
DXY*n*PEXM_*ln*_08	02 - 07	EXPORT TABLE	Name
DXY*n*PXM1_*ln*	01 - 05	EXPORT, then Enter (in TSO)	To
DXY*n*PXM1_*ln*_01	01 - 05	EXPORT QUERY, then Enter (in TSO)	To
DXY*n*PXM1_*ln*_02	01 - 05	EXPORT PROC, then Enter (in TSO)	To
DXY*n*PXM1_06	–	EXPORT, then Enter (in TSO)	Member
DXY*n*PXM1_06_01	–	EXPORT QUERY, then Enter (in TSO)	Member
DXY*n*PXM1_06_02	–	EXPORT PROC, then Enter (in TSO)	Member
DXY*n*PXM2_01_07	–	EXPORT CHART, then Enter (in TSO)	Member
DXY*n*PXM3_*ln*_05	01 - 05	EXPORT REPORT, then Enter (in TSO)	To
DXY*n*PXM3_06_05	–	EXPORT REPORT, then Enter (in TSO)	Member
DXY*n*PXM3_08_05	–	EXPORT REPORT, then Enter (in TSO)	Dataformat
DXY*n*PXM4_*ln*_06	01 - 05	EXPORT DATA, then Enter (in TSO)	To
DXY*n*PXM4_06_06	–	EXPORT DATA, then Enter (in TSO)	Member
DXY*n*PXM4_08_06	–	EXPORT DATA, then Enter (in TSO)	Dataformat
DXY*n*PXM4_09_06	–	EXPORT DATA, then Enter (in TSO)	Outputmode
DXY*n*PXM4_10_06	–	EXPORT DATA, then Enter (in TSO)	Header
DXY*n*PXM4_*ln*_08	01-05	EXPORT TABLE, then Enter (in TSO)	To

Table 48. Mapping between DXY global variables and panel field names (continued)

Global variable name (where *n* is a national language identifier and *ln* is an ID associated with a line of a multiline field)	**Range of *ln* values**	**Command**	**Field name**
DXY*n*PXM4_06_08	-	EXPORT TABLE, then Enter (in TSO)	Member
DXY*n*PXM4_08_08	-	EXPORT TABLE, then Enter (in TSO)	Dataformat
DXY*n*PXM4_09_08	-	EXPORT TABLE, then Enter (in TSO)	Outputmode
DXY*n*PXM4_10_08	-	EXPORT TABLE, then Enter (in TSO)	Header
DXY*n*PXM5_*ln*_03	01 - 05	EXPORT FORM, then Enter (in TSO)	To
DXY*n*PXM5_06_03	–	EXPORT FORM, then Enter (in TSO)	Member
DXY*n*PXM5_08_03	–	EXPORT FORM, then Enter (in TSO)	Language
DXY*n*PXC1_01	–	EXPORT, then Enter (in CICS)	Queue Name
DXY*n*PXC1_01_01	–	EXPORT QUERY, then Enter (in CICS)	Queue Name
DXY*n*PXC1_01_02	–	EXPORT PROC, then Enter (in CICS)	Queue Name
DXY*n*PXC1_02	–	EXPORT, then Enter (in CICS)	Queue Type
DXY*n*PXC1_02_01	–	EXPORT QUERY, then Enter (in CICS)	Queue Type
DXY*n*PXC1_02_02	–	EXPORT PROC, then Enter (in CICS)	Queue Type
DXY*n*PXC1_04	–	EXPORT, then Enter (in CICS)	Suspend
DXY*n*PXC1_04_01	–	EXPORT QUERY, then Enter (in CICS)	Suspend
DXY*n*PXC1_04_02	–	EXPORT PROC, then Enter (in CICS)	Suspend
DXY*n*PXC3_01_05	–	EXPORT REPORT, then Enter (in CICS)	Queue Name
DXY*n*PXC3_02_05	–	EXPORT REPORT, then Enter (in CICS)	Queue Type
DXY*n*PXC3_04_05	–	EXPORT REPORT, then Enter (in CICS)	Suspend

Table 48. Mapping between DXY global variables and panel field names (continued)

Global variable name (where *n* is a national language identifier and *ln* is an ID associated with a line of a multiline field)	**Range of *ln* values**	**Command**	**Field name**
DXY*n*PXC3_05_05	–	EXPORT REPORT, then Enter (in CICS)	Dataformat
DXY*n*PXC4_01_06	–	EXPORT DATA, then Enter (in CICS)	Queue Name
DXY*n*PXC4_02_06	–	EXPORT DATA, then Enter (in CICS)	Queue Type
DXY*n*PXC4_04_06	–	EXPORT DATA, then Enter (in CICS)	Suspend
DXY*n*PXC4_05_06	–	EXPORT DATA, then Enter (in CICS)	Dataformat
DXY*n*PXC4_06_06	–	EXPORT DATA, then Enter (in CICS)	Outputmode
DXY*n*PXC4_07_06	–	EXPORT DATA, then Enter (in CICS)	Header
DXY*n*PXC5_01_03	–	EXPORT FORM, then Enter (in CICS)	Queue Name
DXY*n*PXC5_02_03	–	EXPORT FORM, then Enter (in CICS)	Queue Type
DXY*n*PXC5_04_03	–	EXPORT FORM, then Enter (in CICS)	Suspend
DXY*n*PXC5_05_03	–	EXPORT FORM, then Enter (in CICS)	Language
DXY*n*PIMM_*ln*	02 - 07	IMPORT (in TSO)	Name
DXY*n*PIMM_*ln*_01	02 - 07	IMPORT QUERY (in TSO)	Name
DXY*n*PIMM_*ln*_02	02 - 07	IMPORT PROC (in TSO)	Name
DXY*n*PIMM_*ln*_03	02 - 07	IMPORT FORM (in TSO)	Name
DXY*n*PIMM_*ln*_06	02 - 07	IMPORT DATA (in TSO)	Name
DXY*n*PIMM_*ln*_08	02 - 07	IMPORT TABLE (in TSO)	Name
DXY*n*PIMM_*ln*	08 - 12	IMPORT (in TSO)	From
DXY*n*PIMM_*ln*_01	08 - 12	IMPORT QUERY (in TSO)	From
DXY*n*PIMM_*ln*_02	08 - 12	IMPORT PROC (in TSO)	From
DXY*n*PIMM_*ln*_03	08 - 12	IMPORT FORM (in TSO)	From
DXY*n*PIMM_*ln*_06	08 - 12	IMPORT DATA (in TSO)	From
DXY*n*PIMM_*ln*_08	08 - 12	IMPORT TABLE (in TSO)	From
DXY*n*PIMM_13	–	IMPORT (in TSO)	Member

Table 48. Mapping between DXY global variables and panel field names (continued)

Global variable name (where *n* is a national language identifier and *ln* is an ID associated with a line of a multiline field)	**Range of *ln* values**	**Command**	**Field name**
DXY*n*PIMM_13_01	–	IMPORT QUERY (in TSO)	Member
DXY*n*PIMM_13_02	–	IMPORT PROC (in TSO)	Member
DXY*n*PIMM_13_03	–	IMPORT FORM (in TSO)	Member
DXY*n*PIMM_13_06	–	IMPORT DATA (in TSO)	Member
DXY*n*PIMM_13_08	–	IMPORT TABLE (in TSO)	Member
DXY*n*PIQF_03	–	IMPORT, then Enter (in TSO)	Comment
DXY*n*PIQF_03_01	–	IMPORT QUERY, then Enter (in TSO)	Comment
DXY*n*PIQF_03_02	–	IMPORT PROC, then Enter (in TSO)	Comment
DXY*n*PIQL_03_03	–	IMPORT FORM, then Enter (in TSO)	Comment
DXY*n*PIQL_04_03	–	IMPORT FORM, then Enter (in TSO)	Language
DXY*n*PITB_02_08	–	IMPORT TABLE, then Enter (in TSO)	Comment
DXY*n*PITB_04_08	–	IMPORT TABLE, then Enter (in TSO)	Space
DXY*n*PITB_*ln*_08	05 - 07	IMPORT TABLE, then Enter (in TSO)	Accelerator
DXY*n*PIMC_*ln*	02 - 07	IMPORT (in CICS)	Name
DXY*n*PIMC_*ln*_01	02 - 07	IMPORT QUERY (in CICS)	Name
DXY*n*PIMC_*ln*_02	02 - 07	IMPORT PROC (in CICS)	Name
DXY*n*PIMC_*ln*_03	02 - 07	IMPORT FORM (in CICS)	Name
DXY*n*PIMC_*ln*_06	02 - 07	IMPORT DATA (in CICS)	Name
DXY*n*PIMC_*ln*_08	02 - 07	IMPORT TABLE (in CICS)	Name
DXY*n*PIMC_08	–	IMPORT (in CICS)	Queue Name
DXY*n*PIMC_08_01	–	IMPORT QUERY (in CICS)	Queue Name
DXY*n*PIMC_08_02	–	IMPORT PROC (in CICS)	Queue Name
DXY*n*PIMC_08_03	–	IMPORT FORM (in CICS)	Queue Name
DXY*n*PIMC_08_06	–	IMPORT DATA (in CICS)	Queue Name
DXY*n*PIMC_08_08	–	IMPORT TABLE (in CICS)	Queue Name

Table 48. Mapping between DXY global variables and panel field names (continued)

Global variable name (where *n* is a national language identifier and *ln* is an ID associated with a line of a multiline field)	**Range of *ln* values**	**Command**	**Field name**
DXY*n*PIMC_09	–	IMPORT (in CICS)	Queue Type
DXY*n*PIMC_09_01	–	IMPORT QUERY (in CICS)	Queue Type
DXY*n*PIMC_09_02	–	IMPORT PROC (in CICS)	Queue Type
DXY*n*PIMC_09_03	–	IMPORT FORM (in CICS)	Queue Type
DXY*n*PIMC_09_06	–	IMPORT DATA (in CICS)	Queue Type
DXY*n*PIMC_09_08	–	IMPORT TABLE (in CICS)	Queue Type
DXY*n*PIMC_10	–	IMPORT (in CICS)	Suspend
DXY*n*PIMC_10_01	–	IMPORT QUERY (in CICS)	Suspend
DXY*n*PIMC_10_02	–	IMPORT PROC (in CICS)	Suspend
DXY*n*PIMC_10_03	–	IMPORT FORM (in CICS)	Suspend
DXY*n*PIMC_10_06	–	IMPORT DATA (in CICS)	Suspend
DXY*n*PIMC_10_08	–	IMPORT TABLE (in CICS)	Suspend
DXY*n*PLST_01	–	LIST (QUERIES, PROCS, FORMS, ANALYTICS, QMF, TABLES, or ALL)	Type
DXY*n*PLST_*ln*	02 - 04	LIST (QUERIES, PROCS, FORMS, ANALYTICS, QMF, TABLES, or ALL)	Owner
DXY*n*PLST_*ln*	05 - 07	LIST (QUERIES, PROCS, FORMS, ANALYTICS, QMF, TABLES, or ALL)	Name
DXY*n*PLST_08	–	LIST (QUERIES, PROCS, FORMS, ANALYTICS, QMF, TABLES, or ALL)	Location
DXY*n*PPRT_*ln*	02 - 07	PRINT (in TSO)	Name
DXY*n*PPRT_*ln*_01	02 - 07	PRINT QUERY (in TSO)	Name
DXY*n*PPRT_*ln*_02	02 - 07	PRINT PROC (in TSO)	Name
DXY*n*PPRT_*ln*_03	02 - 07	PRINT FORM (in TSO)	Name
DXY*n*PPRT_*ln*_04	02 - 07	PRINT PROFILE (in TSO)	Name
DXY*n*PPRT_*ln*_05	02 - 07	PRINT REPORT (in TSO)	Name
DXY*n*PPRT_*ln*_07	02 - 07	PRINT CHART (in TSO)	Name
DXY*n*PPRT_*ln*_08	02 - 07	PRINT TABLE (in TSO)	Name

Table 48. Mapping between DXY global variables and panel field names (continued)

Global variable name (where *n* is a national language identifier and *ln* is an ID associated with a line of a multiline field)	**Range of *ln* values**	**Command**	**Field name**
DXY*n*PPR2_01_07	–	PRINT CHART, then Enter (in TSO)	Printer
DXY*n*PPR3_01_01	–	PRINT QUERY, then Enter (in TSO)	Printer
DXY*n*PPR3_01_02	–	PRINT PROC, then Enter (in TSO)	Printer
DXY*n*PPR3_01_03	–	PRINT FORM, then Enter (in TSO)	Printer
DXY*n*PPR3_01_04	–	PRINT PROFILE, then Enter (in TSO)	Printer
DXY*n*PPR3_01_08	–	PRINT TABLE, then Enter (in TSO)	Printer
DXY*n*PPR4_01_05	–	PRINT REPORT, then Enter (in TSO)	Printer
DXY*n*PPR5_*ln*	02 - 07	PRINT (in CICS)	Name
DXY*n*PPR5_*ln*_01	02 - 07	PRINT QUERY (in CICS)	Name
DXY*n*PPR5_*ln*_02	02 - 07	PRINT PROC (in CICS)	Name
DXY*n*PPR5_*ln*_03	02 - 07	PRINT FORM (in CICS)	Name
DXY*n*PPR5_*ln*_04	02 - 07	PRINT PROFILE (in CICS)	Name
DXY*n*PPR5_*ln*_05	02 - 07	PRINT REPORT (in CICS)	Name
DXY*n*PPR5_*ln*_07	02 - 07	PRINT CHART (in CICS)	Name
DXY*n*PPR5_*ln*_08	02 - 07	PRINT TABLE (in CICS)	Name
DXY*n*PPR5_08	–	PRINT (in CICS)	Queue Name
DXY*n*PPR5_08_01	–	PRINT QUERY (in CICS)	Queue Name
DXY*n*PPR5_08_02	–	PRINT PROC (in CICS)	Queue Name
DXY*n*PPR5_08_03	–	PRINT FORM (in CICS)	Queue Name
DXY*n*PPR5_08_04	–	PRINT PROFILE (in CICS)	Queue Name
DXY*n*PPR5_08_05	–	PRINT REPORT (in CICS)	Queue Name
DXY*n*PPR5_08_07	–	PRINT CHART (in CICS)	Queue Name
DXY*n*PPR5_08_08	–	PRINT TABLE (in CICS)	Queue Name
DXY*n*PPR5_09	–	PRINT (in CICS)	Queue Type
DXY*n*PPR5_09_01	–	PRINT QUERY (in CICS)	Queue Type

Table 48. Mapping between DXY global variables and panel field names (continued)

Global variable name (where *n* is a national language identifier and *ln* is an ID associated with a line of a multiline field)	**Range of *ln* values**	**Command**	**Field name**
DXY*n*PPR5_09_02	–	PRINT PROC (in CICS)	Queue Type
DXY*n*PPR5_09_03	–	PRINT FORM (in CICS)	Queue Type
DXY*n*PPR5_09_04	–	PRINT PROFILE (in CICS)	Queue Type
DXY*n*PPR5_09_05	–	PRINT REPORT (in CICS)	Queue Type
DXY*n*PPR5_09_07	–	PRINT CHART (in CICS)	Queue Type
DXY*n*PPR5_09_08	–	PRINT TABLE (in CICS)	Queue Type
DXY*n*PPR5_10	–	PRINT (in CICS)	Suspend
DXY*n*PPR5_10_01	–	PRINT QUERY (in CICS)	Suspend
DXY*n*PPR5_10_02	–	PRINT PROC (in CICS)	Suspend
DXY*n*PPR5_10_03	–	PRINT FORM (in CICS)	Suspend
DXY*n*PPR5_10_04	–	PRINT PROFILE (in CICS)	Suspend
DXY*n*PPR5_10_05	–	PRINT REPORT (in CICS)	Suspend
DXY*n*PPR5_10_07	–	PRINT CHART (in CICS)	Suspend
DXY*n*PPR5_10_08	–	PRINT TABLE (in CICS)	Suspend
DXY*n*PRNM_*ln*	02 - 07	RENAME	Old Name
DXY*n*PRNM_*ln*	08 - 10	RENAME	New Name
DXY*n*PRST_01	–	RESET	Type
DXY*n*PRSG_01	–	RESET GLOBAL	Enter ALL ...
DXY*n*PRSG_*ln*	02 - 11	RESET GLOBAL	Global variable name
DXY*n*PRUN_*ln*	02 - 07	RUN	Name
DXY*n*PRUN_*ln*_01	02 - 07	RUN QUERY	Name
DXY*n*PRUN_*ln*_02	02 - 07	RUN PROC	Name
DXY*n*PRU3_*ln*	01 - 06	RUN QUERY, then Enter	Form
DXY*n*PRU3_08	–	RUN QUERY, then Enter	Rowlimit
DXY*n*PRU3_*ln*	09 - 14	RUN QUERY, then Enter	Analytic
DXY*n*PRU3_*ln*	15 - 20	RUN QUERY, then Enter	Table
DXY*n*PRU3_22	–	RUN QUERY, then Enter	Comment
DXY*n*PRU3_23	–	RUN QUERY, then Enter	Space
DXY*n*PRU3_*ln*	24 - 26	RUN QUERY, then Enter	Accelerator
DXY*n*PRU4_01	–	RUN PROC, then Enter	Arg

Table 48. Mapping between DXY global variables and panel field names (continued)

Global variable name (where *n* is a national language identifier and *ln* is an ID associated with a line of a multiline field)	**Range of *ln* values**	**Command**	**Field name**
DXY*n*PSAV_01	–	SAVE	Type
DXY*n*PSA2_*ln*	01 - 06	SAVE DATA	Name
DXY*n*PSA2_08	–	SAVE DATA	Comment
DXY*n*PSA2_10	–	SAVE DATA	Space
DXY*n*PSA2_*ln*	11 - 13	SAVE DATA	Accelerator
DXY*n*PSA3_*ln*_01	01 - 06	SAVE QUERY	Name
DXY*n*PSA3_*ln*_02	01 - 06	SAVE PROC	Name
DXY*n*PSA3_09_01	–	SAVE QUERY	Comment
DXY*n*PSA3_09_02	–	SAVE PROC	Comment
DXY*n*PSA3_*ln*_01	10 - 12	SAVE QUERY	Folder
DXY*n*PSA3_*ln*_02	10 - 12	SAVE PROC	Folder
DXY*n*PSA4_*ln*	01 - 06	SAVE FORM	Name
DXY*n*PSA4_09	–	SAVE FORM	Comment
DXY*n*PSA4_*ln*	11 - 13	SAVE FORM	Folder
DXY*n*PSET_01	–	SET	Type
DXY*n*PSGL_*ln*	01 - 19 (even numbers)	SET GLOBAL	Var
DXY*n*PSGL_*ln*	02 - 20 (odd numbers)	SET GLOBAL	Value
DXY*n*PSHO_01	–	SHOW	Enter the name ...

Appendix C. QMF functions that require specific support

Support for these functions varies with the database or environment.

Functions that vary according to database type

Support for these functions varies by database.

Table 49. Functions that require the support of specific database management systems

Function supported	Db2 for z/OS	Db2 for Linux, UNIX, and Windows	DB2 for iSeries	DB2 for VSE and VM
Length of query statement supported	2 MB*	65 KB*	65 KB*	8 KB
Number of columns in SELECT statement	750	255	255	255
Importing single-precision floating point numbers	X			X
Long fields with LIKE statement	X			X
Database synonyms				X
Database aliases for tables or views	X	X	X	
SAVE=IMMEDIATE option available in Table Editor (supports CURSOR HOLD)	X	X	X	
Setting Db2 global variables	X	X		
QMF commands that include three-part names	Commands with three-part names can be initiated from this type of database. They can also be directed to this type of database unless QMF was started as a stored procedure.	Commands with three-part names can be directed to this type of server unless QMF was started as a stored procedure.	Commands with three-part names can be directed to this type of server unless QMF was started as a stored procedure.	Commands with three-part names cannot be directed to these server types.

* To activate support for SQL queries up to 2 MB on Db2 for z/OS databases and up to 65 KB on Db2 for Linux, UNIX, and Windows databases, set the DSQEC_SQLQRYSZ_2M global variable to 1 before running the query.

Functions not available in CICS

Certain functions are supported by TSO only.

The following functions are not available in CICS:

- Use of multiple thread support.
- Use of QMF Analytics for TSO
- Use of the QMF Enhanced Editor.
- Use of extended storage for spilling report data no longer needed in active storage; a spill file must be used instead
- Ability to start QMF as a Db2 for z/OS stored procedure
- Interfaces:
  - Command interface
  - Document interface
- Program parameters:
  - DSQSCMD (QMF callable interface only)
  - DSQSMTHD
  - DSQSPLAN
  - DSQSPRID
  - DSQSPTYP
  - DSQSRSTG
  - DSQSSUBS
- Commands:
  - BATCH (and its associated application)
  - DPRE (and its associated REXX exec)
  - EDIT QUERY
  - EDIT PROC
  - ISPF (and its associated application)
  - LAYOUT (and its associated application)
  - SET GLOBAL commands that reference the following global variable:
    - DSQEC_EXTND_STG
  - SHOW GLOBAL commands that reference the following global variables:
    - DSQEC_EXTND_STG
    - DSQAO_STO_PROC_INT
  - STATE (requires the command interface)
- Macros: GETQMF
- Form functions:
  - Report calculations or expressions that require REXX
  - Conditional formatting
  - Column definition
  - Locally defined edit codes TDL and TTL (for formatting dates and times, respectively)
- Procedures with logic (which require REXX)
- The ability to cancel transactions

- The ability to update data at remote locations (all tables and views at remote locations are read-only in QMF for CICS)
- External variables

Notices

This information was developed for products and services offered in the US. This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 jurisdictions 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.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at http://www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

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 of terms and acronyms

abnormal end of task (abend)
The termination of a task, job, or subsystem because of an error condition that recovery facilities cannot resolve during execution.

address space
The range of addresses available to a computer program or process. Address space can refer to physical storage, virtual storage, or both.

Advanced Program-to-Program Communication
See *APPC*.

aggregate function
Any of a group of functions that summarizes data in a column. They are requested with these usage codes on the form panels: AVERAGE, CALC, COUNT, FIRST, LAST, MAXIMUM, MINIMUM, STDEV, SUM, CSUM, PCT, CPCT, TPCT, TCPCT.

aggregation variable
An aggregation function that is placed in a report using the FORM.BREAK, FORM.CALC, FORM.DETAIL, or FORM.FINAL panels. Its value appears as part of the break footing, detail block text, or final text when the report is produced.

alias
An alternative name used to identify a table, view, database, or nickname. An alias can be used in SQL statements to refer to a table, view, or database in the same Db2 system or subsystem or in a remote Db2 system or subsystem.

APAR (Authorized Program Analysis Report)
A request for correction of a defect in a supported release of an program supplied by IBM.

APF (authorized program facility)
In a z/OS environment, a facility that permits the identification of programs that are authorized to use restricted functions.

API (application programming interface)
An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

application
One or more computer programs or software components that use QMF services to provide functionality in direct support of a specific business process or processes.

APPC (Advanced Program-to-Program Communication)
An implementation of the SNA LU 6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

application plan
The control structure that is produced during the bind process. The default name for the QMF Version 12.1 application plan is QMF1210.

application programming interface
See *API*.

application requester
The source of a request to a remote DRDA-enabled relational database management system (RDBMS). Only Db2 for z/OS databases can function as application requesters because this is the only type of database in which QMF can be started.

application server
The target of a request from an application requester. The database management system (DBMS) at the application server site services the request. Connectivity with remote servers is not supported when QMF for TSO is running as a Db2 for z/OS stored procedure.

argument
A value passed to or returned from a function or procedure at run time.

authorization identifier (authorization ID)
A character string that designates a set of privileges and can be used to verify authority. An authorization ID can represent an object, an individual user, an organizational group, a function, or a database role. QMF authenticates either the database authorization ID or, optionally, the QMF TSO logon ID, against the CREATOR column of the Q.PROFILES table during QMF initialization.

Authorized Program Analysis Report
See *APAR*.

Authorized program facility
See *APF*.

auxiliary table
A table that stores columns outside the table in which they are defined. See also *base table*.

base product
The English-language version of QMF, established when QMF is installed. Any other language environment is established after installation by installing the National Language Feature (NLF) associated with that language.

base table
A table that is created by the SQL CREATE TABLE statement and that holds persistent data.

binary string
A sequence of bytes that is not associated with a coded character set and therefore is never converted. For example, the BLOB data type is a binary string. See also *CCSID*.

bind
To convert the output from the DBMS precompiler to a usable control structure, such as an access plan, an application plan, or a package.

bit data
Data with a data type of CHAR or VARCHAR that is not associated with a coded character set and therefore is never converted.

buffer pool
An area of memory into which data pages are read and in which they are modified and held during processing. See also *address space*.

built-in function
A strongly typed, high-performance function that is integral to the Db2 database. A built-in function can be referenced in SQL statements anywhere that an expression is valid.

CAF (call attachment facility)
A Db2 for z/OS attachment facility for application programs that run in TSO or z/OS batch. The CAF is an alternative to the DSN command processor and provides greater control over the execution environment.

call attachment facility
See *CAF*.

callable interface
A programming interface that provides access to QMF objects and services.

cascade delete
A process by which the Db2 database manager enforces referential constraints by deleting all descendent rows of a deleted parent row.

catalog
A collection of tables and views that contains descriptions of objects such as tables, views, and indexes. See also *QMF object catalog*.

CCSID (coded character set identifier)
A 16-bit number that includes a specific set of encoding scheme identifiers, character set identifiers, code page identifiers, and other information that uniquely identifies the coded graphic-character

representation. Because QMF uses display services provided by GDDM, the GDDM application code page must agree with the CCSIDs in use for the database. See also *binary string*.

character string
A sequence of bytes that represents bit data, single-byte characters, or a mixture of single-byte and multibyte characters.

check constraint
A user-defined constraint that specifies the values that specific columns of a base table can contain. See also *constraint*.

CICS (Customer Information Control System)
An IBM licensed program that provides online transaction-processing services and management for business applications.

clause
In SQL, a distinct part of a statement in the language structure, such as a SELECT clause or a WHERE clause.

CM (Compatibility Mode)
An installation mode of QMF Version 8.1 and QMF Version 9.1 that limited owner and object names in the QMF object catalog to eight and 18 characters, respectively. See also *NFM*.

code page
A particular assignment of code points to graphic characters. Within a given code page, a code point can have only one specific meaning. A code page also identifies how undefined code points are handled.

coded character set identifier
See *CCSID*.

coexistence
The state during which two QMF releases exist in the same Db2 subsystem. QMF Version 12.1 can coexist with QMF Version 9.1 New Function Mode or QMF Version 8.1 New Function Mode only.

column
The vertical component of a database table. A column has a name and a particular data type (for example, character, decimal, or integer).

column function
See *aggregate function*.

column wrapping
The value formatting in a report where the values occupy several lines within a column. Column wrapping is often used when a column contains values whose length exceeds the column width, such as cases requiring the display of XML data.

command interface
An interface for issuing QMF commands. The command interface allows you to issue QMF commands from an ISPF dialog running under QMF. Using this interface, QMF communicates with the dialog through the ISPF variable pool.

command synonym
The verb or verb/object part of a site-defined command. After command synonyms are defined and activated in the QMF profile, users can enter the synonyms on the QMF command line as they do with regular QMF commands.

command synonym table
A table that stores one site-defined command in each row. You assign a set of command synonyms to a user by storing the name of this table in the user's profile.

comparison operator
In SQL, a symbol used in comparison expressions to specify a relationship between two values. Comparison operators are = (equal to), <> (not equal to), < (less than), > (greater than), <= (less than or equal to), and >= (greater than or equal to).

Compatibility Mode
See *CM*.

commit
To end a unit of work by releasing locks so that the database changes made by that unit of work can be perceived by other processes. This operation makes the data changes permanent.

concatenation
Joining two characters or strings to form one string.

connection
In data communication, an association established between entities for conveying information. See also *SQL connection*. Connectivity with remote servers is not supported when QMF for TSO is running as a Db2 for z/OS stored procedure.

constant
A language element that specifies an unchanging value. Constants are classified as string constants or numeric constants.

constraint
A rule that limits the values that can be inserted, deleted, or updated in a table.

control section
See *CSECT*.

control tables
A set of tables that QMF uses to store information about QMF objects and manage QMF operations. See also *QMF object catalog*.

correlated reference
A reference to a column of a table or view that is outside a subquery.

correlation name
An identifier specified and used within a single SQL statement as the exposed name for objects such as a table, view, table function reference, nested table expression, or data change table reference. Correlation names are useful in an SQL statement to allow two distinct references to the same base table and to allow an alternative name to be used to represent an object.

CSECT (control section)
The part of a program specified by the programmer to be a relocatable unit, all elements of which are to be loaded into adjoining main storage locations.

current location
The application server to which the QMF session is currently connected. After the connection is made, this server processes all SQL statements. When initializing QMF, the current location can be indicated using the DSQSDBNM startup parameter. Connectivity with remote servers is not supported when QMF for TSO is running as a Db2 for z/OS stored procedure.

current object
A QMF object that is held in temporary storage so that, with each use, it can be readily accessed without requiring database retrieval. There are seven temporary storage areas: QUERY, FORM, PROC, PROFILE, REPORT, DATA, and CHART. Users can navigate to all areas but the DATA area using the SHOW and DISPLAY commands. See also *temporary storage*.

cursor
A named control structure used by an application program to point to and select a row of data from a set.

Customer Information Control System
See *CICS*.

data type
A classification identifying one of various kinds of data. In SQL, the data type is an attribute of columns, literals, host variables, special registers, parameters, and the results of functions and expressions.

database
A collection of interrelated or independent data items that are stored together to serve one or more applications.

database administrator
A person who is responsible for the design, development, operation, security, maintenance, and use of a database.

database management system
See *DBMS*.

database manager
A program that manages data by providing centralized control, data independence, and complex physical structures for efficient access, integrity, recovery, concurrency control, privacy, and security.

database server
A software program that uses a database manager to provide database services to other software programs or computers.

DBCS (double-byte character set)
A set of characters in which each character is represented by two bytes. These character sets are commonly used by national languages such as Japanese and Chinese, which have more symbols than can be represented by a single byte. See also *SBCS*.

DBMS (database management system)
A software system that controls the creation, organization, and modification of a database and the access to the data that is stored within it.

DCT (destination control table)
A table describing each of the transient data destinations used in CICS. This table contains an entry for each extrapartition, intrapartition, and indirect destination.

default form
The QMF form created when a saved form is not specified on the RUN QUERY command.

default value
A predetermined value, attribute, or option that is assumed when no other value is specified. A default value can be defined for column data in Db2 tables by specifying the DEFAULT keyword in an SQL statement that changes data (such as INSERT, UPDATE, and MERGE).

dependent row
A row that contains a foreign key that matches the value of a parent key in the parent row. The foreign key value represents a reference from the dependent row to the parent row.

dependent table
A table that is a dependent of an object. For example, a table with a foreign key is a dependent of the table containing the corresponding primary key.

destination control table
See *DCT*.

detail block text
The text in the body of a report that is associated with a particular row of data.

detail heading text
The text in the heading of a report.

detail variation
A data formatting definition specified on a FORM.DETAIL panel that can be used to conditionally format a report or part of a report.

distinct type
A user-defined data type that shares a common representation with built-in data types.

distributed data
Data that is stored on more than one system and is available to remote users and application programs.

distributed database
A database that appears to users as a logical whole, locally accessible database, but consists of databases in multiple locations that are connected by a data communications network.

Distributed Relational Database Architecture™
See *DRDA*.

distributed unit of work
A form of distributed relational database processing that enables a user or application program to read or update data at multiple locations within a unit of work. Within one unit of work, an application, such as QMF, running in one system can direct SQL requests to multiple remote database management systems using the SQL supported by those systems. The request is made through a QMF command that includes a three-part table or view name. QMF commands with three-part names cannot be directed to Db2 for VM or VSE databases or used when QMF for TSOhas been started as a Db2 for z/OS stored procedure. Three-part names in QMF commands also cannot refer to a table that contains large object (LOB) data types.

double-byte character set
See *DBCS*.

double-precision floating-point number
A 64-bit approximate representation of a real number.

DRDA (Distributed Relational Database Architecture)
The architecture that defines formats and protocols for providing transparent access to remote data. DRDA defines two types of functions: the application requester function and the application server function.

environment
A named collection of logical and physical resources used to support the performance of a function.

exit routine
A program that receives control from another program to perform specific functions.

Extensible Markup Language
See *XML*.

extended syntax
Syntax that is used for the QMF SET GLOBAL and GET GLOBAL commands and certain function calls in a callable interface application. Extended syntax defines parameters used by QMF callable interface applications written in Assembler, C, COBOL, Fortran, or PL/I.

fallback
The process of returning to a prior release of a software program after attempting or completing migration to a current release.

fetch
The process of retrieving rows from the database or a file to create a QMF DATA object. QMF supports multirow fetch through the use of the DSQSMRFI parameter.

foreign key
In a relational database, a key in one table that references the primary key in another table.

GDDM (Graphical Data Display Manager)
Graphics software that defines and displays text and graphics for output on a display device or printer.

global variable
A named entity whose value persists for the duration of a QMF session by default. QMF uses global variables to manage both session and database activity. Some global variables can be set with the SET GLOBAL command, while others record information about the state of the current QMF session and therefore cannot be set.

graphic string
A sequence of double-byte character set (DBCS) characters.

Graphical Data Display Manager
See *GDDM*.

host
The controlling or highest-level system in a data communications configuration.

HTML (hypertext markup language)
A markup language that conforms to the Standard Generalized Markup Language (SGML) standard and was designed primarily to support the online display of textual and graphical information, including hypertext links.

hypertext markup language
See *HTML*.

ICU (Interactive Chart Utility)
A menu-driven component of IBM's Graphical Data Display Manager (GDDM) product that allows non-programmers to display, print, or plot charts, graphs, and diagrams.

identity column
A column that provides a way for the Db2 database manager to automatically generate a numeric value for each row that is inserted into a table. Identity columns are defined with the AS IDENTITY clause. A table can have no more than one identity column.

index
A set of pointers that is logically ordered by the values of a key. Indexes provide quick access to data and can enforce uniqueness of the key values for the rows in the table.

inner join
The result of a join operation that includes only the matched rows of both tables that are being joined. See also *outer join*.

installation verification procedure
See *IVP*.

Integrated Exchange Format
See *IXF*.

Interactive Chart Utility
See *ICU*.

Interactive System Productivity Facility
See *ISPF*.

ISPF (Interactive System Productivity Facility)
An IBM licensed program that serves as a full-screen editor and dialog manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogs between the application programmer and terminal user.

IVP (installation verification procedure)
A procedure or program whose purpose is to verify that a product has been correctly installed.

IXF (Integrated Exchange Format)
A protocol for transferring tabular data among various software products.

JCL (job control language)
A command language that identifies a job to an operating system and describes the job's requirements.

job control language
See *JCL*.

join
An SQL relational operation that allows retrieval of data from two or more tables based on matching column values.

key
A column or an ordered collection of columns that is identified in the description of a table, index, or referential constraint. The same column can be part of more than one key.

keyword
One of the predefined words of a programming language, artificial language, application, or command.

keyword parameter
A parameter that consists of a keyword followed by one or more values. See also *positional parameter*.

large object
See *LOB*.

link-edit
To create a loadable computer program by means of a linkage editor.

linkage editor
A computer program for creating load modules from one or more object modules or load modules by resolving cross-references among the modules and, if necessary, adjusting addresses.

literal
A character string whose value is defined by the characters themselves. For example, the numeric constant 7 has the value 7, and the character constant 'CHARACTERS' has the value CHARACTERS.

linear procedure
A sequenced set of QMF commands or command synonyms that can be used to perform several operations at once. See also *procedure with logic*.

linear syntax
QMF command syntax that is entered in one statement of a program or procedure, or that can be entered on the QMF command line.

load module
A program in a form suitable for loading into main storage for execution.

LOB (large object)
A sequence of bytes with a size ranging from 0 bytes to 2 gigabytes (less 1 byte). There are three LOB data types: binary large object (BLOB), character large object (CLOB, which can include single-byte characters only or a mixture of single-byte and double-byte characters), and double-byte character large object (DBCLOB). QMF supports a LOB column size of up to 32 KB.

local
Pertaining to databases, objects, or applications that are installed or stored in the system in which QMF is currently running.

location
A specific relational database server in a distributed relational database system. Each location has a unique location name.

location name
The unique name of a database server. An application uses the location name to access a Db2 database server.

lock
A means of serializing a sequence of events or serializing access to data.

log
A collection of records that sequentially describes the events that occur in a system.

LUW
An abbreviation for Linux, UNIX, and Windows.

National Language Feature
See *NLF*.

New Function Mode
See *NFM*.

NFM (New Function Mode)
An installation mode of QMF Version 8.1 and QMF Version 9.1 that allowed owner and object names in the QMF object catalog to be the maximum length allowed by the database. QMF Version 12.1 allows owner and object names to be as long as the database allows as well. See also *CM*.

NLF (National Language Feature)
Any of several optional features available with QMF. NLFs allow users to interact with QMF in specific native languages.

object
A named storage space that consists of a set of characteristics that describe the space and, in some cases, data. An object is anything that occupies space in storage, can be located in a library or directory, can be secured, and on which defined operations can be performed. See also *QMF object*.

outer join
The result of a join operation that includes the matched rows of both tables that are being joined and preserves some or all of the unmatched rows of the tables that are being joined. See also *inner join*.

package
A control-structure database object produced during program preparation that can contain both executable forms of static SQL statements or XQuery expressions and placement holders for executable forms of dynamic SQL statements.

panel
A formatted display of information on a screen that can also include entry fields.

parameter
A value or reference passed to a function, command, or program that serves as input or controls actions. The value is supplied by a user or by another program or process.

partition
A portion of a page set. Each partition corresponds to a single, independently extendable data set. Partitions can be extended to a maximum size of 1, 2, or 4 gigabytes, depending on the number of partitions in the partitioned page set. All partitions of a given page set have the same maximum size.

plan
See *application plan*.

positional parameter
A parameter that must appear in a specified location, relative to other parameters. See also *keyword parameter*.

precision
An attribute of a number that describes the total number of significant digits.

predicate
An element of a search condition that expresses or implies a comparison operation.

primary authorization ID
The authorization identifier used to identify an application process to Db2 for z/OS.

primary key
In a relational database, a key that uniquely identifies one row of a database table.

privilege
In SQL, a capability given to a user by the processing of a GRANT statement.

procedure
A sequenced set of statements or commands used to perform one or more tasks. See also *linear procedure* and *procedure with logic*.

procedure with logic
A set of statements that performs one or more tasks. A procedure with logic begins with a REXX comment and allows conditional logic (which uses REXX), calculations, build strings, and TSO or CICS commands. See also *linear procedure*.

profile
An object that contains information about the characteristics of the user's session.

program temporary fix
See *PTF*.

prompted query
A menu-driven query controlled by user-provided parameters.

PTF (program temporary fix)
For System i®,System p, and System z®, products, a fix that is tested by IBM and is made available to all customers.

QBE (Query-by-Example)
A component of QMF that allows users to create queries graphically.

QMF administrator authority
Users with this authority can perform the following commands on QMF queries, forms, and procedures that are owned by other users without forcing the owners to share these objects with all users: SAVE, ERASE, IMPORT, EXPORT, and DISPLAY. QMF administrator authority is determined either by the user's profile MODEL column value, or, if the user has INSERT or DELETE authority for the Q.PROFILES table itself.

QMF administrator
A user who has QMF administrator authority.

Query-by-Example
See *QBE*.

QMF object
An object used by QMF users to query, format, and present data or otherwise manage interaction between QMF and the database. QMF objects include queries and query result data, forms, procedures, reports, charts, and the QMF profile. Each QMF object has a named temporary storage area that is used to display the object. All objects except reports and charts can be saved in the database; reports and charts are created dynamically upon user request by applying the formatting specifications of a particular QMF form to result data that has been returned from the database. See also *temporary storage*.

QMF object catalog
A set of control tables that stores information about QMF queries, procedures, forms, folders, and analytics objects. These control tables include Q.OBJECT_DIRECTORY, Q.OBJECT_DATA, and Q.OBJECT_REMARKS.

qualifier
When referring to a QMF object, the part of the name that identifies the owner or the location of an object. When referring to a TSO data set, any part of the name that is separated from the rest of the name by periods. For example, 'TCK', 'XYZ', and 'QUERY' are all qualifiers in the data set name 'TCK.XYZ.QUERY'.

query
A request for information from a database based on specific conditions: for example, a request for a list of all customers in a customer table whose balances are greater than $1000. In QMF, a query also refers to SQL statements submitted from the Prompted Query, QBE, or SQL query panel, even if these statements are not requests for information (SELECT statements).

RCT (resource control table)
A Db2 control table that defines the relationship between CICS transactions and Db2 resources.

RDBMS (relational database management system)
A collection of hardware and software that organizes and provides access to a relational database.

RDO (resource definition online)
In CICS, a facility that allows the user to define certain CICS resources interactively while CICS is running. Specifically, RDO allows the user to define terminals, programs, and transactions interactively.

record
The storage representation of a row or other data.

record length
The length of storage that represents a row or other data.

reentrant
Executable code that can reside in storage as one shared copy for all database threads. Reentrant code is not self-modifying and provides separate storage areas for each thread.

referential constraint
The requirement that the nonnull values of a designated foreign key are valid only if they also appear as values of the primary key of the parent table. The referential constraint is always defined from the perspective of the dependent file.

relational database
A database that can be perceived as a set of tables and manipulated in accordance with the relational model of data. Each database includes a set of system catalog tables that describe the logical and physical structure of the data, a configuration file containing the parameter values allocated for the database, and a recovery log with ongoing transactions and archivable transactions.

relational database management system
See *RDBMS*.

remote
Pertaining to databases, objects, or applications that are installed or stored on a system other than the system where QMF is currently executing. You can access objects (including QMF queries, forms, procedures, folders, and analytics objects) at a remote server by using the QMF CONNECT command. You can also use a QMF command with a three-part table or view name if you want to access just tables or views at a remote location. Remote access is not permitted when QMF for TSO is running as a Db2 for z/OS stored procedure.

remote unit of work
A form of distributed relational database processing in which an application program, such as QMF, can access data on a remote database within a unit of work. The connection is established by the QMF CONNECT command. The CONNECT command cannot be used when QMF for TSO is running as a Db2 for z/OS stored procedure.

requester
See *application requester.*

resource
The object of a lock or claim, which could be a table space, an index space, a data partition, an index partition, or a logical partition.

resource control table
See *RCT*.

resource definition online
See *RDO*.

Restructured Extended Executor
See *REXX*.

REXX (Restructured Extended Executor)
A general-purpose, high-level programming language, particularly suitable for EXEC procedures or programs for personal computing.

roll back
To restore data that is changed by an SQL statement to the state at its last commit point. If a failure occurs in a query that contains multiple statements and no COMMIT statements, all statements, except those that affect the QMF session (such as SET), are rolled back. If a failure occurs in a query that contains one or more COMMIT statements, all updates after the last successful COMMIT statement are rolled back. In either case, the query ends after the failure.

routine
A program or sequence of instructions called by a program. Typically, a routine has a general purpose and is frequently used.

row
The horizontal component of a table, consisting of a sequence of values, one for each column of the table.

runtime variable
A variable in a procedure or query whose value is specified by the user when the procedure or query is run. The value of a runtime variable is only available in the current procedure or query. See also *global variable*.

SBCS (single-byte character set)
A coded character set in which each character is represented by a 1-byte code. A 1-byte code point allows representation of up to 256 characters. See also *double-byte character set*.

scalar function
An SQL function that optionally accepts arguments and that returns a single scalar value each time that it is invoked. A scalar function can be referenced in an SQL statement wherever an expression is valid.

scratchpad area
A work area used in conversational processing to retain information from an application program across executions of the program.

search condition
A criterion for selecting rows from a table. A search condition consists of one or more predicates.

secondary authorization ID
In Db2 for z/OS, an authorization identifier that is associated with a primary authorization ID by an authorization exit routine. See also *primary authorization ID*.

segmented table space
A table space that is divided into equal-sized groups of pages called segments. Segments are assigned to tables so that rows of different tables are never stored in the same segment. See also *table space*.

server
See *application server*.

session
All interactions between the user and QMF from the time the user invokes QMF until the EXIT command is issued.

shift-in character
A control character (X'0F') that is used in EBCDIC systems to denote that the subsequent bytes represent SBCS characters. See also *shift-out character.*

shift-out character
A control character (X'0E') that is used in EBCDIC systems to denote that the subsequent bytes, up to the next shift-in control character, represent DBCS characters. See also *shift-in character.*

single-byte character set
See *SBCS*.

single-precision floating-point number
A 32-bit approximate representation of a real number.

SQL (Structured Query Language)
A standardized language for defining and manipulating data in a relational database.

SQL authorization ID
See *SQLID*.

SQL connection
An association between an application process and a local or remote application server or database server. See also *remote unit of work*, *distributed unit of work*.

SQL function
A function that is implemented entirely by using a subset of SQL statements and SQL PL statements.

SQL ID (SQL authorization ID)
In Db2 for z/OS, the ID that is used for checking the authorization of dynamic SQL statements in some situations.

SQL return code
The SQLSTATE or SQLCODE that indicates whether the previously run SQL statement completed successfully, with one or more warnings, or with an error.

SQLCA (Structured Query Language Communication Area)
A set of variables that provides an application program with information about the execution of its SQL statements or requests from the database manager. When an error is associated with an SQL code, the QMF message help (available by pressing the Help key) displays the contents of the SQLCA.

stored procedure
A routine that can be invoked using the SQL CALL statement to perform operations that can include both host language statements and SQL statements.

stored procedure interface
An interface to QMF for TSO that allows you to start QMF as a Db2 for z/OS stored procedure, pass the name of a QMF query or procedure that performs the work you require, and receive up to 21 result sets back, including a result set for trace output. QMF for TSO can be started in this manner from any product that can run a Db2 for z/OS stored procedure.

Structured Query Language
See *SQL*.

Structured Query Language Communication Area
See *SQLCA*.

subquery
A complete SQL query that appears in a WHERE or HAVING clause of another query.

substitution variable
(1) A variable in a procedure or query whose value is specified either by a global variable or by a runtime variable. (2) A variable in a QMF form whose value is specified by a global variable.

substring
A part of a character string.

subsystem
In Db2 for z/OS, a distinct instance of a relational database management system (RDBMS).

table
In a relational database, a database object that consists of a specific number of columns and is used to store an unordered set of rows. See also *base table*.

table space
A logical unit of storage in a database. In Db2 for z/OS, a table space is a page set and can contain one or more tables. In Db2 for Linux, UNIX, and Windows, a table space is a collection of containers, and the data, index, long field, and LOB portions of a table can be stored in the same table space or in separate table spaces.

temporary storage
An area used to store a QMF object temporarily while the user is working on it so that, with each use, it can be readily accessed without further database retrieval. There are seven temporary storage areas: QUERY, DATA, FORM, PROC, REPORT, CHART, or PROFILE. With the exception of query result data (the DATA object), the QMF objects in these areas can be displayed using the SHOW command followed by the name of the storage area. Though the contents of the DATA area cannot be directly displayed, users can issue the SHOW REPORT or SHOW CHART commands to see the query result data formatted with the specifications of the form currently in the FORM area. See also *QMF object*, *current object*.

temporary storage queue
In CICS, a queue of data items which can be read and reread, in any sequence. The queue is created by a task, and persists until the same task or a another task deletes it. See also *transient data queue*.

thread
The Db2 structure that describes an application's connection, traces its progress, processes resource functions, and delimits its accessibility to Db2 resources and services. Most Db2 functions execute under a thread structure.

three-part name
The full name of a table, view, or alias that consists of a location name, an authorization identifier, and an object name, separated by periods. QMF commands that include three-part names can be initiated only from Db2 for z/OS databases and can be directed to all databases except DB2 for VM or VSE. When QMF for TSO has been started as a Db2 for z/OS stored procedure, QMF commands with three-part names are not supported.

Time Sharing Option
See *TSO*.

trace
A record of the processing of a computer program or transaction. The information collected from a trace can be used to assess problems and performance.

transaction
A unit of processing consisting of one or more application programs, affecting one or more objects, that is initiated by a single request.

transient data queue
A CICS storage area where objects are stored for subsequent internal or external processing. See also *temporary storage queue*.

trigger
A database object that is associated with a single base table or view and that defines a rule. The rule consists of a set of SQL statements that runs when an insert, update, or delete database operation occurs on the associated base table or view.

TSO (Time Sharing Option)
A base element of the z/OS operating system that allows users to work interactively with the system.

two-phase commit
A two-step process by which recoverable resources in an external subsystem are committed. During the first step, the database manager subsystems are polled to ensure that they are ready to commit. If all subsystems respond positively, the database manager instructs them to commit.

UDF (user-defined function)
A function that is defined to the Db2 database system by using the CREATE FUNCTION statement and that can be referenced thereafter in SQL statements. A UDF can be an external function or an SQL function.

Unicode
A character encoding standard that supports the interchange, processing, and display of text that is written in the common languages around the world, plus some classical and historical texts. The Unicode standard has a 16-bit character set defined by ISO 10646.

unit of recovery (UR)
A sequence of operations within a unit of work between points of consistency.

unit of work (UOW)
A recoverable sequence of operations within an application process. At any time, an application process is a single UOW, but the life of an application process can involve many UOWs as a result of commit or rollback operations. In a multisite update operation, a single UOW can include several units of recovery. In QMF SQL queries that include multiple statements and no COMMIT statements, all statements comprise a single unit of work, so all statements except those that affect the session (such as SET) are rolled back in the event of a failure. In QMF SQL queries that include multiple statements and one or more COMMIT statements, a unit of work consists of a COMMIT statement and all previous statements back to the beginning of the query or the last COMMIT statement. If a failure occurs, all updates after the last successful COMMIT statement are rolled back.

user-defined function
See *UDF*.

view
A logical table that is based on data stored in an underlying set of tables. The data returned by a view is determined by a SELECT statement that is run on the underlying tables.

XML (Extensible Markup Language)
A standard metalanguage for defining markup languages that is based on Standard Generalized Markup Language (SGML).

z/OS
An IBM mainframe operating system that uses 64-bit real storage.

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Product Number: 5650-DB2
5615-DB2
5697-QM2

SC27-8880