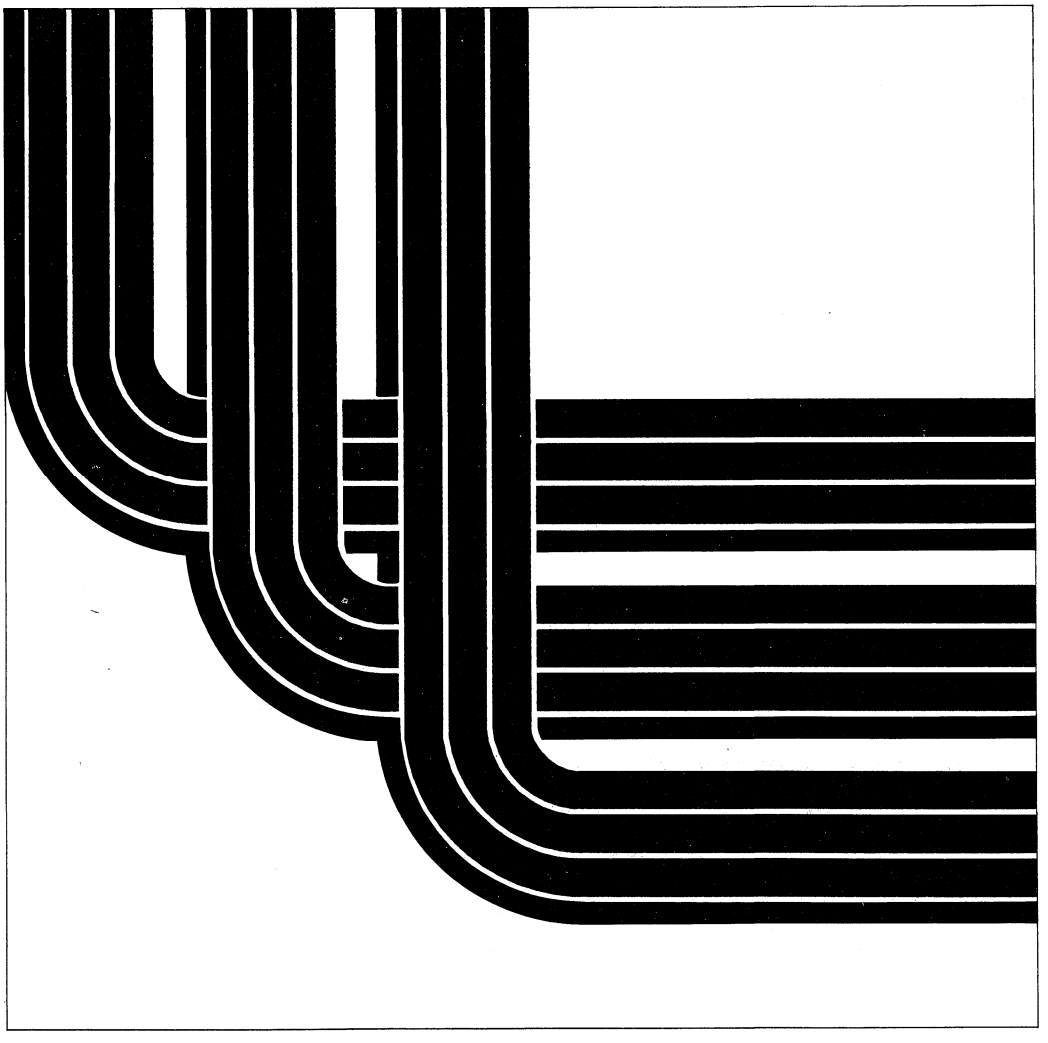
Application System/400™

SC41-8193-00

Programming: Query Management/400 Reference

Version 2



Application Development

Take Note!

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

First Edition (May 1991)

This edition applies to the licensed program IBM Operating System/400 (Program 5738-SS1), Version 2 Release 1 Modification 0, and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure that you are using the proper edition for the level of the product.

Order publications through your IBM representative or the IBM branch serving your locality. Publications are not stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, you may address your comments to:

Attn Department 245
IBM Corporation
3605 Highway 52 N
Rochester, MN 55901-7899

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you or restricting your use of it.

Contents

Notices ix
Programming Interface x

About This Manual xi
Who Should Use This Manual xi
What the SAA Solution Is xi
Supported Environments xii
Common Programming Interface xii
Query Management/400 and the SAA Solution xiii
How to Read the Syntax Diagrams xiii

Chapter 1. Commands 1-1
What Query Management/400 Can Do 1-1
Naming Conventions 1-2
Query Objects 1-2
System Naming 1-2
SAA Naming 1-3
Specifying Commands and Keywords 1-5
Security and Authorization 1-5
Command Reference 1-6
Command Parsing 1-7
ERASE 1-9
Examples of the ERASE command 1-9
EXIT 1-10
Examples of the EXIT command 1-10
EXPORT 1-11
Examples of the EXPORT Command 1-12
GET 1-13
Examples of the GET Command 1-14
IMPORT 1-15
Examples of the IMPORT Command 1-17
PRINT 1-18
Examples of the PRINT Command 1-20
RUN 1-21
Examples of the RUN Command 1-22
SAVE DATA AS 1-23
Examples of the SAVE DATA AS Command 1-24
SET 1-25
Examples of the SET Command 1-26
START 1-27
Examples of the START Command 1-29
CL Commands 1-30
ANZQRY (Analyze Query) Command 1-30
CRTQMFORM (Create Query Management Form) Command 1-30
CRTQMQRY (Create Query Management Query) Command 1-30
DLTQMFORM (Delete Query Management Form) Command 1-30
DLTQMQRY (Delete Query Management Query) Command 1-30
RTVQMFORM (Retrieve Query Management Form) Command 1-30
RTVQMQRY (Retrieve Query Management Query) Command 1-31
STRQMPRC (Start Query Management Procedure) Command 1-31
STRQMQRY (Start Query Management Query) Command 1-31

WRKQMFORM (Work with Query Management Form) Command 1-31
WRKQMQRY (Work with Query Management Query) Command 1-31

Chapter 2. Procedures 2-1
Rules for Creating Procedures 2-1
Example 2-1
Procedure Interaction 2-2
The Procedure Objects 2-2
Delimited Areas in the Procedure 2-2
Example 2-3
Error Handling 2-3

Chapter 3. Callable Interface 3-1
Callable Interface Description 3-1
Interface Communications Area (DSQCOMM) 3-3
Return Codes 3-4
Return Variables 3-4
Command Message Variables 3-4
Query Message Variables 3-4
Query Management/400 Command Syntax Extension 3-5
C Language Interface 3-5
C Variable Support 3-5
Interface Communications Area (DSQCOMM) 3-7
Return Codes 3-7
Sample C Language Query CI Program 3-8
COBOL Language Interface 3-11
Interface Communications Area (DSQCOMM) 3-12
Return Codes 3-13
Sample COBOL Query CI Program 3-14
RPG Language Interface 3-16
Interface Communications Area (DSQCOMMR) 3-17
Return Codes 3-18
Sample RPG Language Query CI Program 3-18
Extended Variable Support 3-21
Creating Variables 3-21
Referencing Variables 3-21
Variable Names 3-22
Variable Values 3-22
Query Management/400 Defined Variables 3-22

Chapter 4. Query Capability 4-1
Rules for Creating Queries 4-1

Chapter 5. Report FORM Definition 5-1
How Applications Can Use the FORM 5-1
Formatting Terminology 5-1
COLUMN Fields 5-2
Column Heading 5-3
Usage 5-4
Indent 5-6
Width 5-6
Datatype 5-6
Edit 5-7
Seq 5-8
Run-Time Defaults 5-8

Datatype 5-8
Column Heading 5-9
Edit 5-9
Width 5-10
PAGE Fields 5-10
Blank Lines Before Heading/Footing 5-11
Blank Lines After Heading/Footing 5-11
Heading Text Lines 5-11
Line 5-11
Align 5-12
Page Heading Text 5-12
Footing Text Lines 5-13
Line 5-13
Align 5-13
Page Footing Text 5-13
FINAL TEXT Fields 5-13
New Page for Final Text 5-14
Put Final Summary at Line 5-14
Blank Lines before Text 5-14
Line 5-14
Align 5-15
Final Text Lines 5-15
BREAK Fields 5-15
New Page for Break/New Page for Footing 5-16
Repeat Column Heading 5-16
Blank Lines before Heading/Footing 5-16
Blank Lines after Heading/Footing 5-16
Put Break Summary at Line 5-16
Break Heading Text Lines 5-17
Line 5-17
Align 5-17
Break Heading Text 5-17
Break Footing Text Lines 5-17
Line 5-17
Align 5-18
Break Footing Text 5-18
OPTIONS Fields 5-18
Detail Line Spacing 5-18
Outlining for Break Columns 5-18
Default Break Text 5-19
Column Wrapped Lines Kept on a Page 5-19
Column Heading Separators 5-19
Break Summary Separators 5-19
Final Summary Separators 5-19

Chapter 6. Exported Objects 6-1
General Object Formats 6-1
Comments in Externalized Query Management/400 Objects 6-1
External Formats 6-2
Panel Format 6-2
Encoded Format 6-2
Size of the Encoded Format 6-2
Records that Make Up the Base Encoded Format 6-2
Header ("H") Record 6-3
Value ("V") Records 6-7

Table Description ("T") Records 6-9
Table Row ("R") Records 6-11
End-of-Object ("E") Record 6-13
Application Data ("*") Record 6-14
Record Format Rules 6-15
Specific Query Object Formats 6-16
Externalized FORM Format 6-16
Externalized PROC and QUERY Formats 6-28

Chapter 7. Using Query/400 Definition Information 7-1
Query Definition Conversion 7-1
Applying Query Management/400 to QRYDFN Objects 7-1
QRYDFN Conversion Considerations 7-2
Analyzing a QRYDFN 7-3
Inspecting the Output 7-4
Applying QRYDFN Option Guidelines 7-4
Conversion Details 7-6
Creating Query Management/400 Objects from QRYDFN Objects 7-12
Adding SAA Function 7-13

Appendix A. DBCS Data A-1
What Is Double-Byte Data? A-1
How Does Double-Byte Data Look When Displayed? A-1
What Data Types Can Be Used with DBCS Data? A-1
Using DBCS Data in Query Management/400 . . . . . A-2
Using DBCS Data in Input Fields A-2
Using DBCS Data in Queries A-2
Using DBCS in the FORM A-2
How Data Truncation is Handled A-3
Exporting DBCS Data A-3
Importing DBCS Data A-3
Printing DBCS Reports A-3

Glossary G-1

Bibliography H-1
For the AS/400 System H-1
For the SAA Solution H-1
For Implementation on the System/370 Computer H-1
For Implementation with the OS/2 Licensed Program H-1

Index X-1

Figures

3-1. Callable Interface Diagram 3-2
3-2. Sample C Program 3-9
3-3. Sample COBOL Program 3-14
3-4. Sample RPG Program 3-19
5-1. Basic Parts of a Report 5-1
5-2. Basic Parts of a Report with One Level of Control Break 5-2
5-3. Default Values for Column Fields 5-3
5-4. Use of Edit Codes 5-8
5-5. Values for Columns Datatype Field 5-9
5-6. Default Values for Page Fields 5-11
5-7. Default Values for Final Text Fields 5-14
5-8. Default Values for Break Fields 5-15
5-9. Default Values for Options Fields 5-18
6-1. Header Record Description 6-4
6-2. Header Record Fields 6-6
6-3. Value Record Description 6-7
6-4. Value Record Fields 6-8
6-5. Table Record Description 6-9
6-6. Table Record Fields 6-10
6-7. Row Record Description 6-11
6-8. Row Record Fields 6-12
6-9. End-of-Object Record Description 6-13
6-10. End-of-Object Record Fields 6-13
6-11. Application Data Record Description 6-14
6-12. Application Data Record Fields 6-14
6-13. Encoded (Externalized) FORM Field Summary 6-17
6-14. Sample Externalized Form 6-19
6-15. Descriptive Names of Encoded Format Form Fields 6-23
6-16. Preferred Format for Encoded Break Information 6-25
6-17. Original Format for Encoded Break Information. 6-26
7-1. Correlation between the Work with Query Display and Query Management/400 Objects 7-7
7-2. Conversion Data Flow 7-12
7-3. Sample CL Command Sequence for QRYDFN Conversion 7-13

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

AD/Cycle	Application System/400
AS/400	C/400
COBOL/400	IBM
Operating System/2	Operating System/400
OS/2	OS/400
PS/2	QMF
SAA	Systems Application Architecture
SQL/400	System/370
400	

This publication could contain technical inaccuracies or typographical errors.

This manual may refer to products that are announced but are not yet available.

Information that has changed since Version 1 Release 3 Modification 0 is indicated by a vertical bar (|) to the left of the change.

This publication 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.

This manual contains small programs that are furnished by IBM as simple examples to provide an illustration. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. All programs contained herein are provided to you "AS IS". THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED.

Programming Interface

This *Programming: Query Management/400 Reference*, SC41-8193, is intended to provide the customer with reference information about Query Management/400. It primarily contains general-use programming interfaces, which allow the customer to write programs that use the services of Query Management/400.

About This Manual

This manual describes the functions provided by Query Management/400 and provides examples for using them.

This manual was produced in conjunction with *Query Management/400 Programmer's Guide*.

You may need to refer to other IBM manuals for more specific information about a particular topic. The *Publications Guide*, GC41-9678, provides information on all the manuals in the AS/400 library.

For a list of related publications, see the "Bibliography."

Who Should Use This Manual

This manual is intended to be used by:

- An applications programmer familiar with the query interface, the Query/400 licensed program, and with Query/400 definitions.
- A system operator who has had formal AS/400 system training, is familiar with operating the AS/400 system, and is familiar with query functions.
- A user who is performing problem analysis.
- IBM programming service personnel who are responsible for resolving non-customer problems with system programs and this product.

Note: Before you use this manual, you should be knowledgeable in the following:

- The Systems Application Architecture (SAA) solution—specifically the common programming interface for query applications
- Query/400 licensed program
- AS/400 system operation and functions.

What the SAA Solution Is

The SAA solution is based on a set of software interfaces, conventions and protocols that provide a framework for designing and developing applications.

The SAA solution:

- Defines a common programming interface that you can use to develop applications that can be integrated with each other, and transported to run in multiple SAA environments
- Defines common communications support that you can use to connect applications, systems, networks, and devices
- Defines a common user access that you can use to achieve consistency in panel layout and user interaction techniques
- Offers some applications and application development tools written by IBM*.

Supported Environments

Several combinations of IBM hardware and software have been selected as SAA environments. These are environments in which IBM will manage the availability of support for applicable SAA components and the conformance of those components to SAA specifications. The SAA environments are the following:

- MVS
  - TSO/E
  - CICS
  - IMS
- VM/CMS
- Operating System/400* (OS/400*)
- Operating System/2* (OS/2*).

Common Programming Interface

As its name implies, the common programming interface (CPI) provides languages, commands, and calls that programmers can use to develop consistent applications. These applications can be easily integrated and transported across the supported environments.

The components of the interface currently fall into two general categories:

- Languages

  Application Generator
  C
  COBOL
  FORTRAN
  PL/I
  Procedures Language
  RPG

- Services

  Communications Interface
  Database Interface
  Dialog Interface
  Presentation Interface
  Query Interface
  Repository Interface.

The CPI is not in itself a product or a piece of code. But—as a definition—it does establish and control how IBM products are being implemented, and it establishes a common base across the applicable SAA environments.

Thus, when you want to create an application that can be used in more than one environment, you can stay within the boundaries of the CPI and obtain easier portability. (Naturally, the design of such applications should be done with portability in mind as well.)

The last part of this manual contains a glossary and an index. Use the glossary to find the meaning of an unfamiliar term. Use the index to find the page or pages on which a particular topic is discussed.

Query Management/400 and the SAA Solution

The SAA query interface defines the elements that are consistent across the applicable SAA environments. Preparing and running programs requires the use of an SAA query product that implements the interface on one of these systems.

For the query interface, these products and functions are:

- Query Management Facility on the System/370* computer
- Query Management/400
- OS/2 Query Manager.

Because Query Management/400 is the implementing product for the OS/400 environment, you can use it to run SAA-conforming query applications.

How to Read the Syntax Diagrams

Throughout this manual, syntax is described using the structure defined below.

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

  The ▶▶— symbol indicates the beginning of a statement.

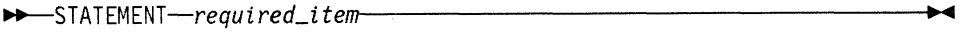
  The —▶ symbol indicates that the statement syntax is continued on the next line.

  The ▶— symbol indicates that a statement is continued from the previous line.

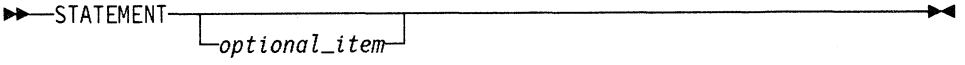
  The —▶◀ symbol indicates the end of a statement.

  Diagrams of syntactical units other than complete statements start with the ▶— symbol and end with the —▶ symbol.

- Required items appear on the horizontal line (the main path).

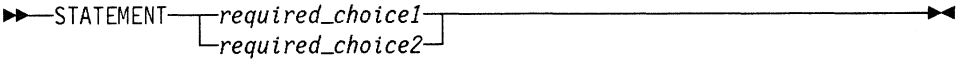


- Optional items appear below the main path.

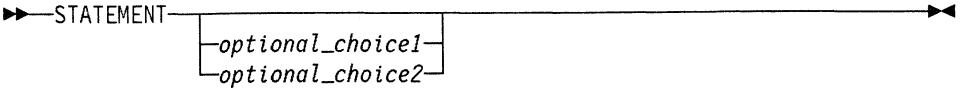


- If you can choose from two or more items, they appear vertically, in a stack.

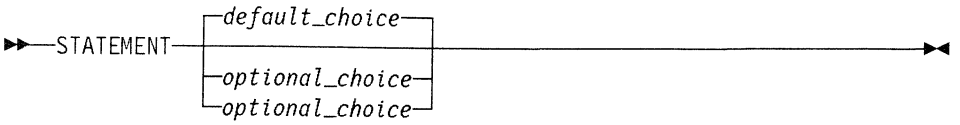
  If you *must* choose one of the items, one item of the stack appears on the main path.



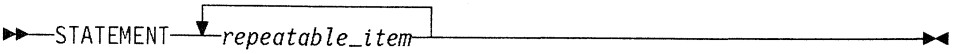
  If choosing one of the items is optional, the entire stack appears below the main path.



  If one of the optional items is the default, it will appear above the main path and the remaining choices will be shown below.



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



  A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Keywords appear in uppercase (for example, PARM1). They must be spelled exactly as shown. Variables appear in all lowercase letters (for example, *parmx*). They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or such symbols are shown, you must enter them as part of the syntax.

Chapter 1. Commands

This chapter describes the following:

- What Query Management/400 Can Do
- Naming Conventions
- Specifying Commands and Keywords
- Command Reference
- CL Commands.

What Query Management/400 Can Do

Query Management/400 allows you to access information in a relational database and to control how this data appears when formatted into a report. Query Management/400 provides services that fall into two major categories: querying and report writing.

Application programs can use Query Management/400 services through a program-to-program call interface or through Query Management/400 objects (Query Management/400 objects are files that contain queries, procedures, and forms). Applications may build and manipulate these objects and then use them in conjunction with Query Management/400 functions to produce the desired data and reports.

Applications that use Query Management/400 have the following advantages:

- Reduced requirements for error handling and interpretation. The application may not have to check for SQL error codes.
- Use of queries, procedures, and forms that are defined and stored outside of the application code. You can update your application by changing Query Management/400 objects without having to change or recompile your application program.
- No need to understand and handle relational database manager protocols. The application can pass a few simple commands to allow access to data in an exported format.

The Query Management/400 commands can be issued by processing a statement in a procedure or by running a program that uses the callable interface.

Naming Conventions

The following rules apply when you create a table or view or name an object that you want to save in the database.

Query Objects

Query objects may be specified on commands using either SAA* names or system names. The naming convention to use is specified in the query command procedure on the START command or the Start Query Management Query (STRQMQRY) CL command. Query Management/400 uses SAA (*SAA) naming as the default. The naming convention specified for a query instance is used throughout the entire instance and applies only to that instance. It cannot be changed after the START command has been issued for the query instance.

Note: These naming conventions only apply to the query object specified on a command. System naming conventions always apply on the file name specified on the IMPORT and EXPORT commands.

System Naming

When a query instance uses system names, the following rules apply for a query object name specified in a query command:

- Query objects may be specified using delimited names. A delimited is a character string with the following characteristics:
  - One to eight characters long
  - Beginning and ending with quotation marks ("MYFORM")
  - Contains any character except:
    - A blank
    - An asterisk (*)
    - A question mark (?)
    - An apostrophe (')
    - Quotation marks (")
    - The numbers hex 00 through 3F, or hex FF.

  A delimited name may be qualified, but the qualifier and the name must be surrounded by quotation marks separately from each other ("MYLIB"/"MYFORM").
- Query objects may be specified using simple names. Simple names are character strings up to 10 characters long and must begin with an alphabetic character (A through Z, $, #, or @). Periods and blanks are not allowed in simple names.
- A query command file name can be qualified by a library name up to 10 characters long. A slash (/) must separate the qualifying library name and the file name. For example, MYLIB/FILE1 (file FILE1 in library MYLIB) is a qualified name. The rules applying to AS/400* names in quotation marks and simple names apply to the library name used as the qualifier.
- Objects of the same type that are stored in the same library must have different names (you cannot have two files named TEST, for example). Queries and forms are different AS/400 object types; therefore, a query and form may have the same name. Names for procedures, tables, and views must be different because they are all AS/400 files. A procedure, table, or view can have the same name as a query object or form object, but not another procedure, table, or view.

- Names for queries, forms and procedures can use reserved words (like FORM, QUERY, COUNT, NULL and so on), though naming something with an SQL keyword is not recommended.
- If a query, form, or procedure specified in a query command is not qualified, AS/400 search conventions are followed. If an unqualified query object name is specified, Query Management/400 searches the library list (*LIBL) for the query object. If the query object is being created, Query Management/400 places the object in the current library (*CURLIB).
- If an unqualified table or view name is specified on a query command, see *SQL/400* Reference* for the search conventions followed.

For more information on system naming and AS/400 search conventions see *CL Reference*.

SAA Naming

When the query instance is using SAA names, the following rules apply for a query object name specified in a query command. These rules are similar to the AS/400 object naming conventions. In most cases, they are an extension of the object naming conventions stated in *SAA CPI Query Reference*. The deviations from *SAA CPI Query Reference* are noted.

- Query objects may be specified using delimited names. A delimited name is a character string with the following characteristics:
  - One to eight characters long
  - Beginning and ending with quotation marks ("MYFORM")
  - Contains any character except
    - A blank
    - An asterisk (*)
    - A question mark (?)
    - An apostrophe (')
    - Quotation marks (")
    - The numbers hex 00 through 3F, or hex FF.

  Note: An SAA name can be 12 characters including the quotation marks. AS/400 only allows 10 characters including the quotation marks.

  A delimited name may be qualified, but the qualifier and the name must be surrounded by quotation marks separately from each other ("MYLIB"/"MYFORM").
- Query objects may be specified using simple names. Simple names are character strings up to 10 characters long and must begin with an alphabetic character (A through Z, $, #, or @). Periods and blanks are not allowed in simple names.
- A name can be qualified by another name (usually a user or an authorization identification) of up to 10 single byte characters with a period (.) separating the qualifier and the name. For example, Q.QUERY1 (the query in the Q collection) is a qualified name. Query Management/400 uses the AS/400 SQL/400* conventions of treating the authorization ID as a user profile. If SAA naming conventions apply, Query Management/400 attempts to find the object in the library with the same name as the authorization ID. The rules applying to delimited and simple names apply to the library name used as the qualifier.

- Objects of the same type that are stored in the same library must have different names (you cannot have two files named TEST, for example). Queries and forms are different AS/400 object types; therefore, a query and form may have the same name. Names for procedures, tables, and views must be different because they are all AS/400 files. A procedure, table, or view can have the same name as a query object or form object, but not another procedure, table, or view.
- If an unqualified query object name is specified, Query Management/400 searches the library with the same name as the current user profile for the query object. If the query object is being created, Query Management/400 places the object in the library with same name as the current user profile. These are the same conventions followed by the SQL/400 licensed program.

AS/400 Objects: The *filename* specified on the IMPORT and EXPORT query commands will follow the AS/400 naming conventions for a source physical file.

- Query objects may be specified using delimited names. A delimited name is a character string with the following characteristics:
  - One to eight characters long
  - Beginning and ending with quotation marks ("MYLIB")
  - Contains any character except
    - A blank
    - An asterisk (*)
    - A question mark (?)
    - An apostrophe (')
    - Quotation marks (")
    - The numbers hex 00 through 3F, or hex FF.

  A delimited name may be qualified, but the qualifier and the name must be surrounded by quotation marks separately from each other ("MYLIB"/"MYFORM").
- Query objects may be specified using simple names. Simple names are character strings up to 10 characters long and must begin with an alphabetic character (A through Z, $, #, or @). Periods and blanks are not allowed in simple names.
- AS/400 rules also apply when the *filename* is specified with a qualified name. A *filename* in a query command can be qualified by a library name of up to 10 characters with a slash (/) separating the qualifier and the name. For example, MYLIB/FILE1 (a file in library MYLIB) is a qualified name.
- If a *filename* specified in a query command is not qualified, Query Management/400 searches the library list (*LIBL) for a source file named *filename.* If the file is being created, Query Management/400 will place the file in the current library (*CURLIB).
- If the physical file is a multiple member source file, the following rules apply for specifying the member:
  - If no member name is specified, the member name used will default to *FIRST on the IMPORT and EXPORT commands.
  - Query Management/400 will process a specified member in a physical file if a member name is given as part of the file name. The member name must follow the file name and be delimited by parenthesis with no

intervening blanks. For example, the member MEMBER1 in the file FILE1 can be specified by entering a file name as follows:

```
FILE1(MEMBER1)
```

Variable Names: See "Extended Variable Support" on page 3-21 for the rules that apply when you use variables in SQL queries across the callable interface. AS/400 specific rules are:

- Variable names that are used in SQL must start with a single-byte character letter queries must be preceded by an ampersand (&). The ampersand delimits the beginning of a variable name and is not included as one of the 18 characters allowed for the name. You cannot have more than one ampersand in a variable name, since each ampersand delimits the beginning of a distinct variable name.
- User-defined variables may not start with DSQ. An error is generated if an attempt is made to set a variable that starts with DSQ.
- Variable names within Query Management/400 are case sensitive. Therefore, the variable i_owe_you, is not the same as the variable I_OWE_YOU.

The following are valid variable names:

In an SQL Query	In the GET/SET Command
&I_owe_you	I_owe_you
&MYVAR123	MYVAR123
&THIS_IS_A_BIG_NAME	THIS_IS_A_BIG_NAME

Other Query Names: The naming convention being used by the query instance also applies to the SQL statements in any SQL query run during the instance. If system names are being used in the query instance, system names apply to the SQL query. If SAA naming conventions apply, then SQL naming conventions apply to the SQL query. See *SQL/400* Reference* for a description of the SQL and system naming conventions as followed by the SQL/400 product.

Specifying Commands and Keywords

For consistency when moving applications across systems, you should specify all commands, whether used in procedures or passed through the callable interface, in uppercase letters. Similarly, when manipulating a query object, you should specify all character keywords in uppercase letters. Text lines (for example, page headings) may be specified in either uppercase or lowercase letters.

Security and Authorization

Query Management/400 uses the AS/400 security and authorization model instead of the security and authorization model described in *SAA CPI Query Reference*. See *Security Concepts and Planning* for information about security concepts for the AS/400 system.

This section will discuss general security authorization considerations for the following items:

- Query objects
- AS/400 objects
- SQL

- Query Management/400.

Query objects: When query objects are created through a query command, there are various ways to specify the type of public authority for the query object that you want to give to other users.

- You can specify a default public authority for all objects created during a query instance by setting a value in the DSQOAUTH keyword in the query command procedure or on the START command. The values you can specify are:

  ***LIBCRTAUT**
  : *LIBCRTAUT authority for the object is the same as the value specified on the CRTAUT parameter of the library in which the object is being created. If the CRTAUT parameter is changed, the new value will not affect the authority of existing objects.

  ***CHANGE** *CHANGE authority allows other users to perform all operations on the object except those limited to the owner or controlled by object existence authority and object management authority. A user can change or use the query object in any way, except for deleting it or transferring it to a new owner.

  ***ALL** *ALL authority allows other users to perform all operations on the object except those limited to the owner or controlled by authorization list management rights. A user can do anything with the query object (including erasing it), except for transferring it to a new owner.

  ***EXCLUDE**
  : *EXCLUDE authority prevents other users from doing anything with the query object. Unless given specific types of authority, no user except its owner can use the query object.

  ***USE** *USE authority allows other users to run, export, or print the query object, but prevents them from importing it or saving to it.

  authorization list name
  : If you specify the name of an authorization list, its authority is used to control the users ability to use a query object. For more information, see *Security Guide*.

- If you do not specify an authority through the query command procedure, other users will have *EXCLUDE access to the query object.

AS/400 objects: Query Management/400 uses the same public authority for creating a nonquery object, such as the source physical file on an EXPORT, as it does when creating query objects.

SQL: See *SQL/400 Reference* for information for object authority as it applies to the SQL statement within an SQL query.

Command Reference

This section is a reference for the commands available to Query Management/400. Each command is described in full and has a syntax diagram provided as a means of quickly referring to the syntax of a command. For an explanation of the diagrams, see "How to Read the Syntax Diagrams" on page xiii.

Within the command syntax, one or more blanks are allowed wherever a blank is used to delimit words in the command string.

Command Parsing

For all commands, Query Management/400 allows the keywords and variables associated with the commands to be presented as a part of the command string and as part of the extended parameter list on the callable interface. Keywords specified on the command string will take precedence over keywords specified in the extended parameter list if duplicates occur.

Parsing of keywords and values for command strings and for extended parameter lists differ in the following manner:

Command String Keywords and Variables

The following rules describe how Query Management/400 uses command string keywords and variables:

- Query Management/400 assumes all command values are character strings.
- Values can be delimited by either quotation marks (") or apostrophes (').
- Delimiters are not considered part of the value.
- Quotation marks must be doubled if found inside a value that is delimited by quotation marks ('Joe''s' or "Joe""s", for example). (This rule applies to both apostrophes and quotation marks.)
- Quotation marks found inside a value delimited by apostrophes need not be doubled and vice versa.
- Blanks found at the beginning or end of values delimited by quotation marks will not be removed.
- Values that are not delimited by apostrophes or quotation marks will be delimited by a blank or by the end of a command string.
- The values for keywords that are used as integers (such as WIDTH and LENGTH on the PRINT command) can be presented to Query Management/400 as integer values or as character string values. If presented as character strings, they will be converted to integers before being used.

Extended Parameter List Keywords and Variables

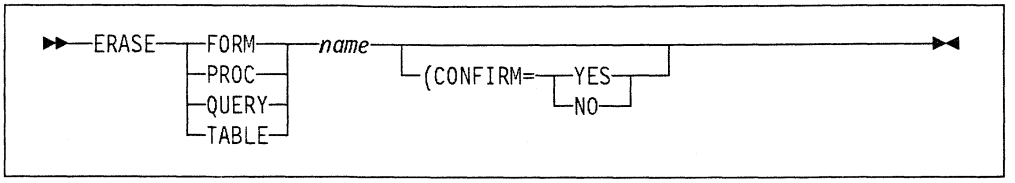
The following rules describe how Query Management/400 uses parameter list keywords and variables:

- Leading and trailing blanks will be stripped from keyword names before being used.
- Trailing blanks will be stripped from keyword values, but leading blanks will not.
- Leading and trailing blanks will not be stripped from variable values before the variable name and value are added to the query management variable pool.
- Apostrophes and quotation marks will not be stripped by Query Management/400.
- Query Management/400 will not remove quotation marks or apostrophes.

- The values for keywords that are used as integers (such as WIDTH and LENGTH on the PRINT command) can be presented to Query Management/400 as integer values or as character string values. If presented as character strings, they will be converted to integers before being used.

ERASE

The ERASE command removes a FORM, PROC, QUERY, or TABLE from the database.



name
Names a FORM, PROC, QUERY, or TABLE to be removed. You must have the appropriate ownership and database or query authorization to remove an object.

CONFIRM=YES | NO
This option provides for a check before performing your ERASE request. The confirmation request occurs only when an existing object in the database is about to be erased. You will be asked if you want the pending ERASE command performed.

CONFIRM=YES forces a display of the confirmation. CONFIRM=NO suppresses a display of the confirmation.

The default value is CONFIRM=YES, but you can change the default by setting the DSQCONFIRM variable as a START command parameter or in the start procedure.

Examples of the ERASE command

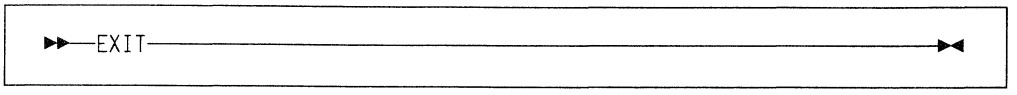
```
ERASE TABLE EMP

ERASE TABLE SMITH.EMP (CONFIRM=YES
```

Command keywords are supported in the extended parameter list on the callable interface as well as in the command string. For more information on using the extended parameter list, see "START" on page 1-27.

EXIT

The EXIT command stops your application's session with Query Management/400 and terminates the associated instance of Query Management/400 in your process. No parameters are allowed with this command. No messages are generated.

The EXIT command is valid only when issued through the Callable Interface.

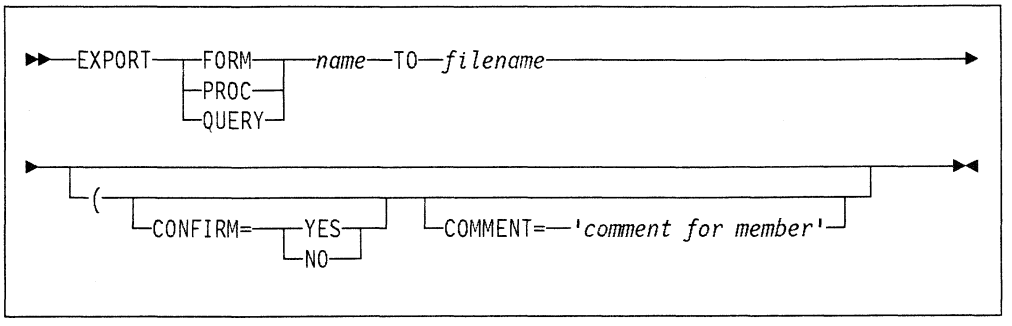


Examples of the EXIT command

See Chapter 3, "Callable Interface" on page 3-1 for examples of programs that use the EXIT command.

EXPORT

The EXPORT command is used to create a file containing the contents of certain Query Management/400 objects. Chapter 6, "Exported Objects" on page 6-1 discusses exported objects in detail. The following objects can be exported: FORM, PROC, or QUERY.



name

Names a FORM, PROC, or QUERY to be exported. This name can be a qualified name.

If the form or query specified is not found, and DSQSCNVT=YES is specified on the START command, Query Management/400 searches for a Query/400 definition with that name. If a query definition is found, the information is used to create a temporary query or form that is usable by Query Management/400.

filename

Names the system file that receives the exported object.

To be consistent across systems, you should not specify more than a single file name without qualifiers or extensions. This enables system defaults to take effect.

Query Management/400 qualifies the single name to fit the data naming requirements of the operating environment.

You must be aware of these naming restrictions and be able to deal with them when transporting Query Management/400 objects between operating environments.

CONFIRM=YES | NO

This option provides for a check before performing your EXPORT request. The confirmation request occurs only when an existing file is about to be replaced. You will be asked if you want the pending change to occur.

CONFIRM=YES forces a display of the confirmation. CONFIRM=NO suppresses a display of the confirmation.

The default value is CONFIRM=YES, but you can change the default by setting the DSQCONFIRM variable as a START command parameter or in the start procedure.

COMMENT=comment for member

Use the comment option to specify the member text when exporting a form, procedure, or query object. Comments are useful for preserving information about the object. Because comments usually include embedded blanks, they

must be enclosed in apostrophes. Apostrophes within a comment must be specified by two adjacent apostrophes.

Examples:

```
COMMENT='SALES QUERY'
COMMENT='THIS QUERY DOESN''T INCLUDE SALES'
```

The maximum comment length in Query Management/400 is 50 characters.

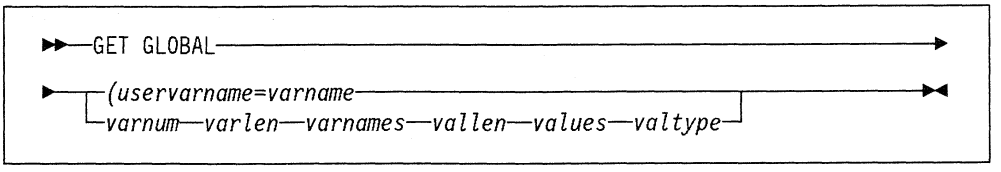
Examples of the EXPORT Command

```
EXPORT QUERY SAMP1 TO SAMP1EX

EXPORT FORM EXLIB/EX1 TO EXLIB/FILE(EX1F) (CONFIRM=YES
```

Command keywords are supported in the extended parameter list on the callable interface as well as in the command string. For more information on using the extended parameter list, see "START" on page 1-27.

GET

The GET command is used to get the value of an Query Management/400 variable and provide it to a user program or procedure. When using the GET command from a procedure, the short version of the command syntax must be used. When using the GET command from a program, the extended version of the command syntax must be used.



GLOBAL
In Query Management/400, the variable varname located in the global variable pool is returned to the requester. If the variable is not found in the global pool, an error message is returned.

uservarname
Name of a procedure variable to contain the varname variable value.

varname
Name of the variable located in the Query Management/400 variable pool. For rules that apply to variable names used across the callable interface, see "Variable Names" on page 3-22.

Extended Parameter List:

varnum

Number of varnames that are requested for this call.

varlen

Length of each varname that is specified.

varnames

Name of the variable located in the Query Management/400 variable pool.

vallen

Length of program storage that is to contain the varname value.

values

Program storage area that is to contain the varname value.

valtype

Data type of the storage area that is to contain the varname value.

Examples of the GET Command

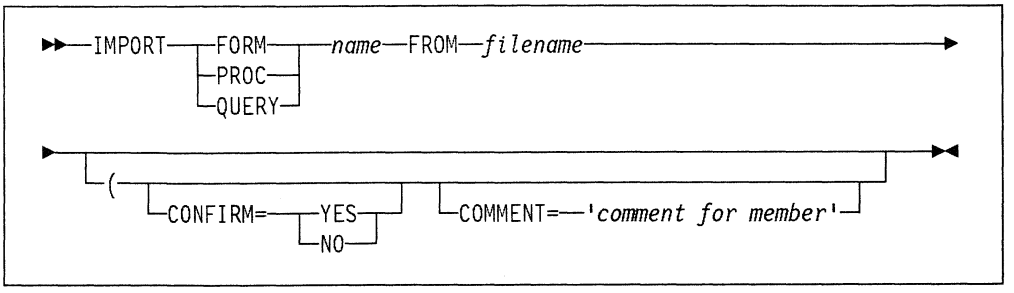
The following are examples of the GET command as used in a PROC. For examples of using the GET command in a program, see Chapter 3, "Callable Interface" on page 3-1.

```
GET GLOBAL (item = DSQAITEM

GET GLOBAL (msg = DSQCIQMG
```

Command keywords are supported in the extended parameter list on the callable interface as well as in the command string. For more information on using the extended parameter list, see "START" on page 1-27.

IMPORT

You can use the IMPORT command to copy a file containing an exported object into one of the following Query Management/400 objects: FORM, PROC, or QUERY. The IMPORT command does not affect the external file.



name
Name of the FORM, PROC, or QUERY to be imported. This can be a qualified name.

filename
Name of the system file that Query Management/400 is to read (i.e., the source file for the imported object).

In order to be consistent across systems, do not specify any more than a single name without qualifiers or extensions; you must allow system defaults to be taken.

CONFIRM=YES | NO
This option provides for a check before performing your IMPORT request. The confirmation request occurs only when an existing object in the database is about to be replaced. You will be asked if you want the pending database changes to occur.

CONFIRM=YES forces a display of the confirmation. CONFIRM=NO suppresses a display of the confirmation.

The default value is CONFIRM=YES, but you can change the default by setting the DSQSCNRM variable as a START command parameter or in the start procedure.

COMMENT=comment for member
Use the comment option to specify the member text when exporting a form, procedure, or query object. Comments are useful for preserving information about the object. Because comments usually include embedded blanks, they must be enclosed within apostrophes. Apostrophes within a comment must be specified by two adjacent apostrophes.

Examples:

```
COMMENT='My form'
COMMENT='This form doesn''t include breaks'
```

The maximum comment length in Query Management/400 is 50 characters.

The IMPORT command is typically used in the following situations:

- To copy or propagate FORM, PROC, and QUERY objects from one Query Management/400 installation to another (the objects are exported by the sending installation and imported by the receiving installation).

- To use a full-function editor outside of Query Management/400. The following is a typical scenario:
  1. First, export the Query Management/400 object. This causes the creation of an external file.
  2. Next, invoke a text editor. Once inside the editor, you can perform normal editing functions like copying and moving.
  3. Once you finish editing, return to Query Management/400 and import the file. The edited object is stored in the database.
- For application programmers who want to migrate SQL queries from program libraries that typically reside outside of Query Management/400 to libraries inside Query Management/400, for purposes of modification and interactive processing (testing).

The IMPORT command copies the contents of the specified file into the database. For SQL queries and procedures, each record in the file becomes a separate line in the object. All files that were exported via the Query Management/400 EXPORT command can be imported.

When importing files containing SQL queries and procedures, Query Management/400 accepts records having a logical record length greater than 79, even though the resulting data may be truncated. If Query Management/400 finds a logical record length greater than 79, it displays a warning message.

If the imported query has a fixed record format (and logical record length greater than 79), then Query Management/400 accepts only data in positions 1 through 79 and ignores the rest.

If the imported form has a fixed record format (and logical record length greater than 150), then Query Management/400 accepts only data in positions 1 through 150 and ignores the rest.

When importing query objects with a logical record length less than 79, Query Management/400 pads the record with blanks up to and including position 79. If the line contains an open delimited string, this padding will then be included within the delimited string and may cause unexpected results.

When importing form objects with a logical record length less than 150, Query Management/400 pads the record with blanks up to and including position 150. If the line contains an open delimited string, this padding will then be included within the delimited string and may cause unexpected results.

When importing SQL QUERY and PROC objects, Query Management/400 does not perform any validation or semantic checking on the contents of the files. Therefore, it is possible to establish QUERY and PROC objects containing nondisplayable characters (this could happen if a program's object file were imported as a QUERY). Also, it is possible to IMPORT SQL statements into the PROC object and vice versa. It is your responsibility to avoid such mistakes, since Query Management/400 contains no provision for "re-categorizing" contents.

Query Management/400 validates FORM objects. If some part of the file fails a validation test, then the object is brought into the database, but you are sent warning messages. It is possible for the file to pass the validation test, yet provide unpredictable results when used for formatting.

Examples of the IMPORT Command

```
IMPORT FORM REPORT1 FROM REPT1EX

IMPORT QUERY SALARYWK FROM JENSON
```

Command keywords are supported in the extended parameter list on the callable interface as well as in the command string. For more information on using the extended parameter list, see "START" on page 1-27.

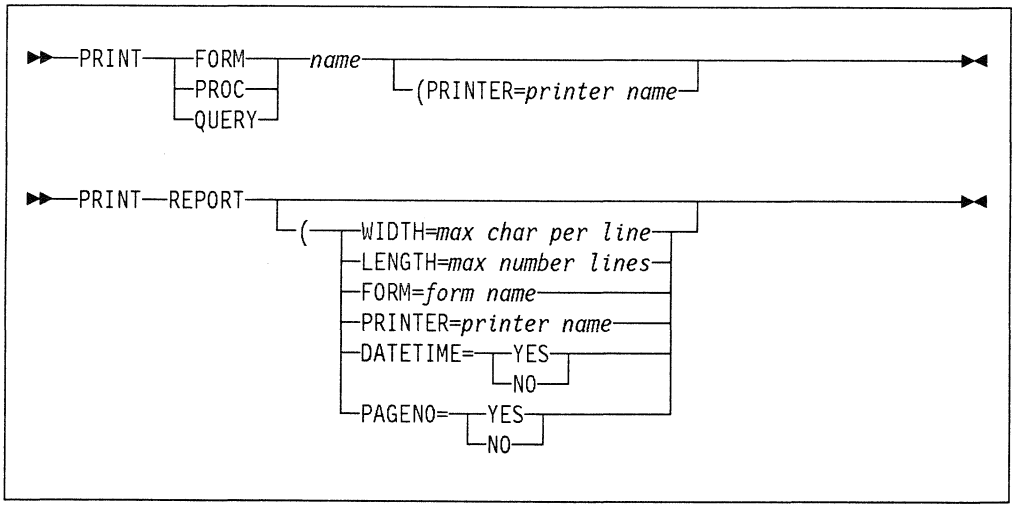
PRINT

The PRINT command is used to print a hard copy listing of an Query Management/400 object.

The PRINT command uses standard system facilities for printing. Query Management/400 does not externalize the printer attributes to your application. Nor does Query Management/400 alter these attributes' values. Instead, it simply honors the printer definitions currently in effect.

An object's printed appearance is very much like its screen appearance. However, on a REPORT, there are some differences between display format and print format.

- The "panel title" line of the displayed REPORT object is replaced with a "page heading" at the top of each page in the printed report (assuming that a page heading has been defined).
- A "page footing" is provided at the bottom of each page of the printed report, but only once at the bottom of the displayed object.



name

The name of the object to be printed. The name specified may be a FORM, PROC, or QUERY in the database.

If the form or query specified is not found, and DSQSCNVT=YES is specified on the START command, Query Management/400 searches for a Query/400 definition with that name. If a query definition is found, the information is used to create a temporary query or form that is usable by Query Management/400.

WIDTH=*maximum characters per print line*

An integer between 22 and 378 inclusive.

Reports that are wider than the print WIDTH will be split between pages. Query objects other than reports are not split between pages. If the object is wider than the print width, the lines in the printout will be truncated on the right.

It is important that you ensure the compatibility of WIDTH with the printer you are using. For example, if your current printer settings identify a 10 pitch device (10 characters per inch) mounted with 8.5 inch wide paper, then a

WIDTH value of 132 results in modified output. The exact result may depend on printer hardware and software. Because Query Management/400 does not know the width of the physical printer, no special message displays when this situation occurs.

If you do not specify this option, then Query Management/400 uses the corresponding system or user default. If this value is not available, then the default is set to 80.

LENGTH=*maximum number of lines per page*

An integer between 1 and 255 inclusive.

When a report is to be printed and the value for LENGTH is inadequate (that is, if the value for LENGTH is less than the total number of lines needed for column headings, page headings and footings, plus the line needed to print the page number and/or date and time), then an error message is generated and the report is not printed.

For LENGTH values within the range allowed, Query Management/400 performs a page eject whenever the number of lines of report data printed on a page is equal to LENGTH.

If you do not specify this option, then Query Management/400 uses the corresponding system or user default. If this value is not available, then the default is set to 66.

FORM=*form name*

The name of the FORM that you want to use to format your data. If no form is specified, the form used in the previous RUN QUERY is used.

PRINTER=*printer name*

The name of the printer that produces the output.

Query Management/400 can be directed to a different printer file through the use of the AS/400 OVRPTRF (Override Printer File) CL command. This command cannot be used to permanently change Query Management/400 printer files. However, to run a PRINT command again and use the SAA default page length and width values, use the CHGPRTF CL command to permanently change the printer file.

DATETIME=YES | NO

This option controls the generation and display of the system date and time on the bottom of each page. When DATETIME=YES, the date and time are placed on the last line of each page. When DATETIME=NO, the system date and time do not print. The default for this option is YES.

PAGENO=YES | NO

This option controls the printing of page numbers on the last line of each page. The default for this option is YES.

Note: You may use any or all of the options associated with the PRINT REPORT command, but each option should only be used once. If an option is used more than once, the last one will be used.

Examples of the PRINT Command

```
PRINT QUERY queryname

PRINT FORM formname

PRINT PROC procname

PRINT REPORT

PRINT REPORT (WIDTH=80 LENGTH=60 DATETIME=YES PAGENO=YES
```

RUN

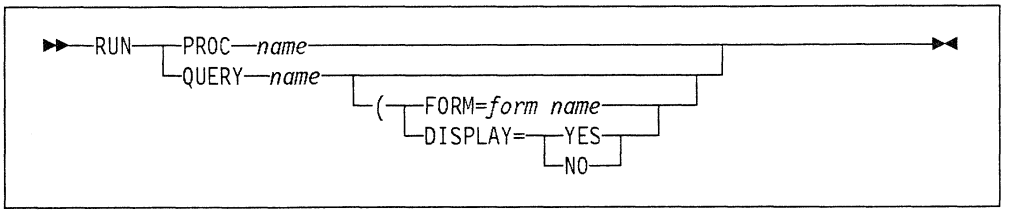
The RUN command processes a PROC or QUERY. When you issue the RUN command, you must identify the PROC or QUERY that you want processed. Therefore, the PROC or QUERY must exist in the database prior to issuing the RUN command.

When used to process a QUERY (SELECT only), the RUN command produces new data, replacing the existing data produced from any previous RUN QUERY.

When running in interactive mode, the results of a query (SELECT only) will be displayed. When running in batch mode, the results will not be displayed.

Normally, a QUERY uses a FORM when run. This can occur in two ways:

1. An existing FORM is explicitly named on the RUN command via the FORM option.
2. A default FORM is created. This default form is constructed using rules that take into consideration the column attributes of the DATA.



name
The name of the QUERY being run.

FORM=*form name*
This option is meaningful only for queries that contain a SELECT statement.

The FORM option specifies the FORM to be used in formatting the REPORT that RUN automatically displays when running in interactive mode.

If the form specified via the FORM option cannot be found (for instance, when the FORM does not exist), then the RUN command is rejected with an error message. If the form does exist but simply *will not work* with the DATA (perhaps different data types for the columns were specified), then Query Management/400 responds with an error message.

If you omit the FORM option, the default FORM is used.

If the form or query specified is not found, and DSQSCNVT=YES is specified on the START command, Query Management/400 searches for a Query/400 definition with that name. If a query definition is found, the information is used to create a temporary query or form that is usable by Query Management/400.

DISPLAY=YES | NO
Use the DISPLAY keyword to indicate whether to display the report. This keyword defaults to YES if you are processing interactively. If you specified batch mode on the START command, this keyword is ignored.

You can use the extended parameter list format for this command. For more information on this format, see "GET" on page 1-13.

Examples of the RUN Command

```
RUN QUERY QN1

RUN PROC WEEKREPT

RUN QUERY SMITH.Q6 (FORM=SMITH.SAL_REPT
```

Command keywords are supported in the extended parameter list on the callable interface as well as in the command string. For more information on using the extended parameter list, see "START" on page 1-27.

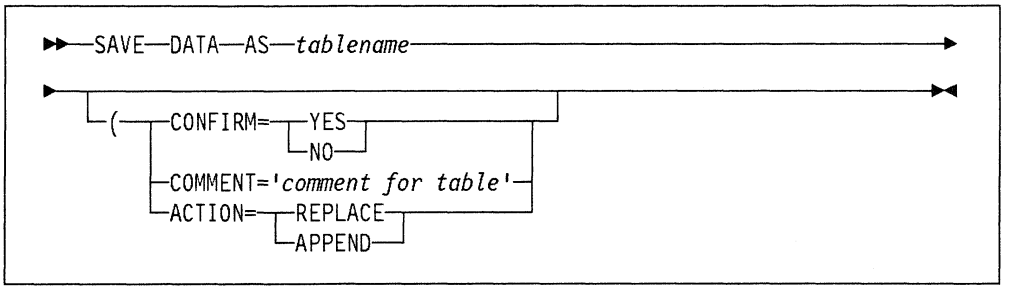
SAVE DATA AS

Use the SAVE DATA AS command to save data in a table in the database. The saved TABLE is named according to the name you specify with the command.

If DATA is saved and a TABLE/VIEW is actually being replaced, then the data must be compatible with the existing definition. Compatible data has matching data types, lengths, and null attributes. Specifically, the number of columns in DATA must match the target, and the columns must have compatible data types and null characteristics. If the two objects are incompatible, then Query Management/400 rejects the SAVE DATA AS command and the database remains unchanged.

If the name on the SAVE DATA AS command already exists as a view in the database, the table on which the view is defined will be changed according to the rules of updating a table through a view.

The column names for a saved TABLE that does not already exist are generated by Query Management/400 using the same algorithm that is used in generating the default column headings in the FORM object. You cannot change the column names.



DATA
Refers to the active result from the previously running QUERY.

tablename
The name of the table or view in which the data is stored in the database. It is normally unqualified.

CONFIRM=YES | NO
This option provides for a check before performing your SAVE request. The confirmation request occurs only when an existing object in the database is about to be replaced. You will be asked if you want the pending database changes to occur.

CONFIRM=YES forces a display of the confirmation. CONFIRM=NO suppresses a display of the confirmation. In either case, a confirmation message is generated to inform you that the SAVE operation is complete.

The default value is CONFIRM=YES, but you can change the default by setting the DSQCONFIRM variable as a START command parameter or in the start procedure.

COMMENT='*comment for table*'
Use this option to supply a comment when saving data as a table. Comments are useful for preserving descriptive information about the object.

Because commentary usually consists of multiple words and embedded blanks, you must enclose it in apostrophes. Apostrophes embedded within the commentary must be entered as two adjacent apostrophes.

Examples:

```
COMMENT='The master EMPLOYEE table-see John (X3971)'

COMMENT='Don''t ERASE this data without telling Phil!'
```

Query Management/400 restricts object commentary to a maximum of 50 characters, excluding the apostrophes.

ACTION=REPLACE | APPEND
The ACTION=REPLACE option causes an existing table or view to be replaced. The ACTION=APPEND option causes the data to be added to the end of an existing table or view. The default is ACTION=REPLACE. The ACTION keyword is ignored if the table or view does not exist.

Note: You can use any or all of the options associated with the SAVE DATA AS command, but each option should only be used once.

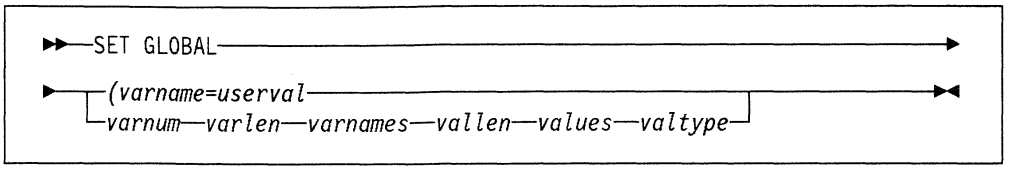
Examples of the SAVE DATA AS Command

```
SAVE DATA AS EMP12

SAVE DATA AS EMP12 (COMMENT='CLASSIC TWO TABLE JOIN'
```

Command keywords are supported in the extended parameter list on the callable interface as well as in the command string. For more information on using the extended parameter list, see "START" on page 1-27.

SET

The SET command is used to set the value of an Query Management/400 variable from the user program or procedure. When using the SET command from a procedure, the short version of the command syntax must be used. When using the SET command from a program, the extended version of the command syntax must be used.



GLOBAL
In Query Management/400, the variable varname located in the global variable pool is set by the requester. If the variable does not exist, a new variable is created. If the variable does exist, its contents are replaced.

varname
Name of the variable located in the Query Management/400 variable pool. For rules that apply to variable names used across the callable interface, see "Variable Names" on page 3-22.

userval
The value that is to be associated with the variable name specified by varname. If it is a constant enclosed in apostrophes, the apostrophes are removed.

Extended Parameter List:

varnum

Number of varnames that are requested for this call.

varlen

Length of each varname that is specified.

varnames

Name of the variable located in the Query Management/400 variable pool.

vallen

Length of program storage that is to contain the varname value.

values

Program storage area that is to contain the varname value.

valtype

Data type of the storage area that is to contain the varname value.

Examples of the SET Command

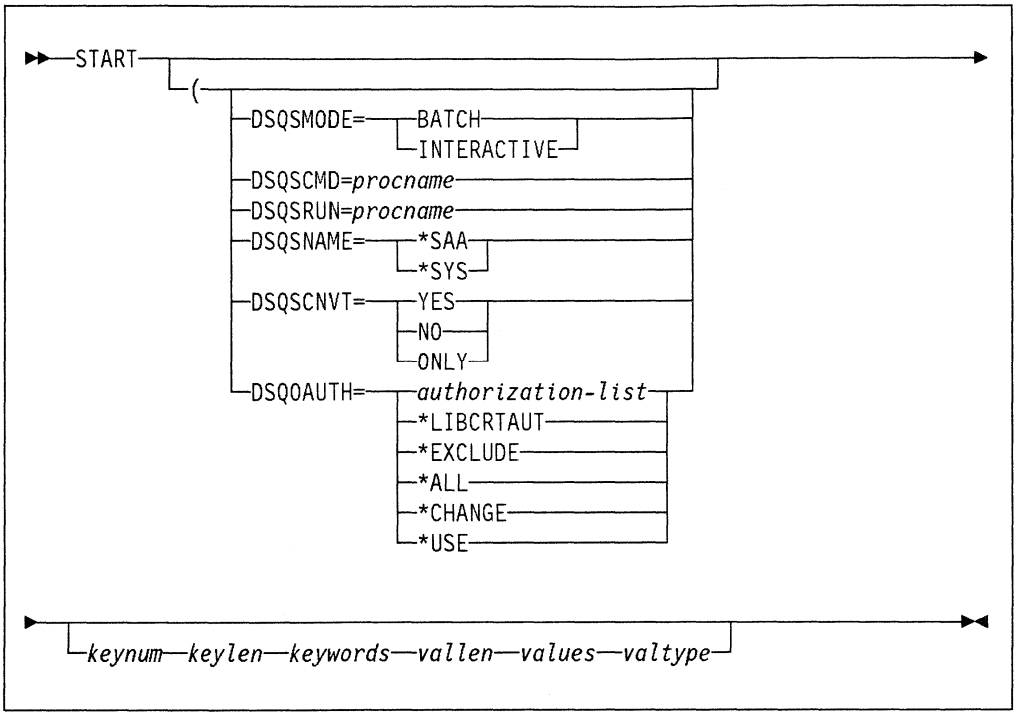
The following are examples of the SET command as used in a PROC. For examples of using the SET command in a program, see Chapter 3, "Callable Interface" on page 3-1.

```
SET GLOBAL (CHARVAR = 'abc'

SET GLOBAL (NUMBVAR = 199
```

Command keywords are supported in the extended parameter list on the callable interface as well as in the command string. For more information on using the extended parameter list, see "START" on page 1-27.

START

The START command provides an interface to start an instance of Query Management/400. This command is only valid when issued through the callable interface. The START command allows for values to be specified that indicate how the Query Management/400 session is to be started.



Extended Parameter List:

keynum

Number of keywords that are passed with this call.

keylen

Length of each specified keyword.

keywords

Name of the START command keyword that is being set.

- DSQSMODE—indicates the mode of Query Management/400 operation when subsequent commands are issued. Valid options are:

  INTERACTIVE

  Allows for the display of screens during Query Management/400 processing. Any reports generated as a result of a RUN QUERY command will be displayed on your screen. Any confirmation messages requiring a response will be displayed on your screen to obtain your reply.

  BATCH

  No screens are displayed during Query Management/400 processing. Any messages requiring a response will result in an error. All other messages will be sent to the job log.

The keyword value set for the DSQSMODE variable on the START command will override any keyword value set for the DSQSMODE variable via the query command procedure.

- DSQSCMD—is the name of a file that is used to start the Query Management/400 session. The SET command is the only type of statement allowed in this procedure. If the DSQSNAME keyword is not specified on the START command, *SAA conventions will be used to find the query command procedure; otherwise, the naming conventions set via the DSQSNAME keyword will be used.
- DSQSRUN—names the Query Management/400 procedure to process after initialization is started.
- DSQSNAME—the naming convention to be used when processing query commands and the SQL query. See "Query Objects" on page 1-2 for more information. The keyword value set for DSQSNAME on the START command will override any keyword value set for DSQSNAME in the query command procedure.

  *SAA — Any qualified query object name specified in commands or query procedures will be of the format `'database.object'`

  *SYS — Any qualified query object name specified in commands or query procedures will be of the format `'library/object'`

- DSQSCNVT—indicates whether Query Management/400 will search for a Query/400 definition object if a Query Management/400 object is not found. The information contained in the query definition is used to create a temporary Query Management/400 object to be used in a command.

  For example, the command `RUN QUERY MYLIB/QRY1` tells Query Management/400 to search for a query management query object named QRY1. If that object is not found, Query Management/400 will search for a query definition object and use the information contained in it to run a query.

  YES — Query Management/400 will search for a Query/400 definition object if a Query Management/400 object is not found.

  NO — Query Management/400 will not search for a Query/400 definition object if a Query Management/400 object is not found.

  ONLY — Query Management/400 will only search for a Query/400 definition object.

- DSQOAUTH—the authority given to any object created by Query Management/400. You can specify a default public authority for all objects created during a query instance by setting a value in the DSQOAUTH keyword in the query command procedure or on the START command. The values you can specify are:

  *LIBCRTAUT
  The authority for the object is the same as the value specified on the CRTAUT parameter of the library in which the object is being created. If the CRTAUT parameter is changed, the new value will not affect the authority of existing objects.

*CHANGE Change authority allows other users to perform all operations on the object except those limited to the owner or controlled by object existence authority and object management authority. A user can change or use the query object in any way, except for deleting it or transferring it to a new owner.

*ALL All authority allows other users to perform all operations on the object except those limited to the owner or controlled by authorization list management rights. A user can do anything with the query object (including erasing it), except for transferring it to a new owner.

*EXCLUDE
Exclude authority prevents other users from doing anything with the query object. Unless given specific types of authority, no user except its owner can use the query object.

*USE Use authority allows other users to run, export, or print the query object, but prevents them from importing it or saving to it.

authorization list name
If you specify the name of an authorization list, its authority is used to control the users ability to use a query object. For more information, see the *Security Guide*.

If you do not specify an authority through the query command procedure, other users will have *EXCLUDE access to the query object.

vallen

Length of program storage that is to contain the keyword value.

values

Program storage area that is to contain the keyword value.

valtype

Data type of the storage area that is to contain the keyword value.

Examples of the START Command

See Chapter 3, "Callable Interface" on page 3-1 for examples of programs that use the START command.

CL Commands

The following control language (CL) commands are commonly used when working with Query Management/400 and writing applications to create Query Management/400 reports. For further information on using CL commands, see *CL Reference*.

ANZQRY (Analyze Query) Command

The Analyze Query (ANZQRY) command allows you to analyze a Query/400 definition (QRYDFN) object for Query Management/400 conversion problems. Query Management/400 returns diagnostic messages about potential differences between Query/400 and Query Management/400 use of query and form information derived from the analyzed QRYDFN object. A completion message shows the highest severity of the problems that are found.

CRTQMFORM (Create Query Management Form) Command

The Create Query Management Form (CRTQMFORM) command allows you to create a query management form from a specified source. The form defines how to format DATA (from running a query) when printing or displaying a report. Form information is encoded in source file member records.

CRTQMQRY (Create Query Management Query) Command

The Create Query Management Query (CRTQMQRY) command allows you to create a query from a specified source. A query is any single SQL statement that can contain variable substitution values. It can be spread over multiple records in a source file member.

DLTQMFORM (Delete Query Management Form) Command

The Delete Query Management Form (DLTQMFORM) command allows you to delete an existing query management form from a library. A generic form name can be used to delete multiple forms from a library or list of libraries.

DLTQMQRY (Delete Query Management Query) Command

The Delete Query Management Query (DLTQMQRY) command allows you to delete an existing query management query from a library. A generic query name can be used to delete multiple queries from a library or list of libraries.

RTVQMFORM (Retrieve Query Management Form) Command

The Retrieve Query Management Form (RTVQMFORM) command allows you to retrieve encoded form source records from a query management form (QMFORM) object. The source records are placed into a source file member that can be edited.

You can also retrieve form source records from a QRYDFN object when the specified QMFORM does not exist.

RTVQMQRY (Retrieve Query Management Query) Command

The Retrieve Query Management Query (RTVQMQRY) command allows you to retrieve an SQL source statement from query management query (QMQRY) object. The source records are placed into a source file member that can be edited.

You can also retrieve query source records from a QRYDFN object when the specified QMQRY object does not exist.

STRQMPRC (Start Query Management Procedure) Command

The Start Query Management Procedure (STRQMPRC) command allows you to run query management procedure that was saved as a member in a source file.

STRQMQRY (Start Query Management Query) Command

The Start Query Management Query (STRQMQRY) command allows you to run an existing query management query. The query runs the SQL statement saved in the query management query. The DATA collected from running an SQL SELECT statement can be displayed, printed, or stored in another database file.

You can also derive the SQL statement from a QRYDFN object when the specified QMQRY object does not exist.

WRKQMFORM (Work with Query Management Form) Command

The Work with Query Management Form (WRKQMFORM) command shows a list of query management forms from a user-specified subset of query management form names. Several query management form-related functions are available from this list.

WRKQMQRY (Work with Query Management Query) Command

The Work with Query Management Query (WRKQMQRY) command shows a list of query management queries from a user-specified subset of query management query names. Several query management query-related functions are available from this list.

Chapter 2. Procedures

You may find yourself creating reports over and over again that use the same Query Management/400 commands. If you do, consider processing these steps together by creating a *procedure*. A procedure allows you to process a set of Query Management/400 commands with a single RUN command.

Procedures also allow flexibility in your application. Your application can be written to run a named procedure. At any time, the procedure can be updated or tailored to fit a new situation, without requiring you to change your application program.

Rules for Creating Procedures

Keep in mind the following rules when creating a procedure:

- Procedures can contain Query Management/400 commands and blank lines. (Blank lines have no effect on the processing of the commands.) They may also optionally contain an H record and a comment V record. The comment record may be used as a text descriptor when importing a procedure.
- Each Query Management/400 command must be in uppercase English letters.
- Each Query Management/400 command must be enclosed in apostrophes or quotation marks.
- Procedures can contain a RUN command that runs another procedure or query.
- The width of a procedure line is limited to the source file record width.
- The width of a query command on a procedure line after procedure parsing is done is limited to 256 characters. Procedure parsing involves stripping leading and trailing blanks and reducing apostrophes and quotation marks.

Example

```
/*H QM4 01 P 01 E V W E R 01 03 90/3/19 14:27 */
/*V 1001 014 Monthy report */
/* This produces the monthly reports. */
'RUN QUERY A'                /* PAYROLL */
'PRINT REPORT (PRINTER=PRT01'
'RUN QUERY B'                /* ACCTS RECEIVABLE */
'PRINT REPORT (PRINTER=PRT01'
```

The format of the H and V records are described in Chapter 6, "Exported Objects" on page 6-1. The text on the V record, Monthly report, is used on the IMPORT PROC command to set the text description on the source file member.

Interaction with a query user depends on the interactive state of Query Management/400. This state is controlled by the startup parameter DSQSMODE on the START command. If you allow the interactive state, there are further considerations when using a procedure. For example, specifying CONFIRM=YES on the ERASE command or DISPLAY=YES on the IMPORT command could cause the procedure to fail.

When a procedure that contains several queries is run, you will see a formatted report display as each query processes. You can then page the report. An exit

from the report returns control to the procedure and causes the next statement to process.

Procedure Interaction

Refer to Chapter 1, "Commands" on page 1-1 to understand what will happen during the processing of each command depending on the mode the procedure is being processed in.

- Procedures are allowed to be called from inside another procedure; this practice is called nesting. Procedures located inside other procedures will use the parameters specified by the first procedure. Therefore, a PRINT command processed in a procedure which has just run a RUN QUERY command will print the data from the RUN QUERY command.
- A nesting level of 15 is allowed by Query Management/400.
- Recursion is not allowed. For example, PROCEDURE A cannot contain the command RUN PROC A.
- The GET command is not functional within a procedure. A GET command within a procedure will result in an informational message sent to the job log that contains the variable name and the value.
- Query Management/400 treats all variable values on a SET command as character strings.

The Procedure Objects

The query procedure is a source physical file. Query Management/400 allows a specific member to be specified on the RUN PROC, IMPORT PROC, EXPORT PROC, PRINT PROC, and ERASE PROC commands by allowing members to be given as part of the query object name. The member name must follow the query object name and be delimited by a parenthesis with no intervening blanks. The following examples show how each of the commands can be changed to point Query Management/400 to a specific member:

```
RUN PROC MYLIB/MYPROCS(MYMEMBER)
PRINT PROC MYLIB/MYPROCS(MYMEMBER)
IMPORT PROC MYPROCS(MYMEMBER) FROM QQMQRYSRC
EXPORT PROC MYLIB/MYPROCS(MYMEMBER) TO QQMQRYSRC(MYMEMBER)
```

If a member is not specified as part of the object name, it is always assumed to be the first member of the file. If an ERASE PROC is issued and more than one member exists, only the first member will be deleted. If a member is not specified and is to be created as part of the IMPORT PROC processing, it will be created with the same name as the PROC file name.

Query Management/400 will process the entire source file when running a RUN PROC or PRINT PROC command. If a PROC file is created on import or export, it will be created with a data length of 79 characters. Truncation will occur on any import or export from a file with a longer record width.

Delimited Areas in the Procedure

All commands must be surrounded by apostrophes or quotation marks. If the command contains a quotation mark, the internal quotation marks are represented by two successive apostrophes('') or quotation marks("").

Example

```
'SAVE DATA AS LASTWKDATA (COMMENT=''Last weeks''''data'''
'IMPORT FORM REPT4 FROM MYLIB/FORMS(REPT4) (CONFIRM=YES'
'SET GLOBAL (TBLENAME=MYFILE'
'SET GLOBAL (CMPVAL2=''Joe A. Customer'''
'RUN QUERY REPT4QRY (FORM=REPT4'
'SAVE DATA AS LASTWKDATA'
'PRINT REPORT'
```

Error Handling

Whenever an error with severity of FAILURE occurs, the processing of the procedure will be stopped and the completion code of the procedure will reflect the error. All messages encountered during the processing of the procedure will be queued to the job log and a summary message will be returned in the communications area.

Chapter 3. Callable Interface

The Query Management/400 callable interface (CI) provides the ability for application programs to perform Query Management/400 functions through calls to the Query Management/400 interface. After completion of a Query Management/400 function, return code and status information is available to the calling program.

The CI consists of the following elements:

- Query Management/400 CI Macros

  Provides a standard interface from different programming languages to Query Management/400 CI modules. The interface also provides common storage and access of program variables between the programming language and Query Management/400.

- Query Management/400

  Provides query and report writing services.

- Query Management/400 CI Modules

  Modules provided by the interface to allow access to the function of Query Management/400.

Callable Interface Description

The callable interface (CI) is an interface that programming languages can use to run Query Management/400 commands. All Query Management/400 commands are supported through the CI.

When a program wishes to run a Query Management/400 command it must first issue a call to start communication between the program and Query Management/400. This call is made to a Query Management/400 supplied routine.

The calling program can issue one or more Query Management/400 commands after the initial start call. Each Query Management/400 command that is processed requires a call to a Query Management/400 supplied routine. Information about the processing of the call is returned to the caller in a return code at the completion of each Query Management/400 command. Other information about the processing of the command is gathered by the CI and stored in shared variables. When control returns to the calling application, these variables are available by reference.

The program issues a call to end communication between the program and Query Management/400 when it no longer needs to use Query Management/400 functions. This call is made to a Query Management/400 supplied routine.

Some considerations in the above processing are:

- A call to Query Management/400 will return to the application only when processing of the command has been completed.
- CI remains in a quiesced state when it is not processing a call.
- All communications to the application will be via return codes and variable data stored in the variable pool or in the Interface Communications Area.
- Commands must be coded in uppercase English letters.
- The length of the passed commands must be at most 256 bytes.

The following diagram shows where the CI fits into the overall scheme.

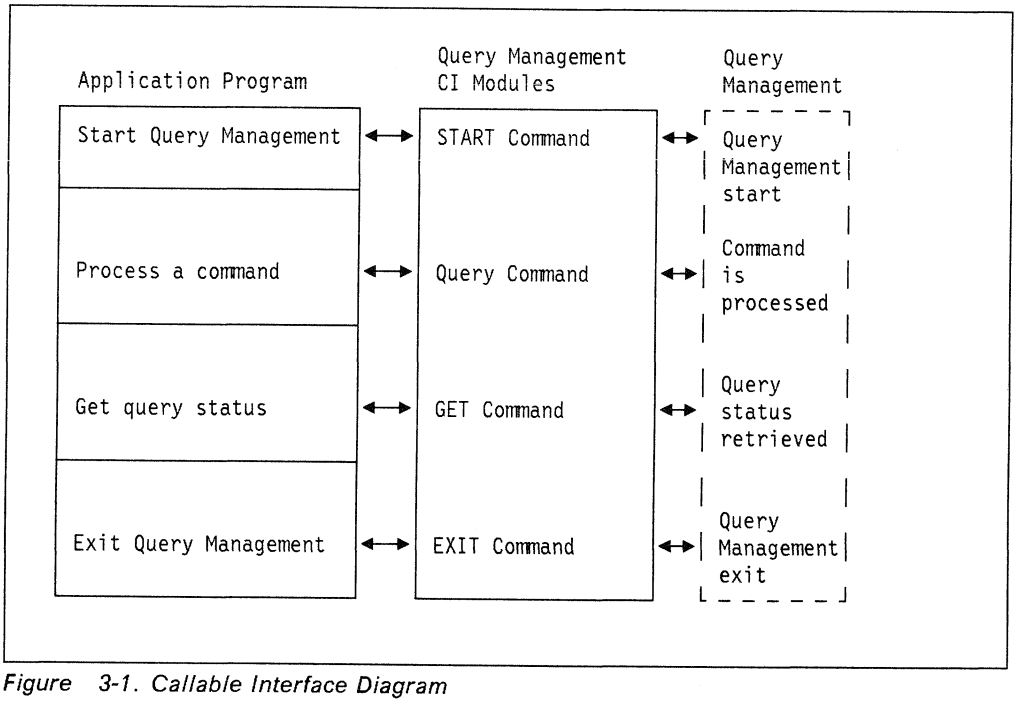


Figure 3-1. Callable Interface Diagram

If an application successfully processes a Query Management/400 command through the CI, the results are, in general, what they would be if the command had been processed on-line without displaying screens.

Query Management/400 CI provides a unique interface communications macro for each language that is supported. The communications macro contains the following definitions, as appropriate:

1. Interface communications area (DSQCOMM)
2. Return codes
3. Call Interface to Query Management/400.

A program that uses the Query Management/400 CI and only uses Query Management/400 commands can be moved from one operating system to another. A Query Management/400 program that is moved must conform to the SAA requirements for the language that the program is written in and be recompiled using the SAA Query communications macro provided by the target SAA Query system.

Interface Communications Area (DSQCOMM)

The Query Management/400 CI communications area is required on all CI calls. Storage for the CI communications area is allocated by the program that is using the Query Management/400 CI.

The START command establishes a unique instance of Query Management/400. As part of START command processing, the CI communications area is updated by Query Management/400. **The CI communications area must never be altered by the application program**. All subsequent calls after the START command must pass the address of the CI communications area that corresponds to an instance of Query Management/400. The user program is responsible for pointing to the correct communications area.

The CI communications area is described by the CI communications macro. There is a unique communications macro for each supported language. For applications to be portable, values must be referenced by variable name rather than by equated value because this value may be different on other systems.

The CI communications area DSQCOMM contains the following information which must **not** be altered by the calling program:

- Return Code

  Indicates the status of Query Management/400 processing after a command is run.

- Instance Identifier

  Identifier that is established by Query Management/400 during processing of the START command.

- Completion Message ID

  Contains the message ID of the message that would have been displayed at the user terminal, if the command had been issued there.

- Query Message ID

  Contains the message ID of a query message, if the command resulted in query processing. This is the message ID of the message that would have been displayed in the job log. This message ID is different across the environments. It is provided to assist in debugging the application and should not be depended on in a portable application.

- START Command Parameter in Error

  Contains the parameter in error when START failed due to a parameter error.

- Cancel Indicator

  Indicates whether the user canceled the command processing while Query Management/400 was running a command.

- Query Derived

  Indicates whether the query information was derived from a Query/400 definition.

- Form Derived

  Indicates whether the form information was derived from a Query/400 definition.

Return Codes

Return codes are returned after each call to the Query Management/400 CI. Return code values are described in the CI communications macro. For applications to be portable, values must be referenced by variable name rather than by equated value because this value may be different on other systems.

Return codes from the CI will include the following:

- Successful processing of the request.
- Command processed with warning condition.
- Command did not process correctly.
- Severe error: Query Management/400 session ended for the applicable instance.

For a definition of each return code, see "C Language Interface" on page 3-5, "COBOL Language Interface" on page 3-11, and "RPG Language Interface" on page 3-16.

Return Variables

When control is returned through the CI, variables will be set which contain information about the completion of a Query Management/400 command. The calling program can obtain the return variables from the variable pool.

Variables are referenced symbolically by name and are obtained from the Query Management/400 variable pool by using the GET command.

Command Message Variables

After a command is initiated by an interactive user, the user sees a message on the screen indicating either a successful completion or an error during processing. This same information is available to the application through command message variables. The following command message variables are provided at the completion of each Query Management/400 command processed through the CI:

DSQCIMNO — Contains the message number. The message number is also returned in the DSQCOMM area.

DSQCIMSG — Contains the first level message text as it would be displayed to the user interactively.

Query Message Variables

If an error occurs when dealing with the processing of a query management query, a query message may be produced to help in problem analysis. For example, an error could have happened during processing of a RUN QUERY command. In this case, additional information may be provided. Query Management/400 message variables consist of the following:

DSQCIQNO — Contains the message number. The message number is also returned in the DSQCOMM area.

DSQCIQMG — Contains the first level message text as it would be displayed to the end user interactively.

DSQCISQL — Contains the SQL return code from the SAA database manager, if any.

Query Management/400 Command Syntax Extension

In order for Query Management/400 to provide variable support to high-level languages such as COBOL, Query Management/400 must have access to the caller's program storage. (For an explanation of Query Management/400 variable support, see "Extended Variable Support" on page 3-21.) A Query Management/400 command extension is used to support commands that require access to the caller's program storage area. The command extension is a different way to specify Query Management/400 command options. The command extension is used for the GET, SET, and START commands because access to the user program area is required to support these commands.

C Language Interface

The Query Management/400 callable interface is accessed by using normal "C" function calls. The exact description of each function call is provided in the callable interface "C" communications include file, DSQCOMMC. The communications include file, DSQCOMMC, is unique for each operating system. Query Management/400 provides two external subroutine calls—DSQCIC and DSQCICE. DSQCIC is used to process Query Management/400 commands that do not require access to program variables. DSQCICE is used to process commands that do require access to program variables.

The following commands must use the DSQCICE function:

START
SET GLOBAL
GET GLOBAL

All other Query Management/400 commands must be specified using the DSQCIC function.

C Variable Support

For C variables that are input character strings (including command strings and START and SET command variables), the user must pass an area that is terminated with a null value. The length of the variable must also include the null value. The length function should be used to obtain the variable length that is passed to Query Management/400. The null value (X'00') indicates the end of a character string.

For C variables that are output character strings (including values set by the GET command), Query Management/400 moves data from Query Management/400 storage to the user's specified storage area and sets the null indicator at the end of the string. If the character string does not fit in the user's storage area, a warning message is issued and the data is truncated on the right. A null indicator is always placed at the end of the data string.

DSQCIC Function Syntax

```
dsqcic(&communication_area,&command_length,&command_string );
```

Where:

- *communication_area* is the structure DSQCOMM.
- *command_length* is the length of command_string.

  The length is specified as a long integer.

- *command_string* is the Query Management/400 command to be processed.

  The command string is specified as an array of character type.

DSQCICE Function Syntax

```
dsqcice (&communication_area,&command_length,&command_string,
         &number_of_parameters,&variable_length,&variable,
         &value_length,&value,&value_type);
```

Where:

- *communication_area* is the structure DSQCOMM.
- *command_length* is the length of command_string.

  The command length is specified as a long integer.
- *command_string* is a pointer to a character string which specifies the Query Management/400 command to be processed.

  The command string is specified as an array of character type.
- *number_of_parameters* is the number of command variables.

  The number of variables is specified as a long integer.
- *variable_length* is the length of each specified variable name.

  The length of the variable name(s) is specified as a long integer type variable or variable array.
- *variable* is the Query Management/400 variable name(s).

  The variable name string is specified as an array of unsigned character type.
- *value_length* is the length of each value associated with the variable.

  The length of the associated values is specified as a long integer type variable or variable array.
- *value* is the value associated with each variable.

  The value string is specified as an array of unsigned character type or a long integer type variable or variable array. The type is specified in the VTYPE parameter.
- *value_type* indicates the Query Management/400 data type of the value string VALUE.

  The value type string contains one of the following values which is provided in the Query Management/400 communications macro:

  DSQ_VARIABLE_CHAR indicates that the value string is unsigned character type.

  DSQ_VARIABLE_FINT indicates that the value string is long integer type.

Interface Communications Area (DSQCOMM)

The Query Management/400 interface communications area is part of the communications macro DSQCOMMC. The interface communications area is described as a structure type named DSQCOMM.

The CI communications area DSQCOMM contains the following information which must ***not*** be altered by the calling program:

dsq_return_code (Unsigned long int)
: Integer that indicates the status of Query Management/400 processing after a command is run.

dsq_instance_ID (Unsigned long int)
: Identifier that is established by Query Management/400 during processing of the START command.

dsq_reserve1 (Char 44)
: Reserved for future use.

dsq_message_id (Char 8)
: Completion message ID.

dsq_q_message_id (Char 8)
: Query message ID.

dsq_start_parm_error (Char 8)
: Parameter in error when START failed due to a parameter error.

dsq_cancel_ind (Char 1)
: Command cancel indicator; indicates whether the user had canceled the command processing while Query Management/400 was running a command:

  dsq_cancel_yes (Char = 1)
  dsq_cancel_no (Char = 0)

dsq_query_derived (Char 1)
: Indicates whether the query information used was derived from a Query/400 definition.

dsq_form_derived (Char 1)
: Indicates whether the form information used was derived from a Query/400 definition.

dsq_reserve2 (Char 17)

dsq_reserve3 (Char 156).

Return Codes

Return codes are returned after each call to the Query Management/400 CI. Return code values are described by the data interface macro. For applications to be portable, values must be referenced by variable name rather than by equated value because this value may be different on other systems.

Return code values for "dsq_return_code" are:

DSQ_SUCCESS Successful processing of the request.

DSQ_WARNING Normal completion with warnings.

DSQ_FAILURE	Command did not process correctly.
DSQ_SEVERE	Severe error: Query Management/400 session ended for the applicable instance. Because the Query Management/400 session ended, additional calls to Query Management/400 cannot be made using this instance ID.

Sample C Language Query CI Program

Figure 3-2 on page 3-9 is an example of a C language program written for Query Management/400 CI.

```
/*****************************************************************/
/* Sample Program:  DSQABFC                                      */
/* C Version of the SAA Query Callable Interface                 */
/*****************************************************************/

/*****************************************************************/
/* Include standard and string "C" functions                     */
/*****************************************************************/
#include <string.h>
#include <stdlib.h>

/*****************************************************************/
/* Include and declare query interface communications area       */
/*****************************************************************/
#include <DSQCOMMC.H>

int main()
    {

struct dsqcomm communication_area;        /* DSQCOMM from include */

/*****************************************************************/
/* Query interface command length and commands                   */
/*****************************************************************/
signed long command_length;
static char start_query_interface[] = "START";
static char set_global_variables[] = "SET GLOBAL";
static char run_query[] = "RUN QUERY Q1";
static char print_report[] = "PRINT REPORT (FORM=F1";
static char end_query_interface[] = "EXIT";

/*****************************************************************/
/* Query command extension, number of parameters and lengths     */
/*****************************************************************/
signed long number_of_parameters;    /* number of variables      */
signed long keyword_lengths[10];     /* lengths of keyword names */
signed long data_lengths[10];        /* lengths of variable data */

/*****************************************************************/
/* Variable data type constants                                  */
/*****************************************************************/
static char char_data_type[] = DSQ_VARIABLE_CHAR;
static char int_data_type[]  = DSQ_VARIABLE_FINT;

/*****************************************************************/
/* Keyword parameter and value for START command                 */
/*****************************************************************/
static char start_keywords[] = "DSQSCMD";
static char start_keyword_values[] = "USERCMD1";
```

Figure 3-2 (Part 1 of 3). Sample C Program

```
/*****************************************************************/
/* Keyword parameter and values for SET command                  */
/*****************************************************************/
#define SIZE_VAL 8
char set_keywords [3][SIZE_VAL];  /* Parameter name array        */
signed long set_values[3];        /* Parameter value array       */

/*****************************************************************/
/* MAIN PROGRAM                                                  */
/*****************************************************************/

/*****************************************************************/
/* Start a Query Interface Session                               */
/*****************************************************************/
    number_of_parameters = 1;
    command_length = sizeof(start_query_interface);
    keyword_lengths[0] = sizeof(start_keywords);
    data_lengths[0] = sizeof(start_keyword_values);
    dsqcice(&communication_area,
            &command_length,
            &start_query_interface[0],
            &number_of_parameters,
            &keyword_lengths[0],
            &start_keywords[0],
            &data_lengths[0],
            &start_keyword_values[0],
            &char_data_type[0]);

/*****************************************************************/
/* Set numeric values into query using SET command               */
/*****************************************************************/
    number_of_parameters = 3;
    command_length = sizeof(set_global_variables);
    strcpy(set_keywords[0],"MYVAR01");
    strcpy(set_keywords[1],"SHORT");
    strcpy(set_keywords[2],"MYVAR03");
    keyword_lengths[0] = SIZE_VAL;
    keyword_lengths[1] = SIZE_VAL;
    keyword_lengths[2] = SIZE_VAL;
    data_lengths[0] = sizeof(long);
    data_lengths[1] = sizeof(long);
    data_lengths[2] = sizeof(long);
    set_values[0] = 20;
    set_values[1] = 40;
    set_values[2] = 84;
    dsqcice(&communication_area,
            &command_length,
            &set_global_variables[0],
            &number_of_parameters,
            &keyword_lengths[0],
            &set_keywords[0][0],
            &data_lengths[0],
            &set_values[0],
            &int_data_type[0]);
```

Figure 3-2 (Part 2 of 3). Sample C Program

```
/*****************************************************************/
/* Run a Query                                                   */
/*****************************************************************/
    command_length = sizeof(run_query);
    dsqcic(&communication_area,&command_length,&run_query [0]);

/*****************************************************************/
/* Print the results of the query                                */
/*****************************************************************/
    command_length = sizeof(print_report);
    dsqcic(&communication_area,&command_length,&print_report[0]);

/*****************************************************************/
/* End the query interface session                               */
/*****************************************************************/
    command_length = sizeof(end_query_interface);
    dsqcic(&communication_area,&command_length,&end_query_interface[0]);
    exit(0);
}
```

Figure 3-2 (Part 3 of 3). Sample C Program

COBOL Language Interface

The Query Management/400 CI is accessed by using normal COBOL function calls. The exact description of each function call is provided in the Query Management/400 COBOL communications macro DSQCOMMB. The communications macro DSQCOMMB is unique for each operating system. Query Management/400 provides an external subroutine called DSQCIB which is used to run all Query Management/400 commands. The parameters that are passed on the call to DSQCIB determine whether program variables are being passed. Program variables must be passed on the following Query Management/400 commands:

START
SET GLOBAL
GET GLOBAL

The other Query Management/400 commands do not specify program variables.

DSQCIB Function Syntax

```
CALL DSQCIB USING DSQCOMM, CMDLTH, CMDSTR.
```

Where:

- *DSQCOMM* is the structure DSQCOMM.
- *CMDLTH* is the length of the command string CMDSTR.

  The length is specified as an integer "PIC 9(8)" variable.
- *CMDSTR* is the Query Management/400 command to be processed.

  The command string is specified as a character string of the length specified by CMDLTH.

DSQCIB Extended Function Syntax

```
CALL DSQCIB USING
            DSQCOMM CMDLTH CMDSTR
            PNUM VNLTH VNAME VLTH VALUE VTYPE.
```

Where:

- *DSQCOMM* is the structure DSQCOMM.
- *CMDLTH* is the length of the command string CMDSTR.

  The length is specified as an integer "PIC 9(8)" variable.
- *CMDSTR* is the Query Management/400 command to be processed.

  The command string is specified as a character string of the length specified by CMDLTH.
- *PNUM* is the number of command variables.

  PNUM is specified as an integer "PIC 9(8)" variable.
- *VNLTH* is the length of each specified variable name.

  The length of the variable name(s) is specified as an integer "PIC 9(8)" variable or variable array.
- *VNAME* is the Query Management/400 variable name(s).

  The variable name string is specified as a character or a structure of characters whose length is the same as specified by VNLTH. An array of characters may be used provided all of the characters are of the same length.
- *VLTH* is the length of each value associated with the variable.

  The length of the associated values is specified as an integer "PIC 9(8)" variable or variable array.
- *VALUE* is the value associated with each variable.

  The value string is specified as a character or a structure of characters or an integer "PIC 9(8)" variable or variable array. The type is specified in the VTYPE parameter.
- *VTYPE* indicates the Query Management/400 data type of the value string VALUE.

  The value type string contains one of the following values which is provided in the Query Management/400 communications macro:

  DSQ-VARIABLE-CHAR indicates that value is character.
  DSQ-VARIABLE-FINT indicates that value is integer "PIC 9(8)".

Interface Communications Area (DSQCOMM)

The Query Management/400 interface communications area is part of the communications macro DSQCOMMB. The interface communications area is described as a structure named DSQCOMM.

The CI communications area DSQCOMM contains the following information which must ***not*** be altered by the calling program:

DSQ-RETURN-CODE "PIC 9(8)"
: Integer that indicates the status of Query Management/400 processing after a command is run.

DSQ-INSTANCE-ID "PIC 9(8)"
Identifier that is established by the Query Management/400 during processing of the START command.

DSQ-RESERVE1 "PIC X(44)"
Reserved for future use.

DSQ-MESSAGE-ID "PIC X(8)"
Completion message ID.

DSQ-Q-MESSAGE-ID "PIC X(8)"
Query message ID.

DSQ-START-PARM-ERROR "PIC X(8)"
Parameter in error when START failed due to a parameter error.

DSQ-CANCEL-IND "PIC X(1)"
Command cancel indicator; indicates whether the user had canceled command processing while Query Management/400 was running a command:

DSQ-CANCEL-YES "VALUE 1"
DSQ-CANCEL-NO "VALUE 0"

DSQ-QUERY-DERIVED "PIC X(1)"
Indicates whether the query information used was derived from a Query/400 definition.

DSQ-FORM-DERIVED "PIC X(1)"
Indicates whether the form information used was derived from a Query/400 definition.

DSQ-RESERVE2 "PIC X(17)"

DSQ-RESERVE3 "PIC X(156)".
Reserved for future use.

Return Codes

Return codes are returned after each call to the Query Management/400 CI. Return code values are described by the data interface macro. For applications to be portable, values must be referenced by variable name rather than by equated value because this value may be different on other systems.

Return code values for "DSQ-RETURN-CODE" are:

DSQ-SUCCESS	Successful processing of the request.
DSQ-WARNING	Normal completion with warnings.
DSQ-FAILURE	Command did not process correctly.
DSQ-SEVERE	Severe error: Query Management/400 session ended for the applicable instance. Because the Query Management/400 session ended, additional calls to Query Management/400 cannot be made using this instance ID.

Sample COBOL Query CI Program

Figure 3-3 is an example of a COBOL language program written for the Query Management/400 CI.

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

* Query interface commands
 01  STARTQI      PIC X(5)  VALUE "START".
 01  SETG         PIC X(10) VALUE "SET GLOBAL".
 01  QUERY        PIC X(12) VALUE "RUN QUERY Q1".
 01  REPT         PIC X(21) VALUE "PRINT REPORT (FORM=F1".
 01  ENDQI        PIC X(4)  VALUE "EXIT".
* Query command length
 01  QICLTH       PIC 9(8) USAGE IS COMP-4.
* Number of variables
 01  QIPNUM       PIC 9(8) USAGE IS COMP-4.
* Keyword variable lengths
 01  QIKLTHS.
   03   KLTHS     PIC 9(8) OCCURS 10 USAGE IS COMP-4.
* Value Lengths
 01  QIVLTHS.
   03   VLTHS     PIC 9(8) OCCURS 10 USAGE IS COMP-4.
* Start Command Keyword
 01  SNAMES.
   03  SNAME1   PIC X(7) VALUE "DSQSCMD".
* Start Command Keyword Value
 01  SVALUES.
   03  SVALUE1   PIC X(8) VALUE "USERCMD1".
* Set GLOBAL Command Variable Names to set
 01  VNAMES.
   03  VNAME1   PIC X(7) VALUE "MYVAR01".
   03  VNAME2   PIC X(5) VALUE "SHORT".
   03  VNAME3   PIC X(7) VALUE "MYVAR03".
* Variable value parameters
 01  VVALUES.
   03   VVALS     PIC 9(8) OCCURS 10 USAGE IS COMP-4.
 01   TEMP     PIC 9(8)          USAGE IS COMP-4.
```

Figure 3-3 (Part 1 of 2). Sample COBOL Program

```
 PROCEDURE DIVISION.
*
* Start a query interface session
     MOVE 0 TO QICLTH.
     INSPECT STARTQI TALLYING QICLTH FOR CHARACTERS.
     MOVE 0 TO TEMP.
     INSPECT SNAME1 TALLYING TEMP FOR CHARACTERS.
     MOVE TEMP TO KLTHS(1).
     MOVE 0 TO TEMP.
     INSPECT SVALUE1 TALLYING TEMP FOR CHARACTERS.
     MOVE TEMP TO VLTHS(1).
     MOVE 1 TO QIPNUM.
     CALL DSQCIB  USING DSQCOMM, QICLTH, STARTQI,
                        QIPNUM, QIKLTHS, SNAMES,
                        QIVLTHS, SVALUES, DSQ-VARIABLE-CHAR.
*
* Set numeric values into query variables using SET GLOBAL command
     MOVE 0 TO QICLTH.
     INSPECT SETG TALLYING QICLTH FOR CHARACTERS.
     MOVE 0 TO TEMP.
     INSPECT VNAME1 TALLYING TEMP FOR CHARACTERS.
     MOVE TEMP TO KLTHS(1).
     MOVE 0 TO TEMP.
     INSPECT VNAME2 TALLYING TEMP FOR CHARACTERS.
     MOVE TEMP TO KLTHS(2).
     MOVE 0 TO TEMP.
     INSPECT VNAME3 TALLYING TEMP FOR CHARACTERS.
     MOVE TEMP TO KLTHS(3).
     MOVE 4 TO VLTHS(1).
     MOVE 4 TO VLTHS(2).
     MOVE 4 TO VLTHS(3).
     MOVE 20 TO VVALS(1).
     MOVE 40 TO VVALS(2).
     MOVE 84 TO VVALS(3).
     MOVE 3 TO QIPNUM.
     CALL DSQCIB  USING DSQCOMM, QICLTH, SETG,
                        QIPNUM, QIKLTHS, VNAMES,
                        QIVLTHS, VVALUES, DSQ-VARIABLE-FINT.
*
* Run a Query
     MOVE 0 TO QICLTH.
     INSPECT QUERY TALLYING QICLTH FOR CHARACTERS.
     CALL DSQCIB USING DSQCOMM, QICLTH, QUERY.
*
* Print the results of the query
     MOVE 0 TO QICLTH.
     INSPECT REPT TALLYING QICLTH FOR CHARACTERS.
     CALL DSQCIB USING DSQCOMM, QICLTH, REPT.
*
* End the query interface session
     MOVE 0 TO QICLTH.
     INSPECT ENDQI TALLYING QICLTH FOR CHARACTERS.
     CALL DSQCIB USING DSQCOMM, QICLTH, ENDQI.
     STOP RUN.
```

Figure 3-3 (Part 2 of 2). Sample COBOL Program

RPG Language Interface

The Query Management/400 callable interface is accessed by using normal RPG function calls. The exact description of each function call is provided in the Query Management/400 RPG communications include member DSQCOM. Query Management/400 provides an external subroutine called DSQCIR which is used to run all Query Management/400 commands. The parameters that are passed on the call to DSQCIR determine whether program variables are being passed. Program variables must be passed on the following Query Management/400 commands:

START
SET GLOBAL
GET GLOBAL

The other Query Management/400 commands do not specify program variables.

DSQCIR Function Syntax

```
C                   CALL DSQCIR
C                   PARM            DSQCOM
C                   PARM            CMDLTH
C                   PARM            CMDSTR
```

Where:

- *DSQCOM* is the structure DSQCOM.
- *CMDLTH* is the length of the command string CMDSTR.

  The length is specified as a 4-byte binary field.
- *CMDSTR* is the Query Management/400 command to be processed.

  The command string is specified as a character string of the length specified by CMDLTH.

DSQCIR Extended Function Syntax

```
C                   CALL DSQCIR
C                   PARM            DSQCOM
C                   PARM            CMDLTH
C                   PARM            CMDSTR
C                   PARM 1          PNUM
C                   PARM            KLTH
C                   PARM            KWORD
C                   PARM            VLTH
C                   PARM            VALUE
C                   PARM            VTYPE
```

Where:

- *DSQCOM* is the structure DSQCOM.
- *CMDLTH* is the length of the command string CMDSTR.

  The length is specified as a 4-byte binary field.
- *CMDSTR* is the Query Management/400 command to be processed.
- The command string is specified as a character string of the length specified by CMDLTH.
- *PNUM* is the number of command keywords.

PNUM is specified as a 4-byte binary field.

- *KLTH* is the length of each specified keyword.

  The length of the keyword or keywords is specified as a 4-byte binary field.

- *KWORD* is the Query Management/400 keyword or keywords.

  The keyword string is specified as a character or a structure of characters whose length are the same as specified by KLTH. An array of characters may be used, provided all of the characters are of the same length.

- *VLTH* is the length of each value associated with the keyword.

  The length of the associated values is specified as a 4-byte binary field.

- *VALUE* is the value associated with each keyword.

  The value string is specified as a character or a structure of characters or a 4-byte binary field. The type is specified in the VTYPE parameter.

- *VTYPE* indicates the Query Management/400 data type of the value string VALUE.

  The value type string contains one of the following values, which is provided in the Query Management/400 communications include member:

  DSQVCH indicates that value is character.
  DSQVIN indicates that value is an integer (4-byte binary).

Interface Communications Area (DSQCOMMR)

The Query Management/400 interface communications area is part of the communications include member DSQCOMMR. The interface communications area is described as a structure named DSQCOM.

The CI communications area DSQCOM contains the following information that must **not** be altered by the calling program:

DSQRET Binary (4 bytes)
: Integer that indicates the status of Query Management/400 processing after a command is run.

DSQINS Binary (4 bytes)
: Identifier that is established by the Query Management/400 during processing of the START command.

DSQRS1 Character (44 bytes)
: Reserved for future use.

DSQMSG Character (8 bytes)
: Completion message ID.

DSQQMG Character (8 bytes)
: Query message ID.

DSQSPE Character (8 bytes)
: Parameter in error when START failed due to a parameter error.

DSQCNL Character (1 byte)
: Command cancel indicator; indicates whether the user had canceled command processing while Query Management/400 was running a command:

  DSQCLY "VALUE 1"
  DSQCLN "VALUE 0"

DSQQDR Character (1 byte)
Indicates whether the query information used was derived from a Query/400 definition.

DSQDRY "VALUE 1" — Object was derived
DSQDRN "VALUE 0" — Object was not derived

DSQFDR Character (1 byte)
Indicates whether the form information used was derived from a Query/400 definition.

DSQDRY "VALUE 1" — Object was derived
DSQDRN "VALUE 0" — Object was not derived

DSQRS2 Character (23 bytes)
Reserved for future use.

DSQRS3 Character (156 bytes)
Reserved for future use.

Return Codes

Return codes are returned after each call to the Query Management/400 CI. Return code values are described by the data interface. For applications to be portable, values must be referenced by variable name rather than the equated value because this value may be different on other systems. **Return code** values for DSQRET are:

DSQSUC	Successful processing of the request.
DSQWAR	Normal completion with warnings.
DSQFAI	Command did not process correctly.
DSQSEV	Severe error: Query Management/400 session ended for the applicable instance. Because the Query Management/400 session ended, additional calls to Query Management/400 cannot be made using this instance ID.

Sample RPG Language Query CI Program

Figure 3-4 on page 3-19 is an example of an RPG language program written for the Query Management/400 CI.

```
*****************************************************************
*                                                               *
*          SAMPLE RPG PROGRAM USING QUERY INTERFACE             *
*          ----------------------------------------             *
*                                                               *
*  1) Include member DSQCOMMMR contains the communications      *
*     area to be passed to the query interface.                 *
*  2) Command name lengths, command names,                      *
*     variable name lengths, variable names,                    *
*     variable value lengths, variable values,                  *
*     are loaded as compile time arrays.                        *
*  3) It is necessary to pass all interface lengths and         *
*     numeric variable information in binary format.            *
*                                                               *
*****************************************************************
H
 *
 * Compile time arrays of command   name   lengths  and  values
 *                         variable  name   lengths  and  values
 *                         variable content lengths  and  values
 *
E                   CNL     1   7  9 0 CNV    25     commands
E                   VNL     1   3  9 0 VNV     7     variable names
E                   VCL     1   3  9 0 VCV     9 0   variable values
 *
I            DS
I                                      B   1  280CNL
I            DS
I                                      B   1  120VNL
I            DS
I                                      B   1  120VCL
I            DS
I                                      B   1   40BINARY
```

Figure 3-4 (Part 1 of 3). Sample RPG Program

```
 *
 * Pull in the communications area
 *
I/COPY DSQCOMMR
 *
 * Start a query interface session:
 *
C                   CALL DSQCIR
C                   PARM           DSQCOM          comms area
C                   PARM           CNL,1           command length
C                   PARM           CNV,1           START
C                   PARM 1         BINARY          # keywords
C                   PARM           CNL,6           keyword length
C                   PARM           CNV,6           DSQSCMD
C                   PARM           CNL,7           value length
C                   PARM           CNV,7           USERCMD1
C                   PARM DSQVCH    DATA    4       CHAR
 *
 * Set numeric values into query variables using SET GLOBAL command:
 *
C                   CALL DSQCIR
C                   PARM           DSQCOM          comms area
C                   PARM           CNL,2           command length
C                   PARM           CNV,2           SET GLOBAL
C                   PARM 3         BINARY          # variables
C                   PARM           VNL             name lengths
C                   PARM           VNV             name values
C                   PARM           VCL             variable lengths
C                   PARM           VCV             variable values
C                   PARM DSQVIN    DATA            FINT
 *
 * Run a query:
 *
C                   CALL DSQCIR
C                   PARM           DSQCOM          comms area
C                   PARM           CNL,3           command length
C                   PARM           CNV,3           RUN QUERY Q1
 *
 * Print the results of the query:
 *
C                   CALL DSQCIR
C                   PARM           DSQCOM          comms area
C                   PARM           CNL,4           command length
C                   PARM           CNV,4           PRINT REPORT
 *                                                   (FORM=F1
 * End the query interface session:
 *
C                   CALL DSQCIR
C                   PARM           DSQCOM          comms area
C                   PARM           CNL,5           command length
C                   PARM           CNV,5           EXIT
 *
C                   SETON                      LR
 *
```

Figure 3-4 (Part 2 of 3). Sample RPG Program

```
**  CNL/CNV          Command lengths and command values
000000005START
000000010SET GLOBAL
000000012RUN QUERY Q1
000000021PRINT REPORT (FORM=F1
000000004EXIT
000000007DSQSCMD
000000008USERCMD1
**  VNL/VNV          Variable name lengths and variable names
000000007MYVAR01
000000007MYVAR02
000000007MYVAR03
**  VCL/VCV          Variable value lengths and variable values
000000004000000020
000000004000000040
000000004000000084
```

Figure 3-4 (Part 3 of 3). Sample RPG Program

Extended Variable Support

A variable is a named entity within Query Management/400 which can be assigned a value. Extended variable support allows applications to define global variables within Query Management/400.

Variables may be used as substitution values in SQL queries and are available when using the CI. Once a variable is created, it is available to the Query Management/400 session for the life of the session. In addition to application-defined variables, Query Management/400 maintains a set of product variables. These variables are also available to SQL queries and the CI.

Creating Variables

Variables are created implicitly at run time, when they are first referenced during SQL query processing. They are also created explicitly, when they are set to a specific value using the SET GLOBAL command. When a variable is created implicitly, the variable value exists only during processing of the RUN QUERY command.

Referencing Variables

Variables may be referenced by specifying the variable name within an SQL query or a user program via the GET GLOBAL command. When a variable name is referenced within an SQL query, the variable name must be prefixed with an ampersand (&) in order for Query Management/400 to recognize it as a variable. For example:

```
SELECT * FROM &TNAME
```

The value &TNAME is considered a Query Management/400 variable.

Variable Names

The following rules apply when you use variables in SQL queries across the callable interface.

- Names can contain the following characters:
  - **Letters**. A letter is any of the single-byte characters (A through Z, or any alphabetic character from a national alphabet).
  - **Arabic numbers** (0 through 9)
  - **Underscore** (_)
- Variable names must start with a letter (see exception for variable names used in SQL queries below).
- Names cannot be longer than 18 characters.
- Variable names that are used in SQL queries must be preceded by an ampersand (&), and the next character must be a single-byte character set letter. The ampersand does not take up one of the 18 characters allowed for the name. Be aware that the ampersand character delimits the beginning of a variable name; you cannot have more than one ampersand in a variable name because each ampersand would delimit the beginning of a distinct variable name.
- User-defined variables should not start with DSQ.

The following are valid variable names:

```
In an SQL Query              In the GET/SET Command
-------------------          ------------------------
&I_OWE_YOU                   I_OWE_YOU
&MYVAR123                    MYVAR123
&THIS_IS_A_BIG_NAM           THIS_IS_A_BIG_NAME
```

Variable Values

Character variable values may consist of any value up to 55 bytes in length. Integer variable values may be any integer within the limits of the operating system. The value must observe the rules of SQL when used in an SQL query.

Query Management/400 Defined Variables

Query Management/400 provides global variables which may be useful to user programs. The current set of Query Management/400 variables can be used to determine the current status of the Query Management/400 environment and particular objects. Query Management/400 variables can't be altered by the user, program, or procedure. A subset of these variables may be set by the user using the query command procedure which is specified on the START command. (See "START" on page 1-27.)

The following variables are available in Query Management/400. The length given is the maximum length for the variable.

DSQAAUTH — Current connect authorization ID. This name will contain the name of the user profile under which the job is running.

Type Character
Length 10
Value -

DSQOAUTH — Default object public authority to be given to objects created through query commands. Refer to "START" on page 1-27 for a description of the values.

Type Character
Length 10
Value

- *LIBCRTAUT
- *EXCLUDE
- *ALL
- *USE
- *CHANGE
- An authorization list name

DSQSNAME — Naming convention be used. Refer to "START" on page 1-27 for a description of this variable.

Type Character
Length 4
Value

- SAA
- *SYS

DSQAPRNM — Current default printer.

Type Character
Length 10
Value

- *SAME
- *JOB
- Printer Device Name

DSQCATTN — Last command cancel indicator.

Type Character
Length 3
Value

- YES
- NO

DSQCISQL — Last SQL return code.

Type Integer
Length 4
Value See the *SQL/400 User's Guide*.

DSQAROWS — Current number of rows fetched for data.

Type Integer
Length 4
Value 0 - maximum # rows

DSQAROWC — Current data is completed.

Type Character
Length 3
Value

- YES
- NO

DSQSMODE — Current processing mode.

Type Character

Length 11
Value
- BATCH
- INTERACTIVE

DSQCONFIRM — Confirm processing default.

Type Character
Length 3
Value
- YES
- NO

DSQSCNVT — Allows the use of information derived from a query definition.

Type Character
Length 4
Value
- NO
- YES
- ONLY

DSQCIMNO — The query message ID. It is the same value that is returned in the query message line of the communications area.

Type Character
Length 8
Value -

DSQCIQNO — The message ID. It is the same value that is returned in the completion message line of the communications area.

Type Character
Length 8
Value -

DSQCIMSG — Contains the message text as it would be displayed to the user interactively.

Type Character
Length 55
Value -

DSQCIQMG — Contains the query message text as it would be displayed to the user interactively.

Type Character
Length 55
Value -

Chapter 4. Query Capability

Query Management/400 supports queries against relational data using SQL. The basic statements, SELECT expressions, data definition, and authorization statements defined in *SQL/400* Reference* are specifically supported. By using these SQL features in a query, your application can perform table definitions, data access authorizations, database queries and data insertions, updates, and deletions. The results of a query can be displayed or printed as a report.

Queries can be created, named, stored, and retrieved. Stored queries can be shared among multiple users and applications. The queries used by your application can be defined and stored at application development time, or they can be created by your application and used by Query Management/400 at application run time.

Stored queries allow flexibility in two ways. First, you can change a query and store it independently from your application program. Second, queries can contain variables. The values assigned to the variables can be set prior to, or in conjunction with your application. Both of these facilities allow data query parameters to be changed without rewriting or recompiling your application.

Rules for Creating Queries

The query object is a text string containing an SQL statement. The string can contain comments. The SQL statement can contain variables. A variable can appear in any clause of the query and can represent anything that can be written into a query, such as column names, search conditions, subselects, or specific values.

The following is an example of a query.

```
-- This query lists the name, years of employment, and salary for
-- employees in a specified department.  The department is a variable
-- and should be set before the query is run.
   SELECT NAME, YEARS, SALARY        -- names the columns used
   FROM Q.STAFF                      -- names the table used
   WHERE DEPT=&DEPTNUM               -- variable selection condition
```

Query Management/400 strips the comments and performs variable substitution before the query is passed to the database manager for processing.

The following restrictions apply to queries handled by Query Management/400:

- A query is limited by the size of the source file.
- A single line of the query cannot exceed 79 bytes.
- Substitution variable values can be up to 55 characters long.
- Comments are preceded by a double hyphen (--). Everything between the double hyphen and the end of the line is considered to be part of the comment.
- The total query cannot exceed 32KB minus 1 byte (after comments and blanks are removed and variable substitution is made).

Chapter 5. Report FORM Definition

How Applications Can Use the FORM

An application can create or alter a FORM by directly changing or creating the exported FORM.

You may use an application to export an existing FORM from Query Management/400, change it, import the FORM, and then format a report. But a FORM does not have to be exported every time. An application can access and change an existing exported FORM, and then import it into Query Management/400 for reporting.

You can also import just part of a FORM: only the header (H) record followed by the T and R records for column information, for example. The rest of the FORM fields can be filled in by defaults.

Formatting Terminology

In order to understand all of the options available in the FORM, the following two figures, Figure 5-1 and Figure 5-2, show you the effect of some of the FORM options on a formatted report.

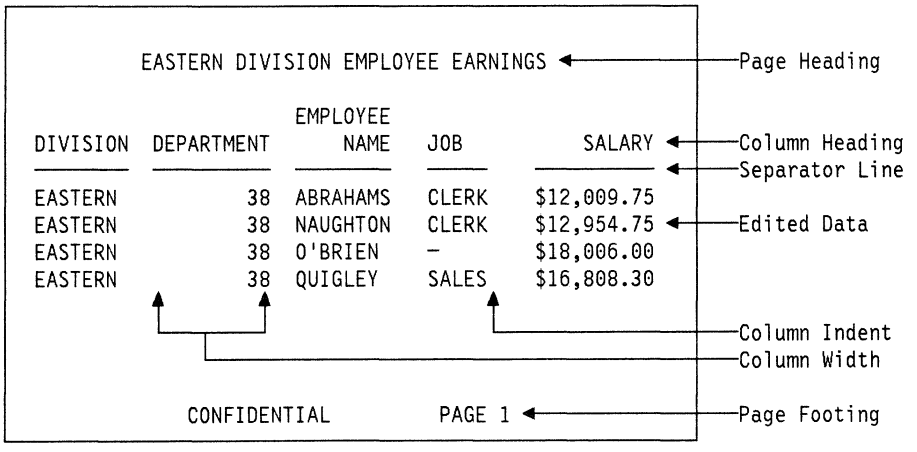


Figure 5-1. Basic Parts of a Report

"Edited Data" is information from the database that displays according to the relevant edit code.

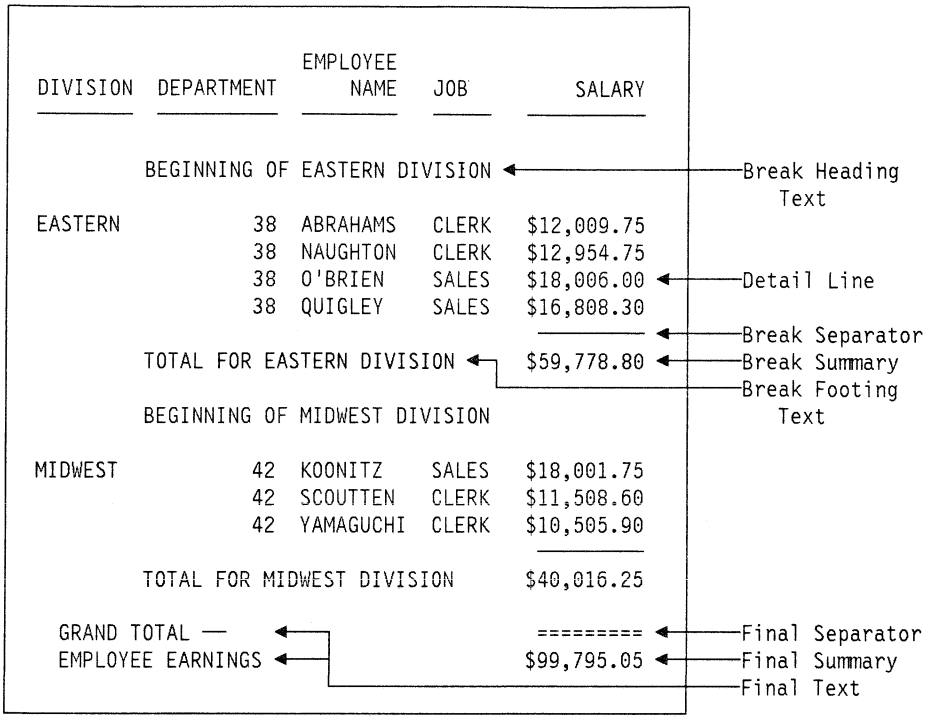


Figure 5-2. Basic Parts of a Report with One Level of Control Break

The FORM object contains fields that describe the report. Query Management/400 supports up to 255 columns of information. Query Management/400 has a limit of 32KB of available data from the database for any one row. Defaults are provided for all the fields except "Usage." Defaults depend on the resulting data of the processed query.

Keywords used in the FORM must be in uppercase English. Text fields (headings, footings, and final text) can be in uppercase and lowercase letters.

The following is a description of the fields in the FORM object.

COLUMN Fields

Each column in a Query Management/400 report is described by a set of field values. The values for the *n*th column in the FORM are applied to the *n*th column selected by the query with which it is used. The following sections describe the fields available for use in defining the Column fields.

Figure 5-3 shows the defaults and possible values for the attributes on the *Column* field.

Figure 5-3. Default Values for Column Fields

Attribute	Default	Possible Values
Column heading	Column heading in table	1 - 62 characters with up to 8 underscores
Usage	—	AVG, MIN, MAX, SUM, COUNT, BREAK1 - BREAK6, AVERAGE, MINIMUM, MAXIMUM, OMIT
Indent	2	0 to 999
Width	Depends on data type used.	1 to 32,767 SBCS
Datatype	Data type of field in table	CHAR, NUMERIC
Edit code—numeric	Edit code of field in table	E, D, I, J, K, L, P
Edit code—character	C	C, CW, CT
Seq	Column number	1 to 999

Column Heading

This field represents the heading for a column in the report.

A heading can be up to 62 characters long.

You can embed underscore characters in the heading and use them to indicate a new line for multiple-line headings. Query Management/400 processes a maximum of eight (8) underscores in a heading. Leading and trailing underscores produce blank segments before and after the column headings.

For example, a column heading of "AMOUNT_LAST_INCREASE" results in the following 3-line column heading:

```
AMOUNT
 LAST
INCREASE
```

Consecutive underscores (in any position) will introduce blank lines.

Note that the underscore rule prevents you from seeing an underscore character in a column heading. The only exception to this rule is when more than eight (8) underscores appear, in which case the extra underscores print as part of the last line of the heading.

Whenever you specify multiple-line headings, Query Management/400 automatically centers the smaller lines within the space of the longest line. Headings for character data are automatically left-justified and headings for numeric data are automatically right-justified. Data justification takes place within the width of the column. The width specification must reflect the length of the longest segment of this field.

If the number of characters in this field is greater than the number of characters specified in width, then the field truncates to the width specified for the column.

If you do not specify a column heading, Query Management/400 provides a run-time default. You cannot cause a column to be shown without a heading by

importing a form with a blank heading unless the database definition for the column indicates it should not have a column heading.

Usage

This field determines use of the column in the detail line of the formatted report result.

There can be only one *usage* specified for each column. If you want a column to have more than one usage, you must select the column multiple times in the query and define a usage code for each column in the FORM.

The usage options are:

[blank]
Column to be included in the report.

OMIT
Column to be excluded from the report.

AVERAGE, COUNT, FIRST, LAST, MAXIMUM, MINIMUM, SUM
These keywords name aggregating usages that summarize the data in a column. The result of the usage is given at a break or final summary.

Usage Code	Definition
AVERAGE (or AVG)	the average of the values in the column.
COUNT	the count of the values in the column.
FIRST	the first value in the column.
LAST	the last value in the column.
MAXIMUM (or MAX)	the maximum value in the column.
MINIMUM (or MIN)	the minimum value in the column.
SUM	the sum of the values in the column.

AVERAGE and SUM work only on numeric data; COUNT, FIRST, LAST, MAXIMUM, and MINIMUM work with character data as well as with numeric data.

When you compare characters using MAXIMUM and MINIMUM, the shorter string is padded with blanks and the strings are compared based on the internal binary codes. For example, the character string "ab" is greater than the character string "aaa". There is no special processing on a character by character basis. Be aware when you use MAXIMUM and MINIMUM on applications ported from one system to another that the collating sequences for the different machines may not be the same. Since EBCDIC and ASCII do not collate characters identically, using MAXIMUM or MINIMUM on different systems may produce different results.

The following rules apply to the aggregating usages AVERAGE, COUNT, FIRST, LAST, MAXIMUM, MINIMUM, and SUM.

1. If aggregation overflow occurs, the value in the field is represented by ">>>>" for the width of the column.
2. If the aggregation cannot be displayed due to the column width being too small, the value in the field is represented by "*****" for the width of the column.

BREAK1

BREAK1 is the value used to specify a column as the first level, or highest, *control break.* A control break is the break point where the column value changes.

For example, if a set of rows of employees is ordered by department number and job title, a BREAK1 can be used to total the salaries of all the employees in the department, and a BREAK2 can be used to total the salaries by job title within department. Each time a row with a different job title is read, a BREAK2 generates and displays the appropriate data and totals. Each time a row with a different department number is read, a BREAK2 and BREAK1 generate, and both sets of appropriate data display.

Before each break summary displays, a line is placed in the report consisting of a row of hyphens ("−") under any displayed column with an aggregation usage. You can suppress this line via an option in the "Options" part of the FORM. A blank line is normally placed after each set of break data.

The aggregation usages (AVERAGE, COUNT, FIRST, LAST, MAXIMUM, MINIMUM, SUM) may be used at control breaks. For example, summary data appears as subtotals of all columns with a usage of SUM, or an average of the columns with a usage of AVERAGE.

The data printed as part of the break is determined by the break definition, described later in this section.

The break level numbers are not required to be consecutive. In this respect, control break numbers are not absolute; you could specify control breaks 2, 4, and 6, without specifying 1, 3, and 5. However, control break text assignments continue to be absolute; text for control break 2 is always associated with control break number 2—not with the second control break.

You may assign multiple columns to the same BREAKn value. When this occurs and Query Management/400 needs to determine control breaks, Query Management/400 considers all columns having the same control break level as a single concatenated column. This is particularly useful when LAST_NAME and FIRST_NAME are stored as separate columns. Likewise, it is needed when MONTH, DAY, and YEAR are each separate columns.

When using a control break, the data in the column should be ordered. For the data to be in order, the SELECT that produces the report must use ORDER BY.

There is no automatic reordering of columns due to break specifications, but break text is usually displayed to the left of any summary columns. Therefore, IBM recommends using the "Seq" value to display the break columns to the left, and the aggregated columns to the right, on the report.

BREAK2
BREAK2 is the usage value used to specify a column as the second level control break. A BREAK2 automatically generates whenever either the column(s) upon which it is defined changes, or when there is a BREAK1 generated. Note that a BREAK1 causes a BREAK2, but a BREAK2 does not cause a BREAK1.

BREAK3 through BREAK6
BREAK3 through BREAK6 name control columns for breaks at levels 3 through 6.

Indent

This field represents the relative location of the column within a row. Its units are the number of blank characters between the column and either:

- The right edge of the previous column
- The left edge of the screen or paper.

You may set Indent to n, where 0 < = n < = 999. In the default format, Query Management/400 initializes Indent to 2.

Width

This is the column's output width. It specifies how many character spaces to reserve for displaying the column heading and data. Names that are wider than Width are truncated. Query Management/400 defines the width field as numerics only. The maximum Width is 32767 single-byte characters. If the length of the value to display exceeds the width of the column, the value is either replaced with a row of asterisks (****) if it is numeric data, or truncated at the right if it is character data. The desired result may be obtained by changing Width and displaying the report again.

If you do not specify a width value, Query Management/400 provides a run-time default.

Datatype

This field represents the type of data that is contained in the corresponding column in the queried table. The Datatype options are the following:

CHARACTER
The data in the column in the table is character.

NUMERIC
The data in the column in the table is numeric. It can be binary, packed, zoned, or floating point.

DATE, TIME, TIMEST, GRAPHIC
These data types will be imported with a warning. Because these data types are not supported on the AS/400 sytem, an error will be issued when the importing form is applied to the queried DATA. This error results because the data type in the form does not match the data type of the column in the queried table.

Edit

Edit codes are used to format character and numeric data for display.

Below are the edit codes for **character** data.

C makes no change in the display of a value. If the value cannot fit onto one line in the column, Query Management/400 truncates the text according to the width of the column.

CW
makes no change in the display of a value, but if the value cannot fit on one line in the column, Query Management/400 wraps the text according to the width of the column. That is, instead of truncating the data at the end of the column, Query Management/400 puts as much data as possible on one line in the column and then continues the data on the next line in the column.

The CW edit code can be used on columns of mixed DBCS and single-byte character data.

CT makes no change in the display of a value. But if the value cannot fit onto one line in the column, Query Management/400 wraps the column according to the text in the column. That is, instead of truncating the data at the end of the column, Query Management/400 fits as much data as possible on a line, interrupts the line when it finds a blank and continues the data on the next line. If a string of data is too long to fit into the column and does not contain a blank, Query Management/400 wraps the data by width until the point where it finds a blank and can therefore continue wrapping by text.

The CT edit code can be used on columns of mixed DBCS and single-byte characters. Query Management/400 interrupts the line when it finds a single-byte or double-byte blank.

Below are the edit codes for **numeric** data.

E displays numbers in scientific notation. For example, the number -1234.56789 displays as -1.234E+03. As many digits as can display are placed in the report, up to a maximum of 15. One space is always reserved for a leading sign, although it does not display for positive numbers. There is always a sign and at least two digits after the E. Up to three digits will display.

D, I, J, K, L, and P
display numbers in decimal notation with different combinations of leading zeros, negative symbols, thousands separators, currency symbols, and percent signs. Examples are in the following table.

Each code may be followed by a number (from 0 to 31) that tells how many places to allow after the decimal point. If no number is specified, zero places after the decimal point are assumed. Numbers that have more decimal places than fit into the allowed space are rounded; numbers with fewer decimal places are padded with zeros.

How to read Figure 5-4 on page 5-8: This figure shows how the numeric edit codes format the number -1234567.885.

Edit Code	Lead Zeros	Negative Sign	Thousands Separators	Currency Symbol	Percent Sign	Display of -1234567.885
E	No	Yes	No	No	No	-1.23456789E+06
D2	No	Yes	Yes	Yes	No	-$1,234,567.89
I3	Yes	Yes	No	No	No	-0001234567.885
J2	Yes	No	No	No	No	000001234567.89
K3	No	Yes	Yes	No	No	-1,234,567.885
L2	No	Yes	No	No	No	-1234567.89
P2	No	Yes	Yes	No	Yes	-1,234,567.89%

Figure 5-4. Use of Edit Codes

If you do not specify an edit value, Query Management/400 provides a run-time default.

Seq

You can specify "Seq" to order the columns in the generated report.

The following rules apply to evaluating "Seq" values:

- Defaults to *n* for the *n*th column in the FORM.
- Any number from 1 to 999.
- Numbers need not be consecutive.
- Columns with the same "Seq" number appear in the report in the same order that they appear in the form.

Run-Time Defaults

Query Management/400 uses system-provided defaults for Datatype, Column Heading, Edit, and Width values when columns of data extracted by running a query need to be formatted for a report. This condition occurs when the following occur:

- You did not specify a form or you specified *SYSDFT to refer to the default form for the extracted data.
- The specified form does not contain the information needed to format the report or the form was imported with warnings about blank values or missing column table fields.

Datatype

Figure 5-5 shows data type defaults that are assumed from internal numeric identifiers returned with the data.

Figure 5-5. Values for Columns Datatype Field

Datatype Value	Datatype Number	Meaning
NUMERIC	496	Integer
	500	Small Integer
	484	Decimal
	480	Floating Point
CHAR	452	Fixed Character

Column Heading

You can establish column heading defaults for file data when you define a field to the system. Use field names unless other text is specified when you use interactive data definition utility (IDDU).

You can define files that do not cause column headings to be defaulted. Column heading defaults for calculated data and for file fields without established defaults are taken from the set of unique column names manufactured at run time for the selected columns. These are the column names that are used to create a new file for a SAVE DATA request.

To manufacture these names, Query Management/400 processes the selected columns in order, from first to last. The unique name for the *n*th selected column is created in the following way:

1. The column name from the file (table) definition (or SEL for a calculated field) is used as the created name if it does not match any previously created name.
2. If the matched name is too long to add to the next available number, that number is added to COL to create the name.
3. If the matched name is not too long to add to the next available number, that number is added to the column name to create the name.

Note: The first number added to a column name or COL to make the name unique is 1, the next number added is 2, and so on.

The following example shows the unique names created for a particular SELECT list.

```
SELECT 7*WEEKS, SALARY, SALARY, SALARY/7*WEEKS, MAXBENEFIT, MAXBENEFIT
         |       |       |          |              |          |
         |       |       |          |              |          |
        SEL    SALARY  SALARY1     SEL2        MAXBENEFIT     COL3
```

If some of the columns have column heading defaults that were previously defined, the column headings in a formatted report are not necessarily unique. This can be true for the *SYSDFT form as well as a form specified by name.

Edit

Editing defaults are intended to be the same as those used by other data-displaying products on the system, such as Query/400. Character data is unchanged or truncated (such as SAA edit code C). Scientific notation is used for floating data (such as SAA edit code E). Editing defaults usually have no corresponding SAA representation for numeric fields, and can include edit words, edit descriptions, and RPG edit codes.

File (table) data defaults are established when a field (column) is defined to the system. System-level values determine the editing for calculated data. These values come from a translatable message and are used for building a default edit description whenever one is needed.

Changeable system-level values can also change the editing applied to numeric data, for example, using the QDECFMT system value to change the editing applied for RPG edit code J.

Width

Width defaults are intended to provide room for everything that has to be shown in the column. For a particular column, the default is the maximum of the following:

- The length of the longest segment of the column heading
- The edited data width (after adjustment of the raw data length by 3 if SUM Usage aggregation values have to be shown in the column)
- A length of 9 if COUNT Usage aggregation values have to be shown in the column

Column names used as defaults for column headings are unique. When necessary to make the default names unique, Query Management/400 adds a number as a suffix to the end of the column name. When this happens, the first occurrence of the column name remains unchanged. The number is added to all other occurrences of the name. Numbers are assigned sequentially across all column names. For example:

```
SELECT ID, ID, DEPT, JOB, ID, DEPT
```

results in default heading of

```
ID  ID1  DEPT  JOB  ID2  DEPT3
```

When the default column names cannot be made unique, Query Management/400 assigns COL*n* for column names. For example:

```
SELECT ABCDEFGHIJKLMNOPQR, ABC, ABCDEFGHIJKLMNOPQR
```

results in default headings of

```
ABCDEFGHIJKLMNOPQR  ABC  COL1
```

PAGE Fields

The following fields are used to specify headings and footings on a report. Figure 5-6 shows the defaults and possible values for the attributes on the Page fields.

Figure 5-6. Default Values for Page Fields

Attribute	Default	Possible Values
Blank lines before heading	0	0 to 999
Blank lines before footing	2	0 to 999
Blank lines after heading	2	0 to 999
Blank lines after footing	0	0 to 999
Alignment on heading text	CENTER	LEFT, CENTER, RIGHT
Alignment on footing text	CENTER	LEFT, CENTER, RIGHT

Blank Lines Before Heading/Footing

These fields indicate the number of blank lines before the page heading or page footing and must be specified as numbers. An acceptable value is any number from zero to 999. The default value for the page heading is *zero (0)* and for the page footing is *two (2)*. A page eject always precedes the heading on each page. The "Blank lines before heading" field controls the number of blank lines between the heading and the top of the page. The "Blank lines before footing" field controls the number of blank lines between the report body and the first footing line.

Blank lines are included in the count of the number of lines printed on the page.

Blank Lines After Heading/Footing

These fields indicate the number of blank lines after the page heading or page footing. The fields are defined in Query Management/400 as numerics only. An acceptable value is any number from zero to 999. The default value for the page heading is *two (2)* and for the page footing is *zero (0)*. The "Blank lines after heading" field controls the number of blank lines between the last heading line and the report body. The "Blank lines after footing" controls:

- The number of blank lines between the last footing line and the end of the page
- The last footing line and the line containing the "Date and time" and/or "Page number."

Blank lines are included in the count of the number of lines printed on the page.

"Blank lines after footing" takes precedence over "Blank lines before footing"; on a report page that has extra space left after the body of the report, extra blank lines are inserted so the "Blank lines after footing" value is the correct number of lines.

Heading Text Lines

The heading text lines contain a maximum of five (5) heading text lines.

Line

The Line value denotes the line positioning of the heading text lines.

For example, if you include text lines for line numbers 1 and 5, the text is displayed in the report as:

```
Text for line 1
[blank line]
[blank line]
[blank line]
Text for line 5
```

If you specify text line number 1 and then repeat the specification later, the last occurrence of line number one prevails over the first occurrence.

Align

The Align field controls the positioning of the page heading text within the report line. Acceptable values are:

RIGHT Right-justify the text.
LEFT Left-justify the text.
CENTER Center the text.

The default value for headings is *center*.

Page Heading Text

These fields allow you to enter text that appears as the page heading in the report. You can enter a maximum of 55 characters for each text line.

If the text line *n* is the biggest line with nonblank text, then *n* indicates how many lines are to be formatted. This is true even though it could result in some of the formatted lines being completely blank.

Page headings may contain four types of special variables. All variables must be coded with a leading "&" to identify them within the heading text.

&n where n refers to the *n*th column defined in the FORM, &n is assigned the first value on the page for the specified column. &n is set at each page break.

&DATE where Query Management/400 replaces the value with the current date.

&TIME where Query Management/400 replaces the value with the current time.

&PAGE where Query Management/400 replaces the value with the current page number. The format of the page number is a four digit number ranging from 1 to 9999 with leading zeros suppressed. After 9999, the counter wraps to 0 and continues to increment for subsequent pages *without* leading zero suppression.

When the report prints, the page heading appears at the top of each page, formatted according to the format specification. The variable &n formats according to the edit code specification, except when column wrapping is specified. If column wrapping is specified for the column, it is ignored when the data formats into the text.

All variables are resolved when the report is created.

Footing Text Lines

The footing text contains a maximum of five (5) footing text lines.

Line

The Line value denotes the line positioning of the footing text lines.

For example, if you include text lines for line numbers 1 and 5, the text is displayed in the report as:

```
Text for line 1
[blank line]
[blank line]
[blank line]
Text for line 5
```

If you specify text line number 1 and then repeat the specification later, the last occurrence of line number one prevails over the first occurrence.

Align

The Align field controls the positioning of the page footing text within the report line. Acceptable values are:

RIGHT Right-justify the text.
LEFT Left-justify the text.
CENTER Center the text.

The default value for footings is *center*.

Page Footing Text

These fields allow you to enter text for the page footing that is to appear in the report. You can enter a maximum of 55 characters for each text line. You can use the variables described above for the heading text field. When the report formats, appropriate values are substituted for the variables. When the report prints, the page footing appears at the bottom of each page, formatted according to the format specification. The variable &n formats according to the edit code specification, except when column wrapping is specified. If you specify column wrapping, it is ignored when the data formats into the text. The formatted page footing appears once at the bottom of the displayed report.

If the text line *n* is the biggest line with nonblank text, then *n* indicates how many lines are to be formatted. This is true even though it could result in some of the formatted lines being completely blank.

FINAL TEXT Fields

The following fields are used to specify the final text that appears on a report. Figure 5-7 shows the defaults and possible values for the attributes on the Final text fields.

Figure 5-7. Default Values for Final Text Fields

Attribute	Default	Possible Values
New page	NO	YES, NO
Put final summary at line	1	1 to 999 or NONE
Blank lines before text	0	0 to 999 or BOTTOM
Alignment for final text	RIGHT	LEFT, CENTER, RIGHT

New Page for Final Text

This field indicates whether the subsequent part of the report (final text) must begin on a separate page when printed. The default is *no*. When you specify Yes for New Page, the final text formats on a new page.

Put Final Summary at Line

This field indicates whether the final summary should be in formatted form and where to vertically position it in the report final text. Acceptable values are the numbers one (1) to twelve (12), or the word NONE, where NONE indicates there is no presentation of final summary data. The default value is *one (1)*.

A value of one to 12 indicates the relative line number within the final text at which the summary data must format. This is strictly vertical placement. For horizontal placement in the line, the final summary is always formatted under the columns being summarized. If there are no column widths with aggregating usages, this value is ignored.

Blank Lines before Text

This field indicates the number of blank lines between the body of the report and the first line of final text. An acceptable value is any number from zero to 999. The default value is *zero*. You may also specify BOTTOM, which positions the final text at the bottom of the printed page. BOTTOM causes insertion of a number of blank lines, in order to position the final text immediately before the page footing text specification on the page. If there is not enough space for the final text on the current page, it is placed at the bottom of the next page.

Line

This field indicates the line positioning of the final text lines.

For example, if you include text lines for line numbers 1 and 5, the text is displayed in the report as:

```
Text for line 1
[blank line]
[blank line]
[blank line]
Text for line 5
```

If you specify text line number 1 and then repeat the specification later, the last occurrence of line number one prevails over the first occurrence.

Align

This field controls the positioning of the final text within the report line; it refers to alignment between the left margin and the first summary column. If the report does not contain final summary data, then the alignment refers to the entire width of the displayed or printed report. Acceptable values are:

RIGHT Right-justify the text.
LEFT Left-justify the text.
CENTER Center the text.

The default value is *right* for the final text. If there is no associated final text, the Align value is ignored.

Final Text Lines

There are twelve lines available for the final text. You specify what appears on these lines. A maximum of 55 characters can be entered for each text line. Only &n is allowed as a variable.

If the text line *n* is the biggest line with nonblank text, then *n* indicates how many lines are to be formatted. This is true even though it could result in some of the formatted lines being completely blank.

If the *Put final summary at line* value is *m* and $m > n$, then there are *m* final lines formatted in the report.

BREAK Fields

You can specify information for break levels one (1) to six (6). You also change or specify the exported FORM by selecting the proper FORM field numbers. See Chapter 6, "Exported Objects" on page 6-1. There are distinct field numbers for each of the break levels. You specify options for each break level in a similar manner. Each set of options is independent from the others.

Figure 5-8 shows the defaults and possible values for the attributes in the Break fields.

Figure 5-8. Default Values for Break Fields

Attribute	Default	Possible Values
New page for break	NO	YES, NO
New page for footing	NO	YES, NO
Repeat column heading	NO	YES, NO
Blank lines before heading	0	0 to 999
Blank lines before footing	0	0 to 999 or BOTTOM
Blank lines after heading	0	0 to 999
Blank lines after footing	1	0 to 999
Put break summary at line	1	1 to 999 or NONE
Alignment on break heading text	LEFT	LEFT, CENTER, RIGHT
Alignment on break footing text	RIGHT	LEFT, CENTER, RIGHT

New Page for Break/New Page for Footing

These fields indicate whether the subsequent part of the report begins on a new page. The default value is *no* for both. When you specify Yes for the New Page for Break field, the member lines for the break format on a new page. If you specify a break heading, it precedes the break member lines on the new page. When you specify Yes for the New Page for Footing field, the break footing formats on the next page (if a footing exists).

Repeat Column Heading

This field indicates whether the column headings should repeat above the member lines for a particular break level. *No* is the default value.

When paging or printing a report, the column headings always appear at the top of the screen or page. In addition to these headings, a set of headings appears at the start of the break if Yes is specified for Repeat Column Headings for that break. This happens regardless of whether there is any break heading text. However, if the break starts at the top of a printed page, only one set of column headings, the set preceding the break member line, formats.

Blank Lines before Heading/Footing

These fields indicate the number of blank lines that appear before the break heading or break footing. If no break heading is specified, then the value for this field is the number of blank lines before the break member lines. Acceptable values are any number from zero to 999. The default is *zero* for both the heading and the footing.

The Blank Lines Before Heading field may contain a number only.

For a break footing, you may also specify BOTTOM. Applicable only to a printed report, BOTTOM causes the break footing to position at the bottom of the current page on a printed report. BOTTOM causes insertion of blank lines in order to position the text immediately before the page footing text specification on the page. This also implies that a page eject occurs, since the next line must print on the next page.

Blank Lines after Heading/Footing

These fields indicate the number of blank lines after the break heading or break footing. If no break heading is specified, then the value of this field defaults to the number of blank lines after the break member lines. If no break footing is specified, then the blank footing is included in the blank lines after the break member lines. An acceptable value is any number from zero to 999. The default is *zero* for the heading and *one (1)* for the footing.

Put Break Summary at Line

This field indicates whether the break summary is to format and, if it does, where to place it relative to the lines of break footing text. The value can be from one (1) to five (5), or NONE, with NONE indicating that no break summary information is to display for the break. The default is *one (1)*. The number used corresponds to the number of the line of break footing text with which the break summary is to display.

This placement is strictly vertical. For horizontal placement in the line, the break summary always formats under the columns being summarized. If there are no column widths with aggregating usages, this value is ignored because there are no columns to summarize.

Break Heading Text Lines

The heading text lines field contains a maximum of five (5) heading text lines.

Line

The Line value denotes the line positioning of the heading text lines.

For example, if you include text lines for line numbers 1 and 5, the text is displayed in the report as:

```
Text for line 1
[blank line]
[blank line]
[blank line]
Text for line 5
```

If you specify text line number 1 and then repeat the specification later, the last occurrence of line number one prevails over the first occurrence.

Align

This field controls the positioning of the break heading text within the report line. Acceptable values are:

RIGHT Right-justify the text.
LEFT Left-justify the text.
CENTER Center the text.

The default value is *left*. The alignment is based on the entire width of the displayed or printed report.

Break Heading Text

Fifty-five (55) characters per line are allowed on break heading text. Only &n is allowed to be used as a variable.

If the text line *n* is the biggest line with nonblank text, then *n* indicates how many lines are to be formatted. This is true even though it could result in some of the formatted lines being completely blank.

Break Footing Text Lines

The footing text lines field contains a maximum of five footing text lines.

Line

The Line value denotes the line positioning of the footing text lines.

For example, if you include text lines for line numbers 1 and 5, the text is displayed in the report as:

```
Text for line 1
[blank line]
[blank line]
[blank line]
Text for line 5
```

If you specify text line number 1 and then repeat the specification later, the last occurrence of line number one prevails over the first occurrence.

Align

This field controls the positioning of the page footing text within the report line. Acceptable values are:

RIGHT Right-justify the text.
LEFT Left-justify the text.
CENTER Center the text.

The default value is *right*. The alignment refers to the space between the first character position on the left and the first summary column. If the report does not contain break summary data, the alignment refers to the entire width of the displayed or printed report.

Break Footing Text

Only &n is allowed as a variable. You can use up to 55 characters per line.

If the text line *n* is the biggest line with nonblank text, then *n* indicates how many lines are to be formatted. This is true even though it could result in some of the formatted lines being completely blank.

If the *Put break summary at line* value is *m* and *m* > *n*, then there are *m* break lines formatted in the report.

OPTIONS Fields

The Options fields allow you to specify various report formatting options.

Figure 5-9 shows the defaults and possible values for the attributes on the Options fields.

Figure 5-9. Default Values for Options Fields

Attribute	Default	Possible Values
Detail line spacing	1	1 to 4
Outlining for break columns	YES	YES, NO
Default break text	YES	YES, NO
Column-wrapped lines kept on page	YES	YES, NO
Column heading separators	YES	YES, NO
Break summary separators	YES	YES, NO
Final summary separators	YES	YES, NO

Detail Line Spacing

This field indicates the spacing you request between each detail line in the report. Numbers are the only valid input for this field. Acceptable values are 1, 2, 3, or 4, where 1 is single spacing, 2 is double spacing, and so on. The default value is *1*.

Outlining for Break Columns

If you assign a usage code for a break to one of the columns, then this field is used to determine when the value in the break column displays in the report. *Yes* is the default and displays the value in the break column only when the value changes. A No in this field displays the value in the break column on every detail line in the report.

Default Break Text

This field indicates whether you are requesting inclusion of the default break text in the report. The default value is *yes*. Use default break text to mark the break aggregation line. The break aggregation line is one or more asterisks for each break level; one asterisk for the highest numbered break level text, two asterisks for the next highest numbered break level text, and so on. For example, if the report has two control breaks, BREAK2 and BREAK4, Query Management/400 generates one asterisk to mark the level-4 break and two asterisks to mark the level-2 break.

Column Wrapped Lines Kept on a Page

If you specified column wrapping for one or more columns in the report, this field is used to determine whether the wrapped columns can be split between two pages. The default for this field is *yes*. It prevents splitting wrapped columns between two pages (unless the wrapped column is longer than the page depth). A No in this field allows splitting of wrapped columns between pages.

Column Heading Separators

This field indicates whether the column heading separators (dash lines) appear in the report. The default value is *yes*.

Break Summary Separators

This field indicates whether the break summary separators (dash lines) appear in the report. The default value is *yes*. A blank separator line is generated if there are no summary columns and the value specified is yes.

Final Summary Separators

This field indicates whether the final summary separators (equal signs) appear in the report. The default value is *yes*. A blank separator line is generated if there are no summary columns and the value specified is yes.

Chapter 6. Exported Objects

You can export certain Query Management/400 objects to manipulate them with an editor or an application or to transport them from one environment to another. The objects can be imported by the same or another query product. The following sections describe the formats for exporting a query, procedure, and form object.

General Object Formats

This section presents Query Management/400 formats that are common across a number of systems. Other formats, specific to each Query Management/400 object, are discussed in later sections.

Comments in Externalized Query Management/400 Objects

An object comment becomes an externalized object when it is displayed either online or on printed copy. An object comment is allowed in an externalized Query Management/400 form, query, or procedure object. The comment must be specified as a V record with a field number of 1001 and a maximum length of 50. A comment longer than 50 characters will be truncated. The comment is generated in the externalized object when it is exported. The comment record, if present, must immediately follow the H record. The H record type field must be:

- F for a form
- Q for a query
- P for a procedure.

Follow this sequence when *importing* to determine which text description is used on the Query Management/400 object:

- If the COMMENT= option is specified, the value for this option will be used.
- If member text is in the member, the member text will be used.
- If a comment is in the object, this comment will be used.
- If no comment exists, the text description will be blank.

Follow this sequence when *exporting* to determine which text description is written to the externalized Query Management/400 object:

- If the COMMENT= option is specified, the value on this option will be used.
- If the COMMENT= option is not specified, the text description in the Query Management/400 object will be used.

The H and V records are the only parts of the encoded format that are allowed for query or procedure objects. Attempting to use other record types such as T, R, E, and * will generate unpredictable results.

Comments may not span multiple lines and may not be used inside other comments. A comment begins with '/*' and ends with '*/'.

The procedures have object and user comments. An object comment in a procedure must be inside comment delimiters with no intervening blanks between the start comment and the record identifier. User comments are ignored at run time.

The following is an example of a procedure with both an object comment and some user comments:

```
/*H QM4 01 P 01 E V W E R 01 03 90/07/24 13:30 */
/*V 1001 016 SALES PROCEDURE */
'IMPORT QUERY SALES FROM QRYSRC'
'RUN QUERY SALES'  /*Total sales query*/
'PRINT REPORT'  /*Management report*/
```

The following is an example of an object comment in a query:

```
H QM4 01 Q 01 E V W E R 01 03 90/06/38 01:25
V 1001 011 SALES QUERY
SELECT SALARY FROM SALESFILE WHERE
SALARY > 50000
```

External Formats

The summary below explains the formats for objects. Respective formats for these objects are discussed in later sections.

Procedures	Panel format
SQL Queries	Panel format
Form	Encoded format

Panel Format

This format consists of a number of fixed-length records containing the object as a series of text strings. The object is not formatted in any special way within the records. It remains as composed by your application or as entered from your terminal.

Objects written out in this format have the following attributes:

- Logical record length of 79 bytes
- Fixed length record format.

Encoded Format

This external format allows you to access those Query Management/400 objects that contain more structure than the simple panel format objects. Use this format only for FORM objects.

Size of the Encoded Format

The encoded format must have record lengths of up to 150 characters.

Records that Make Up the Base Encoded Format

The formats of those records that make up the base encoded format are described below. Other encoded format record types (associated with specific Query Management/400 objects) are discussed in later sections that deal with the external formats of the individual objects.

For each record type there is a description of its purpose, actual contents, format, and a set of notes on its usage. There are also record descriptions that provide a precise and exhaustive list of the possible values for each field in the record. Some of these fields (particularly in the header record) may contain only a single value. This is indeed intentional and often signifies that other values will be allowable for the field in future releases of Query Management/400.

The base set of record types in the encoded format includes:

Record Type	Descriptive Name
"H"	"Header ("H") Record" on page 6-3
"V"	"Value ("V") Records" on page 6-7
"T"	"Table Description ("T") Records" on page 6-9
"R"	"Table Row ("R") Records" on page 6-11
"E"	" End-of-Object ("E") Record" on page 6-13
"*"	"Application Data ("*") Record" on page 6-14

Applications should be written to tolerate fields that are not defined here; that is, applications should ignore fields and records that are not defined or understood. Toleration of such fields will allow an application to run with an object generated by a future implementation of Query Management/400. This design would also allow toleration of an object containing non-SAA functions.

Header ("H") Record

The H record identifies the contents of an externalized object (an object is an externalized object when it is being displayed either online or on hard copy). It contains information describing the characteristics of the object as well as the file format.

The following figures summarize the contents of the header record:

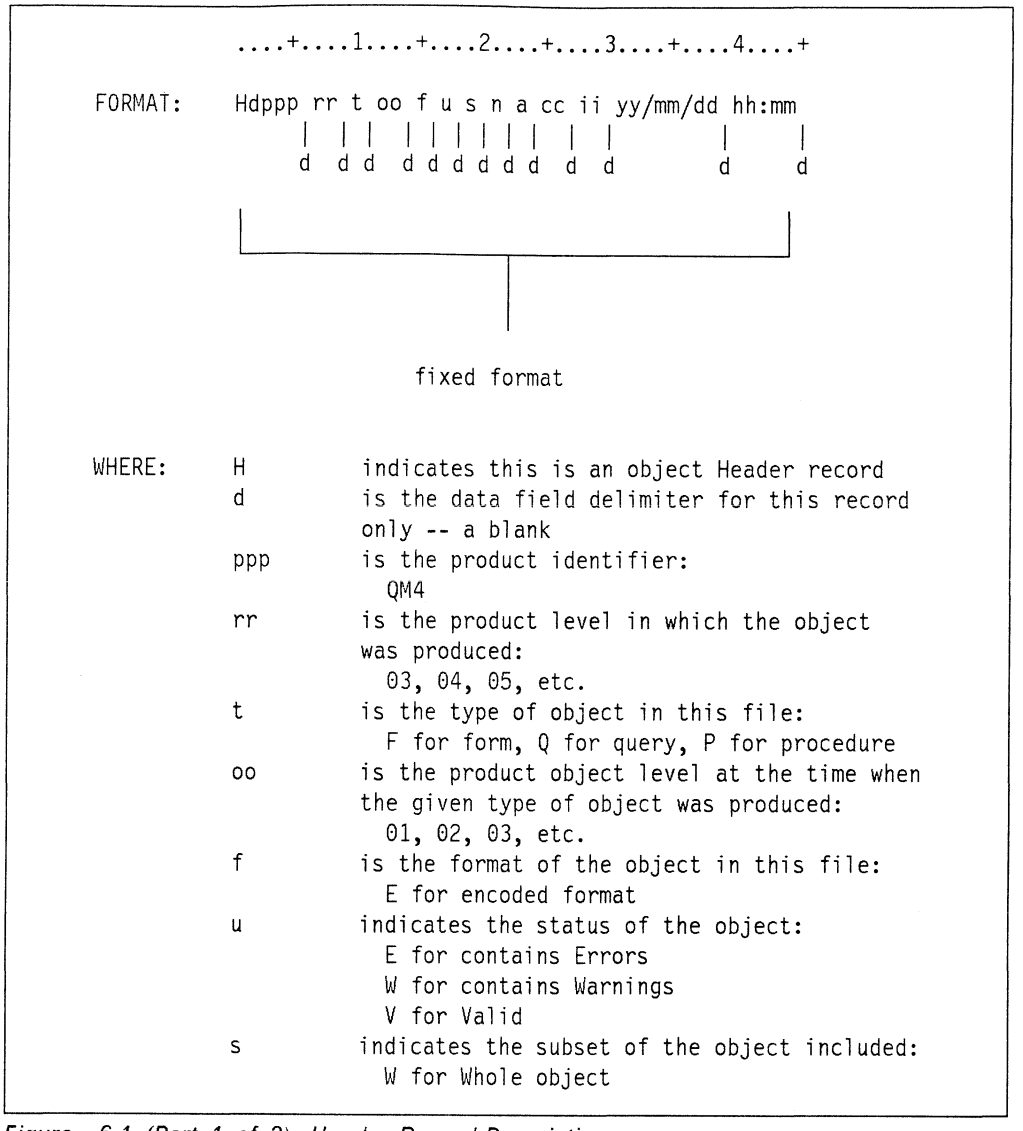


Figure 6-1 (Part 1 of 2). Header Record Description

```
              n          indicates the language of the exported
                         object.
                              E          English
              a          is the action against the item:
                           R for Replace object
              cc         is the length of the control area in the
                         beginning of each following record (including
                         the 1-byte record type):
                           01 for Forms
              ii         is the length of the integer length fields
                         specified in "V" and "T" type records:
                           03 for all objects
              yy/mm/dd   date stamp
              hh:mm      time stamp

    EXAMPLE:  H QM4 03 F 03 E V W E R 01 03 89/09/23 15:21

              (QM4 FORM file at "object level" 3, written in the
               encoded format, with no errors or warnings, in entirety
               in English, usable for complete replacement, with 1
               byte of control area, and 3 bytes for integer lengths)
```

Figure 6-1 (Part 2 of 2). Header Record Description

The following figure summarizes the location and contents of each of the fields in the header record. The field names used here were defined in the previous figure describing the entire header record. Object-specific values are noted as appropriate.

Field	Columns	Possible Values	Required on Input	Default if Blank on Input
H	01	H	yes	
d	02	(blank)	no	(not used on input)
ppp	03–05	QM4	yes	
rr	07–08	03	no	(not used on input)
t	10	F (form) Q (query) P (procedure)	yes	
oo	12–13	03	no	current object level
f	15	E	yes	
u	17	E W V	no	(not used on input)
s	19	W	no	(not used on input)
n	21	E (English)	no	(not used on input)
a	23	R	yes	
cc	25–26	01 (form)	no	(not used on input)
ii	28–29	03	no	(not used on input)
yy/mm/dd	31–38	(dates)	no	(not used on input)
hh:mm	40–44	(times)	no	(not used on input)

Figure 6-2. Header Record Fields

Notes to Figure 6-2:

- The header record must be the first record in the external file.
- If the record is shorter than its fixed format length, those fields (or portions of fields) left unspecified are assumed to be blank.
- The object level ("oo") is used to denote a change in the externalized format of an object. When a particular level of Query Management/400 changes the external format of an object, the object's level number is incremented. In this way, an application can use this number to determine the format of the object's records.
- The control area (with length "cc") is a fixed area in the beginning of each of the encoded format records (except the header record) that contains control information pertaining to the given record; the control area contains information such as the record type and a record continuation indicator.

- The subset, format, action, control area length, and integer length fields have been included in the header record for future extensions to the encoded format.
- In the future, additional fields will be added to the end of the header record.

Value ("V") Records

The V record is used to provide a value for a single nontabular field in an object (like a FORM OPTIONS field). It includes the unique field number, the field's value, and its length.

The following figure summarizes the contents of the V record:

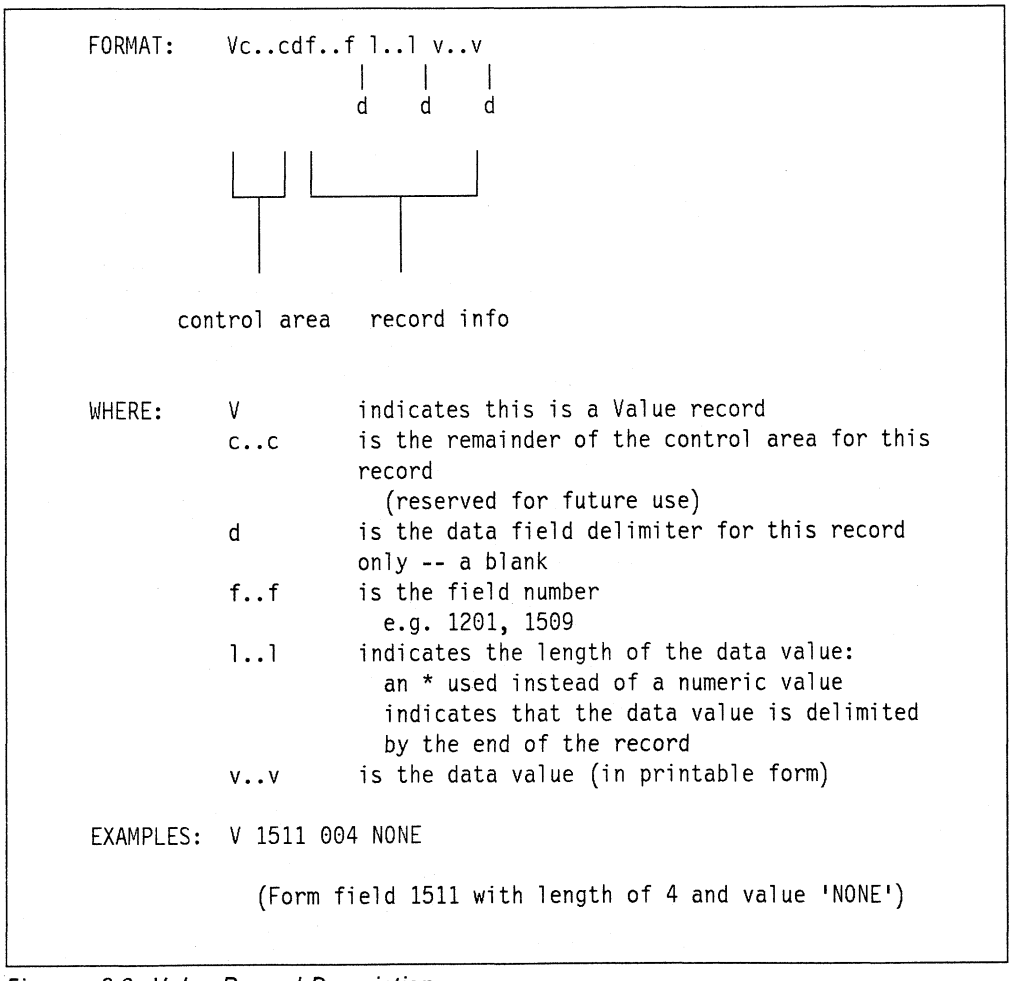


Figure 6-3. Value Record Description

The following figure summarizes the location and contents of each of the fields in the V record. The field names used here were defined in the previous figure describing the entire V record. Object-specific values are noted as appropriate.

Control Area				
Field	Columns	Possible Values	Required on Input	Default if Blank on Input
V	01	V	yes	
c..c	02	(x)	n/a	
x	02	(blank)	n/a	

Remainder of Record				
Field	Offsets past Control Area	Possible Values	Required on Input	Default if Blank on Input
d	+01	(blank)	no	
f..f	+02–05	1001–9999	yes	
l..l	+07–09	* 000–999	yes	
v..v	+11–end	(data)	no	(blank)

Figure 6-4. Value Record Fields

Notes to Figure 6-4:

- An omitted data value (like end-of-record), or blanks only following the "*" implicitly indicate that a null (the same as a blank) value is to be applied to this field.
- To explicitly set a field to blank, the field must have a specified positive length and a blank data value.
- Fields are set to their default values when the object is updated if
  - The specified length is zero, or
  - No length is specified.
- Query Management/400 issues a warning when it finds a field length of zero to indicate that the default value is set for this field.
- If the specified length is shorter than the supplied data value, Query Management/400 uses the specified length and issues a warning message.
- If the specified length is longer than the supplied data value, Query Management/400 sets the data value without extending beyond the end of the record and issues a warning message.
- IBM is retaining the length field for future V record uses in which an explicit length may be specified (for example, to indicate significant blanks), and for the possibility of V record expansion.

Table Description ("T") Records

The T record is used to describe the content and format of the table of values that follows. The contents of a T record determine the contents of all R (row) records for this table. A T record indicates which table is being described (by its unique table number), which columns are included (by their unique field numbers), in what order they appear, and the lengths of the values in these columns.

The following figure summarizes the contents of the T record:

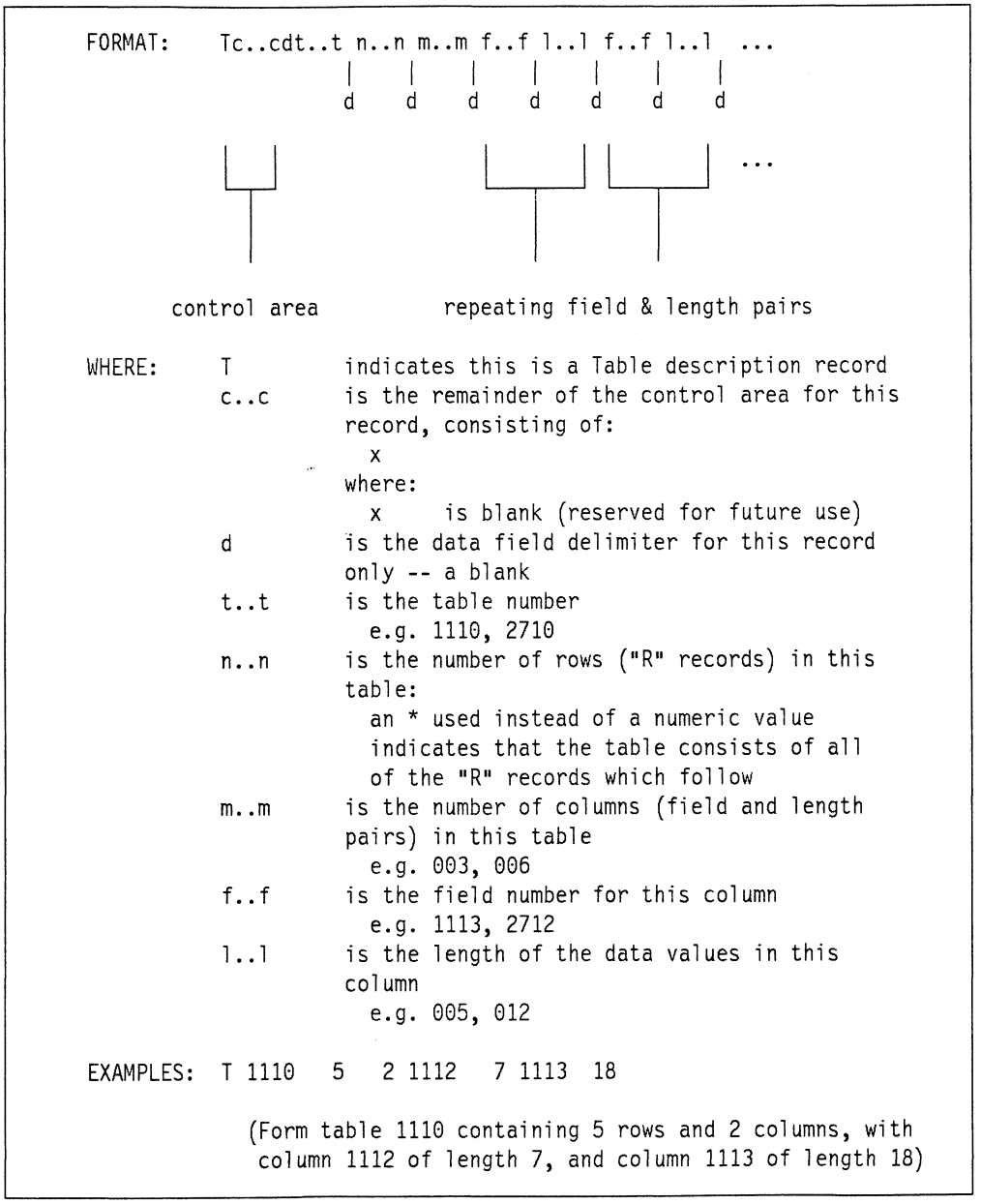


Figure 6-5. Table Record Description

The following figure summarizes the location and contents of each of the fields in the T record. The field names used here were defined in the previous figure describing the entire T record. Object-specific values are noted as appropriate.

Control Area

Field	Columns	Possible Values	Required on Input	Default if Blank on Input
T	01	T	yes	
c..c	02	(x)	n/a	
x	02	(blank)	n/a	

Remainder of Record

Field	Offsets past Control Area	Possible Values	Required on Input	Default if Blank on Input
d	+01	(blank)	no	
t..t	+02–05	1001–9999	yes	
n..n	+07–09	* 000–999	yes	
m..m	+11–13	000–999	yes	
f..f	+15–18 +24–27	1001–9999	yes	
l..l	+20–22 +29–31 etc.	000–999	yes	

Figure 6-6. Table Record Fields

Notes to Figure 6-6:

- If the number of R records following the T record does not exactly match the numeric row count specified in the T record, an error condition results.
- The number of f..f / l..l pairs is limited to the number of columns in the given table.
- The number of columns should agree with the following number of column field numbers and lengths. If not, a warning message is issued and the number of columns used is the number of field numbers/column data value lengths in the T record.
- The order of f..f/l..l pairs is arbitrary.
- All of the R records immediately following a T record (that is, those associated with a single table) must contain values of the exact lengths specified for each column in the T record. Records shorter than the implied length result in blank or blank-padded values.
- Query Management/400 sets columns with a length of zero (or not specified) to their default values when the object is updated.

- Query Management/400 issues a warning message for columns with a specified length of zero to indicate that it set the default value for this column.
- A table with zero rows in it (or not included in the file) has the same effect as applying columns of length zero to the table; Query Management/400 sets all of the columns to their default values.
- To set a column field to blank, the column must have a positive length in the T record and a blank value in the R record.

Table Row ("R") Records

The R record is used to provide a set of values for a single row in the current table. It consists of an ordered list of values as described by the associated T record. An R record must exactly match the description of the positions and lengths of the data values, as specified in the T record.

The following figure summarizes the contents of the R record:

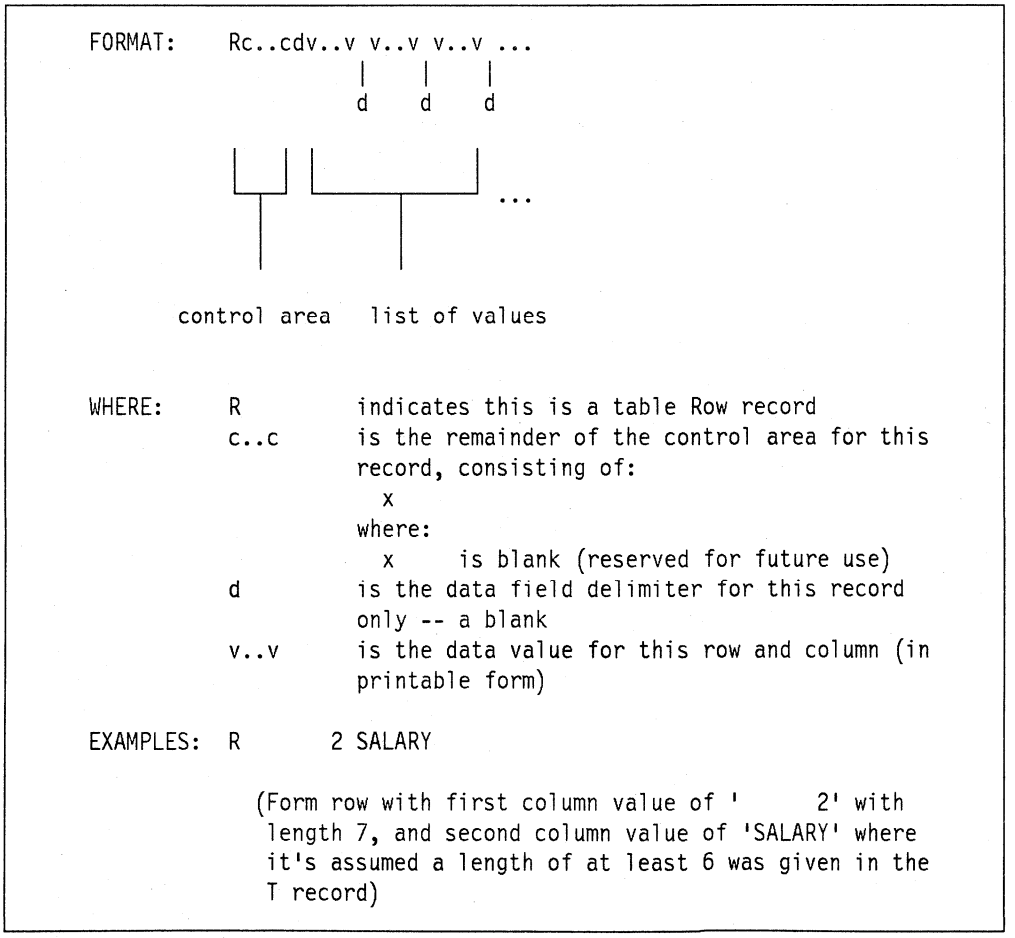


Figure 6-7. Row Record Description

The following figure summarizes the location and contents of each of the fields in the R record. The field names used here were defined in the previous figure describing the entire R record. Object-specific values are noted as appropriate.

Control Area				
Field	Columns	Possible Values	Required on Input	Default if Blank on Input
R	01	R	yes	
c..c	02	(x)	n/a	
x	02	(blank)	n/a	

Remainder of Record				
Field	Offsets past Control Area	Possible Values	Required on Input	Default if Blank on Input
d	+01	(blank)	no	
v..v	+02–xx +(xx+2)–yy +(yy+2)–zz etc.	(data)	no	(blank)

Figure 6-8. Row Record Fields

Notes to Figure 6-8:

- An R record must immediately follow another R record, or a T record.
- The number of v..v values must exactly match the description in the associated T record.
- A data value length of zero in the associated T record indicates that no value is to be applied to this row and column of the object; it is set to its default value. However, the presence of the field in the T record requires that the R record contain an extra delimiter for this field; a zero-length value results in one delimiter followed by another in the R record.

End-of-Object ("E") Record

The E record is used to delimit the end of the object.

The following figure summarizes the contents of the E record:

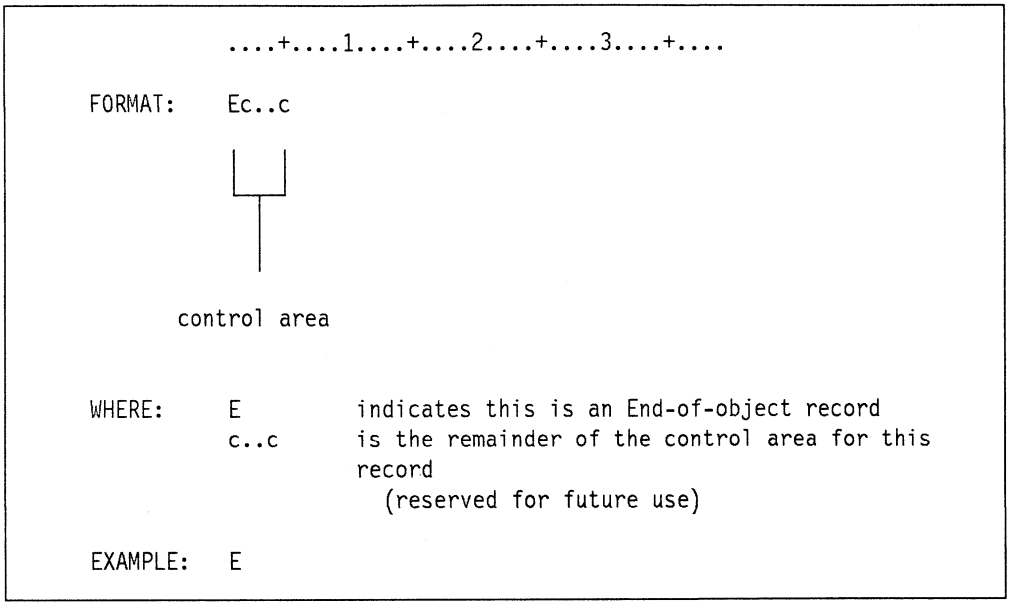


Figure 6-9. End-of-Object Record Description

The following figure summarizes the location and contents of each of the fields in the E record. The field names used here were defined in the previous figure describing the entire E record. Object-specific values are noted as appropriate.

Control Area				
Field	Columns	Possible Values	Required on Input	Default if Blank on Input
E	01	E	yes	
c..c	02	(x)	n/a	
x	02	(blank)	n/a	

Figure 6-10. End-of-Object Record Fields

Notes to Figure 6-10:

- The E record should be the last record in the external file.
- If the record is shorter than its fixed format length, those fields (or portions of fields) left unspecified are assumed to be blank.

Application Data ("*") Record

The application data record allows you to include your own data associated with the given object in the external file. You may choose to use these as comment records to further describe the object in the file.

The following figure summarizes the contents of the application data record:

```
FORMAT:    *v..v

WHERE:     *        indicates this is an application data record
           v..v     is the data value(s) produced by an appli-
                    cation program (preferably in printable form)

EXAMPLE:   * This is the Form that groups by DEPT.

             (comment record in a Form file)
```

Figure 6-11. Application Data Record Description

The following figure summarizes the location and contents of each of the fields in the * record. The field names used here were defined in the previous figure describing the entire * record. Object-specific values are noted as appropriate.

Control Area				
Field	Columns	Possible Values	Required on Input	Default if Blank on Input
*	01	*	yes	
v..v	02–end	(data)	no	(not used on input)

Figure 6-12. Application Data Record Fields

Notes to Figure 6-12:

- Application data records may appear anywhere in the external file, except ahead of the header record.
- Other than validating the format of the record, the application data record is ignored and has no effect on the input process.
- If the record is shorter than its fixed format length, those fields (or portions of fields) left unspecified are assumed to be blank.

Record Format Rules

For Input (Import):

1. The file may consist of variable or fixed-length records.
2. The minimum allowed logical record length is 23, based upon the required header record format and contents.
3. The record type character (H, V, T, R, E, and *) must appear in the first position of every record.
4. The first "cc" bytes of all records are reserved for control information and therefore require a fixed format ("cc" varies with the object).
5. Every data object (including field numbers, lengths, and values) must be preceded and followed by precisely one delimiter character.

   There are two exceptions to this rule: 1) End-of-record counts as a delimiter, and 2) Zero-length (null) values require a pair of delimiters surrounding the "nonexistent" value.
6. All the fields that are required on input are validated.
7. Duplicate occurrences of any single data value or table override any previous settings, with one exception. Query Management/400 does not override previous settings if the new object violates the rules established for a particular object, such as when the number of columns provided for a form may not be varied after the first column's table has been processed.
8. Portions of an object not included in the input file are set to their default values.
9. Numeric lengths and table and field numbers may be specified with leading zeros and/or leading blanks, but may not be padded with trailing blanks (other than a single blank delimiter); they must be right justified in their positions in the record.
10. Nonnumeric lengths (an * specification) must be padded with trailing blanks to the current length of integer length values (specified in the header record).
11. Object field values shorter than the data entry field on the object panel are padded with trailing blanks.
12. Object field values longer than the data entry field on the object panel are truncated.
13. If the format of the FORM file is in error, the IMPORT is aborted. In most cases, a single message describes the error and its location in the file.
14. Any records in a file following the E record are ignored.
15. If the input file does not contain an E record, it is assumed that end-of-file implies the end of the object.

For Output (Export):

1. All table and field numbers appear as 4-digit numbers.
2. All lengths are written with leading zeros to a length of 3 digits (as specified in the header record).
3. The blank character is the data object delimiter used in all records.
4. Delimiter characters do not appear as the final character on each record.
5. All reserved fields appear with blanks.
6. All table columns appear in numeric field number order in the externalized form, except the column heading field, which is last.
7. An E record appears as the last record in the target file.

Specific Query Object Formats

The following sections list the Query Management/400 objects and examples of the externalized formats.

Externalized FORM Format

The FORM is externalized by Query Management/400 in the encoded format. The specification of this format was described earlier in "General Object Formats" on page 6-1.

The FORM objects make use of the H, V, T, R, and E records discussed in "Encoded Format" on page 6-2.

Deviations from the base encoded format and other special considerations specific to the form are described below.

- The input FORM must contain all of the columns for the underlying data.
- The COLUMNS table must be the first portion of the FORM specified following the header or V record (and may not be altered in size by later duplicate occurrences of this table).
- The COLUMNS table must include a numeric count of the number of rows in the TABLE (rather than the "*" row count specification).
- If the entire COLUMNS table was read in, those fields not specified are set to their default values.
- The COLUMNS.DATA_TYPE column is always written out.

Figure 6-13 on page 6-17 shows how the fields described for a FORM object are represented in an externalized form that is used to create a Query Management/400 form (QMFORM) object. You must specify the possible field values, except for text fields, in uppercase characters. The defaults shown in Figure 6-13 are applied at import time, and appear in the exported form source. For any given text table:

- The 0 in the count range shown indicates the table does not need to be present in the form source when importing. If not present when importing, it will not be present in the exported form source.
- The high number in the count range shown is the highest number that can be used as a *Line* value. The lowest number is 1.

- Text line numbers do not have to be encountered in sequence to import, but are shown in sequence in the exported form source.
- Numbered lines with blank text will be added to fill any gaps before the highest numbered line imported with nonblank text.

Record type	Table number	Field number	Description	Count range	Import default
V		1001	Object Comment		
			**********		
			* COLUMN *		
			**********		
T	1110		Column Fields	1-255	
		1112	--Column data type		-
		1113	--Column heading		-
		1114	--Column usage		-
		1115	--Column indent		2
		1116	--Column width		-
		1117	--Column edit		-
		1118	--Column sequence		n
			(n is 5 for the 5th R record, for example)		
			**********		
			* PAGE *		
			**********		
V		1201	Blank lines before heading		0
V		1202	Blank lines after heading		2
T	1210		Page heading text table	0-999	
		1212	--Page heading line number		-
		1213	--Page heading align		CENTER
		1214	--Page heading text		-
V		1301	Blank line before footing		
V		1302	Blank line after footing		
T	1310		Page footing text table	0-999	
		1312	--Page footing line number		-
		1313	--Page footing align		CENTER
		1314	--Page footing text		-
			**********		
			* FINAL *		
			**********		
V		1401	New page for final text		NO
V		1402	Put final summary at line		1
V		1403	Skip lines before final text		1
T	1410		Final text table	0-999	
		1412	--Final text line number		-
		1413	--Final text align		RIGHT
		1414	--Final text		-

Figure 6-13 (Part 1 of 2). Encoded (Externalized) FORM Field Summary

```
                        ************************************************
                        *                    OPTIONS                   *
                        ************************************************
V           1501   Detail line spacing                          1
V           1502   Outlining for break columns                 YES
V           1503   Default break text                          YES
V           1505   Column wrapped lines kept on page           YES
V           1507   Column heading separators                   YES
V           1508   Break summary separators                    YES
V           1510   Final summary separators                    YES

                        ************************************************
                        *                     BREAK                    *
                        ************************************************
V           3080   Break level indicator                       -
V           3101   New page for break heading                  NO
V           3102   Repeat column headings                      NO
V           3103   Blank lines before heading                  0
V           3104   Blank lines after heading                   0
T    3110           Break heading table              0-5
V           3112     --Break heading line number               -
V           3113     --Break heading align                     LEFT
V           3114     --Break heading text                      -
V           3201   New page for break footing                  NO
V           3202   Put break summary at line                   1
V           3203   Blank lines before footing                  0
V           3204   Blank lines after footing                   1
T    3210           Break footing table              0-5
V           3212     --Break footing line number               -
V           3213     --Break footing align                     RIGHT
V           3214     --Break footing text                      -
```

Figure 6-13 (Part 2 of 2). Encoded (Externalized) FORM Field Summary

Figure 6-14 on page 6-19 is an example of an exported Query Management/400 form. A form may be edited and subsequently imported to obtain the desired report formatting. Refer to the list of record types and field numbers to match the field numbers to specific report attributes. The * records, which are valid comment records, are used to explain the meaning of certain parts of the form. The examples given are of certain T records with R records (tables) and V records. The same interpretation applies to records of the same types in the remainder of the form.

```
H QM4 01 F 01 E V W E R 01 03 90/3/19 14:27
* The 'H' record must be the first record in the file as above.
* The columns table must immediately follow the 'H' record unless there
*  are comment records.
* The T record describes the information that follows in the R records
* The field number '1110' identifies the table as the columns table.
* |    The '005' means that there are 5 columns (R records)
* |    |  following the 'T' record.
* |    |   The '006' means that there are 6 field number, field
* |    |   |   pairs in the T record.
* |    |   |   Starting with '1112' are the field number, field
* |    |   |   pairs which describe the values in the R record.
* |    |   |   For example '1112' corresponds to 'Data type'
* |    |   |   |   (see the table of field numbers) and has a
* |    |   |   |   length of 8.
* |    |   |   |   If I wanted to change the indent for a column
* |    |   |   |     I would look in the table of field numbers
* |    |   |   |     and find that the indent identifier is '1115'.
* |    |   |   |                 Counting the field numbers, starting
* |    |   |   |                 |    with '1112', I find that '1115'
* |    |   |   |                 |    is 3rd in the series of field
* |    |   |   |                 |    field length pairs and has a
* |    |   |   |                 |       field width of 6.
* |__  |__ |__ |_____|           |_____|
T 1110 005 006 1112 008 1114 007 1115 006 1116 005 1117 005 1113 040
*                 The indent value is the third field over in the
*                 |    R records and contains a 3 for every column.
*                 |    The values are left-justified and separated
*                 |_   by blank delimiters.
*                 |_|
R CHAR             3      6     C     NAME
R CHAR     BREAK1  3      6     C     DEPARTMENT
R NUMERIC  SUM     3      6     L     YEARS
R NUMERIC  SUM     3      6     L     SALARY
R NUMERIC  SUM     3      6     L     COMMISSION
* A 'V' record describes a single attribute in the form.
* The '1201' field number is the blank lines before page
* |  heading attribute.
* |    It has a value length of 1.
* |    |  It has a value of 1.
* |    |  |
* |__  |_ |_|
V 1201 001 1
V 1202 001 2
```

Figure 6-14 (Part 1 of 4). Sample Externalized Form

```
* The following table describes the page heading text.
* The fields used are the line number, alignment, and text, respectively.
T 1210 005 003 1212 004 1213 006 1214 055
R 1    CENTER ********************************************
R 2    CENTER ****                                        ****
R 3    CENTER ****           COMPANY REPORT               ****
R 4    CENTER ****                                        ****
R 5    CENTER ********************************************
*     The '*' in place of a numeric length indicates to use the
*     |  remainder of the record for the length of the data value.
*     |
*     |¯|
V 1301 *   1
V 1302 *   2
* Page footing text
T 1310 003 003 1312 004 1313 006 1314 055
R 1    CENTER XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
R 2    CENTER XXXXXXXX Internal Use Only XXXXXXXX
R 3    CENTER XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
V 1401 003 YES
V 1402 001 2
T 1410 005 003 1412 004 1413 006 1414 055
R 1    LEFT   ************************************************
R 2    LEFT   *******                                 ********
R 3    LEFT   *******          END OF REPORT          ********
R 4    LEFT   *******                                 ********
R 5    LEFT   ************************************************
V 1501 *   1
V 1502 003 YES
V 1503 003 YES
V 1505 003 YES
V 1507 003 YES
V 1508 003 YES
V 1510 003 YES
* The following section shows the break information using the
* new format.
*          The value in the '3080' V record indicates the break
*          |  level, which applies to all of the break information
*          |  until the next '3080' V record is encountered.
*        |¯|  The break level in this example is 1.
V 3080 001 1
V 3101 002 NO
V 3102 002 NO
V 3103 001 0
V 3104 001 0
T 3110 001 003 3112 004 3113 006 3114 055
R  1   CENTER BREAK 1 HEADING
V 3201 002 NO
V 3202 002 NO
V 3203 001 0
V 3204 001 0
T 3210 003 003 3212 004 3213 006 3214 055
```

Figure 6-14 (Part 2 of 4). Sample Externalized Form

```
R  1    CENTER ****************************
R  2    CENTER **** BREAK 1 FOOTING    ****
R  3    CENTER ****************************
* Break level 2 information
V 3080 001 2
V 3101 003 YES
V 3102 003 YES
V 3103 001 0
V 3104 001 0
T 3110 002 003 3112 004 3113 006 3114 055
R 1    CENTER BREAK 2 HEADING
R 2    CENTER ---------------
V 3201 002 NO
V 3202 002 NO
V 3203 001 0
V 3204 001 0
T 3210 003 003 3212 004 3213 006 3214 055
R 1    CENTER ****************************
R 2    CENTER **** BREAK 2 FOOTING    ****
R 3    CENTER ****************************
* Break level 3 information
V 3080 001 3
V 3101 003 YES
V 3102 003 YES
V 3103 001 0
V 3104 001 0
T 3110 002 003 3112 004 3113 006 3114 055
R 1    CENTER BREAK 3 HEADING
R 2    CENTER ---------------
V 3201 002 NO
V 3202 002 NO
V 3203 001 0
V 3204 001 0
T 3210 003 003 3212 004 3213 006 3214 055
R 1    CENTER ****************************
R 2    CENTER **** BREAK 3 FOOTING    ****
R 3    CENTER ****************************
* Break level 4 information
V 3080 001 4
V 3101 003 YES
V 3102 003 YES
V 3103 001 0
V 3104 001 0
V 3201 002 NO
V 3202 002 NO
V 3203 001 0
V 3204 001 0
T 3210 003 003 3212 004 3213 006 3214 055
```

Figure 6-14 (Part 3 of 4). Sample Externalized Form

```
R 1    CENTER ****************************
R 2    CENTER **** BREAK 4 FOOTING    ****
R 3    CENTER ****************************
* Break level 5 information
V 3080 001 5
V 3101 003 YES
V 3102 003 YES
V 3103 001 0
V 3104 001 0
V 3201 002 NO
V 3202 002 NO
V 3203 001 0
V 3204 001 0
* Break level 6 information
V 3080 001 6
V 3101 003 YES
V 3102 003 YES
V 3103 001 0
V 3104 001 0
V 3201 002 NO
V 3202 002 NO
V 3203 001 0
V 3204 001 0
T 3210 003 003 3212 004 3213 006 3214 055
R 1    CENTER ++++++++++++++++++++++++
R 2    CENTER +++ BREAK 6 FOOTING +++
R 3    CENTER ++++++++++++++++++++++++
* The 'E' record is the last record in the file. Records after
* the 'E' record will be ignored.
E
```

Figure 6-14 (Part 4 of 4). Sample Externalized Form

You can edit an externalized form object to change your report format. The externalized form layout is in the encoded format, which uses record types and field number identifiers to represent the form. Each field number identifier in the externalized form object represents a different attribute in the report. After making changes to the externalized form, you must import it for the changes to take effect.

Figure 6-15 on page 6-23 shows the descriptive names of the encoded form fields.

Record Type	Table Number	Field Number	Description
V		1001	Object comment
			**********
			*** Columns section of the report ***
			**********
T	1110		Column Fields
		1112	--Column data type
		1113	--Column heading
		1114	--Column usage
		1115	--Column indent
		1116	--Column width
		1117	--Column edit
		1118	--Column sequence
			**********
			*** Page section of the report ***
			**********
V		1201	Blank lines before heading
V		1202	Blank lines after heading
T	1210		Page heading text table
		1212	--Page heading line number
		1213	--Page heading align
		1214	--Page heading text
V		1301	Blank line before footing
V		1302	Blank line after footing
T	1310		Page footing text table
		1312	--Page footing line number
		1313	--Page footing align
		1314	--Page footing text

Figure 6-15 (Part 1 of 2). Descriptive Names of Encoded Format Form Fields

```
                                    ************************************
                                    ***  Final section of the report  ***
                                    ************************************
          V                 1401    New page for final text
          V                 1402    Put final summary at line
          V                 1403    Skip lines before final text


          T      1410               Final text table
                            1412            --Final text line number
                            1413            --Final text align
                            1414            --Final text

                                    ****************************************
                                    ** Options fields section of the report **
                                    ****************************************
          V                 1501    Detail line spacing
          V                 1502    Outlining for break columns
          V                 1503    Default break text
          V                 1505    Column wrapped lines kept on a page
          V                 1507    Column heading separators
          V                 1508    Break summary separators
          V                 1510    Final summary separators
```

Figure 6-15 (Part 2 of 2). Descriptive Names of Encoded Format Form Fields

A new format exists for the break information in the encoded object. To support the forms that use the original format, Query Management/400 supports both the original format and new format to describe the break information. An attempt to use a combination of the two formats is not allowed and results in ending the import request. All forms are exported using the new format.

Figure 6-16 is a description of the new format that provides for a break level indicator (V record with field number 3080) to indicate the break level. All of the break information that follows each break level indicator is applied to the break level value in the 3080 V record.

The new format uses one set of field numbers to describe the break heading and footing information, which allows for more efficient future expansion of the number of break levels supported.

Record Type	Table Number	Field Number	Description
			************************************
			* Break fields section of the report *
			************************************
V		3080	Break level indicator
V		3101	New page for break heading
V		3102	Repeat column headings
V		3103	Blank lines before heading
V		3104	Blank lines after heading
T	3110		Break heading table
V		3112	--Break heading line number
V		3113	--Break heading align
V		3114	--Break heading text
V		3201	New page for break footing
V		3202	Put break at summary line
V		3203	Blank lines before footing
V		3204	Blank lines after footing
T	3210		Break footing table
V		3212	--Break footing line number
V		3213	--Break footing align
V		3214	--Break footing text

Figure 6-16. Preferred Format for Encoded Break Information

Figure 6-17 on page 6-26 is a description of the original format for representing the break information in the encoded object. This format uses a unique field number for each of the break attributes. This format cannot be used in combination with the new break format.

Record Type	Table Number	Field Number	Description
			************************************
			* Break fields section of the report *
			************************************
V		1601	Break 1: New page for heading
V		1602	Break 1: Repeat column headings
V		1603	Break 1: Blank lines before heading
V		1604	Break 1: Blank lines after heading
T	1610		Break 1: Heading table
V		1612	--Break 1: Heading line number
V		1613	--Break 1: Heading align
V		1614	--Break 1: Heading text
V		1701	Break 1: New page for break footing
V		1702	Break 1: Put break at summary line
V		1703	Break 1: Blank lines before footing
V		1704	Break 1: Blank lines after footing
T	1710		Break 1: Footing table
V		1712	--Break 1: Footing line number
V		1713	--Break 1: Footing align
V		1714	--Break 1: Footing text
V		1801	Break 2: New page for heading
V		1802	Break 2: Repeat column headings
V		1803	Break 2: Blank lines before heading
V		1804	Break 2: Blank lines after heading
T	1810		Break 2: Heading table
V		1812	--Break 2: Heading line number
V		1813	--Break 2: Heading align
V		1814	--Break 2: Heading text
V		1901	Break 2: New page for break footing
V		1902	Break 2: Put break at summary line
V		1903	Break 2: Blank lines before footing
V		1904	Break 2: Blank lines after footing
T	1910		Break 2: Footing table
V		1912	--Break 2: Footing line number
V		1913	--Break 2: Footing align
V		1914	--Break 2: Footing text
V		2001	Break 3: New page for heading
V		2002	Break 3: Repeat column headings
V		2003	Break 3: Blank lines before heading
V		2004	Break 3: Blank lines after heading
T	2010		Break 3: Heading table
V		2012	--Break 3: Heading line number
V		2013	--Break 3: Heading align
V		2014	--Break 3: Heading text

Figure 6-17 (Part 1 of 3). Original Format for Encoded Break Information.

V		2101	Break 3: New page for break footing
V		2102	Break 3: Put break at summary line
V		2103	Break 3: Blank lines before footing
V		2104	Break 3: Blank lines after footing
T	2110		Break 3: Footing table
V		2112	--Break 3: Footing line number
V		2113	--Break 3: Footing align
V		2114	--Break 3: Footing text
V		2201	Break 4: New page for heading
V		2202	Break 4: Repeat column headings
V		2203	Break 4: Blank lines before heading
V		2204	Break 4: Blank lines after heading
T	2210		Break 4: Heading table
V		2212	--Break 4: Heading line number
V		2213	--Break 4: Heading align
V		2214	--Break 4: Heading text
V		2301	Break 4: New page for break footing
V		2302	Break 4: Put break at summary line
V		2303	Break 4: Blank lines before footing
V		2304	Break 4: Blank lines after footing
T	2310		Break 4: Footing table
V		2312	--Break 4: Footing line number
V		2313	--Break 4: Footing align
V		2314	--Break 4: Footing text
V		2401	Break 5: New page for heading
V		2402	Break 5: Repeat column headings
V		2403	Break 5: Blank lines before heading
V		2404	Break 5: Blank lines after heading
T	2410		Break 5: Heading table
V		2412	--Break 5: Heading line number
V		2413	--Break 5: Heading align
V		2414	--Break 5: Heading text
V		2501	Break 5: New page for break footing
V		2502	Break 5: Put break at summary line
V		2503	Break 5: Blank lines before footing
V		2504	Break 5: Blank lines after footing
T	2510		Break 5: Footing table
V		2512	--Break 5: Footing line number
V		2513	--Break 5: Footing align
V		2514	--Break 5: Footing text

Figure 6-17 (Part 2 of 3). Original Format for Encoded Break Information.

```
V                    2601    Break 6: New page for heading
V                    2602    Break 6: Repeat column headings
V                    2603    Break 6: Blank lines before heading
V                    2604    Break 6: Blank lines after heading
T          2610              Break 6: Heading table
V                    2612    --Break 6: Heading line number
V                    2613    --Break 6: Heading align
V                    2614    --Break 6: Heading text

V                    2701    Break 6: New page for break footing
V                    2702    Break 6: Put break at summary line
V                    2703    Break 6: Blank lines before footing
V                    2704    Break 6: Blank lines after footing
T          2710              Break 6: Footing table
V                    2712    --Break 6: Footing line number
V                    2713    --Break 6: Footing align
V                    2714    --Break 6: Footing text
```

Figure 6-17 (Part 3 of 3). Original Format for Encoded Break Information.

Externalized PROC and QUERY Formats

The PROC and SQL QUERY objects are externalized (exported and saved) in the panel format described in "Panel Format" on page 6-2.

Chapter 7. Using Query/400 Definition Information

Query/400 definitions contain specifications for functions that are common to the following:

- Query/400
- SAA CPI Query

Query Management/400 uses this information, saved in Query/400 definition (QRYDFN) objects, to produce reports. This chapter describes how to control Query Management/400 use of the information contained in QRYDFN objects, what to do if the conversion results are unsatisfactory, and how to add function not available through the Query/400 prompted interface.

Query Definition Conversion

Query Management/400 will use information from a query definition (QRYDFN) object when you specify a query management query (QMQRY) or form (QMFORM) object during processing. Query Management/400 then searches the library or library list for an object with a name that matches the one specified. If Query Management/400 finds a QRYDFN object in the library or library list specified, query or form information from the object is used to satisfy the Query Management/400 request issued.

The information derived from the QRYDFN object is available for immediate use as though it came from a QMQRY or QMFORM. When Query Management/400 uses Query/400 information to satisfy a request, a message is returned to alert the user that the information being used is from a QRYDFN. Query Management/400 does not send return codes or messages about possible defects or differences when this information is used. Query Management/400 uses information from a QRYDFN even if problems occurred while deriving it.

The derived information is discarded when the request is completed. To permanently convert information to Query Management/400 objects, retrieve the query or form source from a QRYDFN object. Then use that source to create the Query Management/400 object of that type.

Applying Query Management/400 to QRYDFN Objects

Query Management/400 normally uses information only from Query Management/400 objects. You can request that Query Management/400 use Query/400 information if Query Management/400 form or query information is not available. You can also prevent Query Management/400 form or query information from being used.

On the START command, specify either:

- DSQSCNVT=YES or
- DSQSCNVT=ONLY

When using the following CL commands:

- STRQMPRC (Start Query Management Procedure)
- STRQMQRY (Start Query Management Query)
- RTVQMQRY (Retrieve Query Management Query)
- RTVQMFORM (Retrieve Query Management Form)

specify either ALWQRYDFN(*YES) or ALWQRYDFN(*ONLY).

You can issue the Query Management/400 commands to use information contained in QRYDFN objects. Issue the START command to set the DSQSCNVT variable to YES or ONLY. Specifying ALWQRYDFN(*YES) or ALWQRYDFN(*ONLY) causes the DSQSCNVT variable to be set to YES or ONLY for processing a procedure referred to in an STRQMPRC request.

If you do not want Query/400 definitions to be used during Query Management/400 processing, allow Query Management/400 to default to DSQSCNVT=NO on the START command or ALWQRYDFN(*NO) on the STRQMPRC, STRQMQRY, RTVQMQRY, or RTVQMFORM CL command. Another way to stop Query Management/400 from using a QRYDFN object is to exclude the library containing the Query/400 definition from the library or library list that Query Management/400 searches for the information.

You can also convert queries created in a System/36 environment. Use the Convert System/36 Query (CVTS36QRY) command to convert a System/36 query to a Query/400 definition. Query Management/400 information can then be derived from the QRYDFN object converted from the System/36 query.

QRYDFN Conversion Considerations

A complete conversion of all of the choices specified in the QRYDFN may not be possible. Some Query/400 functions are not applicable to Query Management/400 reports and queries, and other functions, while similar, cannot be converted without loss or distortion. The information derived from a QRYDFN may be unacceptable for use as a QMQRY or QMFORM. The report or data record output produced from it can have obvious defects, or it can be so different from the report or data record output produced by Query/400 that it cannot be used for the same purpose.

For more detailed information about the differences between Query/400 output and Query Management/400 output, see the *Query Management/400 Programmer's Guide*.

Do the following to ensure satisfactory results when using Query Management/400 regularly to run a particular query:

1. Analyze the QRYDFN using the Analyze Query (ANZQRY) command and read all the diagnostic messages produced.
2. Use the STRQMQRY command and inspect the output carefully. You may find undiagnosed defects or discover that some of the ANZQRY diagnostic messages can be disregarded.
3. Use the Work with Query (WRKQRY) command to display the QRYDFN object. Check the definition displays for deviations from the compatibility guidelines shown in "Applying QRYDFN Option Guidelines" on page 7-4. Use the *Change* option to make adjustments to improve your output.
4. Take appropriate action regarding the ANZQRY diagnostic text suggestions that do not involve changing the QRYDFN.

The following sections describe how you can detect and avoid differences and defects in QRYDFN objects.

Analyzing a QRYDFN

Use the Analyze Query (ANZQRY) command to inspect a Query/400 definition to find problems that could occur when deriving information for Query Management/400 use. The ANZQRY command returns diagnostic messages that detail the potential differences between how Query/400 and Query Management/400 use information derived from the analyzed QRYDFN object.

A completion message shows the highest severity of the potential problems found. If the severity code is greater than 10, the message help may contain warnings about easily overlooked and possibly serious problems involving the system actions stated in the messages.

A low severity code does not necessarily mean there will be no serious problem when you use the derived information, nor does a high severity code mean that the QRYDFN should not be used. Consider the following when using the ANZQRY command:

- Analyze processing checks the information contained in the specified file but ignores any database file overrides in effect. There is no verification that the information saved in the QRYDFN object is still comparable to that found in the file definitions.
- Analyze processing does not predict errors that could occur during conversion, such as a SELECT statement that becomes too large.
- Analyze processing does not predict errors that could occur during run time, such as the following:
  - Structured Query Language (SQL) syntax errors
  - SQL data errors (missing fields, mismatched type comparisons)
  - Excessive formatted report width
  - Inappropriate field data types for the specified usage or editing
- Analyze processing ignores the following differences between Query/400 and Query Management/400:
  - Loss of translatable final totals heading text
  - Loss of leading blanks in column heading lines
  - Different alignment of oversized column headings
  - Breaking of column heading lines at underscore characters
  - No extension of break or final text into space before summary
  - No extension of break or final text past last data column
  - Different summary types on the same line
  - Different defaults for result field size
  - Different algorithms for calculation of result scale and precision
  - Different values from calculations involving data of certain types
  - Different handling of double-byte character set (DBCS) character data comparisons
- Severity codes do not take into account the number of messages generated by the analyze processing, and can be misleading because a diagnosed problem may not be as severe as indicated. For example, no longer ignoring decimal data errors may not be a problem for a particular query, but it is diagnosed as a severity 30 problem even for a query with no numeric fields.

Inspecting the Output

In some cases, the only way to detect defects and unacceptable differences is to inspect the output. This may also be the only way to evaluate the actual severity of an ANZQRY-diagnosed problem. Use the STRQMQRY command for Query Management/400 and the RUNQRY command for Query/400 to get comparable command output.

Look for the following differences when running a derived query:

- Records not coming from the *LAST member or the member specified by name
- Omission of unmatched records
- Loss of columns or different column order
- Columns added for extra summary functions
- Different record order or selection set
- Different length or precision of result field data columns
- Different values or unexpected numeric overflow in result fields
- Different values or unexpected numeric overflow of sums or averages
- Divide by zero errors for numeric calculations

Look for the following differences when displaying or printing a report using a derived form:

- Text truncation
- Unresolved text insertion variables
- Editing changes
- Report column width changed
- Heading or footing alignment changed
- No line wrapping
- No cover page
- No page number or date and time information in page headings

Applying QRYDFN Option Guidelines

You can avoid many potential problems by following established guidelines when creating or changing a Query/400 definition intended for Query Management/400 use. Use the WRKQRY command to create a new QRYDFN object, or change an existing object and ensure the option fields contain values that are recommended for Query Management/400 compatibility. Use the following guidelines when specifying values for the following options in a QRYDFN object destined for Query Management/400 use:

- Specify file selections.
  - Select only files with single formats.
  - Select only the *FIRST member.
  - Specify the type of join.
  - Use only matched record joining.
- Define result fields.
  - Use size overrides if an expression involves division.
  - Use size overrides only for numeric column formatting control.
  - Use an underscore character in a column heading only where you want the line to break text.
  - Do not use column headings with data alignment dependencies.
  - Do not use multiple-line figures in column headings.
  - Do not use a line for column heading separator characters.
  - Use SUBSTR (not a formatting override) to reduce character field size.

- Select and sequence fields.
  - Make specific field selections.
  - Do not select more than 255 fields.
- Select records.
  - Do not use LIKE to test a result field defined using || or SUBSTR.
  - Use dependent value syntax where a global variable is desired.
- Select sort fields.
  - Do not sort a character field unless EBCDIC sequencing is acceptable.
- Select collating sequence.
  - Select EBCDIC sequencing.
- Specify report column formatting.
  - Allow space for break or final text to the left of summaries.
  - Override the column heading if overriding the length of a file field.
  - Use an underscore character in a column heading only where you want the line to break text.
  - Do not use column headings with data alignment dependencies.
  - Do not use multiple-line figures in column headings.
  - Do not use a line for column heading separator characters.
  - Do not omit break fields.
  - Override editing when overriding numeric file field size.
- Define numeric field editing.
  - Use edit code or numeric editing choices.
- Specify edit code.
  - Use only J and M edit codes.
  - Do not use the modifier for the asterisk fill option.
  - Do not use the modifier for a floating currency symbol with M specified as the edit code.
- Describe numeric field editing.
  - Use a description that can be converted to an SAA edit code.
  - Do not request suppression of zero values.
  - Do not request asterisk fill.
  - Do not request a single leading zero.
- Select report summary functions.
  - Do not request more than one summary function per field unless you want columns added for the extra functions.
  - Do not request a summary function for a break field unless you want a column added for that function.
- Define report breaks.
  - Do not break on a result field if you want detail records to be omitted.
  - Define only one level if you want detail records to be omitted.
- Format report breaks.
  - Do not use break text if the first report field has a summary function.
  - Use any report field name for a break text variable.
  - Define final text to replace the *Final Totals* default.
  - Omit level 0 summaries if some other level is defined and you want detail records to be omitted.

- Select output type and output form.
  - Do not define a query for producing summary-only database file output.
  - Put run-time device options in CL commands or procedures.
  - Use line spacing if desired.
  - Do not define a cover page unless the output will be final summaries only.
  - Do not use line wrapping.
  - Do not use more than 55 characters for page heading or footing text.
  - Use any report field name for a page text variable.
- Specify processing options.
  - Do not request numeric overflow truncation.
  - Do not request ignoring decimal data errors.

Conversion Details

Figure 7-1 on page 7-7 shows the relationship between the Work with Query displays used to prompt for query definition specifications and the corresponding sections of query management query and query management form objects. Some Structured Query Language (SQL) functions that are supported by Query Management/400 but not represented in Figure 7-1 (such as sublists and certain scalar functions) are not available through the prompted interface but are available in SQL.

Work with Query Display	Query Management Object
Specify File Selections	
- File, Library	SELECT FROM
- Member	N/A
- Format	N/A
- File ID	SELECT FROM
	SELECT list, ORDER BY
	(used as Field qualifier)
Specify Type of Join	
- Type of join	N/A
Specify How to Join Files	
- Field__Test__Field	SELECT WHERE
Define Result Fields	
- Field__ (if no column heading)	Column: Column heading, Width
- Expression__	SELECT list, WHERE
	(substituted for Field)
- Column heading__ (if no override)	Column: Column heading, Width
- Len__	Column: Edit, Width
- Dec	Column: Edit, Width
Select and Sequence Fields	
- Seq__Field	SELECT list
Select Records	
- AND/OR__Field__Test__Value	SELECT WHERE
Select Sort Fields	
- Sort Prty__A/D__Field	SELECT ORDER BY
	SELECT GROUP BY
	SELECT HAVING
Select Collating Sequence	
- Collating sequence option	N/A
- Table, Library	N/A
Define Collating Sequence	
- Sequence__Char	N/A
Specify Report Column Formatting	
- Field__	
	Column: Seq
	Column: Datatype
- Column spacing__	Column: Indent
- Column heading__ (override only)	Column: Column heading, Width
- Len__ (override only)	Column: Usage, Edit, Width
(0)	(OMIT)
- Dec__ (override only)	Column: Edit, Width
(#)	(#)
- Edit (override only)	Column: Edit, Width
(Untransformable numeric editing)	(K)
Define Numeric Field Editing	
- Edit option	N/A
Describe Numeric Field Editing	
- Decimal point, and so on	Column: Edit, Width
(Transformable description)	(matched SAA code)

Figure 7-1 (Part 1 of 3). Correlation between the Work with Query Display and Query Management/400 Objects

Work with Query Display	Query Management Object
Describe Date/Time Field Editing	
- Date/time separator	N/A
Specify Edit Code	
- Edit code, Optional modifier	Column: Edit, Width
(J)	(K)
(J$)	(D)
(M)	(L)
Specify Edit Word	
- Edit word	N/A
- Edit word for summary total	N/A
Select Report Summary Functions	
- Options__Field	Column: Usage, Width
(1=Total)	(SUM)
(2=Average)	(AVERAGE)
(3=Minimum)	(MINIMUM)
(4=Maximum)	(MAXIMUM)
(5=Count)	(COUNT)
	(FIRST)
	(LAST)
Define Report Breaks	
- Break level__Sort Prty__Field	Column: Usage
(1-6)	(BREAK1-BREAK6)
Format Report Break (level 0)	
	Final: New page for final text
- Suppress summaries	Final: Put final summary at line
(Y=Yes, N=No (if text present))	(NONE, 2)
	Final: Blank lines before footing
- Break text	Final: Final footing text
(1st and only line)	(Line 1)
	(Lines 2-12)
Format Report Break (level 1-6)	
- Skip to new page	Break: New page for heading
	Break: Repeat column heading
	Break: Blank lines before heading
	Break: Blank lines after heading
	Break: New page for footing
	Break: Blank lines before footing
	Break: Blank lines after footing
- Suppress summaries	Break: Put summary at line
(Y=Yes, N=No (if text present))	(NONE, 2)
	Break: Break heading text
- Break text	Break: Break footing text
(1st and only line)	(Line 1)
	(Lines 2-5)
Select Output Type and Output Form	
- Output type	N/A
- Form of output	N/A
- Line wrapping	N/A

Figure 7-1 (Part 2 of 3). Correlation between the Work with Query Display and Query Management/400 Objects

Work with Query Display	Query Management Object
Define Printer Output	
- Printer	N/A
- Form size	N/A
- Start line	N/A
- End line	N/A
- Line spacing (1-3)	Options: Detail line spacing (1-3) (4)
	Options: Outlining for break columns
	Options: Default break text
	Options: Column-wrapped lines kept
	Options: Column heading separators
	Options: Break summary separators
	Options: Final summary separators
- Print definition	N/A
Define Spooled Output	
- Spool the output	N/A
- Form type	N/A
- Copies	N/A
- Hold	N/A
Specify Cover Page	
- Print cover page	N/A
- Cover page text (up to 5 lines)	N/A
Specify Page Headings and Footings	
- Print standard page headings	
	Page: Blank lines before heading
	Page: Blank lines after heading
- Page heading (lines 1-3)	Page: Page heading text (Lines 1-3) (Lines 4-5)
	Page: Blank lines before footing
	Page: Blank lines after footing
- Page footing (1st and only line)	Page: Page footing text (Line 1) (Lines 2-5)
Define Database File Output	
- File, Library, Member	N/A
- Data in file	N/A
- Authority (for new file)	N/A
- Text (for new file)	N/A
- Print definition	N/A
Specify Processing Options	
- Use rounding	N/A
- Ignore decimal data errors	N/A

Figure 7-1 (Part 3 of 3). Correlation between the Work with Query Display and Query Management/400 Objects

The following actions resulting from conversion may be unexpected. You should be aware of these actions to obtain the best results.

- Every character in an expression or test value is converted to uppercase unless it is in a string constant. This includes the characters in a delimited name.
- SQL reserved words used as field names are placed in quotation marks in a derived SELECT statement.

- The corresponding expression is substituted for each result field name that would otherwise appear in a SELECT list or record selection test. Substitution is recursive because result field names can be used in the expressions for other result fields. Column numbers are substituted for result field names in the ORDER BY clause.
- If any join tests exist, they are used to start the WHERE clause, and any record selection tests are performed using the AND operator.
- Dependent values in record selection tests are converted to global variables. For example, the dependent value :t01."collection" is converted to &T01_COLLECTION.
- Valid decimal point delimiters are converted to the delimiter indicated by the Query Decimal Format (QDECFMT) system value in numeric constants in a derived SELECT statement.
- SAA database processing truncates decimal positions when dividing unscaled values. For example, the value calculated for 2 divided by 3 is 0. For this reason, the decimal point delimiter indicated by the QDECFMT system value is joined to every number without a decimal point delimiter in expressions in a derived SELECT statement.
- The form retrieved from a QRYDFN object may have blank column entries causing warnings when the form is used to create a QMFORM object.
- The *Column heading* field is left blank on a column with no summary function unless the column is a result field or there is a column heading override for it. The field name is the default if no heading is defined.
- Underline characters in the heading text are not replaced with substitute characters. A single line heading such as 1_9_9_0 is converted with no changes and causes four lines of heading when the derived form information is used.

  The following actions may also occur:

  - *NONE, designating no column heading, is converted to a single underline character; or, in a summary function column, to the caption Query/400 uses for the values.
  - A column heading string is built by removing leading and trailing blanks from each line, up to and including the last nonblank line, and then connecting the resulting segments with single underline characters.
  - The heading for a column with a summary function is built by adding the column heading string to the caption that Query/400 uses for the values. If the resulting string is longer than 62 characters, it is truncated.
  - If no heading string exists to add to the summary function caption, the field name is used. Any heading defined for the field as part of a file definition is ignored.
- The character part of the Edit value is left blank if any of the following occur:
  - The column holds only COUNT summary function values.
  - No size override exists, unless the column is a result field for which a size has been specified.
  - No override editing is defined, the column is not a result field, and there is a numeric decimal positions override.

- Query Management/400 uses the *C* edit code if the decimal positions override indicates a character field. The *K* edit code is used for a numeric field if a closer match to an SAA CPI Query edit code cannot be determined for the edit override, or if no override exists and the column is for a result field.
- Only the type of override editing indicated by the *Edit* option choice is considered.
- The effect of system defaults on RPG edit code editing is disregarded. The J, J with currency symbol, and M edit codes are always converted to K, D, and L, respectively.
- Edit description choices not involved in the actual editing (a left or right currency symbol when no currency symbols are to be used, for example) are ignored when comparisons are made.
- The numeric part of the *Edit* value is left blank if the character part is blank or C. This value is also left blank if the number of decimal positions to be used cannot be determined from a decimal positions value saved as an override (with a nonzero length override) or as part of the definition of a result field.
- The Width value for a column is left blank unless the information saved in the QRYDFN object specifies the width requirements for everything that appears in the column. A column formatting length override can influence the Width, but only determines the number of digits or decimal positions formatted when nothing else in the column (such as a heading segment or count summary value 9,999,999) is wider than the edited data and the override length is smaller than the data width assumed by database processing.
- Page, break, and final text are adjusted in the following ways:
  - Strings of consecutive blanks are compressed to single blanks.
  - Fields referencing insert variables are converted to column referencing variables and special variables (such as &time, &date, &page) are converted to all uppercase characters. Strings starting with an ampersand (&) are recognized as text insert variables only if the character after the & is not a blank or a numeric digit, and if the string is ended by one of the following characters:
    - Blank
    - Slash
    - Colon
    - Dash
    - Ampersand
    - Underscore
    - DBCS shift-out character

    All selected fields are considered (in report order) when converting a field reference to a column reference.
  - If the resolved text exceeds 55 characters, it is truncated at the first blank or ampersand of the first 56 positions. If no blank or ampersand is found, the text is not used.
  - DBCS strings left open due to truncation are closed.

Creating Query Management/400 Objects from QRYDFN Objects

The following steps represent a typical way in which a QRYDFN could be permanently converted to a Query Management/400 form object and a Query Management/400 query object:

1. Run the ANZQRY command on the selected QRYDFN. The messages produced give you suggestions for adjusting the object before attempting to convert it to Query Management/400 objects.
2. Create separate source file members—one for externalized queries and one for externalized forms.
3. Use the WRKQRY command to change and save the resolved QRYDFN object if there have been any changes to the file definitions referred to. You can also use this command to change the object to improve the conversion or to add functions tolerated but not supported by Query/400.
4. Use CL or Query Management/400 commands (RTVQMQRY or EXPORT QUERY, for example) to extract the usable information.
5. Edit the externalized objects if you want to add functions not available through the Query/400 prompted interface.
6. Use CL or Query Management/400 commands (CRTQMQRY or IMPORT QUERY, for example) to create Query Management/400 objects.

Figure 7-2 illustrates how a Query/400 QRYDFN is converted for use as query management query and query management form objects.

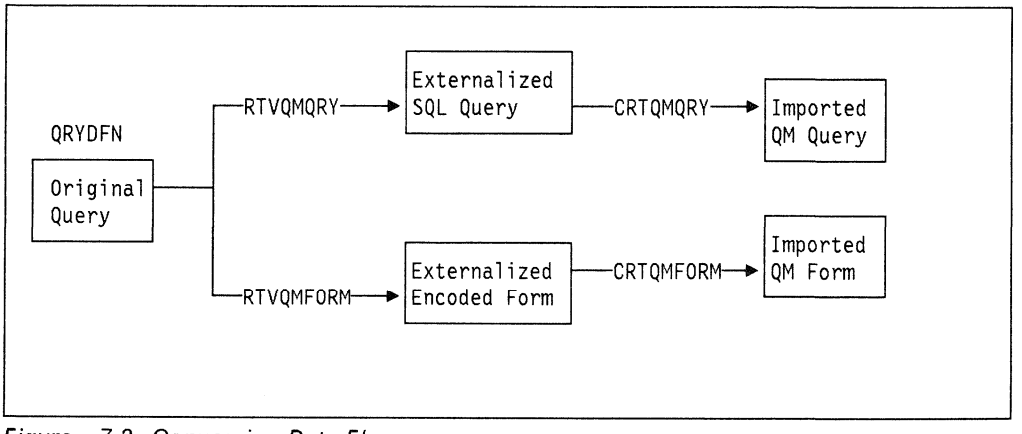


Figure 7-2. Conversion Data Flow

Figure 7-3 on page 7-13 is an example of how you can use CL commands to convert Query/400 QRYDFN objects to Query Management/400 objects.

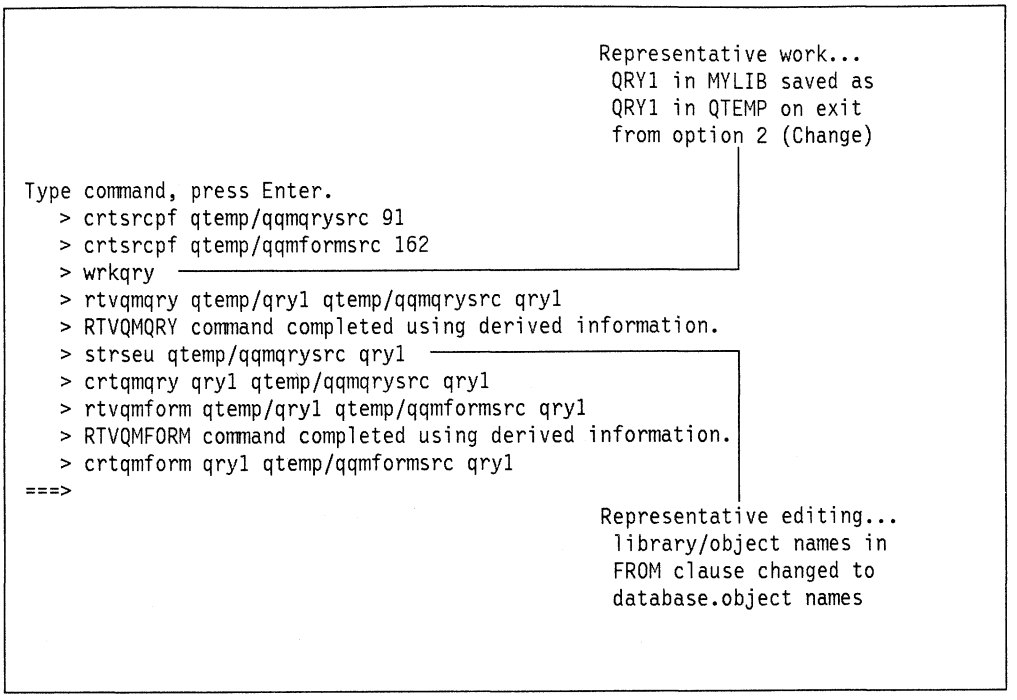


Figure 7-3. Sample CL Command Sequence for QRYDFN Conversion

Adding SAA Function

You can add SAA function that is not supported by Query/400 by changing a QRYDFN object, or a Query Management/400 source retrieved from a QRYDFN object. The following list describes how to obtain additional SAA function:

- Use the Work with Query (WRKWRY) command to define functions that are tolerated but not supported by Query/400:
  - &field insertion variables in page text
  - &field insertion variables ended with an underscore character
  - Other than break field insertion variables in break or final text
  - &column# insertion variables ended with any nondigit character
- Use the Start Source Entry Utility (STRSEU) command to edit retrieved source to add functions that cannot be defined using the WRKQRY command
  1. Retrieve the source from a QRYDFN object or query management query (QMQRY) or form (QMFORM) object using the RTVQMQRY and RTVQMFORM CL commands with the ALWQRYDFN(*YES) parameter.
  2. Edit the source to add the desired functions.
  3. Use the edited source to create a QMQRY or QMFORM object that can be referred to in subsequent Query Management/400 requests using the CRTQMQRY and CRTQMFORM CL commands.

You can add the following functions to Query Management/400 query source:

- DISTINCT records instead of ALL records
- Selection of all fields using an asterisk (*)
- SAA naming conventions in the FROM clause
- Column or scalar function as an SQL expression or predicate test value
- NOT as a search condition qualifier
- NOT IN and NOT LIKE predicate tests

- GROUP BY and HAVING clauses
- Use of parentheses with connectors

You can add the following functions to Query Management/400 form source:

- Character field editing
- Different SAA Query edit codes for numeric editing
- Explicit column width control
- FIRST and LAST uses
- Ability to use more than 9 break columns
- Resequencing in form definition
- Report area spacing
- Summary value placement on a line other than line 2
- Break heading text
- Additional (more than AS/400 limits) text lines
- Text line alignment
- Control of framing (separators, default break text, or outlining, for example)

See the *SAA CPI Database Reference* for detailed information about SQL syntax for editing query source, and the *SAA CPI Query Reference* for detailed information about the encoded form layout for editing form source.

Appendix A. DBCS Data

This section addresses the concerns you will have when using double-byte character set (DBCS) data with Query Management/400. It is intended to point out how using Query Management/400 with DBCS is different from using Query Management/400 with single-byte data.

Ask your Information Center if you have the proper hardware and software necessary to operate with DBCS.

What Is Double-Byte Data?

Double-byte character sets are so named because the internal representation for each character requires two bytes of space. Languages such as Japanese and Korean require such double-byte representations.

Those character sets that require only a single byte to represent each character internally are called single-byte character sets. Languages such as English, German, and French are represented with single-byte character sets. In some cases, Katakana can also be represented by a single-byte character set, because it can be represented internally in single bytes.

References made in this chapter to "mixed" data mean that strings of both single and double-byte data are contained together in one data field.

How Does Double-Byte Data Look When Displayed?

DBCS data differs from single-byte data when it displays on your terminal. It occupies twice as much space on the screen as single-byte data.

You can display any Query Management/400 object containing DBCS data whether you have a DBCS terminal or not.

What Data Types Can Be Used with DBCS Data?

You can save DBCS data in your database if you define the columns in which you save the data as one of two basic types: character or graphic. Whether you save your DBCS data in character or graphic columns depends on your needs.

- If the column contains DBCS data with the appropriate SO and SI characters, the data can be put into character data type columns.
- If the column contains only DBCS data, you can define the column as either character or graphic.

Specifically, DBCS data may be saved in database columns defined as these data types:

- Character

  DBCS data, when preceded and followed by the single-byte apostrophe ('), can be put into character data type columns. DBCS data strings can also be mixed with single-byte character strings. Use this data type if all entries in the column have the same length.

For additional information on the maximum lengths of these columns, refer to *SQL/400* Reference*.

Using DBCS Data in Query Management/400

The following sections explain how using DBCS data in Query Management/400 is *different* from using single-byte data.

Using DBCS Data in Input Fields

All Query Management/400 input fields, except names, allow DBCS data.

Using DBCS Data in Queries

The following can be in either DBCS, or mixed single-byte and DBCS:

- Substitution values
- Delimited strings in character data type fields
- Comments

SQL keywords must be in English.

Using DBCS in the FORM

DBCS or mixed DBCS and single-byte data can be used in the FORM panels as:

- Break text
- Page text
- Final text

The following sections describe how DBCS or mixed single-byte and DBCS data are used in Query Management/400.

USAGE Specifies how to use a column. FORM usages must be single-byte characters.

INDENT The number of blank spaces to the left of a column; this field is used to separate a column from the column before it, or the left margin.

WIDTH Specifies how wide you want the column to be.

If the WIDTH is less than 4 spaces (the minimum space required to display a single DBCS character), the column will be filled with asterisks (***).

EDIT Codes

Determine how values in a column are punctuated, if at all.

These codes must be entered on the form in single-byte characters.

- **C** is the edit code for character data type columns. It makes no change in the display of a value.
- **CW** is the edit code for character data columns to be wrapped. It makes no change in the display of a value. But if the value cannot fit on one line in the column, the text wraps according to the width of the column. That is, instead of cutting off the data at the end of the column, as much data as possible is put on a line in the column, and then the data continues wrapping on the next line.

- **CT** is the edit code for columns of character data to be wrapped according to the column text. It makes no change in the display of a value, but if the value cannot fit on one line in the column, the column wraps according to the text in the column. That is, instead of cutting off the data at the end of the column, as much data as possible is fitted into a line; then the line interrupts at a single-byte blank and the data continues wrapping on the next line. If a string of data is too long to fit in the column and does not contain a single-byte blank, the data wraps by width until a single-byte blank is found.

How Data Truncation is Handled

Query Management/400 truncates displayed DBCS data at a field or screen boundary in a way that avoids splitting DBCS characters. Paging is necessary to view the characters on the truncated lines.

Shift-in and shift-out characters and column widths are contained in the width or the column and are not placed in the indent or margins of the report.

Exporting DBCS Data

Data defined as character that contains DBCS data can be exported.

The column width of exported data is the number of DBCS characters in the data, which is half the number of bytes used to store the data. Column data is stored in the data record exactly as it comes from the database.

Importing DBCS Data

DBCS data can be imported in queries, procedures, and forms. When importing DBCS queries and procedures this way, be certain that record length does not exceed 79 bytes.

Printing DBCS Reports

If you are using DBCS data and the page splits, printing resumes on the second and subsequent pages of the report, at the fourth byte position from the left side of the page.

Glossary

This glossary includes terms and definitions from:

- The *American National Dictionary for Information Processing Systems*, copyright 1982 by the Computer and Business Equipment Manufacturers Association (CBEMA). Copies may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The *Information Technology Vocabulary*, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Committee (ISO/IEC JTC1/SC1). Definitions of published segments of the vocabularies are identified by the symbol (I) after the definition; definitions from draft international standards, draft proposals, and working papers in development by the ISO/IEC JTC1/SC1 vocabulary subcommittee are identified by the symbol (T) after the definition, indicating final agreement has not yet been reached among participating members.

application program. A program used to perform a particular data processing task, such as inventory control or payroll.

asterisk fill. A type of numeric editing that puts asterisks to the left of a number to fill unused positions. Example: *****476.12

authorization list. A list of two or more user IDs and their authorities for system resources. The system-recognized identifier for the object type is *AUTL.

callable interface. (1) The name of the interface program, the definition of the arguments passed to the interface program, and the definition of the data structures passed to the interface program. (2) In query management, the Common Programming Interface (CPI) that includes the definitions of the control blocks and constants used for the interface.

character set. A group of characters used for a specific reason; for example, the set of characters the display station can display, the set of characters a printer can print, or a particular set of graphic characters in a code page; for example, the 256 EBCDIC characters.

character string. A sequence of consecutive characters that are used as a value.

COBOL (COmmon Business Oriented Language). A high-level programming language, based on English, that is used primarily for commercial data processing. See also *IBM SAA AD/Cycle COBOL/400 Version 2 (COBOL/400)*.

COBOL/400. See *IBM SAA AD/Cycle COBOL/400 Version 2 (COBOL/400)*.

command name. In query management, the verb in a query command that specifies the action to be performed.

command string. In query management, a character string that contains a query command.

Common Programming Interface (CPI). In the Systems Application Architecture (SAA) solution, a set of software interfaces, conventions, and protocols that provide a framework for writing applications with cross-system consistency.

communications area. In query management, a control block used to communicate between the system code supporting the Common Programming Interface (CPI) and the application program using the CPI.

constant. Data that has an unchanging, predefined value to be used in processing.

data type. A characteristic used for defining data as numeric or character. variable can assume or that a function can return.

database file. One of several types of the system object type *FILE kept in the system that contains descriptions of how input data is to be presented to a program from internal storage and how output data is to be presented to internal storage from a program. See also *physical file* and *logical file*.

DBCS. See *double-byte character set (DBCS)*.

double-byte character set (DBCS). A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, displaying, and printing of DBCS characters requires hardware and programs that support DBCS. Four double-byte character sets are supported by the system: Japanese, Korean, Simplified Chinese, and Traditional Chinese. Contrast with *single-byte character set*.

EBCDIC. See *extended binary-coded decimal interchange code (EBCDIC)*.

EBCDIC character. Any one of the symbols included in the 8-bit EBCDIC set.

edit. (1) To interactively add, change, delete, or rearrange the data; for example, to insert or remove characters, sentences, or paragraphs, or to insert or remove characters in dates or decimal numbers. (2) To make changes to a document by adding, changing, or removing text. (3) To change a numeric field for output by suppressing zeros and inserting commas, periods, currency symbols, the sign status, or other constant information.

edit code. A letter or number indicating that editing should be done according to a defined pattern before a field is displayed or printed. Contrast with *edit word*.

edit description. A description of a user-defined edit code. The system-recognized identifier is *EDTD.

edit word. A user-defined word with a specific format that indicates how editing should be done. Contrast with *edit code*.

element. In a list of parameter values, one value.

exported form. In query management, the source file member that results from running an EXPORT FORM command.

expression. In Query/400, a representation of a value with variables or constants appearing alone or in combination with arithmetic operators.

extended binary-coded decimal interchange code (EBCDIC). A coded character set consisting of 8-bit coded characters.

externalized form. In query management, the name of the file resulting from running an EXPORT command against a form.

externalized query. In query management, the name of the form resulting from running an EXPORT command against a query.

file overrides. Attributes specified at run time that change the attributes specified in the file description or in the program.

form. In query management, an object that describes how to format the data for printing or displaying a report.

format. (1) A defined arrangement of such things as characters, fields, and lines, usually used for displays, printouts, files, or documents. (2) The arrangement or layout of fields in a record.

function. Any instruction or set of related instructions that perform a specific operation.

generic name. The characters common to object names that can be used to identify a group of objects. A generic name ends with an asterisk (*). For example, ORD* identifies all objects whose names begin with the characters ORD.

global variable. A named entity within query management that can be assigned a value used for communications between an application program and Query Management/400.

IBM Operating System/2 (OS/2). Pertaining to the IBM licensed program that can be used as the operating system for personal computers. The OS/2 licensed program can perform multiple tasks at the same time.

IBM Operating System/400 Version 2 (OS/400). Pertaining to the IBM licensed program that can be used as the operating system for the AS/400 system.

IBM SAA AD/Cycle COBOL/400 Version 2 (COBOL/400). The IBM licensed program that is the Systems Application Architecture platform COBOL programming language available on the AS/400 system, including system-specific functions.

IBM SAA AD/Cycle RPG/400 Version 2 (RPG/400). The IBM licensed program that is the Systems Application Architecture platform RPG programming language available on the AS/400 system, including system-specific functions.

IBM SAA Structured Query Language/400 Version 2 (SQL/400). The IBM licensed program that is the Systems Application Architecture platform of SQL.

instance ID. In query management, an identifier in the communications area. An instance ID is used to identify a particular query instance being used by an application program. See also *query instance*.

interactive mode. In query management, the query mode associated with a query instance that allows users to interact with the query commands while a procedure is running.

key. The value used to identify a record in a keyed sequence file.

keyword. (1) A name that identifies a parameter in a command. (2) In query management, one of the predefined words associated with a query command.

logical file. A description of how data is to be presented to or received from a program. This type of database file contains no data, but it defines record formats for one or more physical files. See also *database file*. Contrast with *physical file*.

mode. The session limits and common characteristics of the sessions associated with advanced-program-to-program communications (APPC) devices managed as a unit with a remote location.

null. (1) The name for an EBCDIC character that represents hex 00. (2) In SQL, a special value that indicates the absence of information.

object name. The name of an object. Contrast with *qualified name*.

OS/2. See *IBM Operating System/2 (OS/2)*.

OS/400. See *IBM Operating System/400 Version 2 (OS/400)*.

override. (1) To specify attributes at run time that change the attributes specified in the file description or in the program. (2) The attributes specified at run time that change the attributes specified in the file description or in the program.

physical file. A description of how data is to be presented to or received from a program and how data is actually stored in the database. A physical file contains one record format and one or more members. See also *database file*. Contrast with *logical file*.

procedure. In query management, a query object that consists of a related set of query commands. A procedure allows an application to run multiple query commands through one call to the callable interface.

public authority. The authority given to users who do not have any specific (private) authority to an object, who are not on the authorization list (if one is specified for the object), and whose group profile has no specific authority to the object.

qualified name. The name of the library containing the object and the name of the object. Contrast with *object name*.

Query. The shortened name for the Query/400 licensed program.

query. (1) A request to select and copy from a file or files one or more records based on defined conditions. For example, a request for a list of all customers in a customer master file, whose balance is greater than $1000. (2) The query management object that is used to define queries against relational data.

query command. The name of an action, and any associated parameters, that can be performed by query management. The query commands include ERASE, EXIT, EXPORT, GET, IMPORT, PRINT, RUN, SAVE, SET, and START.

query command procedure. In query management, a type of query procedure that contains a subset of the query commands allowed in a query procedure. The query command procedure can be used for initializing global variables.

query definition. In Query/400, information about a query that is stored in the system. The system-recognized identifier for the object type is *QRYDFN.

query instance. In query management, a collection of system resources and a set of query commands within an application program.

query management form. In query management, the type name of the OS/400 object on the AS/400 system that is comparable to the term form object as used for the Systems Application Architecture (SAA) solution. The system-recognized identifier for a query management form is *QMFORM.

query management object. In query management, a collective term to describe any of the query management objects: query, form, or procedure.

query management procedure. The name used in Query Management/400 to describe a source physical file member that contains query procedure language statements.

query management query. In query management, the type name of the OS/400 object on the AS/400 system that is comparable to the term query object as used for the Systems Application Architecture (SAA) solution. The system-recognized identifier for a query management query is *QMQRY.

Query Management/400. A function of the OS/400 licensed program that is the AS/400 implementation of the SAA solution for the Query common programming interface.

query mode. In query management, the processing mode associated with a query instance.

report. In query management, the formatted data that results from running a query and applying a form to it.

report break. In Query/400, a blank line or new page that appears in a report when the contents of a specified field in the report change. A report break can contain column summaries.

RPG. Report Program Generator. A programming language designed for writing application programs for business data processing requirements. The application programs range from report writing and inquiry programs to applications, such as payroll, order entry, and production planning. See also *IBM SAA AD/Cycle RPG/400 Version 2 (RPG/400)*.

SAA. See *Systems Application Architecture (SAA)*.

SAA solution. See *Systems Application Architecture (SAA) solution*.

source file. A file of programming code that is not compiled into machine language. A source file can be

created by the specification of FILETYPE(*SRC) on the Create command. A source file can contain source statements for such items as high-level language programs and data description specifications. Contrast with *data file*.

SQL. See *Structured Query Language (SQL)*.

SQL query. In query management, a type of query that is created by running an IMPORT command against a file containing an SQL statement.

SQL/400. See *IBM SAA Structured Query Language/400 Version 2 (SQL/400)*.

Structured Query Language (SQL). A language that can be used within host programming languages or interactively to put information into a database and to get and organize selected information from a database. SQL can also control access to database resources. See also *IBM SAA Structured Query Language/400 Version 2 (SQL/400)*.

Systems Application Architecture (SAA). Pertaining to an architecture defining a set of rules for designing a common user interface, programming interface, application programs, and communications support for strategic operating systems such as the OS/2, OS/400, VM/370, and MVS/370 operating systems.

Systems Application Architecture (SAA) application. A software program that provides end-user function, runs on an SAA platform, and uses implementations of SAA specifications.

Systems Application Architecture (SAA) component. A programming service or language that carries out an SAA specification.

Systems Application Architecture (SAA) environment. The operating system (and subsystem where applicable) that runs on the supported hardware, such as a PS/2* work station AS/400 system, and a System/370 computer.

Systems Application Architecture (SAA) platform. A complete base for building applications and solutions, comprised of the implementation of SAA specifications in an SAA environment. For example, a platform is the OS/400 operating system, which provides the environment, plus Common User Access (CUA), Common Communications Support (CCS), Common Programming Interface (CPI), and application enablers on an AS/400 system.

Systems Application Architecture (SAA) solution. One or more SAA applications that meet a customer requirement and may exist across multiple SAA platforms.

Systems Application Architecture (SAA) specification. A definition of an architecture, interface, protocol, model, or structure designed for an SAA environment.

time stamp. In Query/400, the identification of the day and time a query report is created, which Query automatically provides on each report.

use authority. An object authority that allows the user to run a program or to display the contents of a file. Use authority combines object operational authority and read authority.

value. (1) Data (numbers or character strings) entered in any entry field, and data supplied in parameters of CL commands. (2) The smallest unit of data manipulated by the Structured Query Language. (3) In query management, a quantity assigned to a keyword or variable associated with a query command. If the keyword is part of the command string, its value is separated from it with an equal sign (=). If the keyword is an argument on the extended interface, its value will also be an argument.

value type. In query management, one of the arguments passed to the extended interface. The value type specifies the data type of the value associated with the keyword.

Bibliography

The following books contain information about topics described or referred to in this book. They are listed with their full title and base order number. When these books are referred to in text, a shortened version of the title is used.

For the AS/400 System

The following AS/400 manuals contain additional information you may need when you use Query Management/400:

- *Programming: Control Language Reference*, SBOF-0481, provides information about using control language (CL) commands.
- *OS/400 Query Management/400 Programmer's Guide*, SC21-8192, contains information about using Query Management/400 with the AS/400 system.
- *Programming: Structured Query Language/400 Programmer's Guide*, SC21-9609, provides information about working with Structured Query Language (SQL) and applications common with Query Management/400.
- *Query: User's Guide*, SC21-9614, provides information about working with the AS/400 Query product.
- *Security Concepts and Planning*, SC41-8083, provides information about security concepts for the AS/400 system.

For the SAA Solution

An introduction to the SAA solution in general can be found in *SAA: An Overview*, GC26-4341.

An introduction to the common programming interface can be found in *Common Programming Interface: Summary*, GC26-4675.

More detailed information on the components of the common programming interface is available in the following SAA manuals:

Application Generator Reference, SC26-4355
C Reference—Level 2, SC09-1308
COBOL Reference, SC26-4354
Communications Reference, SC26-4399
Database Reference, SC26-4348
Dialog Reference, SC26-4356
FORTRAN Reference, SC26-4357
PL/I Reference, SC26-4381
Presentation Reference, SC26-4359
Procedures Language Reference, SC26-4358
Query Reference, SC26-4349
RPG Reference, SC09-1286.

General programming advice may be found in *Writing Applications: A Design Guide*, SC26-4362. An introduction to the use of the AD/Cycle application development tools can be found in *AD/Cycle Concepts*, GC26-4531.

A definition of the common user access can be found in *Common User Access: Advanced Interface Design Guide*, SC26-4582, and *Common User Access: Basic Interface Design Guide*, SC26-4583.

More information on the common communications support can be found in *Common Communications Support: Summary*, GC31-6810.

For Implementation on the System/370 Computer

The following manuals contain additional information that you may need for implementation on the System/370 computer:

- *Query Management Facility: Advanced Users Guide*, SC26-4343, provides guidance information for creating queries, modifying the form, and printing reports in QMF.
- *Query Management Facility: Reference*, SC26-4344, provides a reference of the SQL statements and options you can use to create and run QMF queries.
- *Query Management Facility Application Development Guide for MVS*, SC26-4237, provides information for creating a QMF application in the MVS environment.
- *Query Management Facility Application Development Guide for VM/SP*, SC26-4238, provides information for creating a QMF application in the VM environment.

For Implementation with the OS/2 Licensed Program

The following manuals contain additional information that you may need for implementation with the OS/2 licensed program:

- *Operating System/2 Extended Edition User's Guide*, S01F-0285, provides information concerning the basic tasks supported by the base operating system, Communications Manager and Database Manager (including writing a query, generating a report, and creating and editing a table), and concerning more advanced tasks such as maintaining a database and creating customized interfaces.

- *Operating System/2 Extended Edition Database Manager SQL Reference*, S01F-0265, provides a reference of the SQL statements applicable to the Database Manager.
- *Operating System/2 Extended Edition Database Manager Programming Guide and Reference*, S01F-0269, provides detailed information on Database Manager function calls and on the use of SQL with the Database Manager.

Index

A

application data record 6-14
authorization 1-5

B

BREAK fields 5-15
break level definition 5-15

C

C edit code 7-11, A-2
C language sample
  callable interface 3-5
callable interface (CI)
  C language sample 3-5
  COBOL sample 3-11
  command syntax extension 3-5
  communication area
    DSQCOMM 3-3
  create variables 3-21
  defined variables 3-22
  description 3-1
  elements 3-1
  extended variable support 3-21
  reference variables 3-21
  return codes 3-4
  return variables 3-4
    command message 3-4
    query message 3-4
  variable names 3-22
CI (callable interface)
  See callable interface (CI)
COBOL sample
  callable interface 3-11
column definition 5-2
COLUMN fields 5-2
command syntax extension
  callable interface 3-5
commands
  parsing 1-7
  Query Management/400
    ERASE 1-9
    EXIT 1-10
    EXPORT 1-11
    GET 1-13
    IMPORT 1-15
    PRINT 1-18
    RUN 1-21
    SAVE DATA AS 1-23
    SET 1-25
    START 1-27
  used in a procedure 2-1
COMMENT option
  Query Management/400
    EXPORT command 1-11
    IMPORT command 1-15
  SAVE DATA AS command 1-23
comments
  in a procedure 2-1
  in a query 4-1
CONFIRM option
  ERASE command 1-9
  EXPORT command 1-11
  IMPORT command 1-15
  SAVE DATA AS command 1-23
conversion
  command example 7-12
  considerations 7-2
  example steps 7-12
  problem detection 7-4
  Query Management/400 7-1
  specifying 7-1
create variables
  callable interface 3-21
CT edit code A-2
CW edit code A-2

D

DATA
  DBCS A-1
  exporting 1-11
  saving 1-23
data set
  exporting 1-11
  importing 1-16
data types
  list of A-1
database
  capability 4-1
  names
    qualifiers 1-2
    restrictions 1-2
DATETIME option for PRINT command 1-19
DBCS A-1
defined variables
  callable interface 3-22
double-byte data
  types A-1
DSQCOMM
  callable interface communication area 3-3
DSQSCMD
  START command 1-29
DSQSMODE
  START command 1-29
DSQSRUN
  START command 1-29

E

E record 6-13
edit code
  specify option 7-5
edit codes 5-7, A-2
elements of callable interface 3-1
encoded format 6-2
  records 6-2
end-of-object record 6-13
ERASE command
  CONFIRM option 1-9
  NAME option 1-9
example
  conversion commands 7-12
EXPORT command
  CONFIRM option 1-11
  FILENAME option 1-11
  NAME option 1-11
exported data set 1-11
exported objects 6-1
exported procedure 1-11
extended variable support
  callable interface 3-21
external format 6-2

F

FILENAME option
  EXPORT command 1-11
  IMPORT command 1-15
final text definition 5-13
FINAL TEXT fields 5-13
footing definition 5-10
form
  erasing 1-9
  exporting 1-11
  importing 1-16
  printing 1-18
FORM format 6-16
FORM option
  PRINT command 1-18
format of objects 6-1
formatting terminology 5-1

G

GET command
  parameters
    value lengths 1-13
    value type 1-13
    values 1-13
    varname 1-13
    varname lengths 1-13
GET GLOBAL 1-13

H

H record 6-3
header record 6-3
heading definition 5-10

I

IMPORT command
  CONFIRM option 1-15
  FILENAME option 1-15
  NAME option 1-15
imported data set 1-16
imported procedure 1-16

K

K edit code 7-11
keywords
  See also parameters
  specifying 1-5
  values
    authorization list name 1-6
    *ALL 1-6
    *CHANGE 1-6
    *EXCLUDE 1-6
    *LIBCRTAUT 1-6
    *USE 1-6

L

LENGTH option
  PRINT command 1-19
line spacing definition 5-18
line wrapping
  what happens when not specified 1-18

N

NAME option
  ERASE command 1-9
  EXPORT command 1-11
  IMPORT command 1-15
names, database 1-2
  qualified 1-2
  qualifiers 1-2
  restrictions 1-2
NAME2 option for SAVE DATA AS command 1-23
naming conventions
  in database 1-2
number of keywords parameter
  START command 1-27
number of varnames parameter
  SET command 1-25

O

object format 6-1
objects
  inspection 7-4

objects *(continued)*
query definition (QRYDFN) 7-1
options
COMMENT
SAVE DATA AS command 1-23
CONFIRM
ERASE command 1-9
EXPORT command 1-11
IMPORT command 1-15
SAVE DATA AS command 1-23
DATETIME 1-19
PRINT command 1-19
DSQSCMD
START command 1-27, 1-29
DSQSMODE
START command 1-27, 1-29
DSQSRUN
START command 1-27, 1-29
FILENAME
EXPORT command 1-11
FORM
PRINT command 1-18
RUN command 1-21
LENGTH
PRINT command 1-19
NAME
ERASE command 1-9
EXPORT command 1-11
IMPORT command 1-15
RUN command 1-21
PAGENO 1-19
PRINT command 1-19
PRINTER
PRINT command 1-19
PROC
RUN command 1-21
QUERY
RUN command 1-21
query definition (QRYDFN) 7-4
TABLENAME
SAVE DATA AS Command 1-23
WIDTH
PRINT command 1-18
OPTIONS fields 5-18
output inspection 7-4

P

PAGE fields 5-10
page splitting
width less than print line 1-18
PAGENO option for PRINT command 1-19
panel format 6-2
parameters
keyword length
START command 1-27
keywords
START command 1-27
number of varnames
SET command 1-25
parameters *(continued)*
userval
SET command 1-25
value length
SET command 1-25
value lengths
GET command 1-13
START command 1-27
value type
GET command 1-13
SET command 1-25
START command 1-27
values
GET command 1-13
SET command 1-25
START command 1-27
varname
GET command 1-13
SET command 1-25
varname lengths
GET command 1-13
SET command 1-25
parsing of commands 1-7
PRINT command 1-18
DATETIME option 1-19
FORM option 1-18
LENGTH option 1-19
PAGENO option 1-19
PRINTER option 1-19
WIDTH option 1-18
PRINTER option
PRINT command 1-19
PROC object 1-21
exporting 1-11
importing 1-16
printing 1-18
running 1-21
procedures
erasing 1-9
how to create 2-1
printing 1-18

Q

qualified names
in database 1-2
queries
printing 1-18
query 1-21
capability 4-1
erasing 1-9
running 1-21
query definition (QRYDFN)
adding SAA 7-13
analyzing 7-3
conversion 7-1
guidelines 7-4
objects 7-1
using 7-1

Query Management Facility
  documentation H-1
Query Management/400
  documentation H-1
  relationship to SAA solution xiii

R

R record 6-11
record format rules
  for input 6-15
  for output 6-16
records in encoded format 6-2
reference variables
  callable interface 3-21
relational data queries 4-1
REPORT
  break level 5-15
  column 5-2
  final text 5-13
  footing 5-10
  heading 5-10
  options 5-18
  printing 1-18
return codes
  callable interface 3-4
return variables
  callable interface 3-4
  command message
    callable interface 3-4
  query message
    callable interface 3-4
rules for record format 6-15
RUN command 1-21
  FORM option 1-21
  NAME option 1-21
  PROC option 1-21
  QUERY option 1-21

S

SAA solution xi, xiii
SAVE DATA AS command 1-23
  NAME1 option 1-23
  NAME2 option 1-23
  tablename option 1-23
security 1-5
SET command
  parameters
    number of varnames 1-25
    userval 1-25
    value length 1-25
    value type 1-25
    values 1-25
    varname 1-25
    varname length 1-25
SQL 4-1
START command
  keywords
    DSQSCMD 1-27
START command *(continued)*
  keywords *(continued)*
    DSQSMODE 1-27
    DSQSRUN 1-27
  parameters
    keyword length 1-27
    keywords 1-27
    number of keywords 1-27
    value lengths 1-27
    value type 1-27
    values 1-27
syntax diagrams, how to read xiii
Systems Application Architecture
  in a query definition 7-13

T

T record 6-9
table description record 6-9
table row record 6-11
tablename option for SAVE DATA AS command 1-23
terms used in formatting 5-1

U

userval parameter
  SET command 1-25

V

V record 6-7
value
  lengths parameter 1-13
  type parameter 1-13
value length parameter
  SET command 1-25
  START command 1-27
value record 6-7
value type parameter
  SET command 1-25
  START command 1-27
values parameter 1-13
  GET command 1-13
  SET command 1-25
  START command 1-27
variable data
  restrictions 1-2
variable names
  callable interface 3-22
variables
  create 3-21
  defined 3-22
  extended support 3-21
  getting 1-13
  naming 3-22
  reference 3-21
varname
  getting 1-13
  lengths parameter 1-13

varname *(continued)*
setting 1-25
varname parameter 1-13
GET command 1-13
SET command 1-25

W

WIDTH option
PRINT command 1-18

Special Characters

* record 6-14

Readers' Comments

Application System/400™
Programming:
Query Management/400 Reference
Version 2

Publication No. SC41-8193-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

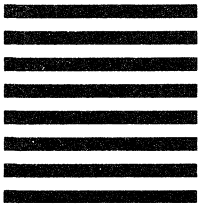
Phone No.

IBM®

Cut o
Along

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape **Please do not staple** Fold and Tape

Cut o
Along

Readers' Comments

Application System/400™
Programming:
Query Management/400 Reference
Version 2

Publication No. SC41-8193-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

Phone No.

Readers' Comments
SC41-8193-00

IBM®

Cut o
Along

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape | **Please do not staple** | Fold and Tape

Cut o
Along

Program Number: 5738-SS1

Printed in Denmark by Interprint

SC41-8193-00