

CA QLQ™ Online Query for CA IDMS™

OLQ Online Query Reference Guide

Release 18.5.00



This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2013 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At <http://ca.com/support>, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at <http://ca.com/docs>.

Contents

Chapter 1: Introduction	11
Syntax Diagram Conventions	11
Chapter 2: Command Summary	15
Default PF key assignments	15
System management commands	16
Data Retrieval Commands	17
Report formatting commands	18
Report output commands	20
Qfile commands.....	20
Table processing commands	21
Chapter 3: Entering Commands in CA OLQ	25
What Is An CA OLQ Session?	25
Signing On And Off To CA OLQ	25
Suspending an CA OLQ session.....	26
Setting Session Options.....	26
Using Commands.....	27
Chapter 4: Coding Considerations	29
Commands.....	29
Defining session characters	31
Entering Data	31
Chapter 5: Global Syntax	35
SELECT Selection Criteria	35
SELECT Comparison-Expression.....	37
FIND / GET Selection Criteria	42
FIND / GET Comparison-Expression.....	43
Expression.....	45
FIND / GET and COMPUTE Field-Reference Clause.....	46
DISPLAY and SORT Field-Reference Clause.....	48
Field-List Clause.....	50

Chapter 6: Commands and Syntax

53

BYE	53
CLEAR CURRENCY	53
CLEAR FUNCTION	53
COMPUTE	54
COMPUTE ... GROUP BY	57
DEFINE FILE	61
DEFINE PATH	66
DELETE COMPUTATION	67
DELETE QFILE	68
DELETE REPORT	68
DELETE TABLE—OLQ access mode	69
DELETE TABLE—IDMS access mode	70
DELETE USER	71
DISPLAY	72
EDIT	81
EDIT COMPUTATION	90
EDIT GROUP	93
END PATH	94
EXECUTE PATH	94
FIELDS FOR	95
FIND / GET Logical Record	97
FIND / GET MOST RECENT	101
FIND / GET PHYSICAL SEQUENTIAL	103
FIND / GET OWNER WITHIN SET	106
FIND / GET Using Storage Key	109
FIND / GET WITHIN DBKEYLIST	114
FIND / GET WITHIN Index SET	117
FIND / GET WITHIN SET	121
FIND / GET WITHIN SET Using SORTKEY	125
FUNCTION	132
HELP	132
MENU	141
OPTIONS	144
PAGE HEADER / FOOTER	155
PRINT	157
QFILE	163
SAVE QFILE	165
SAVE REPORT	166
SELECT—OLQ access mode	168
SELECT—IDMS access mode	177

SEND TABLE—OLQ access mode	178
SEND TABLE—IDMS access mode	183
SET	186
SIGNON	194
SIGNON TABLE	196
SORT	198
SUSPEND	203
SWAP	204
SWITCH.....	207
UNSORT.....	207

Chapter 7: Built-In Functions and Syntax 209

Built-In Functions	211
Invoking Built-In Functions	211
Parameters Of Built-In Functions	220
Absolute Value.....	220
Arc Cosine.....	221
Arc Sine	222
Arc Tangent	223
Average	223
Capitalization	224
Concatenate	225
Cosine.....	226
Count	227
Date Change	227
Date Difference.....	230
Date Offset	230
Extract	231
Fix.....	232
Index	233
Initial Uppercase.....	234
Insert.....	235
Invert Sign.....	236
Left Justify.....	237
Length	238
Logarithm.....	239
Lowercase.....	239
Maximum.....	240
Minimum	241
Modulo.....	242
Next Integer Equal or Higher.....	242

Next Integer Equal or Lower	243
Product.....	244
Random Number	245
Right Justify	246
Sign Value.....	246
Sine	247
Square Root.....	248
Standard deviation.....	249
Standard deviation population.....	250
Substring.....	250
Sum	251
Tangent	252
Today.....	253
Tomorrow	255
Translate	256
Uppercase.....	257
Variance.....	258
Variance population.....	259
Verify	260
Weekday	261
Yesterday	263

Chapter 8: Tailoring the CA OLQ Environment for Ease of Use **265**

What This Chapter Is About	265
Data Retrieval	265
SELECT (IDMS access mode).....	266
SELECT (OLQ access mode).....	266
Logical Records.....	267
Using qfiles	268
Building qfiles.....	269
Executing qfiles.....	269
Reporting on qfiles.....	270
Special Uses of qfiles	270
Including Parameters in qfiles.....	272
Defining Report Headers.....	277
Synonyms.....	277
Code Tables	278
Date Option	279
External Pictures.....	280

Chapter 9: Using CA OLQ Efficiently 281

Controlling Data Retrieval.....	281
Qfiles.....	281
Logical records.....	282
OLQ DML User Exit.....	282
Interrupt count.....	288
Using db-keys for retrieval.....	289
Controlling Resource Consumption.....	290
Sorts.....	291
Saved reports.....	292
Db-key list.....	294
Saving qfiles.....	295

Chapter 10: Security 297

Assigning Authority to Access CA OLQ.....	297
Limiting Access through Central Security.....	297
Initiating CA OLQ Dictionary Security.....	298
Securing User Access to CA OLQ Components.....	298
Subschema Access.....	298
Qfile Access.....	299
Securing Retrieval Interruption.....	300
Using LRF to Secure The Database.....	301
Security for ASF tables.....	301
Security for Saved Reports.....	302

Chapter 11: Batch Processing 303

JCL for z/OS and CMS Commands for OLQBATCH.....	305
z/OS JCL (central version).....	306
z/OS JCL (local mode).....	307
z/OS Local Mode Considerations.....	309
CMS commands (central version).....	310
Usage.....	311
CMS commands (local mode).....	313
z/VSE JCL (central version).....	313
z/VSE JCL (local mode).....	315
IDMSLBLS Procedure.....	317
Setting Defaults for Batch Processing.....	324
Defining files.....	325
Signing on in batch.....	327
OLQBatch Notification.....	328

OLQBNOTE Example for z/OS.....	328
OLQBNOTE example for CMS.....	331
OLQBNOTE example for z/VSE.....	333
Batch Class Specification.....	336
Operating System Dependent Installation Instructions	336
Examples of Batch.....	338
Creating Multiple Reports in One Job	339
Wide Reports	340
Creating a Report with SELECT	340
Writing to a Disk File with SELECT	342
Chapter 12: Setting Defaults	345
System Generation Options	345
Integrated Data Dictionary Options.....	349
Index	355

Chapter 1: Introduction

CA OLQ is a query tool and report writer used to retrieve information from a CA IDMS/DB database or other external files.

With CA OLQ command mode, you can do the following:

- Set up the CA OLQ environment to suit your needs (system management commands)
- Retrieve information from a CA IDMS/DB database
- Build report files based on information retrieved (report output commands)
- Perform sorts, computations, and functions on report files (report formatting commands and built-in functions)
- Create and save sequences of CA OLQ commands for regular use (qfile commands)
Note: A sequence of CA OLQ commands saved in the data dictionary is called a query file and referred to in this manual as a qfile. In some earlier releases, qfiles were referred to as express routines in menu mode.
- Create and retrieve from stored tables (table processing commands)
- Run jobs in batch mode (CA OLQ batch commands)

Syntax Diagram Conventions

The syntax diagrams presented in this guide use the following notation conventions:

UPPERCASE OR SPECIAL CHARACTERS

Represents a required keyword, partial keyword, character, or symbol that must be entered completely as shown.

Lowercase

Represents an optional keyword or partial keyword that, if used, must be entered completely as shown.

italicized lowercase

Represents a value that you supply.

Lowercase bold

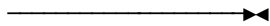
Represents a portion of the syntax shown in greater detail at the end of the syntax or elsewhere in the document.

←

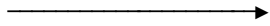
Points to the default in a list of choices.

▶—————

Indicates the beginning of a complete piece of syntax.



Indicates the end of a complete piece of syntax.



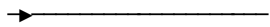
Indicates that the syntax continues on the next line.



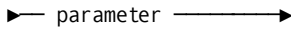
Indicates that the syntax continues on this line.



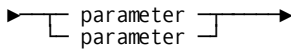
Indicates that the parameter continues on the next line.



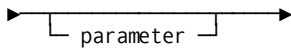
Indicates that a parameter continues on this line.



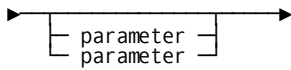
Indicates a required parameter.



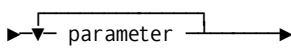
Indicates a choice of required parameters. You must select one.



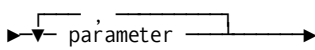
Indicates an optional parameter.



Indicates a choice of optional parameters. Select one or none.



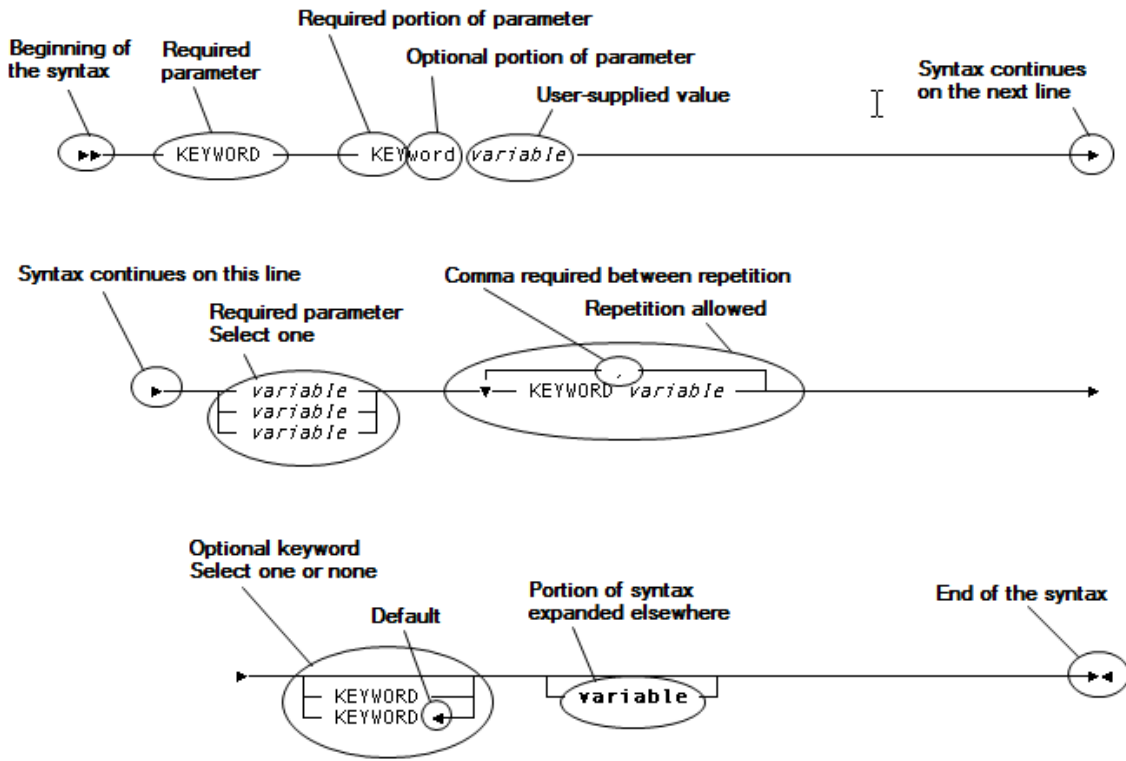
Indicates that you can repeat the parameter or specify more than one parameter.



Indicates that you must enter a comma between repetitions of the parameter.

Sample Syntax Diagram

The following sample explains how the notation conventions are used:



Chapter 2: Command Summary

Default PF key assignments

The PF key assignments as defined at system installation are presented below.

PF Key	Function
[PF1]	Help
[PF2]	Help commands
[PF3]	Bye
[PF4]	Display help
[PF5]	Display
[PF6]	Menu
[PF7]	Page backward
[PF8]	Page forward
[PF9]	Swap
[PF10]	Display left
[PF11]	Display right
[PF12]	Print
[Clear]	Undefined
[Enter]	Process input
[PA1]	Undefined
[PA2]	Undefined

System management commands

You can govern the general use of CA OLQ with these system management commands and tailor the CA OLQ session environment to suit your needs.

The following table summarizes the system management commands available.

Use ...	Or ...	To ...
BYE	EXIT, GOODBYE, OFF, QUIT, SIGNEDOFF	Terminate a CA OLQ session.
CLEAR CURRENCY		Release all database currencies that have been established by a CA OLQ session.
CLEAR FUNCTION		Clear control key functions in command mode CA OLQ.
DELETE USER		Delete the report directory associated with a particular user ID.
FUNCTION		Invoke control key functions in command mode CA OLQ.
HELP	SHOW	Display how to use CA OLQ commands and provide information about the data the current subschema can access.
MENU		Switch between CA OLQ command mode and a specific screen of the menu facility.
OPTIONS		Set default parameters for a session.
SET		Set system management parameters.
SIGNON		Initiate a CA OLQ session.
SUSPEND		Suspend the current session and return control to the transfer control facility or CA IDMS/DC or CA IDMS UCF.

Use ...	Or ...	To ...
SWAP		Switch from CA OLQ command mode to the menu facility.
SWITCH		Pass control to another CA IDMS/DC product.

For more information:

[Commands and Syntax](#) (see page 53)

Data Retrieval Commands

You can retrieve data from the database with these data retrieval commands. The commands available for data retrieval are presented in the table below. For the syntax and syntax rules of these commands, see the alphabetical listing of commands in [num=6.Commands and Syntax](#) (see page 53).

Use ...	To ...
DEFINE PATH	Place CA OLQ in database path definition mode.
END PATH	Terminate path definition mode.
EXECUTE PATH	Execute the retrieval commands specified in the database path definition and build a report file of retrieved records.
FIND/GET logical record	Retrieve records by using DBA-defined paths through the database.
FIND/GET MOST RECENT	Retrieve the current of record type for the specified record name.
FIND/GET OWNER WITHIN SET	Retrieve the owner of a database set occurrence.
FIND/GET PHYSICAL SEQUENTIAL	Retrieve records based on their physical position in a database area.
FIND/GET using STORAGE KEY	Retrieve records based on their CALC-key or database-key value.
FIND/GET WITHIN DBKEYLIST	Retrieve records based on the results of previous retrieval commands.

Use ...	To ...
FIND/GET WITHIN index SET	Retrieve records by using the name of an index set and the index-sort-key fields specified in the WHERE clause.
FIND/GET WITHIN SET	Retrieve records based on their membership in a database set.
FIND/GET WITHIN SET using SORTKEY	Retrieve member records in sorted database sets based on a specified sort key.
REPEAT for each of the above FIND/GET commands	Duplicate an immediately preceding FIND/GET command.
SELECT	Retrieve information using the SELECT command.
SHOW PATH	Display the current path.

For more information:

[Commands and Syntax](#) (see page 53)

Report formatting commands

You can specify the display format of reports containing data retrieved by CA OLQ commands. The following table presents the commands available for formatting reports in CA OLQ.

Use ...	To ...
COMPUTE	Perform computations on fields in a report file by using: <ul style="list-style-type: none">■ Arithmetic expressionsBuilt-in functions
COMPUTE ... GROUP BY	Perform summary computations.
DELETE COMPUTATION	Delete computed fields.

Use ...	To ...
EDIT	<p>Edit a field for display by specifying:</p> <ul style="list-style-type: none"> ■ Edit characteristics, such as hexadecimal display, lead zeros, commas, a specific external picture, and a code table translation ■ A report heading ■ Sparse, to suppress the display of repeating column values <p>The alignment of a column</p>
EDIT COMPUTATION	<p>Edit a computed field for display by specifying:</p> <ul style="list-style-type: none"> ■ Edit characteristics, such as hexadecimal display, lead zeros, commas, a specific external picture, and a code table translation ■ A report heading ■ Sparse, to suppress the display of repeating column values ■ The alignment of a column <p>The column under which a computed field is displayed</p>
EDIT GROUP BY	<p>Edit the group defined by the COMPUTE...GROUP BY command. Use this command to change the level number of a group, specify spacing between groupings or specify the separator character which separates the grouping from the computed value.</p>
ON BREAK	<p>Display computed values at designated points within the report file. This command is provided for compatibility with prior releases but its use is discouraged. Use COMPUTE...GROUP BY instead.</p>
PAGE HEADER/FOOTER	<p>Include a user-specified page header or footer in a report.</p>
SORT	<p>Request that records within a report file be ordered by user-specified order criteria.</p>
UNSORT	<p>Return the report file to the original retrieval sequence following one or more SORT commands.</p>

For more information:

[Commands and Syntax](#) (see page 53)

Report output commands

You can save, display, print, and delete **report files** in CA OLQ command mode with report output commands.

The report output commands available are presented in the table below.

Use ...	To ...
DELETE REPORT	<ul style="list-style-type: none">■ Delete the report specified■ Delete all reports saved under a specified user name
DISPLAY	Direct CA OLQ to send a page of report file data to the user's terminal.
PRINT	Direct a formatted CA OLQ report to a specific printer for a hard copy.
SAVE REPORT	Associate a name with a report file and save it in the user's directory for future use.
SHOW DIRECTORY	List the reports available for the specified user.

For more information:

[Commands and Syntax](#) (see page 53)

Qfile commands

A **qfile**, like a path, is a sequence of commands used to build online reports. Unlike paths, qfiles are stored in the data dictionary and can contain any CA OLQ command, not just retrieval commands. With qfiles you can set up defaults for the CA OLQ operating environment, as well as construct reports.

Use the qfile commands presented in the table below to create, save, and execute qfiles in CA OLQ command mode.

Use ...	To ...
DELETE QFILE	Delete the named qfile.
QFILE	Execute the named qfile.
SAVE QFILE	Save the current path and report formatting commands as the named qfile.

Use ...	To ...
SHOW QFILE=	List the commands in the named qfile.
SHOW QFILES	List the qfiles available.

For more information:

[Commands and Syntax](#) (see page 53)

Table processing commands

You can use data table processing commands to maintain information in either ASF or SQL tables.

ASF tables:

ASF tables refers to tables associated with the IDMSR schema.

In order to use ASF tables for the session, you must set the access switch to **olq**.

Note: For more information about ASF tables, see the *CA IDMS ASF User Guide*.

SQL tables:

SQL tables refers to tables associated with an SQL schema.

In order to use SQL tables for the session, you must set the access switch to **idms**.

Note: For more information about SQL tables, see the *CA IDMS SQL Quick Reference Guide*.

How to specify the access switch:

The access switch can be set in the following ways:

1. At system generation time
2. For an individual user in the Dictionary (IDD)
3. For the session, interactively (or until the switch is changed)

Note: For more information about setting the access switch, see [SET](#) (see page 186).

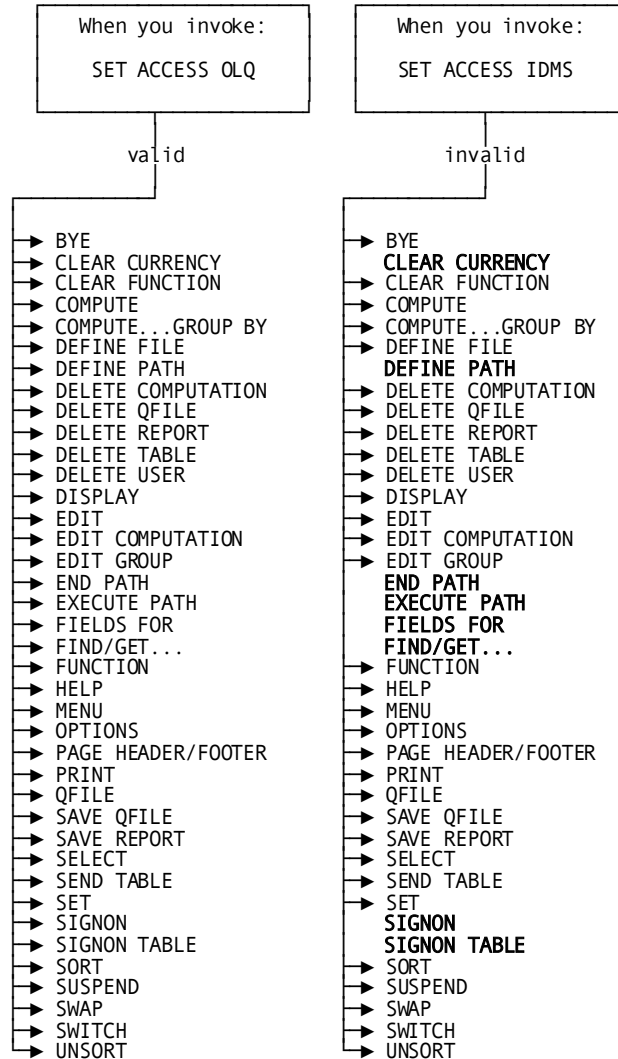
Table processing commands:

The following table lists the CA OLQ table processing commands:

Use this command	To do this
DELETE TABLE	Delete SQL and ASF tables.
SELECT	Retrieve specific information from SQL and ASF tables, logical and database records, and sequential files (batch only).
SEND TABLE	Store information from the current or named report file in SQL and ASF tables.
SIGNON TABLE	Access a specific ASF table to increase efficiency. This command is only relevant in OLQ access mode.
HELP TABLES	List tables saved by the current or named user.

Invalid OLQ commands:

The following figure shows which OLQ commands become invalid when you set the access switch to IDMS access mode:



Note: For the syntax and syntax rules of CA IDMS SQL commands, see the *CA IDMS SQL Quick Reference Guide*.

For more information:

[Commands and Syntax](#) (see page 53)

Chapter 3: Entering Commands in CA OLQ

This section contains the following topics:

[What Is An CA OLQ Session?](#) (see page 25)

[Signing On And Off To CA OLQ](#) (see page 25)

[Suspending an CA OLQ session](#) (see page 26)

[Setting Session Options](#) (see page 26)

[Using Commands](#) (see page 27)

What Is An CA OLQ Session?

A CA OLQ session is the interaction between you and CA OLQ. The session begins when you sign on to CA OLQ and ends when you sign off from CA OLQ.

Signing On And Off To CA OLQ

Signing on:

To sign on to CA OLQ, first sign on to the system, then type **olq** beneath the system prompt:

```
ENTER NEXT TASK CODE:
```

```
olq
```

The system responds with the CA OLQ prompt for entering commands in command mode:

```
OLQ 091057 00 Please enter next command
```

This prompt is normally positioned on the fifth line of the screen. You enter your commands on the **command line** (the first four lines on the screen). You can set the number of lines dedicated to entering commands during system generation.

Signing off:

To sign off from CA OLQ, enter BYE on the command line:

bye

OLQ 091057 00 Please enter next command

You are then returned to the system prompt:

ENTER NEXT TASK CODE

Suspending an CA OLQ session

Suspending a session does not end it. When you return to a session after suspending it, any session options you previously set will still be in effect.

Note: You can use the SUSPEND command to suspend a current session and return control to the transfer control facility or CA IDMS/DC or CA IDMS UCF.

Setting Session Options

Tailoring your CA OLQ session

CA OLQ **session options** allow you to set up your CA OLQ environment to suit your needs.

Example

Among other options, you can specify the use of:

- **Echo/No echo** —User-entered commands are repeated by CA OLQ on the 3270-type device.
- **Full/sparse**—The display format for path retrieval report lines does or doesn't suppress repeating column values.
- **Header/No header** —The report file built is or isn't displayed with a header line.

Session options remain set for only one session. At the next signon (after the termination of a session) all the session options are returned to each user's defaults.

Note: For more information about setting session options, see the `OPTIONS=` command in [num=6.Commands and Syntax](#) (see page 53).

For more information:

[Commands and Syntax](#) (see page 53)

Using Commands

When you see the CA OLQ prompt:

```
OLQ 091057 00 Please enter next command
```

The cursor is positioned on the top line of the screen. The top line of the screen is where you begin to type your commands. You can enter as many commands on the screen as you want, using up to the default of four lines (unless you have set a larger line size during system generation), provided you follow these rules:

- Use **command separators** to separate commands. The default command separator is the exclamation point (!).
- If the command exceeds the length allowed, use the **continuation character** to extend the command beyond the pseudo-converse. The default continuation character is the hyphen (-). You can also change the length of the command lines so you can enter longer commands.

For more information:

[Coding Considerations](#) (see page 29)

Chapter 4: Coding Considerations

When you enter commands in CA OLQ command mode, keep in mind the following considerations as described in this chapter.

Commands

- Using abbreviations
- Ending command strings
- Issuing multiple commands per line
- Invoking function keys

Defining session characters

- Defining comment characters
- Defining separator characters

Entering data

- Specifying data values
- Specifying subscripts

Commands

Using abbreviations:

Three-letter abbreviations are valid for most commands and keywords.

Ending command strings:

You can end a command string with a **comment character** or a **separator character**.

Comment characters signal to CA OLQ that everything following is to be ignored. You can use comment characters to document qfiles. Use separator characters for separating commands in the input field. Even when commands are one to a line in the input field, they must be separated by a separator character.

Issuing multiple commands:

You can specify multiple commands in a single pseudo-converse by using a **separator character**. The separator character is defined at system generation as the exclamation point (!).

Example:

```
signon ss empss01! select * from employee
```

Use with SELECT (IDMS mode)

Anytime you use separators with SELECT (IDMS mode) they *must* precede the SELECT statement.

For instance, the following is valid syntax:

```
delete table employee.job!select all from emp_id
```

However, CA IDMS/DB does *not* accept the syntax below because the separator (!) comes *after* the SELECT statement:

```
select all from emp_id!delete table employee.job
```

Invoking function keys from the command line:

To invoke a predefined function from a terminal that does not have function keys, the user can enter the following commands on the command line:

PF pf-key-number or

PA pa-key-number

Pf-key-number

Specifies any number in the range 1 through 99; *pa-key-number* must be either 1 or 2.

Continuation character:

You can continue commands across pseudo-converses with a continuation character. You can also use the continuation character to continue batch or qfile commands on a following line.

For more information:

[Commands and Syntax](#) (see page 53)

Defining session characters

Defining the comment character:

The default comment character is the semi-colon (;), as defined at system generation. You can change the definition at any time in command mode CA OLQ by issuing a **SET COMMENT CHARACTER** command.

Access mode: You cannot use comment characters with SELECT (IDMS mode) statements.

Defining the separator character:

You can change the definition of the separator character with the **SET SEPARATOR CHARACTER** command.

Defining the continuation character:

You can change the definition of the continuation character with the **SET CONTINUATION CHARACTER** command.

For more information:

[Commands and Syntax](#) (see page 53)

Entering Data

The following input considerations should be noted:

- **Commas as decimal points**— Any numeric value that contains an embedded comma can be interpreted as a decimal number. The ability to recognize commas as decimal points is enabled at system generation. For more information about international options, see [Integrated Data Dictionary Options](#) (see page 349).
- **Record names and CA OLQ keywords**— If a database record name is the same as a CA OLQ keyword, the name should be enclosed in quotation marks.

Specifying data values:

CA OLQ commands sometimes require user-supplied data values, such as a CALC key, sort key, database key, or user-defined item. The following types of data values are recognized by CA OLQ:

- **Integer**— A whole number in the range -32767 through +32767; for example, 2220.
- **Binary**— A binary number; for example, B'0110'.

- **Real number**—A decimal number; for example, 12.23.
- **Hexadecimal number**—A string of up to 64 hexadecimal digits enclosed in single or double quotation marks and preceded by an X. An even number of digits must be specified; for example, X'00AFD6'.
- **Simple character string**—A 1- to 64-character alphanumeric value that can consist of alphabetic letters, numeric digits, and special characters. Special characters include the at sign (@), dollar sign (\$), pound sign (#), and hyphen (-); for example, @DEPT-NAME or EMP-#.
- **Complex character string**—A 1- to 256-character alphanumeric value enclosed in single or double quotation marks. If a quotation mark appears within a string, it must appear twice or be distinguishable from the enclosing quotation marks. The following examples illustrate acceptable formats for specifying quotation marks within complex character strings:

```
"LEE'S"  
'LEE"S'  
'LEE' 'S'
```

Note: Enclosing quotation marks are not evaluated by the CA OLQ command processor. If the value of a complex character string matches a field name specified in the same command, the results are unpredictable.

- **Graphics literal**—A string of up to 32 double-byte characters string (DBCS) characters enclosed with shiftstrings and single quotation marks and prefixed by a G.

Example:

```
G'<DBCS characters>'
```

A mix of EBCDIC and DBCS characters is allowed as long as the correct shiftin/shiftout sequences identifies the code set.

Example:

```
'ABC<DBCS characters>UW'
```

- **Group value**—A series of values specified in any of the forms described above, separated by commas or blanks. Group values cannot exceed 64 characters in length. Group values can be enclosed in parentheses; for example, ('8 Cedar St','Worcester, MA',02312).
- **Database key**—A decimal page and line number that define the actual location of the record in the database; For example, 10023:14, where 10023 is the database page number and 14 is the line number of the specified record.

- **Floating point constant**—A representation of a real number as a fixed-point value with an exponent. Floating point constants are entered as follows:

- *value E+/-integer-value*

The following examples illustrate typical floating point constants:

1.2E-22
-3E2

Numeric and alphanumeric character strings for database fields are interpreted according to their formats in the subschema view. The user need not supply leading zeros or trailing blanks; CA OLQ automatically pads values, as follows:

- **Alphanumeric**— Left-justified, padded with blanks
- **Numeric**— Decimal-aligned, sign-extended (default is plus), zero-filled to the right of the decimal

Specifying subscripts

Subscripts are used to identify specific field entries in an array.

Example:

If a record contains sales figures for two divisions of a company for each quarter in the last five years, the array of values is as follows:

Year 1	Quarter 1	Div-sales 1
		Div-sales 2
	Quarter 2	Div-sales 1
		Div-sales 2
	Quarter 3	Div-sales 1
		Div-sales 2
	Quarter 4	Div-sales 1
		Div-sales 2
Year 2	Quarter 1	Div-sales 1
		Div-sales 2
	Quarter 2	Div-sales 1
		Div-sales 2
	Quarter 3	Div-sales 1
		Div-sales 2
	Quarter 4	Div-sales 1
		Div-sales 2
	.	
	.	
	.	

In the above example, two subscripts must be specified to reference a specific quarter.

Example:

QUARTER(1,4) selects the fourth quarter of the first year.

Three subscripts must be specified to reference a specific division in a specific quarter.

Example:

DIV-SALES(2,3,1) selects the third quarter sales for the first division in year 2 of the array.

Chapter 5: Global Syntax

This chapter presents syntax and rules for the following global syntax:

- **Selection criteria** used with:
 - SELECT command WHERE clause and HAVING clause
 - FIND/GET command WHERE clause and COMPUTE command HAVING clause
- **Expression**
- **Field-reference** used with:
 - WHERE clause and COMPUTE command
 - DISPLAY and SORT commands
- **Field-list**

SELECT Selection Criteria

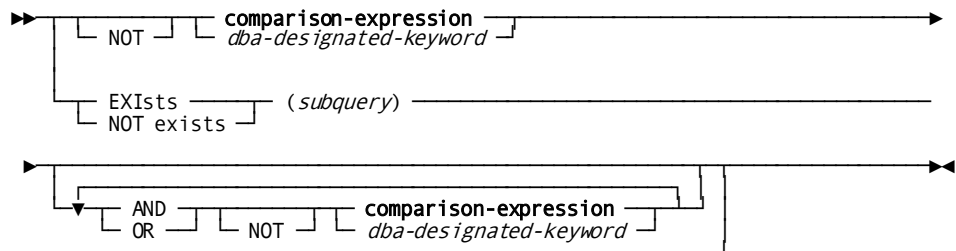
The **WHERE** clause criteria specifies criteria for selecting record occurrences based on field values of a named database record, logical record, or ASF table.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

This selection criteria is an expansion of the:

- SELECT WHERE *criteria*
- SELECT HAVING *criteria*

Syntax:



Syntax rules:

comparison-expression

Specifies a comparison operation to be performed using the named operands and operator.

Note: Complete syntax rules for *comparison-expression* are discussed later in this chapter.

dba-designated-keyword

Specifies a logical record keyword that is predefined by the DBA. *DBA-designated-keyword* is a keyword that applies to the logical record named in the command. The keyword represents an operation to be performed at the logical record path level and serves only to route the logical record request to the appropriate path; it has no meaning to CA OLQ.

When a SHOW LOGICAL RECORDS command is issued, CA OLQ lists the keywords associated with each logical record defined in the current subschema.

Note: This parameter is used in the WHERE clause only.

EXISTS/NOT EXISTS (subquery)

Evaluates the outcome of *subquery* in terms of whether it is true (EXISTS) or false (NOT EXISTS).

Subquery is a nested SELECT statement. The SELECT statement must be enclosed in parentheses. The column list of SELECT statements containing the EXISTS or NOT EXISTS operands must be an asterisk (*), specifying all columns. The subselect statement cannot contain:

- GROUP BY clauses
- HAVING clauses
- DISTINCT option
- ORDER BY clauses
- UNION clauses

Note: This parameter is used in the WHERE clause only.

AND/OR/NOT

Names the logical operators to be used in evaluating the WHERE clause.

Table 1 lists the logical operators and their meanings in order of precedence. NOT has the highest precedence, followed by AND, then OR. Parentheses can be used to force the order of evaluation.

Table 1: Logical Operators

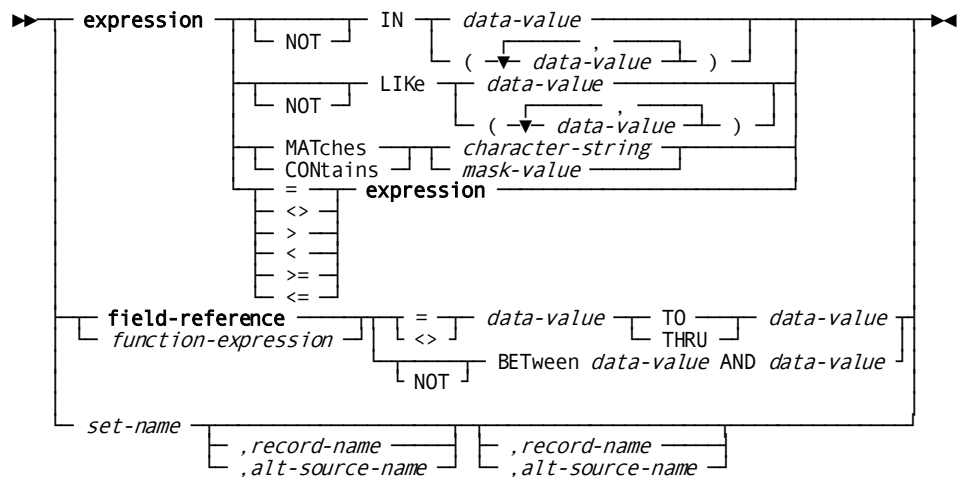
Operator	Example	Meaning
NOT	NOT comparison	If comparison is false, expression is true.
AND	Comparison A AND comparison B	If comparison A and comparison B are both true, expression is true.
OR	Comparison A OR comparison B	If either comparison A or comparison B is true, expression is true.

SELECT Comparison-Expression

Comparison-Expression is used in the SELECT WHERE *criteria* clause and in the SELECT HAVING *criteria* clause.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

Syntax:



Syntax rules:

expression

Specifies a series of constants or variables separated by operators that yields a single value.

IN data-value

Compares an expression to a data value or a list of data values:

- **IN**— Matches an expression to a list of one or more data values. The comparison is true if the expression matches one or more of the data values.

The IN predicate is equivalent to coding a series of OR expressions. For example:

```
select * from employee
where emp-last-name in ('jones', 'tanaka', 'anderson')
```

is equivalent to:

```
select * from employee
where emp-last-name = 'jones' or
      emp-last-name = 'tanaka' or
      emp-last-name = 'anderson'
```

- **NOT IN**— Compares a column expression to a list of one or more data values. The comparison is true if the column expression does not match any of the data values.

The NOT IN predicate is equivalent to coding a series of AND expressions. For example:

```
select * from employee
where emp-last-name not in ('jones', 'tanaka', 'anderson')
```

is equivalent to:

```
select * from employee
where emp-last-name <> 'jones' and
      emp-last-name <> 'tanaka' and
      emp-last-name <> 'anderson'
```

- *data-value*— The data value or list of data values (*data-value,...*) to which the expression is compared. Each data value must be enclosed in quotation marks. If more than one data value is specified, the list must be enclosed in parentheses and the data values separated by commas.

LIKE data-value

Searches the expression for a data value.

- **LIKE**— Determines whether an expression contains a data value. The comparison is true if the column expression contains the data value.
- **NOT LIKE**— Determines whether an expression does not contain a data value. The comparison is true if the expression does not contain the data value.

- *Data-value*— The data value to which the expression is compared. The data value can contain:
 - Alphanumeric characters for an exact match
 - Special characters to use as wild cards
 - Escape characters to exactly match the special characters

Object String	Data Value	Example of Syntax	Example of True comparison
Underscore (_)	Any single character	NAME LIKE 'S _ _'	True if NAME is exactly 3 characters long and the first character is S
Percent sign (%)	Any sequence of zero or more characters	NAME LIKE '%C _ _'	True if NAME is 3 or more characters long AND the third from last character is C
Single alphanumeric character	Exact match to that alphanumeric character	NAME LIKE 'MAC'	True if NAME is MAC
Escape character + underscore (_)	Exact match to the underscore (_)	PARTNUM LIKE '* _ 115' ESCAPE '* _ 1'	True if PARTNUM is ' _ 115'
Escape character + percent sign (%)	Exact match to the percent sign (%)	PARTNUM LIKE '*%15' ESCAPE '* _ 1'	True if PARTNUM is '%15'
Escape character alone	Exact match to the escape character	PARTNUM LIKE '****' ESCAPE '* _ 1' (note below)	True if PARTNUM is '**'

The escape character can be any single alphanumeric character and is set by specifying ESCAPE '*escape-character*' in your SELECT statement.

MATCHES/CONTAINS

Specifies search conditions as follows:

- **MATCHES**— A *character-string* or *mask-value* against which the named field is to be evaluated, character by character. The match must be exact, starting with the first character in the mask. The special characters that can be used for the mask are:
 - Asterisk (*) specifies any character. If an asterisk is specified, the entire **MATCHES** string must be enclosed in quotation marks.
 - At sign (@) specifies any alphabetic character.
 - Pound sign (#) specifies any numeric character.

If you specify any other character, the match is for that character only. Only the left-most significant characters of the mask need be specified explicitly when the remaining characters in the field are allowed to have any value. For example, to retrieve all addresses where the first two digits of the zip code are 02, the mask value can be specified as follows:

'02'

Unspecified mask characters are treated as if any character were specified. However, if you want to test the zip code field for numeric values only, the mask must be specified as '02###'. If the specified mask value is longer than the field being checked, the extra mask characters are ignored.

- **CONTAINS**— Specifies a character string or mask value that you want to search for. The **CONTAINS** value can appear anywhere in the named field. For example, the character string *EL* appears in *FIELD*, in *ELEMENT*, and in *COMPEL*.

Note: **MATCHES** and **CONTAINS** apply only to fields with a usage of **DISPLAY** and do not allow values that contain double-byte string characters.

= <> > < >= <=

Specifies the comparison operator:

- = means Equal to
- <> means Not equal to
- > means Greater than
- < means Less than
- >= means Greater than or equal to
- <= means Less than or equal to

expression

Specifies the expression the named condition is compared to.

Note: See the expansion of [Expression](#) (see page 45), later in this chapter.

field-reference

Identifies a field.

Note: For more information, see the expansion of [FIND / GET and COMPUTE Field-Reference Clause](#) (see page 46), later in this chapter.

function-expression

An expression containing a built-in function. Note that built-in functions can be nested.

= <> data-value

Data-value represents data values to which the named field is compared. Specifies that a column expression or column name equals (=) or doesn't equal (<>) the specified data value.

TO/THRU data-value

Specifies a range of data values to which the named field is compared. THRU indicates an inclusive range. TO indicates an exclusive range.

BETWEEN data-value and data-value

Specifies a range of data values to which the named field is compared. BETWEEN indicates that the named field meets the requirements inclusive of the boundaries specified by *data-value* AND *data-value*. NOT BETWEEN indicates that the named field doesn't meet the requirements inclusive of the boundaries specified by *data-value* AND *data-value*.

set-name

Identifies a set relationship. This sub-clause is valid in the WHERE clause only. *Set-name* is required.

record-name

Is the name of either the owner or member record of the set. Either the owner or member record can be specified or omitted.

alt-source-name

Specifies an alternative name used to identify records with the same name. *Alt-source-name* is a 1- to 8-character alphanumeric literal.

For more information:

[Coding Considerations](#) (see page 29)

FIND / GET Selection Criteria

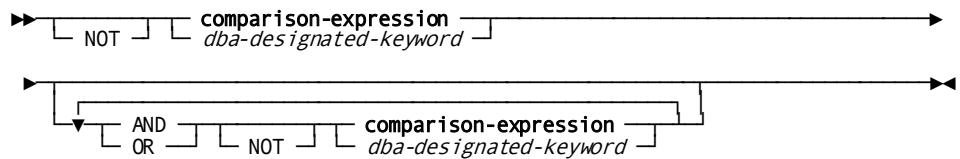
The **WHERE** clause selection criteria specifies criteria for selecting record occurrences based on field values of a named database record or logical record.

Access mode: This criteria clause is **invalid** when the access switch is set to **IDMS**.

This selection criteria is an expansion of the:

- FIND/GET WHERE *criteria*
- COMPUTE GROUP BY HAVING *criteria*

Syntax:



Syntax rules:

comparison-expression

Specifies a comparison operation to be performed using the indicated operands and operator.

Note: Complete syntax rules for *comparison-expression* are presented later in this chapter.

dba-designated-keyword

Specifies a logical record keyword that is predefined by the DBA. *Dba-designated-keyword* is a keyword that applies to the logical record named in the command. The keyword represents an operation to be performed at the logical record path level and serves only to route the logical record request to the appropriate path; it has no meaning to CA OLQ.

When a SHOW LOGICAL RECORDS command is issued, CA OLQ lists the keywords associated with each logical record defined in the current subschema.

Note: This parameter is used in the WHERE clause only.

AND/OR/NOT

Specifies the logical operators to be used in evaluating the WHERE clause.

Table 2 lists the logical operators and their meanings in the WHERE clause. Logical operators are evaluated from left to right in order of precedence; NOT has the highest precedence, followed by AND, then OR. You can use parentheses to force the order of evaluation.

Table 2: Logical Operators:

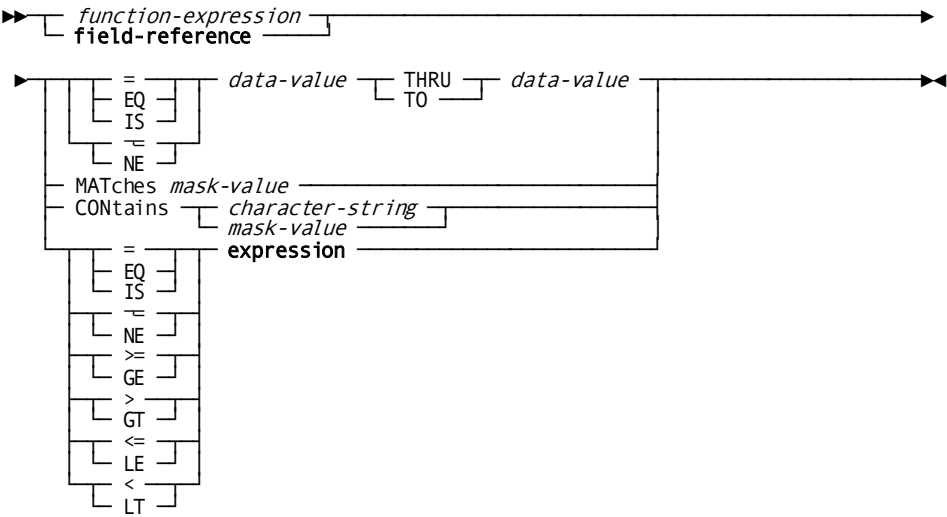
Operator	Example	Meaning
NOT	NOT comparison	If comparison is false, the expression is true.
AND	Comparison A AND comparison B	If comparison A and comparison B are both true, the expression is true.
OR	Comparison A OR comparison B	If either comparison A or comparison B is true, the expression is true.

FIND / GET Comparison-Expression

Comparison-expression is used in the FIND/GET WHERE criteria clause and in the COMPUTE GROUP BY HAVING criteria clause.

Access mode: The syntax below is invalid when the access switch is set to IDMS.

Syntax:



Syntax rules:

function-expression

Allows you to use built-in aggregate functions to evaluate data. You can nest built-in functions.

field-reference

Note: For more information, see [FIND / GET and COMPUTE Field-Reference Clause](#) (see page 46), later in this chapter.

= **→** **data-value**

Specifies a range of data values to which the named field is compared.

THRU/TO **data-value**

Specifies a range of data values to which the named field is compared.

THRU indicates an inclusive range; TO indicates an exclusive range.

MATCHES **mask-value**

Specifies a mask value against which the named field is evaluated, character by character. The match is exact, starting with the first character in the mask. The following characters are available for use in the MATCHES clause:

- Asterisk (*) specifies any character. If an asterisk is specified, the entire MATCHES string must be enclosed in quotation marks.
- At sign (@) specifies any alphabetic character.
- Pound sign (#) specifies any numeric character.
- Any alphanumeric character — Specifies a match against itself.

CONTAINS

Specifies a character string or mask value for which the named field is searched. The CONTAINS value can appear anywhere within the named field. For example, the character string *EL* appears in *FIELD*, in *ELEMENT*, and in *COMPEL*.

- *character-string*—An exact sequence of characters for which the named field is searched.
- *mask-value*—A combination of specific (for example: T, 22) and general (for example, any numeric digit) characters for which the named field is to be searched. The special mask characters described above for MATCHES also apply to CONTAINS.

Note: MATCHES and CONTAINS apply only to fields with a usage of display and do not allow values that contain double-byte character strings.

= -= >= > <= < **expression**

Specifies the comparison operator with which the named field is compared.

Note: For more information, see the expansion of *expression* (see page 45) later in this chapter.

For more information:

[Coding Considerations](#) (see page 29)

[Built-In Functions and Syntax](#) (see page 209)

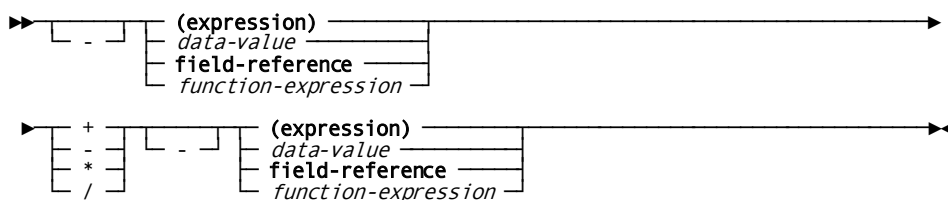
Expression

Expression is used in the WHERE *criteria* clause and in the COMPUTE GROUP BY HAVING *criteria* clause. *Expression* can include fields of the following data types:

- **Doubleword binary**— COMP PIC S9(16)
- **Fullword binary**— COMP PIC S9(8)
- **Halfword binary**— COMP PIC S9(4)
- **Packed decimal**— COMP-3
- **Zoned decimal**— DISPLAY PIC 9(*n*) or PIC S9(*n*)
- **Floating point**— COMP-1 and COMP-2
- **Display**— DISPLAY PICX(*n*)

Expressions that include bit fields, nonnumeric fields, or nonnumeric constants are flagged as errors.

Syntax:



Syntax rules:

-

Minus sign denotes a negative value in the expression.

(expression)

Allows you to nest expressions. Parentheses override the standard order of precedence.

data-value

field-reference

Note: For more information, see [FIND / GET and COMPUTE Field-Reference Clause](#) (see page 46) and [DISPLAY and SORT Field-Reference Clause](#) (see page 48), later in this chapter.

function-expression

Allows you to use built-in aggregate functions to evaluate data. You can nest built-in functions.

+ - * /

Specify the arithmetic operation to be performed, as follows:

- Plus sign (+) means addition
- Minus sign (-) means subtraction (the minus sign must be surrounded by blank spaces)
- Asterisk (*) means multiplication
- Slash (/) means division

When evaluating expressions, CA OLQ observes the standard order of precedence: multiplication, division, addition, and subtraction, from left to right, with operations in parentheses resolved first.

For more information:

[Coding Considerations](#) (see page 29)

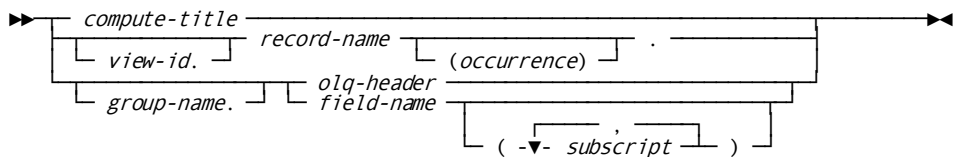
[Built-In Functions and Syntax](#) (see page 209)

FIND / GET and COMPUTE Field-Reference Clause

Field-reference is used in the *expression* clause, which is used in the FIND/GET WHERE *criteria* clause and the COMPUTE GROUP BY HAVING *criteria* clause.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

Syntax:



Syntax rules:**compute-title**

Specifies the name of a previously computed field used as an operand in the expression. This specification is valid only for the COMPUTE command.

view-id

Qualifies the record name by specifying the alternate name of the subschema or table from which the record is retrieved.

record-name (occurrence)

Specifies the record in which the selected field participates. If more than one occurrence of the record appears in the report file or in a retrieval path, the occurrence number of the record can be specified.

If you use *occurrence*, separate it from the field name with a period. When this parameter is not used, it defaults to the first record type retrieved in the path definition.

group-name

Fields in the record can be specified as follows:

- *olq-header*— The alternative header defined for CA OLQ use in the data dictionary or by the user. When the header contains more than one line, only the first line is displayed.
- *field-name (subscript)* — A field in a one-, two-, or three-dimensional array. Enclose the subscript parameter in parentheses.

Examples:*Matches mask-value*

When specifying a match, if a character other than one of the *mask-value* characters is specified, the match is for that character only.

matches 'string'

This MATCHES clause specifies that the characters of the string to be found must be STRING.

Matching any characters

Only the left-most significant characters of the mask need be specified explicitly when the remaining characters in the field are allowed to have any value.

To retrieve all addresses where the first two digits of the zip code are 02, the mask value can be specified as follows:

'02'

Unspecified mask characters are treated as if * (any character) were specified.

Numeric values

If the zip code is to be tested for numeric values only, the mask must be specified as:

'02###'

If the specified mask value is longer than the field being checked, the extra mask characters will be ignored.

For more information:

[Coding Considerations](#) (see page 29)

DISPLAY and SORT Field-Reference Clause

This *field-reference* clause is used in the DISPLAY, and SORT commands and allows you to identify a field in several different ways. With the *field-reference* clause, you are not restricted to specifying a field name when manipulating or displaying report files. You can identify fields in any of the following ways:

- **Relative column position** of the field in the report file (1 for the first column)
- **Computed column header** (AVG-SALARY for the computed average salary)
- **CA OLQ header** for the field (MANAGER-NAME)
- **Field name** for a particular record, or a field name, which can be further qualified either by a record name and record occurrence or by subscripts.

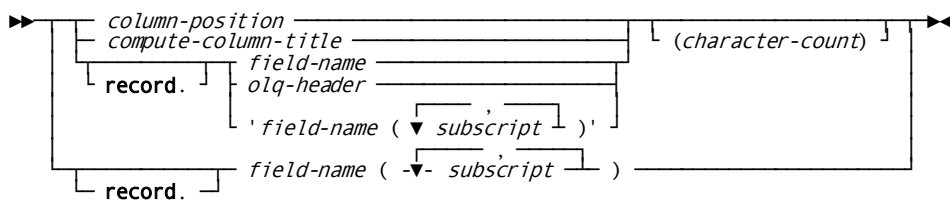
Example:

DEPT.PHONE for the phone field in the Department record, PHONE(2) for the second occurrence of the field that contains the phone number.

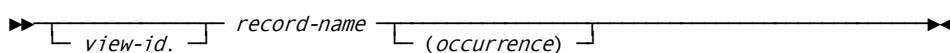
Note: The DISPLAY command can also reference the initial portion of a report-file field whose usage is DISPLAY.

The HELP/SHOW REPORT command displays the field names, column numbers, and CA OLQ headers available for use in the *field-reference* clause.

Syntax:



Expansion of record



Syntax rules:

column-position

Specifies the field in the *n*th column of the report file. Column numbers appear under the header COL in the HELP REPORT display.

compute-column-title

Specifies the name of a field created by a COMPUTE command. If the computed field name includes blanks or delimiters, it must be enclosed in quotation marks. Computed fields are listed as *COMPUTED* in the HELP REPORT display.

record

- *view-id*—Qualifies the record name by specifying the alternate name of the subschema or table from which the record is retrieved.
- *record-name (occurrence)*—Specifies the record in which the selected field participates. If the record appears more than once in the path, you can specify the occurrence number of the record. If you use *occurrence*, separate it from the field specification with a period. Record names appear under the header RECORD in the HELP REPORT display.
 - *field-name*—The name of the field in the report file. Db-key field names can be specified in the SORT command, but not in the WHERE clause or the COMPUTE statement. Field names appear under the header FIELD in the HELP REPORT display.
 - *olq-header*—The alternative record header defined for the field in the data dictionary or by the user. This name appears under the header OLQ HEADER in the HELP REPORT display. When the header comprises more than one line, only the first header line is displayed in the HELP REPORT file.

User-defined headers are specified by using the EDIT command; these headers apply only to the session in which they are specified.

field-name

Specifies one or more fields within a retrieved record type included in the internal field list for that record. Keep in mind these rules when specifying *field-name*:

- Specification of a group item automatically places all elementary fields within the group item in the internal field list.
- Specification of an elementary field name refers to that field name only.
- Fields that redefine other fields aren't placed in the internal field list unless specified individually.

(subscript)

Specifies one or more occurrences of a repeating field. Each occurrence is identified by a subscript enclosed in parentheses. Multiple entries are separated by commas and are limited to the number specified in the OCCURS clause of the schema record description. If a repeating field name is specified without a subscript, a second set of parentheses is required.

If a repeating field is specified with one or more references to a repetition, only the specified repetitions are displayed. These rules apply to nested repeating fields:

- Specifying a high-level field displays all associated repetitions of lower-level fields.
- Specifying a low-level field displays that repetition of the low-level field in all repetitions of the high-level field.
- Specifying a low-level field followed by a two- or three-part subscript (the repetition of the high-level field and the repetition of low-level fields) displays a single repetition of that field.

For more information:

[Command Summary](#) (see page 15)

[Coding Considerations](#) (see page 29)

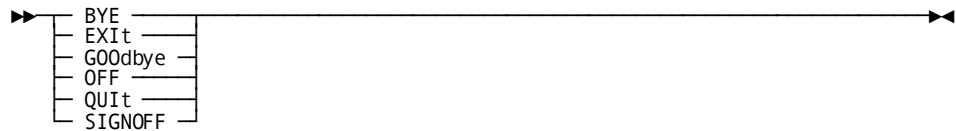
Chapter 6: Commands and Syntax

This chapter presents commands and syntax.

BYE

BYE terminates the CA OLQ session. When you terminate a session with BYE, CA OLQ deletes the current report file.

Syntax:



Example:

When you issue BYE, CA OLQ displays the following message:

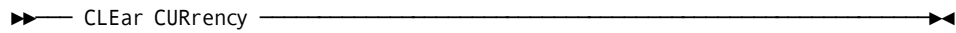
```
OLQ 100029 00 Signoff accepted - CA OLQ session terminated
```

CLEAR CURRENCY

CLEAR CURRENCY releases all database currencies for the current subschema. With this command, you can start new retrievals without repeating the signon procedure.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

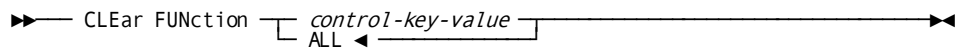
Syntax:



CLEAR FUNCTION

CLEAR FUNCTION nullifies one or all of the function key settings.

Syntax:



Parameters:

control-key-value

Specifies a single control key whose assigned command is to be cleared. Valid values are [PA1], [PA2], and 1 through 99 (corresponding to PF keys 1 through 99).

ALL

Clears all control keys of their current functions (default).

Example:

This example nullifies the [PF8] key:

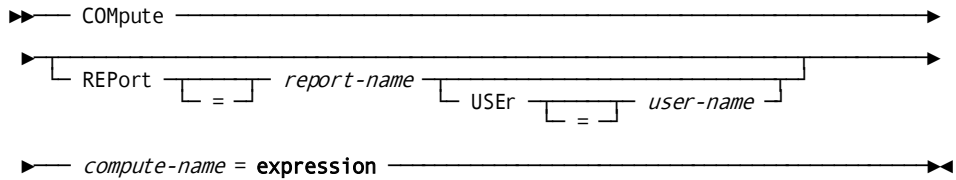
```
clear function 8
```

COMPUTE

COMPUTE performs computations on fields in a report file. Computed fields can be manipulated similarly to database records.

To display a computed field in a structured report, specify the COLS= parameter in the DISPLAY commands.

Syntax:



Parameters:

REPORT= report-name

Identifies the saved report for which the computation is performed. If you don't specify the name of the report, the computations are performed on fields in the current report.

- **USER=*user-name***—Identifies the user ID of the report owner. If you don't specify the user ID, the report is retrieved from the current user's directory.

compute-name

Specifies the field name to be used to reference the computation in any of the OLQ reporting functions (that is, DISPLAY COLS=, EDIT, SORT). A *compute-name* that contains embedded blanks or special characters must be enclosed in quotation marks.

expression

Defines the computations used to create the new column.

Examples:

The following examples use the report built by the SELECT statement:

```
select emp-last-name-0415,salary-amount-0420,bonus-percent-0420
from employee, emposition where emp-emposition
```

Compute Bonus, Total Salary

COMPUTE commands are used to define the computation of the BONUS, and TOTAL SALARY fields:

```
compute bonus=salary-amount-0420 * bonus-percent -0420
compute 'total salary'=salary-amount-0420 + bonus
display columns=emp-last-name-0415, 'total salary', bonus
```

```

                EMPLOYEE/EMPOSITION REPORT
                mm/dd/yy

EMP- LAST- NAME -0415      TOTAL SALARY      BONUS

LINGER                    38654.000      154.000
TERNER                    13052.000       52.000
LINGER                    42797.500     297.500
LINGER                    38152.000     152.000
PENMAN                    39156.000     156.000
LINGER                    85850.000     850.000
LINGER                    75750.000     750.000
LITERATA                  37762.500     262.500
WILCO                     80800.000     800.000
HEAROWITZ                 33231.000     231.000
TYRO                      20000.000      80.000
KAHALLY                   20000.000      80.000
PAPAZEUS                  101000.000    1000.000
PAPAZEUS                   90900.000     900.000

```

- 1 -

To see more of the report, page down:

```
display next page
```

EMPLOYEE/EMPOSITION REPORT mm/dd/yy		
EMP- LAST-NAME-0415	TOTAL SALARY	BONUS
ARM	46322.000	322.000
KING	14558.000	58.000
CLOUD	53119.250	369.250
HENDON	242400.000	2400.000
PEOPLES	80800.000	800.000
DOUGH	33231.000	231.000
ORGRATZI	39273.000	273.000
WAGNER	47329.000	329.000
GALLWAY	33231.000	231.000
GARDNER	14056.000	56.000
JACOBI	55385.000	385.000
WILDER	90900.000	900.000
MUNYON	36252.000	252.000
CLOTH	38266.000	266.000

- 2 -

Compute with a Built-In Function

This example uses a built-in function to define a compute statement:

```
compute name = concatenate(extract(emp-last-name-0415),', ',
emp-first-name-0415) !
display columns = name, salary-amount-0420
```

EMPLOYEE/EMPOSITION REPORT mm/dd/yy	
NAME	SALARY-AMOUNT-0420
GRANGER, PERCY	34500.00
FERNDAL, JANE	22500.00
ZEDI, BETSY	37000.00
ANDALE, ROY	33500.00
CROW, CAROLYN	37500.00
GARFIELD, JENNIFER	65000.00
GARFIELD, JENNIFER	55000.00
GARFIELD, JENNIFER	45000.00
CRANE, HERBERT	75000.00
CRANE, HERBERT	70000.00
CRANE, HERBERT	60000.00
LIPSICH, HERBERT	18500.00
JENSON, RUPERT	82000.00
JOHNSON, CYNTHIA	13500.00

- 3 -

For more information:

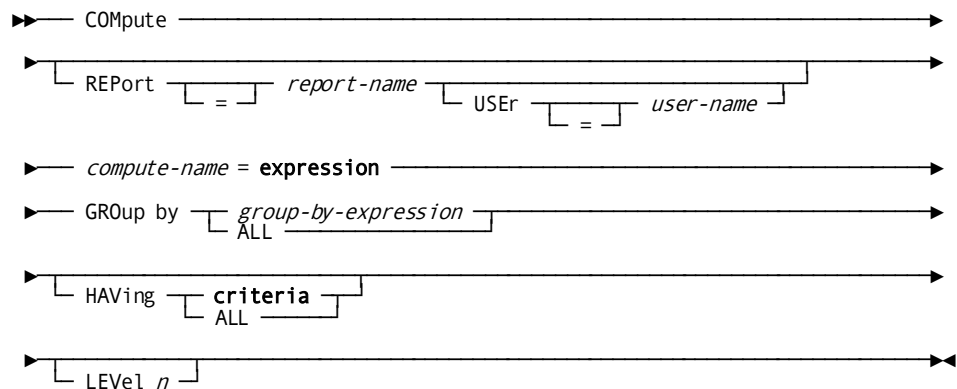
[Global Syntax](#) (see page 35)

COMPUTE ... GROUP BY

COMPUTE GROUP BY performs computations on fields in a report file. Computed fields can be manipulated similarly to database records.

These computations are then displayed at break points that have been defined by the GROUP BY clause.

Syntax:



Parameters:

REPORT= report-name

Identifies the saved report for which the computation is performed. If you don't specify the name of the report, the computations are performed on fields in the current report.

- **USER=***user-name*— Identifies the user ID of the report owner. If you don't specify the user ID, the report is retrieved from the current user's directory.

compute-name

Specifies the field name to be used to reference the computation in any of the OLQ reporting functions (that is, DISPLAY COLS=, EDIT, SORT). A *compute-name* that contains embedded blanks or special characters must be enclosed in quotation marks.

expression

Defines the computations used to create the new column.

Expression typically contains an aggregate function.

GROUP BY

Specifies a break will occur.

- *group-by-expression* — Specifies the field to break on; can be any compute expression that does *not* contain an aggregate function
- ALL—Final break processing for the entire report

HAVING

Applies selection criteria to the groupings of data values determined by the GROUP BY expression.

- criteria—Restricts the groups displayed; can contain aggregate functions; for example, HAVING AVE(SALARY) > 40000
- ALL—Specifies that no selection criteria on groups should be applied

Note: For more information about HAVING clause, see [FIND / GET Selection Criteria](#) (see page 42).

LEVEL *n*

Specifies the break level, in numeric order, with 1 representing the highest level. For example, if LEVEL 1 is specified, a break at the highest level causes a break at each subsequent lower level. The default *n* value is 1.

Examples:

The following example uses the report built by the SELECT statement:

```
select dept-id-0410,emp-last-name-0415,salary-amount-0420
from department,employee,emposition
where dept-employee and emp-emposition
```

Compute Total Salary

COMPUTE commands are used to define the computation of the 'TOTAL SALARY' field:

```
compute 'total salary' = sum(salary-amount-0420)
group by dept-id-0410 ! display
```

```

                DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
                mm/dd/yy

DEPT-ID-0410  EMP- LAST-NAME-0415          SALARY-AMOUNT-0420

    6666  HENDON                          240000 .00
    6666  PAPAZEUS                        100000 .00
    6666  PAPAZEUS                         90000 .00
    6666  RUPEE                            80000 .00
    6666  RUPEE                            76000 .00
    6666  WILDER                          90000 .00
                -----
                TOTAL SALARY              676000 .00

    2000  BLOOMER                          15000 .00
    2000  HUTTON                           44000 .00
    2000  JENSON                           82000 .00
    2000  KIMBALL                          45000 .00
    2000  KING                             14500 .00

```

- 1 -

```

DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
                mm/dd/yy

DEPT-ID-0410  EMP- LAST-NAME-0415          SALARY-AMOUNT-0420

    2000  NICEMAN                          14000 .00
                -----
                TOTAL SALARY              214500 .00

```

- 2 -

Compute Having

The HAVING clause is used to display the total salary of any department with more than 5 employees:

```
compute 'total salary' = sum(salary-amount-0420)
group by dept-id-0410 having count > 5 ! display
```

DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
mm/dd/yy

DEPT-ID-0410	EMP- LAST-NAME-0415	SALARY-AMOUNT-0420
6666	HENDON	240000.00
6666	PAPAZEUS	100000.00
6666	PAPAZEUS	90000.00
6666	RUPEE	80000.00
6666	RUPEE	76000.00
6666	WILDER	90000.00
	TOTAL SALARY	676000.00
2000	BLOOMER	15000.00
2000	HUTTON	44000.00
2000	JENSON	82000.00
2000	KIMBALL	45000.00
2000	KING	14500.00

- 1 -

DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
mm/dd/yy

DEPT-ID-0410	EMP- LAST-NAME-0415	SALARY-AMOUNT-0420
2000	NICEMAN	14000.00
	TOTAL SALARY	214500.00
3100	DOUGH	33000.00
3100	GALLWAY	33000.00
3100	GARFIELD	65000.00
3100	GARFIELD	55000.00
3100	GARFIELD	45000.00
3100	GRANGER	34500.00
3100	HEAROWITZ	33000.00
3100	JACOBI	55000.00
3100	JENSEN	37000.00
3100	LINGER	42500.00

- 2 -

DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
mm/dd/yy

DEPT-ID-0410	EMP- LAST-NAME-0415	SALARY-AMOUNT-0420
3100	LINGER	38000.00
3100	LITERATA	37500.00
3100	TYRO	20000.00
	TOTAL SALARY	528500.00

For more information:

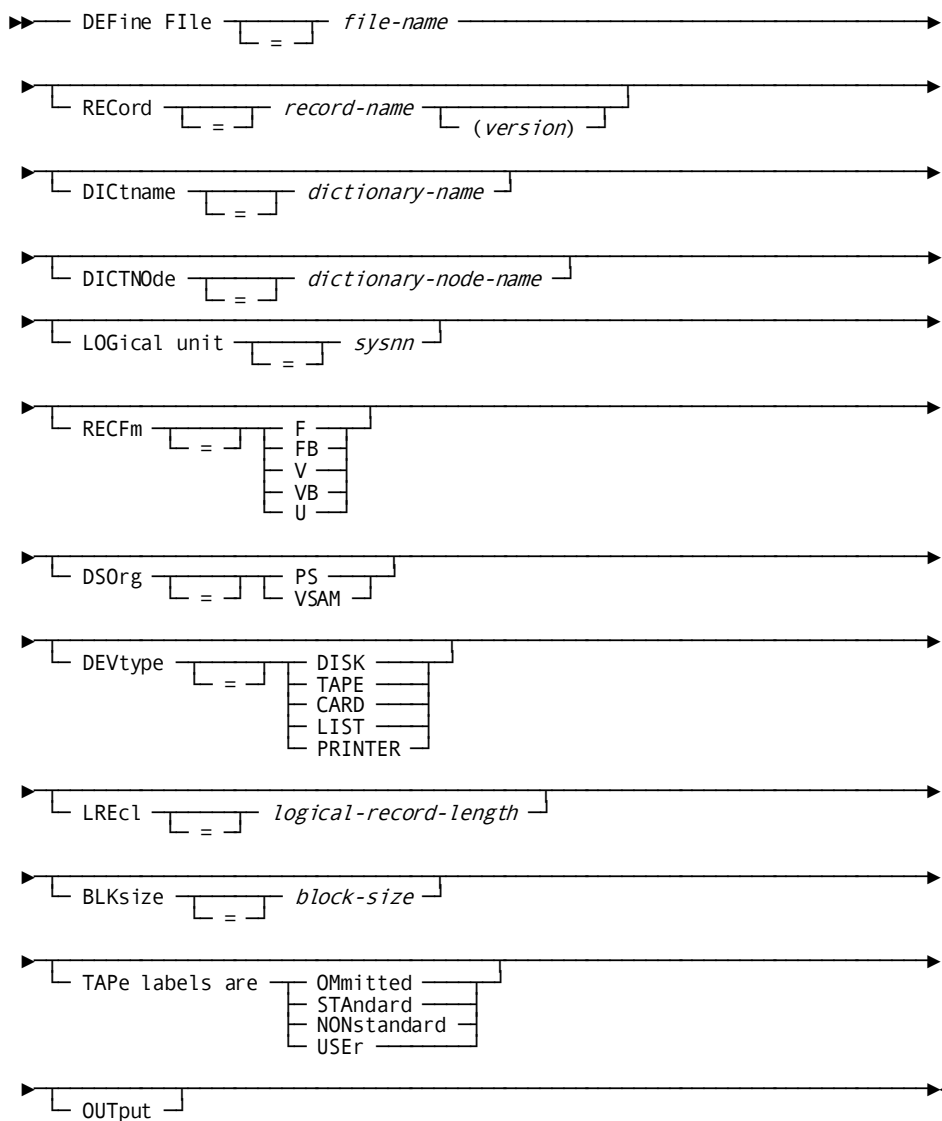
[Global Syntax](#) (see page 35)

DEFINE FILE

DEFINE FILE allows OLQ/Batch to include a data set as input to a query or to send to a data set the unedited data that was captured during a query. A DEFINE command identifies a file and relates an input data set to an IDD-defined record.

- File name
- Record name
- Dictionary name
- Dictionary node
- Logical unit size
- Record format
- Data set organization
- Device type
- Logical record length
- Block size
- Tape labels

Syntax:



Parameters:

file-name

In OS and CMS, a 1- to 8-character alphabetic file name that assigns a file name and file characteristics to an input or an output file. In DOS, *file-name* can be a maximum of 7 characters long. The file name must be referenced by a DD name in your batch job stream.

RECORD= record-name

A 1- to 32-character alphabetic name of a record stored in the data dictionary that names an Integrated Data Dictionary (IDD®) record corresponding to the file definition.

- (*version*)—An integer that indicates the record version (defaults to 1).

DICTNAME= dictionary-name

A 1- to 8-character alphabetic dictionary name that identifies the dictionary where the IDD record definition resides.

DICTNODE= dictionary-node-name

A 1- to 8-character alphabetic dictionary node name, Distributed Database System (DDS) only, that identifies the dictionary node in which the IDD record definition resides.

LOGICAL UNIT= sysnnn

Specifies the name of the logical unit.

RECFM= record-format

Specifies the record format of the named file. *Record-format* can be:

- F—Fixed
- FB—Fixed block
- V—Variable
- VB—Variable block
- U—Undefined

Record-format is required for DOS.

DSORG= data-set

Specifies the data set organization of the named file.

- PS—Physical sequential
- VSAM—VSAM entry sequenced data set

DEVTYPE= device-type

Specifies the device type for the named file. *Device-type* can be:

- DISK
- TAPE
- CARD
- LIST
- PRINTER

LRECL= logical-record-length

Specifies the logical record length, in bytes, of the named file. *Logical-record-length* is an integer in the range of 1 to 32,767. This parameter is required for DOS.

BLKSIZE= block-size

Specifies the block size, in bytes, of the named file. *Block-size* is an integer in the range 1 to 32,767.

This parameter is required for DOS.

TAPE LABELS ARE label-status

Specifies tape labels for the file definition. *Label-status* can be:

- OMITTED
- STANDARD
- NONSTANDARD
- USER

OUTPUT

Identifies the file as an output file.

Example 1:

To use a data set as input to a query, describe the data in the input file as a record in IDD such as:

```
ADD
  RECORD NAME IS INPUT-REC VERSION IS 1.

02 INPUT-ELE-1
  PICTURE IS S999
  USAGE IS COMP-3.

02 INPUT-ELE-2
  PICTURE IS 9(18)
  USAGE IS DISPLAY.
```


and include input cards such as:

```
//SYSIPT DD *
  DEFINE FILE INFILE RECORD INPUT-REC
  SELECT * FROM INPUT-REC
  DISPLAY
//INFILE DD *
  123456789012345678 ** COLUMNS 1 AND 2 CONTAIN x'013C' **
  456789012345678901 ** COLUMNS 1 AND 2 CONTAIN x'123C' **
*/
```

INPUT-REC REPORT 05/17/07	
INPUT-ELE-1	INPUT-ELE-2
13	123456789012345678
123	456789012345678901

The corresponding job control language statement must name the file in the DD name:

Op. System	JCL Statement
z/OS	//INFILE DD DSN= <i>infile</i> ,DISP=SHR
z/VSE	// DLBL INFILE,' <i>infile</i> '
	// EXTENT <i>sysnnn</i> , <i>nnnnnn</i> ,,1, <i>ssss</i>
	// ASSIGN <i>sysnnn</i> ,DISK,VOL= <i>nnnnnn</i> ,SHR
CMS	FILEDEF INFILE DISK <i>filename filetype filemode</i> (OPTIONS

Example 2:

To use a data set as output from a query, include commands:

```
//OUTFILE DD DSN=
//SYSIPT DD *
SIGNON SS=EMPSS01
DEFINE FILE OUTFILE OUTPUT
SELECT 'DEPARTMENT'.* 'EMPLOYEE'.* FROM 'DEPARTMENT', 'EMPLOYEE' -
OUTPUT OUTFILE
```

It is not required to define a record in the data dictionary to describe the output. The output will contain all the fields from the DEPARTMENT record followed by every field in the EMPLOYEE record. No editing of any fields will be done, and the fields will be contiguous.

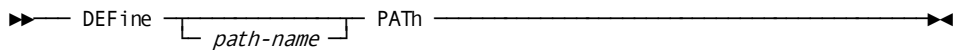
```
5300BLUE SKIES2          03210023CATHERINE O'HEAR
5100BRAINSTORMING      00150023CATHERINE O'HEAR
2000ACCOUNTING AND PAYROLL  00110023CATHERINE O'HEAR
```

DEFINE PATH

DEFINE PATH places CA OLQ in database path definition mode.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

Syntax:



Parameters:

path-name

The 1- to 32-character name of the path being defined. When a path name is specified, it appears as a title on each report file output page.

Example:

This example defines a simple path:

```
define path
get all seq office
get all employee in office-employee
execute path
display cols=office-code-0450 emp-last-name-0415 emp-city-0415
```

This path retrieves all the occurrences of the office record, the employees belonging to that office, and displays the office code, the employees' last names, and the employees' cities.

DELETE COMPUTATION

DELETE COMPUTATION deletes the named computed fields from the report.

Syntax:

```

▶▶ DELEte Computation [compute-name]
                        ALL
▶▶
  
```

Parameters:

compute-name

Specifies the name of the computed field to be deleted.

ALL

Specifies that all computed fields be deleted.

Example:

This example operates on the report created and modified by the following SELECT and COMPUTE statements:

```

select emp-id-0415, salary-amount-0420
  from employee, emposition where emp-emposition !
compute 'average salary' = avg(salary-amount-0420) group by all !
display
  
```

EMP-ID-0415	SALARY-AMOUNT-0420
1204	25000
1140	23000
0145	35000
0532	30000
=====	
AVERAGE SALARY: 28250	

To delete the computed field AVERAGE SALARY:

```

delete computation 'average salary' ! display
  
```

EMP-ID-0415	SALARY-AMOUNT-0420
1204	25000
1140	23000
0145	35000
0532	30000

DELETE QFILE

DELETE QFILE allows you to delete qfiles saved in the dictionary with the SAVE QFILE command.

Syntax:

```
DELEte QFIle [ = ] qfile-name [ ( version ) ]  
[ DICTname [ = ] dictionary-name ]  
[ DICTNDe [ = ] dictionary-node-name ]  
[ USEr [ = ] user-name ]
```

Parameters:

qfile-name

Identifies the qfile to be deleted

(version)

Specifies the version number of the named qfile.

DICTNAME= dictionary-name

Identifies the dictionary where the named qfile is stored.

DICTNODE= dictionary-node-name

Identifies the dictionary node that controls the named dictionary.

USER= user-name

Specifies the owner of the qfile.

DELETE REPORT

DELETE REPORT allows you to delete any reports you have passkey authority to delete.

Batch considerations:

DELETE REPORT is invalid when running local mode.

Syntax:

```

DELEte [ ALL REpOrts | REpOrt ] report-name
[ USER= user-name ]

```

Parameters:**ALL REPORTS**

Specifies that all saved reports are to be deleted.

REPORT= report-name

Specifies deletion of a specific report.

USER= user-name

Specifies the user who saved the specified report. If no user is specified, the default is the current user.

Note: In order to specify the deletion of another user's table, you must have the appropriate passkeys.

DELETE TABLE—OLQ access mode

DELETE TABLE allows you to delete ASF tables provided you hold the appropriate passkeys.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

Syntax:

```

DELEte TABLE [ = ] asf-table-name
[ OWNer [ = ] user-name ]
[ CATalog [ = ] dictionary-name ]
[ LOCation [ = ] dictionary-node ]

```

Parameters:

asf-table-name

Specifies the ASF table to be deleted

OWNER= user-name

Specifies the user ID of the owner of the ASF table. If *user-name* isn't specified, the default is the current user ID.

CATALOG= dictionary-name

Specifies the dictionary containing the catalog entry for the named ASF table.

LOCATION= dictionary-node

Specifies the DDS node controlling the catalog.

Example:

In this example, the table definition and associated occurrences for the EMP-HOSPITAL table are deleted:

```
delete table=emp-hospital owner=bdm catalog=asfdict
```

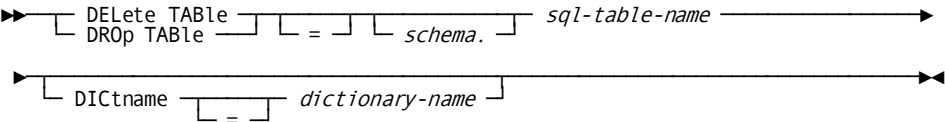
```
OLQ 116006 TABLE DEFINITION EMP-HOSPITAL SUCCESSFULLY DELETED
```

DELETE TABLE—IDMS access mode

DELETE TABLE allows you to delete SQL tables provided you hold the appropriate security.

Access mode: The syntax below is **invalid** when the access switch is set to **OLQ**.

Syntax:



Parameters:***sql-table-name***

Specifies the SQL table to be deleted

schema

The name of the schema associated with the SQL table.

DICTNAME= *dictionary-name*

Specifies the dictionary containing the catalog entry for the named SQL table.

Example:

In this example, the table definition and associated occurrences for the EMP-HOSPITAL table are deleted:

```
drop table=emp-hospital
```

```
OLQ 090016 00 Table "EMP-HOSPITAL" successfully DELETED.
```

DELETE USER

DELETE USER deletes the report directory associated with a particular user ID.

When DELETE USER is issued, all saved reports in the specified user's directory are deleted.

Note: The DELETE USER command does not remove a user from the dictionary.

Syntax:

```
▶▶ DELEte USEr          user-name ▶▶▶▶
```

Parameters:**USER= user-name**

Specifies the user ID whose report directory is to be deleted.

Note: *User-name* cannot be the user ID for the user currently signed on to CA OLQ.

Example:

The following DELETE USER command deletes the report directory for user TDB:

```
delete user = tdb
```

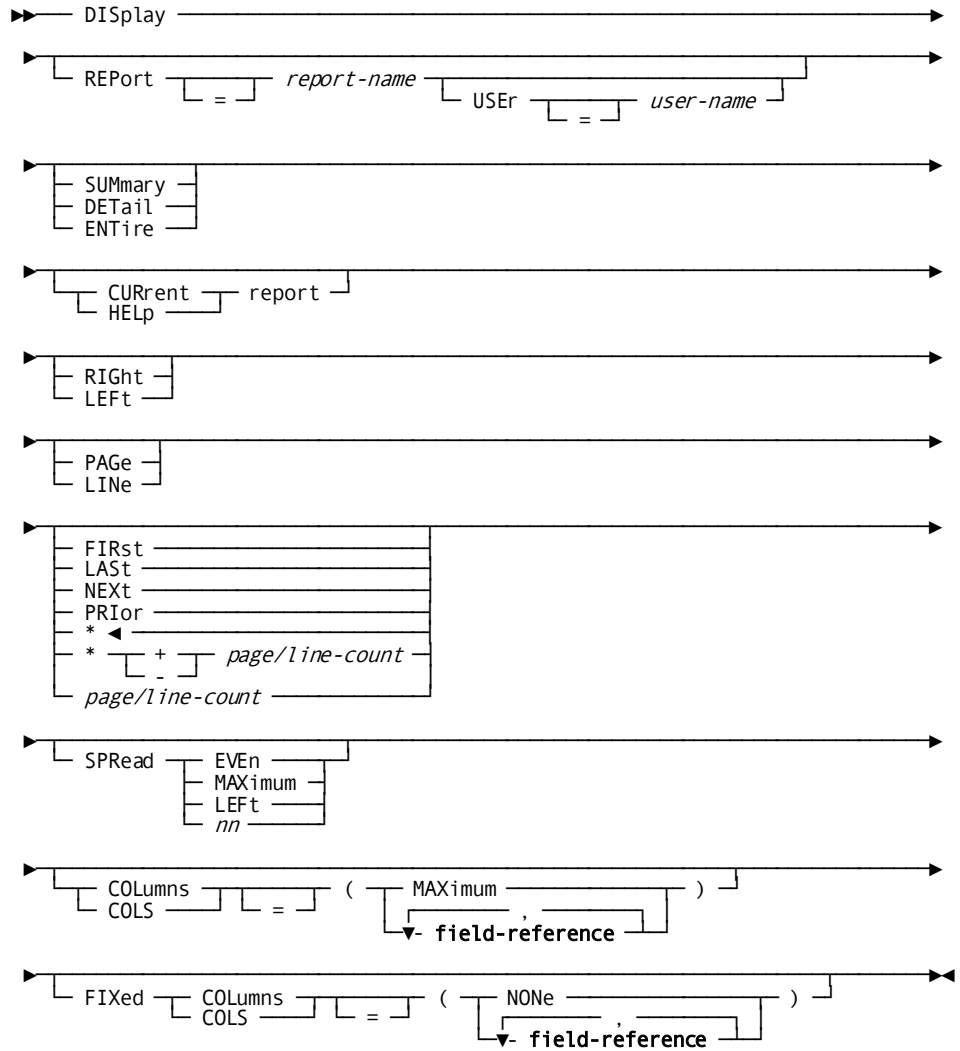
```
OLQ 107009 00 "DELETE USER " COMMAND SUCCESSFULLY COMPLETED.
```

DISPLAY

DISPLAY lists report files page-by-page on your terminal screen.

Subsequent DISPLAY commands keep the parameters set in previous DISPLAY commands unless overwritten.

Syntax:



Parameters:**REPORT= report-name**

Specifies the saved report to be displayed.

USER= user-name

The user whose report dictionary contains the named report file.

SUMMARY

Requests output of summary report lines only.

DETAIL

Requests output of detail report lines only.

ENTIRE

Request output of both detail and summary lines.

CURRENT REPORT

Requests output of the last report displayed.

HELP REPORT

Requests output of the help report file built by the last HELP command.

RIGHT/LEFT

Specifies horizontal movement within the report file.

PAGE/LINE

Requests the report begin with a specific page or line:

- PAGE requests that the report file output begins at the current or specified page number.
- LINE requests that the report file page begins at the current or specified line number.

The default is PAGE. When PAGE or LINE is specified without DISPLAY, the optional parameters SUMMARY/DETAIL/ENTIRE and CURRENT/HELP REPORT do not apply.

FIRST

Outputs a page of report file data, beginning at page 1, line 1.

LAST

Outputs the last page of report file data.

NEXT

Outputs a page of report file data, beginning at the page or line number immediately following the current page or line number.

PRIOR

Outputs a page of report file data, beginning at the page or line number immediately preceding the current page or line number.

*

Outputs the current page of report file data, beginning at the first line of that page.

* + -

Outputs a page of report file data, beginning *n* pages or lines before (-) or after (+) the current page or line number. The asterisk (*) is a required character that explicitly references the current page or line.

- *page*—The number of pages
- *line-count*—The number of lines

page/line-count

Specifies the starting point of the output relative to the current page and line number: *Page/line-count* outputs a report page, beginning at the specified page or line number.

SPREAD EVEN/MAXIMUM/LEFT/nn

Specifies the space between the columns.

- EVEN—The same number of spaces between each column (Space the columns evenly).
- MAXIMUM—The maximum number of spaces between each column.
- LEFT—Displays columns starting in the left most position with one space separating each column.
- *nn*—*nn* spaces between each column. Zero is not a valid number. The minimum number of spaces allowed is one.

COLUMNS

Specifies the columns included in the output and, optionally, the order and width of those columns. Column specifications remain in effect until altered by a subsequent DISPLAY command.

- MAXIMUM—The output of as many columns, starting with column 1, as can appear on one page of the report. Excess columns are ignored. No warning message is produced.
- field-reference—The columns and the number of characters in each output column.

FIXED COLUMNS

Specifies the columns, and their order, to remain on the screen when paging left and right. The columns specified with this parameter precede the columns specified in the `COLUMNS=` parameter. They remain fixed on the left side of the terminal screen.

Column specifications remain in effect until altered by a subsequent `DISPLAY` command, or a `FIXED COLUMNS=NONE` command.

- `NONE`—No report fields are fixed on the screen. This cleans out the fixed columns list.
- field-reference—The columns and the number of characters in each column to be output.

Considerations:

Report files sometimes contain information that cannot be displayed:

- **(@)**—The at sign indicates an unprintable character. CA OLQ provides a translation function that handles all characters written to a terminal or to the print queue.

Note: For an explanation of how to modify the CA OLQ translation table, see the CA IDMS installation guide for your operating system.

- **(*)**—The asterisk indicates invalid data. The invalid data flag appears when data is not stored in the defined format or when a `COMPUTE` command yields invalid results (as with decimal overflows and division by zero).

If you want to view the characters represented, you can use `EDIT HEXADECIMAL` to display the value in its hexadecimal representation.

Null character considerations

The null character is by default a period (.). You can override this by invoking the `SET NULL` command.

Note: For more information about the `SET` command see, [SET](#) (see page 186), later in this chapter.

Data retrieved in SQL tables can contain null values. To display them, CA OLQ pads the entire length of the display field with the null character.

Examples:

The report file used for these examples has been built by executing the SELECT statement shown below:

```
select dept-name-0410, emp-last-name-0410, salary-amount-0420
      from department, employee, emposition
      where dept-employee and emp-emposition
```

DISPLAY

When the DISPLAY keyword is specified with no subsequent parameters, the first page of the report file is output as shown below. Specification of PAGE, LINE, PAGE FIRST, PAGE 1, LINE FIRST, or LINE 1 parameters produces the same results:

```
display
OLQ 104009 04 DISPLAY RIGHT to see more report columns

      DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
            mm/dd/yy

      DEPT-NAME-0410                EMP-LAST-NAME-0415

EXECUTIVE ADMINISTRATION           HENDON
EXECUTIVE ADMINISTRATION           PAPAZEUS
EXECUTIVE ADMINISTRATION           PAPAZEUS
EXECUTIVE ADMINISTRATION           RUPEE
EXECUTIVE ADMINISTRATION           RUPEE
EXECUTIVE ADMINISTRATION           WILDER
ACCOUNTING AND PAYROLL             BLOOMER
ACCOUNTING AND PAYROLL             HUTTON
ACCOUNTING AND PAYROLL             JENSON
ACCOUNTING AND PAYROLL             KIMBALL
ACCOUNTING AND PAYROLL             KING
ACCOUNTING AND PAYROLL             NICEMAN
PERSONNEL                          FITZHUGH
PERSONNEL                          JOHNSON

**** BUFFER OVERFLOW; DISPLAY LINES LOST ****
```

DISPLAY RIGHT

The DISPLAY RIGHT command displays the report right side of the report file, if the report is too wide to fit on the screen:

```
display right

DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
mm/dd/yy

EMP- LAST-NAME-0415          SALARY-AMOUNT-0420

HENDON                      240000.00
PAPAZEUS                    100000.00
PAPAZEUS                    90000.00
RUPEE                      80000.00
RUPEE                      76000.00
WILDER                      90000.00
BLOOMER                    15000.00
HUTTON                     44000.00
JENSON                     82000.00
KIMBALL                    45000.00
KING                      14500.00
NICEMAN                    14000.00
FITZHUGH                   13000.00
JOHNSON                   13500.00

- 1 -
```

DISPLAY COLUMNS

You can display whichever columns you want in any order with the COLS= parameter:

```
display cols=2,3

DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
mm/dd/yy

EMP- LAST-NAME-0415          SALARY-AMOUNT-0420

HENDON                      240000.00
PAPAZEUS                    100000.00
PAPAZEUS                    90000.00
RUPEE                      80000.00
RUPEE                      76000.00
WILDER                      90000.00
BLOOMER                    15000.00
HUTTON                     44000.00
JENSON                     82000.00
KIMBALL                    45000.00
KING                      14500.00
NICEMAN                    14000.00
FITZHUGH                   13000.00
JOHNSON                   13500.00

- 1 -
```

SPREAD LEFT

You can use the SPREAD parameter to specify the distances between the columns. In this example the following report was modified with the SPREAD LEFT command:

```

DEPARTMENT/EMPLOYEE REPORT
mm/dd/yy
DEPT-ID-0410      EMP-ID-0415      EMP-ZIP-FIRST-FIVE-0415

6666              30      02198
6666              471     03256
6666              1       02312
6666              472     03145
2000              69      01675
2000              100     02176
2000              11      02176
2000              67      01239
2000              106     02176
2000              101     02176
1000              81      03458
1000              8683   10996
1000              51      02546
1000              91      06182

- 1 -
SPREAD LEFT squished the columns to the left side of the screen:
display spread left

```

```

DEPARTMENT/EMPLOYEE REPORT
mm/dd/yy
DEPT-ID-0410 EMP-ID-0415 EMP-ZIP-FIRST-FIVE-0415

6666      30 02198
6666      471 03256
6666      1 02312
6666      472 03145
2000      69 01675
2000      100 02176
2000      11 02176
2000      67 01239
2000      106 02176
2000      101 02176
1000      81 03458
1000      8683 10996
1000      51 02546
1000      91 06182

- 1 -

```

DISPLAY FIXED COLUMNS

This example illustrates the use of FIXED COLUMNS to keep a column on the screen while you page right and left to look at other columns:

```
display fixed columns = emp-last-name-0415

OLQ 104009 04 DISPLAY RIGHT to see more report columns
                DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
                mm/dd/yy

EMP-LAST-NAME-0415          DEPT-NAME-0410

HENDON          EXECUTIVE ADMINISTRATION
PAPAZEUS        EXECUTIVE ADMINISTRATION
PAPAZEUS        EXECUTIVE ADMINISTRATION
RUPEE           EXECUTIVE ADMINISTRATION
RUPEE           EXECUTIVE ADMINISTRATION
WILDER          EXECUTIVE ADMINISTRATION
BLOOMER         ACCOUNTING AND PAYROLL
HUTTON          ACCOUNTING AND PAYROLL
JENSON          ACCOUNTING AND PAYROLL
KIMBALL         ACCOUNTING AND PAYROLL
KING            ACCOUNTING AND PAYROLL
NICEMAN         ACCOUNTING AND PAYROLL
FITZHUGH        PERSONNEL
JOHNSON         PERSONNEL
**** BUFFER OVERFLOW; DISPLAY LINES LOST ****
```

Now if you page right, the EMP-LAST-NAME-0415 column remains on the

```
                DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
                mm/dd/yy

EMP-LAST-NAME-0415 SALARY-AMOUNT-0420

HENDON          240000.00
PAPAZEUS        100000.00
PAPAZEUS        90000.00
RUPEE           80000.00
RUPEE           76000.00
WILDER          90000.00
BLOOMER         15000.00
HUTTON          44000.00
JENSON          82000.00
KIMBALL         45000.00
KING            14500.00
NICEMAN         14000.00
FITZHUGH        13000.00
JOHNSON         13500.00
```

- 1 -

Use DISPLAY SPREAD EVEN to put an equal number of spaces between the columns:

```
display spread even

                DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
                mm/dd/yy

EMP - LAST-NAME -0415          SALARY-AMOUNT -0420

HENDON                        240000.00
PAPAZEUS                      100000.00
PAPAZEUS                      90000.00
RUPEE                         80000.00
RUPEE                         76000.00
WILDER                        90000.00
BLOOMER                       15000.00
HUTTON                        44000.00
JENSON                        82000.00
KIMBALL                       45000.00
KING                          14500.00
NICEMAN                       14000.00
FITZHUGH                      13000.00
JOHNSON                       13500.00

                - 1 -
```

Truncating columns

You can fit all report columns on the screen by truncating the values in one or more alphanumeric fields:

```
display col=1(15),2,3

                DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
                mm/dd/yy

DEPT-NAME -0410          DEPT-ID -0410 SALARY-AMOUNT -0420

EXECUTIVE ADMIN          6666          240000.00
EXECUTIVE ADMIN          6666          100000.00
EXECUTIVE ADMIN          6666          90000.00
EXECUTIVE ADMIN          6666          80000.00
EXECUTIVE ADMIN          6666          76000.00
EXECUTIVE ADMIN          6666          90000.00
ACCOUNTING AND           2000          15000.00
ACCOUNTING AND           2000          44000.00
ACCOUNTING AND           2000          82000.00
ACCOUNTING AND           2000          45000.00
ACCOUNTING AND           2000          14500.00
ACCOUNTING AND           2000          14000.00
PERSONNEL                1000          13000.00
PERSONNEL                1000          13500.00
```

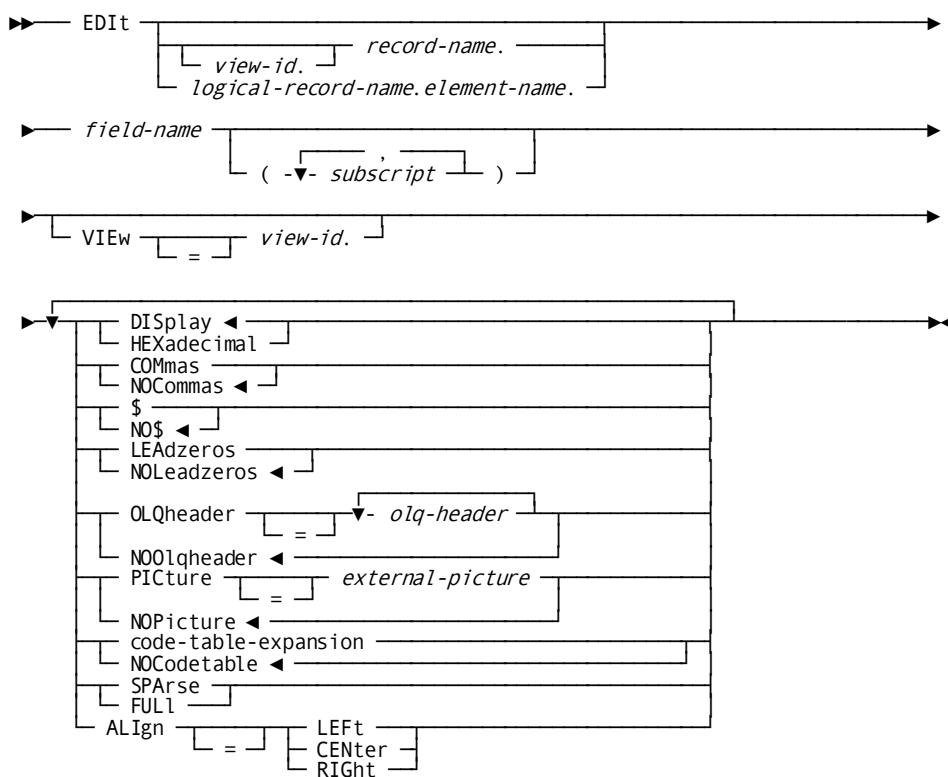
For more information:

[Global Syntax](#) (see page 35)

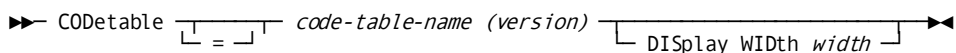
EDIT

EDIT edits a field for display. To edit a computed field for display, see [EDIT COMPUTATION](#) (see page 90).

Syntax:



Expansion of code-table



Parameters:

view-id

The alternate ID of the record or logical record name.

record-name

The database record name containing the field to be edited.

logical-record-name.element-name

The logical record element name to be edited.

field-name

The name of the field to be edited. You can request editing for fields in database records and for dbkey fields. When *field-name* is a subscripted field, only OLQHEADER/NOOLQHEADER applies to individual occurrences of the repeating field.

- (*subscript*)—One or more occurrences of a repeating field. Each occurrence is identified by a subscript enclosed in parentheses. Multiple entries are separated by commas and are limited to the number specified in the OCCURS clause of the schema record description.

If a repeating field name is specified without a subscript, all fields are displayed. If a repeating field requires more than one subscript, a second set of parentheses is required.

VIEW=view-id

Specifies the alternate ID of the record or logical record name.

DISPLAY/HEXADECIMAL

Specifies whether report fields are output in display or hexadecimal format.

When HEXADECIMAL is specified, all other output format options except OLQHEADER are ignored.

COMMAS/NOCOMMAS

Specifies whether report fields are displayed with or without commas; nonnumeric fields are unaffected.

When you specify COMMAS, commas follow every third digit of numeric displays (counting backwards from implicit or explicit decimal positions).

\$ NO\$

Specifies whether report fields are displayed with or without dollar signs; nonnumeric fields are unaffected.

LEADZEROS/NOLEADZEROS

Specifies whether the numeric fields in the field list are displayed with or without leading zeros; nonnumeric fields are not affected.

OLQHEADER= *olq-header*

Specifies that CA OLQ headers are used as headers for displayed data.

If the OLQHEADER option is used, associated CA OLQ headers replace the field names if any CA OLQ headers are defined in the data dictionary or if any are specified by the user with *olq-header*.

olq-header specifies one or more lines of user-supplied field headers. Any number of lines can be specified, up to one less than the maximum number of lines output on the terminal. CA OLQ reserves space for the display of at least one report detail line. If a blank space is included in any header line, it must be enclosed in quotation marks. When *header-line* is not specified for a particular field, CA OLQ uses the field header previously defined either in the data dictionary or through CA OLQ.

Note: To display CA OLQ headers (whether defined in the dictionary or supplied by the user), OPTIONS=OLQHEADER must be in effect.

NOOLQHEADER

Specifies that field names are used as headers for displayed data.

PICTURE=

Specifies that external pictures are used to edit report fields.

- *external-picture*—An external picture to edit a report field. External pictures override editing characteristics specified with the LEADZEROS/NOLEADZEROS, COMMAS/NOCOMMAS, and \$/NO\$ parameters of the EDIT command.
 - If an external picture is constructed with the EDIT command, CA OLQ uses it to edit the named report field.
 - If an external picture is defined for the field in the data dictionary, CA OLQ uses the stored external picture to format the field.

A user-specified external picture overrides any external picture that exists for a field in the data dictionary.

The characters available for constructing alphanumeric, alphabetic, and numeric external pictures are presented in **Table 3**.

The following rules apply to external picture construction:

- A user-specified external picture must contain at least one X, 9, G, A, Z, or *. CA OLQ uses the first X, 9, G, A, Z, or * in an external picture to determine whether the picture describes an alphanumeric, numeric, graphic, or alphabetic field, as follows:

If the first significant edit character is an **X**, CA OLQ recognizes the field as containing alphanumeric data; characters other than X, B, or parentheses are treated as insertion characters.

If the first significant edit character is an **A**, CA OLQ recognizes the field as containing alphabetic data; characters other than A, B, or parentheses are treated as insertion characters.

If the first significant edit character is a **9, Z**, or *****, CA OLQ recognizes the field as containing numeric data; characters other than 9, Z, \$, *, +, -, B, or parentheses are treated as insertion characters.

If the first significant edit character is a **G**, CA OLQ recognizes the field as containing double-byte character string (DBCS) characters; characters other than G, B, or parentheses are treated as insertion characters.

- A user-specified external picture can contain one or more insertion characters. In alphanumeric and alphabetic pictures, CA OLQ displays all insertion characters, regardless of their position in the picture format. In numeric pictures, characters other than 9, Z, \$, *, +, -, B, or parentheses are recognized as insertion characters only when embedded in a series of 9, Z, or * characters; insertion characters at the beginning or end of a numeric picture description are suppressed.

Note: When the value of a field is negative, CA OLQ will display insertion characters at the end of the numeric picture, thereby allowing the user to specify accounting information.

- A user-specified external picture can be no longer than 34 characters.

Several examples of user-specified pictures are shown in **Table 4**.

Note: External picture formats, whether specified with the EDIT command or defined in the data dictionary, are only used to edit fields when `OPTIONS=PICTURE` is in effect.

NOPICTURE

Requests a default picture to edit a report field. The default picture for a report field is derived from the internal picture stored in the data dictionary. Keep in mind the following:

- The default picture length is determined by the number of characters that is specified by the internal picture.
- The default picture data type is the same as that defined for the internal picture.
- The following translations are made for numeric internal picture characters:
 - The internal picture character **S** for a numeric field translates to a plus sign (+) or a negative sign (-) in the default picture.
 - The internal picture character **V** for a numeric field translates to a decimal point (.) in the default picture.

CODETABLE= codetable(version)

Specifies whether a code table is used to format a report field. Code tables are defined and stored in the data dictionary by using the IDD DDDL Compiler.

Note: For further information on how to create code tables, see the *CA IDMS IDD DDDL Reference Guide*.

- *codetable-name(version)*—The name of the code table used to format a report field. The code table converts encoded data to its decoded form for display on the terminal. Code tables contain:
 - Encoded values specify stored data.

The **NOT FOUND** DDDL compiler keyword often is included in a code table to define a catch-all for an encoded value. NOT FOUND ensures that an unanticipated stored value is not displayed in the report file for the edited field. When an unanticipated value is retrieved for a field for which code table editing has been requested, the user-specified literal that is paired with the NOT FOUND keyword is automatically displayed.

If the NOT FOUND keyword is not listed in a table, CA OLQ displays asterisks (*) when an invalid stored value is retrieved.
 - Decoded values specify data to be displayed on the terminal screen.
- *width*—The number of characters that you want to display on the report. For example, if you specify a width of 3 for a code table that translates 1 to January, Jan appears on the report.

A sample code table of months is shown below. For each encoded number found in the report file, CA OLQ displays a decoded month value. If an invalid number is found, the literal 'INVALID MONTH' is displayed:

ENCODED VALUE	DECODED VALUE
01	JANUARY
02	FEBRUARY
03	MARCH
.	.
.	.
.	.
.	.
.	.
12	DECEMBER
NOT FOUND	'INVALID MONTH'

Two types of code tables exist as follows:

- Built-in tables are part of the elements for which they are defined. When a HELP REPORT command is issued, CA OLQ indicates built-in tables defined for report fields by displaying ****DICTIONARY**** in the corresponding (CODE TABLE) columns. The built-in table that exists for a field in the data dictionary can be overridden by specification of a stand-alone table.
- Stand-alone tables are defined separately from the elements to which they pertain. To list the names of stand-alone tables, issue the DISPLAY ALL TABLES DDDL statement in IDD.

Code table editing overrides all other editing that might be requested for a report column, including leading dollar signs, commas, leading zeros, hexadecimal notation, and external picture formatting.

Note: When code table formatting is requested for a report field, CA OLQ uses the decoded values for subsequent processing in both the SORT command and the WHERE clause.

NOCODETABLE

Specifies that no code table is used to format a report field.

SPARSE

Specifies that only the first occurrence of a repeating column value is displayed. Note that OLQ suppresses a column display only when all of the following conditions are met:

- The column has been assigned a sparse attribute.
- The previous row contains an identical value for that column.
- The columns to the left of that column have not changed values.
- The column is not the last (right-most column) in the display. If you want the last column sparsed, you must change its display sequence so it is no longer the last column.

FULL

Specifies that all the occurrences of a repeating report line are displayed.

ALIGN

Indicates that the data values for the specified columns are to be aligned within the column boundaries as follows:

- LEFT—Aligned on the left
- CENTER—Centered
- RIGHT—Aligned on the right

The default for numeric fields is right, and the default for all other fields is left.

Table 3. Characters Used for Constructing External Pictures

Character	Type of Data Described by the Character
X	A single alphanumeric character or double-byte character string (DBCS) character stored with no shiftstrings.
B	A single blank character. B can appear anywhere in the picture.
(n)	Follows any character to represent n consecutive repetitions of the specified character. N must be an integer in the range 1 through 9999.
other	Characters other than A, B, or parentheses can be used as insertion characters.
9	A single numeric character (0 through 9).
G	A double-byte character string (DBCS) character stored with no shiftstrings.
Z	Z is an insertion character when it is preceded by a 9, a decimal point, or a zero-suppression character. Otherwise, Z is zero-suppression character.
\$	Multiple dollar signs at the beginning of an external picture represent a floating dollar sign. The dollar sign is an insertion character when it is preceded by a 9, a decimal point, or a zero-suppression character.
*	Multiple asterisks at the beginning of an external picture provide check protection. The asterisk is an insertion character when it is preceded by a 9, a decimal point, or a zero-suppression character.
+	A plus sign in the first position of an external picture indicates signed data. Multiple plus signs at the beginning of an external picture represent a floating sign. The plus sign is an insertion character when it is preceded by a 9, a decimal point, or a zero-suppression character.
-	A minus sign in the first position of an external picture indicates signed data. Multiple minus signs at the beginning of an external picture represent a floating sign. The minus sign is an insertion character when it is preceded by a 9, a decimal point, or a zero-suppression character.
.	The period character is used as a decimal point. Data is aligned with the decimal point in an external picture, and is truncated or padded when necessary. The decimal point terminates zero suppression when zero-suppression precede the decimal point. Zero-suppression characters become insertion characters if placed after a decimal point. The first period in a series of period characters is the decimal point in a picture. If no decimal point exists in the data, a decimal point is assumed after the rightmost numeric character.

Table 4. Examples of User-Specified Pictures

Data Stored	Internal Picture	External Picture	Data Displayed
123400M	X(7)	X(7)	123400M
123400M	X(7)	XBXXXXXB	1 23400 M
JOHNSON	A(4)	A(4)	JOHN
TWOWORDS	A(9)	A(3)BA(5)	TWO WORDS
2350000	9(7)	9(7)	2350000
2350000	9(7)	9(7).99	2350000.00
2350000	9(7)	\$\$\$,\$\$\$,\$\$9.99	\$2,350,000.00
2350000	9(7)	99/99/999	23/50/000
00120	9(5)	ZZZZZ	120
9876	9(4)	+++99	+9876

Examples:

The following examples illustrate the use of the EDIT command to format report fields, based on the report built with the SELECT statement shown below:

```

select emp-last-name-0415, ss-number-0415, salary-amount-0420
from employee, emposition where emp-emposition

EMPLOYEE/EMPOSITION REPORT
mm/dd/yy

EMP- LAST- NAME -0415      SS- NUMBER- 0415      SALARY- AMOUNT- 0420

LINGER                    92345812              38500.00
TERNER                    45672222              13000.00
LINGER                    19556712              42500.00
LINGER                    19556712              38000.00
PENMAN                    14593186              39000.00
LINGER                    10673343              85000.00
LINGER                    10673343              75000.00
LITERATA                  23567831              37500.00
WILCO                     111000023             80000.00
HEAROWITZ                 31896154              33000.00
TYRO                      19893456              20000.00
KAHALLY                   29661234              20000.00
PAPAZEUS                  22887770              100000.00
PAPAZEUS                  22887770              90000.00

- 1 -

```


Edit a Field

The report built above is too wide to fit on one terminal screen. Use EDIT to fit all the columns on one terminal screen:

```
edit emp-last-name-0415 pic 'x(10)' ! display

      EMPLOYEE/EMPOSITION REPORT
      mm/dd/yy

EMP- LAST- NAME-0415      SS- NUMBER-0415      SALARY- AMOUNT- 0420

LINGER                    92345812                38500.00
TERNER                    45672222                13000.00
LINGER                    19556712                42500.00
LINGER                    19556712                38000.00
PENMAN                    14593186                39000.00
LINGER                    10673343                85000.00
LINGER                    10673343                75000.00
LITERATA                  23567831                37500.00
WILCO                     11100023                80000.00
HEAROWITZ                 31896154                33000.00
TYRO                      19893456                20000.00
KAHALLY                   29661234                20000.00
PAPAZEUS                  22887770                100000.00
PAPAZEUS                  22887770                90000.00

      - 1 -
```

Edit Commas

In this example, the EDIT command requests display of the SALARY-AMOUNT-0420 field with commas and dollar signs:

```
edit salary-amount-0420 commas $ ! display

      EMPLOYEE/EMPOSITION REPORT
      mm/dd/yy

EMP- LAST- NAME-0415      SS- NUMBER-0415      SALARY- AMOUNT- 0420

LINGER                    92345812                $38,500.00
TERNER                    45672222                $13,000.00
LINGER                    19556712                $42,500.00
LINGER                    19556712                $38,000.00
PENMAN                    14593186                $39,000.00
LINGER                    10673343                $85,000.00
LINGER                    10673343                $75,000.00
LITERATA                  23567831                $37,500.00
WILCO                     11100023                $80,000.00
HEAROWITZ                 31896154                $33,000.00
TYRO                      19893456                $20,000.00
KAHALLY                   29661234                $20,000.00
PAPAZEUS                  22887770                $100,000.00
PAPAZEUS                  22887770                $90,000.00

      - 1 -
```

Edit External Picture

This example shows EDIT describing an external picture for the JOB-ID-0440 field that specifies an insertion character:

```
edit ss-number-0415 pic '999-99-9999' ! display

      EMPLOYEE/EMPOSITION REPORT
      mm/dd/yy

EMP- LAST-NAME-0415      SS-NUMBER-0415      SALARY-AMOUNT-0420

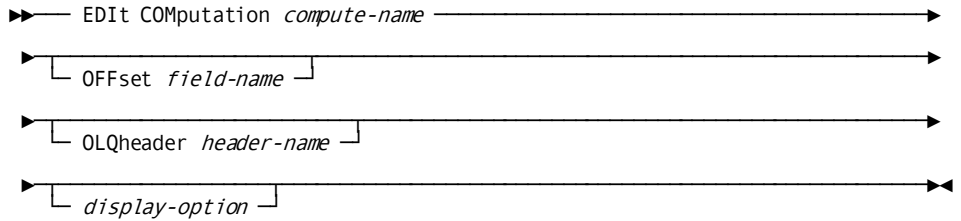
LINGER                   092-34-5812                $38,500.00
TERNER                   045-67-2222                $13,000.00
LINGER                   019-55-6712                $42,500.00
LINGER                   019-55-6712                $38,000.00
PENMAN                   014-59-3186                $39,000.00
LINGER                   010-67-3343                $85,000.00
LINGER                   010-67-3343                $75,000.00
LITERATA                023-56-7831                $37,500.00
WILCO                   111-00-0023                $80,000.00
HEAROWITZ               031-89-6154                $33,000.00
TYRO                    019-89-3456                $20,000.00
KAHALLY                 029-66-1234                $20,000.00
PAPAZEUS                022-88-7770                $100,000.00
PAPAZEUS                022-88-7770                $90,000.00

      - 1 -
```

EDIT COMPUTATION

EDIT COMPUTATION allows you to edit computed fields for display. You have the same options as with the EDIT statement.

Syntax:



Parameters:

compute-name

The name of the field to be edited.

OFFSET field-name

Specifies under which column the computed field specified by a COMPUTE GROUP BY command should be displayed. The default is the first column named in the COMPUTE statement's GROUP BY expression.

OLQHEADER header-name

Provides a report heading containing the column value of the group field. The user-supplied header takes on the edit characteristics of the computed field. *Header-name* can consist of:

- A symbolic parameter that contains a dollar sign (\$) preceding a report column name (for example, \$DEPT-NAME-0410, which would translate into the name of the department on the displayed report)
- A user-supplied title (for example, DEPARTMENT NAME instead of DEPT-NAME-0410)
- A combination of the above two

display-option

Any of the valid display options for the EDIT statement. These options are listed under the EDIT statement.

Examples:

These examples are based on the report built with the following commands:

```
select dept-id-0410,emp-last-name-0415,
salary-amount-0420
from department, employee, emposition
where dept-employee and emp-emposition
compute average-salary=avg(salary-amount-0420)
group by dept-id-0410 ! display
```

```

                DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
                mm/dd/yy

DEPT-ID-0410    EMP-LAST-NAME-0415    SALARY-AMOUNT-0420
-----
        6666    HENDON                240000.00
        6666    PAPAZEUS             100000.00
        6666    PAPAZEUS              90000.00
        6666    RUPEE                 80000.00
        6666    RUPEE                 76000.00
        6666    WILDER                90000.00
                -----
                AVERAGE-SALARY    112666.66

        2000    BLOOMER              15000.00
        2000    HUTTON               44000.00
        2000    JENSON               82000.00
        2000    KIMBALL              45000.00
        2000    KING                 14500.00
```

- 1 -

Edit Olqheader

You can change the header for the AVERAGE-SALARY field:

```
edit computation average-salary
olqheader 'avg-sal for dept $dept-id-0410' ! display

                DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
                mm/dd/yy

DEPT-ID-0410   EMP-LAST-NAME-0415           SALARY-AMOUNT-0420
-----
6666   HENDON                               240000.00
6666   PAPAZEUS                             100000.00
6666   PAPAZEUS                              90000.00
6666   RUPEE                                 80000.00
6666   RUPEE                                 76000.00
6666   WILDER                                90000.00
-----
                AVG-SAL FOR 6666 112666.66

2000   BLOOMER                               15000.00
2000   HUTTON                                44000.00
2000   JENSON                                 82000.00
2000   KIMBALL                                45000.00
2000   KING                                  14500.00

                - 1 -
```

Edit \$, Commas

You can include dollar signs and commas in the computed field:

```
edit computation average-salary $ commas ! display

                DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
                mm/dd/yy

DEPT-ID-0410   EMP-LAST-NAME-0415           SALARY-AMOUNT-0420
-----
6666   HENDON                               240000.00
6666   PAPAZEUS                             100000.00
6666   PAPAZEUS                              90000.00
6666   RUPEE                                 80000.00
6666   RUPEE                                 76000.00
6666   WILDER                                90000.00
-----
                AVG-SAL FOR 6666 $112,666.66

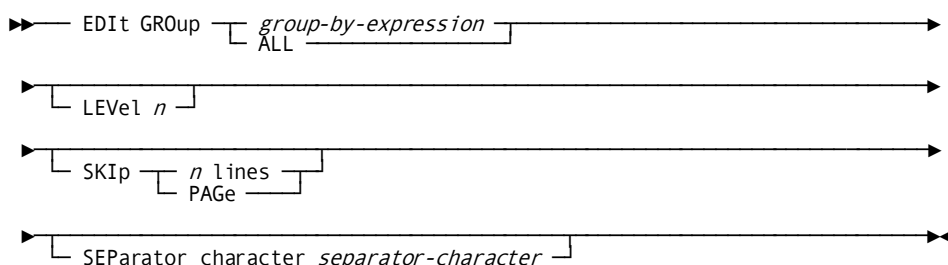
2000   BLOOMER                               15000.00
2000   HUTTON                                44000.00
2000   JENSON                                 82000.00
2000   KIMBALL                                45000.00
2000   KING                                  14500.00

                - 1 -
```

EDIT GROUP

EDIT GROUP allows you to edit fields defined by COMPUTE ... GROUP BY. You can also specify how many lines to skip between groups and define the separator character that separates a grouping from its computed value.

Syntax:



Parameters:

group-by-expression

An expression specified in the GROUP BY clause of the COMPUTE command is edited. This *group-by-expression* must identically match the expression specified in the GROUP BY clause of the COMPUTE command.

ALL.

All of the expressions specified in the GROUP BY clause of the COMPUTE command are edited.

LEVEL n

Specifies the break level, in numeric order, with 1 representing the highest level. The default *n* value is 1.

SKIP n LINES/PAGE

Specifies to skip *n* lines or a page between the computed expression and the next grouping.

SEPARATOR CHARACTER

Defines the character that separates the grouping from its computed expression. *Separator-character-value* must be one character in length.

Example:

If you specify a separator character of - (hyphen), your report would look like:

DEPARTMENT/EMPLOYEE/EMPOSITION REPORT		
mm/dd/yy		
DEPT-ID-0410	EMP-LAST-NAME-0415	SALARY-AMOUNT-0420
2000	NICEMAN	14000.00
	TOTAL SALARY	214500.00
1000	FITZHUGH	13000.00
1000	JOHNSON	13500.00
1000	ORGRATZI	39000.00
1000	PEOPLES	80000.00
	TOTAL SALARY	145500.00
	-2-	

END PATH

END PATH terminates path definition. Subsequent path commands delete the existing path.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

Syntax:

▶▶ — END PATH —————▶▶

EXECUTE PATH

EXECUTE PATH executes the retrieval commands specified in the database path definition and builds a report file of retrieved records.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

Syntax:

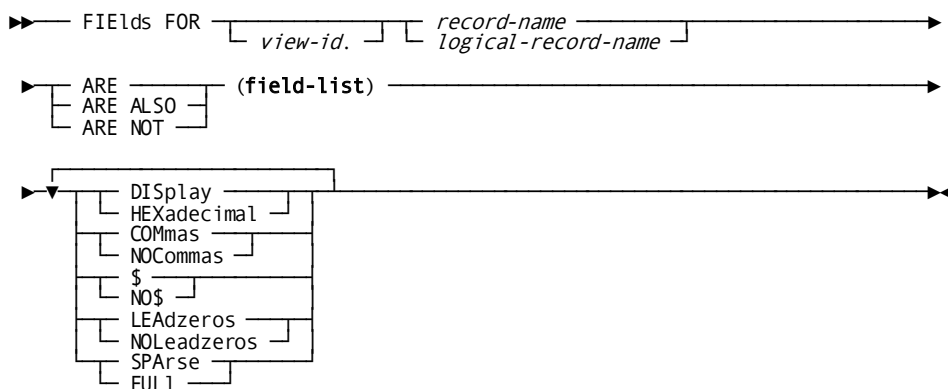
▶▶ — EXEcute PATH —————▶▶

FIELDS FOR

FIELDS FOR modifies the internal field list for a record. Use this command to reduce the size of report files. The field list you specify applies to all subsequent retrievals of the named record type. FIELDS FOR can be associated with either database record retrieval or logical record retrieval.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

Syntax:



Parameters:

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record or logical record can be found.

- *record-name*—The database record type to which field list modifications apply.
- *logical-record-name*—The logical record type to which field list modifications apply.

ARE/ARE ALSO/ARE NOT

- **ARE**—The named fields replace those in the internal field list.
- **ARE ALSO**—The named fields are added to the internal field list.
- **ARE NOT**—The named fields are deleted from the internal field list.
- *(field-list)*—The database record or logical record fields to be substituted for, added to, or deleted from the existing field list.

DISPLAY/HEXADECIMAL

Specifies whether named fields are output in display or hexadecimal format.

COMMAS/NOCOMMAS

Specifies whether the named fields are displayed with or without commas. Nonnumeric fields are unaffected.

\$ NO\$

Specifies whether named fields are displayed with or without dollar signs. Nonnumeric fields are unaffected.

LEADZEROS/NOLEADZEROS

Specifies whether the numeric fields in the field list are displayed with or without leading zeros; nonnumeric fields are unaffected.

SPARSE/FULL

Specifies whether repeating column values are displayed or not.

Examples:

The fields of the JOB record are listed below:

(LEVEL)	(FIELD NAME)	(#OCCURS)	PAGE 1.1 LINE 1 (USAGE) (PICTURE)
02	JOB-ID-0440		DISPLAY 9(4)
02	TITLE-0440		DISPLAY X(20)
02	DESCRIPTION-0440		GROUP
03	DESCRIPTION-LINE-0440	2	DISPLAY X(60)
02	REQUIREMENTS-0440		GROUP
03	REQUIREMENT-LINE-0440	2	DISPLAY X(60)
02	MINIMUM-SALARY-0440		DISPLAY S9(6)V99
02	MAXIMUM-SALARY-0440		DISPLAY S9(6)V99
02	SALARY-GRADES-0440	4	DISPLAY 9(2)
02	NUMBER-OF-POSITIONS-0440		DISPLAY 9(3)
02	NUMBER-OPEN-0440		DISPLAY 9(3)
02	FILLER		DISPLAY XX

Fields Are

When you issue a retrieval command for the JOB record without a field-list, you receive all the fields above providing OPTION ALL is in effect. You can, however, modify the field-list before or after retrieval with a FIELDS FOR command:

fields for job are (job-id-0440, title-0440)

The internal field list becomes:

JOB-ID-0440
TITLE-0040

Fields Are Also

If you want to add fields to the field list:

fields for job are also (minimum-salary-0440)

The internal field list becomes:

JOB-ID-0440
TITLE-0440
MINIMUM-SALARY-0440

Fields Are Not

You can omit fields from the field list:

fields for job are not (title-0040)

The internal field list becomes:

JOB-ID-0440
MINIMUM-SALARY-0440

For more information:

[Global Syntax](#) (see page 35)

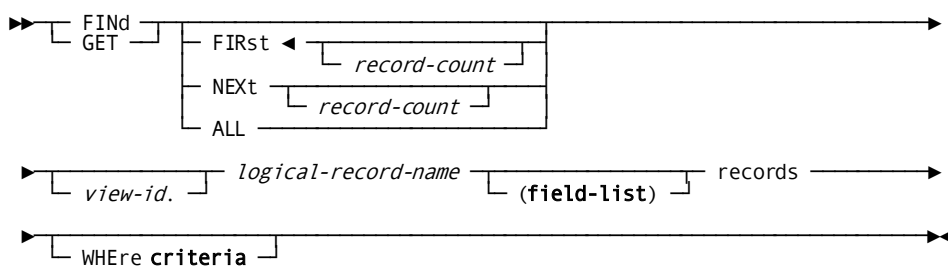
FIND / GET Logical Record

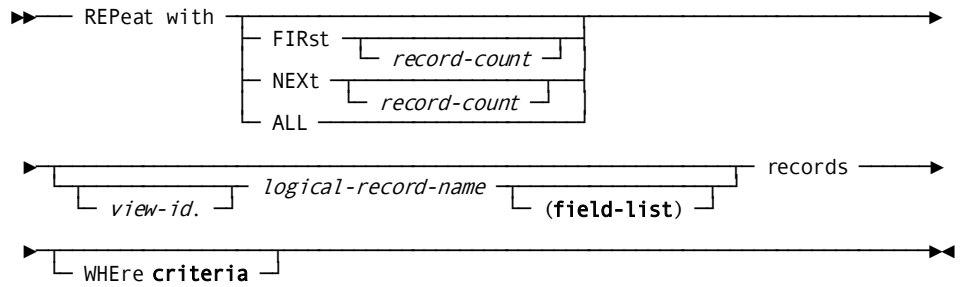
FIND/GET logical record retrieves records by using paths (defined by DBAs) through the database. Retrieval of logical records continues until either the number of records specified in the retrieval command is obtained or an error in processing is encountered.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

FIND locates database records but does not retrieve them into the report file. **GET** locates database records and does retrieve them into the report file.

Syntax:





Parameters:

FIRST record-count

Retrieves the first *n* (where *n* defaults to 1) occurrences of the named logical record (default).

NEXT record-count

Retrieves the next *n* (where *n* defaults to 1) occurrences of the named logical record.

ALL

Retrieves all occurrences of the named logical record.

view-id

The qualifying ID for the logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the logical record can be found.

logical-record-name

Specifies the logical record to be retrieved (required).

(field-list)

Specifies fields within *logical-record-name* to be displayed in the report. *Field-list* must be enclosed in parentheses.

When a logical record contains more than one field with the same name, field names entered in the field list can be qualified with the name of a database record. For example, the field name X is used in database records EMP and DEPT. If both of these fields are included in the logical record being retrieved, the user can distinguish them by specifying EMP.X or DEPT.X, respectively.

Note: CA OLQ treats the database records that make up a logical record as group level fields. When you specify a database record name in the field list for a logical record, CA OLQ includes all fields from the named database record that are contained in the logical record.

WHERE

Specifies criteria to be used by CA OLQ when selecting a record occurrence.

REPEAT

Duplicates an immediately preceding FIND/GET LOGICAL RECORD (or REPEAT for FIND/GET LOGICAL RECORD) command, with modifications as specified.

FIRST record-count

Repeats the previous command and specifies the retrieval of the first *n* record occurrences.

NEXT record-count

Repeats the previous command and specifies the retrieval of the next *n* record occurrences.

ALL

Repeats the previous command and specifies the retrieval of all occurrences.

- *logical-record-name* —The logical record name being retrieved
- *field-list* —The logical record fields being retrieved

WHERE

Specifies the criteria used in selecting records.

Considerations:

Path status is a Logical Record Facility (LRF) concept used to indicate the result of a logical record retrieval request. Path statuses can be system- or user-defined. System-defined path statuses are:

- LR-FOUND —Returned when a logical record request has been successfully executed
- LR-NOT-FOUND —Returned when LRF is unable to construct the requested logical record because one or more necessary database occurrences are not found
- LR-ERROR —Returned when LRF is unable to construct the requested logical record because of an error

A report file is built whenever the path status is LR-FOUND. Alternatively, a report file will be built for any user-defined path status if `OPTIONS=PATHSTATUS` is in effect.

Examples:

Get Logical Record

A GET logical record command retrieves the first EMP-JOB-LR logical record:

get first emp-job-lr (emp-name-0415 dept-name-0410 title-0440)

```
EMP-JOB-LR
EMP-NAME-0415      :
EMP-FIRST-NAME-0415 : 'PHINEAS  '
EMP-LAST-NAME-0415  : 'FINN      '
DEPT-NAME-0410     : 'THERMOREGULATION
TITLE-0440         : 'KEEPER OF BALLOONS '
END OF RECORD
```

Repeat Get

A REPEAT command retrieves the first record that meets the specified selection criteria:

repeat with first emp-job-lr where dept-name-0410 = thermoregulation

```
EMP-JOB-LR
EMP-NAME-0415      :
EMP-FIRST-NAME-0415 : 'PHINEAS  '
EMP-LAST-NAME-0415  : 'FINN      '
DEPT-NAME-0410     : 'THERMOREGULATION
TITLE-0440         : 'KEEPER OF BALLOONS '
END OF RECORD
```

Logical Record Keyword

A logical record keyword is used in the following example to retrieve the first record for which the TITLE-0440 field value is PROGRAMMER/ANALYST:

get first emp-job-lr where progmr-analysts

```
EMP-JOB-LR
EMP-NAME-0415      :
EMP-FIRST-NAME-0415 : 'JAMES    '
EMP-LAST-NAME-0415  : 'GALWAY   '
DEPT-NAME-0410     : 'INTERNAL SOFTWARE
TITLE-0440         : 'PROGRAMMER/ANALYST '
END OF RECORD
```

For more information:

[Global Syntax](#) (see page 35)

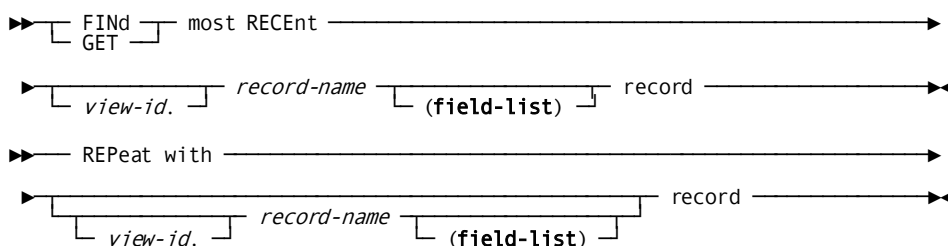
FIND / GET MOST RECENT

FIND/GET MOST RECENT retrieves the current of record type for the specified record name. If currency has not been established, an error occurs and the records cannot be retrieved.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

FIND locates database records but does not retrieve them into the report file. **GET** locates database records and does retrieve them into the report file.

Syntax:



Parameters:

MOST RECENT

Retrieves the most recent occurrence of a particular record type.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

record-name

The record type to be retrieved (required). Currency must be established for the named record. If a database record name is the same as a CA OLQ keyword, the name should be enclosed in quotation marks.

(field-list)

The fields within *record-name* to be displayed in the report file. *Field-list* must be enclosed in parentheses.

REPEAT

Duplicates an immediately preceding FIND/GET MOST RECENT (or REPEAT for FIND/GET MOST RECENT) command with modifications.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

record-name

Specifies the database record type to retrieve.

- *(field-list)* The fields to retrieve

Since a FIND/GET MOST RECENT command retrieves only one record, the REPEAT command will duplicate the same data unless the *field-list* parameter is used.

Examples:

The following examples illustrate the use of the FIND/GET MOST RECENT and associated REPEAT commands.

Get Most Recent

A GET command retrieves the most recent DEPARTMENT record:

get most recent department (dept-name-0410 dept-id-0410)

```
DEPARTMENT
DEPARTMENT-DBKEY : 2/5007103:1
DEPT-ID-0410     : 5200
DEPT-NAME-0410  : 'THERMOREGULATION'
END OF RECORD
```

Get with a Field List

A GET command retrieves the most recent EMPLOYEE record and specifies a field list:

get most recent employee (emp-id-0415 emp-city-0415)

```
EMPLOYEE
EMPLOYEE-DBKEY  : 1/5007045:1
EMP-ID-0415     : 479
EMP-CITY-0415   : 'EASTON'
END OF RECORD
```

Get Most Recent

A GET command retrieves the most recent OFFICE record:

get most recent office (office-code-0450 office-phone-0450(1))

```
sp.1
OFFICE
OFFICE-DBKEY    : 0/5007135:1
OFFICE-CODE-0450 : '005'
OFFICE-PHONE-0450(1) 4578123
END OF RECORD
```

For more information:

[Global Syntax](#) (see page 35)

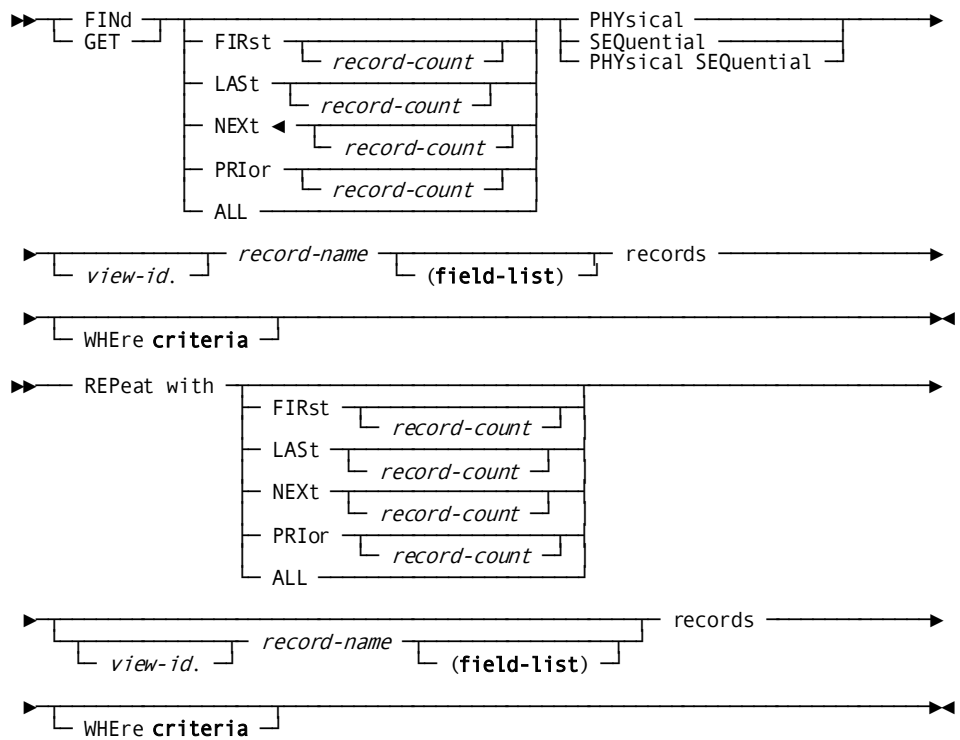
FIND / GET PHYSICAL SEQUENTIAL

FIND/GET PHYSICAL SEQUENTIAL retrieves records based on their physical position in a database area.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

FIND locates database records but does not retrieve them into the report file. **GET** locates database records and does retrieve them into the report file.

Syntax:



Parameters:

FIRST record-count

Retrieves the first n (where n defaults to 1) records at the beginning of the database area containing the named record type. The records retrieved are those with the lowest dbkey values.

LAST record-count

Retrieves the last n (where n defaults to 1) records at the end of the database area containing the named record type. The records retrieved are those with the highest dbkey values.

NEXT record count

Retrieves the next n (where n defaults to 1) occurrences of the named record type with the next-highest dbkey value. Currency for retrieval is based on the last record retrieved in the same database area. If currency has not been established, record retrieval cannot occur and no report file is built.

Note: Because NEXT records are retrieved based on current of area, the record retrieved may not always be the record required.

PRIOR record-count

Retrieves the previous n (where n defaults to 1) occurrences of the named record type with the next lower dbkey value. Currency for retrieval is based on the last record retrieved in the same database area. If currency has not been established, record retrieval cannot occur and no report file is built.

Note: Because PRIOR records are retrieved based on current of area, the record retrieved may not always be the record required.

ALL

Retrieves all occurrences of the specified record type within its associated database area.

PHYSICAL SEQUENTIAL

Specifies a serial sweep of the database. PHYSICAL and SEQUENTIAL are synonymous keywords; one is required, but both can be specified.

- *view-id*—The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.
- *record-name*—The record type to be retrieved. If a database record name is the same as a CA OLQ keyword, the name should be enclosed in quotation marks.

(field-list)

Specifies the fields within *record-name* to be displayed in the report file. *Field-list* must be enclosed in parentheses.

WHERE

Specifies criteria to be used by CA OLQ to select record occurrences.

REPEAT

Duplicates an immediately preceding FIND/GET PHYSICAL (or REPEAT for FIND/GET PHYSICAL) command.

FIRST record-count

Specifies that the first *n* records be retrieved.

LAST record-count

Specifies that the last *n* records be retrieved.

NEXT record-count

Specifies the next *n* records be retrieved.

PRIOR record-count

Specifies the prior *n* records be retrieved.

ALL

Specifies that all of the records be retrieved.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

record-name

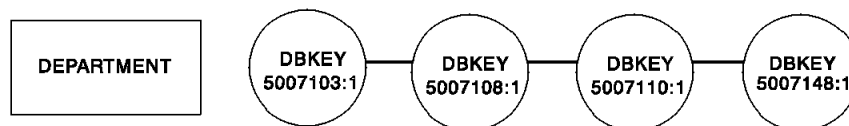
Specifies the database record type:

- *(field-list)*— The fields to retrieve

If no parameters are specified, the previous command is duplicated. If the *field-list* parameter or WHERE clause is used, *record-name* must also be specified.

Examples:

The following examples illustrate the use of the FIND/GET and associated REPEAT commands, based on the set occurrence diagram shown below:



For more information:

[Global Syntax](#) (see page 35)

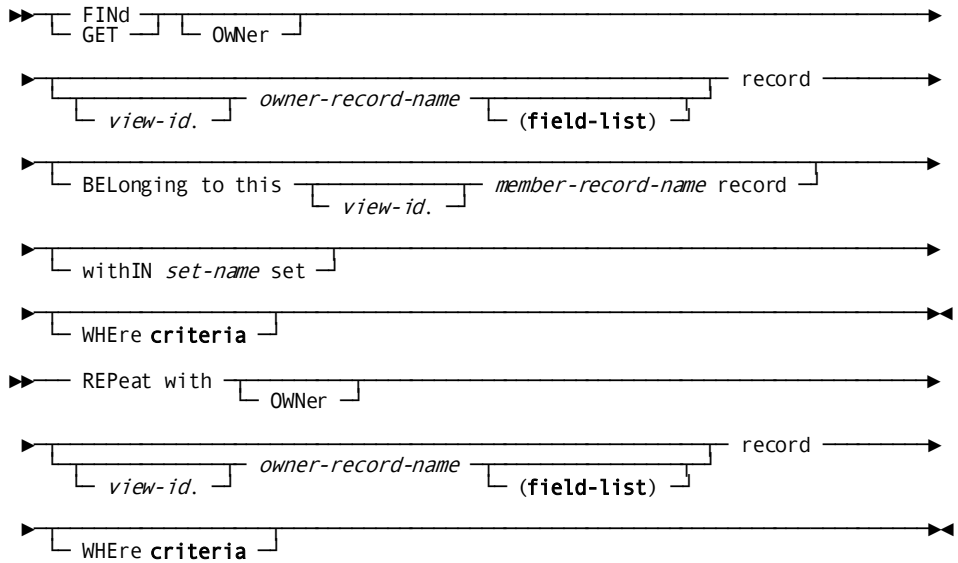
FIND / GET OWNER WITHIN SET

FIND/GET OWNER WITHIN SET retrieves the owner record of a database set occurrence. For all set membership options other than mandatory automatic (that is, mandatory manual, optional manual, and optional automatic), an occurrence of a member record type need not be a member of a set occurrence unless the parameter BELONGING TO is specified.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

FIND locates database records but does not retrieve them into the report file. **GET** locates database records and does retrieve them into the report file.

Syntax:



Parameters:**OWNER**

Specifies the retrieval of the owner record. This parameter is required if *owner-record-name* is not specified.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

owner-record-name

Specifies the owner record type in the set. *Owner-record-name* must be used if the OWNER keyword is not specified. If *member-record-name* and *set-name* are not specified, the owner record type must be the owner of only one set type in the subschema view.

- *(field-list)*—The fields within *owner-record-name* to be displayed in the report file. *Field-list* must be enclosed in parentheses.

BELONGING TO THIS member-record-name

Retrieves the owner record of the current member record type. If *owner-record-name* is not used, the member record type must participate as member in only one set in the subschema view. If both member and owner record types are named, the specified records must participate as member and owner within only one set in the subschema view.

WITHIN set-name

Specifies the *owner-record-name* set and retrieves the owner occurrence of that set type. You must use *set-name* if the owner record participates as owner in more than one set in the subschema view. Using *set-name* may be required to resolve ambiguity.

WHERE

Specifies criteria for selecting a record occurrence. If you specify a WHERE clause, you must also specify the *owner-record-name*.

REPEAT

Duplicates an immediately preceding FIND/GET OWNER WITHIN SET (or REPEAT for FIND/GET OWNER WITHIN SET) command.

OWNER= owner-record-name

Specifies retrieval of the owner record type.

- *(field-list)*— The owner record fields retrieved

WHERE

Specifies criteria used in selecting record occurrences.

Examples:

The following examples illustrate the use of FIND/GET OWNER WITHIN SET and associated REPEAT commands based on the set occurrence diagram shown below:



Get First Physical

GET enters the database and establishes currency:

**get first phys office (office-code-0450
office-phone-0450(1) where calckey = 002**

```
OFFICE
OFFICE-DBKEY      :    0/5007132:1
OFFICE-CODE-0450 : '002'
OFFICE-PHONE-0450(1)  9562377
END OF RECORD
```

Repeat Get

A REPEAT command retrieves the next OFFICE record in the in the database:

repeat with next

```
OFFICE
OFFICE-DBKEY      :    0/5007142:1
OFFICE-CODE-0450 : '003'
OFFICE-PHONE-0450(1)  3297700
END OF RECORD
```

For more information:

[Global Syntax](#) (see page 35)

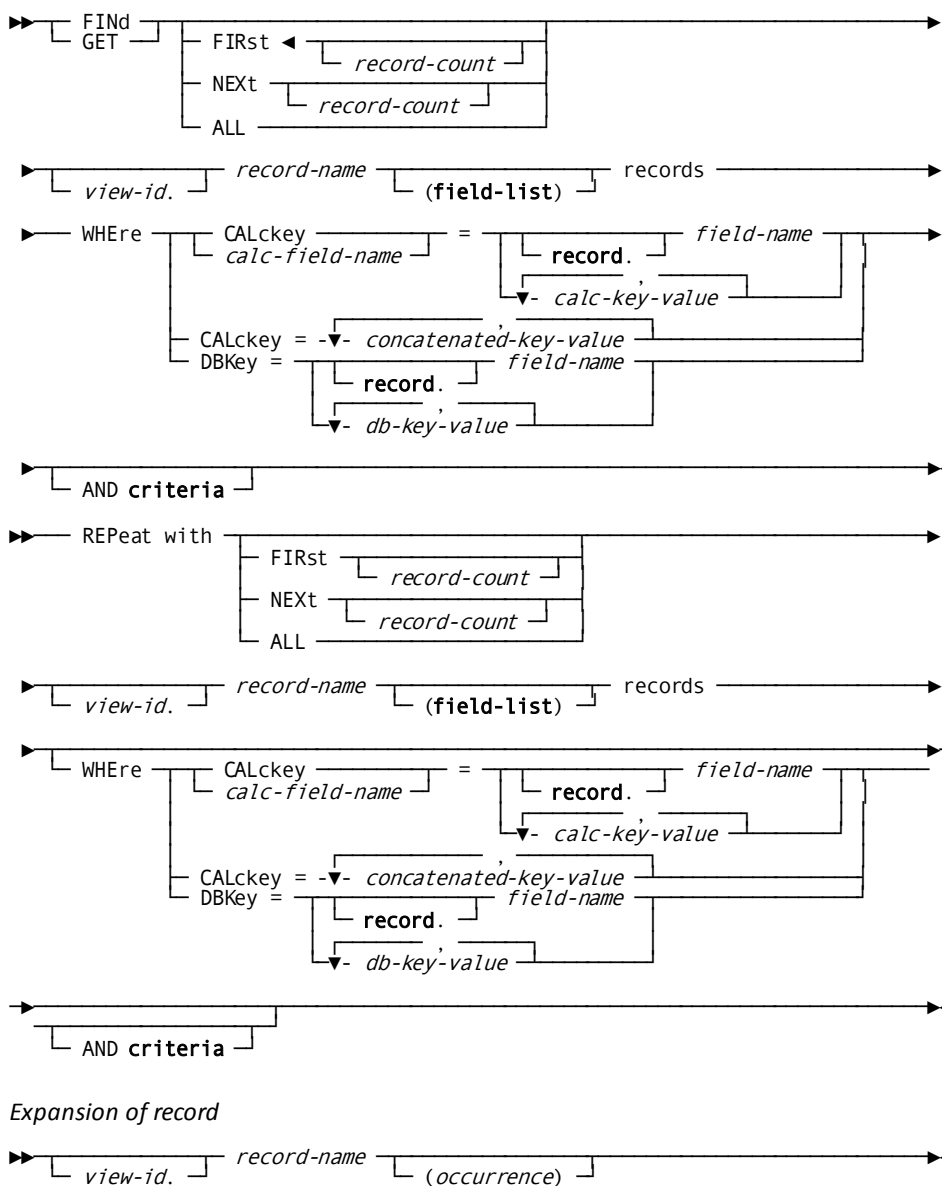
FIND / GET Using Storage Key

FIND/GET using storagekey retrieves records based on their CALC key or dbkey value.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

FIND locates database records but does not retrieve them into the report file. **GET** locates database records and does retrieve them into the report file.

Syntax:



Parameters:

FIRST record-count

Retrieves the first *n* (where *n* defaults to 1) records in the database that have the specified key value. Multiple records with the same CALC key are known as duplicates. Multiple records with the same dbkey value are not allowed.

NEXT record-count

Retrieves the next *n* (where *n* defaults to 1) records in the database that have the specified key value.

ALL

Retrieves all records in the database that have the specified key value.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

record-name

The record type to be retrieved (required). If a database record name is the same as a CA OLQ keyword, the name should be enclosed in quotation marks.

- *(field-list)*—The fields within *record-name* to be in the report file. *Field-list* must be enclosed in parentheses.

WHERE

Specifies the criteria for selecting records:

- **CALCKEY=** —The CALC key used for record retrieval. For duplicate CALC keys, the order of retrieval depends on the DUPLICATES specification in the schema record description.

Note: If the CALC key defined for the named record type is a group item, all subordinate fields must be specified.

- **calc-field-name=**—The field within *record-name* containing the CALC key, or the name of any field in the record that redefines the CALC key.
 - *record*—The qualifying record name for a *field-name*. Expanded syntax can be found following the REPEAT parameters.
 - *field-name*—The field containing the CALC key value.
 - *calc-key-value*—The associated values of the CALC key or *calc-field-name* used for record retrieval. Separate multiple values with blanks or commas.
- **CALCKEY=**—A concatenated CALC key value used for record retrieval.
 - *concatenated-key-value*—The concatenated key value of the record to be retrieved. Separate each of the partial key values with blanks or commas.

- DBKEY=—A dbkey value used for record retrieval:
 - *record*—The qualifying record name for a *field-name*. Expanded syntax can be found following the REPEAT parameters.
 - *field-name*—The field name containing the dbkey value.
 - *dbkey-value*—The dbkey value of the record to be retrieved.

Separate multiple values with blanks or commas.

Note: The NEXT ordinal clauses cannot be specified for dbkey value retrieval.

AND criteria

Specifies selection criteria used for retrieving record occurrences.

REPEAT

Duplicates an immediately preceding FIND/GET using storage key (or REPEAT for FIND/GET using storage key) command.

FIRST record-count

Specifies the retrieval of the first *n* record occurrences.

NEXT record-count

Specifies the retrieval of the next *n* record occurrences.

ALL

Specifies the retrieval of all record occurrences.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

record-name

Specifies the database record name.

- (*field-list*)—The fields to be retrieved

WHERE

Specifies criteria used in selecting records:

- CALCKEY= —The CALC key is used for record retrieval. For duplicate CALC keys, the order of retrieval depends on the DUPLICATES specification in the schema record description.

Note: If the CALC key defined for the named record type is a group item, all subordinate fields must be specified.

- *calc-field-name*—The field within *record-name* containing the CALC key, or the name of any field in the record that redefines the CALC key.
- *record*—Specifies the database record that qualifies *field-name*. Expanded syntax for *record* following the REPEAT parameter.
- *field-name*—The name of the field containing the CALC key value.
- *calc-key-value*—The associated values of the CALC key or *calc-field-name* used for record retrieval. Separate multiple values with commas or blanks.
- CALCKEY=—A concatenated CALC key value is used for record retrieval.
 - *concatenated-key-value*—The concatenated key value of the record retrieved. Separate each of the partial key values with blanks or commas.
- DBKEY=—A dbkey value is used for record retrieval:
 - *record*—The qualifying record name for a *field-name*. Expanded syntax can be found following the REPEAT parameters.
 - *field-name*—The name of the field containing the dbkey value.
 - *dbkey-value*—The dbkey value of the record retrieved.

Separate multiple values with blanks or commas.

Note: The NEXT ordinal clauses cannot be specified for dbkey value retrieval.

AND criteria

Specifies criteria used in selecting record occurrences.

If no parameters are specified and the last retrieval was performed by database-key, the same record is retrieved. The CALC key or database key does not change unless a change is specified in the REPEAT command; any new key value replaces the previous value.

record

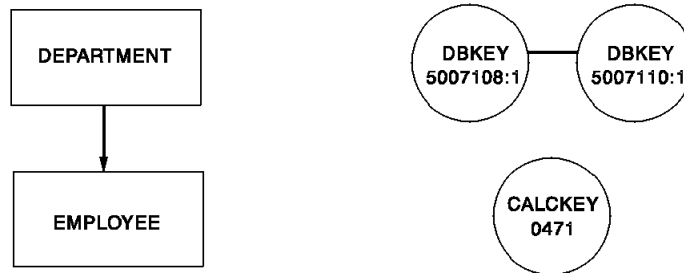
- *view-id*—The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.
- *record-name (occurrence)*—The record in the path where the *field-name* occurs.

If the *field-name* occurs in more than one record, use *record-name*.

If the record occurs more than once in the same path, use *occurrence*. If a database record name is the same as a CA OLQ keyword, the name should be enclosed in quotation marks.

Examples:

The following examples illustrate the use of FIND/GET using storage key and associated REPEAT commands, based on the set occurrence diagram shown below:

*Get Using Storage Key*

A GET using storagekey command retrieves the first DEPARTMENT record with a dbkey value of 5007108:1:

```
get first department (dept-id-0410,dept-name-0410) where calckey=200
```

```
DEPARTMENT
DEPT-ID-0410      : 2000
DEPT-NAME-0410   : ACCOUNTING AND PAYROLL
END OF RECORD
```

Repeat Get

A REPEAT command with a WHERE clause retrieves the same fields as the previous command for the new database-key value:

```
repeat with department where calckey=1000
```

```
DEPARTMENT
DEPT-ID-0410      : 1000
DEPT-NAME-0410   : PERSONNEL
END OF RECORD
```

For more information:

[Coding Considerations](#) (see page 29)

[Global Syntax](#) (see page 35)

FIND / GET WITHIN DBKEYLIST

FIND/GET WITHIN DBKEYLIST retrieves records, based on the list of dbkeys collected during previous retrieval commands.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

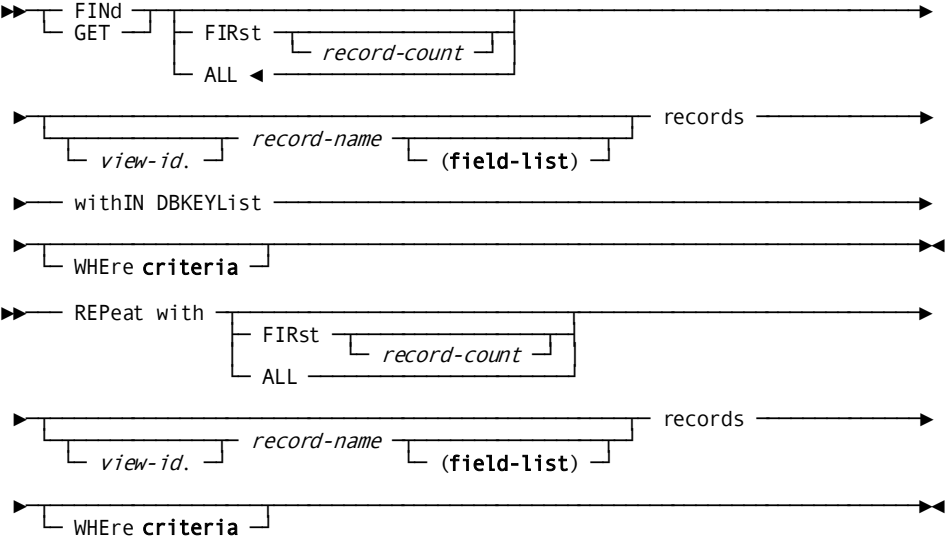
CA OLQ automatically stores the database key of each record as it is retrieved (when OPTION DBKEY is specified). When a path definition is executed, CA OLQ stores only database keys for record occurrences associated with the primary record type in the path. The resulting dbkey list provides the basis for GET WITHIN DBKEYLIST retrieval.

Use dbkey list retrieval to optimize retrieval of a large number of records using FIND. Issue GET WITHIN DBKEYLIST commands to specify progressively more restrictive selection criteria until your exact retrieval requirements are met. The optional WHERE clause is often associated with this command.

Note: The GET WITHIN DBKEYLIST command cannot be used to access Key Sequence Data Set (KSDS) VSAM files; database keys have no meaning for KSDS VSAM records.

FIND locates database records but does not retrieve them into the report file. **GET** locates database records and does retrieve them into the report file.

Syntax:



Parameters:**FIRST record-count**

Retrieves the first *n* (where *n* defaults to 1) records in the database key list.

ALL

Retrieves all the records in the dbkey list.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

record-name

The name of the record type last retrieved. Use *record-name* if you specify a field list. If a database record name is the same as a CA OLQ keyword, enclose the name in quotation marks.

- *(field-list)*—The fields within *record-name* to be stored in the report file. *Field-list* must be enclosed in parentheses.

WITHIN DBKEYLIST

Specifies retrieval of records directly, by means of the database key list (required).

WHERE

Specifies criteria used in selecting a record occurrence.

REPEAT

REPEAT FIND/GET WITHIN DBKEYLIST duplicates an immediately preceding FIND/GET WITHIN DBKEYLIST (or REPEAT for FIND/GET WITHIN DBKEYLIST) command.

FIRST record-count

Retrieves the first *n* (where *n* defaults to 1) records in the database key list.

ALL

Retrieves all the records in the database key list.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

record-name

The record type last retrieved. Use *record-name* if specifying a field list. Enclose the database record in quotation marks if it is the same as a CA OLQ keyword.

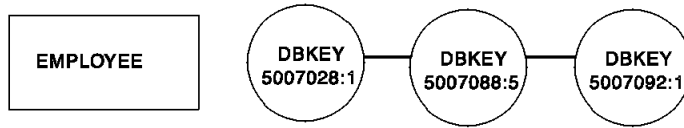
- (*field-list*)—The fields within *record-name*. Enclose *field-list* in parentheses.

WHERE

Specifies criteria used in selecting a record occurrence.

Examples:

The following examples illustrate the use of FIND/GET WITHIN DBKEYLIST and associated REPEAT commands, based on the set occurrence diagram shown below:



Find

FIND retrieves 56 EMPLOYEE records from the database:

**option dbkey !
find all sequential employee records**

OLQ 098006 00 57 whole lines and 0 partial lines in report.
OLQ 098007 00 57 records read. 57 records selected.

Get Within Dbkeylist

A GET WITHIN DBKEYLIST command retrieves all the EMPLOYEE records in which the EMP-CITY-0415 field value is WESTWOOD:

get all within dbkeylist where emp-city-0415 is westwood

OLQ 098006 00 3 whole lines and 0 partial lines in report.
OLQ 098007 00 50 records read. 3 records selected.

Repeat Within Dbkeylist

A REPEAT WITHIN DBKEYLIST selects all the EMPLOYEE records in which the EMP-CITY-0415 field value is ARLINGTON:

repeat with all where emp-city-0415 is arlington

OLQ 098006 00 3 whole lines and 0 partial lines in report.
OLQ 098007 00 50 records read. 3 records selected.

For more information:

[Global Syntax](#) (see page 35)

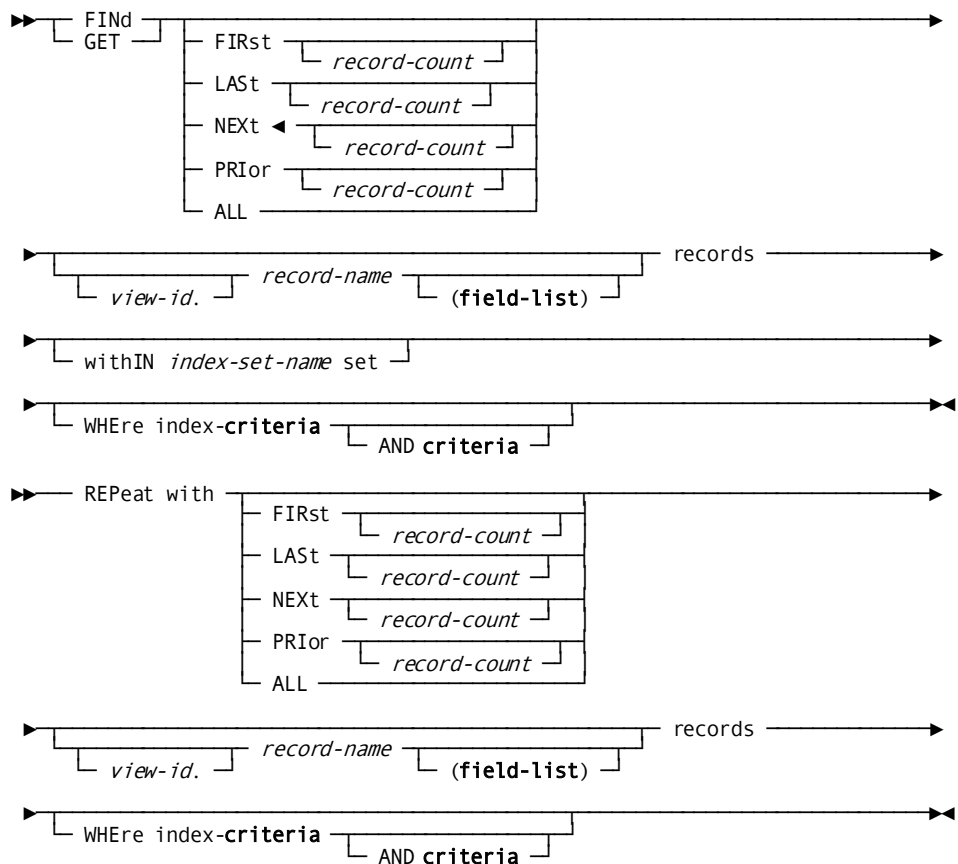
FIND / GET WITHIN Index SET

FIND/GET WITHIN index SET retrieves records using the name of an index set and the index-sort-key fields specified in the WHERE clause.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

FIND locates database records but does not retrieve them into the report file. **GET** locates database records and does retrieve them into the report file.

Syntax:



Parameters:**FIRST record-count**

Retrieves the first n (where n defaults to 1) records at the beginning of the current occurrence of the set type.

LAST record-count

Retrieves the last n (where n defaults to 1) records at the end of the current occurrence of the set type.

NEXT record-count

Retrieves the next n (where n defaults to 1) records, starting with the record that follows the current record in the current occurrence of the set.

PRIOR record-count

Retrieves the prior n (where n defaults to 1) records starting with the record that precedes the current record in the current occurrence of the set.

ALL

Retrieves all member records in the current occurrence of the set type.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

record-name

The name of the indexed record to be retrieved. If the WITHIN SET clause does not specify the index-set name, use *record-name*. *Record-name* must be a record in an index set. If you have a WHERE clause, specify *record-name*. If a database record name is the same as a CA OLQ keyword, the name should be enclosed in quotation marks.

- *(field-list)*—Specifies the fields within *record-name* to be displayed in the report file. *Field-name* must be enclosed in parentheses.

WITHIN index-set-name

Specifies the name of the index set used for retrieval. If *record-name* is not specified or if *record-name* participates in more than one index set, this parameter is required.

WHERE

Specifies comparison expressions based on the index field or its subfields. The criteria is evaluated for each entry in the index; a record is retrieved only when an index satisfies the conditions specified in this parameter.

AND criteria

Specifies criteria to be used by CA OLQ when selecting a record occurrence.

REPEAT

Duplicates an immediately preceding FIND/GET WITHIN index SET (or REPEAT for FIND/GET WITHIN index SET) command.

FIRST record-count

Retrieves the first n (where n defaults to 1) records at the beginning of the current set.

LAST record-count

Retrieves the last n (where n defaults to 1) records at the end of the current set.

NEXT record-count

Retrieves the next n (where n defaults to 1) records starting with the record that follows the current record in the current set.

PRIOR record-count

Retrieves the prior n (where n defaults to 1) records starting with the record that precedes the current record in the current set.

ALL

Retrieves all member records in the current set.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

record-name

The indexed record to be retrieved. Use this parameter if the WITHIN SET clause does not specify the index-set name. *Record-name* must be a record in an index set. If you specify a WHERE clause, also specify *record-name*. Enclose the database record name in quotation marks if it is the same as a CA OLQ keyword.

- *(field-list)*—The fields within *record-name* enclosed within parentheses.

WHERE

Specifies comparison expressions based on the index field or its subfields. The criteria is evaluated for each entry in the index; a record is retrieved only when an index satisfies the conditions specified in this parameter.

The rules for *index-criteria* are the same as for the WHERE clause *criteria* parameter.

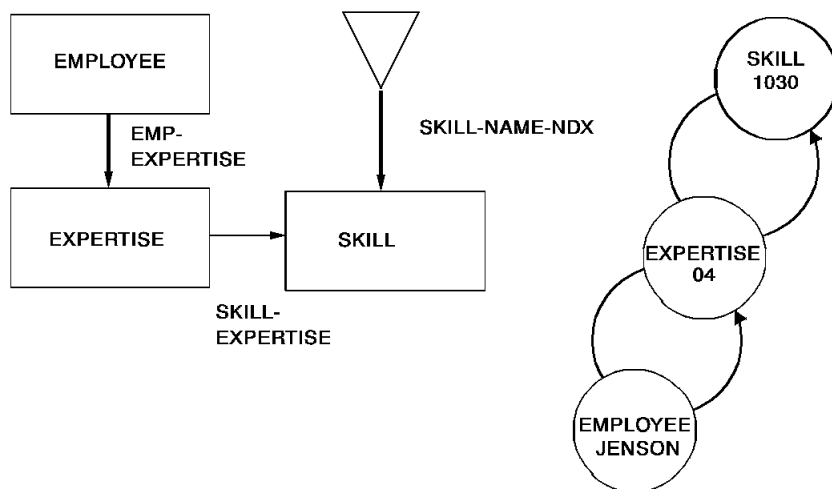
AND criteria

Specifies criteria used in selecting a record occurrence.

If the selection criteria are changed and index-set retrieval is preferred, specify the record name in the REPEAT command.

Examples:

The following examples illustrate the use of the GET WITHIN index SET command and associated REPEAT commands based on the set occurrence diagram shown below:



Get First

A GET command retrieves the first SKILL record in the SKILL-NAME-NDX index set:

```
get first skill (skill-id-0455 skill-name-0455 skill-dbkey)
record in skill-name-ndx set
```

```
SKILL
  SKILL-DBKEY      : 0/5007106:1
  SKILL-ID-0455   : 1030
  SKILL-NAME-0455 : 'ACCT MGT '
END OF RECORD
```


Repeat

A REPEAT command retrieves the next SKILL record in the SKILL-NAME-NDX index set:

repeat with next

```
SKILL
  SKILL-DBKEY      :    0/5007107:1
  SKILL-ID-0455   :    1040
  SKILL-NAME-0455 :    'DEV MGT  '
END OF RECORD
```

For more information:

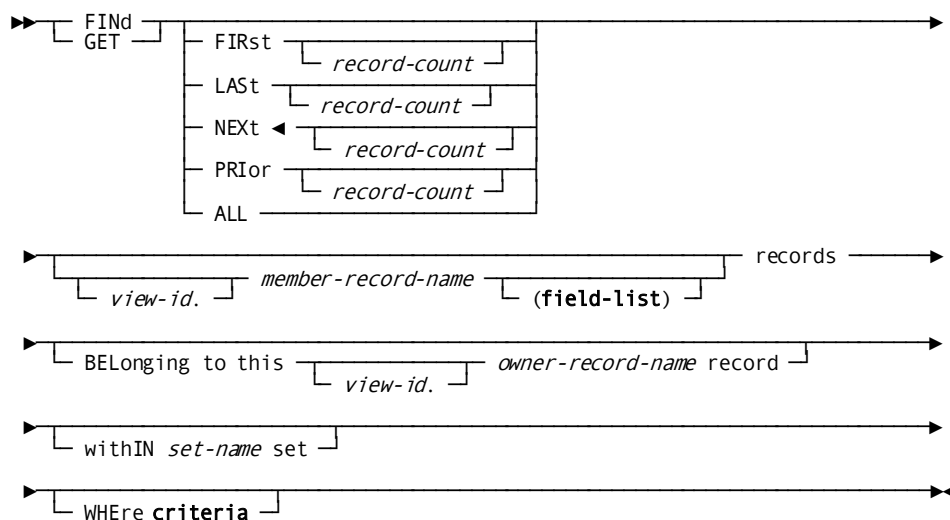
[Global Syntax](#) (see page 35)

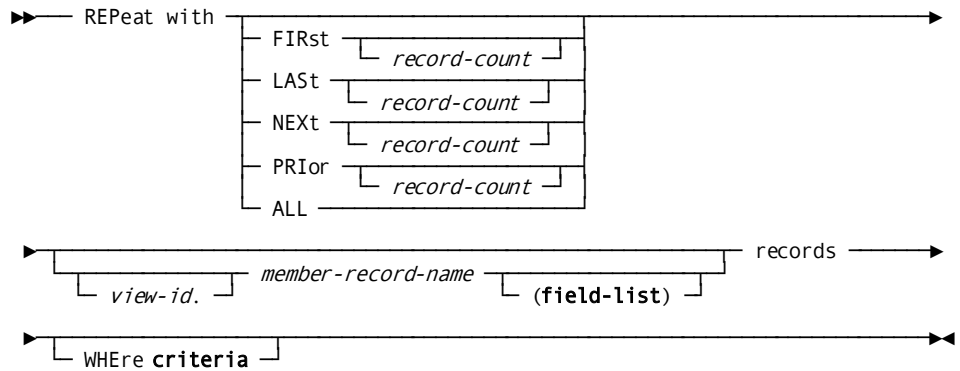
FIND / GET WITHIN SET

FIND/GET WITHIN SET retrieves records based on their membership in a database set. Use this command to retrieve records only after currency is established within the object set. (Currency need not have been established previously for system-owned index sets.)

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

FIND locates database records but does not retrieve them into the report file. **GET** locates database records and does retrieve them into the report file.

Syntax:



Parameters:

FIRST record-count

Retrieves the first *n* (where *n* defaults to 1) records at the beginning of the current occurrence of the set type.

LAST record-count

Retrieves the first *n* (where *n* defaults to 1) records at the end of the current occurrence of the set type.

NEXT record-count

Retrieves the next *n* (where *n* defaults to 1) records, starting with the record following the current record in the current occurrence of the set.

PRIOR record-count

Retrieves the prior *n* (where *n* defaults to 1) records, starting with the record preceding the current record in the current occurrence of the set. If no PRIOR pointers have been defined for the record, CA OLQ displays an error message.

ALL

Retrieves all member records in the current occurrence of the set type.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

member-record-name

Specifies a member record type in the set. If a member record type is named, the retrieval applies only to records of that type. If no member record type is specified, all member record types in the set are retrieved.

- *(field-list)*—Fields within *member-record-name* to be stored in the report file. *Field-list* must be enclosed in parentheses.

BELONGING TO THIS owner-record-name

Specifies the owner record type in the set. If you don't specify *set-name* or *member-record-name*, then the specified owner record type must participate as owner in only one set in the subschema view.

WITHIN set-name

Specifies the set where the participating record is retrieved. *Set-name* is required if the named or implied member record participates in more than one set. It is optional if the set name can be determined from the named member or owner record type.

Note: To retrieve all member record types in the set, specify either *set-name* or *owner-record-name*, but do not specify *member-record-name*.

WHERE

Specifies criteria used to select record occurrences.

REPEAT

Duplicates an immediately preceding FIND/GET WITHIN SET (or REPEAT for FIND/GET WITHIN SET) command.

FIRST record-count

Retrieves the first *n* (where *n* defaults to 1) records at the beginning of the current set.

LAST record-count

Retrieves the first *n* (where *n* defaults to 1) records at the end of the current set.

NEXT record-count

Retrieves the next *n* (where *n* defaults to 1) records, starting with the record that follows the current record in the current set.

PRIOR record-count

Retrieves the prior *n* (where *n* defaults to 1) records, starting with the record that precedes the current record in the current set. If no PRIOR pointers have been defined for the record, CA OLQ displays an error message.

ALL

Retrieves all member records in the current set.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

member-record-name

The member record type in the set. If a member record type is named, the retrieval applies only to records of that type. If no member record type is specified, all member record types in the set are retrieved.

- *(field-list)*—The fields within *member-record-name* to be stored in the report file. Enclose *field-list* in parentheses.

WHERE

Specifies criteria used in selecting record occurrences.

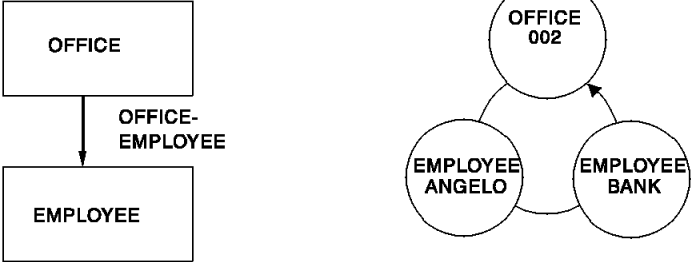
Considerations:

Use the *member-record-name*, *owner-record-name*, and *set-name* clauses to resolve ambiguity. At least one of these clauses must be specified.

If *member-record-name* is specified, but *owner-record-name* and *set-name* are not specified as listed in the syntax, be sure the named member record type participates as member in only one set in the subschema view of the database.

Examples:

The following examples illustrate the use of FIND/GET WITHIN SET and associated REPEAT commands based on the set occurrence diagram shown below:



Get First

A GET command is used to enter the database and establish currency:

```
get first office (office-code-0450 office-phone-0450(1))  
where calckey = 002
```

```
OFFICE  
OFFICE-CODE-0450 : '002'  
OFFICE-PHONE-0450(1) 9562377  
END OF RECORD
```

Get With Set

A GET WITHIN SET command is used to retrieve the first EMPLOYEE record in the OFFICE-EMPLOYEE set:

```
get first employee (emp-last-name-0415 emp-first-name-0415)  
in office-employee set
```

```
EMPLOYEE  
  EMP-FIRST-NAME-0415 : 'MICHAEL  '  
  EMP-LAST-NAME-0415  : 'ANGELO    '  
END OF RECORD
```

Repeat Next

A REPEAT NEXT command duplicates the immediately preceding GET command for the next EMPLOYEE record:

```
repeat next
```

```
EMPLOYEE  
  EMP-FIRST-NAME-0415 : 'MONTE    '  
  EMP-LAST-NAME-0415  : 'BANK     '  
END OF RECORD
```

For more information:

[Global Syntax](#) (see page 35)

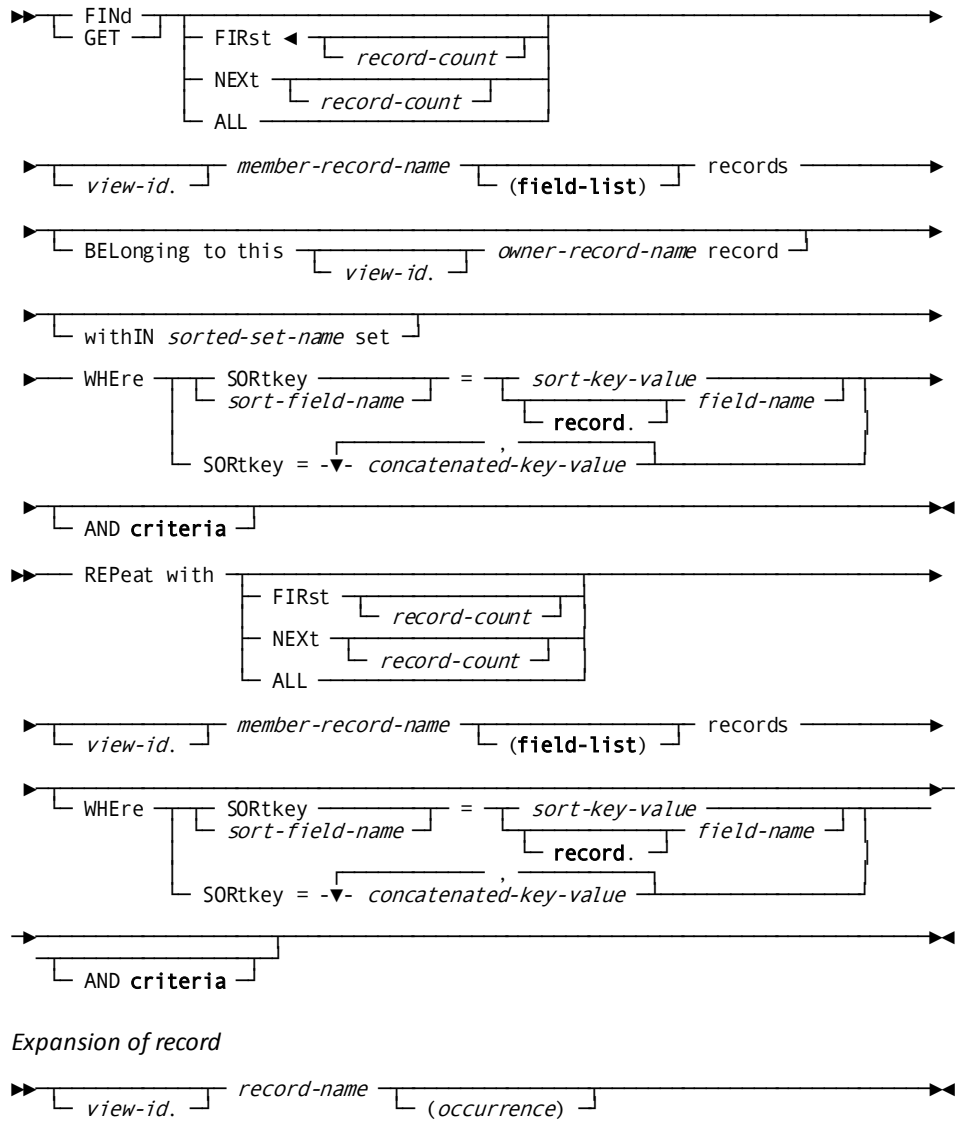
FIND / GET WITHIN SET Using SORTKEY

FIND/GET WITHIN SET using SORTKEY retrieves member records in sorted database sets based on a specified sort key.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

FIND locates database records but does not retrieve them into the report file. **GET** locates database records and does retrieve them into the report file.

Syntax:



Parameters:**FIRST record-count**

Retrieves the first *n* (where *n* defaults to 1) records in the current occurrence of the set type with the specified sort key value.

NEXT record-count

Retrieves the next *n* (where *n* defaults to 1) records in the current occurrence of the set type with the specified sort key value.

Note: Records with the same sort key (duplicates) can be retrieved by specifying the REPEAT WITH NEXT command without changing the preceding sort key value. If retrieval is requested for a record with a different sort key value, use the GET FIRST form of the command.

ALL

Retrieves all records in the current occurrence of the set type with the specified sort key value.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

member-record-name

The member record type in the set (required).

- *(field-list)*—The fields within *record-name* to be stored in the report file. *Field-list* must be enclosed in parentheses.

BELONGING TO THIS owner-record-name

Specifies the owner record type in the set. This clause is required only as needed to resolve ambiguity.

WITHIN sorted-set-name

Specifies the set type. This clause is required only as needed to resolve ambiguity.

WHERE

Supplies the criteria for selecting records:

- SORTKEY=—The sort key used for record retrieval.
- *sort-field-name*=—The name of the sort key field used for record retrieval.
Note: If the sortkey is defined as more than one field, the SORTKEY keyword should be used rather than a list of the individual field names.
- *sort-key-value*—The sort key value of the member record retrieved. In the event of duplicate sort keys, the order of retrieval depends on the DUPLICATES specification for the set type.

If the sort key is a group item, all subordinate fields must be specified.

- *record*—The qualifying record name for a *field-name*. Expanded syntax can be found following the REPEAT parameter.
- *field-name*—The name of the field containing the sort key value.
- SORTKEY=*concatenated-key-value*—A concatenated sort key value used for record retrieval.
- In the *concatenated-key-value*, separate each of the partial key values with blanks or commas.

AND criteria

Specifies criteria used in selecting a record occurrence.

REPEAT

REPEAT GET WITHIN SET using SORTKEY duplicates an immediately preceding FIND/GET WITHIN SET using SORTKEY (or REPEAT for FIND/GET WITHIN SET using SORTKEY) command.

FIRST record-count

Retrieves the first *n* (where *n* defaults to 1) records in the current set with the specified sort key value.

NEXT record-count

Retrieves the next *n* (where *n* defaults to 1) records in the current set with the specified sort key value.

Note: You can retrieve records with duplicate sort keys by specifying the REPEAT WITH NEXT command without changing the preceding sort key value. If you request retrieval for a record with a different sort key value, use the GET FIRST form of the command.

ALL

Retrieves all records in the current set with the specified sort key value.

view-id

The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

member-record-name

The member record type (required).

- *(field-list)*—The fields within *record-name*. Enclose *field-list* in parentheses.

WHERE

Specifies criteria used in selecting records.

- SORTKEY=—The sort key used for record retrieval.
- *sort-field-name*=—The sort key field used for record retrieval.
Note: If the sortkey is defined as more than one field, the SORTKEY keyword should be used rather than a list of the individual field names.
- *sort-key-value*—The sort key value of the member record retrieved. In the event of duplicate sort keys, the order of retrieval depends on the DUPLICATES specification for the set type.

If the sort key is a group item, all subordinate fields must be specified.

- *record*—The qualifying record name for a *field-name*. The expanded syntax can be found below.
- *field-name*—The field name containing the sort key value.
- SORTKEY= *concatenated-key-value*—A concatenated sort key value used for record retrieval.

Separate each of the partial key values with blanks or commas.

AND criteria

Specifies criteria used in selecting record occurrences.

record

The qualifying record name for a *field-name*.

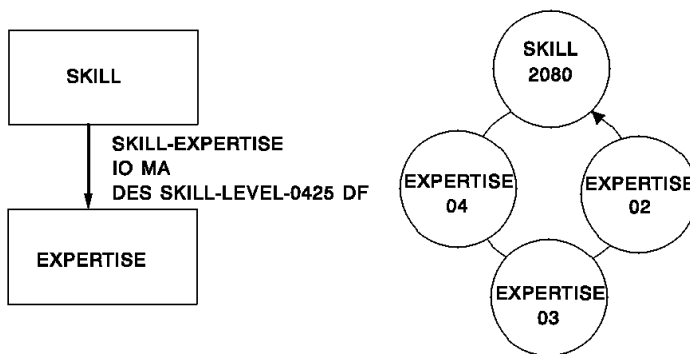
- *view-id*—The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.
- *record-name (occurrence)*—The record in the path where the *field-name* occurs.
- If the *field-name* occurs in more than one record, use *record-name*.
- If the record occurs more than once in the same path, use *occurrence*. If a database record name is the same as a CA OLQ keyword, the name should be enclosed in quotation marks.

Considerations:

If *criteria* is respecified for the REPEAT command, the original selection criteria clause (including the sort key value) is replaced and a REPEAT WITHIN SET command is executed without an implied SORTKEY clause. Therefore, if the selection criteria have been changed and sort key retrieval optimization is preferred, the sort key value must always be used in a REPEAT command.

Examples:

The following examples illustrate the use of the FIND/GET WITHIN SET using SORTKEY and associated REPEAT commands, based on the set occurrence diagram shown below:



Get Using Storage Key

A GET using storage key command is used to enter the database and establish currency for the SKILL record:

get first skill where calckey = 2080

```
SKILL
SKILL-DBKEY      : 0/5007116:1
SKILL-ID-0455   : 2080
SKILL-NAME-0455 : 'RPGII      '
SKILL-DESCRIPTION-0455 : '          '
END OF RECORD
```

Get Within Set

A GET WITHIN SET using SORTKEY command retrieves the first EXPERTISE record with a sort key value of 03:

```
get first expertise in skill-expertise set  
where sortkey = 03
```

```
EXPERTISE  
  EXPERTISE-DBKEY : 0/5007055:17  
  SKILL-LEVEL-0425 : '03'  
  EXPERTISE-DATE-0425 :  
  EXPERTISE-YEAR-0425 : 70  
  EXPERTISE-MONTH-0425 : 10  
  EXPERTISE-DAY-0425 : 10  
END OF RECORD
```

Repeat With First

A REPEAT command retrieves the next EXPERTISE record for which the EXPERTISE-YEAR-0425 field value is greater than or equal to 70:

```
repeat with first expertise  
where expertise-year-0425 ge 70
```

```
EXPERTISE  
  EXPERTISE-DBKEY : 0/5007026:6  
  SKILL-LEVEL-0425 : '04'  
  EXPERTISE-DATE-0425 :  
  EXPERTISE-YEAR-0425 : 72  
  EXPERTISE-MONTH-0425 : 1  
  EXPERTISE-DAY-0425 : 28  
END OF RECORD
```

Repeat

A REPEAT command specifies the 03 sort key again, to achieve sort-key optimization:

```
repeat with first expertise where sortkey = 03  
and expertise-year-0425 ge 70
```

```
EXPERTISE  
  EXPERTISE-DBKEY : 0/5007055:17  
  SKILL-LEVEL-0425 : '03'  
  EXPERTISE-DATE-0425 :  
  EXPERTISE-YEAR-0425 : 70  
  EXPERTISE-MONTH-0425 : 10  
  EXPERTISE-DAY-0425 : 10  
END OF RECORD
```

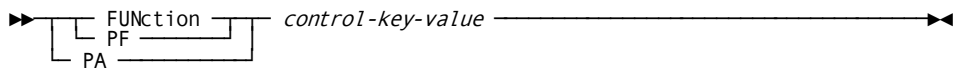
For more information:

[Coding Considerations](#) (see page 29)
[Global Syntax](#) (see page 35)

FUNCTION

FUNCTION permits you to invoke a control key value from a terminal that doesn't have control keys.

Syntax:



Parameters:

control-key-value

Specifies the control key that is invoked. Valid values are [PA1], [PA2], and 1 through 99 (corresponding to PF keys 1 through 99).

Note: When a control key that is not currently associated with a line command is invoked, CA OLQ responds as if the [Enter] key was pressed.

Example:

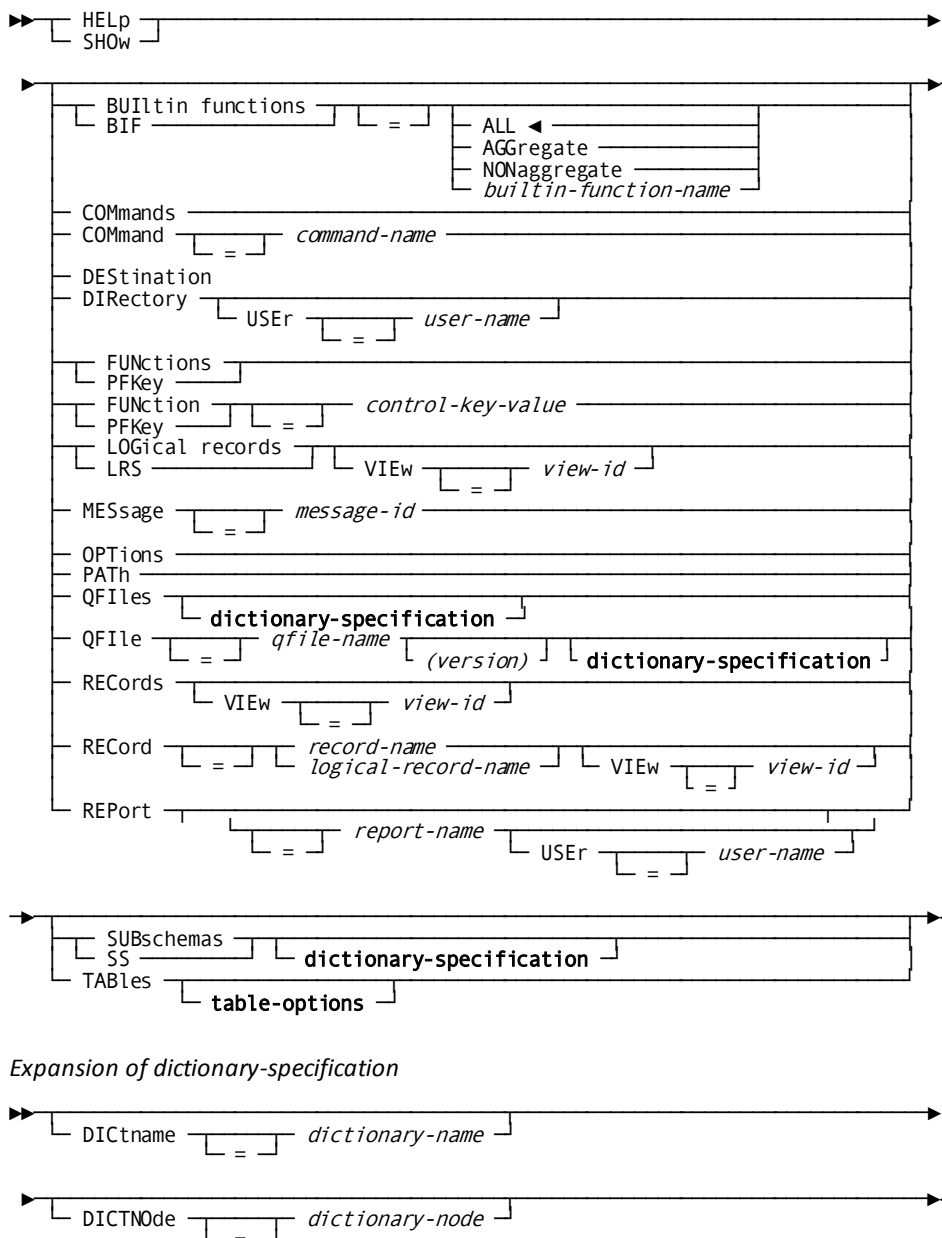
In the following example, the function associated with the PF10 key is invoked from the command line:

```
function 10
```

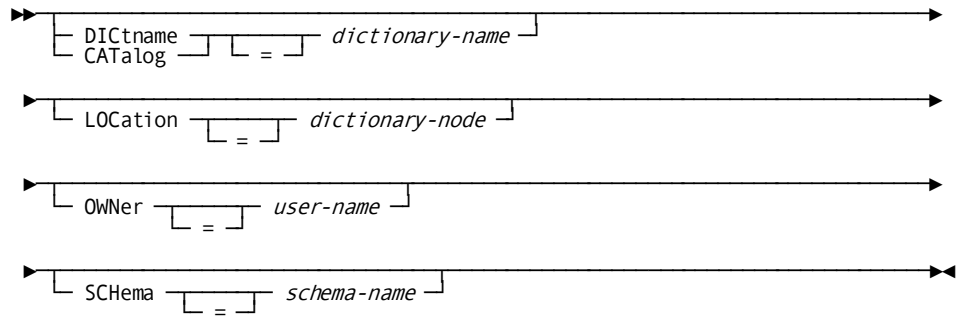
HELP

HELP (SHOW) explains CA OLQ commands and provides information about the data accessed by the current subschema. HELP and SHOW are synonymous.

Access mode: Some HELP parameters are **invalid** when the access switch is set to **IDMS**.

Syntax:

Expansion of table-options



Parameters:

BUILTIN FUNCTIONS

Lists information about built-in functions:

- **ALL**—Lists all the built-in function commands.
- **AGGREGATE**—Lists all the aggregate functions.
An aggregate function is one whose argument includes one or more columns and operates on one or more values in each column. A few examples are SUM, AVERAGE, and COUNT.
- **NONAGGREGATE**—Lists all the nonaggregate functions.
A nonaggregate function is one whose argument includes a single value within a column and only operates that value. A few examples are LOG10, MODULO, and ABSOLUTE-VALUE.
- *builtin-function-name*—Lists information about a particular built-in function.

COMMANDS

Lists the syntax and definitions of all CA OLQ commands.

COMMAND= command-name

Lists the syntax for the specified CA OLQ command.

Example:

HELP SIGNON lists syntax for the SIGNON command.

DESTINATION

Lists printer classes and destinations defined for the CA IDMS/DC or CA IDMS UCF system under which CA OLQ is executing. These printer classes and destinations can be specified in the PRINT command.

DIRECTORY USER= user-name

Lists all saved CA OLQ report files for a specified user. USER=*user-name* is an optional qualifier; *user-name* defaults to the current user. You must have a browse passkey for the specified user in order to see the directory.

FUNCTIONS

Lists CA OLQ commands associated with control keys for command mode.

FUNCTION= control-key-value

Displays the CA OLQ command associated with the named control key.

LOGICAL RECORDS

Displays the following information:

Access mode: This parameter is **invalid** when the access switch is set to **IDMS**.

- All logical records defined in the current subschema associated with an OBTAIN PATH command
- Any keywords associated with each logical record's OBTAIN path
- Any comments associated with each logical record
- VIEW=*view-id*—The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

MESSAGE= message-id

Lists detailed information about the CA OLQ error message identified by *message-id*.

OPTIONS

Lists all session options, indicating the current setting and the alternate setting for each option.

PATH

Displays the sequence of CA OLQ retrieval commands for the current path definition, including the WHERE criteria.

Access mode: This parameter is **invalid** when the access switch is set to **IDMS**.

QFILES

In secure installations, lists all qfiles in the data dictionary associated with the signed on user. In installations that do not use CA OLQ security features, lists all qfiles defined in the data dictionary.

- *dictionary-specification*—Specifies the dictionary and dictionary node containing the named qfile.

QFILE= *qfile-name*

Displays the CA OLQ commands that make up the named qfile.

- *(version)*—Specifies the version number of the named qfile.
- *dictionary-specification*—Specifies the dictionary and dictionary node containing the named qfile.

RECORDS

Lists all database record types and set relationships defined in the current subschema.

Access mode: This parameter is **invalid** when the access switch is set to **IDMS**.

- *VIEW=view-id*—The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

RECORD=

Lists subschema fields and their usage.

Access mode: This parameter is **invalid** when the access switch is set to **IDMS**.

- *record-name*—Specifies the data record type.
- *logical-record-name*—Specifies the logical record type.
 - *VIEW=view-id*—The qualifying ID for the record or logical record name. Use *view-id* when you are signed on to multiple subschemas. *View-id* refers to the subschema where the record (or logical record) can be found.

REPORT= *report-name*

Displays a detailed description of the named report. When not specified, *report-name* defaults to the name of the current report.

- *USER=user-name*— Optional qualifier that allows a user to view report files saved by other users. This parameter is effective only if the user has been assigned the passkeys necessary for accessing other user's reports.

SUBSCHEMAS

Lists the subschemas available to the current user.

- **DICTNAME= *dictionary-name***—Specifies the dictionary containing the specified subschemas. DICTNAME can be used to list information saved in other dictionaries without requiring you to sign on to them.
- **DICTNODE= *dictionary-node***—Specifies the dictionary node. DICTNODE can be used to list information saved in other dictionaries without requiring you to sign on to them.
- ***dictionary-specification***—Specifies the dictionary and dictionary node containing the named qfile.

TABLES

Lists the tables saved by the current or named user.

When the access switch is set to **idms**, CA OLQ only lists SQL schema entries to which the user has access.

Note: For more information about access switch, see [SET](#) (see page 186) later in this chapter.

table-options

Explicitly identifies where the list of tables originates:

- **DICTNAME/CATALOG= *dictionary-name*** —Specifies the dictionary under which the tables are saved.
- **LOCATION= *dictionary-node*** —Specifies the dictionary node name.

Access mode: This parameter is **invalid** when the access switch is set to **IDMS**.

- **OWNER= *user-name***—Allows you to view the ASF tables of a specific user. To view another user's tables, you must have BROWSE passkey authorization for the user's directory.

Access mode: This parameter is **invalid** when the access switch is set to **IDMS**.

Note: For further information on passkeys, see &U\$ICMSAD.

- **SCHEMA= *schema-name***— Allows you to view the SQL tables and views associated with the schema-name.

Access mode: This parameter is **invalid** when the access switch is set to **OLQ**.

Examples:*HELP DESTINATION*

The following HELP DESTINATION command lists available printer classes and destinations:

```
help destination
PRINT CLASSES/DESTINATIONS      PAGE 1.1
                                  LINE 1
(LTERM) (CLASS/DESTINATION)

PRT7026 CLASS=06
PRT7025 CLASS=63
PRT7024 CLASS=01
PRT7023 CLASS=05
PRT7022 CLASS=60
PRT7021 CLASS=02
PRT7020 CLASS=23
PRT7019 CLASS=25
PRT7018 CLASS=54
PRT7017 CLASS=53
PRT7016 CLASS=52
PRT7015 CLASS=51
PRT7014 CLASS=11
PRT7013 CLASS=02
PRT7012 CLASS=02
PRT7011 CLASS=02,04
PRT7010 CLASS=24
PRT7009 CLASS=07,21
          DEST=01,02,03,04,05
```

HELP QFILES

In the example below, a HELP QFILES command displays the qfiles available to the user currently signed on:

```
help qfiles
AVAILABLE QFILES      PAGE 1.1
                      LINE 1
(QFILE NAME)

DICTIONARY NAME *DEFAULT*
DICTIONARY NODE *DEFAULT*

SRKITEM(1)
CNUMORD(1)
CNAMORD(1)
CUSTITEM(1)
ORDITEM(1)
EMP-REPT-01(100)
EMP-REPT(100)
EMP-REPT(1)
EMP-PROF(1)
DENTAL(1)
EMP-SKILL(1)
```

HELP RECORDS

The following HELP RECORDS command displays all database record types and set relationships defined in the current subschema:

```
help records
```

```

                RECORDS IN EMPSS01          PAGE 1.1
                (RECORD NAME)          (OWNS SETS) (MEMBER OF SETS)
                LINE 1
STRUCTURE
VIA MANAGES                                MANAGES
                                           REPORTS-TO

SKILL
CALC: SKILL-ID-0455          SKILL-EXPERTISE SKILL-NAME-NDX

OFFICE
CALC: OFFICE-CODE-0450      OFFICE-EMPLOYEE

NON-HOSP-CLAIM
VIA COVERAGE-CLAIMS              COVERAGE-CLAIMS

JOB
CALC: JOB-ID-0440          JOB-EMPOSITION JOB-TITLE-NDX

INSURANCE-PLAN
CALC: INS-PLAN-CODE-0435
PF 08

```

```

                RECORDS IN EMPSS01          PAGE 2.1
                (RECORD NAME)          (OWNS SETS) (MEMBER OF SETS)
                LINE 18
HOSPITAL-CLAIM
VIA COVERAGE-CLAIMS              COVERAGE-CLAIMS

EXPERTISE
VIA EMP-EXPERTISE              EMP-EXPERTISE
                                           SKILL-EXPERTISE

EMPOSITION
VIA EMP-EMPOSITION              EMP-EMPOSITION
                                           JOB-EMPOSITION

EMPLOYEE
CALC: EMP-ID-0415          EMP-COVERAGE DEPT-EMPLOYEE
                EMP-EMPOSITION EMP-NAME-NDX
                EMP-EXPERTISE OFFICE-EMPLOYEE
                MANAGES
                REPORTS-TO

DEPARTMENT
                DEPT-EMPLOYEE

```

HELP RECORD=EMPLOYEE

The following `HELP RECORD=EMPLOYEE` command displays the subschema fields and descriptions for the `EMPLOYEE` database record:

```

help record=employee

                EMPLOYEE      PAGE 1.1
                LINE 1
(LLEVEL)      (FIELD NAME)    (USAGE) (PICTURE)

02  EMP-ID-0415      DISPLAY  9(4)
02  EMP-NAME-0415    GROUP
03  EMP-FIRST-NAME-0415  DISPLAY X(10)
03  EMP-LAST-NAME-0415  DISPLAY X(15)
02  EMP-ADDRESS-0415   GROUP
03  EMP-STREET-0415    DISPLAY X(20)
03  EMP-CITY-0415     DISPLAY X(15)
03  EMP-STATE-0415    DISPLAY X(2)
03  EMP-ZIP-0415      DISPLAY X(5)
02  EMP-HOME-PHONE-0415  DISPLAY 9(10)
02  STATUS-0415       DISPLAY X(2)
02  SS-NUMBER-0415    DISPLAY 9(9)
02  START-DATE-0415   GROUP
03  START-YEAR-0415   DISPLAY 9(2)

```

HELP SUBSCHEMAS

The following `HELP SUBSCHEMAS` command lists the subschemas available to the current user:

```

help subschemas dictname=docanwk

AVAILABLE SUBSCHEMAS  PAGE 1.1
LINE 1
(SCHEMA)      (SUBSCHEMA)

DICTIONARY NAME  DOCANWK
DICTIONARY NODE  *DEFAULT*

EMPSCHM(2)      EMPSS01
EMPSCHM(100)    EMPSS01
                  EMPSS09
                  NEWVIEW
                  OLDVIEW
TEST(100)       SUBTEST

```

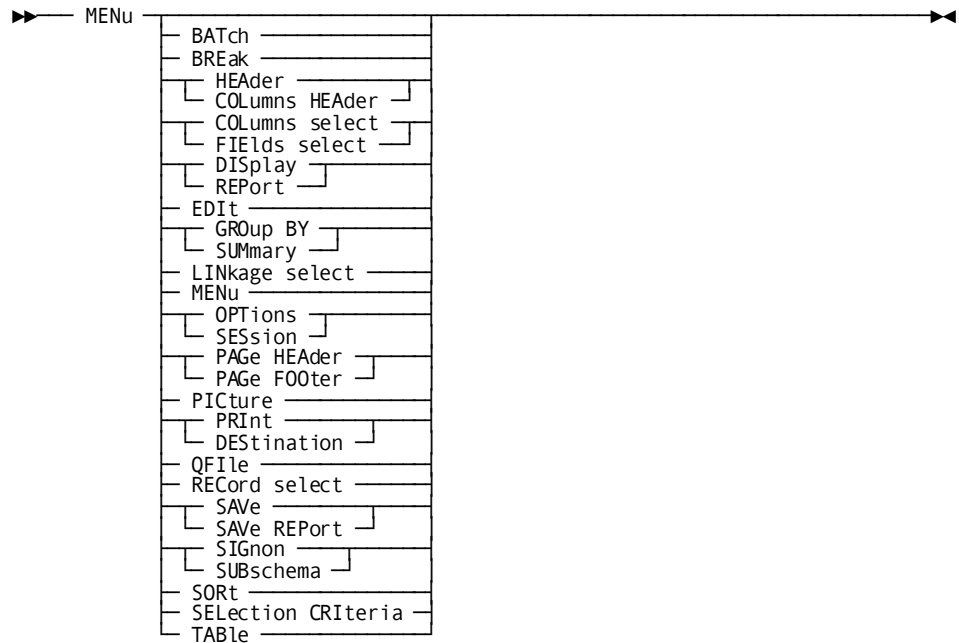
For more information:

[Built-In Functions and Syntax](#) (see page 209)

MENU

MENU allows you to switch from CA OLQ command mode to the menu facility.

Syntax:



Parameters:

BATCH

The Batch Processing screen

BREAK

The Report Format - Sort screen

HEADER/COLUMN HEADER

The Report Format - Header screen

COLUMNS SELECT/FIELDS SELECT

The Column Select screen

TABLE

The Table Processing screen

DISPLAY/REPORT

The Display Report screen

EDIT

The Report Format - Edit screen

GROUP BY/SUMMARY

The Report Format - Group By screen

LINKAGE SELECT

The Linkage Select screen

MENU

The Menu screen

OPTIONS/SESSION

The Session Options screen

PAGE HEADER/PAGE FOOTER

The Page Header/Footer screen

PICTURE

The Report Format - Picture screen

PRINT/DESTINATION

The Print Processing screen

QFILE

The Qfile Processing screen

RECORD SELECT

The Record Select screen

SAVE/SAVE REPORT

The Save Report screen

SIGNON/SUBSCHEMA

The Signon Database View screen

SORT

The Report Format - Sort screen

SELECTION CRITERIA

The Selection Criteria screen

Example:

The command MENU brings you to the Menu screen:

```

CA, Inc.
CA OLQ Release 16.0
*** Menu ***
-> Page 1 of 3
122000 Select an option and press the ENTER key

  Select
Pfkey Option Description Command/ Screen Name Show Help
-----
      X ---> Data Source for Report <---
      - Choose tables TABLE -
      - Choose subschema SUBSchema -

      ---> Retrieval Activity <---
      - Choose records from selected subschema RECOrd -
      - Choose columns for report COLumn -
      - Retrieve data to build report RETRIeve -
      - Alter database access strategy LINKage -

      ---> Processing Mode <---
      - Execute or create a predefined routine QFIle -
      - View existing or save current report SAVE -
      - Submit batch report request BATCH -

1=HELP 2=GLOBAL HELP 3=QUIT 4=MESSAGE 8=FWD

```

The command MENU OPTIONS brings you to the Options screen:

```

CA OLQ Release 16.0
*** Session Options ***
-> Page 1 of 2
107017 CA OLQ Release 16.0
107019 Copyright(C) 2003 CA, Inc.
Current interrupt count: 100 Current underline character: -
Access IDMS SQL tables: Y (Y/N) Current SQL NULL data value: .

User options: Page Columns Spread: (L-Left,E-Even,M-Max,nn)
      Help Change Option Current option Alternate option
-----
      - - NOFiller FILLer
      - - FULL SPArse
      - - HEAder NOHeader

-> Report Processing Options <-
      - - NOFiller FILLer
      - - FULL SPArse
      - - HEAder NOHeader

-> Column Processing Options <-
      - - OLQheader NOOLqheader
      - - PICTure NOPIcture
      - - CODetable NOCODetable

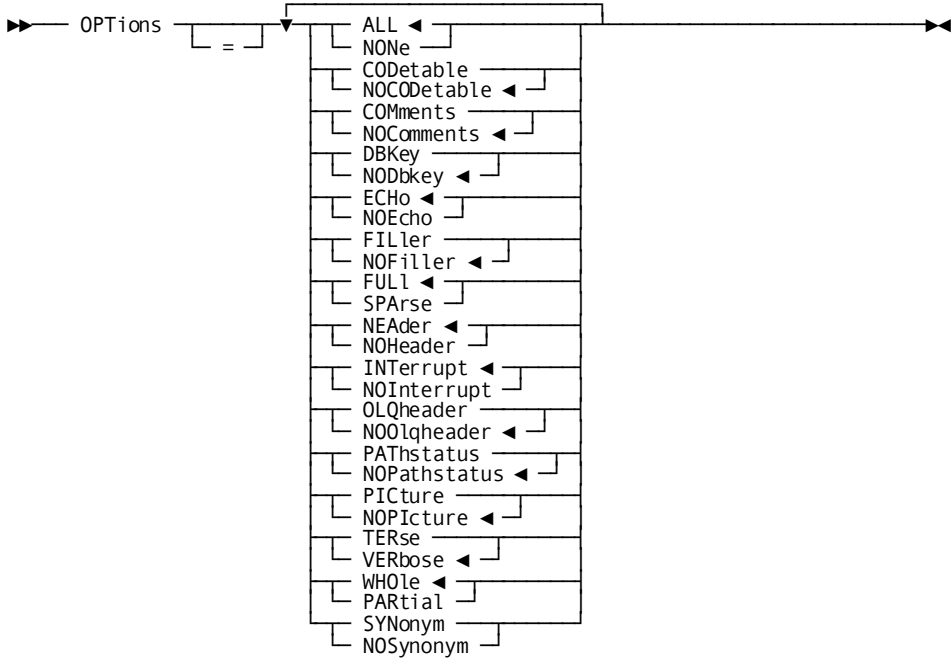
1=HELP 3=QUIT 4=MESSAGE 6=MENU 8=FWD

```

OPTIONS

OPTIONS sets default options for a CA OLQ session.

Syntax:



Parameters:

ALL/NONE

Specifies whether the default internal field list contains all or none of a record's fields.

The signon default for ALL/NONE remains in effect for all records retrieved in a session until changed by a user-specified FIELDS FOR command or by a user-specified field list in a FIND/GET command.

Although the default value of ALL is usually taken, it is recommended that, for lengthy records, the internal field list be limited to only those fields the user requires. The field list can be restricted by specifying OPTIONS=NONE followed by either a FIELDS FOR command or a *field-list* parameter for each record retrieved.

CODETABLE/NOCODETABLE

Specifies whether code tables are used to translate internal codes in one or more report columns into a screen display format.

OPTIONS=CODETABLE requests that code tables be applied to report columns. If a stand-alone table is specified with the EDIT command, CA OLQ uses that table to format the named report column. Otherwise, if a built-in table exists for the field in the data dictionary, this table is used for formatting.

OPTIONS=NOCODETABLE requests that no code table formatting be applied to report fields.

When you issue HELP REPORT, CA OLQ lists all code tables currently associated with report columns.

COMMENTS/NOCOMMENTS

Specifies whether comments are included in the HELP report files built by the HELP RECORDS, HELP SUBSCHEMAS, and HELP QFILES commands.

DBKEY/NODBKEY

Specifies whether dbkey columns are included in report files built by multiple record retrievals. Database keys qualified by page group are only included in a report file if the DBKEY option is in effect when the report file is initially created.

Note: There is a significant storage overhead associated with retrieving a large number of records with the DBKEY option on.

ECHO/NOECHO

Specifies whether the user-entered command is repeated by CA OLQ on the 3270-type output device.

FILLER/NOFILLER

Specifies whether filler field values will be available to the user. If you specify **OPTIONS=NOFILLER** (default), filler fields aren't displayed and don't have to be specified along with other fields in selection criteria.

If you specify **OPTIONS=FILLER**, filler fields are displayed and have to be specified in selection criteria for a group. Filler fields can only be accessed when **OPTIONS=FILLER** is in effect. Once a record is retrieved, changing the FILLER option has no effect on the internal field list.

FULL/SPARSE

Specifies how the format of displayed path retrieval report lines. FULL displays data associated with a record type once for each retrieved occurrence.

SPARSE, used with a SELECT command, displays only the first of a repeating data value; SPARSE, used with a path command, displays only the first of a repeating record type.

HEADER/NOHEADER

Specifies whether the report built by multiple record retrievals will be displayed with a header line.

INTERRUPT/NOINTERRUPT

Sets or disables the processing interrupt feature for multiple record retrievals. INTERRUPT limits the number of retrievals that can occur without intervention. The limit is established during system generation. When interrupt occurs, you can use the OPTIONS command to change processing control before issuing a YES response to the question of Continue processing?

Example

If **OPTIONS=NOINTERRUPT** is specified before a YES response, all remaining records are retrieved regardless of the interrupt level.

When **OPTIONS=NOINTERRUPT** is specified, the entire retrieval is executed with no interruptions. We recommend, however, that the default value of INTERRUPT be chosen to permit interruption of processing at known intervals. Users who regularly execute extensive retrievals and who do not wish to monitor CA OLQ for the duration of command execution may elect to enlarge the interrupt count at installation time or change it using the SET INTERRUPT COUNT command.

Note: Authorization for a user to select the NOINTERRUPT option is enabled through the data dictionary ADD USER DDDL statement. For more information about ADD USER DDDL statement and on CA OLQ security, see [Assigning Authority to Access CA OLQ](#) (see page 297).

OLQHEADER/NOOLQHEADER

Specifies whether field names or user-specified CA OLQ headers will be used as headers for displayed data.

If you specify **OPTIONS=OLQHEADER**, field names are replaced with associated CA OLQ headers if any are defined in the data dictionary or if any are defined by the user. If you specify **OPTIONS=NOOLQHEADER** (the default), field names are used as report headers.

PATHSTATUS/NOPATHSTATUS

Specifies the conditions for logical record retrieval. Path statuses are issued during execution of logical record paths. Path statuses are 1 to 32-character strings. Path statuses can be either standard or defined by the DBA in the subschema. The three standard path status values are:

- LR-FOUND—Returned when the logical record request has been successfully executed
- LR-NOT-FOUND—Returned when the specified logical record cannot be found, either because no such record exists or because all such occurrences have already been retrieved
- LR-ERROR—Returned when an error occurs in the processing of the logical record path

When **OPTIONS=NOPATHSTATUS** is in effect, CA OLQ retrieves a logical record only when the path status is LR-FOUND. If a DBA-defined path status is returned, CA OLQ treats it as if LR-ERROR were the path status. No report file is generated and CA OLQ displays the following message:

```
OLQ 095017 08 ERROR OCCURRED WHILE ATTEMPTING TO RETRIEVE  
A LOGICAL RECORD.  PATH STATUS=dba-defined-path-status
```

The path status can be displayed or printed; the field name is PATH STATUS. PATH STATUS is automatically added to the internal field list whenever OPTIONS=PATHSTATUS is specified.

Note: To refer to the PATH STATUS field, enclose it in quotation marks ('PATH STATUS'). For further information on the status of logical record paths, see *CA IDMS Logical Record Facility Guide*.

PICTURE/NOPICTURE

Specifies whether external pictures or default pictures are used to edit report fields. **OPTIONS=PICTURE** requests that external pictures be used to format report fields. If an external picture has been constructed with the EDIT command, this picture edits the named report field. Otherwise, if an external picture exists for the field in the data dictionary, the stored picture is used for formatting. When **OPTIONS=PICTURE** is in effect, user-specified pictures override external pictures defined for fields in the data dictionary.

OPTIONS=NOPICTURE requests that default pictures be used to edit report fields. Default pictures are derived from the internal pictures defined for fields in the data dictionary.

When a HELP REPORT command is issued, external pictures for report fields are listed.

TERSE/VERBOSE

Specifies the amount of identifying information provided when ON BREAK computations are requested.

When multiple computations are requested for a single output command, it is often difficult to determine which breaks and computations apply to which fields.

VERBOSE provides complete field and record names for all breaks and computations. **TERSE** allows more information to be displayed on the terminal screen but this information may be more difficult to read.

WHOLE/PARTIAL

Specifies the content of displayed path retrieval report lines.

If **OPTIONS=WHOLE** is in effect, only those rows containing a retrieved occurrence for every record type specified in the path definition are displayed.

If **OPTIONS=PARTIAL** is in effect, all lines retrieved are displayed even if the line doesn't contain each of the records specified in the path.

SYNONYM/NOSYNONYM

Specifies that synonyms for record names are (or aren't) defined in the data dictionary and can (or can't) be used in place of record names in CA OLQ. This option only applies to the record and column screens of menu mode CA OLQ.

Examples:

Options = Nocodetable

When **OPTIONS=NOCODETABLE** is in effect, stored values are displayed for the **START-MONTH-0415** field:

```
options = nocodetable !
select emp-last-name-0415 start-year-0415 start-month-0415
from employee ! edit start-month-0415 codetable montab
```

EMPLOYEE REPORT		
09/21/99		
EMP- LAST- NAME-0415	START- YEAR-0415	START- MONTH-0415
LINGER	77	12
TERNER	82	5
LINGER	78	5
PENMAN	77	9
LINGER	78	1
LITERATA	80	9
WILCO	79	11
HEAROWITZ	81	9
TYRO	80	12
KAHALLY	79	9
PAPAZEUS	78	9
ARM	77	12
KING	80	8
CLOUD	77	3

- 1 -

Options = Codetable

When OPTIONS=CODETABLE is in effect, a decoded value is substituted for each encoded value found for the START-MONTH-0415 field in the report file:

```
options = codetable !
select emp-last-name-0415 start-year-0415 start-month-0415
from employee ! edit start-month-0415 codetable montab
```

EMPLOYEE REPORT 09/21/99		
EMP- LAST- NAME -0415	START- YEAR- 0415	START- MONTH- 0415
LINGER	77	DECEMBER
TERNER	82	MAY
LINGER	78	MAY
PENMAN	77	SEPTEMBER
LINGER	78	JANUARY
LITERATA	80	SEPTEMBER
WILCO	79	NOVEMBER
HEAROWITZ	81	SEPTEMBER
TYRO	80	DECEMBER
KAHALLY	79	SEPTEMBER
PAPAZEUS	78	SEPTEMBER
ARM	77	DECEMBER
KING	80	AUGUST
CLOUD	77	MARCH

- 1 -

Options = Dbkey

When OPTIONS=DBKEY is in effect, a database key column qualified by page group appears in any report file that is built by retrieval of more than one record:

```
opt dbkey!
get all sequential department ! display
```

When OPTIONS=NODBKEY is in effect, no column for database key values is displayed:

```
opt nodbkey! get all sequential department
```

DEPT-ID-0410	DEPARTMENT	DEPT-NAME-0410	PAGE 1.1 LINE 1
6666	EXECUTIVE ADMINISTRATION		
2000	ACCOUNTING AND PAYROLL		
1010	EXECUTIVE WEATHER MANAGEMENT		
1011	WEATHER MANAGEMENT REPORTING		
1000	PERSONNEL		
3124	WEATHER REPORTING DIV.		
3100	INTERNAL SOFTWARE		
3125	ORDERING DEPARTMENT (3)		
5300	BLUE SKIES		
3200	COMPUTER OPERATIONS		
3121	WEATHER INFORMATION SERVICES		
4000	PUBLIC RELATIONS		
5100	BRAINSTORMING		

Options = Full

When OPTIONS=FULL is in effect, all information associated with the record type is displayed:

```
options=full !
select dept-id-0410, emp-last-name-0415 from department, employee
where dept-employee ! display
```

DEPARTMENT/EMPLOYEE REPORT
09/21/99

DEPT-ID-0410	EMP-LAST-NAME-0415
6666	HENDON
6666	PAPAZEUS
6666	RUPEE
6666	WILDER
2000	BLOOMER
2000	HUTTON
2000	JENSON
2000	KIMBALL
2000	KING
2000	NICEMAN
1000	FITZHUGH
1000	HEDGEHOG
1000	JOHNSON
1000	ORGRATZI

- 1 -

When OPTIONS=SPARSE is in effect, repeating column values are displayed only once:

```
opt=sparse !
select dept-id-0410, emp-last-name-0415 from department, employee
where dept-employee ! display
```

DEPARTMENT/EMPLOYEE REPORT
09/21/99

DEPT-ID-0410	EMP-LAST-NAME-0415
6666	HENDON
	PAPAZEUS
	RUPEE
2000	WILDER
	BLOOMER
	HUTTON
	JENSON
	KIMBALL
	KING
1000	NICEMAN
	FITZHUGH
	HEDGEHOG
	JOHNSON
	ORGRATZI

- 1 -

Options = Header

Output for OPTIONS=HEADER is shown below:

```

opt=header !
select dept-id-0410, emp-last-name-0415 from department, employee
where dept-employee ! display

      DEPARTMENT/EMPLOYEE REPORT
      09/21/99

DEPT-ID-0410      EMP-LAST-NAME-0415

      6666      HENDON
                PAPAZEUS
                RUPEE
      2000      WILDER
                BLOOMER
                HUTTON
                JENSON
                KIMBALL
                KING
      1000      NICEMAN
                FITZHUGH
                HEDGEHOG
                JOHNSON
                ORGRATZI

      - 1 -
    
```

Output for OPTIONS=NOHEADER is shown below:

```

opt=noheader !
select dept-id-0410, emp-last-name-0415 from department, employee
where dept-employee ! display

      6666      HENDON
                PAPAZEUS
                RUPEE
      2000      WILDER
                BLOOMER
                HUTTON
                JENSON
                KIMBALL
                KING
      1000      NICEMAN
                FITZHUGH
                HEDGEHOG
                JOHNSON
                ORGRATZI
      3100      PEOPLES
                DOUGH
                GALLWAY
                GARFIELD

      - 1 -
    
```

Options = Interrupt

In the following example, the `OPTIONS=INTERRUPT` is in effect and the retrieval limit before interruption is 25 records. After selecting 25 records, CA OLQ interrupts with a `CONTINUE` prompt. A `YES` or `RESUME` response directs CA OLQ to continue retrieving data until either the interrupt limit is reached again or processing is completed:

```
options= interrupt !
set interrupt=25
```

OLQ 092010 00 The interrupt count has been modified.

```
select dept-id-0410, emp-last-name-0415 from department, employee
where dept-employee
```

OLQ 098006 00 17 whole lines and 0 partial lines in report.
OLQ 098007 00 25 records read. 21 records selected.
OLQ 098008 00 17 of 47 primary record pages read.
OLQ 098009 00 Continue (yes/no)?

When `OPTIONS=NOINTERRUPT` is in effect, CA OLQ retrieves all records without further interruption:

```
options=nointerrupt !
select dept-id-0410, emp-id-0415, emp-zip-0415
from department, employee where dept-employee
```

OLQ 098006 00 57 whole lines and 0 partial lines in report.
OLQ 098007 00 75 records read. 66 records selected.

```
display cols 1,2,3
```

DEPARTMENT/EMPLOYEE REPORT		
10/08/99		
DEPT-ID-0410	EMP-ID-0415	EMP-ZIP-FIRST-FIVE-0415
6666	30	02198
6666	471	03256
6666	1	02312
6666	472	03145
2000	69	01675
2000	100	02176
2000	11	02176
2000	67	01239
2000	106	02176
2000	101	02176
1000	81	03458
1000	8683	10996
1000	51	02546
1000	91	06182

- 1 -

Options = No Olqheader

When OPTIONS=NOOLQHEADER is in effect, field names appear as report headers:

DEPARTMENT/EMPLOYEE REPORT	
09/21/99	
DEPT-ID-0410	EMP-LAST-NAME-0415
6666	HENDON
6666	PAPAZEUS
6666	RUPEE
6666	WILDER
2000	BLOOMER
2000	HUTTON
2000	JENSON
2000	KIMBALL
2000	KING
2000	NICEMAN
1000	FITZHUGH
1000	HEDGEHOG
1000	JOHNSON
1000	ORGRATZI

- 1 -

When OPTIONS=OLQHEADER is in effect, predefined CA OLQ headers are used:

DEPARTMENT/EMPLOYEE REPORT	
09/21/99	
DEPT NUMBER	EMPLOYEE NAME
6666	HENDON
6666	PAPAZEUS
6666	RUPEE
6666	WILDER
2000	BLOOMER
2000	HUTTON
2000	JENSON
2000	KIMBALL
2000	KING
2000	NICEMAN
1000	FITZHUGH
1000	HEDGEHOG
1000	JOHNSON
1000	ORGRATZI

- 1 -

Options = Picture

When OPTIONS=NOPICTURE is in effect, default pictures are used to format report fields:

```
DEPARTMENT/EMPLOYEE REPORT
09/21/99

DEPT-ID-0410          EMP-LAST-NAME-0415

6666                HENDON
6666                PAPAZEUS
6666                RUPEE
6666                WILDER
2000                BLOOMER
2000                HUTTON
2000                JENSON
2000                KIMBALL
2000                KING
2000                NICEMAN
1000                FITZHUGH
1000                HEDGEHOG
1000                JOHNSON
1000                ORGRATZI

- 1 -
```

When OPTIONS=PICTURE is in effect, external pictures are used to format report fields. In the following example, a dynamic external picture is specified with the EDIT command:

```
edit dept-id-0410 picture='9-999'
display

DEPARTMENT/EMPLOYEE REPORT
mm/dd/yy

DEPT-ID-0410          EMP-LAST-NAME-0415

6-666                HENDON
6-666                PAPAZEUS
6-666                RUPEE
6-666                WILDER
2-000                BLOOMER
2-000                HUTTON
2-000                JENSON
2-000                KIMBALL
2-000                KING
2-000                NICEMAN
1-000                FITZHUGH
1-000                HEDGEHOG
1-000                JOHNSON
1-000                ORGRATZI

- 1 -
```

For more information:

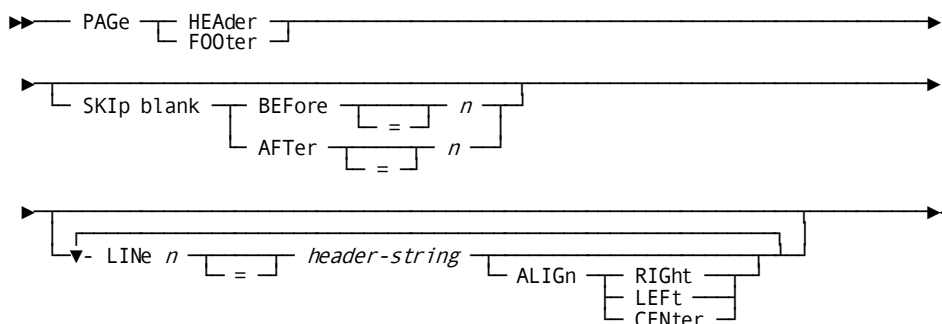
[Global Syntax](#) (see page 35)

PAGE HEADER / FOOTER

PAGE HEADER/FOOTER enables you to locate a user-specified report page header or footer where you want it on the page or the terminal screen.

Note: The page headers should not contain the column heading.

Syntax:



Parameters:

PAGE HEADER

Specifies that you want to include a user-specified page header in the report.

PAGE FOOTER

Specifies that you want to include a user-specified page footer in the report.

SKIP

Specifies that a user-specified number of lines are to be skipped between the header/footer and the report.

BEFORE= n

Specifies that *n* lines are skipped before the report page header/footer is inserted. *N* must be greater than zero and less than 10.

AFTER= n

Specifies that *n* lines are skipped after the report page header/footer is inserted. *N* must be greater than zero and less than 10.

LINE n= header-string

Specifies the line number and the contents of the header/footer. For example, the following command specifies that the first line of the report page header reads 'DEPARTMENT REPORT' and the second line of the report page header reads the date the report was built:

```
Line 1 = 'DEPARTMENT REPORT'
Line 2 = '$DATE'
```

ALIGN RIGHT/LEFT/CENTER

Specifies that the report page header/footer line is aligned on the right, left, or centered on the page.

Example:

This example demonstrates the commands necessary to include both a user-specified report page header and footer in a report:

```
page header skip before 1 skip after 2 line 1 ='fiscal report'  
line 2 ='$date' ! page footer skip before 2 skip after 1  
line 1 ='page $page' line 2 ='CA, Inc.' ! display
```

EMP-ID-0415	EMP- LAST- NAME-0415	SALARY- AMOUNT- 0420
9999	LINGER	38500.00
48	TERNER	13000.00
23	LINGER	42500.00
23	LINGER	38000.00
149	PENMAN	39000.00
15	LINGER	85000.00
15	LINGER	75000.00
35	LITERATA	37500.00
349	WILCO	80000.00

CA, Inc. PAGE 1

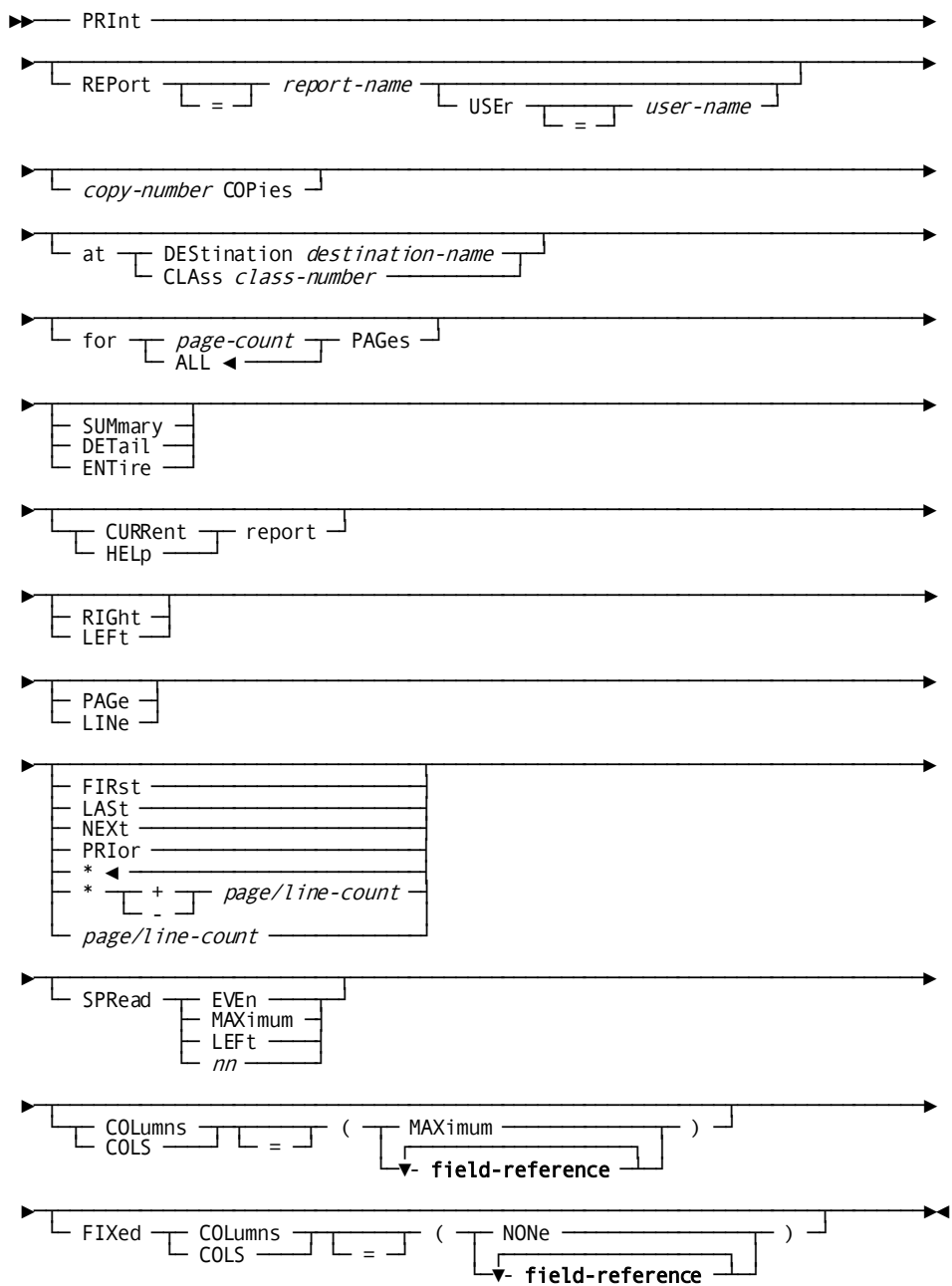
PRINT

PRINT directs a formatted CA OLQ report to a specific CA IDMS/DC printer for a hard copy.

Batch considerations:

PRINT is invalid when running local mode.

Syntax:



Parameters:

REPORT= report-name

Specifies the name of the report to be printed.

USER= user-name

Specifies the user whose report directory contains the named report file.

copy-number

Specifies the number of report copies printed. *Copy-number* is an integer in the range 1 through 255. The default value is 1.

DESTINATION destination-name

Specifies the printer destination the report is sent to. *Destination-name* can be any printer destination specified at CA IDMS/DC system generation. Use SHOW DESTINATION for a list of valid destinations.

CLASS class-number

Specifies the printer class the report file is sent to. *Class-number-n* is an integer in the range 1 through 64.

Note: If the DESTINATION or CLASS options are not used, the report is printed at the default printer defined at CA IDMS/DC system generation for the terminal in use.

page-count

Specifies the number of report pages printed. *Page-count* is an integer.

ALL

Specifies that all the report pages are printed.

SUMMARY

Specifies the printing of summary report lines only.

DETAIL

Specifies the printing of detail report lines only. If summary lines are present in the report file, they are not output.

ENTIRE

Specifies the printing of both detail and summary report lines.

CURRENT REPORT

Requests output of the last report displayed; a help report or a data report.

HELP REPORT

Requests output of the help report file built by the last HELP command.

Note: For more information, see [HELP](#) (see page 132), earlier in this chapter.

RIGHT/LEFT

Specifies horizontal movement within the report file.

PAGE/LINE

Requests the data in the report file be output relative to a designated line or page of the report file:

- PAGE requests the data in the report file be output beginning at the current or specified page number.
- LINE requests a page of the report file be output beginning at the current or specified line number.

The default is PAGE.

FIRST

Outputs a page of report file data, beginning at page 1, line 1.

LAST

Outputs the last page of report file data.

NEXT

Outputs a page of report file data, beginning at the page or line number immediately following the current page or line number.

PRIOR

Outputs a page of report file data, beginning at the page or line number immediately preceding the current page or line number.

*

Asterisk (*)—outputs the current page of report file data, beginning at the first line of that page.

* + -

Outputs a page of report file data beginning *n* pages or lines:

- Minus sign (-)—*before* the current page or line number.
- Plus sign (+)—*after* the current page or line number.
- Asterisk (*)—is a required character that explicitly references the current page or line.
- *page*—The number of pages
- *line-count*—The number of lines

page/line-count

Specifies the starting point of the output relative to the current page and line number. *Page/line-count* outputs a page of report file data, beginning at the specified page or line number.

SPREAD EVEN/MAXIMUM/LEFT *nn*

Specifies the space between the columns.

- **EVEN**—The same number of spaces between each column (Space the columns evenly).
- **MAXIMUM**—The maximum number of spaces between each column.
- **LEFT**—Displays columns starting in the left most position with one space separating each column.
- ***nn***—*nn* spaces between each column. Zero is not a valid number. The minimum number of spaces allowed is one.

COLUMNS

Specifies the columns included in the output and, optionally, the order of appearance and width of those columns. Column specifications remain in effect until altered by a subsequent PRINT command. Column specification is:

- **MAXIMUM**—Output of as many sequential columns, starting with column 1, as can appear on one page of the report. Excess columns are ignored, and no warning message is produced.
- *field-reference*—Output of specific columns and the number of characters in each column.

FIXED COLUMNS

Specifies the columns, and their order, to remain on the screen when paging left and right. The columns specified with this parameter precede the columns specified in the COLUMNS= parameter. They remain fixed on the left side of the report.

Column specifications remain in effect until altered by a subsequent PRINT command or a FIXED COLUMNS = NONE command.

- **NONE**—No report fields are fixed in the report. This cleans out the fixed columns list.
- *field-reference*—The columns and the number of characters in each column to be output.

Considerations:

Report files sometimes contain information that cannot be displayed. CA OLQ indicates data that cannot be displayed, as follows:

- The at sign (@) indicates an unprintable character. CA OLQ provides a translation function that handles all characters written to a terminal or to the print queue.

If you want to view the characters represented, you can use EDIT HEXADECIMAL to display the character in its hexadecimal representation.

Note: For an explanation of how to modify the CA OLQ translation table, see the CA IDMS installation guide for your operating system.

- The asterisk (*) indicates invalid data. The invalid data flag appears when data is not stored in the defined format or when a COMPUTE command yields invalid results (as with decimal overflows and division by zero).

Null character considerations

The null character is by default a period (.). You can override this by invoking the SET NULL command.

Note: For more information about the SET command see, [SET](#) (see page 186), later in this chapter.

Data retrieved in SQL tables can contain null values. To display them, CA OLQ pads the entire length of the display field with the null character.

Example:

This example demonstrates the commands necessary to print 2 copies of the employee table:

```
print dest=la copies=2 columns=maximum
```

For more information:

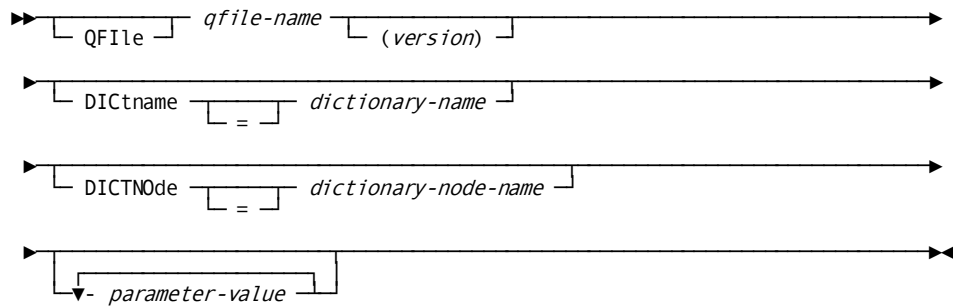
[Global Syntax](#) (see page 35)

QFILE

QFILE accesses CAOLQ command sequences stored in the data dictionary. When you specify QFILE *qfile-name*, all CAOLQ commands contained within the named qfile are automatically executed.

Note: For more information about qfiles, see [Using qfiles](#) (see page 268).

Syntax:



Parameters:

qfile-name

The name of the qfile to be executed. The named qfile must be saved in the data dictionary. CA OLQ interprets any nonreserved first word in a command as a qfile name.

(version)

Identifies the version number of the named qfile. The default is the highest version defined for the named qfile. If used, *version* must be enclosed in parentheses.

DICTNAME= dictionary-name

Identifies the dictionary where the named qfile is stored.

DICTNODE= dictionary-node- name

Identifies the dictionary node controlling the named dictionary.

parameter-value

Specifies one or more values to be substituted into variable parameters in the qfile at execution time.

Example:

Show Qfile

SHOW QFILE displays the REPORT qfile definition, which was previously saved:

```

show qfile report
REPORT                                PAGE 1.1                                LINE 1
DICTIONARY NAME      TSTDICT
DICTIONARY NODE      *DEFAULT*

SET DICTNAME TSTDICT
SIGNON SS EMPSS01  SCHEMA EMPSCHEM ( 100)
OPTIONS ALL HEADER ECHO NOFILLER FULL WHOLE INTERRUPT NOOLQHEADER -
NOPATHSTAT NOSTAT COMMENT VERBOSE NODBKEY NOPICTURE NOCODETAB NOSYN
SET ACCESS OLQ
SELECT EMP-LAST-NAME-0415 DEPT-NAME-0410 SALARY-AMOUNT-0420 -
FROM EMPLOYEE, DEPARTMENT, EMPPOSITION WHERE DEPT-EMPLOYEE AND -
EMP-EMPPOSITION
PAGE HEADER BLANK LINES AFTER 1 -
LINE 1 'DEPARTMENT/EMPLOYEE/EMPOSITION REPORT' CENTER -
LINE 2 '$DATE' CENTER
PAGE FOOTER BLANK LINES BEFORE 1 -
LINE 1 '-$PAGE -' CENTER
EDIT EMP-LAST-NAME-0415 -
ALIGN LEFT -
OLQHEADER 'THIS IS NAME  HEADER'
DISLAY SPREAD EVEN COLUMNS = EMP-LAST-NAME-0415 DEPT-NAME-0410 -
SALARY-AMOUNT-0420
END OF REPORT

```

Qfile Report

The above qfile is invoked by specifying the REPORT qfile name. The QFILE REPORT command builds a list of employee names, their departments and their salaries:

```

report
DEPARTMENT/EMPLOYEE/EMPOSITION REPORT
mm/dd/yy
DEPT-ID-0410  EMP-LAST-NAME-0415  SALARY-AMOUNT-0420
6666  HENDON  240000.00
6666  PAPAZEUS  100000.00
6666  MUNYOUN  90000.00
6666  RUMPLEST  80000.00
6666  RUPEE  76000.00
6666  WILDER  90000.00
2000  BLOOMER  15000.00
2000  HUTTON  44000.00
2000  JENSON  82000.00
2000  KIMBALL  45000.00
2000  KING  14500.00
2000  NICEMAN  14000.00

```

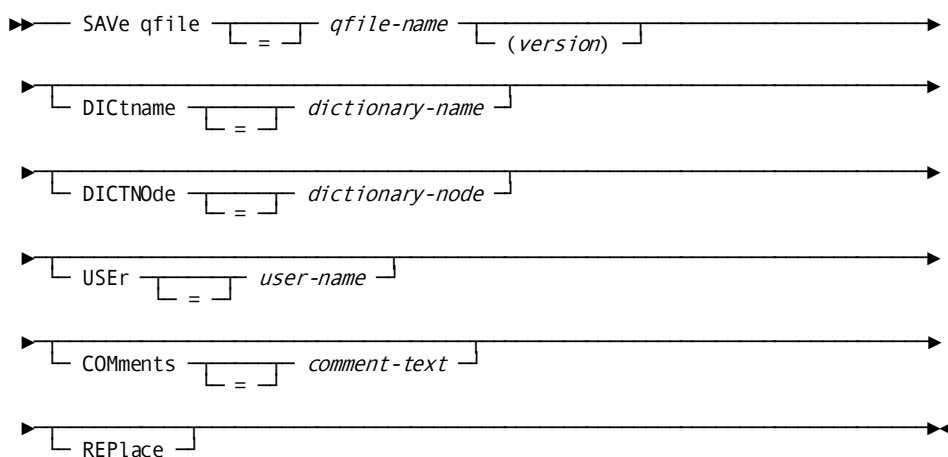
SAVE QFILE

SAVE QFILE stores the retrieval and report formatting commands necessary to recreate the current report. The qfile is saved into the data dictionary.

Batch considerations:

SAVE QFILE is invalid when running local mode.

Syntax:



Parameters:

qfile-name

The name of the qfile to be stored in the data dictionary.

(version)

The version number of the named qfile enclosed in parentheses. **Version** defaults to 1.

DICTNAME= dictionary-name

Specifies the dictionary where the named qfile is stored.

DICTNODE= dictionary-node

Specifies the dictionary node that controls the dictionary.

USER= user-name

Specifies the owner of the qfile.

COMMENTS= comments-text

Specifies comments to be included with the qfile.

REPLACE

Indicates the qfile already exists in the data dictionary and is to be replaced with the current path definition.

Example:

Save Qfile

The following qfile retrieves, formats, and then displays data:

```
options = sparse
select dept-id-0410,emp-id-0415,emp-name-0415,salary-amount-0420 -
  from department, employee, emposition
  where dept-employee and emp-emposition
compute &xq.average salary' = avg(salary-amount-0420) -
  group by dept-id-0410
display
```

Save Qfile

The SAVE QFILE command is used to save the path listed above as the EMP-SAL qfile:

```
save qfile emp-sal
```

```
109017 THE REQUESTED OPERATION FOR EMP-JOB(1) HAS SUCCESSFULLY COMPLETED
```

When the EMP-SAL qfile is executed, CA OLQ retrieves the ids, names, and salaries for all company employees.

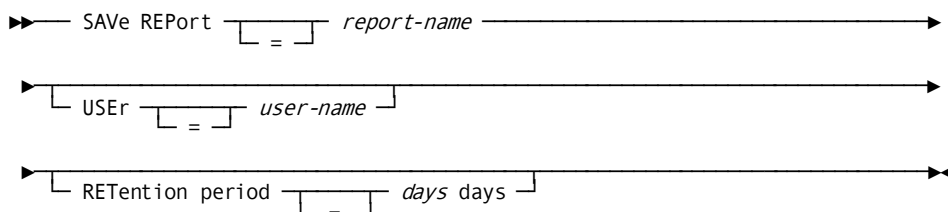
SAVE REPORT

SAVE REPORT stores a report (definition and data) so that it can be viewed at a later date.

Note: For the security associated with saved reports, see [Security for ASF tables](#) (see page 301).

Batch considerations

SAVE REPORT is invalid when running local mode.

Syntax:**Parameters:****report-name**

The 1 to 32-character name of the report to be saved.

USER= user-name

Specifies the user into whose directory the report is saved. If **user-name** is not specified, the report is saved under the current user name.

RETENTION PERIOD= days

Specifies the number of days for which the report is saved.

A default retention period and a maximum retention period are established at DC/UCF system generation. Reports are automatically deleted at the end of their associated retention periods.

Example:

The report file used for this example was initially created by execution of the following SELECT statement:

```

select office-code-0450,office-phone-0450(1),emp-id-0415
from office, employee
where office-employee

```

The following SAVE REPORT command stores the report, created by the SELECT statement shown above, in the data dictionary and associates the report with the name OFFICE#8. After the OFFICE#8 report has been stored, CA OLQ displays a message to indicate that processing has been completed successfully:

```
save report=office#8 retention period=10 days
```

```
OLQ 107001 00 The OFFICE#8 report has been saved successfully
```

This report can then be viewed at a later time using DISPLAY:

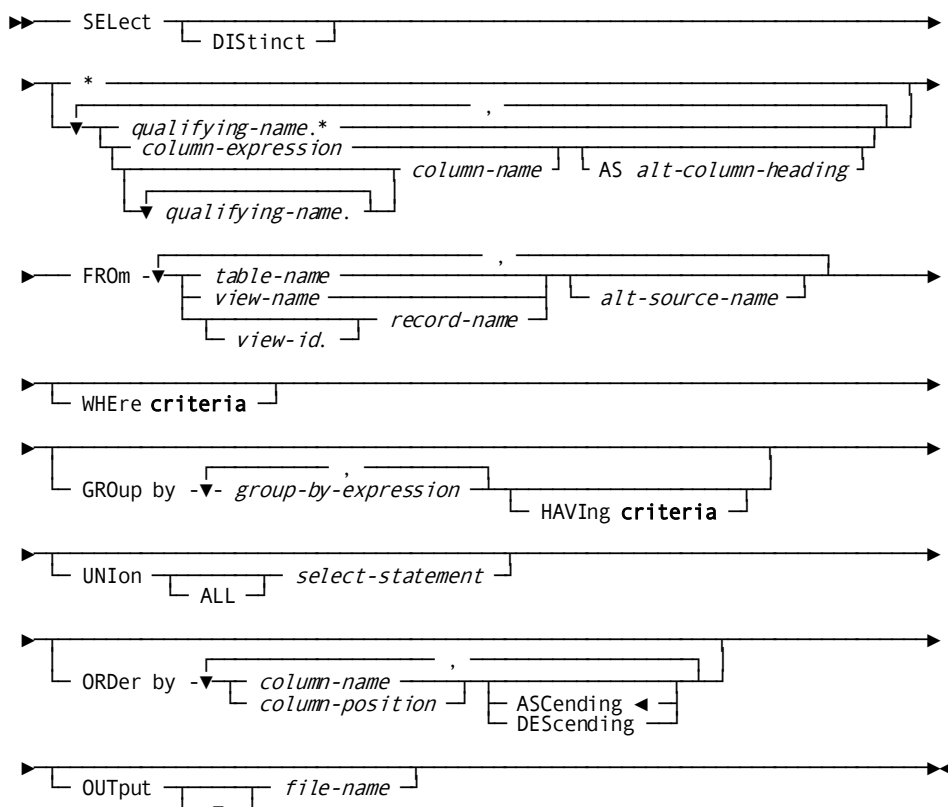
```
display report = office#8
```

SELECT—OLQ access mode

The SELECT statement in OLQ access mode retrieves data for display. A single retrieval request can combine data from multiple ASF tables, database records, logical records, subschemas, and sequential files. Sequential files are used for batch retrieval only.

Access mode: The syntax below is **invalid** when the access switch is set to IDMS.

Syntax:



Parameters:

DISTINCT

Eliminates all but one duplicate row occurrence from the report, based on specified columns.

- Asterisk (*)—Lists all the columns of the named tables or records in the report. The order in which the columns are listed in the component tables or records is the order in which they appear, from left to right, in the report.

- *qualifying-name*—A prefix denoting the table or record from which the column is being retrieved. **Qualifying-name** must be separated from its object by a period (.).

The qualifying name can modify an asterisk (*), requesting all columns of the named table. *Qualifying-name* can be one of the following:

- A stored table
- A table derived from one or more stored tables or network records (a view)
- A network database record
- An alternative table name or record name
- A group-level qualification of a record element

You can specify more than one qualifying name for a column.

- *column-expression*—The columns you want displayed in the report:

- A table column or record column
- A computed arithmetic expression, based on column values
- An aggregate function, modifying a column expression
- A built-in function, modifying a column expression
- An occurrence of a multiply-occurring field
- A fixed value, such as a literal string

Fixed value expressions containing blanks or special characters must be enclosed in quotation marks.

- *column-name*—The object of a qualifying table name. *Column-name* can be the name of a column in the object table or record, or a group-level qualification of a column.

AS alt-column-heading

Specifies an alternative column heading. You can use this to assign a column heading for any column reference, including computed fields or fields modified by a built-in or aggregate function.

Commas separating column specifications are optional, but recommended.

FROM table-name

Represents the table from which data is retrieved. Commas separating multiple table and record entries are required.

FROM view-name

Specifies the name of the table derived from one or more stored data tables or records.

FROM view-id

Specifies a user-supplied name identifying a particular subschema. *View-id* is defined by the VIEW operand of the SIGNON statement. This is used to qualify record names.

FROM record-name

Specifies the name of a record.

alt-source-name

Specifies an alternative name for the table or record.

WHERE criteria

Represents criteria used by SELECT to retrieve records.

Note: Syntax for *criteria* used in the WHERE clause are expanded in [SELECT Selection Criteria](#) (see page 35).

GROUP BY group-by-expression

Groups rows into sets that contain like column values. **Group-by-expression** represents a column value. In each GROUP BY set, all rows contain the same value of the *group-by-expression*.

The GROUP BY *group-by-expression* must be the same value as the column expression that is the object of the SELECT statement and can include aggregate and/or built-in functions.

HAVING criteria

Applies selection criteria to the result of the GROUP BY expression.

Note: Syntax and Purpose for *criteria* used in the HAVING clause are expanded in [SELECT Selection Criteria](#) (see page 35).

UNION ALL select-statement

Concatenates two or more SELECT statements containing like columns. Columns correspond positionally. Corresponding columns must have the same data type, picture, and decimal representation. **UNION** eliminates duplicate rows from the report.

- ALL—Retains duplicate rows in the report.
- *select-statement*—Concatenates two or more SELECT statements containing like columns.

ORDER BY

Sorts the rows of the report by the value of the column you specify. You can sort by:

- *column-name*—The columns being sorted in the report. You can specify any type of column expression in terms of its column position. You cannot specify *column-name* when any of the following types of column expressions are the object of the sort:
 - Computed fields
 - A column expression containing an AS parameter
 - The UNION operand
 - Subscripted fields (arrays)
- *column-position*—The position in the column-list, from left to right, of the column to be sorted. **Column-position** can refer to computed columns or to columns modified by built-in or aggregate functions.

ASCENDING/DESCENDING

Specifies the order in which to sort the columns.

OUTPUT= file-name

Specifies that the report is saved as a sequential file. This option is only valid when running CA OLQ batch.

Examples:

Select all columns

Instead of specifying all the columns of the table, or fields of the record that you want to retrieve, you can specify an asterisk (*) and receive all columns or fields:

```
select * from employee
```

The result report displays the first column of the EMPLOYEE table first, the second column of the EMPLOYEE table second, and so on.

Select column expression

You can specify the names of the columns or fields you want to retrieve explicitly:

```
select emp-name, dept-name from employee, department
```

The result report displays two columns, the employee name from the EMPLOYEE table, and the department name from the DEPARTMENT table.

Select Distinct

Distinct eliminates all but one duplicate row occurrence from the report, based on specific columns.

Example:

If there are two occurrences of the last name field BURR, you can eliminate the row containing the duplicate value:

```
select DISTINCT emp-last-name from employee ! display
```

```
EMP LAST NAME
-----
BURR
GOLD
ILTIS
LIGARE
WAKEFIELD
WONES
```

Select qualifying name

Qualifying-name is a prefix denoting the table or record from which the column is being retrieved. **Qualifying-name** must be separated from its object by a period (.). This is useful when two tables contain columns with the same name.

Example:

If both the EMPLOYEE and the DEPARTMENT tables contain DEPT-ID fields, you can qualify the column names:

```
select EMPLOYEE.dept-id, DEPARTMENT.dept-id
   from employee, department
  where employee.dept-id = department.dept-id
```

The qualifying name can also modify an asterisk (*).

You can also specify a **group-level qualification** as a qualifying name:

```
select EMP-NAME-0415.EMP-LAST-NAME-0415 from employee
```

Select two references

You can use the **AS** parameter to distinguish one of two references to the same column name:

```
select manage.emp-id-0415 AS MANAGER,
       works-for.emp-id-0415 AS EMPLOYEE
   from employee manages, employee works-for
```

Select and sort

A simple SELECT command retrieves the EMPLOYEE record specified with a column list and sorts the fields by last name:

```
select emp-id-0415,emp-name-0415
  from employee
 order by emp-last-name-0415 descending ! display
```

EMP-ID-0415	EMP-FIRST-NAME-0415	EMP-LAST-NAME-0415
0124	SUSAN	SPELLMAN
0528	EDWARD	MCCARTHY
0512	CHERYL	MAYOR
1042	SHARON	CIVITTOLO
0954	ANGELA	BELVAL
0320	JOSEPH	ANTHONY

Joining tables based on equal values

To retrieve hiring information on all employees in a department:

1. Specify a **SELECT** clause, listing the columns you want (EMP-LAST-NAME, START-YEAR, and DEPT-NAME).
2. Specify a **FROM** clause, naming the tables from which you are retrieving data (EMPLOYEE and DEPARTMENT).
3. Specify a **WHERE** clause, indicating join criteria linking the two tables. For example: the DEPT-ID from the EMPLOYEE table is equal to the DEPT-ID from the DEPARTMENT table.

This example lists employees, the year they started, and their department name:

```
select employee.emp-last-name, employee.start-year,
  department.dept-name
  from employee, department
 where employee.dept-id = department.dept-id ! display
```

EMP-LAST-NAME	START-YEAR	DEPT-NAME
WONES	79	DEVELOPMENT
WAKEFIELD	83	PERSONNEL
BURR	80	PERSONNEL
LIGARE	85	DEVELOPMENT
BURR	84	MARKETING
SCHLEY	80	PLANNING
ILTIS	81	PERSONNEL
GOLD	80	MARKETING

Joining two tables

Using a single SELECT statement, you can produce a report containing data from more than one table. The selection criteria in the WHERE clause provide column join and key information. The WHERE clause can contain other comparison expressions.

Example:

The following SELECT statement joins the EMPLOYEE and DEPARTMENT tables on like DEPT-ID values, and lists only those employees who started working before 1980:

```
select employee.*, department.*
  from employee, department
  where employee.dept-id = department.dept-id
        and employee.start-year lt '80'
```

You do not have to display the fields on which you are joining.

Example:

The following SELECT statement joins the EMPLOYEE and DEPARTMENT stored tables, but lists only the employee ID numbers and the name:

```
select distinct employee.emp-id, department.dept-name
  from employee, department
  where employee.dept-id = department.dept-id
```

Reflexive joins

Reflexive joins combine two different rows of the same table. When you are joining a table with itself, it is useful to supply alternative table names to distinguish the two references to the column name.

This example lists employees and their managers. EMPLOYEE MANAGE and EMPLOYEE WORKS-FOR are alternative names for the same record:

```
select works-for.emp-last-name as 'worker',
       manage.emp-last-name as 'manager'
from employee works-for, employee manage
where works-for.manager-id = manage.emp-id ! display
```

WORKER	MANAGER
WONES	WONES
WAKEFIELD	WAKEFIELD
BURR	WAKEFIELD
LIGARE	WONES
BURR	BURR
SCHLEY	WONES
ILTIS	WAKEFIELD
GOLD	BURR

Joining tables and records residing in multiple subschemas

This example uses a SELECT statement to create a report containing data from an ASF-generated table and a network table:

- **For ASF-generated tables**, you do not have to sign on to any subschemas before issuing your SELECT request. You should name the dictionary in which the table is stored.
- **For database records**, you must sign on to the corresponding subschemas with a SIGNON statement before you issue your SELECT request.

In this example, the EMPLOYEE table resides in the EMPSS01 subschema. The DEPARTMENT table is an ASF-generated table:

1. Sign on to the EMPSS01 subschema:


```
signon ss empss01 dictname testdict id=emp
```
2. Sign on to the TEST01 subschema:


```
signon table department dictname asfdict id=dept
```
3. Issue your SELECT statement joining the two tables:


```
select *
from emp.employee a, dept.department b
where a.dept-id = b.dept-id
```

Nesting SELECT statements

You can issue multiple SELECT statements in a single retrieval request. By using more than one SELECT statement, you can apply a more specific search condition than is possible in a single WHERE clause. You can combine SELECT statements in a retrieval request in either of the following two ways:

- You can specify the subselect in the WHERE clause of the higher level SELECT statement.

Example:

To list the departments containing more than two employees:

```
select * from department
  where 2<(select count(*) from employee
  where employee.dept-id = department.dept-id) ! display
```

DEPARTMENT NAME	DEPT ID	DEPT HEAD ID
DEVELOPMENT	20	1127
PERSONNEL	30	4430

- You can include **existential quantifiers** (EXISTS or NOT EXISTS) in the higher level SELECT statement. CA OLQ evaluates the higher level SELECT statement in terms of whether (EXISTS) or not (NOT EXISTS) the nested condition is true.

Nested SELECT statements are enclosed in parentheses. There is no limit to the number of nested SELECT statements, but bear in mind that the statement becomes hard to understand after three or four nesting levels.

The column specification of the higher level SELECT statement must be an asterisk (*), indicating all columns.

Using existential quantifiers

This example lists which department employee Schley works in:

```
select * from department where exists
(select * from employee
  where employee.dept-id = department.dept-id
  and emp-last-name = 'schley') ! display
```

DEPARTMENT NAME	DEPT ID	DEPT HEAD ID
DEVELOPMENT	20	1127

Stringing together SELECT statements (UNION)

You can concatenate two or more tables containing like columns, using the UNION statement. The result table contains data found in one or both source tables. The UNION option eliminates duplicate rows from the report. To display duplicate rows, specify UNION ALL.

The two SELECT statements must have the same number of columns. Corresponding columns must have the same:

- Data length
- Data type. For example, floating point, binary, numeric
- Decimal representation

The DISTINCT operand cannot be specified when using the UNION option.

This example lists employee information for all employees in the Development departments of the Massachusetts and New York EMPLOYEE tables. Because ALL is specified, duplicate rows are displayed:

```
select * from mass.employee
union all
select * from ny.employee
```


For more information:

[Global Syntax](#) (see page 35)

SELECT—IDMS access mode

Retrieves values from one or more SQL tables and views for display in CAOLQ.

Syntax:

▶▶ idms-sql-select-statement 

Note: For the syntax, authorization, parameters, usage notes, and examples for the SELECT (IDMS access mode) command, see the *CA IDMS SQL Reference Guide*.

Coding considerations:

The parsing rules are different for SELECT depending on how the access switch is set. For instance, you *cannot* follow SELECT (IDMS access mode) with a separator or comment character.

Also the use of abbreviations, literals, and operators differs:

In IDMS mode	In OLQ mode
:display.SELECT * from emp where emp-lname = 'Smith'	:display.SEL * from emp where emp-lname EQ Smith

In the above example for OLQ mode:

- SELECT is abbreviated to *SEL*
- The operator *EQ* is used
- The character string *Smith* is *not* enclosed in quotes

You *cannot* do this with the SELECT command in IDMS mode.

Note: Consult *CA IDMS SQL Reference Guide*, for more information about rules for using SELECT (IDMS access mode).

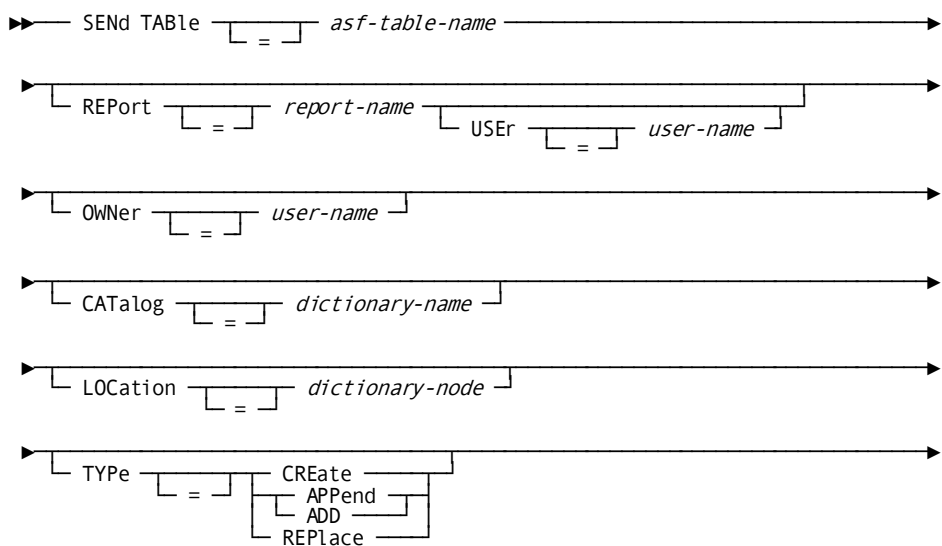
SEND TABLE—OLQ access mode

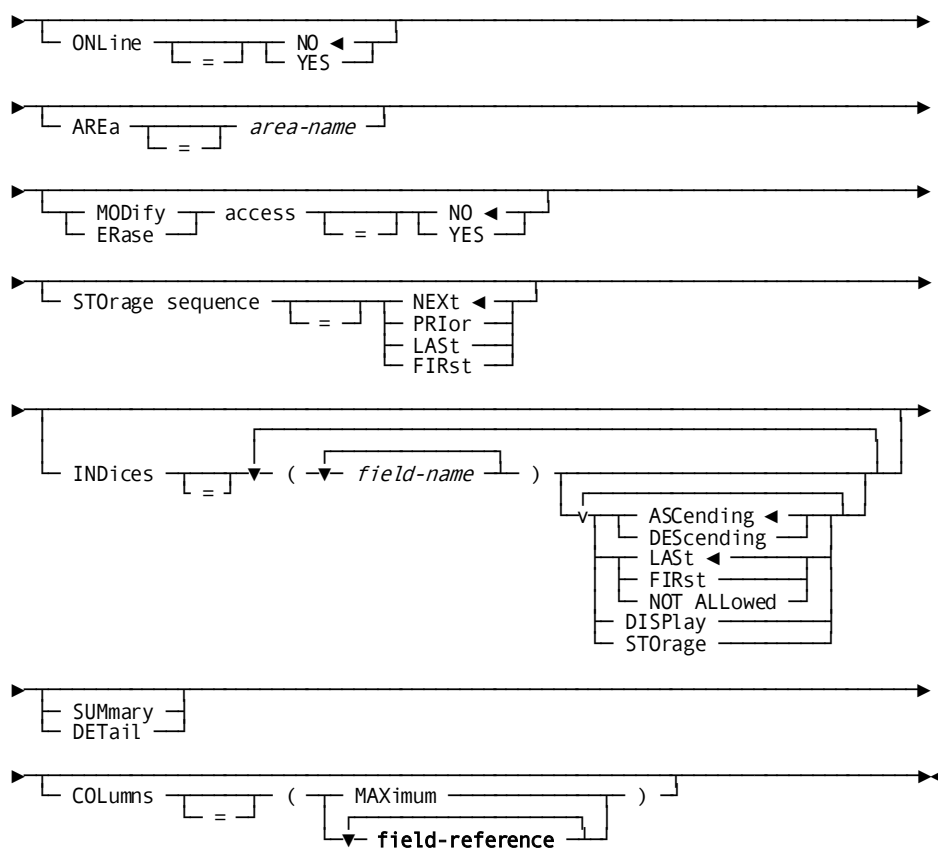
SEND TABLE stores information from the current or named report file as an ASF table.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

The columns that make up the new ASF table are derived from existing fields in the current report file.

Syntax:



**Parameters:****asf-table-name**

The name of the ASF table to be saved in CA IDMS or CA IDMS/DB.

REPORT= report-name

Identifies the saved report containing the columns that define the ASF table. *Report-name* names a report that was created with the SAVE REPORT command. If REPORT=*report-name* is not specified, the ASF table is defined by using columns from the current report.

USER= user-name

Identifies the user who saved the named report. If USER=*user-name* is not specified, the report is retrieved from the current user's directory.

OWNER= user-name

Specifies the user ID for the owner of the ASF table. If *user-name* is not specified, the ASF table is saved under the current user's ID.

CATALOG= dictionary-name

Specifies the name of the IDB or ASF dictionary where the catalog entry to the named ASF table is added.

LOCATION= dictionary-node

Specifies the name of the Distributed Database System (DDS) node controlling the named dictionary.

TYPE=

Specifies whether the named ASF table is being created, added to, or replaced.

Type can be:

- CREATE—The ASF table is new and is being assigned an initial definition.
- APPEND/ADD—New data is added to an existing table definition.
- REPLACE—The ASF table already exists and is replaced by new data.

Note: When either TYPE=APPEND/ADD or TYPE=REPLACE is specified, the columns in the current report must be the same as the columns in the existing table definition.

ONLINE= NO/YES

Specifies whether a map and dialog are built for the ASF table. The default is NO.

AREA= area-name

Names an alternative area to store the ASF table. Users must have DBA authority to specify this option.

MODIFY ACCESS= NO/YES

Specifies whether or not individual rows in the stored ASF table can be modified through ASF. The default is NO. A logical record MODIFY path is not built in the table subschema.

ERASE ACCESS= NO/YES

Specifies whether or not individual rows in the stored ASF table can be deleted through ASF. The default is NO.

STORAGE SEQUENCE=

Specifies how data is added to the database:

- NEXT—Each new DATA record occurrence is connected immediately after the record occurrence that is current of set.
- PRIOR—Each new DATA record occurrence is connected immediately before the record occurrence that is current of set.
- LAST—Each new DATA record occurrence is connected immediately preceding the owner record.

- FIRST—Each new DATA record occurrence is connected to the set in the position immediately following the owner record.

The STORAGE SEQUENCE cannot be specified if the STORAGE parameter is specified in the INDICES statement.

Notes:

- For more information about set order, see the *CA IDMS Performance Monitor System Administration Guide*.
- For more information about ASF, see the *CA IDMS ASF User Guide*.

INDICES=

Defines characteristics of the index set for the ASF table.

- *field-name*—A single or concatenated key field. You can specify more than one *field-reference* value for a table.
- ASCENDING/DESCENDING—The order in which record occurrences are connected to a set sorted by key value. The default is ASCENDING.
- LAST—A new record with a duplicate sort key value is stored immediately after the existing duplicate record.
- FIRST—A new record with a duplicate sort key value is stored immediately before the existing duplicate record.
- NOT ALLOWED—Duplicate sort keys are not allowed.
- DISPLAY—Specifies a display sequence of that defined as the display sequence for the table in ASF.
- STORAGE—Specifies a display sequence of that defined as the storage sequence for the table in ASF.

Note: You can specify **STORAGE** and **DISPLAY** only once each for any table. You *cannot* specify **STORAGE SEQUENCE=** and **STORAGE** for the same table.

SUMMARY

Specifies that summary report lines only be included in the ASF table.

Note: In addition to specifying **SUMMARY**, you can create a table which contains summary information only by specifying **DISPLAY SUMMARY** or by selecting **SUMMARY ONLY** from the Sort screen in menu mode. Any subsequent SEND TABLE will contain only the summary information.

DETAIL

Specifies that *all* detail report lines be included in the ASF table.

COLUMNS=

Specifies the report file columns included in the ASF table. Columns can be:

- **MAXIMUM**—All sequential columns are saved as columns in the ASF table.
- *field-reference*—The columns and the number of characters in each column saved in the ASF table.

Considerations:

When an ASF table is created, the names of the field columns in the report file are assigned to the columns in the table definition.

If CA OLQ headers have been assigned to any report fields, these headers are retained as column names; both dynamic headers and CA OLQ headers retrieved from the data dictionary can be assigned to columns in an ASF table.

For multiple line headers, CA OLQ uses the field name for the internal name.

Examples:*Send Table*

The SEND TABLE command can be used to instruct CA OLQ to replace all data previously associated with an ASF table with new data occurrences from the current report file. The following example replaces the EMP-HOSPITAL table with a single record:

```
send table=emp-hospital owner=bdm catalog=asfdict
type=replace
```

```
OLQ 102017 TABLE PROCESSING HAS BEEN SUCCESSFULLY COMPLETED
```

Send Table Indices

This example presents the use of INDICES to define the characteristics of the index set for the ASF table:

```
send table=emp-salary online=yes
indices=(emp-id-0415) not allowed
(emp-last-name-0415, emp-first-name-0415)
```

For more information:

[Global Syntax](#) (see page 35)

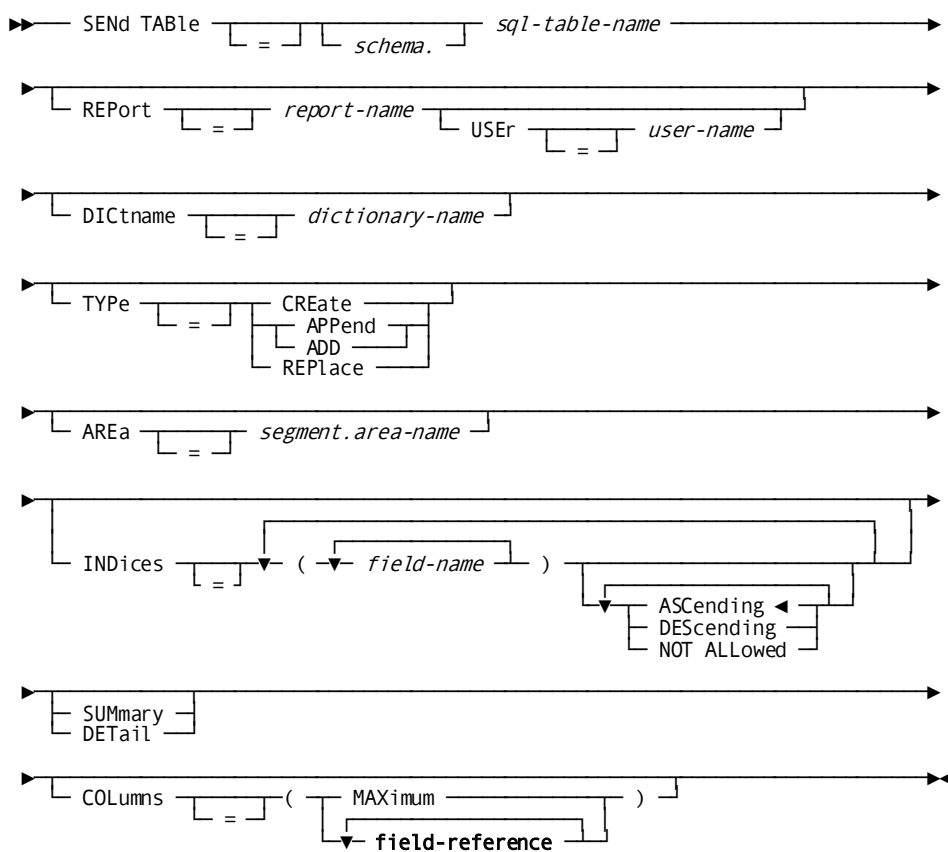
SEND TABLE—IDMS access mode

SEND TABLE stores information from the current or named report file as an SQL table.

Access mode: The syntax below is **invalid** when the access switch is set to **OLQ**.

Whatever data used to construct a report file can be stored as an SQL table.

Syntax:



Parameters:

sql-table-name

The name of the SQL table to be saved in CA IDMS/DB.

schema

The name of the schema associated with the SQL table to be saved in CA IDMS/DB.

REPORT= report-name

Identifies the saved report containing the columns that define the SQL table. *Report-name* names a report that was created with the SAVE REPORT command. If REPORT=*report-name* is not specified, the SQL table is defined by using columns from the current report.

USER= user-name

Identifies the user who saved the named report. If USER=*user-name* is not specified, the report is retrieved from the current user's directory.

DICTNAME= dictionary-name

Specifies the name of the SQL catalog where the named SQL table is added.

TYPE=

Specifies whether the named SQL table is being created, added to, or replaced. Type can be:

- CREATE—The SQL table is new and is being assigned an initial definition.
- APPEND/ADD—New data is added to an existing table definition.
- REPLACE—The SQL table already exists and is replaced by new data.

Note: When either TYPE=APPEND/ADD or TYPE=REPLACE is specified, the columns in the current report must be the same as the columns in the existing table definition.

AREA= segment.area-name

Names an alternative area to store the SQL table.

INDICES=

Defines characteristics of the index for the SQL table.

- *field-name*—A single or concatenated key field. You can specify more than one *field-reference* value for a table.
- ASCENDING/DESCENDING—The order in which record occurrences are connected to a set sorted by key value. The default is ASCENDING.
- NOT ALLOWED—Duplicate sort keys are not allowed. The key must be unique.

SUMMARY

Specifies that summary report lines only be included in the ASF table.

Note: In addition to specifying SUMMARY, you can create a table which contains summary information only by specifying **DISPLAY SUMMARY** or by selecting SUMMARY ONLY from the Sort screen in menu mode. Any subsequent SEND TABLE will contain only the summary information.

DETAIL

Specifies that *all* detail report lines be included in the ASF table.

COLUMNS=

Specifies the report file columns included in the SQL table. Columns can be:

- **MAXIMUM**—All sequential columns are saved as columns in the SQL table.
- *field-reference*—The columns and the number of characters in each column saved in the SQL table.

Considerations:

When an SQL table is created, the names of the field columns in the report file are assigned to the columns in the table definition.

If CA OLQ headers have been assigned to any report fields, these headers are retained as column names; both dynamic headers and CA OLQ headers retrieved from the data dictionary can be assigned to columns in an SQL table.

For multiple line headers, CA OLQ uses the field name for the internal name.

Examples:*Send Table*

The SEND TABLE command can be used to instruct CA OLQ to replace all data previously associated with an SQL table with new data occurrences from the current report file. The following example replaces the EMP-HOSPITAL table with a single record:

```
send table=employee.hospital dictname=empdict
      type=replace
```

```
OLQ 102017 TABLE PROCESSING HAS BEEN SUCCESSFULLY COMPLETED
```

Send Table Indices

This example presents the use of INDICES to define the characteristics of the index set for the SQL table:

```
send table=emp-salary
      indices=(emp-id-0415) not allowed
      (emp-last-name-0415, emp-first-name-0415)
```

For more information:

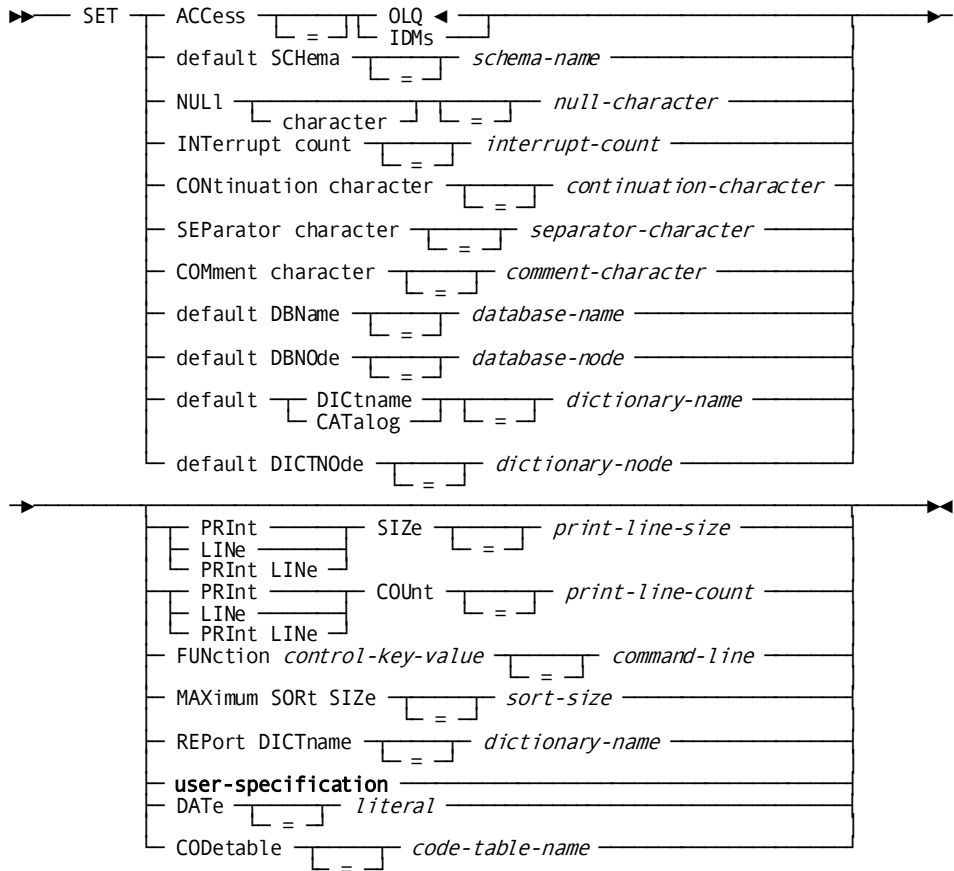
[Global Syntax](#) (see page 35)

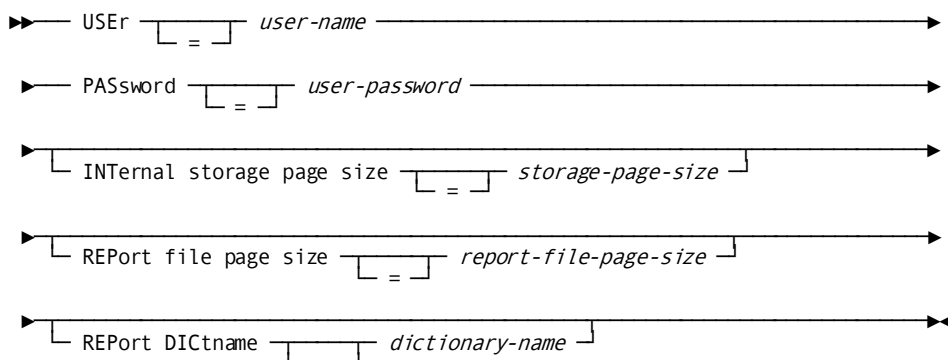
SET

SET permits the user to set system management parameters in a signon profile or during a session. While some parameters are initially defined at system generation, others are assigned values during the signon process.

Note: For more information about system generation, see the *CA IDMS System Generation Guide* and [System Generation Options](#) (see page 345) of this manual.

Syntax:



Expansion of user-specification**Parameters:****ACCESS=**

Specifies the type of table CA OLQ will access.

- *OLQ*— Allows you access to ASF tables.

The OLQ setting also allows you to use the SELECT statement with the following entities:

- ASF tables
- Logical records
- Database records
- Sequential files (batch only)

This is the default.

- *IDMS*— Allows you access to SQL tables when using the SELECT statement:

Access mode: Some CA OLQ commands are **invalid** when the access switch is set to **IDMS**.

Note: For more information about what commands become invalid, see the particular CA OLQ command in this chapter.

DEFAULT SCHEMA=

Sets a default schema so that any reference to table-name becomes schema.table-name.

Access mode: This parameter has **no meaning** when the access switch is set to **OLQ**.

Specifies the name of the schema. **Schema-name** be a name 1-18 characters long that follows the conventions for SQL identifiers.

Note: For more information about schemas, see the *CA IDMS SQL Reference Guide*.

NULL=

Specifies a display character used to portray data columns containing nulls. The default is the period (.).

Note: For more information about null processing, refer to the *CA IDMS SQL Reference Guide*.

INTERRUPT COUNT=

Specifies the number of records read before a retrieval is interrupted.

Note: See [System Generation Options](#) (see page 345) for details on setting the valid values during system generation.

CONTINUATION CHARACTER=

Specifies the character used to denote the continuation of a command. The continuation character is used in qfiles or batch input when the length of a CA OLQ command is greater than one line. The continuation character is a one-character alphanumeric value or special character. The system generation default is the hyphen (-).

SEPARATOR CHARACTER=

Specifies the character used to denote the separation of commands. The separator character is used to concatenate CA OLQ commands, permitting the entry of multiple commands on a single input line. The separator character is a one-character alphanumeric value or special character. The system generation default is the exclamation point (!).

Use with SELECT (IDMS mode)

Anytime you use separators with SELECT (IDMS mode) they *must* precede the SELECT statement.

For instance, the following is valid syntax:

```
delete table employee.job!select all from emp_id
```

However, CA IDMS/DB does *not* accept the syntax below because the separator (!) comes *after* the SELECT statement:

```
select all from emp_id!delete table employee.job
```

COMMENT CHARACTER=

Specifies the character used to denote the beginning of comments. The comment character signifies a remark; all text following the comment character is ignored during execution. The comment character is a one-character alphanumeric value or special character. The system generation default is the semicolon (;). All text following the comment character is ignored during execution.

Access mode: You cannot use comment characters with SELECT (IDMS mode) statements.

DEFAULT DBNAME=

Specifies the default database name.

DEFAULT DBNODE=

Specifies the default database node.

DEFAULT DICTNAME/CATALOG=

Specifies the default dictionary name.

DEFAULT DICTNODE=

Specifies the default dictionary node.

PRINT LINE SIZE=

Specifies the print line size.

PRINT LINE COUNT=

Specifies the print line count.

FUNCTION=

SET FUNCTION specifies a value for a control key.

- *control-key-value*— The control key that is being assigned a value. Valid control keys values are [PA1], [PA2], and 1 through 99 (corresponding to PF keys 1 through 99).
- *command-line*— The CA OLQ command that is assigned to the specified control key. If the command contains any special characters or embedded blanks, enclose *command-line* in quotation marks.

MAXIMUM SORT SIZE=

Specifies the amount of storage allocated for sorts.

- *sort-size*— The maximum size, in kilobytes, of main memory available for sorts. *Sort-size* is an integer in the range 1 to 32767. After this space is used, CA OLQ uses the scratch area for sorts.

REPORT DICTNAME=

Specifies the dictionary used for saving report information. Job control language for batch jobs is also stored here.

This command is valid only in CV batch mode.

- *dictionary-name*— The dictionary where the catalog containing saved report information resides. *Dictionary-name* is a 1 to 8-character alphabetic name.

Note: If you don't set the dictionary name, the report is saved in the primary dictionary. You must set the dictionary name to match the one defined during system generation for storing reports. This keeps the online and batch reports stored in the same place. If you do not match the dictionary names, you can't use online CA OLQ to access the reports saved through batch.

user-specification

- **USER=**—Identifies a user to CA OLQ in online and batch environments. CA OLQ uses the ID and password you assign with the SET USER statement in place of the DC/UCF signon user ID and password.

You can issue this command either in an online command mode CA OLQ session or in your batch job stream.

- **user-id**—The 1 to 32-byte alphanumeric user's identifier.
- **PASSWORD=***user-password*—Assigns a password to the user. *User-password* is a 1 to 8-character alphanumeric literal.
- **INTERNAL STORAGE PAGE SIZE=***storage-page-size*—For use in the CA OLQ batch environment only. Specifies the internal storage page size in bytes. *Storage-page-size* is an integer value in the range 1 to 32,767 and should be equivalent to the page size of the CA IDMS/DC region.

Note: For more information about how to specify page sizes for the CA IDMS/DC region, see the *CA IDMS System Generation Guide*.

- **REPORT FILE PAGE SIZE=***report-file-page-size*—For use in the CA OLQ batch environment only. Specifies the report file page size in bytes. *Report-page-size* is an integer value in the range 256 to 32,767 and should be equivalent to the page size of the CA IDMS/DC region.
- **REPORT DICTNAME=***dictionary-name*—For use in the CA OLQ batch environment only. Specifies the name of the dictionary where the catalog containing saved report information resides. *Dictionary-name* is a 1 to 8-byte alphabetic name.

Note: The SET USER command should always be the first CA OLQ batch command.

DATE=

Specifies the date format in CA OLQ. You can use this date option to change the format of the current date by including \$DATE in a PAGE HEADER/FOOTER command.

- *literal-string*—A literal string that can consist of any combination of the following:
 - **MONTH**—The word MONTH specifies the name of the month in capital letters.
 - **MON**—The word MON specifies just the first three letters of the month name, in capital letters.
 - **Mon**—The word Mon specifies just the first three letters of the month name, with the initial letter capitalized.
 - **Month**—The word Month specifies the name of the month spelled out with the initial letter capitalized.

- month—The word month specifies the name of the month spelled out in lowercase letters.
- mon—The word mon specifies the first three letters of the month name in lowercase letters.
- MM/ZM—The letters MM or ZM represent the month:
MM specifies the month be displayed with leading zeros; for example, February would appear as 02.
ZM specifies the month be displayed without leading zeros; for example, February would appear as 2.
- DD/ZD—The letters DD or ZD:
DD specifies the day be displayed with zeros; for example, Feb 9 would appear as Feb 09.
ZD specifies the day be displayed without zeros; for example, Feb 09 would appear as Feb 9.
- CC/YY/YYYY—The letters CC, YY, or YYYY to represent the year:
CC specifies the year be displayed as the century number. For example, 20 would appear for any date between 1900 and 1999.
YY specifies the year be displayed as the last two digits of the year. For example, the year 1996 would appear as 96.
YYYY specifies the year be displayed as the entire four digit year. For example, the year 1996 would appear as 1996.

You can specify these parameters in any order. Note that you can substitute the values themselves in place of the variables. For instance, instead of specifying DATE=Month DD, YY, you can specify DATE = January 27, 1996.

CODETABLE=

Specifies the code table to translate menu mode syntax and month literals used in the \$DATE function.

Examples:

INTERRUPT COUNT

In the following example, a SELECT statement retrieves all data occurrences for the EMPLOYEE record:

```
select * from employee
```

```
OLQ 098006 00 57 whole lines and 0 partial line in report.  
OLQ 098007 00 57 records read. 57 records selected.
```

The following SET INTERRUPT COUNT command reduces the number of records read before a retrieval interruption occurs:

```
set interrupt count = 25
```

When another SELECT statement is issued for the EMPLOYEE record, CA OLQ retrieves only 25 record occurrences:

```
select * from employee
```

```
SET INTERRUPT COUNT 25
OLQ 092010 00 The interrupt count has been modified.
SELECT * FROM EMPLOYEE
OLQ 098006 00 25 whole lines and 0 partial lines in report.
OLQ 098007 00 25 records read. 25 records selected.
OLQ 098008 00 38 of 98 primary record pages read.
OLQ 098009 00 Continue (yes/no)?
```

SEPARATOR CHARACTER

In the following example, the SET SEPARATOR CHARACTER command identifies the percent sign (%) as the separator character:

```
set separator character = '%'
```

```
OLQ 092014 00 The SEPARATOR CHARACTER has been modified.
```

After the SET SEPARATOR CHARACTER command has been issued, the designated separator character is used to concatenate two CA OLQ commands:

```
select office-code-0450,office-street-0450 from office %display

      OFFICE REPORT
      mm/dd/yy

OFFICE-CODE-0450      OFFICE-STREET-0450

002                  567 BOYLSTON ST
001                  20 W BLOOMFIELD ST
008                  910 E NORTHSOUTH AVE
005                  7690 NEAR SIGHT AVE
012                  734 MASS AVE
END OF REPORT
```

COMMENT CHARACTER :p In the following example, the SET COMMENT CHARACTER command identifies the pound sign (#) as the comment character:

```
set comment character = '#'
```

```
OLQ 092015 00 The COMMENT/TERMINATOR CHARACTER has been modified.
```


The established comment character is used to enter remark text after a SELECT statement is issued:

```
select * from employee
where emp-last-name like '%ing'#retrieves employees whose last names
end in 'ing'

OLQ 104009 04 DISPLAY RIGHT to see more report columns
EMPLOYEE REPORT
09/25/99

EMP-ID-0415 EMP-FIRST-NAME-0415 EMP-LAST-NAME-0415 EMP-STREET-0415

106 DORIS KING 716 MORRIS ST
END OF REPORT
```

DEFAULT DICTNAME

In the following example, the SET DEFAULT DICTNAME command identifies DOCANWK as the default dictionary name:

```
set default dictname = docanwk
```

```
OLQ 092018 00 THE DEFAULT DICTNAME VALUE HAS BEEN
MODIFIED.
```

When the following HELP command is issued, CA OLQ lists all subschemas that exist within the designated default dictionary:

```
help subschemas
```

```
AVAILABLE SUBSCHEMAS PAGE 1.1
LINE 1
(SCHEMA) (SUBSCHEMA)

DICTIONARY NAME DOCANWK
DICTIONARY NODE *DEFAULT*

EMPSCHM(2) EMPSS01
EMPSCHM(100) EMPSS01
EMPSS09
TEST(100) SUBTEST
```

SET FUNCTION

In the following example, a SHOW statement has been assigned to PF8:

```
set function 8 'show report'
```

When [PF8] is pressed in command mode, CA OLQ displays the description of the current report.

SIGNON

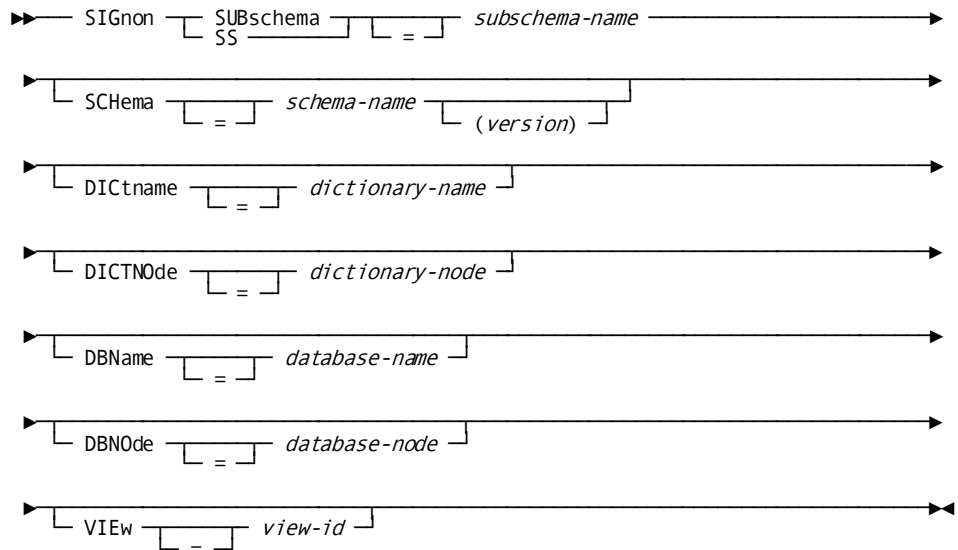
SIGNON indicates to CA OLQ that a named subschema is to be used to perform retrievals. Associating a *view-id* with a subschema allows access to multiple subschemas during a retrieval.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

When you enter SIGNON during a session in progress, CA OLQ releases all database currencies previously established and automatically signs you on to the new database view.

Note: Control key settings and session options maintain their values across subschema signons.

Syntax:



Parameters:**SUBSCHEMA= subschema-name**

Identifies the subschema for the current CA OLQ session. The named subschema must exist as a compiled description in the data dictionary and as a load module in either the dictionary load area or in the load (core-image) library.

SCHEMA= schema-name (version)

Specifies the schema associated with the subschema:

- *schema-name*—The schema associated with the named subschema. *Schema-name* defaults to the schema under which the subschema was last compiled.
- *(version)*—The version number, in the range 1 through 9999, of the named schema. *Version* defaults to the highest version number existing for that schema. If specified, *version* must be enclosed in parentheses.

DICTNAME= dictionary-name

Specifies the dictionary from which CA OLQ takes record and set definitions.

Note: To determine which dictionaries and databases are available, use the DCMT DISPLAY DBNAME TABLE command of DC/UCF.

See the *CA IDMS System Tasks and Operator Commands Guide*, for details on the DCMT DISPLAY DBNAME TABLE command.

DICTNODE= dictionary-node

Specifies the Distributed Database System (DDS) node controlling the named dictionary.

DBNAME= database-name

Specifies the database from which CA OLQ retrieves data. DBNAME can identify a user database or a database dictionary.

DBNODE= database-node

Specifies the DDS node that controls the named database.

VIEW= view-id

Specifies an ID by which the subschema can be referred in retrieval commands. **VIEW=***view-id* is required for retrieval from multiple subschemas.

Examples:

Default values for dictionary name, dictionary node, database name, and database node can be set with the DCUF SET DBNAME command, initiation of a CA OLQ session through the transfer control facility, or with the SET command.

Signon subschema

This example shows a user signon to a subschema and the OLQ response. A subschema (EMPSS01) and dictionary name (DOCUNET) are provided in the SIGNON command:

```
signon dic=docunet ss=empss01
```

```
OLQ 100021 00 Ready to retrieve data from subschema EMPSS01
OLQ 100022 00 Schema:      EMPSCHM          Version:      100
OLQ 100025 00 Dictionary name:  DOCUNET
```

Assigning a view ID to the subschema

This example shows a user signon to CA OLQ and the OLQ response. A subschema (EMPSS01) and subschema view ID (EMP1), and dictionary name (TSTDICT) are provided in the SIGNON command:

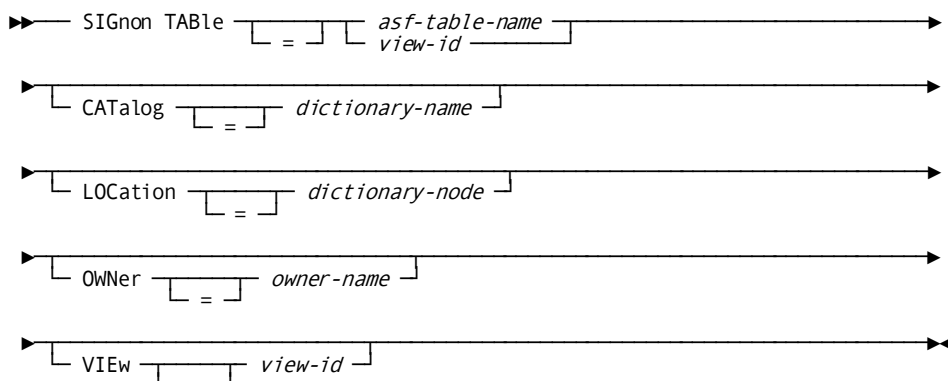
```
signon ss=empss01 view=emp1 dic=tstdict dbname=data1!
signon ss=empss01 view=emp2 dic=tstdict dbname=data2
```

```
SIGNON SS=EMPSS01 VIEW=EMP1 DIC=TSTDICT
OLQ 100021 00 Ready to retrieve data from subschema EMPSS01
OLQ 100022 00 Schema:      EMPSCHM          Version:      100
OLQ 100023 00 Database name:  DATA1
OLQ 100025 00 Dictionary name:  TSTDICT
OLQ 100027 00 View Id:      EMP1
SIGNON SS=EMPSS01 VIEW=EMP2 DIC=TSTDICT
OLQ 100021 00 Ready to retrieve data from subschema EMPSS01
OLQ 100022 00 Schema:      EMPSCHM          Version:      100
OLQ 100023 00 Database name:  DATA2
OLQ 100025 00 Dictionary name:  TSTDICT
OLQ 100027 00 View Id:      EMP2
```

SIGNON TABLE

SIGNON TABLE allows you to efficiently process multiple requests against a single ASF table.

Access mode: The syntax below is **invalid** when the access switch is set to **IDMS**.

Syntax:**Parameters:**

- *asf-table-name*—A 1 to 32-character alphanumeric table name.
- *view-name*—A 1 to 8-character alphanumeric table view ID.

CATALOG= dictionary-name

Specifies the name of the dictionary containing the catalog entry for the named ASF table.

LOCATION= dictionary-node

Specifies the name of the Distributed Database System (DDS) node controlling the named dictionary.

OWNER= owner-name

Specifies the user ID for the owner of the ASF table. If *owner-name* isn't specified the current user ID is used.

VIEW= view-id

Specifies a user-supplied label identifying the subschema. *View-id* is a 1 to 8-character label used to qualify entity names.

Example:

This example shows a user signon to an ASF table and the CA OLQ response. The ASF table name and owner name are supplied in the signon command:

```
signon table=employee owner=dmc
```

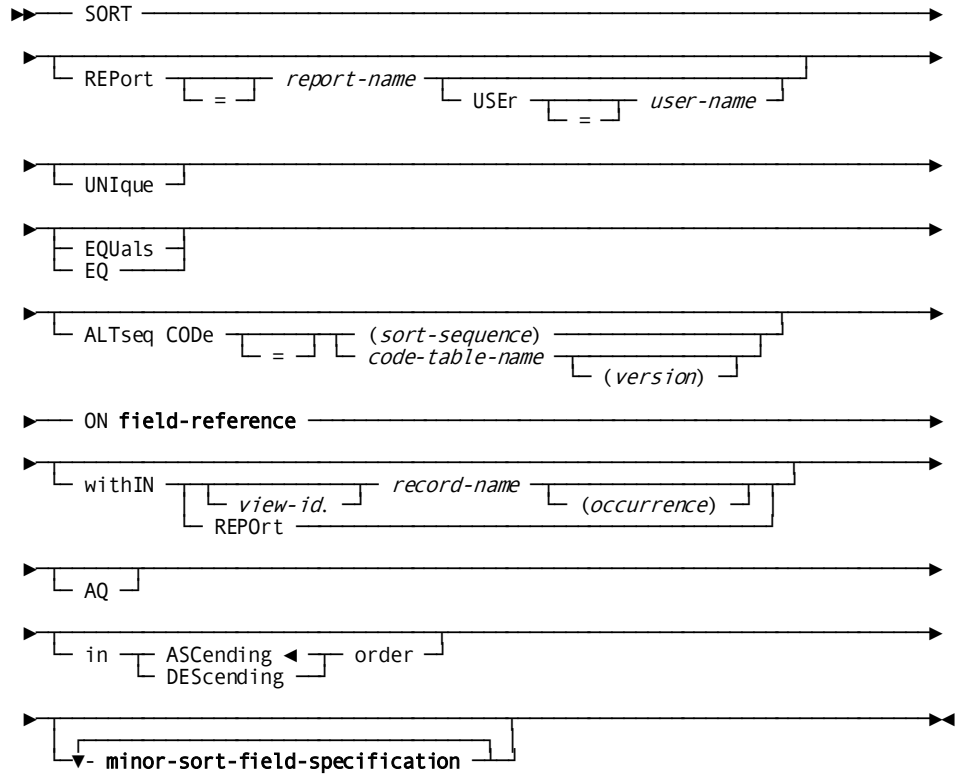
```

OLQ 100021 00 Ready to retrieve data from subschema RU000426
OLQ 100022 00 Schema:      IDMSR          Version:      1
OLQ 100023 00 Database name:  ASFDICT
OLQ 100025 00 Dictionary name: ASFDICT
  
```

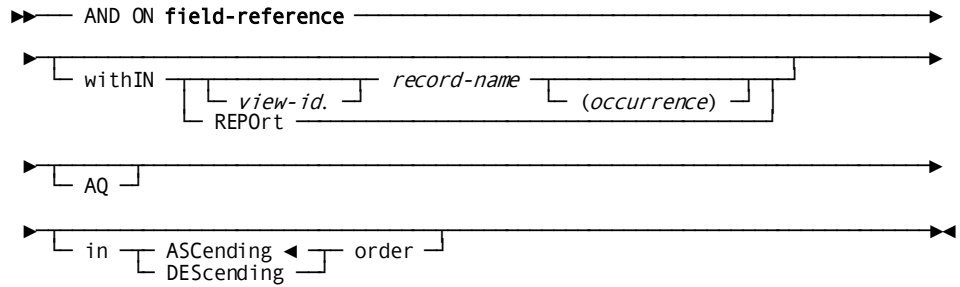
SORT

SORT sequences records within a report file in a user-specified order. A subsequent DISPLAY command displays the report lines in the specified sequence.

Syntax:



Expansion of minor-sort-field-specification



Parameters:**REPORT= report-name**

Identifies the saved report to be sorted. If *report-name* isn't specified, the default is the current report.

USER= user-name

Specifies the user ID of the user who owns the report. If *user-name* isn't specified, the report retrieved is from the current user's directory.

UNIQUE

Eliminates extra report lines containing duplicate sort keys. Specify UNIQUE after the SORT verb and before the sort field reference.

Note that specifying UNSORT after issuing UNIQUE does *not* restore the lost lines.

EQUALS

Maintains the original order of report records with duplicate sort keys. Specify EQUALS after the SORT verb and before the sort field reference.

ALTSEQ CODE

Specifies an alternative sort sequence. ALTSEQ follows the SORT statement and appears before the first sort field.

- *(sort-sequence)*—The alternative sort sequence. The list consists of encode/decode pairs:
 - The encode value represents the hexadecimal value of the data in the database.
 - The decode value represents the hexadecimal value with which to replace the data.

The *sort-sequence* list must be enclosed in parentheses.

- *code-table-name*—A code table stored in the data dictionary. The code table is defined in IDD using DDDL syntax. *Code-table-name* cannot be enclosed in parentheses. The table version number must be enclosed in parentheses. Define the code table with the following options:
 - TYPE=CODE
 - SEARCH=LINEAR
 - DATA=ALPHANUMERIC
 - DUPLICATES NOT ALLOWED
- *version*—The version number of the code table. If *version* isn't specified, it defaults to the highest existing version number of the named code table.

ON field-reference

Identifies the report field used to resequence a record. Both elementary and computed fields can be specified.

WITHIN record-name (occurrence)

. Specifies the scope of the sort. If this clause is omitted, scope defaults to WITHIN REPORT. The scope of the sort is:

- *Record-name (occurrence)*—Sequencing is restarted for each new occurrence of the named record. WITHIN *record-name* has no effect if the record name occurs lower on the path than the record that contains the field used for sorting.

When the path that created the report file contains more than one retrieval command for the same record type, use *occurrence* to identify the desired occurrence. If specified, *occurrence* must be enclosed in parentheses.

REPORT

Specifies that sequencing continues over the entire report without regard to a change in record occurrence.

AQ

Flags fields requiring the alternate collating (sequence) change. AQ can appear anywhere after the field name and before the AND for the next field. AQ cannot interrupt a SCOPE clause; for example, ON DEPT-ID IN AQ REPORT.

IN ASCENDING/DESCENDING ORDER

Specifies whether records are sequenced in ascending or descending order. The default is ASCENDING.

minor-sort-field-specification

Specifies a lower level (minor) sort field. Fields named by *field-reference* are sorted within the previously sorted field. The scope and order of the lower level field can be specified as follows:

- WITHIN *record-name (occurrence)*/REPORT specifies the scope of the sequencing.

If you specify *record-name*, sequencing restarts each time a new occurrence of the named record is encountered. The optional *occurrence* parameter identifies the occurrence when the path that created the report file contains more than one retrieval command for the same record. If specified, *occurrence* must be enclosed in parentheses.

- IN ASCENDING/DESCENDING ORDER—Fields are sequenced in ascending or descending order. The default is ASCENDING.

Considerations:

You can specify up to 22 fields in one SORT command.

The report file can be returned to its original sequence at any time by using the UNSORT command described later in this chapter.

Note that the UNIQUE parameter *permanently* removes lines from the report.

When running batch you can use the OLQ internal sort or the sort facility of your operating facility.

Examples:

The following examples illustrate the use of the SORT command to arrange records in a report file, based on the SELECT statement shown below:

```
select dept-name-0410,emp-id-0415
  from department, employee
 where dept-employee
```

Sort ascending

In the following example, the report is sorted in ascending alphabetical order by department name:

```
sort on dept-name-0410 ! display

          DEPARTMENT/EMPLOYEE REPORT
          mm/dd/yy

          DEPT-NAME-0410              EMP-ID-0415

ACCOUNTING AND PAYROLL                67
ACCOUNTING AND PAYROLL                11
ACCOUNTING AND PAYROLL                101
ACCOUNTING AND PAYROLL                106
ACCOUNTING AND PAYROLL                69
ACCOUNTING AND PAYROLL                100
BLUE SKIES                            371
BLUE SKIES                            321
BLUE SKIES                            366
BRAINSTORMING                         467
BRAINSTORMING                         341
BRAINSTORMING                         458
BRAINSTORMING                         334
BRAINSTORMING                         457

          - 1 -
```

Sort on ... and on

The following SORT command sorts the records in the report by employee last name within department name; both sort fields are sorted in ascending order:

```
sort on dept-name-0410 and on emp-id-0415 ! display

DEPARTMENT/EMPLOYEE REPORT
09/22/99

DEPT-NAME-0410          EMP-ID-0415

ACCOUNTING AND PAYROLL          11
ACCOUNTING AND PAYROLL          67
ACCOUNTING AND PAYROLL          69
ACCOUNTING AND PAYROLL         100
ACCOUNTING AND PAYROLL         101
ACCOUNTING AND PAYROLL         106
BLUE SKIES                      321
BLUE SKIES                      366
BLUE SKIES                      371
BRAINSTORMING                   301
BRAINSTORMING                   334
BRAINSTORMING                   341
BRAINSTORMING                   457
BRAINSTORMING                   458

- 1 -
```

Sort descending

In the final example, records are sorted in descending order by EMP-ID-0415 within the scope of the entire report:

```
sort on emp-id-0415 in descending order ! display cols=2,1

DEPARTMENT/EMPLOYEE REPORT
09/22/99

EMP-ID-0415          DEPT-NAME-0410

9999          PUBLIC RELATIONS
8683          PERSONNEL
479           THERMOREGULATION
476           PUBLIC RELATIONS
472           EXECUTIVE ADMINISTRATION
471           EXECUTIVE ADMINISTRATION
469           THERMOREGULATION
467           BRAINSTORMING
466           BRAINSTORMING
458           BRAINSTORMING
457           BRAINSTORMING
371           BLUE SKIES
366           BLUE SKIES
355           THERMOREGULATION

- 1 -
```

For more information:

[Batch Processing](#) (see page 303)

[Global Syntax](#) (see page 35)

SUSPEND

SUSPEND allows you to suspend the current session and return control to the transfer control facility, DC/UCF.

When a CA OLQ session is initiated under the transfer control facility, the SUSPEND command can be used to return control to either a previously suspended task or to the Selection screen, which lists all tasks available within the facility. Otherwise, control is returned to DC/UCF.

When a session is suspended with the SUSPEND command, CA OLQ retains the current report file.

Syntax:

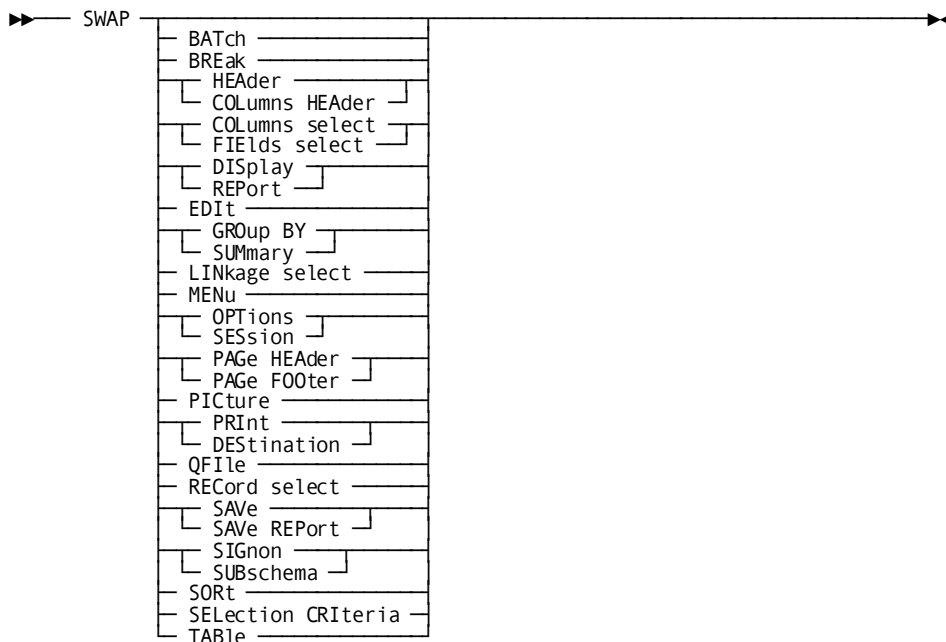
▶▶ — SUSpend —————▶▶

SWAP

SWAP allows you to switch between CA OLQ command mode and the menu facility.

When in command mode, you can specify a menu facility screen name and switch to a particular screen.

Syntax:



Parameters:

BATCH

The Batch Processing screen

BREAK

The Report Format - Sort screen

HEADER/COLUMN HEADER

The Report Format - Header screen

COLUMNS SELECT/FIELDS SELECT

The Column Select screen

TABLE

The Table Processing screen

DISPLAY/REPORT

The Display Report screen

EDIT

The Report Format - Edit screen

GROUP BY/SUMMARY

The Report Format - Group By screen

LINKAGE SELECT

The Linkage Select screen

MENU

The Menu screen

OPTIONS/SESSION

The Session Options screen

PAGE HEADER/PAGE FOOTER

The Page Header/Footer screen

PICTURE

The Report Format - Picture screen

PRINT/DESTINATION

The Print Processing screen

QFILE

The Qfile Processing screen

RECORD SELECT

The Record Select screen

SAVE/SAVE REPORT

The Save Report screen

SIGNON/SUBSCHEMA

The Signon Database View screen

SORT

The Report Format - Sort screen

SELECTION CRITERIA

The Selection Criteria screen

Example:

The command SWAP brings you to the Menu screen:

```

                                CA, Inc.
                                CA OLQ Release 16.0
                                *** Menu ***
->                               Page 1 of 3
122000 Select an option and press the ENTER key

  Pfkey  Select  Description                               Command/  Show
         Option Description                               Screen Name  Help
         -----
         X    ---> Data Source for Report <---
         -    Choose tables                               TABLE    -
         -    Choose subschema                           SUBSchema  -

         ---> Retrieval Activity <---
         -    Choose records from selected subschema     RECOld    -
         -    Choose columns for report                   COlumn    -
         -    Retrieve data to build report               RETRIeve  -
         -    Alter database access strategy              LINKage   -

         ---> Processing Mode <---
         -    Execute or create a predefined routine     QFILE    -
         -    View existing or save current report       SAVE     -
         -    Submit batch report request                BATCH    -

1=HELP      2=GLOBAL HELP      3=QUIT      4=MESSAGE      8=FWD
    
```

The command SWAP OPTIONS brings you to the Options Screen:

```

                                CA OLQ Release 16.0
                                *** Session Options ***
->                               Page 1 of 2
107017 CA OLQ Release 16.0
107019 Copyright(C) 2003 CA, Inc.
Current interrupt count: 100      Current underline character: -
Access IDMS SQL tables: Y (Y/N)  Current SQL NULL data value: .

User options:                    Page Columns Spread: (L-Left,E-Even,M-Max,nn)
  Help      Change      Current option  Alternate option
           Option
           -----
           -> Report Processing Options <-
           -    NOFiller      FILLer
           -    FULL         SPArse
           -    HEAdEr       NOHeAdEr

           -> Column Processing Options <-
           -    OLQheader    NOOLqheader
           -    PICTure      NOPICTure
           -    CODetable    NOCODetable

1=HELP      3=QUIT      4=MESSAGE      6=MENU      8=FWD
    
```


Example:

In the example presented below, the report file that was resequenced in several ways in the SORT command examples is now returned to its original sequence:

```
unsort ! display

                DEPARTMENT/EMPLOYEE REPORT
                09/22/99

EMP-ID-0415          DEPT-NAME-0410

    30    EXECUTIVE ADMINISTRATION
    471    EXECUTIVE ADMINISTRATION
     1    EXECUTIVE ADMINISTRATION
    472    EXECUTIVE ADMINISTRATION
    69    ACCOUNTING AND PAYROLL
    100    ACCOUNTING AND PAYROLL
    11    ACCOUNTING AND PAYROLL
    67    ACCOUNTING AND PAYROLL
    106    ACCOUNTING AND PAYROLL
    101    ACCOUNTING AND PAYROLL
    81    PERSONNEL
   8683    PERSONNEL
    51    PERSONNEL
    91    PERSONNEL

- 1 - Note that the rows removed by UNIQUE processing are not restored.
```


Chapter 7: Built-In Functions and Syntax

This section contains the following topics:

[Built-In Functions](#) (see page 211)
[Invoking Built-In Functions](#) (see page 211)
[Parameters Of Built-In Functions](#) (see page 220)
[Absolute Value](#) (see page 220)
[Arc Cosine](#) (see page 221)
[Arc Sine](#) (see page 222)
[Arc Tangent](#) (see page 223)
[Average](#) (see page 223)
[Capitalization](#) (see page 224)
[Concatenate](#) (see page 225)
[Cosine](#) (see page 226)
[Count](#) (see page 227)
[Date Change](#) (see page 227)
[Date Difference](#) (see page 230)
[Date Offset](#) (see page 230)
[Extract](#) (see page 231)
[Fix](#) (see page 232)
[Index](#) (see page 233)
[Initial Uppercase](#) (see page 234)
[Insert](#) (see page 235)
[Invert Sign](#) (see page 236)
[Left Justify](#) (see page 237)
[Length](#) (see page 238)
[Logarithm](#) (see page 239)
[Lowercase](#) (see page 239)
[Maximum](#) (see page 240)
[Minimum](#) (see page 241)
[Modulo](#) (see page 242)
[Next Integer Equal or Higher](#) (see page 242)
[Next Integer Equal or Lower](#) (see page 243)
[Product](#) (see page 244)
[Random Number](#) (see page 245)
[Right Justify](#) (see page 246)
[Sign Value](#) (see page 246)
[Sine](#) (see page 247)
[Square Root](#) (see page 248)
[Standard deviation](#) (see page 249)
[Standard deviation population](#) (see page 250)
[Substring](#) (see page 250)
[Sum](#) (see page 251)
[Tangent](#) (see page 252)
[Today](#) (see page 253)
[Tomorrow](#) (see page 255)
[Translate](#) (see page 256)
[Uppercase](#) (see page 257)
[Variance](#) (see page 258)
[Variance population](#) (see page 259)

[Verify](#) (see page 260)

[Weekday](#) (see page 261)

[Yesterday](#) (see page 263)

Built-In Functions

Built-in functions are predefined functions in CA OLQ that allow you to:

- Evaluate expressions according to predefined operations and return results that can be used in command mode processing.
- Perform predefined string, arithmetic, trigonometric, and date/time functions.
- Perform aggregate calculations that are based on the GROUP BY processing of the COMPUTE and SELECT commands. These aggregate calculations include:
 - Sum
 - Average
 - Maximum
 - Minimum
 - Count
 - Product
 - Standard deviation
 - Standard deviation population
 - Variance
 - Variance population

Invoking Built-In Functions

Built-in functions are invoked by specifying an *invocation* name.

There are five types of built-in functions: aggregate, arithmetic, date, string, and trigonometric. For a list of built-in functions and what they do, see the tables below.

Where you use them:

You can specify arithmetic, date, string, and trigonometric built-in functions in CA OLQ anywhere you would normally specify arithmetic or comparison expressions. You can specify aggregate built-in functions in a:

- Column list of a SELECT statement that has a GROUP BY clause
- COMPUTE expression that has a GROUP BY clause
- HAVING clause

Table 5: CA OLQ Aggregate Built-In Functions

Function	Invocation	Example
Return the average (median) value	AVERAGE AVE	compute ave-sal = ave(salary) group by dept
Return the number of elements	COUNT COU NUMBER NUM	select count(*) from employee group by dept
Return the highest value	MAXIMUM MAX HIVAL HIV	select max(salary) as 'Top Salary' from employee group by dept
Return the lowest value	MINIMUM MIN LOVAL LOV	select min(salary) as 'Low Sal' from employee group by dept
Return the product for all values of a break	PRODUCT PROD	select product (interest-rate) from mutual-funds group by all
Return the sum of all values	TOTAL SUM	select sum(salary) as 'Total Salaries' from employee group by dept
Return the sample standard deviation of all values	STD	select std(salary) as 'Standard Deviation Based on Sample' from employee group by job-id

Function	Invocation	Example
Return the population standard deviation of all values	STDP	select stdp(salary) as 'Standard Deviation Based on Population' from employee group by job-id
Return the sample variance of all values	VAR	select var(salary) as 'Variance Based on Sample' from employee group by job-id
Return the population variance of all values	VARP	select varp(salary) as 'Variance Based on Population' from employee group by job-id

Table 6: CA OLQ Arithmetic Built-In Functions

Function	Invocation	Example
Return the absolute value of a number	ABSOLUTE-VALUE ABS-VAL @ABS ABS	select abs(oper1) as 'Difference' from table1
Return the value of a number multiplied by -1	INVERT-SIGN INVERT INV	select inv(oper1) as 'Inverted Value' from table1
Return the natural logarithm of a number	LOG-BASE-E LOGNAT NATLOG LOGE @LN	select loge(oper1) as 'Log Base E' from table1
Return the common logarithm of a number	LOG-BASE-10 LOGCOM COMLOG LOG10 @LOG	select log10(oper1) as 'Log Base 10' from table1
Return the modulus (remainder) of a division operation	MODULO MOD @MOD	select mod(oper1 - oper2) as 'Remainder' from table1

Function	Invocation	Example
Return the smallest integer that is equal to or greater than the specified number	NEXT-INT-EQHI NEXTINTEH NEXIH @CEIL	select nexih(balance-due) as 'Balance Due' from invoice
Return the largest integer that is lower than the specified number	NEXT-INT-EQLO NEXTINTEL NEXIL @TRUNC @INT	select nexil(balance-due) as 'Balance Due' from invoice
Return a pseudo-random number based on a seed number	RANDOM-NUMBER RANDOM @RAND RAN	compute 'number' = random (13549)
Return a +1, 0, or -1 depending on whether a number is positive, zero, or negative	SIGN-VALUE SIGN-VAL SIGV	select sigv(oper1) as 'Sign' from table1
Return the square root of a number	SQUARE-ROOT @SQRT SQRT	select sqrt(oper1) as 'Square Root' from table1

Table 7: CA OLQ Date Built-In Functions

Function	Invocation	Example
Return the conversion of a specified date from one format (Gregorian, calendar, European, or Julian) to another format	DATECHG DATECHGX GCDATE GCDATEX GJDATE GJDATEX CEDATE CEDATEX EGDATE EGDATEX EJDATE EJDATEX JCDATE JCDATEX GEDATE GEDATEX CGDATE CGDATEX CJDATE CJDATEX ECDATE ECDATEX JGDATE JGDATEX JEDATE JEDATEX	compute calendar = datechg(start-date,'G','C')
Return the number of days between two specified dates	DATEDIF	select datefig(start-date, end-date) as 'Senior' from table1
Return the date resulting from adding a specified number of days to a specified date	DATEOFF	compute newdate = dateoff(start-date,4)
Return today's date in the format requested	TODAY TODAYX	select today('C') as 'Day-off'
Return tomorrow's date in the format requested	TOMORROW TOMORROWX	select tomorrow('C') as 'Holiday'
Return the weekday (Monday, Tuesday, etc.) of a specified Gregorian, calendar, European, or Julian date	WEEKDAY WEEKDAYX GWEEKDAY GWEEKDAYX CWEEKDAY CWEEKDAYX EWEEKDAY EWEEKDAYX JWEEKDAY JWEEKDAYX	compute weekday = weekday(birthday,'C')

Function	Invocation	Example
Return yesterday's date in the format requested	YESTERDAY YESTERDAYX	select yesterday('C') as 'Day-off'

Table 8: CA OLQ String Built-In Functions

Function	Invocation	Example
Return the concatenation of a specified list of strings	CONCATENATE CONCAT CON	select concatenate(emp-first-name,emp-last-name) from employee
Return the substring that results from removing leading and trailing spaces from a string	EXTRACT EXT	select extract(emp-last-name) from employee
Return a fixed-length string of 20, 40, 60, or 80 characters	FIX20 FIX40 FIX60 FIX80	select fix40 concat((extract(emp-fname), ' ',extract(emp-lname)))
Return the string resulting when the first letter in the specified source string is capitalized and all other characters in the string are converted to lowercase	INITCAP	compute new-emp-lname = initcap(emp-lname)
Return the string resulting from inserting one string into another	INSERT INS	select insert(emp-name,'**',1) from employee where emp-city eq boston
Return the string that results from left justifying a string	LEFT-JUSTIFY LEFT-JUST LEFJUS LEFT	select lefjus(emp-last-name) from employee

Function	Invocation	Example
Return the starting position of a specified substring	STRING-INDEX INDEX INDX	select * from invoice where index(prod-code,'ABC') ne 0
Return the length of a string	STRING-LENGTH SLENGTH SLEN	select length(extract(emp-first-name)) from employee
Return the substring of a string, starting from a specified position, and continuing for a specified length	SUBSTRING SUBSTR SUBS	select substr(emp-id,3,2) from employee
Return the string that results from right justifying a string	RIGHT-JUSTIFY RIGHT-JUST RIGHTJUS RIGHT	select rightjus(emp-name) from employee
Return the string that results from converting all characters to lowercase	TOLOWER	compute new-emp-lname = tolower(emp-lname)
Return the string that results from converting all characters to uppercase.	TOUPPER	compute new-emp-lname = toupper(emp-lname)
Return the string that results from translating characters in a string that also occur in a selection string to corresponding characters in a substitution string	TRANSLATE TRANS	select trans (course-id,'123','abc') from course-list

Function	Invocation	Example
Return the position of the first character in a string that doesn't occur in a second specified string	VERIFY VER	select emp-name from employee where verify(emp-id, '1234567890') ne 0
Return the string resulting when the first letter of each word in the specified source string is capitalized and all other characters in the string are converted to lowercase	WORDCAP	compute new-emp-lname = wordcap(emp-lname)

Table 9: CA OLQ Trigonometric Built-In Functions

Function	Invocation	Example
Return the arc cosine of a number that represents an angle in degrees	ARCCOSINE-DEGREES ARCCOSDEG ACOSD	compute 'Arc Cosine' = acosd(angle-in-degrees)
Return the arc cosine of a number that represents an angle in radians	ARCCOSINE-RADIANS ARCCOSRAD ACOSR	compute 'Arc Cosine' = acosr(angle-in-radians)
Return the arc sine of a number that represents an angle in degrees	ARCSINE-DEGREES ARCSINDEG ASIND	compute 'Arc Sine' = asind(angle-in-degrees)
Return the arc sine of a number that represents an angle in radians	ARCSINE-RADIANS ARCSINRAD ASINR	compute 'Arc Sine' = asinr(angle-in-radians)

Function	Invocation	Example
Return the arc tangent of a number that represents an angle in degrees	ARCTAN-DEGREES ARCTANDEG ATAND	compute 'Arc Tangent' = atand(angle-in-degrees)
Return the arc tangent of a number that represents an angle in radians	ARCTAN-RADIANS ARCTANRAD ATANR	:display. compute 'Arc Tangent' = atand(angle-in-radians)
Return the cosine of a number that represents an angle in degrees	COSINE-DEGREES COSDEG COSD	compute 'Cosine' = cosd(angle-in-degrees)
Return the cosine of a number that represents an angle in radians	COSINE-RADIANS COSRAD COSR	compute 'Cosine' = cosr(angle-in-radians)
Return the sine of a number that represents an angle in degrees	SINE-DEGREES SINEDEG SIND	compute 'Sine' = sind(angle-in-degrees)
Return the sine of a number that represents an angle in radians	SINE-RADIANS SINERAD SINR	compute 'Sine' = sinr(angle-in-radians)
Return the tangent of a number that represents an angle in degrees	TANGENT-DEGREES TANDEG TAND	compute 'Tangent' = tand(angle-in-degrees)
Return the tangent of a number that represents an angle in radians	TANGENT-RADIANS TANRAD TANR	compute 'Tangent' = tanr(angle-in-radians)

Parameters Of Built-In Functions

When coding parameters of built-in functions, use the following guidelines:

- Parameters of a built-in function must be enclosed in parentheses and should be separated by commas.
- Each parameter must appear in a specific position relative to the other parameters.
- Parameters in built-in functions are either string values or numeric values:
 - A **string** value is coded as an EBCDIC variable data field, a nonnumeric literal, or a built-in function that returns a string value.
 - A **numeric** value is coded as an arithmetic expression, a numeric variable data field, a numeric literal, or a built-in function that returns a numeric value.
- Some function parameters have restrictions on the values they can contain.

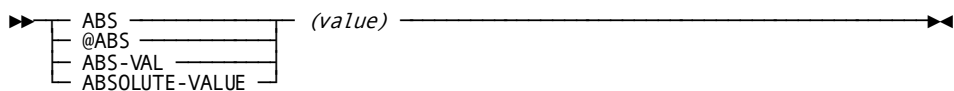
Example:

The values specified in a square root function cannot be negative. These restrictions are specified in the command's syntax rules.

Absolute Value

The absolute value function returns the absolute value of a numeric value, which is the numeric value of the number regardless of sign.

Syntax:



Invocation names:

ABS
@ABS
ABS-VAL
ABSOLUTE-VALUE

Parameters:

(value)

Specifies the numeric value whose absolute value is calculated.

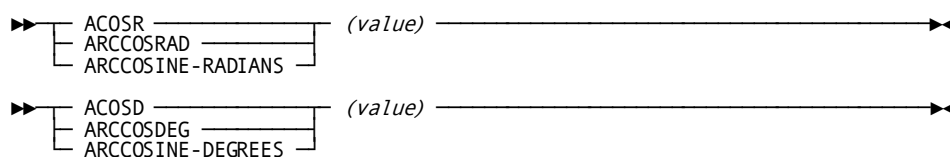
Value may be halfword-binary, fullword-binary, signed or unsigned, or short or longword floating point. Any length appropriate to the data type is acceptable. The result field will be the same data type and length as *value*.

Arc Cosine

The arc cosine functions return the arc cosine (inverse sign) of a numeric value that represents an angle in either degrees or radians.

The single floating point operand returned expresses the angle accurate to decimal places in the range zero to 180 for degrees and zero to π for radians.

Syntax:



Invocation names:

ACOSD	ACOSR
ARCCOSDEG	ARCCOSRAD
ARCCOSINE-DEGREES	ARCCOSINE-RADIANS

Parameters:

(value)

Specifies the numeric value representing the angle, in degrees or radians, whose arc cosine is calculated. *Value* must be a value ranging from -1 to +1.

Example:

This example uses the arc cosine (degrees) function to calculate the cosine of -0.5:

```
compute 'Arc Cosine' = acosd(-0.5)
```

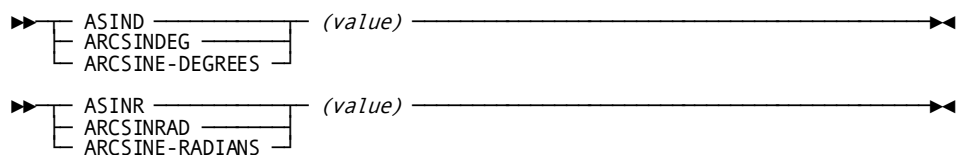
```
Arc Cosine = 120
```

Arc Sine

The arc sine functions return the arc sine of a numeric value that represents an angle in either degrees or radians.

The single operand returned expresses the angle accurate to 10 decimal places in the range -90 to +90 for degrees and $-\pi/2$ to $+\pi/2$ for radians.

Syntax:



Invocation names:

ASIND	ASINR
ARCSINDEG	ARCSINRAD
ARCSINE-DEGREES	ARCSINE-RADIANS

Parameters:

(value)

Specifies the numeric value representing the angle, in degrees or radians, whose arc sine is calculated. *Value* must be a value ranging from -1 to +1.

Example:

This example calculates the arc sine (in degrees) of 0.8660:

```
compute 'Arc Sine' = asind(0.8660)
```

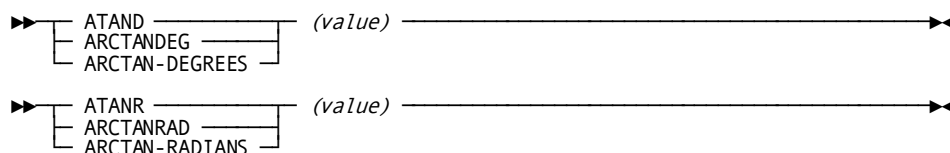
```
Arc Sine = 5.99971
```

Arc Tangent

The arc tangent functions return the arc tangent of a numeric value that represents an angle in either degrees or radians.

The single operand that is returned expresses an angle accurate to 10 decimal places in the range +90 to -90 for degrees and $-\pi/2$ to $+\pi/2$ for radians.

Syntax:



Invocation names:

ATAND	ATANR
ARCTANDEG	ARCTANRAD
ARCTAN-DEGREES	ARCTAN-RADIANS

Parameters:

(value)

Specifies the numeric value representing the angle, in degrees or radians, whose arc tangent is calculated.

Example:

This example calculates the arc tangent (in degrees) of 1.7321:

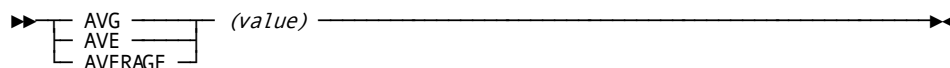
```
compute 'Arc Tangent' = atand(1.7321)
```

```
ARC TANGENT = 6.0007
```

Average

The average function returns the average (mean) value of a specified group of numeric values, based upon changes in the resultant value of the GROUP BY clause.

Syntax:



Invocation names:

AVG
AVE
AVERAGE

Parameters:

(value)

Specifies the numeric value or values whose average is being calculated.

Example:

This example determines the average salary for each department. The total salary for the employees in DEPT1 = 100000. The number of employees in DEPT1 = 6.

compute ave-sal = ave(salary) group by department

AVE-SAL = 16666.66

Capitalization

Returns the string that results when the first letter of each word in the specified source string is capitalized and all other characters in the string are converted to lowercase.

Syntax:

►► WORDCAP (*string*) ◀◀

Invocation names:

WORDCAP

Parameters:

string

Specifies the string to be converted.

String can be:

- A string literal, enclosed in single quotation marks
- The name of a user-defined variable data field containing the string

The first letter in each word is capitalized and all other characters are converted to lowercase.

Example:

In the example below, the word cap function is used on employees' last names:

Initial value:

```
EMP-LNAME: 'O'HEARN      '
```

Statement:

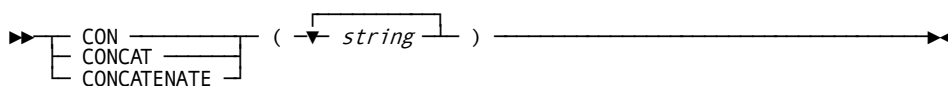
```
compute new-emp-lname = wordcap(emp-lname)
```

Returned string:

```
'O'Hearn      '
```

Concatenate

The concatenate function returns the concatenation of a specified list of string values.

Syntax:**Invocation names:**

```
CON
CONCAT
CONCATENATE
```

Parameters:**(string)**

Specifies one or more string values that are concatenated to form a single string value.

Examples:*Concatenate fields*

You can concatenate EMP-FIRST-NAME-0415 (PIC X(15)) and EMP-LAST-NAME-0415 (PIC X(15)) so the first name precedes the last name. EMP-FIRST-NAME-0415 is 'ELMER '. EMP-LAST-NAME-0415 is 'OTT '.

```
select concatenate(emp-first-name-0415, emp-last-name-0415)
       from employee
```

```
'ELMER      OTT      '
```

Concatenate and extract

You can use the concatenate function in conjunction with the extract function to concatenate EMP-FIRST-NAME-0415 (PICX(15)), up to but not including the first blank, with a blank and then with EMP-LAST-NAME-0415 (PICX(15)). The value of EMP-FIRST-NAME-0415 is 'ELMER ' and the value of EMP-LAST-NAME-0415 is 'OTT '.

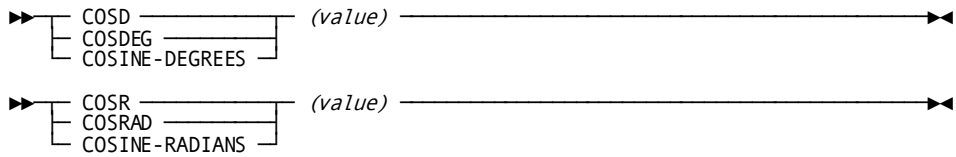
```
select con(extract(emp-first-name-0415),' ',
emp-last-name-0415) from employee

'ELMER OTT      '
```

Cosine

The cosine functions return the cosine of a numeric value that represents an angle in either degrees or radians.

Syntax:



Invocation names:

- COSD
- COSR
- COSDEG
- COSRAD
- COSINE-DEGREES
- COSINE-RADIANS

Parameters:

(value)

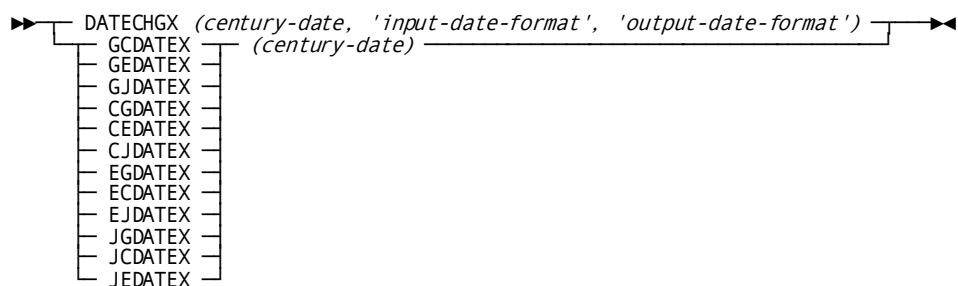
Specifies the numeric value representing the angle, in degrees or radians, whose cosine is calculated.

Example:

This example calculates the cosine (in degrees) of 60:

```
compute 'Cosine' = cosd(60)
```

```
COSINE = 5.000
```

Invocation names:

DATECHG				DATECHGX			
GCDATE	CGDATE	EGDATE	JGDATE	GCDATEX	CGDATEX	EGDATEX	JGDATEX
GEDATE	CEDATE	ECDATE	JCDATE	GEDATEX	CEDATEX	ECDATEX	JCDATEX
GJDATE	CJDATE	EJDATE	JEDATE	GJDATEX	CJDATEX	EJDATEX	JEDATEX

Parameters:

date

Specifies the six-byte object date for the date change function.

Date can be:

- A user-supplied numeric literal (enclosed within quotes)
- The name of a user-defined variable data field

century-date

Specifies the eight-byte object date (containing the century) for the date change function.

Century date can be:

- A user-supplied numeric literal (enclosed within quotes)
- The name of a user-defined variable data field

'input-date-format'

Specifies the format of the date specified by *date* or *century-date*.

Input date can be:

- 'G'— Gregorian
- 'C'— Calendar
- 'E'— European
- 'J'— Julian
- The name of a user-defined variable data field containing the date format

'output-date-format'

Specifies the format to which *date* or *century-date* is to be converted.

Output date can be:

- 'G'— Gregorian
- 'C'— Calendar
- 'E'— European
- 'J'— Julian
- The name of a user-defined variable data field containing the date format

Examples:*Example 1*

In this example, the date change function is used to convert January 28, 1958, from Gregorian to calendar format:

```
compute calendar = datechg(580128,'g','c') group by all
```

```
012858
```

Example 2

In this example, the date change function is used to convert November 12, 1991, from Julian to Gregorian format. The returned date contains the century portion of the year:

```
compute calendar = jgdate(91316,'j','g')
```

```
911112
```

Example 3

In this example, the date change function is used to convert January 28, 1958, from Gregorian to calendar format. The returned date contains the century portion of the year:

```
compute calendar = datechg(19580128,'g','c') group by all
```

```
01281958
```

Date Difference

The date difference function returns the number of days between two specified dates.

Syntax:

►► DATEDIF (*gregorian-date1*, *gregorian-date2*) ◀◀
[DATEDIFX]

Invocation names:

DATEDIF

Parameters:

gregorian-date1

Specifies the date, in Gregorian format, from which the second date is subtracted to derive the difference in days.

gregorian-date2

Specifies the date, in Gregorian format, subtracted from the first date to derive the difference in days.

Example:

This example determines the number of days between January 28, 1958, and August 11, 1955:

```
select datedif(580128,550811) as 'older' from table1
```

```
    OLDER = 901
```

Note: If the dates were supplied in reverse order, the value -901 would have been returned.

Date Offset

The date offset function returns the date, in Gregorian format, resulting from adding a specified number of days to a specified date.

Syntax:

►► DATEOFF (*gregorian-date*, *offset*) ◀◀
[DATEOFFX]

Invocation names:

DATEOFF

Parameters:**gregorian-date**

Specifies the date, in Gregorian format, to which the offset is added.

offset

Specifies the offset, in days, that is added to the specified date. *Offset* can be a numeric variable data field, a numeric literal, a built-in function that returns a numeric value, or an arithmetic expression. *Offset* can be negative.

Example:

This example determines that date that results from adding four days to January 28, 1958:

```
compute newdate = dateoff(580128,4)
```

```
NEWDATE = 580201
```

Extract

The extract function returns the string that results from removing leading and trailing spaces from a string value.

Syntax:

► `[EXT | EXTRACT] (string)` —————►

Invocation names:

EXT

EXTRACT

Parameters:**(string)**

Specifies the string value on which the extract function is performed.

Example:

This example removes leading and trailing spaces from the string contained in EMP-LAST-NAME-0415. The value of EMP-LAST-NAME-0415 is ' VON BUREN '.

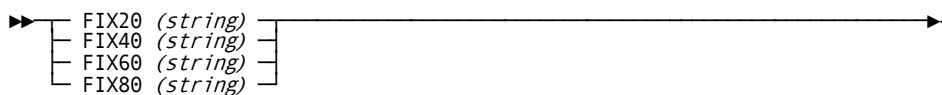
```
select extract(emp-last-name-0415) from employee

'VON BUREN'
```

Fix

Returns a fixed-length string of 20, 40, 60, or 80 characters.

FIX pads with blanks or truncates the string to the appropriate size. FIX is useful for making SELECT statements UNION compatible.

Syntax:**Invocation names:**

```
FIX20  FIX60
FIX40  FIX80
```

Parameters:**string**

Specifies the string value on which the fix function is performed.

String can be:

- A string literal, enclosed in single quotation marks
- The name of a user-defined variable data field containing the string

Example:

In the example below, FIX40 (in OLQ access mode) is used to create a formatted mailing address report:

```
select fix40(concat(substr(emp-fname,1,9),' ',emp-lname))
      emp-lname, emp-id, 1
      from employee
union
select fix40(street),
      emp-lname, emp-id, 2
      from employee
union
select fix40(concat(substr(city,1,10),' ',state,' ',zip-code),
      emp-lname, emp-id, 3
      from employee
union
select fix40(' ') as detail,
      emp-lname, emp-id, 9 as line
      from employee order by 2, 3, 4
```

```
JOHN MULLHOLLOW
114 LAUREL LANE
SHELBURNE FALLS, MA 01210
```

Index

The index function returns the starting position of a specified string within a string value. If the specified string is not found, a zero is returned.

Syntax:

```
▶▶ [ INDX | INDEX | STRING-INDEX ] (string, search-string) ▶▶
```

Invocation names:

```
INDX
INDEX
STRING-INDEX
```

Parameters:

string

Specifies the string that is searched.

search-string

Specifies the string that the index function searches for within *string*. *Search-string* cannot be longer than *string*.

Example:

This example lists invoices where the product code contains the string 'ABC':

```
select * from invoice where index(prod-code, 'abc') ne 0  
  
1
```

The above select returns all rows whose product code contains the literal, 'abc', somewhere in the string. For instance, the search strings 'ABCDEF' and 'XXXABC' produce success; 'CBAXXX' does not.

Initial Uppercase

Returns the string that results when the first letter in the specified source string is capitalized and all other characters in the string are converted to lowercase.

Syntax:

▶▶ INITCAP (*string*) ◀◀

Invocation names:

INITCAP

Parameters:**string**

Specifies the string whose first letter is to be capitalized.

String can be:

- A string literal enclosed in single quotation marks
- The name of a user-defined variable data field containing the string

Example:

In the example below, the initial cap function is used on employees' last names:

Initial value:

```
EMP-LNAME: 'O'HEARN      '
```

Statement:

```
compute new-emp-lname = initcap(emp-lname)
```

Returned string:

```
'O'hearn      '
```

Insert

The insert function returns the string that results from a specified string being inserted into a string value starting at a specified position.

Syntax:

```
►► [ INS | INSERT ] (string, insertion-string, starting-position) ◀◀
```

Invocation names:

```
INS
INSERT
```

Parameters:

string

Specifies the string into which *insertion-string* is inserted.

insertion-string

Specifies the string that is inserted into the string specified by *string*.

starting-position

Specifies the numeric starting position for the insertion. If *starting-position* is one or less, insertion starts at the beginning of the string value. If *starting-position* is greater than the length of *string*, insertion starts at the end of the string value.

Example:

This example flags the last named of employees living in Boston with asterisks '**':

```
select insert(emp-last-name-0415, '**', 1) from employee
where emp-city-0415 eq boston
```

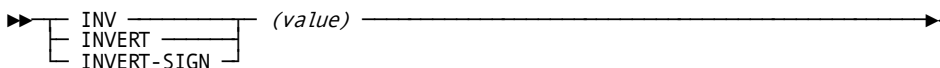
```
  '**OTT          '
  '**LIGARE       '

```

Invert Sign

The sign inversion function reverses the sign of a numeric value. A positive numeric value becomes negative; a negative numeric value becomes positive.

Syntax:



Invocation names:

- INV
- INVERT
- INVERT-SIGN

Parameters:**(value)**

Specifies the numeric value whose sign is to be reversed.

Example:

This example changes 453.29 from a positive to a negative value: The value of OPER1 is 453.29.

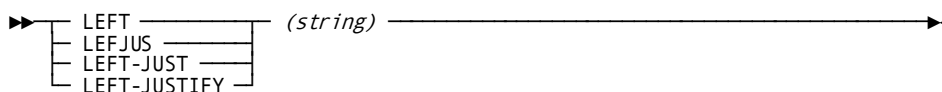
```
select inv(oper1) as 'inverted-value' from table1
```

```
INVERTED-VALUE = -453.29
```

Left Justify

The left-justify function returns the string that results from:

- Removing leading blanks from the left side of a string value
- Shifting the remainder of the string value to the left side
- Filling the right side with as many blanks as were removed from the left side

Syntax:**Invocation names:**

```
LEFT
LEFJUS
LEFT-JUST
LEFT-JUSTIFY
```

Parameters:**(string)**

Specifies the string value on which the left-justify function is performed.

Example:

This example left justifies the value contained in EMP-LAST-NAME-0415 (PIC X(15)): The value of EMP-LAST-NAME-0415 is ' OTT '.

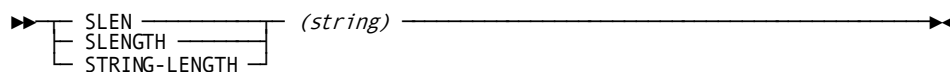
```
select lefjus(emp-last-name-0415) from employee
```

```
'OTT'
```

Length

The length function returns the length (number of characters) of a string value.

Syntax:



Invocation names:

SLEN
SLENGTH
STRING-LENGTH

Parameters:

(string)

Specifies the string value whose length is determined.

To calculate the length of a string value, excluding leading and trailing spaces, use the length function in conjunction with the extract function.

Example:

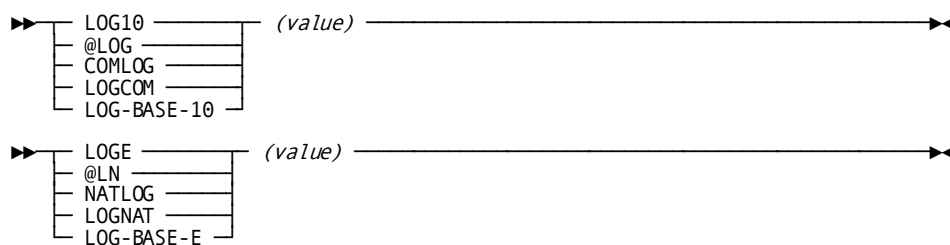
This example determines the length of a name contained in EMP-FIRST-NAME-0415 (PIC X(10)). The value of EMP-FIRST-NAME is 'ELMER '.

```
select slength(extract(emp-first-name-0415))  
from employee  
  
'ELMER'  
  
5
```

Logarithm

The logarithm functions return the common (base10) or natural (baseE) logarithm of a numeric value.

Syntax:



Invocation names:

@LOG	@LN
LOG10	LOGE
COMLOG	NATLOG
LOGCOM	LOGNAT
LOG-BASE-10	LOG-BASE-E

Parameters:

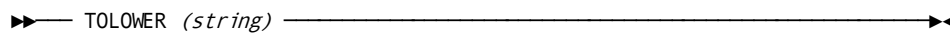
(value)

Specifies the numeric value whose logarithm is calculated. *Value* must be greater than zero.

Lowercase

Returns the string that results from converting all characters to lowercase.

Syntax:



Invocation names:

TOLOWER

Parameters:

string

Specifies the string value on which the lowercase function is performed.

String can be:

- A string literal, enclosed in single quotation marks
- The name of a user-defined variable data field containing the string

Example:

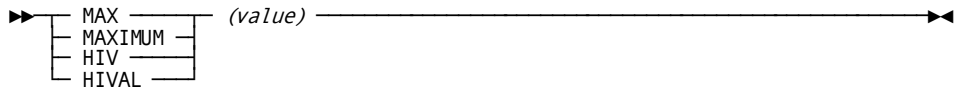
In the example below, the lowercase function is used to convert all characters in the last name to lowercase:

```
Initial value:
EMP-LNAME: 'LANCHESTER      '
Statement:
  compute new-emp-lname = tolower(emp-lname)
Returned string:
  'lanchester      '
```

Maximum

The maximum function returns the highest value in the specified column for all occurrences with the same GROUP BY value.

Syntax:



Invocation names:

MAX HIV
MAXIMUM HIVAL

Parameters:**(value)**

The column or expression for which the maximum value is calculated.

Example:

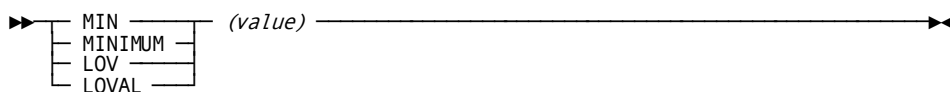
This example determines the maximum salary in each department:

```
select max(salary) as 'top salary' from employee
```

```
TOP SALARY = '75000'
```

Minimum

The minimum function returns the lowest value in the specified column for all occurrences with the same GROUP BY value.

Syntax:**Invocation names:**

```
MIN      LOV  
MINIMUM  LOVAL
```

Parameters:**(value)**

The column or expression for which the minimum value is calculated.

Example:

This example calculates the lowest salary in each department.

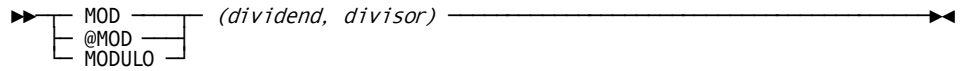
```
select min(salary) as 'low salary' from employee
```

```
LOW SALARY = '17000'
```

Modulo

The modulo function returns the modulus (remainder) of one numeric value divided by another.

Syntax:



Invocation names:

MOD
@MOD
MODULO

Parameters:

dividend

Specifies the numeric value that is divided by *divisor*.

divisor

Specifies the numeric value that is divided into *dividend*.

Example:

This example determines the remainder resulting from the division of two numeric values. The initial value of OPER1 is 43, and the initial value of OPER2 is 10.

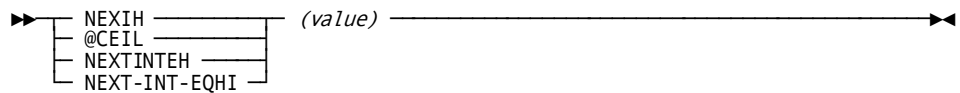
```
select mod(oper1,oper2) as 'remainder' from table1
```

```
REMAINDER = 3
```

Next Integer Equal or Higher

The next integer equal or higher function returns the smallest integer that is equal to or greater than a numeric value.

Syntax:



Invocation names:

@CEIL
 NEXIH
 NEXTINTEH
 NEXT-INT-EQHI

Parameters:**(value)**

Specifies the numeric value to be rounded up to the next integer.

Example:

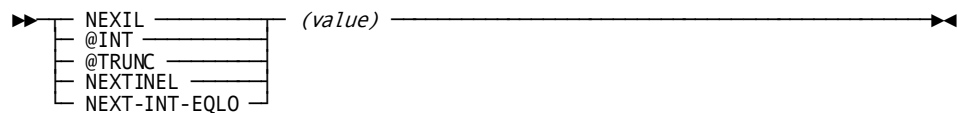
This example raises a balance due amount to the next higher dollar value. The initial value of BALANCE-DUE is 453.29.

```
select nexih(balance-due) as 'Balance-due' from invoice
```

```
BALANCE-DUE = 454
```

Next Integer Equal or Lower

The next integer equal or lower function returns the largest integer that is equal to or less than a numeric value.

Syntax:**Invocation names:**

@INT
 NEXIL
 @TRUNC
 NEXTINEL
 NEXT-INT-EQLO

Parameters:

(value)

Specifies the numeric value to be rounded down to the next integer.

Example:

This example rounds a balance due amount to the next lower dollar value. The initial value of BALANCE-DUE is 453.29.

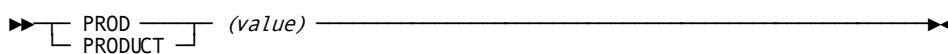
```
select nextil(balance-due) as 'balance-due' from invoice
```

```
BALANCE-DUE = 453
```

Product

Returns the product of all values in a column.

Syntax:



Invocation names:

PROD
PRODUCT

Parameters:

column-name

Specifies the list of values on which the product is calculated.

Example:

In this example, the compounded interest rate for 2 mutual funds is calculated:

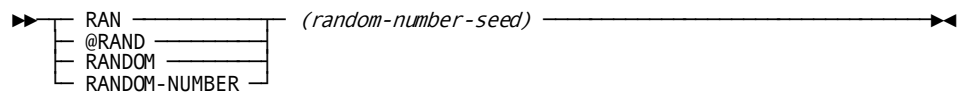
```
select product(interest-rate) from mutual-funds
```

Random Number

The random number function returns a pseudo-random number based on a seed numeric value. The returned random number is greater than zero and less than one, and has a length of nine decimal places.

We suggest you define the seed value with a picture of 9(9) and that you move the result of the function to a variable with a picture of V9(9).

Syntax:



Invocation names:

RAN
@RAND
RANDOM
RANDOM-NUMBER

Parameters:

(random-number-seed)

Specifies the numeric variable data field that contains the seed value from which the pseudo-random number is calculated. *Random-number-seed* cannot be zero.

Example:

This example calculates a random number from the seed value of 13549:

```
compute 'number' = random (13549)
```

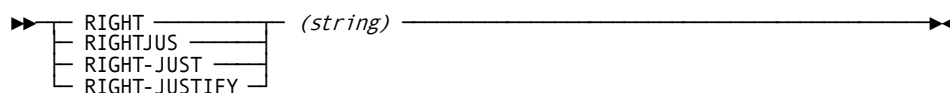
```
NUMBER = 0.627847111
```

Right Justify

The right-justify function returns the string that results from:

- Removing blanks on the right side of a string value
- Shifting the remainder of the string value to the right side
- Filling the left side with as many blanks as were removed from the right side

Syntax:



Invocation names:

RIGHT
RIGHTJUS
RIGHT-JUST
RIGHT-JUSTIFY

Parameters:

(string)

Specifies the string value that is right justified.

Example:

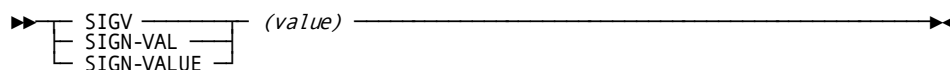
This example right justifies EMP-LAST-NAME-0415 (PICX(15)). The initial value of EMP-LAST-NAME-0415 is ' OTT '.

```
select rightjus(emp-last-name-0415) from employee  
  
          OTT'
```

Sign Value

The sign value function returns a +1, 0, or -1, depending on whether a numeric value is positive, zero, or negative, respectively.

Syntax:



Invocation names:

SIGV
SIGN-VAL
SIGN-VALUE

Parameters:**(value)**

Specifies the numeric value whose sign is determined.

Example:

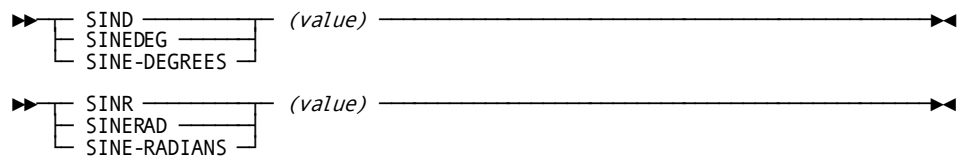
This example moves a zero to the SIGN column if an amount is negative, a one to the column if the amount is positive, or a zero to the column if the amount is zero. The initial value of OPER1 is -453.29.

```
select sigv(oper1) as 'sign' from table1
```

```
SIGN = -1
```

Sine

The sine functions return the sine of a numeric value that represents an angle in either degrees or radians.

Syntax:**Invocation names:**

SIND SINR
SINEDEG SINERAD
SINE-DEGREES SINE-RADIANS

Parameters:

(value)

Specifies the numeric value representing the angle, in degrees or radians, whose sine is calculated.

Example:

This example calculates the sine of -60 (in degrees):

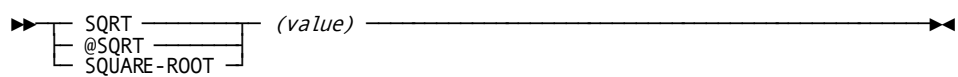
```
compute 'sine' = sind(-60)
```

```
SINE = -8.6602
```

Square Root

The square-root function returns the square root of a numeric value.

Syntax:



Invocation names:

Sqrt
@Sqrt
Square-Root

Parameters:

(value)

Specifies the numeric value whose square root is calculated. *Value* cannot be a negative number.

Example:

This example calculates the square root of OPER1. The initial value of OPER1 is 256.

```
select sqrt(oper1) as 'square root' from table1
```

```
SQUARE ROOT = 16
```


Standard deviation

Returns the standard deviation of the values in *column-name* (the square root of the average differences between the values and their arithmetic mean).

The standard deviation function uses the *n-1* method (*sample* standard deviation).

This function calculates standard deviation using the formula shown below:

$$\sqrt{\frac{\sum_{i=1}^n x_i^2 - \left(\frac{\sum_{i=1}^n x_i}{n}\right)^2}{n-1}}$$

Syntax:

► STD (*column-name*) ◄

Invocation names:

STD

Parameters:

column-name

Specifies the list of values on which the standard deviation is calculated.

Example:

This example displays all rows where the standard deviation of SALARY is greater than \$5,000:

```
select std(salary) as 'SALARY STAND-DEV' from employee
group by dept-id having std(salary) > 5000
```

Standard deviation population

Returns the standard deviation population of the values in *column-name* (the square root of the average differences between the values and their arithmetic mean).

The standard deviation population (STDP) function uses the *n* method (*absolute* standard deviation).

This function calculates standard deviation using the formula shown below:

$$\sqrt{\frac{\sum_{i=1}^n x_i^2 - \left(\frac{\sum_{i=1}^n x_i}{n}\right)^2}{n}}$$

Syntax:

► STDP (*column-name*)

Invocation names:

STDP

Parameters:

column-name

Specifies the list of values on which the standard deviation is calculated.

Example:

This example displays all rows where the standard deviation population of SALARY is greater than \$5,000:

```
select stdp(salary) as 'SALARY STAND-DEV' from employee
group by dept-id having stdp(salary) > 5000
```

Substring

The substring function returns the substring of a string value, starting from a specified position and continuing for a specified length.

Syntax:

► SUBS (*string*, *starting-position* , *length*)

SUBS
 SUBSTR
 SUBSTRING

Invocation names:

SUBS
SUBSTR
SUBSTRING

Parameters:**string**

Specifies the string value from which the substring is taken.

starting-position

Specifies the numeric starting position of the substring within the string value. *Starting-position* must be a positive number and not greater than the length of the string value.

length

Specifies the numeric length of the substring within the string value (optional). The sum of *starting-position* and *length*, minus one, cannot be greater than the length of the string value. If *length-evn* is not specified, the substring is taken from the specified starting position to the end of the string value.

Example:

This example extracts the first seven characters of each employee's last name.

```
select substr(emp-last-name-0415,1,7) from employee
```

```
EMPLOYEE REPORT
09/15/96
```

```
LITERAT
HEAROWI
PAPAZEU
HEDGEHO
MCDOUGA
```

Sum

The sum function returns the sum of the specified numeric column with the same GROUP BY value.

Syntax:

```

▶▶ [SUM | TOTAL | TOT] (value)

```

Invocation names:

SUM TOTAL
 TOT

Parameters:

(value)

Specifies the column or columns to be totaled.

Example:

In this example, you want to find out the total amount of money being spent on salaries in each department:

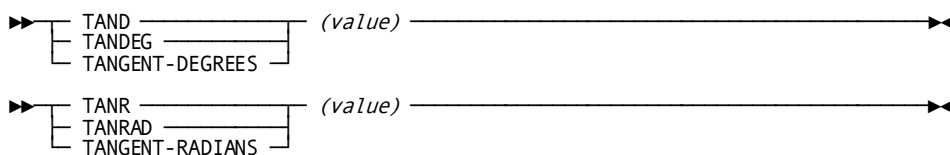
```
select sum(salary-amount-0420) as 'total salaries'  
from emposition
```

```
                                EMPOSITION REPORT  
                                09/22/96  
  
                                TOTAL SALARIES  
  
                                3187500.00  
  
END OF REPORT
```

Tangent

The tangent functions return the tangent of a numeric value that represents an angle in either degrees or radians.

Syntax:



Invocation names:

TAND TANR
TANDEG TANRAD
TANGENT-DEGREES TANGENT-RADIANS

Parameters:**(value)**

Specifies the numeric value representing the angle, in degrees or radians, whose tangent is calculated.

The following rules apply to using *value*:

- For TANGENT-DEGREES, *value* cannot be a value equal to the following expression, where *n* is any integer:

$$(n * 180) + 90$$

Thus, *value* cannot equal values such as -270, +270, -90, or +90 for this function.

- For TANGENT-RADIANS, *value* cannot be a value equal to the following expression:

$$(n * X) + X/2$$

Thus, *value* cannot equal values such as -X/2 or +X/2 for this function.

Example:

This example calculates the tangent of 60 (in degrees):

```
compute 'tangent' = tand(60)
```

```
TANGENT = 1.7321
```

Today

Returns today's date in the format requested.

Syntax:

```
►► [ TODAY ] ( 'output-date-format' )
```

Invocation names:

TODAY
TODAYX

Parameters:

TODAY/TODAYX

Invokes the today function. TODAYX returns a date that contains the century portion of the year.

(output-date-format)

Specifies the output date format. *Output-date-format* can be:

- 'C' for calendar
- 'E' for European
- 'G' for Gregorian
- 'J' for Julian
- The name of a user-defined variable data field containing the date format

Examples:

Example 1

In the example below, the today function is used to display today's date in the calendar format (where today is March 17, 1996):

Statement:
 compute sys-date = today('c')
Returned value: 031796

Example 2

In the example below, the today function is used to display today's date in the calendar format (where today is March 17, 1996). The returned date contains the century portion of the year:

Statement:
 compute sys-date = todayx('c')
Returned value: 03171996

Note: In the example above, SYS-DATE must contain the century portion of the year.

Tomorrow

Returns tomorrow's date in the format requested.

Syntax:

►

┌	TOMORROW	└	('output-date-format')	—————▶
	TOMORROWX			

Invocation names:

TOMORROW
TOMORROWX

Parameters:

TOMORROW/TOMORROWX

Invokes the tomorrow function. TOMORROWX returns a date that contains the century portion of the year.

'date-format'

Specifies the output date format. *Output-date-format* can be:

- 'C' for calendar
- 'E' for European
- 'G' for Gregorian
- 'J' for Julian
- The name of a user-defined variable data field containing the date format

Examples:

Example 1

In the example below, the tomorrow function is used to display tomorrow's date in the calendar format (where today is March 17, 1996):

Statement:
`compute sys-date = tomorrow('c')`
 Returned value: 031896

Example 2

In the example below, the tomorrowx function is used to display tomorrow's date in the calendar format (where today is March 17, 1996). The returned date contains the century portion of the year:

Statement:
`compute sys-date = tomorrowx('c')`
 Returned value: 03181996

Note: In the example above, SYS-DATE must contain the century portion of the year.

Translate

The translate function returns a string that results from:

- Extracting characters from an **original string** that match a **selection string**
- Replacing the extracted characters with corresponding characters in a substitution string

Characters in a selection string correspond positionally to characters in a substitution string. The first character in the selection string corresponds to the first character in the substitution string, the second corresponds to the second, and so forth.

Example:

If the selection string contains the letter A in its tenth position, each occurrence of A in the original string is replaced by the tenth character in the substitution string.

Syntax:

► `TRANS` `TRANSLATE` (`string`, `substitution-string`, `selection-string`) ◀

Invocation names:

TRANS
 TRANSLATE

Parameters:**string**

Specifies the variable on which the translate function is performed.

substitution-string

Specifies the substitution string.

selection-string

Specifies the selection string (optional):

- If *selection-string* is longer than *substitution-string*, the excess characters correspond to blanks.
- If *selection-string* specifies the same character more than once, the translate function uses the first occurrence of the character.
- If *selection-string* is not specified, the 256-character EBCDIC table is used, consisting of hexadecimal 00 through FF.

Example:

This example translates all occurrences of 1, 2, and 3 in course ID values to A, B, and C, respectively. The initial value of COURSE-ID is '321 '.

```
select trans(course-id, '123', 'abc')
from course-list
```

```
      ' CBA          '
```

Uppercase

Returns the string that results from converting all characters to uppercase.

Syntax:

```
►► TOUPPER (string) ◀◀
```

Invocation names:

TOUPPER

Parameters:

string

Specifies the string value on which the uppercase function is performed.

String can be:

- A string literal, enclosed in single quotation marks
- The name of a user-defined variable data field containing the string

Example:

In the example below, the uppercase function is used to convert all characters in the last name to uppercase:

Initial value:

EMP-LNAME: 'Lanchester'

Statement:

compute new-emp-lname = toupper(emp-lname)

Returned string:

'LANCHESTER'

Variance

Returns the variance of the values in *column-name* (the square of the standard deviation).

The variance function uses the *n-1, sample*, method.

This function calculates variance using the formula shown below:

$$\frac{\sum_{i=1}^n x_i^2 - \left(\frac{\sum_{i=1}^n x_i}{n} \right)^2}{n-1}$$

Syntax:

►► VAR (*column-name*) ◀◀

Invocation names:

VAR

Parameters:**column-name**

Specifies the list of values on which the variance is calculated.

Example:

The variance function in this example displays the variance between the rows in SALARY-MANAGER:

```
select var(salary-manager) as 'VARIANCE SALARY'
from employee group by job-id
```

Variance population

Returns the variance population of the values in *column-name* (the square of the standard deviation population).

The variance population function uses the *n, absolute*, method.

This function calculates variance using the formula shown below:

$$\frac{\sum_{i=1}^n x_i^2 - \left(\frac{\sum_{i=1}^n x_i}{n} \right)^2}{n}$$

Syntax:

►► — VARP — (*column-name*) ————— ◀◀

Invocation names:

VARP

Parameters:**column-name**

Specifies the list of values on which the variance population is calculated.

Example:

The variance function in this example displays the variance population between the rows in SALARY-MANAGER:

```
select varp(salary-manager) as 'SALARY VARP'  
from employee group by job-id
```

Verify

The verify function returns the position of the first character in a string value that does not occur in a verification string. If every character in the object string value occurs in the verification string, a zero is returned.

Syntax:

► VER
VERIFY (string, verification-string) ◀

Invocation names:VER
VERIFY**Parameters:****string**

Specifies the string value on which the verify function is performed.

verification-string

Specifies the string value against whose characters the string value's characters are verified.

Example:

This example uses the verify function to return EMP-ID-0415 values that contain numeric values. The initial value of EMP-ID-0415 is 02B6.

```
select emp-last-name-0415 from employee
where verify(emp-id-0415,'1234567890') ne 0
```

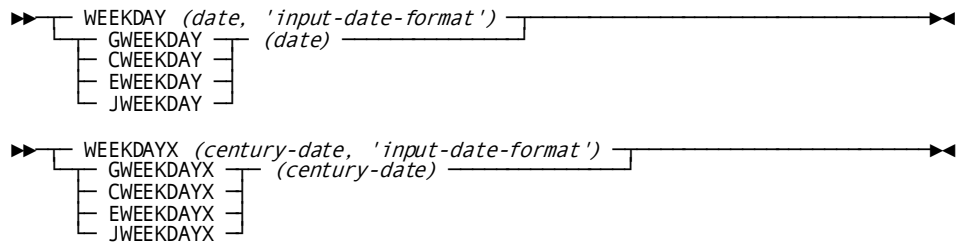
If EMP-ID-0415 contains only numbers and blanks, the verify function returns a zero. If EMP-ID-0415 contains any other characters, the function returns the position of the first character that is not in the verification string.

3

Weekday

The weekday functions return the name of the day that corresponds to a specified date.

Syntax:



Invocation names:

WEEKDAY		WEEKDAYX	
GWEEKDAY	EWEEKDAY	GWEEKDAYX	EWEEKDAYX
CWEEKDAY	JWEEKDAY	CWEEKDAYX	JWEEKDAYX

Parameters:

date

Specifies the six-byte object date for the weekday function.

Date can be:

- A user-supplied numeric literal (enclosed within quotes)
- The name of a user-defined variable data field

century-date

Specifies the eight-byte object date (containing the century) for the weekday function.

Century date can be:

- A user-supplied numeric literal (enclosed within quotes)
- The name of a user-defined variable data field

'input-date-format'

Specifies the format of the date specified by *date* or *century-date*.

Input date can be:

- 'G'— Gregorian
- 'C'— Calendar
- 'E'— European
- 'J'— Julian
- The name of a user-defined variable data field containing the date format

Examples:*Example 1*

This example determines the weekday on which January 28, 1958, fell; the date is provided in calendar format:

```
compute weekday = weekday(012858, 'c')
```

```
'TUESDAY'
```

Example 2

This example determines the weekday on which November 12, 1991 fell; the date is provided in Julian format:

```
compute weekday = jweekday(91316)
```

```
'TUESDAY'
```

Example 3

This example determines the weekday on which January 28, 1958, fell; the date is provided in calendar format. The input date contains the century portion of the year:

```
compute weekday = weekdayx(01281958, 'c')
```

```
'TUESDAY'
```

Yesterday

Returns yesterday's date in the format requested.

Syntax:

```
►► [ YESTERDAY ] ( 'output-date-format' ) ───────────────────────────────────►►
   [ YESTERDAYX ]
```

Invocation names:

YESTERDAY
YESTERDAYX

Parameters:

YESTERDAY/YESTERDAYX

Invokes the yesterday function. YESTERDAYX returns a date that contains the century portion of the year.

'output-date-format'

Specifies the output date format. Output date can be expressed using:

- 'C' for calendar
- 'E' for European
- 'G' for Gregorian
- 'J' for Julian
- The name of a user-defined variable data field that contains the date format

Examples:

Example 1

In the example below, the yesterday function is used to display yesterday's date in the calendar format (where today is March 17, 1996):

Statement:
 compute sys-date = yesterday('c')
Returned value: 031696

Example 2

In the example below, the yesterdayx function is used to display yesterday's date in the calendar format (where today is March 17, 1996). The returned date contains the century portion of the year:

Statement:
 compute sys-date = yesterdayx('c')
Returned value: 03161996

Note: In the example above, SYS-DATE must contain the century portion of the year.

Chapter 8: Tailoring the CA OLQ Environment for Ease of Use

This section contains the following topics:

[What This Chapter Is About](#) (see page 265)

[Data Retrieval](#) (see page 265)

[Using qfiles](#) (see page 268)

[Executing qfiles](#) (see page 269)

[Defining Report Headers](#) (see page 277)

[Synonyms](#) (see page 277)

[Code Tables](#) (see page 278)

[Date Option](#) (see page 279)

[External Pictures](#) (see page 280)

What This Chapter Is About

You can tailor CA OLQ to suit each user's needs and facilitate use by predefining your CA OLQ environment as much as possible in the data dictionary through Integrated Data Dictionary (IDD) and during system generation.

This chapter presents the options available for tailoring the CA OLQ environment for ease of use.

Data Retrieval

In order to facilitate data retrieval in CA OLQ, we encourage the use of:

- The **SELECT (IDMS access mode)** statement— If you have the CA IDMS/DB SQL Option
- The **SELECT (OLQ access mode)** statement— Automatically generates paths
- **Logical records**— Predefine paths through logical records to dictate user access

SELECT (IDMS access mode)

In order to use SELECT to access SQL tables, you must set the access switch to **idms**.

How to specify the access switch:

The access switch can be set in the following ways:

1. At system generation time
2. For an individual user in the Dictionary (IDD)
3. For the session, interactively (or until the switch is changed)

Note: For more information about setting the access switch, see [SET](#) (see page 186).

Allowing MRR:

Multiple Record Retrieval (MRR) must be allowed before you can use the SELECT (IDMS access mode) statement.

MRR is set in the Dictionary (IDD) with the USER statement.

Note: For more information about the the USER statement, see the *CA IDMS IDD Quick Reference Guide*.

SELECT (OLQ access mode)

Retrieving

The SELECT (OLQ access mode) statement allows you to retrieve multiple records with one command and also to sort and group the information during retrieval.

To enable use of SELECT, set the multiple record retrieval option in IDD. The syntax for the IDD statement is:

```
ADD/MODIFY USER user-name ...  
MRR IS ALLOWED
```

Sorting

The **ORDER BY** clause of the SELECT statement allows you to sort information during retrieval instead of after retrieval. Sorting during retrieval eliminates the need for entering sort commands after the report is built.

To use the ORDER BY clause you must allow sorts through IDD. If the sort option in IDD is not allowed, your users cannot use the ORDER BY clause of the SELECT statement. The syntax for the IDD statement is:

```
ADD/MODIFY USER user-name ...  
SORT IS ALLOWED
```

Note: The DISTINCT option implies that CA OLQ does a SORT

Logical Records

With the Logical Record Facility (LRF), you can code logical records (combinations of database records) to decrease the number of calls made to CA IDMS/DB by a CA OLQ query.

With LRF you can eliminate the need for users to choose fields. You can code a logical record subschema and associate it with certain users. The subschema may contain only logical records and dictate what the user can retrieve.

Additionally, you can code many different paths to retrieve records from the database.

Note: For more information about the uses of LRF, see the *CA IDMS Logical Record Facility Guide*.

Using LRF

Example:

Without a logical record and path, a user would generate an employee salary report with:

```
select salary-amount-0420, emp-id-0415,  
       emp-name-0415, dept-id-0410, dept-name-0410  
from emposition, employee, department  
where emp-emposition and dept-employee
```

With a logical record and path coded by the DBA, the user has two choices for generating the report:

- **In command mode:**

```
select * from salary-report
```

Where 'salary-report' is the name of the logical record, and the * (asterisk) stands for all fields within the logical record.

- **In menu mode:**

Select the logical record from the Record Select screen.

Note: For instructions on coding retrieval paths using LRF, see the *CA IDMS Logical Record Facility Guide*.

Using qfiles

A qfile is a sequence of CA OLQ commands used to build reports. Qfiles can be invoked both online and in batch. Qfiles are stored in the data dictionary and can contain almost any CA OLQ command. Commands not valid for the current CA OLQ environment are flagged or ignored by CA OLQ during execution. With qfiles, you can set up defaults for the CA OLQ operating environment, as well as construct reports.

This function calculates variance

[Command Summary](#) (see page 15)

[Commands and Syntax](#) (see page 53)

Building qfiles

You can build qfiles either of two ways:

- Manually through IDD
- Automatically in CA OLQ

Note: For the syntax required to build qfiles in IDD, see the *CA IDMS IDD Quick Reference Guide*.

To build qfiles in CA OLQ you need only create a report in menu mode or command mode, and then issue a `SAVE QFILE qfile-name` command. CA OLQ automatically creates a qfile containing the commands necessary to rebuild that report.

When you save a qfile with the `SAVE QFILE` command, the qfile is stored in the data dictionary.

In CA OLQ, you can replace, delete, and save a qfile. The commands that allow you to maintain qfiles in command mode include:

- `SAVE QFILE qfile-name`
- `SAVE QFILE qfile-name REPLACE`
- `DELETE QFILE qfile-name`

In IDD, you can add, modify, and delete qfiles. The commands that allow you to maintain qfiles in IDD include:

- `ADD QFILE NAME IS qfile-name VERSION IS version-number`
- `MODIFY QFILE NAME IS qfile-name VERSION IS version-number`
- `DELETE QFILE NAME IS qfile-name VERSION IS version-number`

Executing qfiles

Qfiles are executed by issuing the `QFILE qfile-name` command. Qfiles can be executed only after being saved.

To execute a qfile named EMP-REPORT, issue the command:

```
qfile emp-report
```

The keyword QFILE is optional.

Reporting on qfiles

You can generate reports that list each qfile and its associated source statements (CA OLQ commands) with the CA IDMS/DB dictionary report utility (IDMSRPTS). You can also display the definitions of all qfiles stored in the data dictionary with the *Data Dictionary Reporter (DDR)*.

Generating qfile reports

Qfile definitions are output on the module reports. An alternative method for displaying the qfile source is to use the DDDL command:

Note: For the job control language required to execute DDR, see the *CA IDMS Reports Guide*.

```
DISPLAY qfile-name AS SYNTAX
```

Note: For more information about using the dictionary to set up qfiles, see [Integrated Data Dictionary Options](#) (see page 349).

Special Uses of qfiles

There are three types of qfiles automatically invoked in the following order by CA OLQ at signon:

- **PF key module**— Defines the defaults for PF keys in command mode, and can contain default settings for other CA OLQ options.
The PF key module is associated with the CA OLQ task code and is invoked first.
- **Signon profile qfile**— Establishes a CA OLQ environment for a user upon entry to CA OLQ.
The signon profile qfile is associated with a user ID and is invoked after the pfkey module. The options set in this qfile override those set in the PF key module.
- **Signon qfile**— Automatically defines a user's access to a specific subschema at signon to that subschema.
The signon qfile is associated with a subschema (which is in turn associated with a user). The signon qfile is invoked after the signon profile qfile and only when security for CA OLQ is on. The options set in this qfile override those set in the signon profile qfile.

PF Key Module

A PF key module is executed each time a user issues the CA OLQ task code. The PF key module is not associated with specific users, but with the CA OLQ task code.

How to create a PF key module

You create a PF key module as a qfile with IDD, save it in the dictionary, and define it as the pfkey module during system generation:

```
ADD/MODIFY OLQ ...  
PFKEY MODULE IS qfile-name
```

You can also include global settings in the PF key module. For example, a PF key module might look like:

```
SET FUNCTION 1 = 'HELP'  
SET FUNCTION 2 = 'HELP COMMANDS'  
SET FUNCTION 3 = 'QUIT'  
SET FUNCTION 4 = 'HELP COMMANDS'  
SET FUNCTION 5 = 'DISPLAY'  
SET FUNCTION 6 = 'MENU'  
SET FUNCTION 7 = 'PAGE BACK'  
SET FUNCTION 8 = 'PAGE FORWARD'  
SET FUNCTION 9 = 'SWAP'  
SET DATE 'Month DD, YYYY'
```

Signon Profile qfiles

A **signon profile qfile** is a set of commands automatically executed upon user signon to CA OLQ. Signon profile qfiles are **user specific** and can contain PF key assignments, CA OLQ option settings, SET commands, and other global CA OLQ settings.

How to create a signon profile qfile

Create signon profile qfiles through the IDD, and save the sequence of commands as a named qfile in the data dictionary.

You can use IDD to associate a signon profile qfile with a particular user:

```
ADD/MODIFY/REPLACE USER ...  
INCLUDE SIGNON PROFILE IS qfile-name  
VERSION IS version  
LANGUAGE IS OLQ
```

Signon qfiles

The signon qfile is associated with a user and a subschema. Signon qfiles execute automatically when a user signs on to a subschema.

A signon qfile allows you to make the users' jobs easier; you can state the field lists for records, and set the CA OLQ report default options to format reports automatically.

How to create a signon qfile

You create a signon qfile through IDD, and save the sequence of commands as a named qfile in the data dictionary. You then associate the qfile with a subschema through IDD.

Once the qfile is saved, associate the qfile with a particular subschema and define it as the signon qfile for a particular user:

```
ADD/MODIFY/REPLACE USER ...
INCLUDE ACCESS TO SUBSCHEMA ss-name
OF SCHEMA schema-name
VERSION IS version
.
.
.
SIGNON QFILE IS qfile-name
VERSION IS version
```

Below is an example of what a signon qfile might look like:

```
FIELDS FOR EMPLOYEE ARE EMP-ID-0415,
EMP-LAST-NAME-0415, SS-NUMBER-0415
FIELDS FOR DEPARTMENT ARE DEPT-ID-0410, DEPT-HEAD-ID-0410
FIELDS FOR EMPOSITION ARE SALARY-AMOUNT-0420, SKILL-NAME-0420
EDIT EMPLOYEE.SS-NUMBER-0415 PICTURE=99-999-9999
```

Keep in mind that this qfile does not limit retrieval; if users wish to modify the field lists after retrieval, they can use an additional FIELDS FOR command.

Including Parameters in qfiles

When building qfiles, you can include **parameters** and **nest** qfiles:

- **Parameters** enable you to produce variations of a report with a single qfile.
- **Nested** qfiles provide the same type of flexibility as subroutines in computer programs. To switch execution from one qfile to another, embed **QFILE=qfile-name** commands within a qfile. These commands invoke previously stored qfiles.

Access mode: You **cannot** use parameters in qfiles dictionary when the access switch is set to **IDMS**.

Parameters:

You can produce variations of a report using a single qfile. You define parameters in a CA OLQ command into which a user-defined text string can be inserted at execution time.

Access mode: You **cannot** use parameters in qfiles dictionary when the access switch is set to **IDMS**.

Example:

You can tailor a qfile to produce standard summary reports for each department. The department ID is identified as the parameter. You can change the definition of the parameter to reflect the department you want to report on.

How Parameters work:

You include a parameter in a qfile where you want to change the value of something. For instance; replace a field name in order to change the field retrieved, or a qfile name in order to change the next qfile invoked.

Keep in mind that these parameters are positional. The order in which you specify them in the qfile is the exact order you must specify them when executing the qfile.

In the following example, the department ID value (6666) is replaced with a parameter in order to change the department being reported on.

Original SELECT:

```
select * from department, employee, emposition -
  where dept-id-0410 = 6666
```

SELECT with variable parameter:

```
select * from department, employee, emposition -
  where dept-id-0410 = &dept-no.
```

You define the parameter in the first line of the qfile, and if you want to change the default, you specify the value at execution:

```
&dept-no. = 6666
select * from department, employee, emposition -
  where dept-id-0410 = &dept-no.
```

If you wanted to override the default, your QFILE command would look like:

```
qfile emp-report 2364
```

(Where 2364 is the department number)

Defining Parameters:

Once you have your parameters in the qfile, you must define them in the beginning of the qfile. You do this on the first line of the qfile. If you don't have any parameters to define, you **must** leave the first line of the qfile blank. The syntax of the parameter line is:

```
& parameter-name = parameter-value
```

The *parameter-value* you specify acts as the default. You can override the default by specifying a value at execution of the qfile:

```
QFILE qfile-name parameter-value
```

Guidelines for defining parameter statements follow:

- The parameter statement must be the first statement in the qfile.
- The parameters specified in the parameter statement must be in the order in which they appear in the qfile.
- If there are no parameters in the qfile, you must leave the first line of the qfile blank.

Multiple parameters in one command:

You can use more than one variable parameter in a command statement.

Example:

```
select &emp-field-name. &dept-field-name.  
from employee, department
```

The parameter statement that would appear as the first line in the qfile would read:

```
&emp-field-name. = EMP-LAST-NAME-0415 -  
&dept-field-name. = DEPT-ID-0410
```

Nesting qfiles

Nesting qfiles allows you to suspend processing of one qfile to execute another qfile, which, upon completion returns execution back to the first qfile.

Access mode: You **cannot** use parameters in qfiles dictionary when the access switch is set to **IDMS**.

What is nesting used for?

Using this method, you can execute a qfile that retrieves records, calls another qfile to format those records, and returns to the first qfile to then save the formatted records in a report file.

This would be especially helpful if you had one report you wanted formatted in different ways; for example, to format a department report to highlight the employee information, and format the same report to highlight the department information. You could have three qfiles:

- RETRIEVE-DATA— To retrieve the records
- FORMAT-EMP-REPORT— To format a report that highlights the employee information
- FORMAT-DEPT-REPORT— To format a report that highlights the department information

By nesting the qfiles, you could use the same qfile to build the report and then use a variable parameter to call a certain qfile, depending upon how you wanted the report formatted.

Example:

RETRIEVE-DATA retrieves records:

```
&format='. '  
select * from department, employee, emposition -  
  where dept-employee and emp-emposition  
&format.
```

In the example above, the first line (&format='. ') is the parameter definition statement. This statement dictates that the value of the parameter will be supplied at execution time. The last statement (&format). is the parameter that, when specified, calls another qfile.

At execution time, you can issue the command:

```
retrieve-data format-emp-report
```

The `FORMAT-EMP-REPORT` will be substituted in the parameter `&FORMAT.` at execution of the qfile and will result in the execution of qfile `FORMAT-EMP-REPORT` upon the completion of the qfile `RETRIEVE-DATA.`

How to nest qfiles

To nest qfiles, just enter a `QFILE qfile-name` command in the qfile where you want to transfer execution to the second qfile. When the nested qfile completes execution, control is returned to the first qfile at the command immediately after the last executed `QFILE` command.

Examples of Nesting qfiles

Nest to any depth:

You can nest qfiles to any depth, as long as the qfiles are not recursively called. If `QFILE-1` calls `QFILE-2`, then within the same nest of qfiles, `QFILE-2` *cannot* call `QFILE-1`.

Each nested qfile is invoked in turn until the last qfile call is encountered:

```
select * from department, employee
qfile format-report
save report dept-employee
```

At the **QFILE FORMAT-REPORT** command, the qfile named `FORMAT-REPORT` begins to execute:

```
sort on dept-id-0410 in ascending order -
  and on emp-id-0415 in ascending order
compute name-max=max(emp-name-0415)
group by dept-id-0410
```

When the called qfile (`FORMAT-REPORT`) finishes executing, control is returned to the first qfile at the next command statement:

```
save report dept-employee
```

Parameters across qfiles:

You can use parameters across nested qfiles. If you define a parameter in the first qfile, and use the same parameter in the second qfile, the definition stays the same unless you redefine the parameter in the second qfile.

If you do redefine the parameter in the second qfile, the parameter is automatically returned to its first definition when control is returned to the first qfile.

Defining Report Headers

Access mode: CA OLQ **does not** use headers associated with the desired record in the dictionary when the access switch is set to **IDMS**.

You can use IDD to define report headers for specific record fields. By defining report headers in the data dictionary, each user can generate meaningful reports without formatting the report headers each time the report is generated.

Use the IDD RECORD ELEMENT substatement:

```
REMOVE/REPLACE RECORD ELEMENT IS element-name ...
EDIT OLQ HEADER IS new-header
```

Example:

You could change the CA OLQ header from EMP-ID-0415 to EMPLOYEE ID:

```
replace record element is emp-id-0415
edit olq header is employee
```

The resulting report contains the column EMPLOYEE instead of the column EMP-ID-0415. The contents of the column are the same, only the headers have changed.

Note: For more information about using the RECORD ELEMENT statement in IDD, see the *CA IDMS IDD Quick Reference Guide*.

Synonyms

Access mode: CA OLQ **does not** use synonyms associated with the desired record in the dictionary when the access switch is set to **IDMS**.

You can use IDD to define synonyms for records and record elements (fields). Menu mode uses these synonyms instead of the record and record element names on appropriate screens. Command mode recognizes these synonyms as the names of records and record elements.

To define synonyms for records, use the IDD statement:

```
ADD/MODIFY RECORD ...
INCLUDE RECORD NAME SYNONYM IS synonym-name ...
LANGUAGE IS OLQ
```

Note: Note that LANGUAGE IS OLQ is required only for menu mode, in conjunction with the option SYNONYM/NOSYNONYM. Command mode uses any synonym, regardless of language.

To define synonyms for record elements, use the IDD statement:

```
REMOVE/REPLACE RECORD ELEMENT ...  
ELEMENT NAME SYNONYM ...  
IS element-synonym name
```

Code Tables

Access mode: CA OLQ does **not** use code tables in the dictionary with records containing fields that are tightly coupled when the access switch is set to **IDMS**.

You can define a small field to hold a value, and then use a code table to translate that value to the appropriate string upon display of the report.

Example:

In a grocery store, each type of apple has a code for the type:

```
01      for the Macintosh  
02      for the Granny Smith  
03      for the Cortland  
04      for the Golden Delicious  
05      for the Red Delicious
```

In the record, you can use the code value to identify the item. When you want to print the report, however, you want to see the name of the item. Associate a code table with the record, and when you display the record, the code table translates the code into the appropriate string. A sample code table would look like:

```
01      MACINTOSH  
02      GRAN. SMITH  
03      CORTLAND  
04      GOLD.DEL  
05      RED.DEL
```

To associate a code table with a field, use the IDD statement:

```
REMOVE/REPLACE RECORD ELEMENT ...  
INCLUDE CODE TABLE IS LIST/encode-value decode-value/NULL
```

Date Option

You can specify different date formats with the SET DATE command. You can set the format for an entire CA OLQ session by including the SET DATE command in the signon profile qfile or the PF key module. You can also specify a code table to be used when generating the month names.

```
ADD TABLE NAME IS OLQFRNCH VERSION IS 1
LANGUAGE IS TABLE
PUBLIC ACCESS IS ALLOWED FOR ALL
TYPE IS CODE
SEARCH IS LINEAR
ENCODE DATA IS ALPHANUMERIC
TABLE IS UNSORTED
  DUPLICATES ARE NOT ALLOWED
  VALUES ARE (
    01          JANVIER
    02          FEVRIER
    03          MARS
    04          AVRIL
    05          MAI
    06          JUIN
    07          JUILLET
    08          AOUT
    09          SEPTEMBRE
    10          OCTOBRE
    11          NOVEMBRE
    12          DECEMBRE
    .
    .
    .
  )
```

To use a codetable specify the SET CODETABLE command.

Including the date in reports:

You can include the current date in reports by specifying **\$DATE** within a report page header or footer. The current date will be substituted upon display or printing of the report.

For more information:

[Commands and Syntax](#) (see page 53)

External Pictures

Access mode: CA OLQ **does not** use external pictures associated with the desired record in the dictionary when the access switch is set to **IDMS**.

You can use IDD to define external pictures for specific report fields. By defining external pictures in the dictionary, each user can generate meaningful reports without formatting the report fields each time the report is generated.

Use the IDD RECORD ELEMENT substatement:

```
REMOVE/REPLACE RECORD ELEMENT ...  
EXTERNAL PICTURE IS picture/NULL
```


Chapter 9: Using CA OLQ Efficiently

To use CA OLQ more efficiently, you can:

- Control data retrieval
- Control resource consumption

This section contains the following topics:

[Controlling Data Retrieval](#) (see page 281)

[Controlling Resource Consumption](#) (see page 290)

Controlling Data Retrieval

Controlling data retrieval involves:

- Using qfiles and logical records to define data retrieval paths
- Using the DML user exit
- Setting the interrupt option and count
- Using a db-key list after a FIND command
- Allowing multiple record retrieval with the SELECT (OLQ access mode) statement

Qfiles

You can predefine data retrieval paths with qfiles, thus defining the information that users are allowed to retrieve. You can further define the information a user is allowed to access by limiting their retrieval to using qfiles only. If you specify that a user can retrieve with qfiles only, it prevents them from using command mode retrieval commands. Unless you specifically prohibit the use of menu mode, the user will be able to retrieve through menu mode.

The syntax required to limit a user's access to CA OLQ to qfiles only is the IDD statement:

```
ADD/MODIFY USER ...  
QFILE IS ONLY
```

If you want a user to access CA OLQ through qfiles only, we recommend that you then disallow access to menu mode. To disallow access to menu mode, use the IDD statement:

```
ADD/MODIFY USER ...  
MENU MODE IS DISALLOWED
```

Logical records

You can predefine navigation paths and logical records for data retrieval with LRF. This allows you to define the records and fields a user retrieves.

Note: For more information about coding logical records and paths in LRF, see the *CA IDMS Logical Record Facility Guide*. IND\$FILE PUT dmlexit bmaster (ASCII RECFM F)

OLQ DML User Exit

OLQ will now invoke a user exit prior to issuing any native DML command. This facility enables users to examine, modify, or disallow navigational access to data from within OLQ. This might be used for security enforcement, statistics gathering, or checking for special data values.

To use this feature, the OLQSDMLE program must be relinked to include IDMSBALI and the user-written exit program. The user-written program must have an entry point of OLQDMLX.

The exit program is called in user mode. The registers on entry are as follows:

- R1 -- points to a parameter list that is the same as that generated for a native DML request. This is documented in the *CA IDMS Programmer's Quick Reference Guide*.
- R13 -- points to a 16-word save area in which the user exit should save the caller's registers.
- R14 -- contains the address to which control should be returned.
- R15 -- points to the OLQDMLX1 entry point within the user exit.

It is the user exit's responsibility to issue the DML command. If it decides that the command should not be executed, it must set the error status field in the IDMS communications block appropriately. The following is a sample program that can be used as a model for writing an OLQ DML exit.

Sample Exit

```

OLQDMLX1 TITLE 'SAMPLE USER-WRITTEN DML EXIT FOR OLQ'
*OLQDMLX1 RENT EP=DMLXEP1 XA
*****
*****
**
** THIS PROGRAM IS A TEMPLATE TO BE USED AS AN EXAMPLE FOR **
** PROVIDING ENTRY INTO AND EXIT FROM AN OLQ USER-WRITTEN **
** DBMS EDIT MODULE FOR RELEASE 14.0 AND LATER. **
**
*****
*****
** THIS IS A SAMPLE ONLY AND NO GUARANTEE IS GIVEN AS TO **
** FUNCTIONALITY, ACCURACY, COMPLETENESS, OR PERFORMANCE. **
**
*****
*****
EJECT
*-----
*
* USERDMLX1 - USER-WRITTEN EXIT FOR DML COMMANDS IN OLQ
*
*-----
*
* OLQDMLX1 allows user-defined editing of DML commands before they
* are issued by OLQ. The edit routine can be used for things such
* as validating security, keeping statistics, looking for special
* data values, etc.
*
* If certain records, DML commands, or AREAs are to be selected
* for editing, an IDMS database procedure should be used.
*
* If many records or many subschemas are to be edited during OLQ
* processing, this exit should be used.
*
* OLQDMLX1 will be automatically called by OLQ before every
* DML command if program OLQSDMLE is LINK/EDITed with this
* module and with IDMSBALI and command 'DCMT VARY PROGRAM
* OLQSDMLE NEW COPY' is issued.
* //SYSLIB DD DISP=SHR,DSN=IDMS.LOADLIB
* //OBJLIB DD DISP=SHR,DSN=USER.LOADLIB
* //SYSLIN DD *
* INCLUDE SYSLIB(OLQSDMLE)
* INCLUDE OBJLIB(OLQDMLX1)
* INCLUDE SYSLIB(IDMSBALI)
* MODE AMODE(31),RMODE(ANY)
* ENTRY ENTRY
* NAME OLQSDMLE(R)

```

```

*
* REGISTER USAGE -
*   R12 - BASE REGISTER
*   R13 -
*   R14 - RETURN ADDRESS FOR SUBROUTINES
*   R15 - A(DB/DC INTERFACE)
*   R0  -
*   R1  - A(PARAMETER LIST) AT ENTRY AND DURING CALLS
*   R2  - A(SEcurity REQUEST BLOCK)
*   R3  -
*   R4  -
*   R5  - WORK REGISTER
*   R6  -
*   R7  -
*   R8  - A(OLQ GLOBAL WORK AREA)
*   R9  -
*   R10 -
*   R11 -
*
*-----
*           EJECT
DMLXSTG DSECT
*-----
*           Any user-required storage is defined here
*-----
WORKAREA DC    CL80' '
DMLXSTGL EQU   *-DMLXSTG
SSCTRLDS DSECT
           @SSCTRL
           EJECT
*-----
*           Entry code is defined here
*-----
#MOPT CSECT=OLQDMLX1,ENV=USER
@MODE MODE=IDMSDC,WORKREG=R0,QUOTES=YES,DEBUG=YES
USING DMLXEP1,R12
ENTRY DMLXEP1
DC    0F'0',CL8'DMLXEP1'
DMLXEP1 DS    0F
STM   R0,R15,0(R13)          SAVE OLQ'S REGISTERS
LA    R13,16*4(,R13)         ADJUST STACK POINTER
LR    R12,R15                ADDRESSIBILITY
#GETSTG TYPE=(USER,SHORT),LEN=DMLXSTGL,ADDR=(R11),          X
      PLIST=*,STGID='USER',INIT=X'00'
USING DMLXSTG,R11
LR    R5,R13                STACK POINTER
SH    R5,=AL2(16*4)         A(OLQ'S REGISTERS)
L     R14,56(,R5)           OLQ'S RETURN REGISTER
CLC   =AL2(28),0(R14)       IF NOT A DML COMMAND

```

```

        BNE DMLXEXIT          CONTINUE DC PROCESSING
        L   R1,4(,R5)         RESTORE A(OLQ'S PARM LIST)
        L   R5,4(,R1)         A(SSCIDBCM+4)
        LA  R5,5(,R5)
        SR  R5,R1             IDBMSCOM SUBSCRIPT
        EJECT

*-----
*       Edit code is defined here
*-----
*-----
*       'Bind Run Unit' edit code is defined here
*-----
DMLX1000 DS   0H
        CH   R5,=H'59'       IF NOT 'BIND RUN UNIT'
        BNE DMLX2000         SEE IF THIS IS 'OBTAIN'
*       Code 'BIND RUN UNIT' pre-processing here
        B   DMLXEXIT          PERFORM THE 'BIND'
        EJECT

*-----
*       DML edit code is defined here
*-----
DMLX2000 DS   0H
        CH   R5,=H'32'       IF NOT 'OBTAIN CALC'
        BNE DMLX3000         SEE IF THIS IS 'FINISH'
        L   R5,8(,R1)         A(RECORD NAME)
        CLC =CL16'EMPLOYEE',0(R5) IF NOT 'OBTAIN CALC EMPLOYEE'
        BNE DMLXEXIT          PERFORM THE 'OBTAIN CALC'
*       Code 'OBTAIN CALC EMPLOYEE' processing here
        L   R5,160(,R8)       A(RECORD IO BUFFER)
        CLC =C'0048',0(R5)    IF EMP-ID-0415 NOT = 0048
        BNE DMLXEXIT          PERFORM THE 'OBTAIN CALC'
ABND2000 L   R5,0(,R1)       A(SSCTRL)
        USING SSCTRLDS,R5
        MVC ERRSTAT,=C'0399'  'OBTAIN CALC EMPLOYEE' is
        MVC ERRORREC,=C'SECURITY ERROR ' not allowed
        B   DMLXRETN          RETURN TO OLQ
        DROP R5
        EJECT

*-----
*       FINISH edit code is defined here
*-----
DMLX3000 DS   0H
        CH   R5,=H'2'       IF NOT 'FINISH RUN UNIT'
        BNE DMLXEXIT          PERFORM DML
*       Code 'FINISH RUN UNIT' processing here
        B   DMLXEXIT          PERFORM THE 'FINISH RUN UNIT'
        EJECT

*-----
*       Exit code is defined here

```

```
*-----  
DMLXEXIT L    R15,=V(IDCSACON)    A(CSA)  
          SH   R13,=H'64'         POINT TO OLQ'S STACK  
          LM   R0,R14,0(R13)      RESTORE OLQ'S REGISTERS  
          BR   R15                 EXECUTE REQUESTED COMMAND  
DMLXRETN SH   R13,=H'64'         POINT TO OLQ'S STACK  
          LM   R0,R15,0(R13)      RESTORE OLQ'S REGISTERS  
          LA   R14,2(,R14)        A(NEXT INSTRUCTION)  
          BR   R14                 RETURN TO OLQ  
          DROP R11  
          LTORG  
          END  DMLXEP1 x
```

Assembly and Link Edit (z/OS)

```

/*-----
/*          ASSEMBLER IEV90 JOB STREAM
/*-----
//ASMSTEP EXEC PGM=IEV90,
//      PARM='ALIGN,XREF,PUNCH,NODECK',
//      REGION=2048K
//SYSLIB      DD DSN=yourHLQ.CAGJMAC,DISP=SHR
//            DD DSN=yourHLQ.CAGJSRC,DISP=SHR
//            DD DSN=os390.maclib,DISP=SHR
//SYSUT1      DD DSN=&.&SYSUT1.,UNIT=VIO,SPACE=(1700,(600,100))
//SYSUT2      DD DSN=&.&SYSUT2.,UNIT=VIO,SPACE=(1700,(600,100))
//SYSUT3      DD DSN=&.&SYSUT3.,UNIT=VIO,SPACE=(1700,(600,100))
//SYSPRINT    DD SYSOUT=*
//SYSPUNCH    DD DSN=&.&OBJECT.,
//              DISP=(NEW,PASS),
//              UNIT=SYSDA,
//              SPACE=(80,(500,1000))
//SYSIN        DD *
OLQ DML Exit program
/*-----
/*          LINK IEWL
/*-----
//LINK      EXEC PGM=IEWL,
//          PARM='LET,LIST,XREF,RENT',
//          REGION=128K,
//          COND=(8,LT,ASMSTEP)
//SYSLMOD    DD DSN=idms.loadlib,DISP=SHR
//SYSPRINT    DD SYSOUT=*
//SYSUT1      DD DSN=&.&SYSUT1.,
//            UNIT=SYSDA,
//            SPACE=(6400,(80)),
//            DISP=(NEW,PASS)
//IN1         DD DSN=yourHLQ.CAGJLOAD,DISP=SHR
//IN2         DD DSN=&.&OBJECT.,DISP=(OLD,DELETE)
//SYSLIN      DD DDNAME=SYSIN
//SYSIN        DD *
INCLUDE IN1(OLQSDMLE)
INCLUDE IN2
INCLUDE IN1(IDMSBALI)
ENTRY ENTRY
MODE AMODE(31),RMODE(ANY)
NAME OLQSDMLE(R)

```

Item	Description
yourHLQ.CAGJLOAD	data set name of the CA IDMS SMP/E distribution load library

Item	Description
idms.loadlib	data set name of the CA IDMS load library
yourHLQ.CAGJMAC	data set name of the CA IDMS macro library
yourHLQ.CAGJSRC	data set name of the CA IDMS source library
os390.maclib	data set name of the z/OS system macro library

Assembly and Link Edit (z/VSE)

```
// DLBL idmslib,  
// EXTENT ,nnnnn  
// LIBDEF *,SEARCH=(idmslib.sublib)  
// LIBDEF PHASE,CATALOG=idmslib.sublib  
// OPTION CATAL  
// EXEC {\cf1\cgrid0 ASMA90},SIZE=128K  
OLQ DML Exit program  
/*  
    INCLUDE OLQSDMLE  
    INCLUDE IDMSBALI  
    ENTRY ENTRY  
// EXEC LNKEDT,SIZE=128K  
/*
```

Item	Description
idmslib	Filename of the file containing CA IDMS modules
idmslib.sublib	Name of the sublibrary within the library containing CA IDMS modules
Nnnnnn	Volume serial identifier of appropriate disk volume

Interrupt count

What is the interrupt count?

The interrupt count is the number of CA OLQ database requests allowed before execution is interrupted. When interruption occurs, you can choose to resume execution or to halt execution. This interruption allows you to monitor processing during retrieval.

When processing is interrupted, CA OLQ displays this message:

```
OLQ 098006 00 50 whole lines and 0 partial lines in report.  
OLQ 098007 00 50 records read. 50 records selected.  
OLQ 098009 00 Continue (yes/no)?
```


Primary function:

The primary use for interrupt count is to specify the number of CA OLQ database requests after which CA OLQ interrupts the processing. This enables you to limit CA OLQ database requests.

Additional use:

Another use for interrupt count is to determine the number of rows CA OLQ saves in a table before it performs a commit. (When CA OLQ performs a commit, database locks are released.)

Note: Keep in mind that, when any problems occur with saving data, CA OLQ rolls back to the last commit. You could end up with a partial table. (This applies to both SQL and ASF tables.)

Setting interrupt:

You initially set the interrupt count during system generation with the ADD/MODIFY OLQ statement:

```
INTERRUPT COUNT IS nnn  
MAXIMUM INTERRUPT COUNT IS nnn
```

These statements set the default interrupt count and the default maximum interrupt for CA OLQ. In this way, you can limit the number of database calls performed on each retrieval request. The system default set at installation is 100. You can, however, make this number as large or as small as you wish. To keep CA OLQ from performing commits while saving tables, set the interrupt count to zero.

A user can retrieve records without specifying an interrupt count; the interrupt count set during system generation acts as the default. If a user uses the SET INTERRUPT COUNT (in CA OLQ) to specify a count that is higher than the maximum set during system generation, the count defaults to the count set with the MAXIMUM INTERRUPT COUNT IS statement during system generation. To prevent CA OLQ from performing commits, set the interrupt count to zero. Note that even when the interrupt option is off, the interrupt count is still set.

Using db-keys for retrieval

CA OLQ can keep db-key lists to facilitate database retrieval following a FIND command. Since keeping a list of db-keys can use additional resources, it should be done only if **OPTIONS=DBKEY** has been specified during the CA OLQ session. The default option setting is **OPTIONS=NODBKEY**.

Controlling Resource Consumption

Controlling resource consumption involves:

- Allowing sorts
- Saving reports (size and retention period)
- Allowing use of db-key lists
- Allowing SAVE QFILES

Sorts

Sorting retrieved records is a resource-consuming operation. You can limit the ability to sort records to those users who need it.

You can control use of resources by controlling the use of sorts. There are two methods of sorting in CA OLQ:

- Using SORT after a GET/FIND command
- Using the ORDER BY clause of SELECT (OLQ access mode)

Using SORT after building a report with a GET/FIND command is costly. The ORDER BY clause of SELECT is more efficient because SELECT sorts the data while retrieving it.

When using SELECT (OLQ access mode), you can choose to disallow sorts altogether (this also disallows sorting with the ORDER BY clause of SELECT). To disallow sorting, use the IDD statement:

```
ADD/MODIFY USER ...  
SORT IS NOT ALLOWED
```

Access mode: The SORT IS NOT ALLOWED clause is **only valid** when the access switch is set to **OLQ**.

Sort and the scratch area

CA OLQ sorts records in memory. The maximum amount of memory CA OLQ uses for any particular sort is determined by the SET MAXIMUM SORT SIZE system generation option. When sorting more records than fit into the maximum allowed, CA OLQ continues the sort by paging records in and out of scratch.

Specifying a large maximum sort size decreases the time a single sort takes and impacts other DC system users by:

- Using more storage
- Not releasing control to DC until the sort has completed

Note: If you want to use your own operating system sort facility when running a batch job, set up the proper job control language as described in [num=11.Batch Processing](#) (see page 303).

For more information:

[Batch Processing](#) (see page 303)

Saved reports

You can set default values to keep a limit on the size and retention period of saved reports. You can set default values for specifications users are allowed to make when saving reports:

- Report retention period— The amount of time a report remains saved if the user doesn't specify a retention period or specifies one less than the maximum. When the retention period expires, the report is automatically deleted.
- Maximum retention period— The maximum amount of time a report remains saved if the user specifies a retention period greater than the default retention period mentioned above.
- Report dictionary name— The name of the dictionary in which the catalog information is stored. CA OLQ assigns a catalog entry and passkeys to each saved report.
- Report size— The page size and number of pages allowed per report.
- Maximum number of reports — The maximum number of reports allowed per user. If the maximum is exceeded, a user must delete some reports in order to store new ones.

Report retention period:

If a user saves a report without specifying how many days it is to remain saved, the report is saved for the number of days specified by the report retention. This default is set during system generation with:

```
ADD/MODIFY OLQ ...  
REPORT RETENTION IS retention-period
```

Keep in mind when using the batch facility, reports saved through batch do not honor the defaults set during system generation. You must specify these options explicitly when saving a batch report.

Maximum report retention:

You can define the maximum report retention (the maximum length of time in days) that a user can specify when saving a CA OLQ report in the queue area. If the user specifies a retention period greater than the default report retention period, the retention period defaults to this maximum. Use the system generation statement:

```
ADD/MODIFY OLQ ...  
MAXIMUM REPORT RETENTION IS retention-period
```

Report dictionary name:

Since CA OLQ stores reports through the catalog, each saved report has a catalog entry and passkeys assigned. You can, during system generation, specify in which dictionary this catalog information is stored with:

```
ADD/MODIFY OLQ ...  
REPORT DICTNAME IS dictionary-name
```

The specified report dictionary is also the dictionary in which CA OLQ looks to find the job control language needed for batch job submission.

Report size:

You can better estimate the size of the scratch and queue areas if you limit the size of CA OLQ reports retrieved.

To monitor the size of CA OLQ reports retrieved into scratch and saved into queue, you can specify the **page size**, the **number of pages** retrieved from the database, and the **number of reports** allowed per user.

Define the **report page size** during system generation with:

```
ADD/MODIFY OLQ ...  
REPORT FILE PAGE SIZE IS report-file-page-size
```

The specified page size must be at least large enough to accommodate the largest database or logical record to be included in the report.

When a CA OLQ report exceeds the maximum report pages, as set with this option, system ends the retrieval process and stops writing report pages and issues the message:

```
OLQ 097004 00 Maximum report size has been exceeded
```

The incomplete report is retrieved into scratch, is available for viewing, and can be sorted.

Define the **number of pages** allowed per report during system generation with:

```
ADD/MODIFY OLQ ...  
MAXIMUM REPORT PAGES IS max-report-pages
```

Define the **number of reports** allowed per user during system generation with:

```
ADD/MODIFY OLQ ...  
MAXIMUM REPORT COUNT IS max-report-count
```

This way you can control the quantities of reports saved and better estimate the size of the queue area. Use the SELECT (OLQ access mode) statement to retrieve the records. With SELECT (OLQ access mode) you can sort and group the records while retrieving them.

For more information:

[Batch Processing](#) (see page 303)

Db-key list

You can control resources by limiting the use of db-key lists. Whenever the db-key option in CA OLQ is **ON**, CA OLQ builds a db-key list during retrieval. This means that for every record retrieved, CA OLQ stores a db-key in a list. This can be extremely costly in terms of resources if you perform large retrievals.

We recommend that you set the db-key option in CA OLQ **OFF**. If a user needs to build a db-key list for retrieval purposes, that user can turn the option back on for the time required.

Saving qfiles

In order to control resource consumption, you can limit the ability to save qfiles. This prevents users from creating redundant qfiles and also from cluttering up the dictionary with unnecessary qfiles.

To disallow the saving of qfiles, use the `IDD DDDL` statement:

```
ADD/MODIFY USER ...  
QFILE SAVE IS NOT ALLOWED
```


Chapter 10: Security

In this chapter:

CA OLQ provides security options that enable you to govern the accessibility of the database. The security topics covered in this chapter are:

- Assigning authority to access CA OLQ
- Initiating security for CA OLQ
- Securing user access to CA OLQ
 - Accessing subschemas
 - Accessing qfiles
 - Securing retrieval interruption
- Using LRF to establish security
- Security for ASF tables
- Security for saved reports

When a security option is changed, CA OLQ enforces the change the next time the user signs on. CA OLQ uses the DC/UCF signon ID and password to determine what the current user can access.

Assigning Authority to Access CA OLQ

Access to CA OLQ can be limited by defining options in:

- Central Security—Define the category to which task OLQ is associated
- Dictionary Security—Define individual users that can access OLQ and valid functions can be defined uniquely for each user in each dictionary.

Limiting Access through Central Security

The centralized security administrator can assign the task OLQ to a category. Only users granted execution privilege in the category can access CA OLQ.

Note: For more information on category and security, see your security administrator.

Initiating CA OLQ Dictionary Security

You can turn security for CA OLQ on and off with the `IDD SET OPTIONS` statement in the default dictionary. The full statement for initiating security is:

```
SET OPTIONS FOR DICT SECURITY FOR OLQ IS ON
```

This will allow only those `USERS` who are defined in the default dictionary to access OLQ.

Note: For the complete syntax and syntax rules for `IDD DDDL` statements, see the *CA IDMS IDD Quick Reference Guide*.

Securing User Access to CA OLQ Components

You can specify additional CA OLQ security options for users in each dictionary through `IDD` with the statement:

```
SET OPTIONS FOR DICT SECURITY FOR OLQ IS ON
```

```
ADD/MODIFY/DELETE USER
```

Note: For the entire syntax of this command, see the *CA IDMS IDD Quick Reference Guide*.

The options associated with this statement are:

- `AUTHORITY FOR UPDATE IS OLQ`
- Access to subschemas
- Access to qfiles
- Access to menu mode
- Securing retrieval interruption
- Allow retrieval of multiple record occurrences
- Allow access to `SORT`

Subschema Access

CA OLQ uses the `DC/UCF` signon ID and password to determine which subschemas are available to the current user. You can assign individual users authority to access specific subschemas with the following clause of the `IDD ADD USER` statement:

```
ACCESS TO SUBSCHEMA subschema-name
```

This clause specifies that the user has access to the named subschema. Subschema security is enforced on a dictionary-by-dictionary basis.

It is with this statement that you also assign a **signon qfile** to a user. The signon qfile executes when a user signs on to the associated subschema. The clause of the ADD USER statement that associates a signon qfile is:

```
SIGNON QFILE IS qfile-name
```

Note: For more information about signon qfiles, see [Special Uses of qfiles](#) (see page 270).

For the complete syntax and syntax rules of the ADD USER statement, see the *CA IDMS IDD Quick Reference Guide*.

Qfile Access

You can apply security to qfiles in CA OLQ. By specifying restrictions on CA OLQ qfiles in IDD, you can:

- Allow/disallow a specific user to execute qfiles
- Allow/disallow a specific user to save qfiles
- Limit a user to accessing CA OLQ through qfiles only

These restrictions are defined in IDD, but take effect only if CA OLQ security is turned on in the primary dictionary, as described earlier in this chapter.

Sharing qfiles:

In CA OLQ, only the user who creates a qfile has access to that qfile. You can allow users to share qfiles through IDD. It is more efficient to share qfiles than to have duplicate qfiles for many users. Use the following statement to assign a user access to a particular qfile:

```
ADD/MODIFY/REPLACE QFILE ...
  INCLUDE USER user-name ...
  REGISTERED FOR PUBLIC ACCESS ...
  PUBLIC ACCESS IS ALL
```

Note: For more information about this statement, see the *CA IDMS IDD Quick Reference Guide*.

Executing qfiles:

You can allow and disallow qfile execution on a user-by-user basis through IDD with the statement:

```
ADD/MODIFY USER ...
  QFILE IS ALLOWED/NOT ALLOWED
```

Saving qfiles:

You can allow and disallow saving qfiles on a user-by-user basis through IDD with the statement:

```
ADD/MODIFY USER ...  
QFILE SAVE IS ALLOWED/NOT ALLOWED
```

Accessing through qfiles only:

You can limit a user's access to CA OLQ to only executing qfiles with the IDD statement:

```
ADD/MODIFY USER ...  
QFILE IS ONLY
```

Accessing menu mode:

If your intention is to allow a user to only access CA OLQ through qfiles, you must explicitly deny that user access to CA OLQ menu mode. You disallow access to menu mode through IDD:

```
ADD/MODIFY USER ...  
MENU MODE IS NOT ALLOWED
```

Securing Retrieval Interruption

You can use IDD to secure interruption of processing in CA OLQ. IDD enables you to specify whether a user can choose **not** to interrupt data retrieval, or whether data retrieval is automatically interrupted after a certain number of records have been accessed.

You set this security in IDD, but it takes effect only if CA OLQ security is turned on in the primary dictionary, as described earlier in this chapter.

During system generation, you can specify a maximum interrupt count. This maximum provides a ceiling for the user-specified interrupt count.

Note: For instructions on how to set these options, see [num=12.Setting Defaults](#) (see page 345).

For more information:

[Setting Defaults](#) (see page 345)

Using LRF to Secure The Database

You can use LRF to establish security for the database. By coding paths and logical records, you can dictate what a user is allowed to access through CA OLQ.

LRF enables you to:

- **Increase data integrity**— You can write all database navigational instructions in the subschema. This enables you to predefine paths that dictate how a user accesses data through CA OLQ.
- **Secure data**
 - You can restrict the database record occurrences and fields viewed by the application program.
 - You can restrict the operations that the application program can perform on records and fields.

Security for ASF tables

CA OLQ secures ASF tables regardless of the status of CA OLQ security. To ensure table security, we recommend the following:

- IDs use the same passwords across all dictionaries
- The IDMSNWKQ and IDMSNWKS subschemas use the same page ranges

Passkeys:

CA OLQ has automatic security for ASF tables. When a user tries to access a table, CA OLQ uses catalog security to make sure the user has proper authority assigned through passkeys.

The table functions and the passkeys required by the Automatic System Facility (ASF) are presented in **Table 10**. Passkeys affect ASF tables and reports.

Table 10. How Passkeys affect CA OLQ Table Processing

To Perform	User Default Passkey Required	Data Access Passkey Required
Select	COPY	COPY
Create	CREATE	N/A
Append	ADD	ADD
Replace	ADD	ADD

To Perform	User Default Passkey Required	Data Access Passkey Required
Delete	ERASE	ERASE
Show tables	BROWSE	N/A
Show directory	BROWSE	N/A

A **user default passkey** is a passkey given to a user that allows the user to perform a specific function on any ASF table.

A **data access passkey** is a passkey associated with a specific table. It allows a specific user to perform a function on a specific table.

In addition to supporting passkey security, OLQ supports row-level security.

Note: For more information about row-level security, see the *CA IDMS Performance Monitor System Administration Guide*.

Security for Saved Reports

Saving a report:

When you save a report in CA OLQ, information about that report is stored in the CA IDMS catalog. Passkeys are automatically assigned to the user for the report. If you want others to be able to access the report, you must go into CA IDMS and assign those users the appropriate passkeys. The appropriate passkeys and associated functions are presented in [Table 10](#) (see page 301).

Chapter 11: Batch Processing

What batch does:

The CA OLQ batch facility allows you to perform the same functions as command mode CA OLQ. Batch enables you to retrieve information from a CA IDMS/DB database and sequential files, and to build reports using that information. You can run the jobs offline and schedule them at times of the day when there are less demands on computer resources.

Commands you can use in batch:

CA OLQ batch supports the same set of processing commands as command mode CA OLQ except for the subset presented in **Table 11**.

Table 11. Commands Not Supported in CA OLQ Batch

Environment	Invalid Commands
Central version (CV CA OLQ batch)	SHOW TABLES SHOW DESTINATIONS SUSPEND SWAP SWITCH YES/NO
Local mode (Local CA OLQ batch)	SAVE REPORT DISPLAY REPORT DELETE REPORT SHOW DIRECTORY SWAP SWITCH SUSPEND YES/NO

How batch works:

With CA OLQ batch, you can:

- Read input commands one by one from a **command file**.
- Retrieve information from one or more **input files** (sequential files or databases).
- Build a report in the **output file**. This file can contain the data retrieved by the SELECT statement (OLQ mode).

Creating a command file:

A **command file** is a file that contains the CA OLQ commands to be executed during the batch job. You can store a command file in a sequential data set or list the commands in the job control language (JCL).

In batch, a line is the delimiter for entering commands. If a single command spans two or more lines, you must include continuation characters at the end of the continued lines to indicate to CA OLQ that the command continues on the next line. The default continuation character at system installation is the hyphen (-).

You must include a **SYSIPT** statement in your JCL. The SYSIPT statement defines the data set name of the file that contains the CA OLQ commands to be executed.

Executing jobs:

You can execute CA OLQ batch jobs either under the CA IDMS/DB **central version** or in **local mode**.

- **CV batch job streams** execute in the batch region and use the CA IDMS/DB central version for database access. To specify that your batch job is running under the central version, include a SYSCTL card in your z/OS JCL, or an IDMSOPTI card in your z/VSE and CMS JCL.
- **Local mode batch job streams** execute and handle database access within the batch region. Running a local mode batch job requires additional file assignments in your JCL as described later in this chapter.

How to run a batch job:

There are two main steps to using the CA OLQ Batch facility:

1. Run IDMSBCF—IDMSBCF sets up the batch scratch area. This must be run before the CA OLQ local mode batch job step.
2. Run OLQBATCH— The JCL for this job step is presented below.

Note: You can include the IDMSBCF statement in the JCL, but it must be run prior to the OLQBATCH job step.

To set up batch processing in CA OLQ, you:

- Tailor the job control language supplied in this chapter to your environment
- Set the batch class for batch job control language submission
- Set default options normally set during system generation

For more information:

[Commands and Syntax](#) (see page 53)

JCL for z/OS and CMS Commands for OLQBATCH

The sections that follow contain sample JCL for the following operating systems running central version or local mode:

- z/OS
- CMS
- z/VSE

z/OS JCL (central version)

Here is sample z/OS JCL to execute CA OLQ batch jobs, when running central version:

CA OLQ Batch (central version) (z/OS)

```
//FORMAT EXEC PGM=IDMSBCF,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadLib,DISP=SHR
// DD DSN=idms.loadLib,DISP=SHR
//dcmsg DD DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//dclscr DD DSN=&.dclscr.,DISP=(NEW,PASS),
// UNIT=SYSDA,SPACE=(bbbb,nnnn),
// DCB=(RECFM=F,LRECL=1111,BLKSIZE=bbbb)
//SYSLST DD SYSOUT=A
//SYSIDMS DD *
DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
//SYSIPT DD *
FORMAT FILE SYSLOC.dclscr;
/*
/*
//OLQBATCH EXEC PGM=OLQBATCH,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadLib,DISP=SHR
// DD DSN=idms.loadLib,DISP=SHR
//sysctl DD DSN=idms.sysctl,DISP=SHR
//dcmsg DD DSN=idms.sysmsg.dcmsg,DISP=SHR
//dclscr DD DSN=&.dclscr.,DISP=(OLD,DELETE)
//SORTWK01 DD UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK02 DD UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK03 DD UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK04 DD UNIT=disk,SPACE=(CYL,(5,2))
//SYSOUT DD SYSOUT=A
//SYSLST DD SYSOUT=A
//SYSIDMS DD *
DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
//SYSIPT DD *
Put CA OLQ commands here
/*
/*
```

Note: The SORTWK01 through SYSOUT statements are optional. If these statements are included, CA OLQ uses the installation's sort utility. When these are omitted, CA OLQ uses its own internal sort facility.

z/OS JCL (local mode)

Here is sample z/OS JCL to execute CA OLQ batch jobs in local mode:

CA OLQ Batch (local mode) (z/OS)

```
//FORMAT EXEC PGM=IDMSBCF,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
//dcmsg DD DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//dclscr DD DSN=&.dclscr.,DISP=(NEW,PASS),
// UNIT=SYSDA,SPACE=(bbbb,nnnn),
// DCB=(RECFM=F,LRECL=1111,BLKSIZE=bbbb)
//SYSLST DD SYSOUT=A
//SYSIDMS DD *
DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
//SYSIPT DD *
FORMAT FILE SYSLOC.dclscr;
/*
/*
//OLQBATCH EXEC PGM=OLQBATCH,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
//dictdb DD DSN=idms.appldict.ddldml,DISP=SHR
//dloddb DD DSN=idms.appldict.ddldclod,DISP=SHR
//dcmsg DD DSN=idms.sysmsg.dcmmsg,DISP=SHR
//userdb DD DSN=user.userdb,DISP=SHR
//dclscr DD DSN=&.dclscr.,DISP=(OLD,DELETE)
//sysjrn1 DD DUMMY
//SYSLST DD SYSOUT=A
//SYSIDMS DD *
DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
//SYSIPT DD *
Put CA OLQ commands here
/*
/*
```

idms.dba.loadlib Data set name of the load library containing the DMCL and database name table load modules

idms.loadlib Data set name of the load library containing the CA IDMS executable modules

dcmsg DDname of the system message (DDLDCMSG) area

<i>idms.sysmsg.ddldcmsg</i>	Data set name of the system message (DDLDCMSG) area
<i>dclscr</i>	DDname of the local mode system scratch (DDLOCSCR) area Note: See z/OS Local Mode Considerations (see page 309), below, for more information.
<i>&.dclscr.</i>	Temporary data set name of the local mode system scratch (DDLOCSCR) area Note: See z/OS Local Mode Considerations (see page 309), below, for more information.
<i>bbbb</i>	Block size (page size)
<i>nnnn</i>	Number of pages in the area
<i>llll</i>	Logical record length (should be the same as block size)
<i>dmcl-name</i>	Specifies the name of the DMCL load module
<i>dictdb</i>	DDname of the application dictionary definition area
<i>idms.appldict.ddldml</i>	Data set name of the application dictionary definition (DDLML) area
<i>dloddb</i>	DDname of the application dictionary definition load area
<i>idms.appldict.ddldclod</i>	Data set name of the application dictionary definition load (DDLCLOD) area
<i>userdb</i>	DDname of the user database file
<i>user.userdb</i>	Data set name of the user database file
<i>sysjnl</i>	DDname of the tape journal file Note: See z/OS Local Mode Considerations (see page 309), below, for more information.
<i>sysctl</i>	DDname of the SYSCTL file
<i>idms.sysctl</i>	Data set name of the SYSCTL file

Note: If both the system and local scratch areas are defined in the DMCL, CA IDMS only uses the *local* scratch area (SYSLOC.DDLOCSCR, *dclscr*).

z/OS Local Mode Considerations

Scratch areas:

If both scratch areas are defined in the DMCL, CA IDMS only uses the local scratch area (SYSLOC.DDLOCSCR, *dclscr*). When running CA OLQ Batch with a DMCL that only defines the *system* scratch area (SYSTEM.DDLSCR), you must format this area with the following statement:

```
FORMAT FILE SYSTEM.dclscr
```

Accessing SQL Tables:

When accessing SQL tables, you must add the following statements:

```
//sqldd      DD  DSN=idms.syssql.ddlcat,DISP=SHR
//sqlxdd     DD  DSN=idms.syssql.ddlcatx,DISP=SHR
```

<i>sqldd</i>	DDname of the SQL catalog (DDLCAT) area
<i>idms.syssql.ddlcat</i>	Data set name of the SQL catalog (DDLCAT) area
<i>sqlxdd</i>	DDname of the SQL catalog index (DDLCATX) area
<i>idms.syssql.ddlcatx</i>	Data set name of the SQL catalog index (DDLCATX) area

When Creating Tables:

When using CA OLQ Batch to create tables, change:

```
//sysjrnl DD DUMMY
```

To:

```
//sysjrnl DD DSN=idms.tapejrnl,DISP=(NEW,CATLG),UNIT=tape
```

Where:

<i>sysjrnl</i>	DDname of the tape journal file
----------------	---------------------------------

idms.tapejrn1 Data set name of the tape journal file

tape Symbolic device name for tape file

CMS commands (central version)

Here are sample CMS commands to execute CA OLQ batch jobs, when running central version:

CA OLQ Batch (central version) (CMS)

```
FILEDEF DCLSCR DISK scratch file a (RECFM F LRECL 111 BLKSIZE bbb
FILEDEF SYSIDMS DISK sysidms pams a (RECFM F LRECL 111 BLKSIZE bbb
FILEDEF SYSIPT DISK sysipt input a (RECFM F LRECL 111 BLKSIZE bbb
EXEC IDMSFD
OSRUN IDMSBCF
```

```
FILEDEF userdb DISK user userdb a (RECFM FB LRECL ppp BLKSIZE ppp
FILEDEF SYSIDMS DISK sysidms pams a (RECFM F LRECL 111 BLKSIZE bbb
FILEDEF SYSIPT DISK sysipt input a (RECFM F LRECL 111 BLKSIZE bbb
OSRUN OLQBATCH
```

scratch file a FileID of the temporary scratch file for the local mode system scratch (DDLOCSCR) area

111 Logical record length of the user input data file

bbb Block size of the user input data file

sysidms pams a FileID of the file containing SYSIDMS parameters

sysipt input a FileID of the file containing the IDMSBCF or OLQBATCH input parameters

IDMSFD Exec which defines all FILEDEFS, TXTLIBs, and LOADLIBs required by the system

userdb DDname of the user database file

user userdb a FileID of the user database file

Page size of the user database file

ppp

Usage

SYSIDMS file:

To run IDMSBCF or OLQBATCH, you should include these SYSIDMS parameters:

- `DMCL=dmcl-name`, to identify the DMCL
- If you are running IDMSBCF or OLQBATCH against an SQL-defined database `DBNAME=dictionary-name`, to identify the dictionary whose catalog component contains the database definitions

How to create the SYSIDMS file:

To create the SYSIDMS file of SYSIDMS parameters:

1. On the CMS command line, type:
`XEDIT sysidms parms a (NOPROF`
2. Press [Enter]
3. On the XEDIT command line, type:
`INPUT`
4. Press [Enter]
5. In input mode, type in the SYSIDMS parameters
6. Press [Enter] to exit input mode
7. On the XEDIT command line, type:
`FILE`
8. Press [Enter]

Note: For documentation of SYSIDMS parameters, see the *CA IDMS Database Administration Guide*.

How to create the SYSIPT file:

To create the SYSIPT file of IDMSBCF and OLQBATCH input parameters:

1. On the CMS command line, type:
`XEDIT sysipt input a (NOPROF`
2. Press [Enter]
3. On the XEDIT command line, type:
`INPUT`
4. Press [Enter]
5. In input mode, type in the IDMSBCF or OLQBATCH input parameters
6. Press [Enter] to exit input mode
7. On the XEDIT command line, type:
`FILE`
8. Press [Enter]

DCLSCR scratch file:

DCLSCR is a scratch file, which resides on a temporary mini disk. To allocate this disk, enter the following commands:

```
DEF t3380 cuu CYL nnn  
FORMAT cuu fm
```

	Disk device type
<i>t3380</i>	
	Virtual address of the temporary minidisk
<i>cuu</i>	
	Space allocated in CYLS
<i>nnn</i>	
	Filemode for the temporary minidisk
<i>fm</i>	

CMS commands (local mode)

To specify that OLQBATCH is executing in local mode, perform one of the following:

- Link OLQBATCH with an IDMSOPTI program that specifies local execution mode
- Specify *LOCAL* as the first input parameter of the filename, type and mode identified by *sysipt2 input a* in the OLQBATCH exec.
- Modify the OSRUN statement:

```
OSRUN OLQBATCH PARM='*LOCAL*'
```

Note: This option is valid only if the OSRUN command is issued from a System Product interpreter or an EXEC2 file.

z/VSE JCL (central version)

Here is sample z/VSE JCL to execute CA OLQ batch jobs, when running central version:

CA OLQ Batch (central version) (z/VSE)

```
// JOB    OLQBATCH
// DLBL   idmslib,'idmslib.library'
// EXTENT sysxxx,vvvvv,,sss,ttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA IDMS load libraries
// EXEC   PROC=IDMSLBLS
// EXEC   IDMSBCF,SIZE=1048K
//        DMCL=dmc1-name
//        Put other SYSIDMS parameters, as appropriate, here
/*
//        FORMAT FILE SYSLOC.dclscr;
/*
// EXEC   PROC=IDMSLBLS
// EXEC   PROC=sysctl
// EXEC   OLQBATCH,SIZE=1048K
//        DMCL=dmc1-name
//        Put other SYSIDMS parameters, as appropriate, here
/*
//        Put CA OLQ commands here
/*
```

Dtfname of the CA IDMS library

idmslib

Data set name of CA IDMS load libraries, as established during installation

'idmslib.library'

<i>sysxxx</i>	SYS number
<i>vvvvvv</i>	Volume serial number
<i>ssss</i>	Starting extent
<i>tttt</i>	Number of tracts
<i>CA IDMS load libraries</i>	The CA IDMS load libraries, as established during installation
<i>dmcl-name</i>	Name of the DMCL
<i>dclscr</i>	Dtfname of the local modesystem scratch (DDLOCSCR) area
<i>sysctl</i>	Dtfname of the SYSTCL file

Note: For more information about the IDMSLBLS procedure, see [IDMSLBLS Procedure](#) (see page 317), later in this chapter.

z/VSE JCL (local mode)

Here is sample z/VSE JCL to execute CA OLQ batch jobs in local mode:

CA OLQ Batch (local mode) (z/VSE)

```
// JOB    OLQLOCAL
// DLBL   idmslib,'idmslib.library'
// EXTENT sysxxx,vvvvv,,sss,ttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA IDMS load libraries
// EXEC   PROC=IDMSLBLS
// EXEC   IDMSBCF,SIZE=1024K
        DMCL=dmcl-name
        Put other SYSIDMS parameters, as appropriate, here
/*
    FORMAT FILE SYSLOC.dclscr;
/*
// EXEC   PROC=IDMSLBLS
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   userdb.'user.userdb'
// EXTENT sysxxx,vvvvv,,tttt,llll
// EXEC   OLQBATCH,SIZE=1024K
        DMCL=dmcl-name
        Put other SYSIDMS parameters, as appropriate, here
/*
    Put CA OLQ commands here
/*
```

<i>idmslib</i>	Dtfname of the CA IDMS library
<i>'idmslib.library'</i>	Data set name of CA IDMS load libraries, as established during installation
<i>sysxxx</i>	SYS number
<i>vvvvvv</i>	Volume serial number
<i>sss</i>	Starting extent
<i>ttt</i>	Number of tracts
<i>CA IDMS load libraries</i>	The CA IDMS load libraries, as established during installation

<i>dclscr</i>	Dtfname of the local modesystem scratch (DDLOCSCR) area
<i>dmcl-name</i>	Name of the DMCL
<i>dclscr</i>	Dtfname of the local modesystem scratch (DDLOCSCR) area
<i>userdb</i>	Dtfname of the user database file
<i>user.userdb</i>	Data set name of the user database file
<i>////</i>	

Note: For more information about the IDMSLBLS procedure, see [IDMSLBLS Procedure](#) (see page 317), later in this chapter.

IDMSLBS Procedure

What is the IDMSLBS procedure?

IDMSLBS is a procedure provided during a CA IDMS z/VSE installation. It contains file definitions for the CA IDMS components listed below. These components are provided during installation:

- Dictionaries
- Sample databases
- Disk journal files
- SYSIDMS file

Tailor the IDMSLBS procedure to reflect the filenames and definitions in use at your site and include this procedure in z/VSE JCL job streams.

The sample z/VSE JCL provided in this document includes the IDMSLBS procedure. Therefore, individual file definitions for CA IDMS dictionaries, sample databases, disk journal files, and SYSIDMS file are not included in the sample JCL.

IDMSLBS procedure listing

```
/* _____ LABELS _____  
// DLBL    dccat, 'idms.system.dccat', 1999/365, DA
```

```

// EXTENT SYSnnn,nnnnnn,,ssss,31
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dccatl,'idms.system.dccatlod',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dccatx,'idms.system.dccatx',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,11
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dcdml,'idms.system.ddlml',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,101
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dclod,'idms.system.ddlclod',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,21
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dclog,'idms.system.ddlclod',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,401
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dcrun,'idms.system.ddlcrun',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,68
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dcscr,'idms.system.ddlccscr',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,135
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dmsg,'idms.sysmsg.ddlcmmsg',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,201
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dclscr,'idms.sysloc.ddlccscr',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dirldb,'idms.sysdirl.ddlml',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,201
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dirllod,'idms.sysdirl.ddlclod',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,2
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL empdemo,'idms.empdemo1',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,11
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL insdemo,'idms.insdemo1',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL orgdemo,'idms.orgdemo1',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL empldem,'idms.sqldemo.empldemo',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,11
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL infodem,'idms.sqldemo.infodemo',1999/365,DA

```

```

// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL projdem,'idms.projseg.projdemo',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL indxdem,'idms.sqldemo.indxdemo',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL sysctl,'idms.sysctl',1999/365,SD
// EXTENT SYSnnn,nnnnnn,,ssss,2
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL secdd,'idms.sysuser.dlsec',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,26
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dictdb,'idms.appldict.dldml',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,51
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dloddb,'idms.appldict.dldclod',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,51
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL sqldd,'idms.syssql.dlcat',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,101
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL sqllod,'idms.syssql.dlcatl',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,51
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL sqlxdd,'idms.syssql.dlcatx',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,26
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL asfdml,'idms.asfdict.dldml',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,201
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL asflod,'idms.asfdict.asflod',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,401
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL asfdata,'idms.asfdict.asfdata',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,201
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL ASFDEFN,'idms.asfdict.asfdefn',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,101
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL j1jrnL,'idms.j1jrnL',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,54
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL j2jrnL,'idms.j2jrnL',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,54
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL j3jrnL,'idms.j3jrnL',1999/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,54

```

```
// ASSGN  SYSnnn ,DISK,VOL=nnnnnn ,SHR
// DLBL   SYSIDMS, '#SYSIPT' ,0,SD
/+
/*
```

	Filename of the system dictionary catalog (DDLDCAT) area
<i>dccat</i>	
	File-ID of the system dictionary catalog (DDLDCAT) area
<i>idms.system.dccat</i>	
	Logical unit of the volume for which the extent is effective
<i>SYSnnn</i>	
	Volume serial identifier of appropriate disk volume
<i>nnnnnn</i>	
	Starting track (CKD) or block (FBA) of disk extent
<i>ssss</i>	
	Filename of the system dictionary catalog load (DDLDCATLOD) area
<i>dccatl</i>	
	File-ID of the system dictionary catalog load (DDLDCATLOD) area
<i>idms.system.dccatlod</i>	
	Filename of the system dictionary catalog index (DDLDCATX) area
<i>dccatx</i>	
	File-ID of the system dictionary catalog index (DDLDCATX) area
<i>idms.system.dccatx</i>	
	Filename of the system dictionary definition (DDLDCML) area
<i>dcdml</i>	
	File-ID of the system dictionary definition (DDLDCML) area
<i>idms.system.ddldml</i>	
	Filename of the system dictionary definition load (DDLDCLOD) area
<i>dclod</i>	
	File-ID of the system dictionary definition load (DDLDCLOD) area
<i>idms.system.ddldclod</i>	
	Filename of the system log area (DDLDCLOG) area
<i>dclog</i>	
	File-ID of the system log (DDLDCLOG) area
<i>idms.system.ddldclog</i>	

<i>dcrun</i>	Filename of the system queue (DDLDCRUN) area
<i>idms.system.ddldcrun</i>	File-ID of the system queue (DDLDCRUN) area
<i>dcscr</i>	Filename of the system scratch (DDLDCSCR) area
<i>idms.system.ddldcscr</i>	File-ID of the system scratch (DDLDCSCR) area
<i>dcmsg</i>	Filename of the system message (DDLDCMSG) area
<i>idms.sysmsg.ddldcmsg</i>	File-ID of the system message (DDLDCMSG) area
<i>dclscr</i>	Filename of the local mode system scratch (DDLOCSCR) area
<i>idms.sysloc.ddlocscr</i>	File-ID of the local mode system scratch (DDLOCSCR) area
<i>dirldb</i>	Filename of the IDMSDIRL definition (DDLDMML) area
<i>idms.sysdirl.ddldml</i>	File-ID of the IDMSDIRL definition (DDLDMML) area
<i>dirllod</i>	Filename of the IDMSDIRL definition load (DDLDCLOUD) area
<i>idms.sysdirl.dirllod</i>	File-ID of the IDMSDIRL definition load (DDLDCLOUD) area
<i>empdemo</i>	Filename of the EMPDEMO area
<i>idms.empdemo1</i>	File-ID of the EMPDEMO area
<i>insdemo</i>	Filename of the INSDEMO area
<i>idms.insdemo1</i>	File-ID of the INSDEMO area
<i>orgdemo</i>	Filename of the ORGDemo area

<i>idms.orgdemo1</i>	File-ID of the ORDDemo area
<i>empldem</i>	Filename of the EMPLDemo area
<i>idms.sqldemo.empldemo</i>	File-ID of the EMPLDemo area
<i>o</i>	
<i>infodem</i>	Filename of the INFODemo area
<i>idms.sqldemo.infodemo</i>	File-ID of the INFODemo area
<i>projdem</i>	Filename of the PROJDemo area
<i>idms.projseg.projdemo</i>	File-ID of the PROJDemo area
<i>indxdem</i>	Filename of the INDXDemo area
<i>idms.sqldemo.indxdemo</i>	File-ID of the INDXDemo area
<i>sysctl</i>	Filename of the SYSCTL file
<i>idms.sysctl</i>	File-ID of the SYSCTL file
<i>secdd</i>	Filename of the system user catalog (DDLSEC) area
<i>idms.sysuser.ddlsec</i>	File-ID of the system user catalog (DDLSEC) area
<i>dictdb</i>	Filename of the application dictionary definition area
<i>idms.appldict.ddldml</i>	File-ID of the application dictionary definition (DDLML) area
<i>dloddb</i>	Filename of the application dictionary definition load area
<i>idms.appldict.ddldclod</i>	File-ID of the application dictionary definition load (DDLCLOD) area

<i>sqldd</i>	Filename of the SQL catalog (DDLCAT) area
<i>idms.syssql.ddlcat</i>	File-ID of the SQL catalog (DDLCAT) area
<i>sqlld</i>	Filename of the SQL catalog load (DDLCTL) area
<i>idms.syssql.ddlctl</i>	File-ID of SQL catalog load (DDLCTL) area
<i>sqlxdd</i>	Filename of the SQL catalog index (DDLCATX) area
<i>idms.syssql.ddlcatx</i>	File-ID of the SQL catalog index (DDLCATX) area
<i>asfdml</i>	Filename of the asf dictionary definition (DDLML) area
<i>idms.asfdict.ddlml</i>	File-ID of the asf dictionary definition (DDLML) area
<i>asflod</i>	Filename of the asf dictionary definition load (ASFLOD) area
<i>idms.asfdict.asflod</i>	File-ID of the asf dictionary definition load (ASFLOD) area
<i>asfdata</i>	Filename of the asf data (ASFDATA) area
<i>idms.asfdict.asfdata</i>	File-ID of the asf data area (ASFDATA) area
<i>ASFDEFN</i>	Filename of the asf data definition (ASFDEFN) area
<i>idms.asfdict.asfdefn</i>	File-ID of the asf data definition area (ASFDEFN) area
<i>j1jrn1</i>	Filename of the first disk journal file
<i>idms.j1jrn1</i>	File-ID of the first disk journal file
<i>j2jrn1</i>	Filename of the second disk journal file

<i>idms.j2jrn1</i>	File-ID of the second disk journal file
<i>j3jrn1</i>	Filename of the third disk journal file
<i>idms.j3jrn1</i>	File-ID of the third disk journal file
<i>SYSIDMS</i>	Filename of the SYSIDMS parameter file

Setting Defaults for Batch Processing

Since CA OLQ batch processing does not have access to the defaults set during system generation, you must explicitly define these defaults in batch.

Table 12 presents the options available, the batch defaults, and the syntax necessary to override the defaults.

Table 12. Batch Options, Defaults, and Syntax

Option	Default	Override Syntax
Print line size	80	SET PRINT LINE SIZE
Print line count	60	SET PRINT LINE COUNT
Internal storage page size	1920	SET USER ... INTERNAL STORAGE PAGE SIZE
Report page size	4000	SET USER ... REPORT PAGE SIZE
Menu mode	Disallowed	N/A
Continuation character	-	SET CONTINUATION CHARACTER
Separator character	!	SET SEPARATOR CHARACTER
Comment character	;	SET COMMENT CHARACTER
Report retention	1	SAVE REPORT ... RETENTION PERIOD
Maximum report retention	32767	N/A
Maximum report pages	32767	N/A
Terminal line size	132	SET LINE SIZE

Option	Default	Override Syntax
Terminal line count	60	SET LINE COUNT
Report dictionary database name	blanks	SET REPORT DICTNAME SET USER ... REPORT DICTNAME
Maximum sort space in K bytes	384	SET MAXIMUM SORT SIZE=

For more information:

[Commands and Syntax](#) (see page 53)

Defining files

CA OLQ batch retrieves records from the file you specify in the JCL. (If retrieving only from the database, you do not have to specify an input file in the JCL.) The access switch must be set to **olq** to read or write sequential files.

You can define the characteristics of each input file the CA OLQ batch job stream accesses by using the **DEFINE FILE** statement. **DEFINE FILE** associates a file name and file characteristics with IDD record and element entities.

Example:

If your batch job accesses an input file called INFILE, you would include the following **DEFINE FILE** statement in your input data stream:

```
DEFINE FILE INFILE RECORD EMPL-2 (3) DICT TESTDICT
RECFM=F BLKSIZE=80 LRECL=80
```

- The **INFILE** file is used to access input data from the **EMPL-2** record.
- The record definition for the **EMPL-2** record is stored in the **TESTDICT** dictionary.
- The number in parenthesis, **(3)**, is the version number. If no version number is specified, the default is 1.

When using **DEFINE FILE**, the corresponding job control language statement must name the file in the DD name:

Table 13

Operating System	JCL Statement
z/OS	//INFILE DD DSN= <i>infile</i> ,DISP=SHR
z/VSE	//DLBL INFILE,' <i>infile</i> '

Operating System	JCL Statement
	// EXTENT <i>sysnnn, nnnnnn,,,ssss,zzzz</i>
	// ASSIGN <i>sysnnn,DISK,VOL=nnnnnn,SHR</i>
CMS	FILEDEF INFILE DISK <i>filename filetype filemode</i> (RECFM=F BLKSIZE= <i>nn</i> LRECL= <i>nn</i>)

Defining input files:

You can also use DEFINE FILE to define the following characteristics for each input file:

- Record format (required for z/VSE)
- Block size (required for z/VSE)
- Logical record length (required for z/VSE)
- File type
- Device type
- Logical unit (z/VSE only)

Defining output files:

The output file is the file the report data is written to. You use the DEFINE FILE statement to direct the output to a file:

```
DEFINE FILE OUTFILE OUTPUT
```

This command marks the file OUTFILE as containing the report output.

The data set is written out in its unformatted condition to the output file.

If you want to use the output file as an input file, you must do so in a subsequent job step.

Defining sort work files:

During SORT requests and SELECT with GROUP BY requests, CA OLQ attempts to use the operating system sort utility. If you want CA OLQ to use the operating system sort, you must define appropriate sort work files in the batch job stream. If these sort work files are not in the job stream, CA OLQ uses its internal sort routines.

Note: CA OLQ *always* attempts the operating system sort. If the sort attempt fails, CA OLQ switches to the internal sort. An operating system sort utility is required for CMS users. If no operating sort utility is available, contact CA Technical Support for assistance.

For more information:

[Commands and Syntax](#) (see page 53)

Signing on in batch

The **SET** command in batch allows you to identify:

- The user, maximum internal page size, report file page size, and report dictionary—**SET USER**
- The maximum sort size—**SET MAXIMUM SORT SIZE**
- The report dictionary name—**SET REPORT DICTNAME**

Specifying user name:

The **SET USER** statement must be the first statement in your input data stream.

For online batch submission, you can use a substitution string in the batch job control language to specify the user ID. This allows you to share JCL between users without changing the user ID in the JCL. Online CA OLQ performs the substitution during online submission.

The string **\$USER** is changed to the current user ID. This is a straight string substitution. For example:

```
//$USERBAT JOB ...
```

When XYZ is signed on and submits a CA OLQ batch job, this statement becomes:

```
//XYZBAT JOB ...
```

For more information:

[Commands and Syntax](#) (see page 53)

OLQBatch Notification

You can use OLQBNOTE to notify a DC user that a batch job has completed. If the specified user is not signed on at the time, the notification will be discarded. OLQBNOTE runs central version and accepts one or more control cards of the format:

USER= *user-id*,MESSAGE= *message*,BEEP

- **MESSAGE** can be shortened to **MSG** or **MES**.
- **USER** can be shortened to **USE**.
- **BEEP** sounds the terminal alarm, is optional, and should not be used against a non-3270 terminal.
- **Message** should be enclosed in quotation marks (' ') if it contains blanks and double quotation marks (" ") if it contains single quotation marks.

For online batch submission, the user ID may be specified as **\$USER**. CA OLQ substitutes the DC user ID in its place upon job submission.

Setting up notification:

To use the notification facility, you must define the following during **system generation**:

- **OLQQNOTE** as a queue
- **OLQTNOTE** as a task
- **OLQSNOTE** as a program

OLQBNOTE Example for z/OS

z/OS JCL:

OLQBNOTE (z/OS)

```
//NOTIFY EXEC PGM=OLQBNOTE,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
//sysctl DD DSN=idms.sysctl,DISP=SHR
//dcmsg DD DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//SYSLST DD SYSOUT=A
//SYSIDMS DD *
DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
//SYSIPT DD *
USER=$USER MESSAGE='OLQ Batch completed'
/*
/*
```


Data set name of the load library
containing the DMCL and
database name table load
modules

Data set name of the load library
containing the CA IDMS
executable modules

DDname of the SYSCTL file

Data set name of the SYSCTL file

.....

DDname of the system message
(DDLDCMSG) area

Data set name of the system
message (DDLDCMSG) area

Specifies the name of the DMCL load module

Note: \$USER only works when OLQBNOTE is submitted from CA OLQ.

OLQBNOTE example for CMS

CMS commands:

OLQBNOTE (CMS)

```
FILEDEF SYSIDMS DISK sysidms parms a
FILEDEF SYSIPT DISK olqbnote input a (RECFM F LRECL lll BLKSIZE bbb)
EXEC IDMSFD
OSRUN OLQBNOTE
```

<i>sysidms parms a</i>	FileID of the file containing SYSIDMS parameters
<i>olqbnote input a</i>	FileID of the file containing the OLQBNOTE input parameters
<i>lll</i>	Logical record length of the user input data file
<i>bbb</i>	Block size of the user input data file
<i>IDMSFD</i>	Exec which defines all FILEDEFS, TXTLIBs, and LOADLIBs required by the system

SYSIDMS file:

To run OLQBNOTE, you should include these SYSIDMS parameters:

- *DMCL=dmcl-name*, to identify the DMCL load module
- *Any other SYSIDMS parameters*

How to create the SYSIDMS file:

To create the SYSIDMS file of SYSIDMS parameters:

1. On the CMS command line, type:
XEDIT sysidms parms a (NOPROF
2. Press [Enter]
3. On the XEDIT command line, type:
INPUT
4. Press [Enter]
5. In input mode, type in the SYSIDMS parameters
6. Press [Enter] to exit input mode
7. On the XEDIT command line, type:
FILE
8. Press [Enter]

Note: For documentation of SYSIDMS parameters, see the *CA IDMS Database Administration Guide*.

SYSIPT file:

To include SYSIPT, you should add this statement:

```
USER=$USER MESSAGE='OLQ Batch completed'
```

To create the SYSIPT file of OLQBNOTE input parameters:

1. On the CMS command line, type:
XEDIT sysipt input a (NOPROF
2. Press [Enter]
3. On the XEDIT command line, type:
INPUT
4. Press [Enter]
5. In input mode, type in the OLQBNOTE input parameters
6. Press [Enter] to exit input mode
7. On the XEDIT command line, type:
FILE
8. Press [Enter]

OLQBNOTE example for z/VSE

z/VSE JCL:

OLQBNOTE (z/VSE)

```
// JOB    OLQBNOTE
// DLBL   idmslib,'idmslib.library'
// EXTENT sysxxx,vvvvv,,sss,ttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA IDMS load libraries
// EXEC   PROC=IDMSLBLS
// EXEC   PROC=sysctl
// EXEC   OLQBNOTE,SIZE=1024K
        DMCL=dmc1-name
        Put other SYSIDMS parameters, as appropriate, here
/*
        USER=$USER MESSAGE='OLQ Batch completed'
/*
```

Dtfname of the CA IDMS library

Data set name of CA IDMS load
libraries, as established during
installation

SYS number

Volume serial number

Starting extent

Number of tracts

The CA IDMS load libraries, as established during installation

Procedure name containing the SYCTL file

Name of the DMCL

Batch Class Specification

The CA OLQ system generation BATCH CLASS must be non zero to enable CA OLQ batch submission from online.

As with all users exits, you should ensure the exit meets your site's standards.

Note: See the *CA IDMS System Operations Guide*. for a discussion on installing user exits.

Except for the z/OS environment, user exits are required to submit batch jobs. Exits provided are:

- RHDCUX26 for z/VSE
- RHDCUX21 for CMS

For the z/OS environment, turn off "banner" output for the batch print class.

Operating System Dependent Installation Instructions

To run CA OLQ batch jobs efficiently, tailor the job control language according to your operating system.

z/OS:

1. Change CA OLQ system generation parameter BATCH CLASS to a number between 1 and 64.
2. Create a DC printer defined to go to the JESRDR (or HASPRDR) for the print class specified above. For example:

```
//JESRDR DD SYSOUT=(A,INTRDR),  
          DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
```

3. Ensure that the last card in the IDDJCL module(s) is:

```
/*EOF
```

If this is not done, submitted jobs will remain on the reader until another job is submitted.

CMS:

1. Change CA OLQ system generation parameter BATCH CLASS to a number between 1 and 64.
2. Modify RHDCUX21 (source on tape) and assemble:
 - a. Change BATCLASS to same value as above.
 - b. Change WKUSER to desired z/VM batch machine.
3. Modify RHDCUXIT to add user exit 21:
 - a. Add #DEFXIT MODE=SYSTEM,CALL=DC,NAME=RHDCUX21
4. The FILEDEF corresponding to the printer definition for the BATCH CLASS should go to PUNCH and not PRINT. For example:

```
FILEDEF SYSPRT PUNCH (RECFM F LRECL 80 BLKSIZE 80.
```
5. Ensure that the PTE associated with the print class has the batch print class value as the only allowable print class.

Note: When running CA OLQ directly on CMS (that is, not under DC), if you intend to do any CA OLQ request that require a sort, you must have an external sort available at run time.

z/VSE:

1. Change CA OLQ system generation parameter BATCH CLASS to a number between 1 and 64. CA OLQ does not use this value other than to determine if job submission is enabled in z/VSE.
2. Modify RHDCUXIT to add user exit 26:

```
Add #DEFXIT MODE=USER,CALL=IBM,NAME=RHDCUD26
```
3. Ensure RHDCU26B is defined to your DC system as:

```
ASSEMBLER NONREENTRANT NOPROTECT
```
4. Ensure that the last card in the JCL module(s) is:

```
/*EOF
```

The CA provided exit requires this to detect the end of the job stream. This card is not submitted to power.

Examples of Batch

Displaying reports:

This example displays those employees whose employee ID numbers are less than or equal to 40.

Input:

```
SET USER ABC PASS ABCPASS
SET ACCESS OLQ
SIGNON DICT=TESTDICT SS=EMPSS01 DBN=EMPDEMO
SELECT * FROM EMPLOYEE WHERE EMP-ID-0415 LE 40 ORDER BY EMP-LAST-NAME-0415
DISPLAY COLS EMP-ID-0415 EMP-LAST-NAME-0415
```

Output page 1:

```
OLQ RELEASE 16.0 DATE: 96.267 TIME: 11:00 PAGE: 0001
Copyright (C) 2003 CA, Inc.

OLQ 107017 00 CA OLQ Release 16.0
OLQ 107019 00 Copyright(C) 2003 CA, Inc.
SET USER ABC PASS ????????
SIGNON SS EMPSS01 DBN=EMPDEMO DICTNAME TESTDICT

OLQ 100021 00 Ready to retrieve data from subschema IDMSNMKA
OLQ 100022 00 Schema: EMPSCH Version: 1
OLQ 100023 00 Database name: EMPDEMO
OLQ 100025 00 Dictionary name: TESTDICT
SELECT * FROM EMPLOYEE WHERE EMP-ID-0415 LE 40
OLQ 098006 00 18 Whole lines and 0 partial lines in report.
OLQ 098007 00 18 Records read. 18 Records selected.
DISPLAY COLS EMP-ID-0415 EMP-LAST-NAME-0415
```

Output page 2:

EMPLOYEE.REPORT

EMP-ID-0415	EMP-LAST-NAME-0415
7	BANK
40	CRANE
24	DOUGH
32	FERNDALE
29	GALLWAY
3	GARFIELD
28	GRANGER
27	HEAROWITZ
30	HENDON
20	JACOBI
19	JENSEN
11	JENSON
16	KLEWELLEN
31	LIPSICH
35	LITERATA
15	MAKER
23	O'HEARN
12	PEOPLES
21	TYRO

Creating Multiple Reports in One Job

To create several reports in one batch run, enter the CA OLQ processing statements sequentially in your input data stream. The DISPLAY command places the start of each report on a new page, and resets the page count to 1.

Example:

This batch run creates two reports: one listing the names of employees living in Boston and one listing the names of employees living in Medford:

```

DEFINE FILE INFILE RECORD EMPLOYEE (1) DICT TESTDICT
SET OPTION SPARSE
GET ALL SEQ INFILE.EMPLOYEE WHERE EMP-CITY-0415 EQ
  BOSTON
DIS COLS EMP-NAME-0415 EMP-CITY-0415
GET ALL SEQ INFILE.EMPLOYEE WHERE EMP-CITY-0415 EQ
  MEDFORD
DIS COLS EMP-NAME-0415 EMP-CITY-0415

```

Wide Reports

To display wide reports in CA OLQ batch, specify the keywords RIGHT or LEFT with your DISPLAY command.

Example:

If you specify DISPLAY RIGHT, your output report lists the next 132 characters (from left to right) of the report, for as many pages as there are in the report. The display shifts right each time you issue a DISPLAY RIGHT.

Creating a Report with SELECT

This example uses a SELECT statement to retrieve information about departments and employees. It also creates a report (rather than writing to disk). To display your output in a report, issue a DISPLAY command in your input data stream:

INPUT:

```
SET USER ABC PASS ABCPASS
SET ACCESS OLQ
OLQ 092032 00 PROCESSING MODE CHANGED TO OLQ.
SELECT * FROM DEPARTMENT, EMPLOYEE WHERE -
  DEPARTMENT.DEPT-HEAD-ID-0410 EQ EMPLOYEE.EMP-ID-0415
DISPLAY COLS DEPT-NAME-0410 DEPT-HEAD-ID-0410 EMP-LAST-NAME-0415
```

Output:

OLQ RELEASE 16.0 DATE: 96.267 TIME: 11:00 PAGE: 0001
 Copyright (C) 2003 CA, Inc.

OLQ 107017 00 CA OLQ Release 16.0
 OLQ 107019 00 Copyright(C) 2003 CA, Inc.
 SET USER ABC PASS ???????
 SIGNON SS EMPSS01 DBN=EMPDEMO DICTNAME TESTDICT

OLQ 100021 00 Ready to retrieve data from subschema IDMSNWKA
 OLQ 100022 00 Schema: EMPSCHM Version: 1
 OLQ 100023 00 Database name: EMPDEMO
 OLQ 100025 00 Dictionary name: TESTDICT
 SELECT * FROM DEPARTMENT, EMPLOYEE WHERE -
 OLQ 092022 00 Continuation line accepted
 DEPARTMENT.DEPT-HEAD-ID-0410 EQ EMPLOYEE.EMP-ID-0415
 OLQ 098006 00 9 whole lines and 0 partial lines in report.
 OLQ 098007 00 18 records read. 18 records selected.
 DISPLAY COLS DEPT-NAME-0410 DEPT-HEAD-ID-0410 EMP-LAST-NAME-0415

DEPARTMENT/EMPLOYEE REPORT
 9/19/91

DEPT-NAME-0410	DEPT-HEAD-ID-0410	EMP-LAST-NAME-0415
EXECUTIVE ADMINISTRATION		30 HENDON
ACCOUNTING AND PAYROLL		11 JENSON
PERSONNEL		13 PEOPLES
INTERNAL SOFTWARE		3 GARFIELD
BLUE SKIES		321 MOON
COMPUTER OPERATIONS		4 CRANE
PUBLIC RELATIONS		7 BANK
BRAINSTORMING		15 MAKER
THERMOREGULATION		349 WILCO
END OF REPORT		

OLQ RELEASE 16.0 DATE: 96.267 TIME: 11:00 PAGE: 0003
 Copyright (C) 2003 CA, Inc.

BYE
 OLQ 100029 00 Signoff accepted - OLQ session terminated.

Writing to a Disk File with SELECT

This example routes the outcome of a SELECT statement to a disk file. To route your output to an output file, you must:

1. Set the access switch to **olq**
2. Specify the keyword **OUTPUT** and the name of the output file in your SELECT statement

Example:

To route your output to the OUTFILE output file, issue the following statements:

```
set access olq
select * from employee
where emp-id-0415 eq 4500 output outfile
```

This example creates a sequential file containing departments and the name of each department's manager:

Input:

```
SET USER ABC PASS ABCPASS
SIGNON SS EMPSS01 DBN=EMPDEMO DICTNAME TESTDICT
SELECT * FROM DEPARTMENT, EMPLOYEE WHERE -
DEPARTMENT.DEPT-HEAD-ID-0410 EQ EMPLOYEE.EMP-ID-0415 -
OUTPUT OUTFILE
```

Output:

OLQ RELEASE 16.0 DATE: 96.267 TIME: 11:00 PAGE: 0001
 Copyright (C) 2003 CA, Inc.

OLQ 107017 00 CA OLQ Release 16.0
 OLQ 107019 00 Copyright(C) 2003 CA, Inc.
 SET USER ABC PASS ????????
 SIGNON SS EMPSS01 DBN=EMPDEMO DICTNAME TESTDICT

OLQ 100021 00 Ready to retrieve data from subschema IDMSNWKA
 OLQ 100022 00 Schema: EMPSCHM Version: 1
 OLQ 100023 00 Database name: EMPDEMO
 OLQ 100025 00 Dictionary name: TESTDICT
 SELECT * FROM DEPARTMENT, EMPLOYEE WHERE -
 OLQ 092022 00 Continuation line accepted
 DEPARTMENT.DEPT-HEAD-ID-0410 EQ EMPLOYEE.EMP-ID-0415 -
 OLQ 092022 00 Continuation line accepted
 OUTPUT OUTFILE

OLQ 149018 00

File name	Field name	Offset	Size	Dec no	Data type
OUTFILE	DEPT-ID-0410	0000	0004	0000	UNSIGNED ZONE
OUTFILE	DEPT-NAME-0410	0004	0045		CHARACTER
OUTFILE	DEPT-HEAD-ID-0410	0049	0004	0000	UNSIGNED ZONE
OUTFILE	EMP-ID-0415	0053	0004	0000	UNSIGNED ZONE
OUTFILE	EMP-FIRST-NAME-0415	0057	0010		CHARACTER
OUTFILE	EMP-LAST-NAME-0415	0067	0015		CHARACTER
OUTFILE	EMP-STREET-0415	0082	0020		CHARACTER
OUTFILE	EMP-CITY-0415	0102	0015		CHARACTER
OUTFILE	EMP-STATE-0415	0117	0002		CHARACTER
OUTFILE	EMP-ZIP-FIRST-FIVE-0415	0119	0005		CHARACTER
OUTFILE	EMP-ZIP-LAST-FOUR-0415	0124	0004		CHARACTER
OUTFILE	EMP-PHONE-0415	0128	0010	0000	UNSIGNED ZONE
OUTFILE	STATUS-0415	0138	0002		CHARACTER
OUTFILE	SS-NUMBER-0415	0140	0009	0000	UNSIGNED ZONE
OUTFILE	START-YEAR-0415	0149	0002	0000	UNSIGNED ZONE
OUTFILE	START-MONTH-0415	0151	0002	0000	UNSIGNED ZONE
OUTFILE	START-DAY-0415	0153	0002	0000	UNSIGNED ZONE
OUTFILE	TERMINATION-YEAR-0415	0155	0002	0000	UNSIGNED ZONE
OUTFILE	TERMINATION-MONTH-0415	0157	0002	0000	UNSIGNED ZONE
OUTFILE	TERMINATION-DAY-0415	0159	0002	0000	UNSIGNED ZONE
OUTFILE	BIRTH-YEAR-0415	0161	0002	0000	UNSIGNED ZONE
OUTFILE	BIRTH-MONTH-0415	0163	0002	0000	UNSIGNED ZONE
OUTFILE	BIRTH-DAY-0415	0165	0002	0000	UNSIGNED ZONE

Chapter 12: Setting Defaults

The option defaults described here can be overridden with the CA OLQ SET and OPTIONS commands. If these options are not specified during a CA OLQ session, the options take on the default values as described in this chapter.

You set these options during system generation or with IDD.

System Generation Options

Table 14 presents the options available during system generation and the statements that control the options. The system generation statement is ADD/MODIFY/DELETE OLQ.

Table 14. CA OLQ Options Set During System Generation

Option	Clause of the Statement
OLQ	INCLUDE/EXCLUDE AUTHORITY IS OLQ
Access to ASF tables, logical and database records, and sequential files	OLQ ACCESS TO OLQ
Access to SQL tables	OLQ ACCESS TO IDMS
PF key module	PFKEY MODULE IS qfile-name
Interrupt count	INTERRUPT COUNT IS 100/interrupt-count
Maximum interrupt count	MAXIMUM INTERRUPT COUNT IS 32767/maximum-interrupt-count
Menu mode	MENU MODE IS ALLOWED/DISALLOWED/ONLY
Report retention	REPORT RETENTION IS 1/retention-period/FOREVER
Maximum report retention	MAXIMUM REPORT RETENTION IS 5/max-report-retention/FOREVER
Size of report pages (in bytes)	REPORT FILE PAGE SIZE IS 4000/report-file-page-size
Maximum number of reports	MAXIMUM REPORT COUNT IS 5/maximum-report-count
Number of pages per report	MAXIMUM REPORT PAGES IS 5/maximum-report-pages

Option	Clause of the Statement
Maximum storage for sorts	MAXIMUM SORT STORAGE IS 100/max-sort-storage-size
Number of input lines	INPUT LINE SIZE IS 4/input-line-size
Continuation character	CONTINUATION CHARACTER IS '-/continuation-character
Separator character	SEPARATOR CHARACTER IS '!'/separator-character
Comment character	COMMENT CHARACTER IS ';'/'comment-character
Default report dictionary name	REPORT DICTNAME IS dictionary-name
Batch class	BATCH CLASS IS 0/batch-class

Special considerations for the options available during system generation are presented below.

Access to ASF tables:

You can create, replace, modify, and delete ASF tables.

You use the IDD DDDL source statement USER to set the access switch to **OLQ**:

```
OLQ ACCESS IS OLQ
```

OLQ is the default.

The OLQ setting also allows you to use the SELECT statement with the following entities:

- ASF tables
- Logical records
- Database records
- Sequential files (batch only)

Access to SQL tables:

You can create, replace, modify, and delete tables which are associated with an SQL schema.

You use the IDD DDDL source statement `USER` to set the access switch to **IDMS**:

```
OLQ ACCESS IS IDMS
```

Note:

- For information on the `USER` statement, see the *CA IDMS IDD DDDL Reference Guide*.
- For more information about System Generation, see the *CA IDMS System Generation Guide*.

PF key module:

You use the system generation statement `ADD OLQ` to define a qfile as the PF key module to be executed for each user issuing the CA OLQ task code. The intended use of the PF key module is to contain a series of CA OLQ `SET FUNCTION` commands that assign values to each of the PF keys in CA OLQ.

The specific clause of the `ADD OLQ` statement is:

```
PFKEY MODULE IS qfile-name
```

How to set interrupt:

The interrupt count interrupts processing after the specified number of CA OLQ database requests. You can set:

- A **default interrupt count**— To specify the interrupt count should the user not specify one
- A **maximum interrupt count**— To specify the interrupt count should the user specify a count greater than the default interrupt count mentioned above

The system generation statements that set these defaults are:

```
ADD/MODIFY OLQ ...  
INTERRUPT COUNT IS interrupt-count  
MAXIMUM INTERRUPT COUNT IS max-interrupt-count
```

These statements set the default interrupt count and the default maximum interrupt for CA OLQ.

Access to menu mode:

Specify a user's access to menu mode with:

```
MENU MODE IS ALLOWED/NOT ALLOWED/ONLY
```

Users are automatically allowed to access command mode with the ALLOWED and NOT ALLOWED options. **ONLY** specifies that the user is allowed to access CA OLQ through menu mode only.

Maximum sort storage:

You can set the maximum amount of storage, in K bytes, that CA OLQ can use for sort operations. Specifying too low a value prevents CA OLQ from performing sort operations. Specifying too large a value degrades CV performance.

The system generation statement that sets this option is:

```
ADD/MODIFY OLQ ...  
MAXIMUM SORT STORAGE IS 100/max-sort-storage-size
```

Number of input lines:

You can define the number of lines on the screen reserved for command input. The maximum number of lines defined by **input-line-size** is limited to the total number of lines on the screen minus the number of lines allocated for output.

Keep in mind that a large line size may be needed to comfortably code SELECT statements or multiple commands.

The system generation statement that sets this option is:

```
ADD/MODIFY OLQ ...  
INPUT LINE SIZE IS 4/input-line-size
```

Continuation character:

To set the default continuation character during system generation, use the statement:

```
ADD/MODIFY OLQ ...  
CONTINUATION CHARACTER IS -/continuation-character
```

Use the continuation character in qfiles and batch when the length of a CA OLQ command exceeds one line. You can also use the continuation character to continue commands (or a series of commands) across a pseudo-converse. Specify the continuation character at the end of each line to be continued.

Separator character:

The separator character is used to separate commands in the command input area. To set the default separator character during system generation, use the statement:

```
ADD/MODIFY OLQ ...
SEPARATOR CHARACTER IS !/separator-character
```

Comment character:

The comment character is used to separate commands from comments. Anything typed in after the comment character is considered to be a comment and is ignored by CA OLQ. Entering comments can be useful in documenting qfiles.

To set the default comment character during system generation, use the statement:

```
ADD/MODIFY OLQ ...
COMMENT CHARACTER IS ;/comment-character
```

Batch class:

Batch class is used to activate the CA OLQ batch interface. You can specify the print class used by CA OLQ when submitting batch jobs under z/OS. To specify the default print class during system generation, use the statement:

```
ADD/MODIFY OLQ ...
BATCH CLASS IS 0/batch-class
```

You should ensure that the specified class has been assigned to an internal reader.

For more information:

[Batch Processing](#) (see page 303)

Integrated Data Dictionary Options

Table 15 presents the options available through IDD and the statements that control the options.

Table 15. CA OLQ Options Set With the IDD ADD/MODIFY/DELETE USER Statement

Option	Clause of the Statement
OLQ	INCLUDE/EXCLUDE AUTHORITY IS OLQ

Option	Clause of the Statement
Access to ASF tables, logical and database records, and sequential files	OLQ ACCESS TO OLQ
Access to SQL tables	OLQ ACCESS TO IDMS
Subschema access	INCLUDE/EXCLUDE ACCESS TO SUBSCHEMA
rule = no.Signon qfile	<i>ss-name</i> OF SCHEMA <i>schm-name</i> USER <i>user-name</i> SIGNON QFILE IS <i>qfile-name</i>
Signon profile qfiles	SIGNON PROFILE IS <i>qfile-name</i> LANGUAGE IS OLQ
Access to qfiles	INCLUDE/EXCLUDE ACCESS TO QFILE <i>qfile-name</i>
Execution of qfiles	QFILE IS ALLOWED/NOT ALLOWED/ONLY
Menu mode	MENU MODE IS ALLOWED/NOT ALLOWED/ONLY
Saving qfiles	QFILE SAVE IS ALLOWED/NOT ALLOWED
Retrieving multiple records	MRR IS ALLOWED/NOT ALLOWED
Interrupt count	MANDATORY/OPTIONAL INTERRUPT
Sorts	SORT IS ALLOWED/NOT ALLOWED
Default options	DEFAULT OPTIONS ARE ...

Special considerations for the options available through IDD are presented below.

Note: For complete syntax and syntax rules, see the *CA IDMS IDD DDDL Reference Guide*.

OLQ:

You can allow users to distribute authority in IDD with the `IDD USER` statement. `USER` statement clauses are used to control access to CA OLQ qfiles and subschema views and to assign CA OLQ command authorities and processing and reporting options when the default processing options for the session include `SECURITY FOR OLQ IS ON`.

If OLQ is specified, the keyword `UPDATE` must be specified in the `FOR` clause of the `ADD USER` statement.

Access to ASF tables:

You can create, replace, modify, and delete ASF tables.

You use the IDD DDDL source statement `USER` to set the access switch to **OLQ**:

```
OLQ ACCESS IS OLQ
```

OLQ is the default.

The OLQ setting also allows you to use the `SELECT` statement with the following entities:

- ASF tables
- Logical records
- Database records
- Sequential files (batch only)

Access to SQL tables:

You can create, replace, modify, and delete tables which are associated with an SQL schema.

You use the IDD DDDL source statement `USER` to set the access switch to **IDMS**:

```
OLQ ACCESS IS IDMS
```

Note: For information on the `USER` statement, see the *CA IDMS IDD DDDL Reference Guide*.

Subschema access:

You can specify whether the user does or doesn't have access to a particular subschema. CA OLQ uses the DC/UCF signon ID and password of the user to determine which subschemas are available. Subschema security is enforced on a dictionary-by-dictionary basis.

Signon profile qfiles and users:

You can associate a signon qfile with a subschema. The signon qfile executes when a user signs on to the subschema.

You use the IDD DDDL source statement entity `USER` to associate a signon profile with a specific user. The specific clause of the `ADD USER` statement is:

```
SIGNON PROFILE IS module-name  
LANGUAGE IS OLQ
```

Module-name refers to the name of the saved qfile.

Access to qfiles:

You can allow users to access only certain qfiles by naming the qfiles in the user's ADD USER statement, using the clause:

```
INCLUDE/EXCLUDE ACCESS TO QFILE qfile-name
```

Executing qfiles:

Use IDD to specify whether or not specific users are allowed to execute qfiles in general. The IDD statement that applies to this option is the following clause of the ADD USER statement:

```
QFILE IS ALLOWED/NOT ALLOWED/ONLY
```

With the ONLY option of the above clause, you can specify that the named user can access CA OLQ *only* through qfile execution. If you want the user to access CA OLQ only through qfiles, we recommend that you also specify MENU MODE IS DISALLOWED to keep the user from retrieving through menu mode.

Menu mode:

This security option assigns or denies access to the CA OLQ menu mode facility as follows:

- **ALLOWED** (default) authorizes the CA OLQ user to access CA OLQ in command mode and menu mode.
- **NOT ALLOWED** authorizes the CA OLQ user to access CA OLQ in command mode only.
- **ONLY** authorizes the CA OLQ user to access CA OLQ in menu mode only.

The IDD statement that applies to this option is:

```
MENU MODE IS ALLOWED/NOT ALLOWED/ONLY
```

Saving qfiles:

You can specify whether or not you want the user to be able to save qfiles after creating them with the following clause of the ADD USER statement:

```
QFILE SAVE IS ALLOWED/NOT ALLOWED
```


Retrieving multiple records:

This security option specifies whether the user can retrieve multiple record occurrences with a single CA OLQ command. To use the SELECT (IDMS access mode) statement, the user must be assigned authority to retrieve multiple record occurrences.

The IDD statement that applies to this option is:

```
MRR IS ALLOWED/NOT ALLOWED
```

Interrupt count:

Each user in CA OLQ has the ability to set an INTERRUPT COUNT during a CA OLQ session. The interrupt count is the number of records that can be retrieved before the system interrupts processing. As the DBA, you can specify whether the user is allowed to select the NO INTERRUPT option with the IDD option:

```
MANDATORY/OPTIONAL INTERRUPT
```

If you set MANDATORY INTERRUPT, the user cannot choose the NO INTERRUPT option in CA OLQ. If you set OPTIONAL INTERRUPT, the user can choose the NO INTERRUPT option in CA OLQ. When the NO INTERRUPT option is in effect, the default interrupt count set during system generation is used to perform commit checkpoints.

Sorts:

When you specify:

```
SORT IS NOT ALLOWED
```

keep in mind that you are prohibiting sorting in OLQ SELECT commands.

Setting CA OLQ default options:

The CA OLQ default options allow you to tailor the CA OLQ environment. These options come into effect at user signon regardless of the security status. The CA OLQ default options include:

- **HEADER/NO HEADER** specifies whether CA OLQ report files contain a header line. This option has no effect on single-record-occurrence retrieval displays.
- **ECHO/NO ECHO** specifies whether a user-entered command will be repeated by CA OLQ on the output device.

- **ALL/NONE** specifies whether the default internal field list for all records retrieved during the user's CA OLQ session will contain all or none of the fields. Menu mode always defaults to none; no fields are preselected.
- **FILLER/NO FILLER** specifies whether filler field values are displayed.
- **INTERRUPT/NO INTERRUPT** specifies whether the processing interrupt feature for multiple record retrievals is enabled or disabled.

Note: The MANDATORY INTERRUPT specification takes precedence over NO INTERRUPT.

If INTERRUPT is specified, CA OLQ breaks processing after the specified number of records has been retrieved. If NO INTERRUPT is specified (as long as OPTIONAL INTERRUPT has been specified), the interrupt count is used for commit checkpoints.

- **WHOLE/PARTIAL** specifies the content of displayed path retrieval report lines. WHOLE displays only those lines containing a retrieved occurrence for every record type in a path definition. PARTIAL displays all lines, whether or not they contain data for every path record type. SELECT (OLQ access mode), and therefore menu mode, doesn't build partial lines.
- **FULL/SPARSE** specifies the format of displayed path retrieval report lines. FULL displays data associated with a record type once for each retrieved occurrence. SPARSE, used with a SELECT command, displays only the first of a repeating data value; SPARSE, used with a path command, displays only the first of a repeating record type.
- **OLQ HEADER/NO OLQ HEADER** specifies whether CA OLQ is to use predefined headers as columns headers in the report. This option has no effect on single-record-occurrence retrieval displays.
- **COMMENTS/NO COMMENTS** specifies whether comments accompany the output from HELP RECORDS, HELP SUBSCHEMAS, and HELP QFILE requests.
- **CODETABLE/NO CODETABLE** specifies whether CA OLQ accesses a code table to encode and decode data.
- **PATHSTATUS/NO PATHSTATUS** specifies the conditions under which CA OLQ will retrieve a logical record. NO PATHSTATUS requests CA OLQ to retrieve a logical record only when the path status of LR-FOUND is returned. PATHSTATUS requests CA OLQ to retrieve a logical record when any DBA-defined path status is returned.
- **EXTERNAL PICTURE/NO EXTERNAL PICTURE** specifies whether CA OLQ will use external pictures for displaying data.
- **VERBOSE/TERSE** controls the amount of information displayed following record and field-level breaks.

Index

A

- ASF tables • 301
 - ASF table • 301
 - passkey • 301
 - security • 301

B

- batch • 325, 327, 328, 336, 338, 339, 340, 342, 345
 - class specification • 336, 345
 - DEFINE FILE command • 325
 - multiple reports • 339
 - notification • 328
 - OLQBNOTE • 328
 - reports • 338
 - reports, multiple • 339
 - reports, SELECT • 340
 - signing on • 327
 - wide reports • 340
 - writing to disk • 342
- built-in function • 211, 220, 221, 222, 223, 224, 225, 226, 227, 230, 231
 - coding parameters • 220
 - error processing • 220
 - invoking • 211
 - parameters of • 220
 - table of • 211
 - types of • 211

C

- CA OLQ • 25, 26, 265, 267, 271, 281, 288, 289, 292, 297
 - db-key retrieval • 289
 - default options • 292
 - dictionary name • 292
 - efficiency • 281
 - entering commands • 25
 - interrupt count • 288
 - interrupt OFF • 288
 - interrupt ON • 288
 - logical record facility (LRF) • 267
 - maximum number of • 292
 - number of reports • 292
 - options • 26
 - PF key assignment • 271

- report default options • 292
- retention • 292
- saving • 292
- setting • 288
- signing off • 25
- Signing on • 25
- signon profile qfile • 271
- size • 292
- tailoring • 265
- using LRF • 267
- with db-key • 289
- CA OLQ headers • 46, 48, 50
 - defining • 46
 - field list • 50
 - field reference clause • 48
 - specifying • 48
- CMS • 331
 - OLQBNOTE • 331
- CMS commands • 310, 311, 313
 - central version • 310, 311
 - local mode • 313
 - usage • 311, 313
- command • 16, 17, 18, 20, 25, 27, 29, 31, 303, 305
 - abbreviation • 29
 - batch • 303
 - binary • 31
 - CA OLQ • 31
 - character string • 31
 - coding considerations • 31
 - commenting • 29
 - continuing • 27
 - creating command file • 303
 - data retrieval • 17
 - data value • 31
 - database key • 31
 - ending string • 29
 - executing a job • 303
 - floating point constant • 31
 - group value • 31
 - hexadecimal number • 31
 - how to enter • 25, 27, 31
 - integer • 31
 - job control language • 305
 - multiple per line • 29
 - qfile • 20

- real number • 31
- report formatting • 18
- separating • 27, 29
- setting up • 303
- subscript, specifying • 31
- system management • 16
- table of • 303
- commands • 21
 - data table processing • 21
 - SQL table processing • 21
- COMPUTE command • 54, 57
 - examples • 54
- COMPUTE GROUP BY command • 57, 61
 - examples • 57

D

- database key • 109
 - values • 109
- DELETE USER command • 71, 72
 - considerations • 72
 - examples • 71
 - REPORT option • 72

E

- examples • 54, 57, 61, 66, 67, 68, 69, 72, 81, 90, 93, 94, 95, 97, 101, 103, 106, 109, 114, 117, 121, 125, 132, 141, 144, 155, 157, 163, 165, 166, 168, 178, 183, 186, 194, 196, 198, 203, 204, 207, 211
 - access • 186
 - ALTSEQ CODE option • 198
 - AND ON field-reference option • 198
 - AQ option • 198
 - ASCENDING option • 198
 - CATALOG option • 196
 - CODETABLE option • 186
 - COMMENT CHARACTER option • 186
 - COMPUTE command • 54
 - COMPUTE GROUP BY command • 57
 - CONTINUATION CHARACTER option • 186
 - DATE option • 186
 - DBNAME option • 194
 - DBNODE option • 194
 - DEFAULT DBNAME option • 186
 - DEFAULT DBNODE option • 186
 - DEFAULT DICTNAME option • 186
 - DEFAULT DICTNODE option • 186
 - default schema • 186
 - DEFINE FILE command • 61

- define path • 66
- delete computation • 67
- DESCENDING option • 198
- DICTNAME option • 194, 196
- DICTNODE option • 194, 196
- DISPLAY command • 72
- EDIT command • 81
- edit computation • 90
- EQUALS option • 198
- example • 81
- examples • 61, 66, 67, 68, 72, 81, 90, 93, 132, 144, 155, 157, 163, 183, 186, 194, 196, 198, 207
- FIELDS FOR command • 95
- FIND/GET logical record • 97
- FIND/GET MOST RECENT command • 101
- FIND/GET OWNER WITHIN SET • 106
- FIND/GET PHYSICAL SEQUENTIAL command • 103
- FIND/GET USING STORAGE KEY command • 109
- FIND/GET WITHIN DBKEYLIST command • 114
- FIND/GET WITHIN INDEX SET command • 117
- FIND/GET WITHIN SET command • 121
- FIND/GET WITHIN SET USING SORTKEY • 125
- function • 132
- FUNCTION option • 186
- help destination • 132
- INTERNAL STORAGE PAGE SIZE option • 186
- interrupt count • 186
- INTERRUPT COUNT option • 186
- LOCATION option • 196
- MAXIMUM SORT SIZE option • 186
- menu • 141
- naming • 81
- ON field-reference option • 198
- option • 144
- OWNER option • 196
- page header/footer • 155
- PAGE HEADER/FOOTER command • 157
- PASSWORD option • 186
- PRINT LINE COUNT option • 186
- PRINT LINE SIZE option • 186
- qfile • 163
- REPORT DICTNAME option • 186
- REPORT FILE PAGE SIZE option • 186
- report name • 207
- REPORT option • 198
- SAVE QFILE command • 165
- SAVE REPORT command • 166

- SCHEMA option • 194
- SELECT (OLQ mode) command • 168
- SELECT command • 168
- SEND TABLE (IDMS mode) command • 183
- SEND TABLE (OLQ mode) command • 178
- SEPARATOR CHARACTER option • 186
- set null character • 186
- setting • 186
- SIGNON command • 194
- SIGNON TABLE command • 196
- SORT command • 198
- SUBSCHEMA option • 194
- SWAP command • 204
- task code • 207
- UNIQUE option • 198
- UNSORT command • 207
- user name • 207
- USER option • 186, 198
- VIEW option • 196
- WITHIN record-name option • 198

F

- field reference clause • 46
 - field reference clause • 46
 - FIND/GET and COMPUTE commands • 46
- FIELDS FOR command • 95, 97
 - examples • 95
 - FIND/GET command • 97
- FIND/GET command • 97, 101, 103, 106, 109, 114, 117, 121, 125
 - examples • 97, 101, 103, 106, 109, 114, 117, 121, 125
 - logical record • 97
 - MOST RECENT option • 101
 - OWNER WITHIN SET • 106
 - PHYSICAL SEQUENTIAL • 103
 - using storage key • 109
 - WITHIN DBKEYLIST • 114
 - WITHIN index SET • 117
 - WITHIN SET • 121
 - WITHIN SET using SORTKEY • 125

G

- global syntax • 35, 42, 43, 46, 48, 50, 53
 - comparison expression • 35, 42
 - DBA-designated • 42
 - field list • 50
 - field reference clause • 46, 48

- FIND/GET WHERE clause • 42
- SELECT WHERE clause • 35

I

- Integrated Data Dictionary • 349
 - ALL option • 349
 - CA OLQ authority • 349
 - default CA OLQ option • 349
 - ECHO option • 349
 - FILLER option • 349
 - FULL option • 349
 - HEADER option • 349
 - INTERRUPT option • 349
 - interrupt processing • 349
 - menu mode access • 349
 - multiple record retrieval • 349
 - NONE option • 349
 - PARTIAL option • 349
 - qfile, access to • 349
 - qfile, executing • 349
 - qfile, saving • 349
 - signon profile qfiles • 349
 - sort • 349
 - SPARSE option • 349
 - subschema access • 349
 - WHOLE option • 349

J

- JCL • 306, 307, 309, 310, 313, 315, 317, 325
 - IDMSLBLS procedure • 317
 - input file • 325
 - JCL (local mode) • 325
 - specifications • 325
 - z/OS (central version) • 306, 307
 - z/OS (local mode) • 307
 - z/OS local mode considerations • 309, 310
 - z/OS(local mode) • 309
 - z/VSE (central version) • 313, 315
 - z/VSE (local mode) • 315, 317
- Job Control Language • 338
 - examples • 338

K

- keyword • 35
 - DBA-designated • 35

L

- LRF • 267

- command mode • 267
- menu mode • 267
- subschema, security • 267

O

- OLQBNOTE • 328, 331, 333
 - CMS • 331
 - z/OS • 328
 - z/VSE • 333
- option, system generation • 345
 - batch class • 345
 - comment character • 345
 - continuation character • 345
 - input lines • 345
 - interrupt • 345
 - menu mode access • 345
 - PF key module • 345
 - separator character • 345
 - sort storage • 345
- options, CA OLQ security • 298, 299, 300, 301
 - access • 298
 - access to • 299
 - establishing security • 301
 - interrupt processing • 300
 - passkeys • 301
 - qfile access • 299
 - security • 301
 - subschema access • 298
 - using IDD • 300
 - using LRF • 301

P

- PF key • 15, 29, 31, 53, 54, 132, 271
 - assignment • 271
 - CLEAR FUNCTION command • 53
 - comment • 31
 - COMMENT CHARACTER option • 31
 - continuation • 31
 - CONTINUATION CHARACTER option • 31
 - examples • 132
 - FUNCTION command • 132
 - invoking • 29
 - module • 271
 - separator • 31
 - SEPARATOR CHARACTER option • 31
 - table of defaults • 15
- PF key module • 271
 - how to create • 271

- sample • 271

Q

- qfile • 269, 270, 271, 272, 273, 275, 277, 279, 280
 - adding • 269
 - building • 269
 - defining • 273
 - executing • 269
 - maintaining • 269
 - modifying • 269
 - multiple parameter • 273
 - nesting • 272, 277
 - parameter • 272, 273, 277
 - reporting on • 270
 - signon • 272
 - signon profile • 271
 - special uses • 270

R

- REPEAT command • 97, 101, 103, 106, 114, 117, 121, 125
 - logical record • 97
 - MOST RECENT option • 101
 - OWNER WITHIN SET • 106
 - PHYSICAL SEQUENTIAL • 103
 - WITHIN DBKEYLIST • 114
 - WITHIN index SET • 117
 - WITHIN SET • 121
 - WITHIN SET USING SORTKEY • 125
- retrieval • 178
 - examples • 178
 - SELECT (IDMS mode) command • 178

S

- SAVE QFILE command • 165
 - examples • 165
- SAVE REPORT command • 166
 - examples • 166
- security • 297, 298, 299
 - accessing CA OLQ • 297
 - initiating security • 298
 - qfile access • 299
 - setting options • 298
- see=BYE command GOODBYE command • 53
- see=DEFINEPATH command path definition • 66
- see=HELP command SHOW command • 132
- see=individual function name built-in function • 220
 - invocation name • 220

see=IntegratedDataDictionary option • 349
see=LRF Logical Record Facility • 267
see=MATCHES.mask values • 43, 45, 46
see=MENUcommand SWAP command • 204
 examples • 204
see=SWAPcommand MENU command • 141
 examples • 141
see=systemgeneration option • 345
SELECT command • 168
 alternate column heading • 168
 alternate source name • 168
 ASCENDING option • 168
 column name • 168
 DESCENDING option • 168
 DISTINCT • 168
 examples • 168
 FROM record name • 168
 FROM table name • 168
 FROM view name • 168
 FROM view-id • 168
 GROUP BY clause • 168
 HAVING clause • 168
 nesting • 168
 ORDER BY clause • 168
 qualifying name • 168
 UNION operand • 168
 UNION option • 168
 WHERE criteria • 168
selection criteria • 35, 42, 48
 comparison expression • 35, 42
 field reference clause • 48
 FIND/GET WHERE clause • 42
 HAVING clause • 35
 SELECT WHERE clause • 35
 WHERE clause • 35, 42
signon profile qfile • 271, 272, 349
 how to create • 271
 security of • 349
 signon qfile • 272
signon qfile • 272
 how to create • 272
sort key • 125
 field • 125
 FIND/GET command • 125
 value • 125
SQL table • 70, 71
 name • 70

string built-in functions • 232, 233, 234, 235, 236,
237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
247, 248, 249, 250, 251, 252, 253, 255, 256, 257,
258, 259, 260, 261, 263
 FIX • 232
subschemas • 168
 joining tables and records from multiple • 168
 joining tables from multiple • 168
system generation • 345
 input lines, number of • 345
 menu mode, access to • 345
 PF key module • 345
 sort storage • 345

T

table • 69, 70
 name • 69
tables • 168, 177
 concatenating • 168, 177
 joining • 168
 joining from multiple subschemas • 168
 SELECT (IDMS mode) command • 177
 SELECT command • 177

Z

z/OS • 306, 307, 309, 310, 328
 JCL (central version) • 306, 307
 JCL (local mode) • 307, 309
 local mode considerations • 309, 310
 OLQBNOTE • 328
z/VSE • 313, 315, 317, 333
 JCL (central version) • 313, 315
 JCL (local mode) • 315, 317
 OLQBNOTE • 333