

CA-IDMS[®]

Features Guide
Release 14.0



Computer Associates

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

THIS DOCUMENTATION MAY NOT BE COPIED, TRANSFERRED, REPRODUCED, DISCLOSED, OR DUPLICATED, IN WHOLE OR IN PART, WITHOUT THE PRIOR WRITTEN CONSENT OF CA. THIS DOCUMENTATION IS PROPRIETARY INFORMATION OF CA AND PROTECTED BY THE COPYRIGHT LAWS OF THE UNITED STATES AND INTERNATIONAL TREATIES.

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 THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

THE USE OF ANY PRODUCT REFERENCED IN THIS DOCUMENTATION AND THIS DOCUMENTATION IS GOVERNED BY THE END USER'S APPLICABLE LICENSE AGREEMENT.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227.7013(c)(1)(ii) or applicable successor provisions.

First Edition, October 1996

One Computer Associates Plaza, Islandia, NY 11749
All rights reserved.

All trademarks, trade names, service marks, or logos referenced herein belong to their respective companies.

Contents

How to use this manual	vii
Chapter 1. About CA-IDMS Release 14.0	1-1
1.1 Overview	1-3
1.2 Release 14.0 objectives	1-4
1.3 Enhance performance and productivity	1-5
1.4 Exploit new hardware and software technology	1-7
1.4.1 Extended multitasking support	1-7
1.4.2 IBM Parallel Sysplex	1-7
1.5 Simple upgrade path	1-8
1.6 What's next	1-9
Chapter 2. Upgrading to Release 14.0	2-1
2.1 Overview	2-3
2.2 Installing the SVC under MVS	2-4
2.3 Recompiling the RHDCSRTT module	2-5
2.4 Extended multitasking support	2-6
2.5 TCE stack access	2-7
2.6 Catalog conversion for SQL definitions	2-8
2.6.1 Running the Catalog Conversion Utility	2-9
2.7 Performance Monitor map and record changes	2-10
2.8 Linking COBOL/370 programs	2-11
2.9 Review PROGRAM statement ERROR THRESHOLD parameter	2-12
2.10 Monitoring resource management control blocks	2-13
2.11 Dictionary record changes	2-14
2.12 Applying optional functionality	2-15
Chapter 3. IBM Parallel Sysplex Exploitation	3-1
3.1 Overview	3-3
3.2 About a Parallel Sysplex environment	3-4
3.3 Exploiting Parallel Sysplex functionality	3-5
3.4 Using dynamic database session routing	3-6
3.4.1 Planning to use dynamic database session routing	3-7
3.4.2 Implementing dynamic database session routing	3-7
3.4.2.1 Using groups	3-8
3.4.2.2 Backend CV definitions	3-8
3.4.2.3 Frontend CV definitions	3-9
3.4.2.4 Sample group definitions	3-11
3.4.3 Cloning backend CVs	3-12
3.4.3.1 Cloning CVs	3-12
3.4.4 Managing dynamic database session routing	3-14
3.4.4.1 Using the DCMT VARY DBGROUP command	3-14
3.4.5 Monitoring and tuning dynamic database session routing	3-15
3.4.5.1 Using DCMT DISPLAY DBGROUP	3-16
3.4.5.2 Using the Interval Monitor's DBGROUP Detail screen	3-17
3.5 Using shared cache	3-19
3.5.1 Defining shared cache in the Coupling Facility	3-20

3.5.2	Defining shared cache in CA-IDMS	3-21
3.5.2.1	Assigning files using DMCL file override	3-21
3.5.3	Usage	3-22
3.5.3.1	Assigning files to a shared cache dynamically	3-22
3.5.4	Monitoring shared cache	3-23
3.5.4.1	DCMT DISPLAY commands	3-23
3.5.4.2	CA-IDMS Performance Monitor	3-25
3.5.4.3	CA-IDMS System Statistics Report	3-28
3.5.5	Tuning a shared cache	3-28
Chapter 4.	CA-IDMS/DB and CA-IDMS/DC	4-1
4.1	Overview	4-3
4.2	Extended multitasking support	4-4
4.2.1	Implementing multitasking support	4-4
4.3	Extended 24-hour processing support	4-6
4.3.1	Dynamic lines, terminals, and printers	4-6
4.3.2	Dynamic security refresh	4-7
4.3.3	Dynamic resource allocation	4-8
4.4	Utility enhancements	4-11
4.4.1	SEGMENT support in BACKUP, RESTORE, and UNLOCK	4-11
4.4.1.1	BACKUP and RESTORE utility syntax	4-11
4.4.1.2	UNLOCK syntax	4-12
4.4.2	Enhanced area support in FORMAT JOURNAL	4-12
4.4.3	NonSQL data support in UPDATE STATISTICS	4-13
4.4.4	File support in PRINT SPACE	4-14
4.5	Date and Year 2000 support in DISPLAY/PUNCH statements	4-16
4.6	Security enhancements	4-17
4.6.1	Default signon and user ID options in RHDCSRTT	4-17
4.6.2	DISPLAY/PUNCH ALL syntax for security definitions	4-19
4.6.2.1	DISPLAY and PUNCH ALL statement syntax	4-20
4.6.3	Usage	4-23
4.6.4	Example	4-27
4.6.5	Verifying signons for APPC applications	4-28
4.7	Using LE/370-compliant language compilers with CA-IDMS/DC	4-30
4.7.1	Considerations about LE/370 runtime	4-30
4.7.2	Running LE/370-compliant compiler programs under CA-IDMS/DC	4-31
4.7.3	Supported LE/370 functions	4-32
4.7.4	Unsupported LE/370 functions	4-32
4.7.5	COBOL 370 support	4-32
4.8	IDMSIOX2 DB Exit	4-34
4.9	Enhancements to CICS-reentrant programs	4-39
Chapter 5.	CA-IDMS SQL Option	5-1
5.1	Overview	5-3
5.2	DISPLAY and PUNCH syntax	5-4
5.2.1	DISPLAY/PUNCH ALL statement	5-5
5.2.2	Usage	5-8
5.2.3	Example	5-12
5.2.4	DISPLAY/PUNCH ACCESS MODULE	5-12
5.2.5	DISPLAY/PUNCH CALC KEY	5-14
5.2.6	DISPLAY/PUNCH CONSTRAINT	5-15

5.2.7	DISPLAY/PUNCH INDEX	5-17
5.2.8	DISPLAY/PUNCH KEY	5-18
5.2.9	DISPLAY/PUNCH SCHEMA	5-20
5.2.10	DISPLAY/PUNCH TABLE	5-22
5.2.11	Usage	5-24
5.2.12	DISPLAY/PUNCH TABLE PROCEDURE	5-24
5.2.13	DISPLAY/PUNCH VIEW	5-26
5.2.14	Usage	5-28
5.3	Dynamic SQL syntax changes	5-29
5.3.1	Dynamic positioned UPDATE and DELETE	5-29
5.3.2	Dynamically-assigned names	5-29
5.3.3	Global statements and cursors	5-30
5.3.4	Dynamic parameters	5-31
5.3.5	Dynamic SQL statements and expressions	5-35
5.3.6	Expansion of cursor-name	5-35
5.3.7	Usage	5-36
5.3.8	Example	5-37
5.3.9	Expansion of cursor-specification	5-38
5.3.10	Usage	5-39
5.3.11	Example	5-40
5.3.12	Expansion of statement-name	5-41
5.3.13	Usage	5-41
5.3.14	Example	5-42
5.3.15	ALLOCATE CURSOR statement	5-43
5.3.16	Usage	5-43
5.3.17	Examples	5-43
5.3.18	CLOSE statement	5-44
5.3.19	Example	5-44
5.3.20	DEALLOCATE PREPARE statement	5-44
5.3.21	Usage	5-45
5.3.22	Examples	5-45
5.3.23	DELETE statement	5-45
5.3.24	Usage	5-46
5.3.25	Examples	5-47
5.3.26	DESCRIBE statement	5-47
5.3.27	Usage	5-48
5.3.28	EXECUTE statement	5-49
5.3.29	Usage	5-51
5.3.30	FETCH statement	5-51
5.3.31	OPEN statement	5-52
5.3.32	PREPARE statement	5-53
5.3.33	Usage	5-54
5.3.34	UPDATE statement	5-55
5.3.35	Usage	5-57
5.3.36	Examples	5-57
5.4	ALTER INDEX support	5-59
5.4.1	Usage	5-60
5.4.2	Example	5-60
5.5	Establishing default transaction options	5-61
5.6	SQLSTATE field in SQLCA	5-62

5.6.1 SQLSTATE values	5-62
5.6.2 SQLSTATE field placement in the SQLCA	5-65
5.7 Optimization enhancements	5-67
5.8 Migration of SQL syntax by CA-IDMS/Dictionary Migrator	5-68
Chapter 6. CA-ADS and the Mapping Facility	6-1
6.1 Overview	6-3
6.2 Concurrent checkout of maps and dialogs	6-4
6.3 Name and disable dialogs abending on program checks	6-5
6.4 Using IDD record syntax for tables and view	6-6
Chapter 7. CA-OLQ	7-1
7.1 Overview	7-3
7.2 Display report line length	7-4
7.3 HOME and LEFTMAX scroll commands	7-5
7.4 DISTINCT option in Menu Mode	7-6
7.5 Specify report column for aggregate columns	7-7
7.6 Override column length on columns with code tables	7-9
Chapter 8. CA-IDMS Tools	8-1
8.1 Overview	8-3
8.2 CA-IDMS/Dictionary Migrator	8-4
8.2.1 Migrating SQL Entities	8-4
8.2.1.1 Catalog Navigation Report	8-5
8.3 CA-IDMS/ADS Alive	8-6
8.4 CA-IDMS/DB Analyzer, CA-IDMS/DB Audit, CA-IDMS/DB Reorg	8-7
8.4.1 Additional CA-IDMS/DB Reorg features	8-7
8.4.2 Interface to DB/EZReorg	8-7
8.4.3 Specifying a blocking factor for work files	8-8
8.5 Year 2000 support	8-9
Index	X-1

How to use this manual

What this document is about

CA-IDMS® refers to the complete line of systems software products in the CA-IDMS product family. This document describes Release 14.0 features for these CA-IDMS products:

- CA-IDMS/DB
- CA-IDMS/DC
- CA-IDMS/UCF
- CA-IDMS SQL Option
- CA-ADS and the Mapping Facility
- CA-OLQ
- CA-IDMS Tools

The purpose of this document is to describe how to use the new features of Release 14.0.

Who should use this document

This document is for existing CA-IDMS DBAs, System Administrators, System Programmers, Security Administrators, Application Programmers, and CA-OLQ End Users who want to learn how to:

- Use the features of Release 14.0
- Upgrade to Release 14.0 from Release 12.01

This document assumes you have experience using CA-IDMS Release 12.01.

How this document is organized

This document presents Release 14.0 features by product. Each product or product group is addressed in a separate chapter as follows:

- **Chapter 1, “About CA-IDMS Release 14.0” on page 1-1,** — Describes the objectives for Release 14.0 and summarizes new features and enhancements to existing features.
- **Chapter 2, “Upgrading to Release 14.0” on page 2-1,** — Describes how to upgrade to Release 14.0 from Release 12.01.
- **Chapter 3, “IBM Parallel Sysplex Exploitation” on page 3-1,** — Provides an overview of how CA-IDMS exploits IBM's Parallel Sysplex to allow the dynamic routing of retrieval database sessions, the cloning of Central Versions, and the sharing of data in a global cache. These features are available if you use IBM Parallel Sysplex hardware and software. For a complete discussion of using Parallel Sysplex in your CA-IDMS environment, see the *CA-IDMS Parallel Sysplex User Guide*.
- **Chapter 4, “CA-IDMS/DB and CA-IDMS/DC” on page 4-1,** — Describes the new features of CA-IDMS/DB and CA-IDMS/DC, including CA-IDMS Security.
- **Chapter 5, “CA-IDMS SQL Option” on page 5-1,** — Describes the enhancements to the CA-IDMS SQL Option.
- **Chapter 6, “CA-ADS and the Mapping Facility” on page 6-1,** — Describes the enhancements to CA-ADS and the Mapping Facility.
- **Chapter 7, “CA-OLQ” on page 7-1,** — Presents the new features of CA-OLQ.
- **Chapter 8, “CA-IDMS Tools” on page 8-1,** — Describes enhancements to the CA-IDMS Tools, including the Dictionary Migrator.

Related documentation

New Release 14.0 document: In addition to this document, the *CA-IDMS Parallel Sysplex User Guide*, is available to support Release 14.0. It describes how to use CA-IDMS to exploit Parallel Sysplex functionality.

Existing CA-IDMS documents: Because some Release 14.0 features are extensions of existing CA-IDMS features, you may want to reference the following existing CA-IDMS Release 12.0 or 12.01 documents to find out more about a feature:

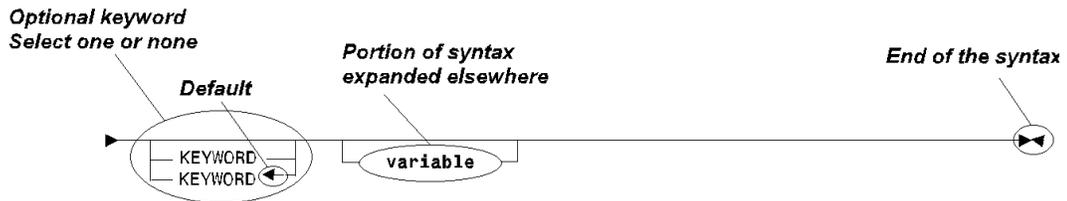
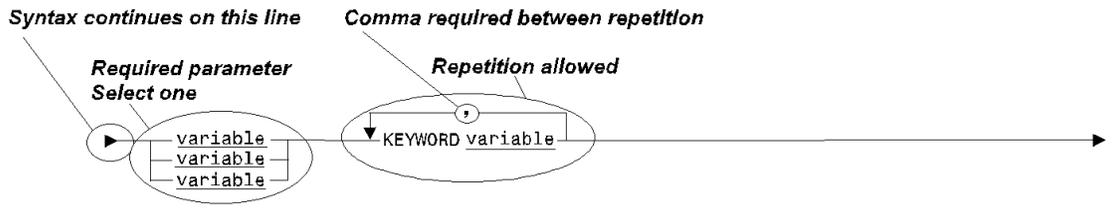
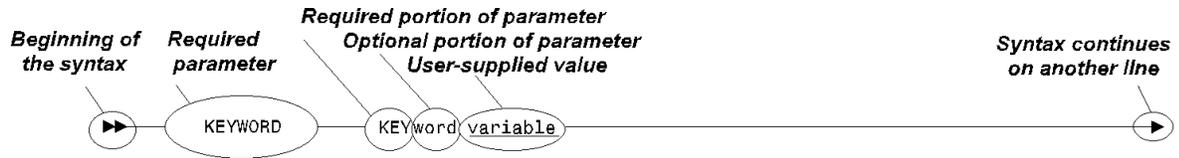
- *CA-IDMS System Operations*
- *CA-IDMS System Generation*
- *CA-IDMS System Tasks and Operator Commands*
- *CA-IDMS Security Administration*
- *CA-IDMS SQL Reference*
- *CA-ADS Reference*
- *CA-OLQ Online Menu Facility*
- *CA-IDMS/Dictionary Migrator User Guide*

Understanding syntax diagrams

Look at the list of notation conventions below to see how syntax is presented in this manual. The example following the list shows how the conventions are used.

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.
<u>underlined lowercase</u>	Represents a value that you supply.
←	Points to the default in a list of choices.
lowercase bold	Represents a portion of the syntax shown in greater detail at the end of the syntax or elsewhere in the document.
▶▶	Shows the beginning of a complete piece of syntax.
◀◀	Shows the end of a complete piece of syntax.
▶	Shows that the syntax continues on the next line.
▶	Shows that the syntax continues on this line.
▶	Shows that the parameter continues on the next line.
▶	Shows that a parameter continues on this line.
▶ parameter ▶	Shows a required parameter.
▶ parameter parameter ▶	Shows a choice of required parameters. You must select one.
▶ parameter ▶	Shows an optional parameter.
▶ parameter parameter ▶	Shows a choice of optional parameters. Select one or none.
▶ parameter ▶	Shows that you can repeat the parameter or specify more than one parameter.
▶ parameter , parameter ▶	Shows that you must enter a comma between repetitions of the parameter.

Sample syntax diagram



Chapter 1. About CA-IDMS Release 14.0

- 1.1 Overview 1-3
- 1.2 Release 14.0 objectives 1-4
- 1.3 Enhance performance and productivity 1-5
- 1.4 Exploit new hardware and software technology 1-7
 - 1.4.1 Extended multitasking support 1-7
 - 1.4.2 IBM Parallel Sysplex 1-7
- 1.5 Simple upgrade path 1-8
- 1.6 What's next 1-9

1.1 Overview

This chapter describes the objectives for CA-IDMS Release 14.0 and provides an overview of Release 14.0 features.

1.2 Release 14.0 objectives

The objectives for Release 14.0 are to:

- Enhance system performance and user productivity
- Exploit new and existing hardware and software technology
- Provide a simple upgrade path to the release

The features that support these objectives are discussed next.

1.3 Enhance performance and productivity

The following CA-IDMS products include new features that provide improved system performance and user productivity:

- **CA-IDMS/DB and CA-IDMS/DC** — More internal modules now run in multi-tasking mode to increase transaction throughput. Also, you can now make the following dynamic changes to a runtime environment without cycling a central version (CV):
 - Add or modify new line definitions and add new PTERM and LTERM definitions to new or existing lines
 - Change security definitions

Additionally, while a CV is active, CA-IDMS now dynamically allocates resource management control blocks when the initial allocation is depleted.

CA-IDMS Release 14.0 also provides for easier maintenance. Patch space has been added to each module, thereby reducing the dependence on the shared CSA for maintenance application. In addition, the need to apply optional APARs is dramatically decreased through the use of a new RHDCOPTF module.

For CA-IDMS utilities, you can now:

- Backup, restore, and unlock by segment
- Specify the number of areas a journal is required to handle
- Report on space usage by file
- Update statistics for nonSQL data

This release also lets you apply and remove optional APARs more easily as well as preserve them across maintenance updates.

- **CA-IDMS Security** — DISPLAY/PUNCH ALL syntax support is now available to make it easier to create and migrate security definitions and to report on existing security definitions.

For sites without an external security system, a new parameter on the #SECRIT macro allows you to extract a user ID, which is used to sign on to all startup and shutdown autotasks.

For APPC applications, this release provides a system generation feature that checks whether a PTERM supports an *already verified* user ID; that is, a user ID that has already been verified by the requesting system.

- **CA-IDMS SQL option** — DISPLAY and PUNCH syntax support is now available for logical entities to make it easier to create and migrate definitions and to report on existing SQL definitions. Also, the SQLSTATE field is now in the SQLCA to enhance application portability.

Additional features for the SQL Option include:

- Support for both dynamic parameters and dynamic positioned updates and deletes

- ALTER INDEX support
- Ability to collect statistics about nonSQL-defined data with the UPDATE STATISTICS utility
- The SET SESSION statement, which lets you establish default transaction options for an SQL session.
- **CA-ADS and the Mapping Facility** — These two new features are available, which make it easier to use CA-ADS and Mapping:
 - The checkout facility for maps and dialogs now recognizes dictionary name as a qualifier.
 - Dialogs abending with a program check are individually reported and disabled. That is, the entire ADS runtime environment is not disabled if a dialog abends excessively.
- **CA-OLQ** — Several new features are now available in Menu Mode to simplify the viewing and formatting of reports.
- **CA-IDMS Tools** — The CA-IDMS/Dictionary Migrator product now supports logical SQL definitions, and the CA-IDMS/DB ANALYZER, CA-IDMS/DB AUDIT, and CA-IDMS/DB REORG products now use all features of CA-IDMS database I/O processing; such as, ESA dataspace and XA database buffers. Also, all tools provide year 2000 support.

1.4 Exploit new hardware and software technology

In Release 14.0, CA-IDMS continues to exploit new and existing hardware and software technology to provide enhanced system performance and user productivity. Multitasking support is extended to include more system tasks and CA-IDMS systems can exploit IBM's Parallel Sysplex.

1.4.1 Extended multitasking support

More CA-IDMS system tasks can execute concurrently when running in MVS, MSP, and BS2000 environments for increased transaction throughput. For more information, see Chapter 4, “CA-IDMS/DB and CA-IDMS/DC” on page 4-1.

1.4.2 IBM Parallel Sysplex

Features: CA-IDMS Release 14.0 exploits IBM's Parallel Sysplex hardware and software to provide these features:

- Dynamic routing of retrieval database sessions
- Cloned central versions (CVs)
- Shared database buffers across multiple CVs

Benefits: CA-IDMS systems that use these features in a Sysplex can realize these benefits:

- Continuous availability of systems
- Dynamic workload balancing
- Increased transaction throughput
- Improved response time during heavy system load
- Easier management of multiple systems
- Less database I/O when using a shared cache

For complete information, see Chapter 3, “IBM Parallel Sysplex Exploitation” on page 3-1.

1.5 Simple upgrade path

To upgrade to Release 14.0, you simply install CA-IDMS software from the Release 14.0 installation tape. If you have SQL definitions stored in your catalog, then you must run a catalog conversion utility to convert your 12.01 catalog(s) to run in Release 14.0. Also, if you use the CA-IDMS Performance Monitor, you should initialize the log area.

You must also install a new SVC and reassemble the RHDCSRTT module to include the new SVC number. Because DSECTs for this release have changed, you should recompile Assembler programs and exits that reference DSECTs. If you use multi-tasking, you must examine user exits and database procedures for compliance.

Upgrade requirements are feature-dependent: Additional upgrade requirements are dependent upon the new features you want to use.

You don't need to make any changes to use some of the new features. For example, CA-IDMS dynamically allocates resource management control blocks when the initial allocation is depleted.

For a complete discussion of upgrade requirements, see Chapter 2, "Upgrading to Release 14.0" on page 2-1.

1.6 What's next

If you will use CA-IDMS in IBM's Parallel Sysplex, you can begin by reading Chapter 3, "IBM Parallel Sysplex Exploitation" on page 3-1; otherwise, begin by reading Chapter 4, "CA-IDMS/DB and CA-IDMS/DC" on page 4-1, to learn about new features in CA-IDMS/DB and CA-IDMS/DC.

Chapter 2. Upgrading to Release 14.0

2.1 Overview	2-3
2.2 Installing the SVC under MVS	2-4
2.3 Recompiling the RHDCSRTT module	2-5
2.4 Extended multitasking support	2-6
2.5 TCE stack access	2-7
2.6 Catalog conversion for SQL definitions	2-8
2.6.1 Running the Catalog Conversion Utility	2-9
2.7 Performance Monitor map and record changes	2-10
2.8 Linking COBOL/370 programs	2-11
2.9 Review PROGRAM statement ERROR THRESHOLD parameter	2-12
2.10 Monitoring resource management control blocks	2-13
2.11 Dictionary record changes	2-14
2.12 Applying optional functionality	2-15

2.1 Overview

This chapter describes tasks that may be necessary to upgrade to CA-IDMS Release 14.0.

The following topics are presented:

- Installing the SVC under MVS
- Recompiling the RHDCSRTT module
- Extended multitasking support
- TCE stack access
- Catalog conversion for SQL definitions
- Performance Monitor record and map changes
- Linking COBOL/370 programs
- Review ERROR THRESHOLD for CA-ADS dialogs
- Monitoring resource management control blocks
- Dictionary record changes

Note: Migration of the DDLDDL dictionary area is *not* required,

- Applying optional functionality

2.2 Installing the SVC under MVS

You *must* change the CAIRIM parameter for loading your SVC. The INIT parameter *must* specify GJE0INIT as the program to be executed.

GJE0INIT loads the SVC, RHDCSSFM, and CAS9SEC (a CA90s module) programs. These programs must be in a library available to CAIRIM. To simplify this process, the CA-IDMS install process allocates a new SMP/E target load library with a low-level qualifier of APFLIB. This library contains GJE0INIT, RHDCSSFM, and the SVC. Since most sites copy the SVC modules to a protected area, this new library was created to let you copy the contents of the library without having to decide what members are needed.

In addition, you must specify an SVC number in the RHDCSRTT module. The default RHDCSRTT module, with no security enabled, is assembled during the install process, using the SVC number specified in the installation parameters.

2.3 Recompiling the RHDCSRTT module

The code generated by the #SECR TT macro is extended for Release 14; therefore, you must recompile the RHDCSRTT module with the Release 14.0 macro library. Be sure that you add the SVC number to the #SECR TT macro before you recompile the RHDCSRTT module. The source to be used is in your PPOPTION file.

Possible error conditions: You may encounter these error conditions if you specify an incorrect SVC:

- If the RHDCSRTT module does not point to an SVC, all jobs will receive an S200 abend.
- If the RHDCSRTT module points to an invalid SVC, batch jobs will receive an SFxx abend, where xx is the hexadecimal equivalent of the SVC number.
- If your CA-IDMS central version does not point to the correct SVC, you will receive an SFxx abend.

2.4 Extended multitasking support

Because the major database component modules now run in a multitasking environment (i.e., with an MPMODE=ANY), you must verify that any database procedures and user exits 14, 15, 23, 27, 28, 29, and 31 running in your environment can run with an MPMODE of ANY.

Note: We recommend that you write DB procedures in assembly language to run in a multitasking environment. The DB procedures must also be reentrant.

►►See *CA-IDMS System Operations* for specific information on writing user exits that can run with an MPMODE of ANY. See *CA-IDMS Database Administration* for specific information about writing database procedures with an MPMODE of ANY.

2.5 TCE stack access

The Task Control Element (TCE) stack now resides in XA storage. The TCE stack is used internally by CA-IDMS as a save area. This frees up non XA storage for client applications.

If you have user exits that access the TCE stack, you must change them to run in 31-bit mode.

2.6 Catalog conversion for SQL definitions

If the SQL Option was previously installed, you must run the catalog conversion utility against each 12.0 catalog in your environment, after you install CA-IDMS Release 14.0.

CAUTION:

Be sure to backup your existing catalogs before you convert them. The converted catalogs are not downward compatible with Release 12.0 or 12.01.

You must run the catalog conversion utility from the Command Facility on your Release 14.0 system. The catalog conversion utility performs the following tasks against your 12.0 catalogs:

- Rebuilds all SYSTEM.SYNTAX table rows associated with table definitions containing CHECK CONSTRAINTS and all view definitions to eliminate the maintenance of redundant syntax and to conform to the ANSI standard on catalog definitions.
- Adjusts existing table definitions in the following tables to include columns described in some DSECTs that are not currently defined:
 - SYSTEM.COLUMN
 - SYSTEM.DBNAME
 - SYSTEM.DBTABLE
 - SYSTEM.DMCLFILE
 - SYSTEM.INDEX
 - SYSTEM.SCHEMA
 - SYSTEM.TABLE
- Adjusts existing CALC key table rows to initialize columns described by the SYSTEM.INDEX table definition.

Rebuilds syntax in SYSTEM.SYNTAX table: Consistent with the catalog definition in the ANSI standard, the SYSTEM.SYNTAX table now contains the SELECT statement for a view and only the CHECK CONSTRAINT syntax for tables containing them. Previously, the complete syntax for a CREATE and ALTER TABLE statement that included a CHECK CONSTRAINT and the complete syntax for a CREATE VIEW statement was stored in the SYSTEM.SYNTAX table.

The catalog conversion utility rebuilds the rows in the SYSTEM.SYNTAX table containing CHECK CONSTRAINT syntax and all VIEW statements. This should significantly reduce the number of rows stored in the SYSTEM.SYNTAX table.

Note: To see the complete syntax for a table definition, use the DISPLAY TABLE *table-name* or DISPLAY ALL TABLE syntax, which is new in Release 14 and discussed in Chapter 5, “CA-IDMS SQL Option” on page 5-1.

2.6.1 Running the Catalog Conversion Utility

The catalog conversion utility is installed in the Release 14.0 load library.

Back up the DDLCAT area: The catalog conversion utility reformats tables in the DDLCAT area. It does not change any other area in the catalog. Before you run it, back up the DDLCAT area in each 12.0 catalog you will convert.

Running the Catalog Conversion Utility: You run the catalog conversion utility from the Release 14.0 Batch Command Facility using this statement:

```
►►— CONVERT CATALOG —————►►
```

After a successful run of the Catalog conversion utility, the Command Facility issues one of two informational messages to indicate the status of the conversion.

- If a catalog conversion is performed, the message indicates the type and number of each type of occurrences converted as shown below:

```
DB002900 Catalog Conversion Completed CALC KEYS n TABLES n SYNTAX n
```

The type of information displayed will vary depending on the contents of the catalog.

- If a conversion was not required, the following message is displayed:

```
DB002900 Catalog Conversion Completed - NO CONVERSION REQUIRED
```

2.7 Performance Monitor map and record changes

Record changes: The Performance Monitor records listed below contain additional fields in this release. Some of them have also been split up. If you have user-written programs that use these records, you need to recompile them with the Release 14.0 macro library. Programs that use records that have been split will have to be adapted.

#PMARADS (PMIM area wait)	This record has also been split into two parts.
#PMBUFDS (PMIM buffer wait)	
#PMBKDS (PMIM db-key wait)	
#PMHDRDS (Performance Monitor record header)	
#PMINTDS (PMIM interval wait summary)	
#PMJRLDS (PMIM journal wait)	
#PMRUSDS (PMIM run units information)	
#PMSMHDS (SMF header)	
#PMSTLDS (DC log record data)	
#PMTASDS (PMAM task)	This record has also been split into three parts (instead of two).
#PMTAWDS (PMAM task wait)	
#PMDBGDS (PMIM DBGroup wait)	This record is new for release 14.0

Map changes: The Performance Monitor screens listed below contain additional fields in this release. All screens in the list that were edited and saved in your 12.01 dictionary load area will have to be recompiled for Release 14.0.

PMAMMBST	: PMAM DB Statistics screen
PMAMMTKL	: PMAM Task List screen
PMIMMARE	: PMIM Area Detail screen
PMIMMBUF	: PMIM Buffer Detail screen
PMIMMDBK	: PMIM DBkey/Area Detail screen
PMIMMIO	: PMIM IO Detail screen
PMIMMMNU	: PMIM Menu Interval Monitor screen
PMIMMRUN	: PMIM Transaction Statistics screen
PMIMMSRU	: PMIM Specific Transaction Information screen
PMRTMBUF	: PMRM Buffer I/O Summary screen
PMRTMIOD	: PMRM Database I/O Driver Detail screen
PMRTMLTR	: PMRM Lterm Resource Usage Summary screen
PMRTMRUS	: PMRM Transaction Detail screen
PMRTMSBF	: PMRM Buffer I/O Detail screen
PMRTMSCR	: PMRM Scratch Manager Detail screen
PMRTMSPE	: PMRM Specific System Transaction Detail screen
PMRTMSQA	: PMRM SQL Detail screen
PMRTMSTK	: PMRM Active System Task Detail screen
PMRTMTKO	: PMRM Task + Prog Pool Overview screen
PMRTMTSK	: PMRM Active User Task Detail screen

The following Performance Monitor screens are new for Release 14.0.

PMIMMDG1	: PMIM DBGroup Detail screen
PMIMMDG2	: PMIM DBGroup Node Detail screen
PMIMMSH1	: PMIM Shared Cache Detail screen
PMIMMSH2	: PMIM Shared Cache Files Detail screen

2.8 Linking COBOL/370 programs

Any programs compiled by IBM COBOL/370 compilers for use with CA-IDMS/DC should be relinked using the procedure described in Chapter 4, “CA-IDMS/DB and CA-IDMS/DC” on page 4-1.

2.9 Review PROGRAM statement ERROR THRESHOLD parameter

Now that dialogs abending on program checks are disabled and reported, you need to examine and, if necessary, adjust the ERROR THRESHOLD parameter on the PROGRAM statement for CA-ADS dialogs.

If you want to enable error thresholds for ADSOMAIN and ADSORUN1 for abends caused by internal errors, issue a DCMT command after the system is started to change the error threshold to a nonzero value.

2.10 Monitoring resource management control blocks

With the automatic secondary allocation of resource management control blocks, you may want to monitor the usage of resource management control blocks and, if necessary, change the values defined in your system definition for the following parameters of the SYSTEM statement:

- DPE COUNT
- RCE COUNT
- RLE COUNT

Monitoring tools: Using the OPER WATCH CRITICAL RESOURCES command, the DCMT DISPLAY STATISTICS SYSTEM command, or the Real Time Monitor's Storage Pool Overview screen monitor the number of DPEs, RCEs, and RLEs your system is using and the number of times it has issued a secondary allocation of them. Additionally, look for the DC010007 message in the CV's SYSOUT file or the job console log file for the number of DPEs, RCEs, and RLEs used in secondary allocations.

For sample screens showing these statistics, see Chapter 4, “CA-IDMS/DB and CA-IDMS/DC” on page 4-1.

What to modify: After monitoring the use of resource management control blocks, you may want to modify relevant system generation parameters. As a guideline, set the value of the DPE COUNT, RCE COUNT, and RLE COUNT at the value of the high water mark (HWM) displayed in response to the OPER WATCH CRITICAL RESOURCES command. The minimum sysgen'd count for these control blocks must be sufficient to start the system. See *CA-IDMS System Generation* for more information on these parameters.

2.11 Dictionary record changes

The dictionary records listed below have changed in Release 14.0:

- CVGDEFS-142
- DBNAME-1031
- DBTABLE-1034
- DMCLFILE-1037
- MODCMT-084
- NAMEDES-186
- QUEUE-DCQ-138
- SA-018
- SCR-054
- SRCD-113
- SDES-044
- SMR-052

If you have user-written programs that access these records, you may want to run the IDMSDIRL utility to include the new fields in your Release 14.0 dictionary environment. This step is optional. You are only required to run IDMSDIRL if you want to access the new fields by name in the updated records. The changes to the dictionary records have all been to replace FILLER fields with data-bearing fields. There are no structural changes to any dictionary records.

2.12 Applying optional functionality

Before this release, optional functionality was made available through the use of optional APARs. This release lets you apply and remove optional functionality more easily as well as preserve it across maintenance updates. Optional APARs from previous releases that do not need to change any specific values within IDMS can be activated by setting one bit in an optional functionality bitmap table; other optional APARs will still be applied as program modifications. The bitmap table is generated in a new RHDCOPTF module that is loaded during central version or mini-cv startup processing, and anchored in the CSA.

To create a new RHDCOPTF module, assemble and link a RHDCOPTF source module that contains #DEFOPTF macros that will activate optional functionality. The last #DEFOPTF macro has to contain the TYPE=GENERATE parameter in order to generate the code.

#DEFOPTF macro: #DEFOPTF accepts two parameters:

- One is positional and can be an internal function number, in the form OPTnnnnn, or a list of internal function numbers, allowing functions to be grouped by subject or module in one macro. An internal function number will be associated with each optional function that can be activated this way.
- The second parameter is the TYPE=keyword that can get the value DEFINE, which is the default, or GENERATE, in the last input macro.

Example

```
TITLE 'User optional bitmap table'
#DEFOPTF OPT00002
#DEFOPTF OPT00010,OPT00011
#DEFOPTF (OPT00020,OPT00021,OPT00022)
#DEFOPTF TYPE=GENERATE
```

The following list shows the association between the CA-IDMS 12.x optional APARs and the CA-IDMS 14.0 internal function numbers. Any of the internal function numbers may be used with the OPT prefix as a parameter to the #DEFOPTF macro to activate the functionality of the corresponding Release 12.x optional APAR.

CA-IDMS 12.x Product/Problem	CA-IDMS 12.x Optional APAR	CA-IDMS 14.0 Function Number
COMPLR 0023	CS74316	00001
IDMSDC 0051	CS76640	00002
IDMSDC 0030	CS76649	00003
IDMS 0173	CS81862	00005
MAPS 0023	CS81905	00006

CA-IDMS 12.x Product/Problem	CA-IDMS 12.x Optional APAR	CA-IDMS 14.0 Function Number
MAPS 0024	CS81906	00007
MAPS 0025	CS81907	00008
OLQ 0013	CS82062	00009
IDMSDC 0079	CS82065	00010
ADS370 0044	CS82230	00011
OLQ 0014	CS82257	00012
OLQ 0026	CS88588	00016
ADS370 0033	CS88593	00017
PERF 0004	CS89116	00018
IDMSDC 0106	CS89183	00020
OLQ 0016	CS90910	00021
MAPS 0058	CS90975	00022
IDMS 0364	CS97386	00025
OLQ 0030	CS98815	00027
OLQ 0031	CS98817	00028
OLQ 0034	CS98820	00029
OLQ 0035	CS98821	00030
ADS370 0094	CS98971	00031
OLQ 0036	GS03086	00033
PERF 0027	GS06142	00034
IDMSDC 0343	GS08592	00038
IDMS 0478	GS08798	00039
ADS370 0144	GS09888	00042
ADS370 0149	GS09891	00043
MAPS 0130	GS09894	00044
IDMSDC 0444	GS18646	00047
IDMSDC 0490	GS19348	00049
IDMSDC 0508	GS19358	00050
IDMSDC 0492	GS20620	00051
IDMSDC 0510	GS20621	00052

CA-IDMS 12.x Product/Problem	CA-IDMS 12.x Optional APAR	CA-IDMS 14.0 Function Number
IDMS 0741	GS23943	00054
IDMS 0787	GS26902	00058
IDMSDC 0546	GS27074	00059
IDMS 0773	GS27455	00061
MAPS 0161	GS29244	00067
MAPS 0207	GS29265	00068
OLQ 0045	GS29271	00069
MAPS 0206	GS29594	00070
ADS370 0178	GS29598	00071
ADS370 0210	GS29600	00072
ADS370 0183	GS30903	00073
OLQ 0055	GS33436	00074
OLQ 0057	GS33439	00075
ADS370 0255	GS33451	00076
ADS370 0097	GS33477	00077
OLQ 0046	GS33482	00078
IDMSDC 0702	GS36402	00079
COMPLR 0280	GS37899	00080
IDMS 0989	GS37943	00081
IDMS 0993	GS37957	00083
OLQ 0070	GS39784	00085
ADSB 0208	GS39792	00086
ADS370 0208	GS39794	00087
ADS370 0172	GS40519	00089
OLQ 0060	GS40553	00091
MAPS 0234	GS40570	00092
IDMS 1116	GS41974	00093
ADS370 0266	GS42284	00094
OLQ 0069	GS42557	00095
OLQ 0071	GS42558	00096

CA-IDMS 12.x Product/Problem	CA-IDMS 12.x Optional APAR	CA-IDMS 14.0 Function Number
OLQ 0072	GS42565	00097
OLQ 0073	GS42567	00098
OLQ 0074	GS42573	00099
OLQ 0081	GS42915	00100
IDMSDC 0829	GS44049	00102
IDMSDC 0852	GS48947	00108
ADS370 0339	GS49031	00111
MAPS 9999	GS53962	00112
IDMSDC 0738	GS53981	00114
OLQ 0093	GS59938	00115
OLQ 0097	GS59939	00116
IDMS 1302	GS77233	00118
IDMS 1358	GS77552	00119
IDMSDC 1034	GS81227	00121
IDMS 1456	GS81238	00122
ADS370 0401	GS81410	00124
ICMS 0009	GS81866	00125
ICMS 0010	GS81870	00126
IDMS 1298	GS82027	00127
OLQ 0100	GS82047	00128
OLQ 0108	GS82057	00129
IDMS 1418	GS82063	00130
OLQ 0110	GS82126	00131
IDMS 1318	GS82179	00132
ICMS 0008	GS82239	00134
IDMS 1477	GS82587	00135
IDMS 1527	GS87630	00136
IDMSDC 0882	GS90621	00139
IDMSDC 1138	GS91059	00140
IDMSDC 1021	GS91072	00141

CA-IDMS 12.x Product/Problem	CA-IDMS 12.x Optional APAR	CA-IDMS 14.0 Function Number
OLQ 0112	GS96457	00142
IDMSDC 1223	GS96487	00145
OLQ 0120	GS96940	00146
IDMS 1613	GS97248	00148
PERF 0041	GS32791	00163

Chapter 3. IBM Parallel Sysplex Exploitation

3.1 Overview	3-3
3.2 About a Parallel Sysplex environment	3-4
3.3 Exploiting Parallel Sysplex functionality	3-5
3.4 Using dynamic database session routing	3-6
3.4.1 Planning to use dynamic database session routing	3-7
3.4.2 Implementing dynamic database session routing	3-7
3.4.2.1 Using groups	3-8
3.4.2.2 Backend CV definitions	3-8
3.4.2.3 Frontend CV definitions	3-9
3.4.2.4 Sample group definitions	3-11
3.4.3 Cloning backend CVs	3-12
3.4.3.1 Cloning CVs	3-12
3.4.4 Managing dynamic database session routing	3-14
3.4.4.1 Using the DCMT VARY DBGROUP command	3-14
3.4.5 Monitoring and tuning dynamic database session routing	3-15
3.4.5.1 Using DCMT DISPLAY DBGROUP	3-16
3.4.5.2 Using the Interval Monitor's DBGROUP Detail screen	3-17
3.5 Using shared cache	3-19
3.5.1 Defining shared cache in the Coupling Facility	3-20
3.5.2 Defining shared cache in CA-IDMS	3-21
3.5.2.1 Assigning files using DMCL file override	3-21
3.5.3 Usage	3-22
3.5.3.1 Assigning files to a shared cache dynamically	3-22
3.5.4 Monitoring shared cache	3-23
3.5.4.1 DCMT DISPLAY commands	3-23
3.5.4.2 CA-IDMS Performance Monitor	3-25
3.5.4.3 CA-IDMS System Statistics Report	3-28
3.5.5 Tuning a shared cache	3-28

3.1 Overview

CA-IDMS Release 14.0 includes new features that work with the parallel processing and shared cache components of IBM's Parallel Sysplex to provide:

- Increased transaction throughput
- Improved response time during heavy system load
- Dynamic workload balancing to use system resources more effectively
- Cloned central versions (CVs), which you can start and stop in response to changing processing requirements to provide continuous system availability
- Shared data with data integrity across multiple systems to minimize the number of disk I/O's to the database

This chapter briefly describes how CA-IDMS exploits these Parallel Sysplex components. For detailed information on how to use Parallel Sysplex components with CA-IDMS, see the *CA-IDMS Parallel Sysplex User Guide*.

3.2 About a Parallel Sysplex environment

A Parallel Sysplex is a multisystem environment that acts like a single system. It's a combination of hardware and software products that cooperate to process work without interruption.

CA-IDMS exploits these parallel processing and Coupling Facility components of IBM's Parallel Sysplex:

- **Parallel processing** — Parallel processing in a Sysplex allows several copies of the MVS/ESA operating system to process work concurrently. For example, multiple applications requesting retrieval access to the same CA-IDMS database can be processed concurrently on different processors using dynamic workload balancing.
- **Coupling Facility** — The Coupling Facility is a dedicated processor in the Sysplex that is connected to other processors in the Sysplex using high speed fiber optic Coupling Facility Links. It allows high speed access to shared data across applications running on different systems in a Sysplex. For example, CA-IDMS exploits the shared cache feature of the Coupling Facility by allowing multiple CVs running in a Sysplex to share global database buffers.

For more information on these components and Sysplex, see the appropriate documentation for IBM's MVS/ESA SP5.

3.3 Exploiting Parallel Sysplex functionality

CA-IDMS uses the parallel processing and Coupling Facility components of IBM's Parallel Sysplex to provide these features:

- **Dynamic routing of database sessions** — A retrieval database session can be dynamically routed to a CV in the Sysplex that has the CPU cycles available to process it. Using this feature, workload balancing is dynamic, which provides improved performance and more effective use of system resources. CA-IDMS environments that are configured with frontend and backend CVs are ideally suited to take advantage of this feature.
- **Cloning CVs** — With simple changes to existing system definitions and startup JCL, you can clone CVs and run them simultaneously in a Sysplex. Using this feature, you can respond quickly to changes in your processing requirements by making more backend systems available to service database transactions and improve transaction throughput and response time. This feature is designed to work in a Sysplex with the dynamic routing of database sessions feature.
- **Shared cache** — Multiple CVs can share global database buffers by accessing a shared cache in the Coupling Facility. CA-IDMS maintains the consistency of data across CVs participating in the cache. Using this feature, the number of physical I/Os to a database is reduced by accessing the shared cache instead of disk.

Each of these features is discussed in separate sections below.

3.4 Using dynamic database session routing

What is dynamic database session routing: The CA-IDMS client/server communications architecture for database sessions is extended to allow a frontend CV to route a retrieval database session to a backend CV that is identified, at run time, to have the CPU cycles available to service the session. The database session can be an SQL transaction or a run unit.

In a non-Sysplex environment, the routing of retrieval database sessions is "static;" database sessions are routed to an explicit, predetermined CV regardless of its availability.

Benefits: Dynamic database session routing provides these benefits:

- **Dynamic workload balancing** — Workload balancing is dynamic and based on actual system load and resource availability; you don't need to predetermine database routing to balance workloads across CVs to get maximum throughput and shorter response time. Retrieval database sessions are routed to a CV with available processing cycles.
- **Parallel processing of retrieval database sessions** — Retrieval database sessions are processed in parallel to reduce elapsed processing time.
- **Automatic routing of retrieval database sessions to an available CV** — Retrieval database sessions are routed to an available CV instead of being routed to the same system whether it's available or not.
- **Use with cloned CVs to make multiple copies of systems available** — You can implement dynamic database session routing with the cloned CV feature to start and stop systems in response to changes in workloads to increase transaction throughput and decrease response time.

Use with CV cloning: This feature is designed to work in a Sysplex with the dynamic routing of database sessions feature so that retrieval database sessions can be dynamically routed to cloned copies of the same CV. See Cloning backend CVs later in this chapter for more information on cloning CVs.

Use with shared cache: You can also use the dynamic database session routing feature with the shared cache feature to share database buffers across multiple CVs. Using the shared cache feature allows CVs to share buffers, minimizes I/O operations to disk, and keeps data current across the retrieval and update CVs running in a Sysplex. See Using shared cache later in this chapter for more information on the shared cache feature.

3.4.1 Planning to use dynamic database session routing

Dynamic routing of retrieval database sessions is designed to work with CA-IDMS environments configured with the following minimum requirements:

- The ability to identify database sessions that perform only retrieval processing against a database. Typically you do this for database run units by assigning a retrieval-only subschema to retrieval run units. Since updates to an area can be performed by only one CV at a time, only retrieval database sessions are available for dynamic routing.
- CVs set up to process terminal and application services separate from database services. This is accomplished by defining frontend CVs to process terminal activities and other application-specific services, and backend CVs to process database requests. Using this configuration, frontend CVs can route database requests to an available backend CV where requests can then be serviced.
- Backend CVs set up to process retrieval-only database sessions separate from update sessions. With this set up, multiple CVs can process retrieval requests for a database while one designated CV can process update requests at the same time.

If your current CA-IDMS environment is configured as described above, you need to make a few simple changes to use dynamic database session routing; otherwise, you need to set up frontend and backend CVs to use dynamic database session routing in a Sysplex. For more information on configuring frontend and backend CVs, see *CA-IDMS System Operations*.

3.4.2 Implementing dynamic database session routing

To implement dynamic database session routing, you assign the backend CVs that will service retrieval database sessions to one or more groups. You also define these groups on the resource name table of each frontend CV that can route requests to them. At run time, database sessions are routed to a group to determine which CV in the group has the CPU cycles available to process the request. Once a CV volunteers to service the request, the request is directed to it and it processes the request on the identified node.

Using cloned backend CVs: To exploit the parallel processing and dynamic workload balancing features of dynamic database session routing, you must have enough backend retrieval CVs available to process any run unit or SQL transaction that may be dynamically routed to them. You can use the CV cloning feature, discussed below, to make multiple copies of retrieval CVs available.

This section describes how to set up your CA-IDMS environment to use dynamic database session routing, beginning with the concept of a group and how it is used to implement dynamic database session routing.

3.4.2.1 Using groups

What is a group A group contains one or more CVs. You assign CVs to groups based on the databases they service. At run time, a request for access to a database is routed to a group, instead of a node. The group name is then replaced with the node name for a CV assigned to it that volunteers (i.e. has the CPU cycles) to service the request.

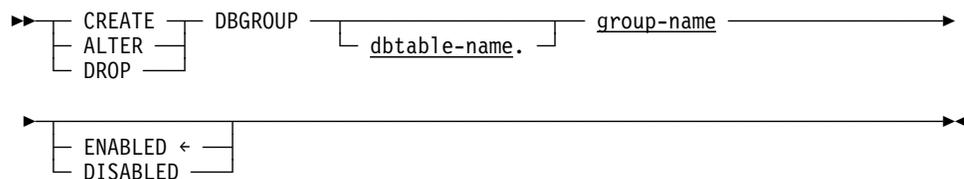
Planning groups: Before you define any groups, you need to plan how you want to group CVs to meet your processing requirements. For example, you can create one group for all backend CVs, if all databases are accessible by all CVs.

3.4.2.2 Backend CV definitions

Assign backend CVs to groups: To assign each backend CV to one or more groups, you update its database name table by adding a DBGROUP statement for each group in which it will participate. This makes the CV a member of the specified group. Be sure that you assign all backend CVs to all groups in which they will participate. When each backend CV is started, it is eligible to process dynamic retrieval database sessions for the databases defined in its database name table.

Note: You can dynamically assign a CV to a group using the DCMT VARY DBGROUP *group-name* JOIN command. See Managing dynamic database session routing, later in this chapter for more information.

DBGROUP syntax: The syntax for the CREATE DBGROUP statement is shown below.



Parameters

Group-name

Specifies the name of a group in which the CV will participate.

Group-name is a 1- through 8-character name that conforms to the naming conventions for node names defined in a system definition. The *group-name* is also added as a node name on the resource name table of each frontend CV that can route requests to it. See *CA-IDMS System Generation* for specific information on defining node names.

Enabled

Specifies that the CV will be an active member of the group and available to process dynamic retrieval database sessions when the CV is started.

Once the CV is started, if you need to disable its active participation in the group, you can do so dynamically using the DCMT VARY DBGROUP command, which

is described in the Managing dynamic database session routing section later in this chapter.

Disabled

Specifies that the CV will not be an active member of the group and will not be eligible to process database sessions dynamically when the CV is started.

After the CV is started, you can dynamically enable its active participation in the group using the DCMT VARY DBGROUP command, which is described in the Managing dynamic database session routing section later in this chapter.

Sample group definitions: The following example shows a DBGROUP definition for the EMPGROUP group. EMPGROUP is a group to which CV IDMS071 is assigned.

Group definition in database name table for IDMS071 CV

```
CREATE DBGROUP SYS71TBL.EMPGROUP  
ENABLED;
```

```
GENERATE DBTABLE SYS71TBL;
```

3.4.2.3 Frontend CV definitions

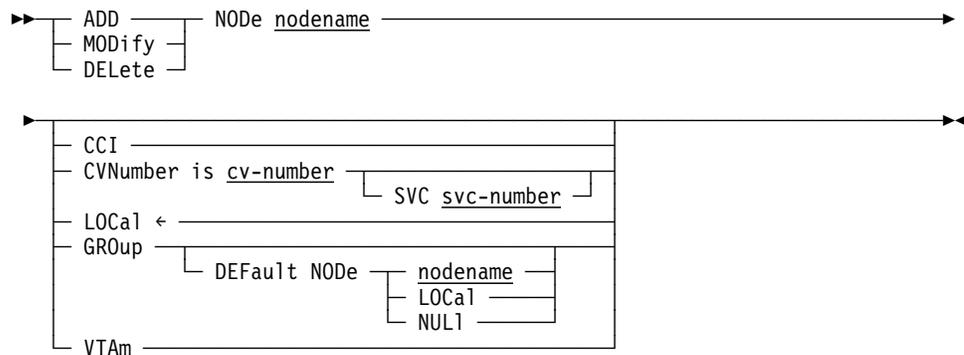
In dynamic database session processing, a frontend CV routes requests for database services to groups. To do this, it must know about the groups to which it can route requests. Groups are defined as nodes in a frontend CV's resource name table using the NODE statement.

Note: Applications executing from CICS, batch, Windows, and any other non CA-IDMS/DC client must first be routed to a specific frontend CV, which can then dynamically route it to a backend CV for servicing.

Adding groups to a resource name table: The resource name table for each frontend CV must contain *both* of the following:

- The name of each group to which it can route requests. The group name is added as a node name on the system generation NODE statement with a type of GROUP.
- For each backend CV that can service requests for a group, a NODE entry specifying the communication method to use to access it.

Using NODE statement to identify groups: The NODE statement syntax and parameters specific to defining groups are shown below. For a complete description of the NODE statement, see *CA-IDMS System Generation*.



Parameters

Nodename

When using a type of GROUP, specify the name of a group to which the current CV can route database sessions.

GRoUp

Specifies that the named node is a group. This is the group name specified on the database name table DBGROUP statement.

DEFault NODe

Specifies the node to use if access to the group fails (i.e., there are no active CVs available to service a group request).

Nodename

Specifies a node name to use if access to the group fails. This node must also be defined with an access type of either CCI, CVNUMBER, or VTAM.

LOCAl

Specifies that the node to use is local, which is the current system. The database session will be processed by the system being defined.

NULI

Specifies that there is no default node. If a database session is routed to the named group, and there are no CVs available to service the request, the database session will fail.

Example: The following example adds a group to the front-end definition for CV IDMS070. Since the default node for the group is not defined on IDMS070's resource name table, it is also added.

```

MODIFY SYSTEM IDMS070

ADD NODE EMPGROUP
GROUP DEFAULT NODE IDMS071.

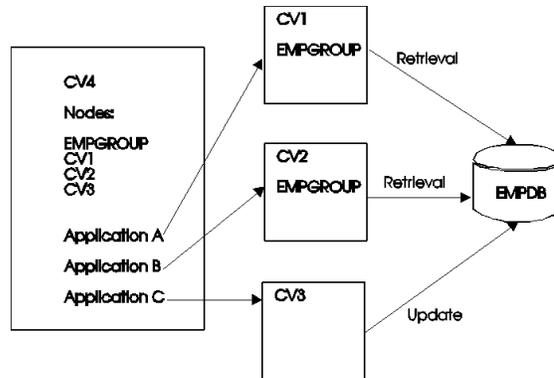
ADD NODE IDMS071
CCI.

GENERATE.
  
```

3.4.2.4 Sample group definitions

This section describes how a sample group is defined and used at run time.

The following illustration shows the use of the EMPGROUP group. It contains CV1 and CV2, which can access the EMPDB database in retrieval mode. CV3 has access to EMPDB for update purposes. Applications updating EMPDB do so by accessing node CV3 explicitly. CV3 is not assigned to a group.



Modifying database name table: The following example illustrates how the EMPGROUP group is defined to the existing system definitions for the CV1 and CV2 backend CVs and the CV4 frontend CV.

Group definition in database name table for CV1

```
CREATE DBGROUP CV1TABLE.EMPGROUP
ENABLED;
```

```
GENERATE DBTABLE CV1TABLE;
```

Group definition on database name table for CV2

```
CREATE DBGROUP CV2TABLE.EMPGROUP
ENABLED;
```

```
GENERATE DBTABLE CV2TABLE;
```

Modifying resource name table: In this example nodes CV1 and CV2 can be reached via dynamic database session routing. Node CV3 can be reached explicitly:

```
MODIFY SYSTEM CV4.
```

```
ADD NODE EMPGROUP
GROUP DEFAULT NODE LOCAL.
```

```
ADD NODE CV1 CCI.
ADD NODE CV2 VTAM.
ADD NODE CV3 CCI.
GENERATE.
```

How EMPGROUP is used at run time: At run time, using either the resource name table or user exit 23, retrieval requests for database services are identified to be serviced by node EMPGROUP using an access type of GROUP.

Note: Exit 23 may be used to override the specified node name on the bind.

The access type of GROUP directs CA-IDMS to solicit a backend CV in the EMPGROUP to service the request. An available CV in the EMPGROUP group volunteers to service the request. From this point on, processing takes place as it normally does in CA-IDMS using the node name of the CV that volunteered to service the database request. Coupling Facility list structures are used in the Coupling Facility to determine which backend CV will be the volunteer.

For a complete discussion of using dynamic database session routing, see the *CA-IDMS Parallel Sysplex User Guide*.

3.4.3 Cloning backend CVs

The backend CVs in a group can be clones of an existing CV. This allows you to run multiple copies of the same backend CV to service the same database. This section describes how to set up backend CVs so they can be cloned.

What is a cloned CV? A cloned CV uses an existing system definition for a CA-IDMS system. It does not exist as another generated system definition in the dictionary. A cloned CV is created by CA-IDMS when an operator starts a CV that is identified as one that can be cloned, using a system number that is reserved for cloning.:

Note: CV cloning is designed to clone backend CVs running in a Parallel Sysplex.

Benefits: You can start and stop cloned CVs to respond to changes in your CA-IDMS workload to increase transaction throughput and improve response time.

For example, when you see a degradation in performance while monitoring your CA-IDMS environment, you can start cloned CVs, as necessary, to increase application throughput and improve response time.

Using this feature with the dynamic routing of retrieval run units, you can make multiple CVs continuously available to process retrieval requests.

3.4.3.1 Cloning CVs

To clone a CV, its system definition must conform to specific naming conventions and its startup JCL must include parameters for cloning. You must also assign the cloned CVs to a group(s) and define them to the resource name table of the frontend CVs that can route requests to them.

The system definition and startup JCL requirements are discussed in this section. For information on setting up backend CVs for dynamic database session routing, see the sections Backend CV definitions and Front-end CV definitions presented earlier in this chapter.

3.4.4 Managing dynamic database session routing

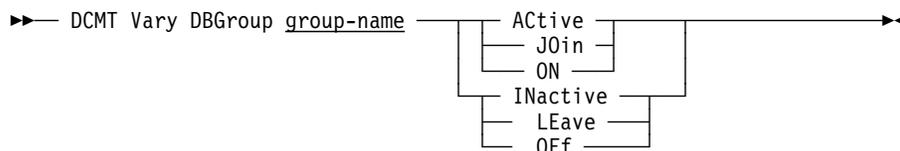
You can manage a CV's participation dynamically using the DCMT VARY DBGROUP command.

What you can do: You issue the DCMT VARY DBGROUP command to activate and inactivate dynamic database session routing and to manage a CV's participation in a group. The tasks you can perform are summarized in the table below.

To do this	Use these DCMT VARY DBGROUP parameters
Enable and disable dynamic database session routing on an executing frontend CV	ACTIVE/INACTIVE
Join a CV to a group or disable it from a group	JOIN/LEAVE
Activate dynamic database session routing and enable the CV to participate in the named group or inactivate dynamic database session routing and disable the CV from participating in the named group	ON/OFF ON is the same as using the ACTIVE and JOIN parameters. OFF is the same as using the INACTIVE and LEAVE parameters.

3.4.4.1 Using the DCMT VARY DBGROUP command

Syntax: The syntax for the DCMT VARY DBGROUP command is shown below.



Parameters

Group-name

Specifies the name of a group. *Group-name* must be a 1- through 8-character name that conforms to the naming conventions for node names defined in a system definition.

Active

Enables dynamic database session routing to the named group. ACTIVE affects the frontend status of a CV. By default, dynamic database session routing is active at CV startup. Use the ACTIVE parameter if the INACTIVE parameter has been previously issued since the startup of the currently executing CV.

INactive

Disables dynamic database session to the named group; all database sessions routed to the named group are statically routed to the default node name specified

for the named group in the NODE statement on the Resource Name table. INACTIVE affects the frontend status of a CV.

JOIN

Joins the currently executing CV to the named group. Use the JOIN parameter to make the CV a member of the named group, even if there is no DBGROUP statement in the DBTABLE for the CV. JOIN affects the backend status of a CV.

LEAVE

Specifies that the currently executing CV is no longer a member of the named group. LEAVE affects the backend status of a CV.

ON

Enables dynamic database session routing and joins the currently executing CV to the named group. ON is equivalent to issuing a DCMT V *group-name* ACTIVE and a DCMT V *group-name* JOIN. It affects the currently executing CV's status as both a frontend and a backend CV. It results in a CV acting as both a frontend CV and a backend CV

OFF

Disables the currently executing CV from the named group and inactivates dynamic database session routing to the named group. OFF is the same as issuing a DCMT V *group-name* INACTIVE and a DCMT V *group-name* LEAVE.

Usage

Joining a CV to a group: In the example below, the JOIN option is used to join the currently executing CV to the DBDCGR group. Database sessions can now be dynamically routed to the currently executing CV through the DBDCGR group. Notice that the DBDCGR contains three backend CVs.

```
VARY DBGROUP DBDCGR JOIN
*** Vary DBGroup request ***
DBGroup DBDCGR has 003 backends
Backend status: Active; Number of requests processed: 0000000000
Frontend status: Active; Number of requests processed: 0000000000
```

3.4.5 Monitoring and tuning dynamic database session routing

Using DCMT commands and the CA-IDMS Performance Monitor, you can monitor how database sessions are being serviced by dynamic database session routing, and as appropriate, modify parameters to suit your processing needs.

The tools you can use are summarized in the table below. Samples of some of them are provided after the table. For complete information on monitoring and tuning the use of dynamic database session routing, see the *CA-IDMS Parallel Sysplex User Guide*.

Tool	Description
DCMT DISPLAY DBGROUP	Displays statistics for all groups to which the currently executing CV can direct requests or for the specified group. Statistics include the number of active backend CVs assigned to groups and the total number of requests processed by each.
DCMT DISPLAY DBTABLE	Lists each group defined in the table and its status.
DCMT DISPLAY NODE	Displays the name of each node with its associated type. You can issue this command on the frontend CV to show all backend CVs to which it can route database sessions.
DCMT VARY DBGROUP	Lets you vary the status of a CV's participation in a group and turn dynamic database session routing on and off.
LOOK DBTABLE (DC)	Lists each group defined in the table and its status.
IDMSLOOK DBTABLE (batch)	Lists each group defined in the table and its status.
CA-IDMS Performance Monitor Interval Monitor (online and batch)	Displays detailed statistics (using DBGROUP category) and wait statistics (using SUMMARY HISTORY, SUMMARY DETAIL, and WAIT Screens).

3.4.5.1 Using DCMT DISPLAY DBGROUP

You can use the DCMT DISPLAY DBGROUP command to display statistics about a currently executing CV's role in dynamic database session routing as follows:

- To display statistics for the CV as both a frontend and a backend CV, including the number of requests processed
- To display statistics on the groups to which it can route requests as a frontend CV

You can display information for all groups to which a currently executing CV can route requests or for a specific group. Each option is discussed next.

Displaying all groups: The DCMT DISPLAY DBGROUP * command displays information about all groups defined to the currently executing CV. It also shows the CV's status as both a frontend and a backend CV in dynamic database session routing.

The example below shows that there are two groups to which the currently executing CV can route requests, DBDCGR and IDMSGR. Dynamic database session routing is active on the currently executing frontend CV and both groups are active as indicated by the ACTIVE status in the STATUS column under FRONTEND. The BACKEND STATUS of INACTIVE indicates that the currently executing CV is not available as a backend CV.

```

DCMT DISPLAY DBG *
*** Display DBGroup request ***
DBGroup * Number of * Backend * Frontend
        * backends * Status ; Requests * Status ; Requests
*****
DBDCGR * 002 * Inactive N/A * Active ; 000000001
IDMSGR * 002 * Inactive N/A * Active ; 000001020
    
```

Displaying information about a specific group: The DCMT DISPLAY DBGROUP *group-name* command displays information about the named group and also displays the back-end and front-end status of the currently executing CV.

The example below displays information about the IDMSGR group. It shows that it is comprised of two backends, IDMS073 and IDMS072, and the number of times each has replied to a request for services from the currently executing CV. IDMS073 has responded to 492 requests and IDMS072 has responded to 528. Additionally, statistics about the currently executing CV as both a backend and a frontend CV are displayed. The total number of requests processed by this frontend CV is 1020.

```

DCMT DISPLAY DBGROUP IDMSGR
*** Display DBGroup request ***
DBGroup IDMSGR has 002 backends
Backend status: Inactive; Number of requests processed: N/A
Frontend status: Active; Number of requests processed: 000001020
Replies on frontend requests distribution: IDMS073 : 000000492
                                           IDMS072 : 000000528
    
```

3.4.5.2 Using the Interval Monitor's DBGROUP Detail screen

The DBGROUP Detail screen displays statistics on the use of groups. The number of requests processed by each group and wait statistics are displayed. The sample screen below displays statistics on the DBDCGR and IDMSGR groups defined to frontend CV IDMS071.

```

PM-R14.0 IDMS071      Computer Associates Intl. V71      96.010
10:19:05.28
CMD→                                                         Window : 02

02 08:30 DGDG DBGroup Detail
          DBGroup      DBGroup      Number      Wait      Average
          Name         Requests    Waits       Time       Time
-         DBDCGR         0         0         .0000S     .0000S
-         IDMSGR        104        42        2.6059S     .0250S
    
```

You can also use Report 10, DBGroup Summary Report, in the Interval Monitor to monitor statistics on DBGroup usage:

3.5 Using shared cache

You can run CA-IDMS systems in a Parallel Sysplex for the purpose of sharing database buffers across multiple CVs running in a Sysplex. Buffers are shared across CVs using a global or shared cache in a Coupling Facility.

What is a shared cache: A shared cache is basically a large, high-speed buffer in a Coupling Facility containing database pages that are accessible by multiple CVs running in a Parallel Sysplex.

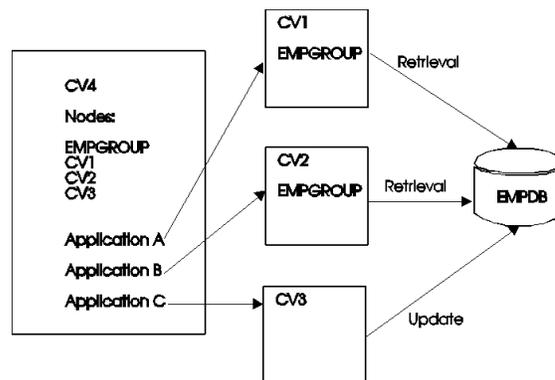
How shared cache works: CA-IDMS manages a shared cache using Coupling Facility list structures. The list structures are used to manage the files participating in the shared cache and the updates made to pages in the cache.

At system startup, CA-IDMS connects to the cache list structure when the first file assigned to a shared cache is opened. CVs containing at least one file assigned to a shared cache will connect to the list structure.

Here is how update and retrieval access to the shared cache and local buffers works at run time:

- When an update CV writes a page to disk, it also updates the page in the shared cache. This causes invalidation of that page in all other CVs that ever read the page.

The illustration below shows how a CV updates a shared cache and invalidates the updated page in all CV buffers in which it exists.



- When a CV needs to access a database page and shared cache is turned on for a file, a current copy of a page is obtained by performing one or more of the following:
 - If the page in the local buffer is still valid, it is retrieved.
 - If the page is not in the local buffer or is no longer valid, the CV looks for it in the shared cache.
 - If the page is found in the shared cache, it is retrieved.

- If it is not found in the shared cache, it is read from disk and placed in both the shared cache and the local buffer.

Benefits In addition to the benefit of having access to current data from multiple CVs, accessing data from a shared cache requires less overhead than accessing data from disk. Because access to data in the Coupling Facility uses high-speed, fiber optic links, accessing pages from the shared cache greatly reduces the overhead otherwise incurred by doing an I/O to the database.

XA storage and shared cache: When the first CA-IDMS file using shared cache is opened, CA-IDMS allocates 300K work space and buffers in XA opsys storage. If the storage is not available, message DC215002 is displayed in the IDMS log and job console log. The open-file-for-shared-cache request fails and CA-IDMS continues without shared cache.

This means that you should always verify that the file is really using shared cache which you can do by issuing these DCMT commands:

```
DCMT D SHARed CAche *  
DCMT D FILE file-name
```

XA storage and buffers: During operation under heavy load, it is possible that CA-IDMS needs more buffers. It acquires 260K of XA opsys storage. If the storage is not available, CA-IDMS displays message DC215002 and the read from/write-to shared-cache request fails in such a way that operation continues. When this occurs, CA-IDMS:

- Does not find the page in shared-cache for the read request
- Deletes the pages from shared cache for the write request to insure database integrity

Set up requirements: To implement the shared cache feature, you need to perform these tasks:

- In the Coupling Facility, define a structure that is used to manage the shared cache and define each shared cache you'll use in CA-IDMS
- In CA-IDMS, identify the files to participate in the shared cache and define the name of the shared cache using a file override in the DMCL or dynamically using the DCMT VARY FILE SHARED CACHE command

These requirements are described separately below.

3.5.1 Defining shared cache in the Coupling Facility

To use the shared cache feature, you must define a list structure in the Coupling Facility with the name ZIDMSCACHE0001.

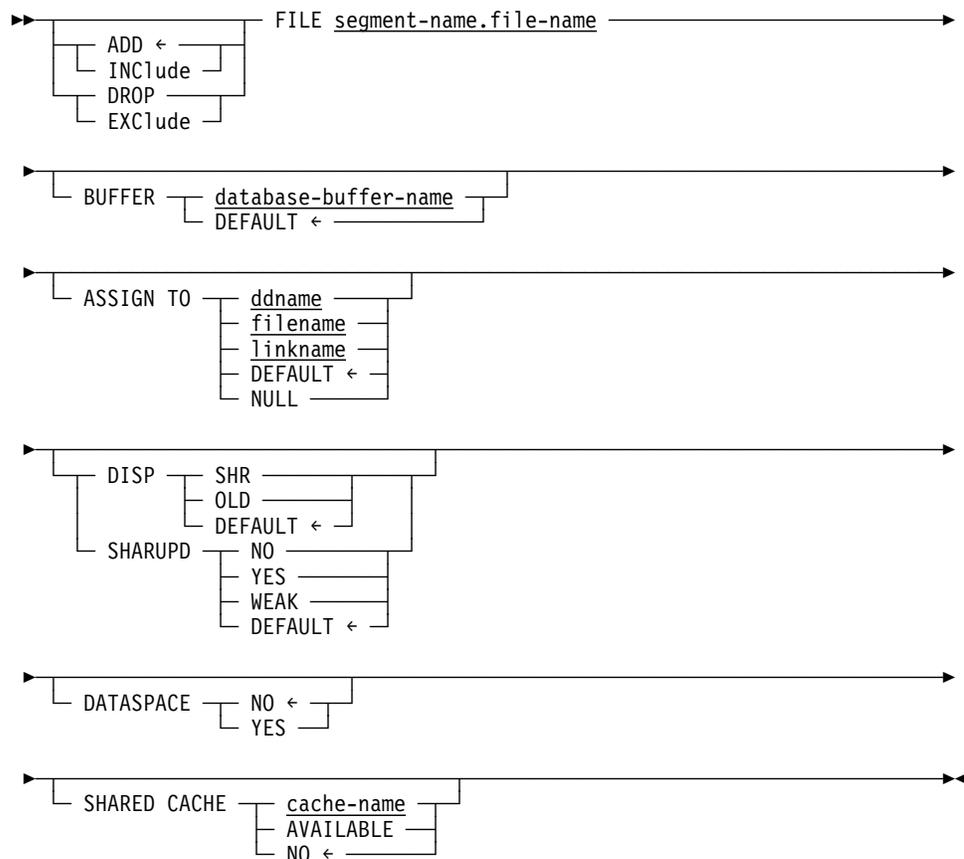
You also need to define each shared cache that you'll implement in CA-IDMS. See the IBM document *MVS/ESA SP 5 Setting Up a SYSPLEX*, for more information on defining shared cache components to the Coupling Facility.

3.5.2 Defining shared cache in CA-IDMS

To implement shared cache in CA-IDMS, you need to identify the files that will participate in the cache using a DMCL or dynamically using a DCMT VARY command. You assign files to a cache by specifying either the name of a cache or the AVAILABLE option. AVAILABLE indicates that the file will use the shared cache defined for it by another CV when that CV opens the file.

3.5.2.1 Assigning files using DMCL file override

You assign a file to a cache using the SHARED CACHE parameter on the FILE override specification of the ALTER DMCL statement. The syntax for the FILE override specification of the ALTER DMCL statement appears below. Parameter descriptions for SHARED CACHE follow. For a complete description of the FILE override specification syntax see *CA-IDMS Database Administration*.



Parameters

cache-name

Specifies the name of a shared cache to which the file is assigned.

AVAILABLE

Specifies that the file is to join an existing shared cache when one is allocated for the same file by another CV, either at startup or dynamically.

NO

Specifies that the file is not to participate in a shared cache, even if another CV has the shared cache option enabled for this file. NO is the default.

3.5.3 Usage

Use only one cache option for a file: If the shared cache file override is used for a file, it is recommended that the dataspace file override is not used. Conversely, if the dataspace file override is used for a file, it is recommended that shared cache is not used.

Example: The example below shows how to modify a DMCL to include the file override SHARED CACHE parameter to do each of the following:

- Indicate that a file is to use the specified cache name
- Indicate that a file can use shared cache when the shared cache becomes available

```
ALTER DMCL CVDML
  INCLUDE FILE FILE1
    SHARED CACHE CACHE1
  INCLUDE FILE FILE2
    SHARED CACHE CACHE1
  INCLUDE FILE FILE3
    SHARED CACHE CACHE2
  INCLUDE FILE FILE4
    SHARED CACHE NO
  INCLUDE FILE FILE5
    SHARED CACHE AVAILABLE
  INCLUDE FILE FILE6
    SHARED CACHE AVAILABLE
;
```

3.5.3.1 Assigning files to a shared cache dynamically

You can also alter the shared cache option for a CV at run time using the SHARED CACHE parameter on these DCMT VARY commands:

- AREA — Assign a shared cache status to all files associated with the area
- FILE — Assign a shared cache status to a file
- SEGMENT — Assign a shared cache status to all the files associated with a segment

Syntax: The syntax for the DCMT VARY FILE SHARED CACHE syntax is shown below. This parameter is also available on the DCMT VARY AREA and DCMT VARY SEGMENT commands. The option descriptions are the same as for those presented above for the SHARED CACHE DMCL override parameter on the FILE statement.

►► For more information on DCMT commands, use the DCMT HELP command or see *CA-IDMS System Tasks and Operator Commands*.

DCMT DISPLAY	Description
SHARED CACHE	Displays the names of the files participating in a shared cache and shows the cache status for each file. It also displays files with a status of AVAILABLE, which are not currently using a shared cache.
FILE <i>file-name</i> AREA <i>area-name</i> FILE BUFFER <i>buffer-name</i> FILE	Displays the status of the shared cache (Yes, No, or Available) and the name of the cache, if applicable, for the named file or files to which the named area or buffer is mapped.
STATISTICS AREA STATISTICS BUFFER STATISTICS FILE	Displays the FND-IN-CACHE column, which shows the number of read requests that were satisfied using pages from a shared cache or ESA dataspace.

You can also use the LOOK DMCL (for DC) and IDMSLOOK DMCL (for batch) to monitor shared cache. Both commands display the status of the shared cache (Yes, No, or Available) and the name of the cache, if applicable, for each file.

Examples: The examples below show how to use DCMT DISPLAY commands to get statistics on shared cache usage by the executing CV.

DCMT DISPLAY SHARED CACHE: To display the status of each shared cache defined to a CV, use the DCMT DISPLAY SHARED CACHE command. This command also displays, in a separate section, files with a cache status of AVAILABLE. A file with a cache status of AVAILABLE begins using a cache when another CV, with a cache enabled, opens it.

```
DCMT DISPLAY SHARED CACHE
*** Display SHARed CAche request ***
List structure: ZIDMSCACHE0001
  Files waiting for cache to become available:
  None.
Cache name: IDMSCACHE00001   Reads: 0000000048   Writes: 0000000048   Status: ON
  File DBDC.SYSTEMXX.DBCRACCA.X           on MVHP91
  File DBDC.SYSTEMXX.DBCRACCB.X           on MVHP91
  File DBDC.SYSTEMXX.DBCRACCE.X           on MVHP91
  File DBDC.SYSTEM71.EMPDEMO.EMPDEMO      on MVHP91
Cache name: IDMSCACHE00002   Reads: 0000000021   Writes: 0000000007   Status: ON
  File DBDC.SYSTEMXX.DBCRBRAA.X           on MVHP91
  File DBDC.SYSTEMXX.DBCRBRAD.X           on MVHP91
  File DBDC.SYSTEMXX.DBCRACCC.X           on MVHP91
  File DBDC.SYSTEMXX.DBCRBRAC.X           on MVHP91
```

Using DCMT DISPLAY FILE to display shared cache status: You can use the DCMT DISPLAY FILE command to display the shared cache status for specific files defined to a CV as shown in the example below.

```

DCMT DISPLAY FILE DBCR*
DBCR.BRANCHA          Upd  0  4000 non-VSAM  No  Yes  BRANCHA
Pre-fetch: Not-Allowed (DMCL)  Pages per Track  12  DISP=SHR (DCMT)
DSname: (DMCL).. DBDC.SYSTEMXX.DBCRBRAA.X
DSname: (JCL)... DBDC.SYSTEMXX.DBCRBRAA.X          VOLSER: MVHP91
Cache-name: IDMSCACHE00002

DBCR.BRANCHB          Upd  0  4000 non-VSAM  No  Yes  BRANCHB
Pre-fetch: Not-Allowed (DMCL)  Pages per Track  12  DISP=SHR (DCMT)
DSname: (DMCL).. DBDC.SYSTEMXX.DBCRBRAA.X
DSname: (JCL)... DBDC.SYSTEMXX.DBCRBRAA.X          VOLSER: MVHP91
Cache-name: IDMSCACHE00002

DBCR.BRANCHC          Upd  0  4000 non-VSAM  No  Yes  BRANCHC
Pre-fetch: Not-Allowed (DMCL)  Pages per Track  12  DISP=SHR (DCMT)
DSname: (DMCL).. DBDC.SYSTEMXX.DBCRBRAA.X
DSname: (JCL)... DBDC.SYSTEMXX.DBCRBRAA.X          VOLSER: MVHP91
Cache-name: IDMSCACHE00002

DBCR.BRANCHD          Upd  0  4000 non-VSAM  No  Yes  BRANCHD
PAGE 002 - NEXT PAGE:

```

DCMT DISPLAY STATISTICS FILE: The DISPLAY STATS FILE command below shows the number of pages found in the cache in the FND-IN-CACHE column. The FND-IN-CACHE statistic represents the number of pages found in a shared cache or an ESA dataspace. Also, the PHY-READS column includes the value in the FND-IN-CACHE. For example, seven physical reads were done against the DBCR.BRANCHA file, four of them were found in the cache. For a specific file, you can compare the value in the FND-IN-CACHE to the number of physical reads to see how efficiently a cache is being used.

```

          DISPLAY STAT FILE DBCR*
----- File -----  Fnd-in-Buf  Phy-Reads  Fnd-in-Cache  Phy-Writes
DBCR.BRANCHA          5           7           4           2
DBCR.BRANCHB         18          18          10           0
DBCR.BRANCHC         10          10           6           0
DBCR.BRANCHD         15          15           9           0
DBCR.ACCOUNTA         0           2           1           2
DBCR.ACCOUNTB         0          19          15           0
DBCR.ACCOUNTC         0          10           8           0
DBCR.ACCOUNTD         0           5           4           0
DBCR.ACCOUNTE        0          14          11           0

```

3.5.4.2 CA-IDMS Performance Monitor

You can use the CA-IDMS Performance Monitor to display statistics on shared cache usage.

Fields displaying shared cache statistics are included on the screens listed in the following table. Samples of some screens are shown after the table.

Tool	Screens/Reports	Description
Real Time Monitor	Buffer IO Summary Buffer IO Detail	Displays the FND-IN-CACHE statistic showing the total number of times a requested page was read from either a shared cache or ESA dataspace
Interval Monitor	Shared Cache (PF24) Detail	Displays detailed statistics on how shared cache is used during an interval
	Summary History	Displays summary statistics on the number of waits for a shared cache and the total wait time on a shared cache
	Summary Detail	Displays the number of waits and the total wait time for a shared cache
	Wait Type Detail	Displays the number of waits for a shared cache and the total wait time on a shared cache, if these statistics are greater than 0 on Summary Detail
	Buffer Detail DBKey/Area Detail I/O Detail Categories corresponding to System area statistics	Displays these types of statistics: number of reads and writes from a shared cache, number of times a page is read from cache, and the number of times a page is not found in cache
Application Monitor	Task Wait Block	Displays the number of waits for a shared cache and the total wait time on a shared cache

In addition, the Interval Monitor provides a new report, Report 9, Shared Cache Summary, that you can use to monitor statistics on shared cache usage:


```

PM-R14.0 IDMS071          Computer Associates Intl. V71    96.010 10:35:34.90
CMD→                      Window : 02

 02 08:40 SUM Summary Detail <
 Start DBGroup DBGroup Sh-Cache Sh-Cache External External Internal Internal
 Time Waits Time Waits Time Waits Time Waits Time
- 08:07 0 .0000S 0 .0000S 0 .0000S 0 .0000S 0 .0000S
- 08:10 0 .0000S 0 .0000S 0 .0000S 0 .0000S 0 .0000S
- 08:20 0 .0000S 148 2.21S 58 .9092S 2 .0008S
- 08:30 42 2.60S 0 .0000S 104 1.34S 0 .0000S
- 08:40 0 .0000S 148 1.90S 14 .2063S 4 .0022S
- 08:50 0 .0000S 22 .1513S 4 .0639S 10 .0060S
- 09:00 0 .0000S 0 .0000S 0 .0000S 26 .0125S
    
```

Interval Monitor — Shared Cache Detail: The following screen shows the Shared Cache Detail for an interval in the Interval Monitor.

```

PM-R14.0 IDMS071          Computer Associates Intl. V71    96.010 10:22:59.75
CMD→                      Window : 02

 02 08:40 SHDT Shared Cache Detail <
 Shared Cache Name Fnd-In Number Failed Number Wait Average
                   Cache Writes Writes Waits Time Time
-  CACHE2          24    12     0     68 .8956S .0186S
-  CACHE1          16     8     0     80 1.0079S .0314S
    
```

3.5.4.3 CA-IDMS System Statistics Report

The CA-IDMS System Statistics Report contains the number of pages found in cache (both shared cache and ESA dataspace) statistic for a DC system.

3.5.5 Tuning a shared cache

Tuning a shared cache involves monitoring its usage and, as necessary, modifying its definition and implementation to use it as efficiently as possible.

You can make the following changes to tune your use of shared cache:

- Change the shared cache status for a file and either assign it to a new or existing cache, or drop it from participating in a shared cache
- Define another shared cache and assign files to it
- Increase the size of an existing cache

For detailed information on tuning a shared cache, see the *CA-IDMS Parallel Sysplex User Guide*.

Chapter 4. CA-IDMS/DB and CA-IDMS/DC

4.1 Overview	4-3
4.2 Extended multitasking support	4-4
4.2.1 Implementing multitasking support	4-4
4.3 Extended 24-hour processing support	4-6
4.3.1 Dynamic lines, terminals, and printers	4-6
4.3.2 Dynamic security refresh	4-7
4.3.3 Dynamic resource allocation	4-8
4.4 Utility enhancements	4-11
4.4.1 SEGMENT support in BACKUP, RESTORE, and UNLOCK	4-11
4.4.1.1 BACKUP and RESTORE utility syntax	4-11
4.4.1.2 UNLOCK syntax	4-12
4.4.2 Enhanced area support in FORMAT JOURNAL	4-12
4.4.3 NonSQL data support in UPDATE STATISTICS	4-13
4.4.4 File support in PRINT SPACE	4-14
4.5 Date and Year 2000 support in DISPLAY/PUNCH statements	4-16
4.6 Security enhancements	4-17
4.6.1 Default signon and user ID options in RHDCSRTT	4-17
4.6.2 DISPLAY/PUNCH ALL syntax for security definitions	4-19
4.6.2.1 DISPLAY and PUNCH ALL statement syntax	4-20
4.6.3 Usage	4-23
4.6.4 Example	4-27
4.6.5 Verifying signons for APPC applications	4-28
4.7 Using LE/370-compliant language compilers with CA-IDMS/DC	4-30
4.7.1 Considerations about LE/370 runtime	4-30
4.7.2 Running LE/370-compliant compiler programs under CA-IDMS/DC	4-31
4.7.3 Supported LE/370 functions	4-32
4.7.4 Unsupported LE/370 functions	4-32
4.7.5 COBOL 370 support	4-32
4.8 IDMSIOX2 DB Exit	4-34
4.9 Enhancements to CICS-reentrant programs	4-39

4.1 Overview

This chapter describes general enhancements to CA-IDMS/DB and CA-IDMS/DC for Release 14.0, including CA-IDMS Security.

The enhancements are grouped and presented as follows:

- Extended multitasking support
- Extended 24-hour processing support
- Utility enhancements
- Date and Year 2000 support in DISPLAY ALL statements
- Security enhancements
- Programming language enhancements
- A new DB exit, IDMSIOX2
- Enhancements to CICS-reentrant programs

4.2 Extended multitasking support

With extended multitasking support, more CA-IDMS system tasks can execute concurrently, each task using a different predefined operating system subtask. Scratch, storage, and security manager modules, as well as CA-IDMS/DB engine modules, can take advantage of multitasking support. The code for these components now run with an MPMODE of ANY.

Benefits: Operating environments that use multitasking should experience enhanced throughput in the areas of security, scratch, and storage management, as well as in the retrieval and updating of CA-IDMS databases. This includes CICS and batch programs as well as CA-IDMS/DC online applications.

Note: Multitasking support is documented in *CA-IDMS System Operations*. If you are a new user of this feature, please refer to this document before implementing it. The discussion of multitasking here is limited to how it is extended in Release 14.0.

4.2.1 Implementing multitasking support

Run CA-IDMS in MVS, BS2000, or MSP: To use multitasking support, you must run CA-IDMS in an environment that supports it: MVS, BS2000, or MSP.

Assign Scratch area to XA storage or ESA dataspace: To get maximum benefit from running scratch manager modules in multitasking mode, assign the Scratch area to XA storage or an ESA dataspace. To assign the Scratch area to XA storage, set the SCRATCH in XA STORAGE parameter on the SYSTEM system generation statement to YES. To assign the scratch area to an ESA dataspace, select the DATASPACE option on the Scratch file definition in the DMCL.

Modify startup JCL: If you have not used multitasking support in a previous release, you implement it in your system startup execution JCL. You specify parameters in specific columns of the job card you use to start your CA-IDMS system. You can optionally specify the number of subtasks to use. The default number of subtasks is the number of processors plus one.

►► See *CA-IDMS System Operations* for Release 12.01 for specific information on implementing multitasking including, setting up your JCL and monitoring a multitasking environment.

For example, in an MVS environment the following PARM specification enables multitasking support for the specified DC/UCF system and defines 8 subtasks:

```

                                Column  Column  Column
                                0        1        2
                                1        0        1
                                ----+----+----+----
//STARTUP EXEC PGM=DCUCFSYS,PARM='S=91                                M8'
.
.
.

```

If you are currently using multitasking in your CA-IDMS environment, you don't need to modify your system startup JCL.

Confirm database procedures and user exits can run in MT Because CA-IDMS/DB engine modules now take advantage of multitasking, you need to confirm that existing database procedures and certain user exit programs that access CA-IDMS internal control blocks can run with an MPMODE of ANY. For more information, see Chapter 2, “Upgrading to Release 14.0” on page 2-1.

Note: You should write DB procedures in assembly language to run in a multitasking environment. They must also be reentrant. For more information about writing DB procedures, see *CA-IDMS Database Administration*.

4.3 Extended 24-hour processing support

You can now perform the following tasks without cycling a central version (CV):

- Add newly generated LINE, LTERM, and PTERM definitions
- Activate new or modified security entities in a security definition

In addition, while a CV remains active, CA-IDMS now dynamically allocates resource management control blocks when the initial allocation is depleted.

Each of these dynamic features is discussed next.

4.3.1 Dynamic lines, terminals, and printers

You can activate the following changes to a system definition without cycling a CV:

- Add new line definitions or modify existing LINE definitions
- Add PTERM and associated LTERM definitions to an existing or new line definition
- Add new printer destinations or modify existing ones
- Add new printer LTERMs to existing or new printer destinations

After generating changes to line, terminal, and printer definitions in your system definition, you issue a new DCMT command to refresh the SYSGEN so that all new or modified lines or a specified line is available for use by a CV.

Note: Only newly generated line, terminal, and printer definitions are activated when you issue the DCMT VARY SYSGEN REFRESH command. Modifications to existing PTERM or LTERM definitions and deletions of any line, terminal, or printer definitions are not processed until you cycle the CV.

Benefit You can make newly generated lines, terminals, and printers available to a CV while it remains active.

New DCMT commands To activate newly generated line, terminal, and printer definitions, issue the new DCMT DISPLAY and VARY SYSGEN REFRESH LINE commands shown below.: To see the new line definitions that you will activate, issue a DISPLAY SYSGEN REFRESH LINES command before you activate the newly-generated definitions.

```

▶▶—— DCMT Vary SYSGen refresh —┐ Lines _____┐
                               └┘ Line line-name ┘
▶▶—— DCMT Display SYSGen refresh —┐ Lines _____┐
                               └┘ Line line-name ┘

```

Parameters

Lines

Specifies that you want to process all newly added line, terminal, and printer definitions, since the last refresh.

Line line-name

Specifies that you want to process the named line.

Example: For example, if you want to add a new printer to an active DC/UCF system without cycling a CV, perform these tasks:

- Add the appropriate entities to the system definition
- Regenerate the system definition
- Issue a DCMT DISPLAY SYSGEN REFRESH LINES command to see the newly generated definition as shown below.

```
dcmt d sysgen refresh lines
*** Display Sysgen request ***
Line UCFLINE was modified
    Added Pterm/Lterm:  UCFTPT05 / UCFLT05
```

- Issue a DCMT VARY SYSGEN REFRESH LINES command to make the definition available to the CV as shown below.

```
dcmt v sysgen refresh lines
*** Vary Sysgen request ***
Line UCFLINE was modified
    Added Pterm/Lterm:  UCFTPT05 / UCFLT05
```

4.3.2 Dynamic security refresh

You can make changes to your security scheme and then activate those changes without cycling a CV. After changing security definitions using the #SECR TT macro and reassembling the RHDCSRTT module, you issue existing DCMT commands to vary the RHDCSRTT nucleus module to new copy and reload it.

Benefit: You can respond to changes in your security environment without bringing down a system and cycling a CV. For example, you can change the security mapping for a resource type or you can make changes to category and activity definitions.

What gets refreshed: When you reload the RHDCSRTT module, the following security definitions are refreshed and any changes you made to them are immediately implemented:

- Access module table
- Category tables
- Activity and category bit map tables

Signon security changes not immediately implemented: Signon and system group security definitions *are not* refreshed when RHDCSRTT is reloaded; users signed on to the system remain signed on even after the reload. Any changes made to signon and system group security for users signed on to a system when a reload is done, do not take place until those users sign off of the system and then sign on again.

Example: After you change a security scheme and modify the RHDCSRTT module, perform the following to activate the changes:

- Issue the DCMT VARY NUCLEUS syntax to vary module RHDCSRTT to new copy
- Issue the DCMT VARY NUCLEUS RELOAD command to reload the changed (new) RHDCSRTT nucleus module

The following example shows these commands.

```
dcmt vary nucleus module rhdcstrtt n c
```

```
          VARY NUCLEUS MODULE RHDCSRTT NEW COPY
IDMS DC283001 V104 USER:ABBTH01  NUCLEUS MODULE RHDCSRTT MARKED TO NEW COPY
```

```
          VARY NUCLEUS RELOAD
IDMS DC283003 V104 USER:ABBTH01  NUCLEUS MODULE RHDCSRTT RELOADED
IDMS DC283004 V104 USER:ABBTH01  CSA/NUCLEUS VECTOR TABLE UPDATED
FOR NUCLEUS MODULE RHDCSRTT
IDMS DC283007 V104 USER:ABBTH01  SECURITY TABLES REFRESHED SUCCESSFULLY
```

For more information

- On the DCMT VARY NUCLEUS command, see *CA-IDMS System Tasks and Operator Commands*.
- On modifying security definitions, see *CA-IDMS Security Administration*.

4.3.3 Dynamic resource allocation

When an executing CA-IDMS system exhausts its primary allocation of resource management control blocks (RCEs, RLEs, and DPEs), it now automatically creates a secondary allocation in XA storage while the system remains active.

Benefit: CA-IDMS systems remain active and do not abend if resource management control blocks are depleted. CA-IDMS systems can accommodate new applications that may consume resources that you didn't account for in your initial resource estimate.

Size of secondary allocation: When CA-IDMS determines that a secondary allocation of resource management control blocks is needed, it allocates the following:

- Under a CV, it allocates a quarter of the primary allocation defined in the system definition
- In batch mode, it allocates twice the primary allocation. This primary allocation, defined internally, is typically large enough to accommodate most of the batch programs so that a secondary allocation is not needed.

Monitoring secondary allocation: You can monitor secondary allocation usage using the following tools:

- Issue the OPER WATCH CRITICAL RESOURCES command and look at the Resource Management statistics as shown below.

IDMS-DC Release 14.0		DC Critical Resource Usage Display		
STORAGE		PROGRAMS		TASKS
# Pools:	4	# Pools:	3	Maximum Tasks: 43
# Pools now SOS:	0	# Rolled out pgms:	0	Active Tasks: 19
# Times SOS:	0	# Programs loaded:	225	System: 18
Amount Available:	9060k	Amount Available:	6800k	Online: 1
Amount Used:	8.52%	Amount Used:	49.04%	External: 0
Amount Fixed:	.00%			
RESOURCE MANAGEMENT				
	RCEs	RLEs	DPEs	
Number				
Avail:	3020	3020	1220	
In Use	13.14%	14.76%	10.90%	
HWM:	15.43%	17.51%	13.60%	
Threshold				
Times:	0	0	0	
Now:	NO	NO	NO	
IDMS DB/DC V81	- Tasks active:19			Time: 16:36:26

- Issue a DCMT DISPLAY STATISTICS SYSTEM command and look at the Internal statistics as shown below.

		0 Tuples Sorted		0 Sort Max	
JOURNAL:		0 Buff Waits		0 User Putjrn1	
Page	311 0-10	3 11-20	151 21-30	5 31-40	0 41-50
Dist	145 51-60	1 61-70	6 71-80	0 81-90	17 91-100
INTERNAL:	RLEs	RCEs	DPEs	Stack	
	529	466	166	724 HWM	
	3000	3000	1200	2000 Sysgen Threshold	
	0	0	0	Times Exceeded	
STORAGE:	12436 Gets	10200 Frees		Gets for type	
	0 PGFIXs	0 PGFREEs		765 DB	
	0 Pages Fxd	0 Pages Freed		344 SHK	
	9032 Scan 1	0 PGRLSEs		0 SHR	
	3404 Scan 2	0 Pages Relsd		3809 SYS	
				30 USK	
	0 SOS COUNT			7488 USR	
PROGRAM:	Act Loads	Pages Load	Wait/Space		
Non-Reent	8	85	0		
Reent	36	803	0		
XA Reent	188	5810	0		
PAGE 002 - NEXT PAGE:					

- Look at the number of DPEs, RCEs, and RLEs allocated at startup and the HWM (high-water mark) on the Storage Pool Overview screen in CA-IDMS Performance Monitor's Real time monitor as shown below.

```
PM-R14.0 SYSTEM71          Computer Associates Intl. V71    96.037 12:13:31.57
CMD→                      Window : 03
                              Refresh: 10

    03 Storage Pool Overview
      Storage Pool Summary
Pools  # Times  Pools
Genned Sys SOS  SOS
3      1      0
      Genned HWM
RLE    3000   757
RCE    3000   636
DPE    5000   438
Stack  1700   817
```

- Look for the DC010007 message in the CV's SYSOUT file or the job console log file for the number of DPEs, RCEs, and RLEs used in secondary allocations.

How to use it: You don't need to make any changes to your system definitions to take advantage of this feature. The control blocks are automatically allocated when they are needed. However, you may want to review the DPE COUNT, RCE COUNT, and RLE COUNT parameters on the system generation SYSTEM statement and optionally adjust them. See Chapter 2, "Upgrading to Release 14.0" on page 2-1 for more information.

4.4 Utility enhancements

The following features are now available to simplify performing database maintenance tasks:

- SEGMENT entity support in UNLOCK, BACKUP, and RESTORE utilities
- Increased area support in the FORMAT utility
- NonSQL data support in the UPDATE STATISTICS utility
- File support for the PRINT SPACE utility

4.4.1 SEGMENT support in BACKUP, RESTORE, and UNLOCK

The BACKUP, RESTORE, and UNLOCK utilities now support operation requests by segment.

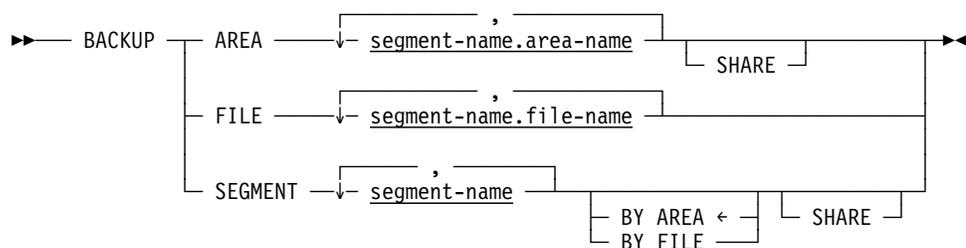
Benefit: When you want to back up, restore, or unlock all the data in a segment, you can simply specify the segment name, and by default, all areas defined to the segment are processed. For BACKUP and RESTORE, you can alternatively specify that all files defined to the segment are processed. This simplifies performing database administration tasks.

4.4.1.1 BACKUP and RESTORE utility syntax

The complete syntax for the BACKUP and RESTORE utilities is shown below. The parameter description for only SEGMENT is provided. For a description of the complete syntax, see *CA-IDMS Utilities*.

When you run BACKUP or RESTORE and specify SEGMENT, you can choose either the default AREA to process all areas defined to the segment, or specify FILE to process all files defined to the segment.

BACKUP Syntax



SEGMENT segment-name

The name of the segment to be backed up.

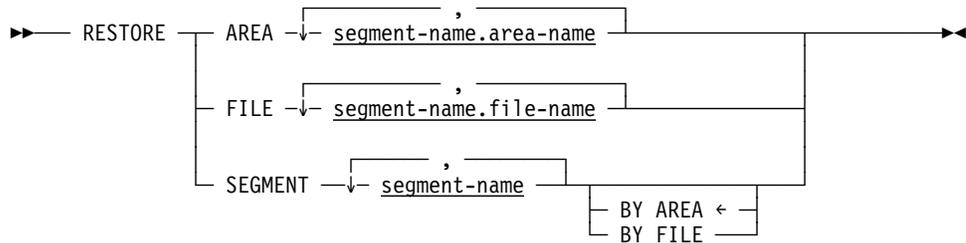
BY AREA

Specifies that each area defined within the segment is to be backed up. AREA is the default.

BY FILE

Specifies that each file within the segment is to be backed up.

RESTORE Syntax



SEGMENT segment-name

The name of the segment to be restored.

BY AREA

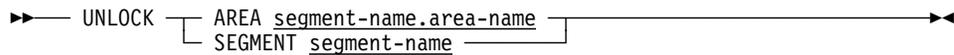
Specifies that each area defined within the segment is to be restored. AREA is the default.

BY FILE

Specifies that each file within the segment is to be restored.

4.4.1.2 UNLOCK syntax

When you run UNLOCK and specify SEGMENT, all areas in the segment are unlocked.



SEGMENT segment-name

The name of the segment to be unlocked.

4.4.2 Enhanced area support in FORMAT JOURNAL

The FORMAT JOURNAL utility now lets you format a journal that can handle more areas without increasing the size of a journal block.

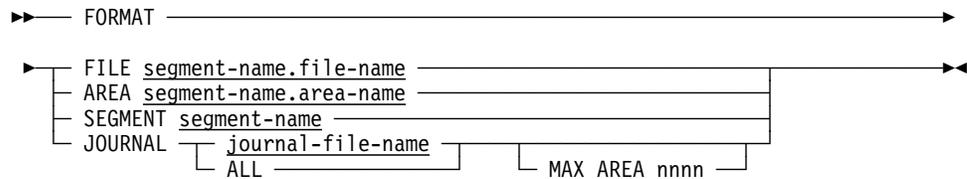
Benefit: Normally when a journal is formatted, CA-IDMS creates a fixed number of JHDA blocks. A JHDA block stores the ready status of areas for warmstart purposes. The size of a journal block and the number of JHDAs limit the number of areas that CA-IDMS can handle.

The new MAX AREA option lets you format a journal that can handle more areas without increasing the size of a journal block. Ideally, the size of a journal block should be optimized to improve runtime efficiency, and should not be affected by the number of areas that may exist.

The MAX AREA option can also reduce the number of default JHDA blocks that CA-IDMS creates, which frees journal space.

Syntax: The complete syntax for the FORMAT utility is shown below. The description for only the MAX AREA parameter is provided. For a description of the complete syntax, see *CA-IDMS Utilities*.

If you do not specify the MAX AREA option, CA-IDMS creates a fixed number of JHDAs.



Parameters

MAX AREA nnn

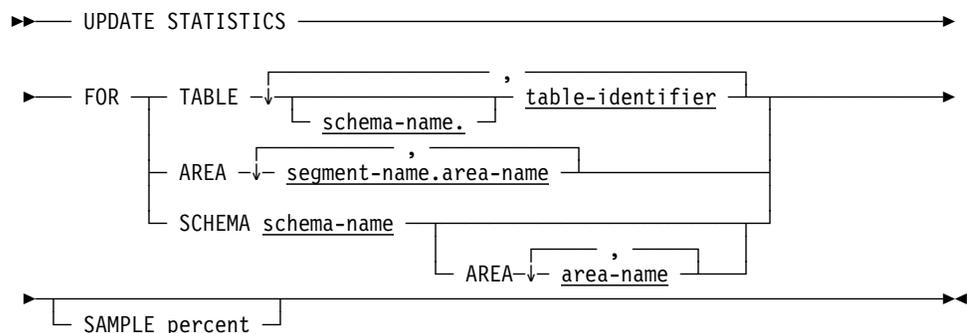
The maximum number of areas to define for the journal, where *nnn* is an integer from 1 to 32,767. The actual number of areas that CA-IDMS can handle may be higher because of rounding and the size of a journal block.

4.4.3 NonSQL data support in UPDATE STATISTICS

The UPDATE STATISTICS utility collects statistics for SQL-defined databases. With this release, it now collects statistics for nonSQL-defined databases too.

Benefit: Statistics for nonSQL-defined databases lets the optimizer select better access strategies. Without such statistics, the optimizer must make assumptions, which may not always be valid.

Syntax: The complete syntax for the UPDATE STATISTICS utility is shown below. The descriptions for only the TABLE and SCHEMA parameters are provided. For a description of the complete syntax, see *CA-IDMS Utilities*.



Parameters

TABLE

Specifies one or more SQL-defined tables and non-SQL-defined tables for which the UPDATE STATISTICS utility is to update statistics.

SCHEMA schema-name

Identifies that SQL-defined schema that references a nonSQL defined-schema.

AREA area-name

Identifies areas of the nonSQL schema from which to collect statistics.

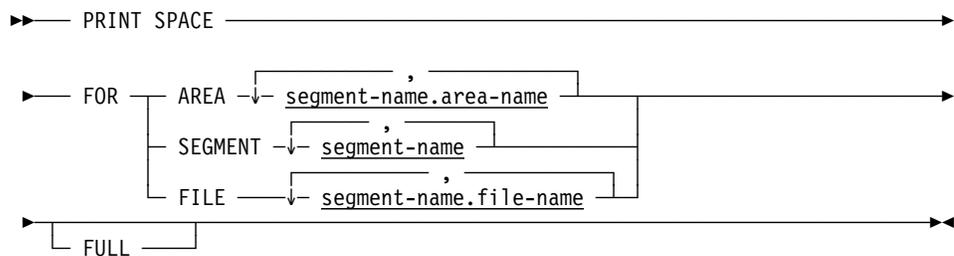
Authorization: To use the SCHEMA option of the UPDATE STATISTICS utility, you must have ALTER authority for the schema.

4.4.4 File support in PRINT SPACE

The PRINT SPACE utility now lets you specify a specific file to report on when processing a multi-file area. Previously, you could only select a specific area to process.

For a *multi-area* file, PRINT SPACE prints a report for each area or portion of an area contained in the file.

Syntax: The complete syntax for the PRINT SPACE utility is shown below. The descriptions for only the AREA and FILE parameters are provided. For a description of the complete syntax, see *CA-IDMS Utilities*.



Parameters

FOR AREA

Directs the PRINT SPACE utility to report on space utilization in one or more areas. This option produces a report for the entire area plus a report for each file in the area.

segment-name

The name of the segment associated with the area.

area-name

The name of the area.

FOR FILE

Directs the PRINT SPACE utility to report on space utilization for each area or portion of an area contained in the file. This option always produces a full report, whether or not you specify the FULL parameter.

segment-name

The name of the segment associated with the file.

file-name

The name of the file.

Authorization: To use the PRINT SPACE BY FILE option, you need DBAREAD authority for each area in the file.

Usage

Logically-deleted records and reports: PRINT SPACE BY FILE sequentially reads the files, letting you include only the files in the JCL stream that you want to process.

When you use this option, PRINT SPACE does not report relocated logically-deleted records as logically deleted. These records are reported as normal records. Therefore, record space utilization reports for an area can produce different results when compared to the file report for the same page range.

4.5 Date and Year 2000 support in DISPLAY/PUNCH statements

With this release, you can use date selection criteria as well as year 2000 support in DISPLAY ALL statements to display dictionary, database, SQL, and security entities.

You implement date selection criteria in these WHERE clause options:

- DATE CREATED
- DATE LAST UPDATED
- DATE COMPILED (in IDD)
- DATE LAST CRITICAL CHANGE (in physical database statements)

You can specify the date as a *value-comparison* string in the form 'MM/DD/YY' in the right-hand side of the conditional expression. CA-IDMS extracts it in CCMMDDYY form to accurately determine the relationship of dates. For example, this DISPLAY ALL statement:

```
DISPLAY ALL RECORDS WHERE DATE CREATED > '01/01/96'.
```

establishes a search criteria to identify the RECORD whose DATE CREATED values are greater than the specified string. The DISPLAY ALL process determines that the date '01/01/96' is greater than the date '12/31/95'.

Alternatively, you may specify the *value-comparison* string on either side of the conditional expression in the form 'CCYYMMDD' to achieve the same results.

You can also substitute day, month, or year for each of these WHERE clause options. For example, this DISPLAY ALL statement specifies a search condition that is based on month and year:

```
DISPLAY ALL RECORDS
  WHERE MONTH CREATED = '01'
  AND YEAR CREATED > '95'.
```

4.6 Security enhancements

With Release 14, CA-IDMS provides these new security enhancements:

- Default signon and user ID options in RHDCSRTT
- DISPLAY/PUNCH ALL syntax for security definitions
- User ID verification for LU6.2/APPC applications

Each topic is described in detail below.

4.6.1 Default signon and user ID options in RHDCSRTT

DFLTSGN parameter: You can now specify the default signon CA-IDMS is to use when a security check is issued and the terminal operator has not signed on using a new DFLTUID parameter in the #SECRIT macro.

Previously, if the DFLTSGN parameter was enabled (set to YES), the following default signons were used:

- For VTAM terminals, the VTAM node name
- For non-VTAM terminals, the physical terminal name (PTERM ID) or the logical terminal name (LTERM ID) if no physical terminal existed

These defaults could be overridden using two Release 12.01 optional APARs (TE20067 and TE20080). The APARs let you override the enabled default signon by specifying either one of the following:

- An explicit user ID (TE20067)
- For users working from VTAM terminals, a PTERM ID (TE20080) instead of the VTAM node name

DFLTUID parameter: Use the DFLTUID parameter to specify the default user ID that CA-IDMS is to use to signon when the DFLTSGN parameter is set to YES, a security check is issued, and the terminal operator has not signed on.

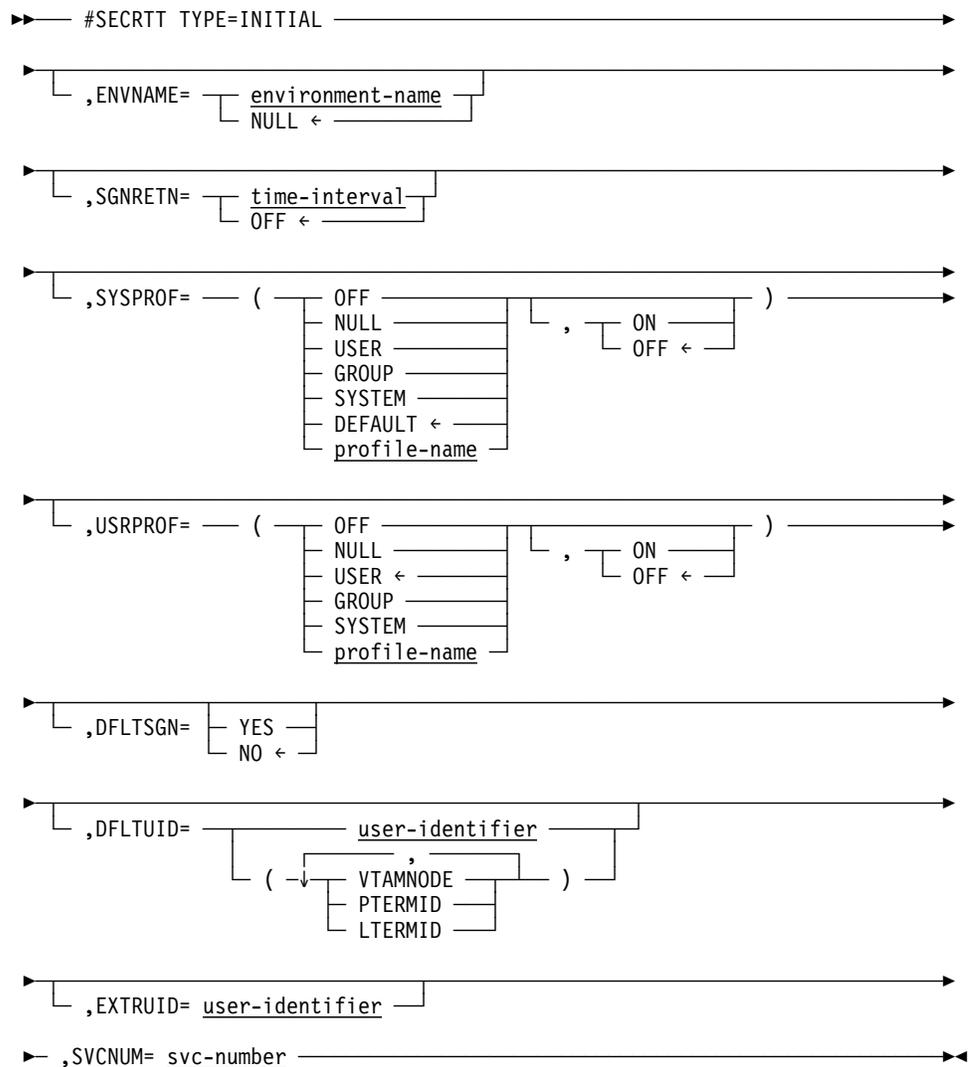
You can specify an explicit user ID or a maximum of three list options: (VTAMNODE, PTERMID, or LTERMID). The sequence in which the list options are specified is not meaningful; the presence of an option determines the way CA-IDMS selects the default signon. If none of the options in the list can satisfy a request (i.e., LTERMID is not specified and the PTERM is not available), then a default signon is not performed.

EXTRUID parameter: For sites that do not have an external security system, you can use the EXTRUID parameter to extract the user ID for the #SECRIT macro call. During the security system initialization phase, CA-IDMS issues an extract call to the external security system to retrieve the current user ID. If no external security system is available, no user ID can be extracted. This parameter allows sites without an

external security system to extract a user ID, which can be used by all startup and shutdown autotasks to signon.

Syntax: The syntax for the #SECRIT macro is shown below, following by descriptions of the DFLTSGN, DFLTUID, and EXTRUID parameters. For a description of the other syntax parameters, see *CA-IDMS Security Administration*.

Note: The SVCNUM parameter is now required; it used to be optional.



Parameters

DFLTUID=

Specifies the default signon CA-IDMS is to use when the DFLTSGN parameter is enabled, a security check is issued, and the terminal operator has not signed on. Specify a *user-identifier* or a list of up to three ID options in parentheses. If DFLTSGN=YES, and you don't specify DFLTUID parameters, the default is the same as it was in Release 12.01: (VTAMNODE,PTERMID,LTERMID).

user-identifier

Specifies the default signon as an unquoted literal from 1- through 18- characters in length.

VTAMNODE

Specifies that for VTAM terminals, the VTAM node name is used as the default signon.

PTERMID

Specifies that the PTERM ID is used as the default signon, if the PTERM is available and the option has not been satisfied by the VTAMNODE parameter (non-VTAM terminals, or VTAMNODE not specified for VTAM terminals).

LTERMID

Specifies that the LTERM ID is used as the default signon, if the option has not been satisfied by the VTAMNODE or PTERMID parameters.

EXTRUID=

Specifies the extract user ID that can be used at sites that do not have an external security system. *User-identifier* is an unquoted literal from 1- to 18-characters. This parameter is not available for BS2000 sites; BS2000 sites can use the CV-USER BS2KSTAR parameter instead.

Note: This parameter replaces the optional APAR GS48846 that was available for releases 12.0 and 12.01.

Usage: If you are using optional APAR TE20067, and you want to activate it in Release 14.0, specify a #SECRIT with DFLTSGN=YES,DFLTUID=*user-identifier* parameters. If you are using optional APAR TE20080, and you want to activate it in Release 14.0, specify a #SECRIT with DFLTSGN=YES,DFLTUID=(PTERMID,LTERMID) parameters.

4.6.2 DISPLAY/PUNCH ALL syntax for security definitions

In addition to using DISPLAY and PUNCH syntax for specific resource definitions in a CA-IDMS security database, you can now issue either a DISPLAY ALL or PUNCH ALL statement for an entity type to display or punch all occurrences defined within that entity type. For example, you can issue a DISPLAY ALL RESOURCE AREAS to see the security definitions for all areas secured in a security database. You can also select occurrences of an entity type to display or punch by specifying:

- Conditional expressions
- First occurrence of the entity type
- Last occurrence of the entity type
- A specific number of occurrences

Choosing which entity occurrences to display: The DISPLAY and PUNCH ALL syntax supports a variety of selection criteria to select occurrences to DISPLAY or PUNCH. You can use a conditional expression with boolean criteria to select occurrences, including a mask comparison. The mask comparison supports the use of

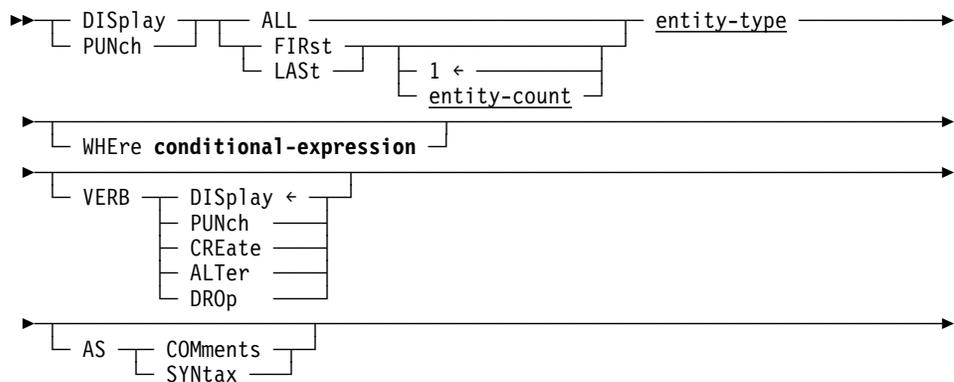
different keywords for each entity type. A table of keywords by entity type is presented under the "Usage" section later in this chapter.

Issue statements from CA-IDMS Command Facility: You can issue DISPLAY/PUNCH statements from either the Online (OCF) or Batch (BCF) Command Facility.

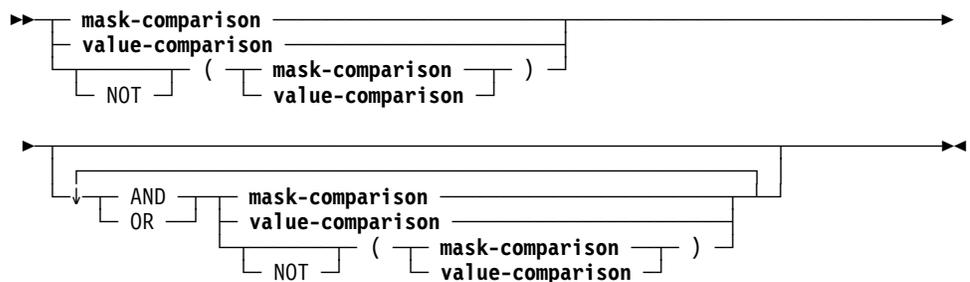
4.6.2.1 DISPLAY and PUNCH ALL statement syntax

The complete syntax and parameter descriptions are shown below. For more information on defining and using security definitions and on the DISPLAY and PUNCH statements for entity types, see *CA-IDMS Security Administration*.

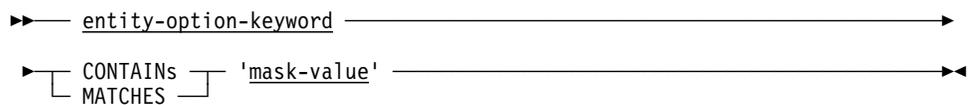
Syntax



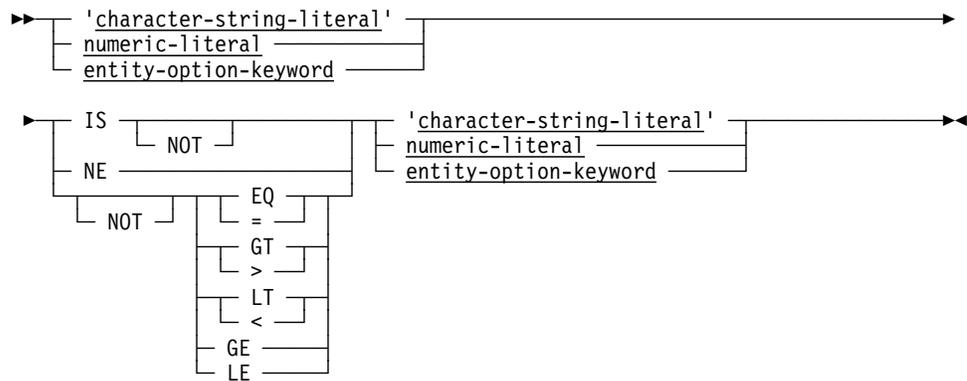
Expansion of conditional-expression



Expansion of mask-comparison



Expansion of value-comparison



Parameters

entity-type

Identifies the entity type that is the object of the request. Valid values are listed in the table in the "Usage" section below.

ALL

Lists all occurrences of the requested entity type that the current user is authorized to display.

Online users: With a large number of entity occurrences, ALL may slow response time.

FIRst

Lists the first occurrence of the named entity type.

LAST

Lists the last occurrence of the named entity type.

entity-count

Specifies the number of occurrences of the named entity type to list. 1 is the default.

entity-type

Identifies the entity type that is the object of the DISPLAY/PUNCH ALL request. Valid values appear in the table under "Usage" below.

VERB DISPLAY/PUNCh/CREate/ALTER/DROp

Specifies the verb that is to accompany DISPLAY/PUNCH output. DISPLAY is the default.

AS SYNTax

Specifies that the text output by the DISPLAY/PUNCH verb is to appear as syntax. In an online session, text displayed as syntax can be edited and resubmitted to the command facility. If the PUNCH command is issued in batch mode, the batch command facility directs the output to the SYSPCH file, where it can be edited and subsequently resubmitted.

AS COMMENTS

Specifies that the text output by the DISPLAY/PUNCH verb be formatted as comments; comments are preceded by *+ and are ignored by the command facility.

WHERE conditional-expression

Specifies criteria to be used in selecting occurrences of the requested entity type.

The outcome of a test for the condition determines which occurrences of the named entity type are displayed.

mask-comparison

Compares an entity type operand with a mask value.

entity-option-keyword

Identifies the left operand as a syntax option associated with the named entity type. The table under "Usage" below lists valid options for each entity type.

CONTAINS

Searches the left operand for an occurrence of the right operand. The length of the right operand must be less than or equal to the length of the left operand. If the right operand is not contained entirely in the left operand, the outcome of the condition is false.

MATCHES

Compares the left operand with the right operand one character at a time, beginning with the leftmost character in each operand. When a character in the left operand does not match a character in the right operand, the outcome of the condition is false.

'mask-value'

Identifies the right operand as a character string; the specified value must be enclosed in quotation marks. *Mask-value* can contain the following special characters:

@	Matches any alphabetic character in <i>entity-option-keyword</i> .
#	Matches any numeric character in <i>entity-option-keyword</i> .
*	Matches any character in <i>entity-option-keyword</i> .

value-comparison

Compares values contained in the left and right operands based on the specified comparison operator.

'character-string-literal'

Identifies a character string enclosed in quotes.

numeric-literal

Identifies a numeric value.

entity-option-keyword

Identifies a syntax option associated with the named entity type; valid options for each entity type are listed in the table presented under "Usage" below.

IS

Specifies that the left operand must equal the right operand for the condition to be true.

NE

Specifies that the left operand must *not* equal the right operand for the condition to be true.

EQ/=

Specifies that the left operand must equal the right operand for the condition to be true.

GT/>

Specifies that the left operand must be greater than the right operand for the condition to be true.

LT/<

Specifies that the left operand must be less than the right operand for the condition to be true.

GE

Specifies that the left operand must be greater than or equal to the right operand for the condition to be true.

LE

Specifies that the left operand must be less than or equal to the right operand for the condition to be true.

NOT

Specifies that the opposite of the condition fulfills the test requirements. If NOT is specified, the condition must be enclosed in parentheses.

AND

Indicates the expression is true only if the outcome of both test conditions is true.

OR

Indicates the expression is true if the outcome of either one or both test conditions is true.

4.6.3 Usage

Output contains only enough information to display/punch entity: Output produced by DISPLAY or PUNCH ALL consists only of the information necessary to execute a DISPLAY/PUNCH request for each entity occurrence. For example, Resource DMCL occurrences are displayed with their name, and AREA occurrences with their fully qualified name (i.e., segmentname.areaname). In an online session, the user can execute the displayed statements by pressing [Enter]. This two-step process allows the user to scan the names of entity occurrences related to the database in which the statement is issued.

Valid entity types and option keywords for conditional expressions: The following table lists valid entity types and keywords that you can specify as *entity-type* and *entity-option-keyword* in the DISPLAY ALL and PUNCH ALL syntax.

Entity type	Entity-option keyword	Selects based on
All Security components		
	NAME	Unqualified Name (1)
	FULL NAME	Qualified Name (1)
	RESOURCE NAME	Unqualified Name (Resources only) (1)
	CREATED by	User who created occurrence
	PREPARED by	User who created occurrence
	LAST UPDATED by	User who last updated occurrence
	REVISED by	User who last updated occurrence
	DATE last UPDATED	Date (MM/DD/YY) occurrence last updated
	MONTH last UPDATED	Month occurrence last updated
	DAY last UPDATED	Day occurrence last updated
	YEAR last UPDATED	Year occurrence last updated
	DATE CREATED	Date (MM/DD/YY) occurrences created
	MONTH CREATED	Month occurrence created
	DAY CREATED	Day occurrence created
	YEAR CREATED	Year occurrence created
Global Security components		
GROups	GROUP name STATUS	Name (ID) of Group Status of GROUP (ACTIVE, INACTIVE, LOGICALLY DELETED)
USERS	USER name STATUS	Name (ID) of User Status of USER (ACTIVE, INACTIVE, LOGICALLY DELETED)
	FULL NAME PROFILE	Full Name of User Profile assigned to User
USER PROFILES	<small>USER PROFILE name</small> PROFILE name	Profile Name Profile Name
Physical Database Security components		
RESOURCE AREAs	resource AREa NAME SEGment name	Unqualified AREA name (1) Areas's segment name
RESOURCE DBs	resource DB NAME	Name of Database
RESOURCE DBTables	resource DBTable NAME	Name of DBTable
RESOURCE DMCLs	resource DMCL NAME	Name of DMCL
RESOURCE NONSQL SCHEmas	resource NONSQL SCHEma NAME	Name of NON SQL Schema
SQL Security Components		

Entity type	Entity-option keyword	Selects based on
RESource ACCess MODules or RESource AMS	resource ACCess MODule NAME resource AM NAME AM NAME SCHema name	Unqualified Name of Access Module (1) Unqualified Name of Access Module (1) Unqualified Name of Access Module (1) Schema Name of Access Module
RESource SCHemas	resource SCHema NAME	Name of SQL Schema
RESource TABles	resource TABle NAME SCHema NAME	Unqualified Name of Table (1) Schema Name of Table
System Security Components		
RESource ACTivities	resource ACTivity name NUMBER	Name of Activity Activity Number
RESource CATegories	resource CATegory NAME NUMBER	Name of Category Category Number
RESource SYSTems	resource SYSTem NAME	Name of System
SYSTem PROfiles	system PROfile NAME	Profile Name
The following Resource Category Components can be selected using the specified entity-option keyword (in addition to those specified in Resource Categories above).		
RESource CATegory ACCess MODules or RESource CATegory AMS	ACCess MODule name DICTName DICTionary name SCHema name	Unqualified Access Module Name (1) Dictionaary Name Dictionaary Name SQL Schema Name
RESource category LOAD MODules	LOAD MODule name DICTName DICTionary name Version	Unqualified Load Module Name (1) Dictionaary Name Dictionaary Name Version Number (in Vnnnn format)
RESource category PROgrams	PROgram name FILE name Version	Unqualified Program name File Name (CDMSLIB) Version Number (in Vnnnn format)
RESource category QUEues	QUEue name	Name of Queue
RESource category RUNunits	RUNunit name DATAbase NAME DBName SUBSchema name PROgram name	Unqualified Rununit name (1) Database Name Database Name Subschema Name Program Name
RESource category TASKs	TASK name	Name of task

Entity type	Entity-option keyword	Selects based on
(1) Unqualified name selections are based on the primary name of the entity occurrence only. To select based on the fully qualified occurrence name, token FULL NAME must be specified. Security components with qualified names are specified in the table below.		

Fully qualified names of security components: The fully qualified names of security components are listed in the table that follows.

Resource	Fully qualified name
ACCESS MODULE	<i>schema-name.access-module-name</i>
AREA	<i>segment-name.area-name</i>
TABLE	<i>schema-name.table-name</i>
CATEGORY ACCESS MODULE	<i>dictname.schema-name.access-module-name</i>
CATEGORY LOAD MODULE	<i>dictname.Vnnnn.load-module-name</i>
CATEGORY RUNUNIT	<i>dbname.subschema-name.program-name</i>
CATEGORY PROGRAM	<i>CDMSLIB.program-name</i> or <i>Vnnnn.program-name</i>

For all other security components, unqualified and qualified names are the same.

Date and Year 2000 support: You can use date selection criteria as well as year 2000 support in DISPLAY/PUNCH ALL statements to display security entities.

You implement date selection criteria in these WHERE clause options:

- DATE CREATED
- DATE LAST UPDATED

You can specify the date as a *value-comparison* string in the form 'MM/DD/YY' in the right-hand side of the conditional expression. CA-IDMS extracts it in CCMMDDYY form to accurately determine the relationship of dates. For example, this DISPLAY ALL statement:

```
DISPLAY ALL USERS WHERE DATE CREATED > '01/01/96';
```

establishes a search criteria to identify the USERS whose DATE CREATED values are greater than the specified string. The DISPLAY ALL process determines that the date '01/01/96' is greater than the date '12/31/95'.

Alternatively, you may specify the *value-comparison* string on either side of the conditional expression in the form 'CCYYMMDD' to achieve the same results.

You can also substitute day, month, or year for each of these WHERE clause options. For example, this DISPLAY ALL statement specifies a search condition that is based on month and year:

```
DISPLAY ALL RESOURCE AREAS
  WHERE MONTH CREATED = '01'
  AND YEAR CREATED > '95';
```

Default order of precedence applied to logical operators: Conditional expressions can contain a single condition, or two or more conditions combined with the logical operators AND or OR. The logical operator NOT specifies the opposite of the condition. The command facility evaluates operators in a conditional expression one at a time, from left to right, in order of precedence. The default order of precedence is as follows:

- MATCHES or CONTAINS keywords
- EQ, NE, GT, LT, GE, LE operators
- NOT
- AND
- OR

If parentheses are used to override the default order of precedence, the command facility evaluates the expression within the innermost parentheses first.

4.6.4 Example

The following examples show sample DISPLAY statements for security definitions.

DISPLAY ALL GROUPS WHERE STATUS IS 'ACTIVE'

```

                                OCF 14.0 ONLINE IDMS NO ERRORS                1/8
DISPLAY ALL GROUPS WHERE STATUS IS 'ACTIVE'
** Status = 0          SQLSTATE = 00000
** DISPLAY GROUP "TESTGROUP" ;
** DISPLAY GROUP "PUBLIC" ;
** DISPLAY GROUP "MIS" ;
** DISPLAY GROUP "HR" ;
** DISPLAY GROUP "ACCOUNTING" ;
** I DC601157 NO MORE ENTITY OCCURRENCES FOUND                                WORD 1
```

DISPLAY ALL USERS WHERE USER NAME MATCHES 'SP'

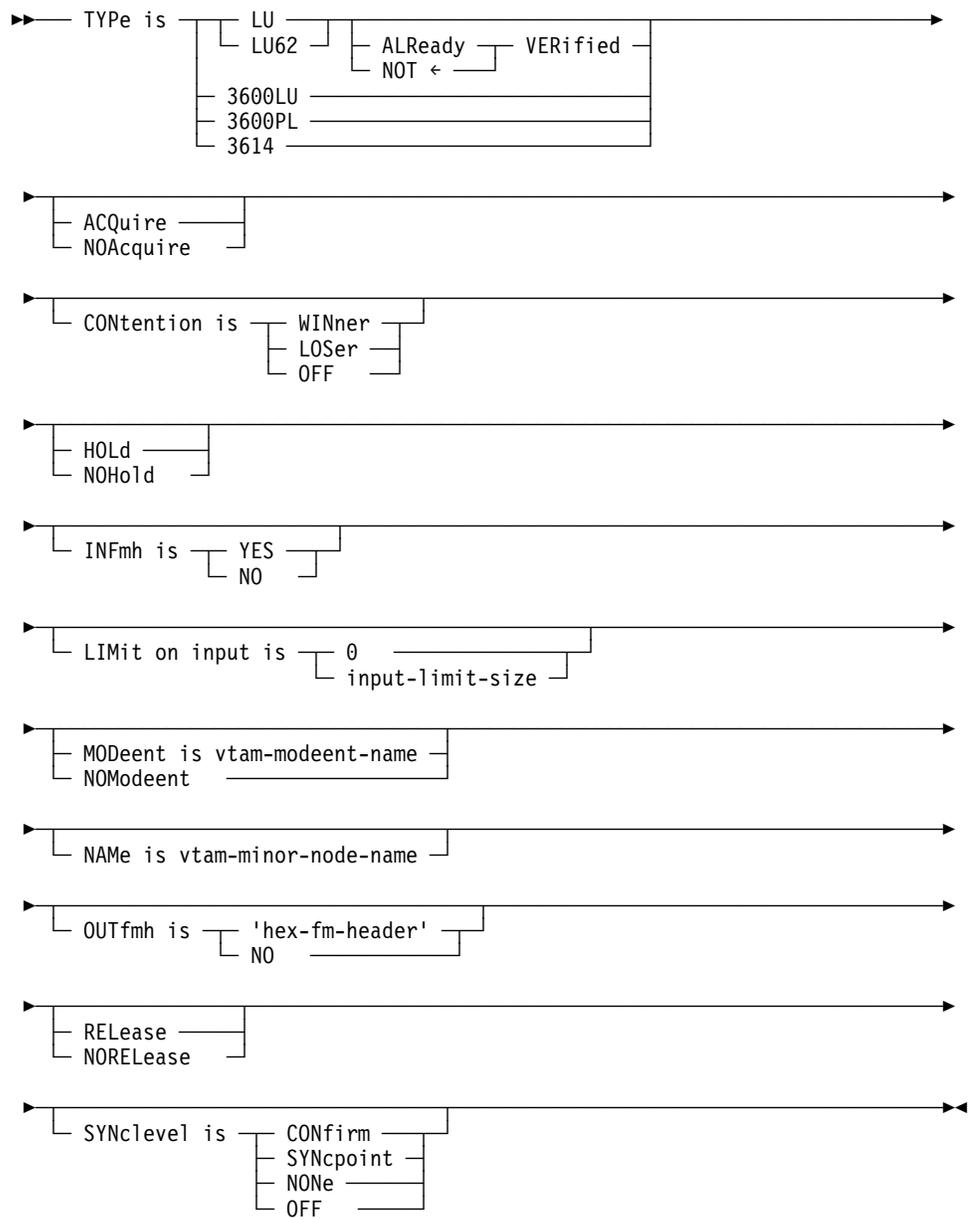
```

                                OCF 14.0 ONLINE IDMS NO ERRORS                1/4
DISPLAY ALL USERS WHERE USER NAME MATCHES 'SP'
** Status = 0          SQLSTATE = 00000
** DISPLAY USER "SPILL01" ;
** DISPLAY USER "SPANL01" ;
** I DC601157 NO MORE ENTITY OCCURRENCES FOUND                                WORD 1
```

4.6.5 Verifying signons for APPC applications

For applications using the LU6.2/APPc protocol, this release offers the ALREADY VERIFIED parameter on the system generation PTERM statement. The ALREADY VERIFIED parameter determines if a user ID has already been verified by the requesting system.

Syntax: The complete syntax for the TYPE IS system generation statement for VTAMLU PTERMs is shown below. Only the description of the ALREADY VERIFIED parameter is provided. For a description of the complete syntax, see *CA-IDMS System Generation*.



Parameters

ALReady VERified/NOT

For VTAMLU types LU and LU62, determines whether the requesting system has verified the user ID. The default is NOT.

4.7 Using LE/370-compliant language compilers with CA-IDMS/DC

What is LE/370?: LE/370 is a runtime environment that replaces the language-specific runtime environments that existed previously. For example, PL/I had its own runtime environment; COBOL II had another. CA-IDMS Release 14.0 can execute programs that use the LE/370 runtime environment. It can also execute programs compiled with pre-LE/370 programs that use the LE/370 runtime environment.

Note: This section applies only to runtime support in CA-IDMS/DC. It does not apply to batch or CICS programs that access CA-IDMS.

How can you use LE/370 with CA-IDMS/DC?: To execute programs compiled by a LE/370-compliant compiler in CA-IDMS, follow these steps to bring up your CA-IDMS environment:

1. Ensure that the CA-IDMS system has been generated with a 24-bit reentrant pool (or program pool, if no reentrant pool is generated) that is large enough to contain the IBM-supplied LE/370 application program interface module CEEPIPI. The size of this module is approximately 325K.
2. Include the LE/370 runtime load libraries in the CDMSLIB loadlib concatenation before any other IBM language loadlibs you are using. For example, before COBOL II or PL/I 2.3.

4.7.1 Considerations about LE/370 runtime

Running pre-LE/370 programs: There are restrictions that apply when you run pre-LE/370 programs in a LE/370 runtime environment within CA-IDMS/DC. *Pre-LE/370 programs* are programs that were compiled with a non-LE/370 compliant compiler, such as:

- COBOL II V4.0
- PL/I V2.3

Some of these restrictions are already documented in the CA-IDMS DML language manuals. Additional restrictions with this release are:

- Pre-LE/370 programs have to run without storage protection
- Restrictions mentioned in the IBM documentation (for example, the *IBM COBOL/370 Migration Guide*) apply

Note: Running pre-LE/370 programs with LE/370 runtime can degrade performance in some circumstances. If you notice poor performance, you should consider recompiling the programs with the newer compiler.

Running LE/370 programs: *LE/370 programs* are programs that were compiled with a LE/370-compliant compiler. CA-IDMS/DC supports these LE/370-compliant compilers:

- COBOL/370 V1R1
- IBM COBOL for MVS & VM V1R2
- PL/I for MVS & VM V1R1

Note: We do not support LE/370-compliant assembler or C programs.

4.7.2 Running LE/370-compliant compiler programs under CA-IDMS/DC

This section describes what you need to do to run a program compiled with a LE/370-compliant compiler in CA-IDMS/DC.

General preparation: The following steps describes how to prepare LE/370-compiled programs for use the CA-IDMS/DC:

Note: If you site does not have site-specific modifications to CEEBINT, the high-level language user exit member, *skip steps 1 and 2*.

1. If you must make site-specific modifications to CEEBINT, you *must* replace the default version of CEEBINT supplied by IBM with the CEEBINT source module in the CA-IDMS SMP PPOPTION library and use it as the basis for any site-specific modifications.
2. Link-edit the updated CEEBINT with CA-IDMS modules IDMSBALI and RHDCLINT. Name the resulting load module RHDCLINT and put it in a load library accessible to the LE/370-compliant language compiler. You can use a JCLIN type USERMOD to modify the RHDCLINT module installed into the CA-IDMS SMP target library.
3. The RHDCLINT load module delivered by CA *or* the one modified in steps 1 and 2 above, must be included in the link edit of every LE/370-compiled program that is to run on a CA-IDMS system, including database or table procedures. The RHDCLINT module is compatible with batch programs, so the same load module can be used for database procedures in batch or under the central version.

Note: You do not need to link-edit the RHDCLINT module for VS-COBOL, VS-COBOL II, or PL/I 2.3 programs that run in a LE/370 online run-time environment.

Application-specific preparation: Include the IDMSCOBI module in the link-edit for COBOL programs. Include the IDMSPLI module in the link edit the PL/I programs.

For each application, you may need to adapt and assemble the IBM-supplied CEEUOPT module. Use this module as the basis for modifying the application-specific runtime options module. This member assumes that the default has been accepted for all parameters not specified. Required settings for CA-IDMS/DC are:

```

TERMTHDACT=(QUIET)
TRAP=(ON)
INTERRUPT=(OFF)
POSIX=(OFF)

```

We strongly recommend that you use smaller values than the default ones for the various heap (e.g. ANYHEAP, BELOWHEAP, HEAP) and stack (e.g. LIBSTACK, STACK) parameters since these are allocated on a task thread basis. Use the output generated by the RPTSTG option to tune these values. CA-IDMS/DC ignores the MSGFILE assignment and sends output to the CA-IDMS log. All other CEEUOPT parameters may be modified as needed. When modifications are complete, consult the section "Creating an Application-Specific Run-Time Options Module" in IBM's *LE/370 Installation and Customization Manual* for further details.

4.7.3 Supported LE/370 functions

CA-IDMS/DC supports these LE/370 functions:

- Math services
- Date and time services
- National language support services

CA-IDMS/DC also supports storage management services, but for performance reasons, they are not recommended. The storage management services are:

- CEECRHP: Create heap segment
- CEECZST: Re-allocate (change size of) heap storage
- CEEDSHP: Discard heap segment
- CEEFRST: Free heap storage
- CEEGTST: Get heap storage

4.7.4 Unsupported LE/370 functions

CA-IDMS/DC does not support the following LE/370 functions:

- CEE3PRM: Get exec parms
- CEETDLI: Call IMS
- CEETEST: Invoke debugging environment

4.7.5 COBOL 370 support

CA-IDMS/DC supports the COBOL/370 V1R1 and IBM COBOL for MVS and VM V1R2 compilers. For both compilers, it supports these features:

- Reentrancy, XA support, static and dynamic calls, optimizer, STRING/UNSTRING/INSPECT, compiler options, COBPACKs, and COBOL II-supported features.
- Execution-time options and space management tuning, as described under LE/370 runtime options above.

CA-IDMS/DC does not support features that were not supported by COBOL II or by the ENVIRONMENT DIVISION clause WITH DEBUGGING MODE.

4.8 IDMSIOX2 DB Exit

This release offers a new DB exit that IDMSDBIO can call when it detects an I/O error, after an I/O has completed, or before issuing certain calls or commands.

Entry Point:: IDMSIOX2

Link Edit with:: IDMSDBIO

Description:: IDMSDBIO calls this routine when it detects an I/O error, after an I/O has completed (even when completed from the cache), or prior to issuing:

- A file open command
- A file close command
- An I/O call (Both Read and Write)
- A #WAIT/CHECK on an I/O call

This exit is NOT called for native VSAM files.

Sample uses:: You can use this exit:

- In place of or in addition to the exits: IDMSDPLX, IDMSJNL2, and IDMSIOXT.
- To maintain a duplicate database and/or journal file.
- To capture I/O statistics.
- To replace the normal I/O calls that CA-IDMS issues with I/O calls issued by the exit.

Calling the exit:: This exit uses standard CA-IDMS/DC system mode calling conventions. Use a #CALL statement to call this exit. You must compile the exit with the #MOPT ENV=SYS macro. The entry point must be defined using a #START macro and control returned using a #RTN macro.

You should code the #START macro with the MPMODE=CALLER option to reduce call overhead and to preserve the current MPMODE lock, if any, that may be held by the current task.

Using the exit with XA systems:: On XA systems, the exit is called in Amode 31. If the exit issues CA-IDMS/DC calls or when the control is returned, the same Amode must be in effect.

Using the exit in multitasking systems:: In a multitasking system, it is the exit's responsibility to establish affinity on the correct TCB before issuing OPEN and CLOSE macros. You can use the #AFFINTY macro for this.

The only resources locked for the current task (TCE) are the storage owned by the current task, and in the case of an open or close call, the FCB/JCB. It is the responsibility of the exit to control concurrent access to other resources.

To avoid putting CA-IDMS/DC into an opsys wait use the #WAIT macro on an ECB before doing any opsys function that may wait on that ECB.

Shared or dataspace cache:: If a file is defined to be in a shared cache (Parallel Sysplex environment) or dataspace cache, IDMSDBIO continues to read from and write to the cache even if the physical I/O is suppressed by the exit. Depending on how you use the exit, this may or may not be desirable. For instance I/O's written to a shadow file should not be written to a shared cache because that could corrupt the primary file through another CV, but writing the same I/O's to a non-shared dataspace cache may be desirable. It is the responsibility of the exit to disable undesirable caching at open time. You can do this by setting the correct flags in #IOX2DS parameter list.

Pages in cache buffer:: If a page being read is in a cache buffer, the physical I/O is bypassed. As a result, the calls to the Pre-Read and Pre-Read-Check exits is not made. However the Post-Read exit is called with a flag set in the parameter list indicating this condition.

Prefetch enabled:: When Prefetch is enabled for a file, multiple reads may be issued for a file before a CHECK is done. The work storage associated with an I/O on the Pre-Read exit remains constant for the pre-read and post-read calls for the same I/O, but other storage could change.

The Pre-Read exit precedes IDMSQSAM processing. If the Read is suppressed, it bypasses IDMSQSAM processing as well as the normal Read. If IDMSQSAM finds the record, the Pre-Check exit is still called, but the I/O on the primary file will have completed. If IDMSQSAM is enabled and the IOX2 exit is waiting on any I/O, it may negate the benefits of IDMSQSAM.

When I/O to a primary file does not require a check macro to be issued, for example, VSAM under DOS or QSAM, the Pre-Check exit is called anyway. In this case, a flag is set in the IOX2 parameter list indicating that the I/O is complete.

User anchor words:: The exit is provided with an address of a work field at the system level, another at the file level, and a third at the I/O level. The exit should not rely on any of these work fields residing in a particular control block. The exit should use the addresses provided in the parameter list.

We recommend that the work fields at the system level and the file level be used with the following rules:

- The word is used to anchor storage, not to store data.
- The storage should contain an 8-byte prefix:
 - CL4'xxxx' — An eyecatcher unique to the storage.

- A(0) — A "next" address for future storage.
- 0X — User fields would follow.
- The first exit would anchor its storage in the word provided by the exit. If another exit also needed to anchor storage at the same level, it would follow the chain of user storage blocks and chain its storage to the last block in the chain.
- The four-byte eyecatcher should be unique to each block of storage, so an exit can identify its own storage.
- Once allocated a storage block should not be deleted, as this could break the chain.

The I/O level work field remains constant for the life of an I/O, but it is not guaranteed to last beyond the Post-I/O exit call. So you should not use this field to chain storage as you would with the more permanent work fields. The concern is that as storage comes, goes, and is reused, it is difficult to maintain a reliable chain.

You should not issue #GETSTG statements for each I/O call because this can affect performance.

Register usage:: Standard IDMS/DC conventions:

- R15/R14 contain entry point and return addresses. These are automatically handled by the #START and #RTN macros.
- R15 — On Exit must contain a return code value.
- R13 — Current stack pointer.
- R12 — Base register for exit after #START.
- R11 — A register automatically saved across DC calls.
- R10 — CSA - Do not modify.
- R9 — TCE - Do not modify.
- R8 — R2 — Available. They were saved prior to calling IDMSIOX2; no guarantee as to content.
- R1 — Parameters - On entry.
- R0 — No guarantee as to content.

Parameters:: When the IDMSIOX2 exit is called, R1 points to a parameter list described by the #IOX2DS copy book. The parameter list contains the following information:

- A function code defining when the exit was called:

- 0 – Pre File Open
- 1 – Pre File Close
- 2 – Pre Read
- 3 – Pre Read Check
- 4 – Post Read
- 5 – Read I/O Error
- 6 – Pre Write
- 7 – Pre Write Check
- 8 – Post Write
- 9 – Write I/O Error.

- Flags that are used in some situations to coordinate control between IDMSDBIO and IDMSIOX2.
- The address of:
 - A fullword associated with the system, for exit use.
 - A fullword associated with the file, for exit use.
 - The current FCB/JCB.
 - The IOP.
 - A list of buffer addresses and RBNs to be read/written. The last pair is marked with the X'80000000' bit in the buffer address.
 - A work area reserved for the exit for the life of the I/O from Pre to Post I/O.

Return codes:: On return from the exit, R15 should contain one of the following values:

- 0 — No Errors, Proceed with normal processing. Supported on all functions.
- 4 — No Errors, Suppress next I/O function. Supported on the "Pre" functions; for example, pre-open, pre-read, pre-check, etc.
- 8 — I/O Error. Supported on I/O functions only. IDMSDBIO behaves as if an I/O error had occurred on this file, returning a 30xx code to its caller.
- 12 — Retry an I/O after an error. Supported on the I/O Error function only.

I/O error function:: The IDMSIOX2 exit is called with the I/O Error function when an I/O error occurs on the primary file. No information about the error is passed.

This function is provided so the IDMSIOX2 exit can Cleanup or Wait on a pending I/O. You can also use this function to issue a request to retry the I/O. For example, suppose this exit were being used for duplexing. When the function requests to try the I/O in error again, the exit could suppress the I/O to the primary file when it retries the I/O and satisfy it from the duplex file.

Suppressing I/O:: With the IDMSIOX2 exit, it is possible to suppress the I/O normally generated by IDMSDBIO. However it is the exit's responsibility to handle the I/O itself.

For example the Pre-Read exit is called passing the address of a buffer and the RBN of the page that needs to be read into that file. The exit could issue the read to a duplex file and suppress the read to the primary file.

When the Pre-Read-Check exit is called, the exit would #WAIT on the ECB associated with its read. When complete, it could fill in the DBIO buffer and then suppress the DBIO CHECK. IDMSDBIO would then behave as if it had read the block itself. The Post-Read exit would not be needed in this case and could simply return. Or it could verify that the contents were in sync with its own version of the page.

4.9 Enhancements to CICS-reentrant programs

Before this release, CICS-reentrant programs that were linked to the IDMSCINT or IDMSCINL modules produced a non-reentrant load module.

With this release, the IDMSCINT and IDMSCINL are fully reentrant. Therefore, when a CICS-reentrant program is linked with these modules now, the resulting load module is fully reentrant.

Chapter 5. CA-IDMS SQL Option

5.1 Overview	5-3
5.2 DISPLAY and PUNCH syntax	5-4
5.2.1 DISPLAY/PUNCH ALL statement	5-5
5.2.2 Usage	5-8
5.2.3 Example	5-12
5.2.4 DISPLAY/PUNCH ACCESS MODULE	5-12
5.2.5 DISPLAY/PUNCH CALC KEY	5-14
5.2.6 DISPLAY/PUNCH CONSTRAINT	5-15
5.2.7 DISPLAY/PUNCH INDEX	5-17
5.2.8 DISPLAY/PUNCH KEY	5-18
5.2.9 DISPLAY/PUNCH SCHEMA	5-20
5.2.10 DISPLAY/PUNCH TABLE	5-22
5.2.11 Usage	5-24
5.2.12 DISPLAY/PUNCH TABLE PROCEDURE	5-24
5.2.13 DISPLAY/PUNCH VIEW	5-26
5.2.14 Usage	5-28
5.3 Dynamic SQL syntax changes	5-29
5.3.1 Dynamic positioned UPDATE and DELETE	5-29
5.3.2 Dynamically-assigned names	5-29
5.3.3 Global statements and cursors	5-30
5.3.4 Dynamic parameters	5-31
5.3.5 Dynamic SQL statements and expressions	5-35
5.3.6 Expansion of cursor-name	5-35
5.3.7 Usage	5-36
5.3.8 Example	5-37
5.3.9 Expansion of cursor-specification	5-38
5.3.10 Usage	5-39
5.3.11 Example	5-40
5.3.12 Expansion of statement-name	5-41
5.3.13 Usage	5-41
5.3.14 Example	5-42
5.3.15 ALLOCATE CURSOR statement	5-43
5.3.16 Usage	5-43
5.3.17 Examples	5-43
5.3.18 CLOSE statement	5-44
5.3.19 Example	5-44
5.3.20 DEALLOCATE PREPARE statement	5-44
5.3.21 Usage	5-45
5.3.22 Examples	5-45
5.3.23 DELETE statement	5-45
5.3.24 Usage	5-46
5.3.25 Examples	5-47
5.3.26 DESCRIBE statement	5-47
5.3.27 Usage	5-48
5.3.28 EXECUTE statement	5-49
5.3.29 Usage	5-51
5.3.30 FETCH statement	5-51

5.3.31 OPEN statement	5-52
5.3.32 PREPARE statement	5-53
5.3.33 Usage	5-54
5.3.34 UPDATE statement	5-55
5.3.35 Usage	5-57
5.3.36 Examples	5-57
5.4 ALTER INDEX support	5-59
5.4.1 Usage	5-60
5.4.2 Example	5-60
5.5 Establishing default transaction options	5-61
5.6 SQLSTATE field in SQLCA	5-62
5.6.1 SQLSTATE values	5-62
5.6.2 SQLSTATE field placement in the SQLCA	5-65
5.7 Optimization enhancements	5-67
5.8 Migration of SQL syntax by CA-IDMS/Dictionary Migrator	5-68

5.1 Overview

This chapter describes general enhancements to the CA-IDMS SQL Option. The enhancements are presented as follows:

- DISPLAY and PUNCH syntax
- Dynamic cursors and dynamic positioned UPDATES and DELETES
- ALTER INDEX support for an SQL-defined index
- Default transaction options using SET SESSION statement
- SQLSTATE field added to SQLCA
- Optimization enhancements
- Migration of SQL entities by Dictionary Migrator

In addition, Chapter 4, “CA-IDMS/DB and CA-IDMS/DC” on page 4-1 describes changes to the UPDATE STATISTICS utility statement, which can now collect statistics from nonSQL-defined schemas.

5.2 DISPLAY and PUNCH syntax

You can now use DISPLAY and PUNCH statements for these logical SQL entities:

- ACCESS MODULE
- CALC KEY
- CONSTRAINT
- INDEX
- KEY
- SCHEMA
- TABLE
- TABLE PROCEDURE
- VIEW

About DISPLAY and PUNCH operations: DISPLAY and PUNCH operations produce as output the SQL statements that describe the named entity. DISPLAY and PUNCH operations do not update the entity description. You can choose to display or punch all the entity occurrences defined within an entity or only specific entity occurrences.

The location of the output depends on which verb is used and whether you are using the online or batch command facility:

- **DISPLAY** displays online output at the terminal and lists batch output in the command facility's activity listing.
- **PUNCH** writes the output to the system punch file. All punched output is also listed in the command facility's activity listing.

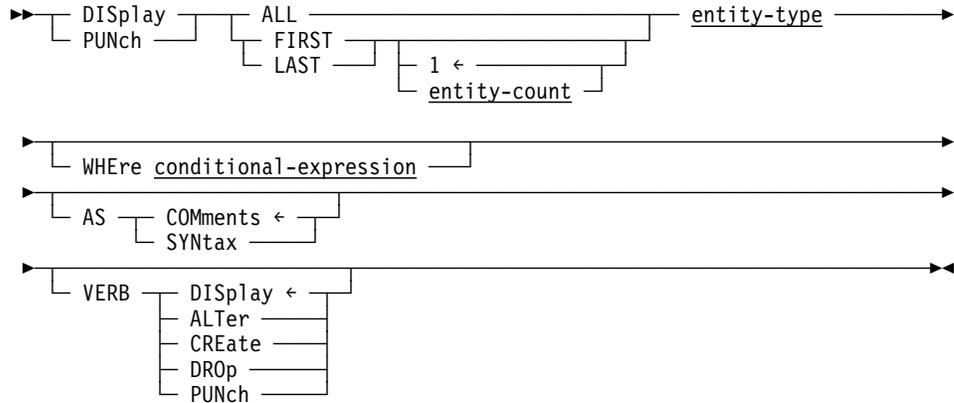
Benefit: With the DISPLAY and PUNCH support for SQL DDL entities, you can easily display or punch entity definitions and change them, or migrate their definitions from one environment to another. For example, you can migrate definitions from one schema to another in the same catalog, and from one catalog to another in the same or different Central Version.

DISPLAY and PUNCH syntax: DISPLAY/PUNCH ALL syntax for SQL DDL entities is presented next, followed by DISPLAY/PUNCH statements for each entity type.

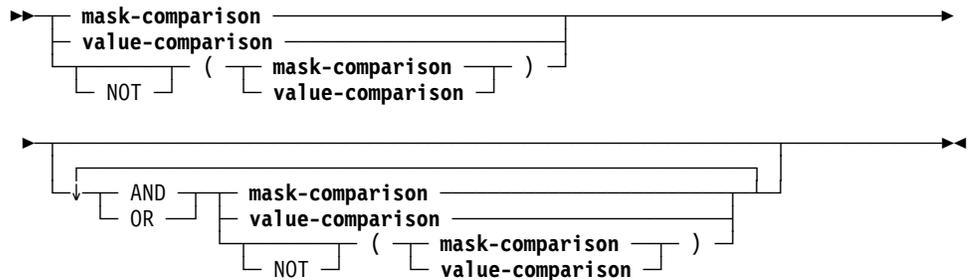
5.2.1 DISPLAY/PUNCH ALL statement

The DISPLAY/PUNCH ALL statement displays all occurrences of an entity type. The basic syntax for each entity type is the same. The entity-option keywords vary by entity type and are presented in a table in "Usage" later in this section.

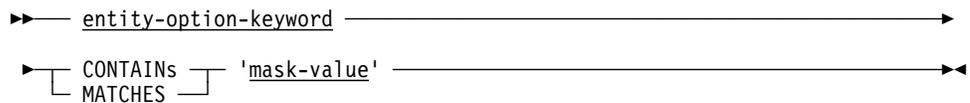
Syntax



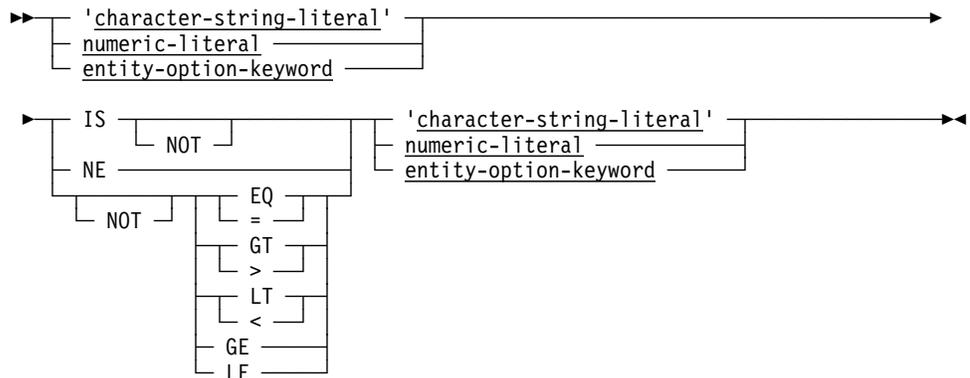
Expansion of conditional-expression



Expansion of mask-comparison



Expansion of value-comparison



Parameters

ALL

Lists all occurrences of the requested entity type that the current user is authorized to display.

Online users: With a large number of entity occurrences, ALL may slow response time.

FIRst

Lists the first occurrence of the named entity type.

LASt

Lists the last occurrence of the named entity type.

entity-count

Specifies the number of occurrences of the named entity type to list. 1 is the default.

entity-type

Identifies the entity type that is the object of the DISPLAY/PUNCH ALL request. Valid values appear in the table under "Usage" below.

WHEre conditional-expression

Specifies criteria to be used in selecting occurrences of the requested entity type.

The outcome of a test for the condition determines which occurrences of the named entity type are selected for display.

mask-comparison

Compares an entity type operand with a mask value.

entity-option-keyword

Identifies the left operand as a syntax option associated with the named entity type. The table under "Usage" below lists valid options for each entity type.

CONTAINS

Searches the left operand for an occurrence of the right operand. The length of the right operand must be less than or equal to the length of the left operand. If the right operand is not contained entirely in the left operand, the outcome of the condition is false.

MATCHES

Compares the left operand with the right operand one character at a time, beginning with the leftmost character in each operand. When a character in the left operand does not match a character in the right operand, the outcome of the condition is false.

'mask-value'

Identifies the right operand as a character string; the specified value must be enclosed in quotation marks. *Mask-value* can contain the following special characters:

@	Matches any alphabetic character in <i>entity-option-keyword</i> .
---	--

#	Matches any numeric character in <i>entity-option-keyword</i> .
*	Matches any character in <i>entity-option-keyword</i> .

value-comparison

Compares values contained in the left and right operands based on the specified comparison operator.

'character-string-literal'

Identifies a character string enclosed in quotes.

numeric-literal

Identifies a numeric value.

entity-option-keyword

Identifies a syntax option associated with the named entity type; valid options for each entity type are listed in the table presented under "Usage" below.

IS

Specifies that the left operand must equal the right operand for the condition to be true.

NE

Specifies that the left operand must *not* equal the right operand for the condition to be true.

EQ/=

Specifies that the left operand must equal the right operand for the condition to be true.

GT/>

Specifies that the left operand must be greater than the right operand for the condition to be true.

LT/<

Specifies that the left operand must be less than the right operand for the condition to be true.

GE

Specifies that the left operand must be greater than or equal to the right operand for the condition to be true.

LE

Specifies that the left operand must be less than or equal to the right operand for the condition to be true.

NOT

Specifies that the opposite of the condition fulfills the test requirements. If NOT is specified, the condition must be enclosed in parentheses.

AND

Indicates the expression is true only if the outcome of both test conditions is true.

OR

Indicates the expression is true if the outcome of either one or both test conditions is true.

AS COMments

Outputs access module syntax as comments with the characters *+ preceding the text of the statement. AS COMMENTS is the default.

AS SYNTAX

Outputs access module syntax which can be edited and resubmitted to the command facility.

VERB DISPLAY/ALTER/CREATE/DROP/PUNCH

Specifies the verb with which the entity statement is to be displayed or punched. For example, if VERB CREATE is specified, the output of the DISPLAY/PUNCH statement is a CREATE statement; if VERB ALTER is specified, the output is an ALTER statement; and so on. The default is VERB DISPLAY.

5.2.2 Usage

Output contains only enough information to display/punch entity: Output produced by DISPLAY or PUNCH ALL consists only of the information necessary to execute a DISPLAY/PUNCH request for each entity occurrence.

Valid entity option keywords for conditional expressions: The following table lists entity type options that you can specify in a conditional expression.

Entity type	Entity-option keyword	Selects based on
All entity types	entity-type NAME	Unqualified name (1)
	entity-type	Unqualified name (1)
	FULL entity-type NAME	Qualified name (1)
	DATE CREATED	Date (MM/DD/YY) occurrence created
	MONTH CREATED	Month occurrence created
	DAY CREATED	Day occurrence created
	YEAR CREATED	Year occurrence created
	DATE LAST UPDATED	Date (MM/DD/YY) occurrence last updated**
	MONTH LAST UPDATED	Month occurrence last updated**
	DAY LAST UPDATED	Day occurrence last updated**
	YEAR LAST UPDATED	Year occurrence last updated**
	CREATED BY	User who created occurrence**
	PREPARED BY	User who created occurrence**
	REVISED BY	User who last updated occurrence**
LAST UPDATED BY	User who last updated occurrence**	

Entity type	Entity-option keyword	Selects based on
ACCESS MODULE	AM name	Unqualified access module name (1)
	SCHema name	Name of access module's schema
	Version	Version number
	FULl TABle NAME	Qualified name of a table referenced by the access module
	TABle SCHema name	Schema name of a table referenced by the access module
	TABle name	Unqualified name of a table referenced by the access module
	DATE COMpiled	Date (MM/DD/YY) access module compiled
	COMpiled	Date (MM/DD/YY) access module compiled
	MONth COMpiled DAY COMpiled YEAR COMpiled	Month access module compiled Day access module compiled Year access module compiled
CALC KEY	SCHema name	Schema name of the table containing the CALC key
	TABle name	Unqualified name of table containing the CALC key
CONSTRAINT	SCHema name	Schema name of the constraint
	REFERENCED FULl TABle NAME	Qualified name of the referenced table
	REFERENCED table SCHema name	Schema name of the referenced table
	REFERENCED TABle name	Unqualified name of the referenced table
	REFERENCIng FULl TABle NAME	Qualified name of the referencing table
	REFERENCIng table SCHema name REFERENCIng TABle name	Schema name of the referencing table Unqualified name of the referencing table
INDEX	SCHema name	Schema name of the indexed table
	TABle name	Unqualified name of the indexed table
	FULl AREa NAME	Qualified name of the area containing the index
	SEGment name	Segment name of the area containing the index
	AREa name	Unqualified name of the area containing the index

Entity type	Entity-option keyword	Selects based on
KEY	SCHEMA name	Schema name of the keyed table procedure
	TABLE PROCEDURE name	Unqualified name of the keyed table procedure
	FULL TABLE PROCEDURE NAME	Qualified name of the keyed table procedure
SCHEMA	TYPE	Type of Schema (NONSQL or SQL)
	full DICTname	NonSQL Schema Dictionary name
	DBName	NonSQL Schema DBName
	NODE name	NonSQL Schema Node Name
	NODename	NonSQL Schema Node Name
	NONsql SCHEMA	Name of the NonSQL Schema
	nonsql schema Version	Version of the NonSQL Schema
	DEFAULT FULL AREA NAME	Qualified area name of the Schema's default area
default SEGMENT name	Segment name of Schema's default area	
default AREA name	Unqualified area name of the Schema's default area	
TABLE	SCHEMA name	Schema name of the table
	FULL AREA NAME	Qualified area name containing the table
	SEGMENT name	Segment name of the area containing the table
	AREA name	Unqualified name of the area containing the table
TABLE PROCEDURE	SCHEMA name	Schema name of the table procedure
	EXTERNAL NAME	Name of the program called to process the procedure
VIEW	SCHEMA name	Schema name of the View
	REFERENCED FULL TABLE NAME	Qualified name of a table referenced by the View
	REFERENCED table SCHEMA NAME	Schema name of a table(s) referenced by the View
	REFERENCED TABLE name	Unqualified name of a table referenced by the View

(1)Unqualified name selections are based on the primary name of the entity occurrence only. To select based on the fully qualified occurrence name, token FULL NAME must be specified. SQL components with qualified names are specified in the table below.

**You can specify this keyword option only when using SCHEMA, TABLE, TABLE PROCEDURE, and VIEW entities.

Fully qualified names of SQL components: The fully qualified names of SQL components are listed in the table below.

Resource	Fully qualified name
ACCESS MODULE	<i>schema-name.access-module-name</i>
TABLE	<i>schema-name.table-name</i>
TABLE PROCEDURE	<i>schema-name.procedure-name</i>
VIEW	<i>schema-name.view-name</i>

Date and Year 2000 support in DISPLAY/PUNCH statements: You can use date selection criteria as well as year 2000 support in DISPLAY ALL statements to display SQL entities.

You implement date selection criteria in these WHERE clause options:

- DATE CREATED
- DATE LAST UPDATED

You can specify the date as a *value-comparison* string in the form 'MM/DD/YY' in the right-hand side of the conditional expression. CA-IDMS extracts it in CCMMDDYY form to accurately determine the relationship of dates. For example, this DISPLAY ALL statement:

```
DISPLAY ALL SCHEMAS WHERE DATE CREATED > '01/01/96';
```

establishes a search criteria to identify the schemas whose DATE CREATED values are greater than the specified string. The DISPLAY ALL process determines that the date '01/01/96' is greater than the date '12/31/95'.

Alternatively, you may specify the *value-comparison* string on either side of the conditional expression in the form 'CCYYMMDD' to achieve the same results.

You can also substitute day, month, or year for each of these WHERE clause options. For example, this DISPLAY ALL statement specifies a search condition that is based on month and year:

```
DISPLAY ALL VIEWS
  WHERE MONTH CREATED = '01'
  AND YEAR CREATED > '95';
```

Default order of precedence applied to logical operators: Conditional expressions can contain a single condition, or two or more conditions combined with the logical operators AND or OR. The logical operator NOT specifies the opposite of the condition. The command facility evaluates operators in a conditional expression 1 at a time, from left to right, in order of precedence. The default order of precedence is as follows:

- MATCHES or CONTAINS keywords
- EQ, NE, GT, LT, GE, LE operators
- NOT

- AND
- OR

If parentheses are used to override the default order of precedence, the command facility evaluates the expression within the innermost parentheses first.

5.2.3 Example

The following example displays all ACCESS MODULES compiled since June 1, 1995:

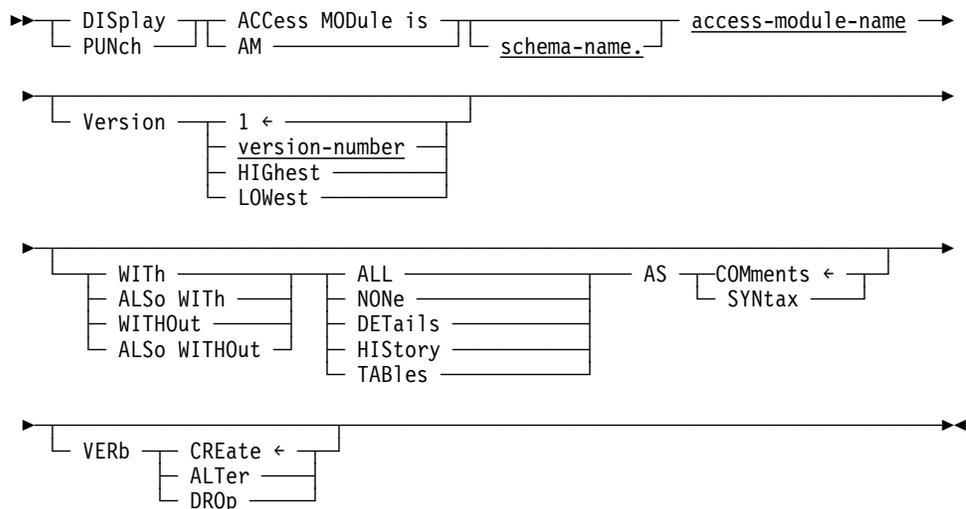
```
DISPLAY ALL ACCESS MODULES
  WHERE DATE CREATED GT '06/01/95'
  AS SYNTAX.
```

5.2.4 DISPLAY/PUNCH ACCESS MODULE

Purpose: Displays or punches an access module.

Authorization: To issue a DISPLAY/PUNCH ACCESS MODULE statement, you must either hold the DISPLAY privilege on or own the access module named in the statement.

Syntax



Parameters

schema-name.

Specifies the schema for the access module. *Schema-name* must identify the schema associated with the version of the access module being modified. If you do not specify *schema-name*, the value used by the command facility is the current schema for your SQL session.

access-module-name

Specifies the name of the access module to display or punch. *Access-module-name* must identify an access module defined and stored in the dictionary.

Version is version-number

Specifies the version number of the access module. *Version-number* is a unique integer in the range 1 through 9999. 1 is the default.

HIGhest

Specifies the highest version number associated with the access module.

LOWest

Specifies the lowest version number associated with the access module.

WITH

Lists the requested information, in addition to the information that is always included, such as the entity occurrence name.

WITHOut

Does *not* list the specified options. *Other* options in effect through the WITH or ALSO WITH clauses in the current DISPLAY statement are displayed.

ALSo WITH

Lists the requested information, in addition to the information requested in previously issued DISPLAY WITH and DISPLAY ALSO WITH statements for the named entity.

ALSo WITHOut

Does *not* list the specified options.

ALL

Specifies the display of all of the information associated with the requested entity occurrence.

NONE

Specifies the display of the name of the requested entity occurrence. NONE is meaningful only when the WITH clause is specified.

DETailS

Specifies the display of entity-specific descriptions.

HIStory

Specifies the display of the date the access module was compiled.

TABles

Specifies the display of all tables associated with the requested access module.

AS COMments

Outputs access module syntax as comments with the characters *+ preceding the text of the statement. AS COMMENTS is the default.

AS SYNtax

Outputs access module syntax which can be edited and resubmitted to the command facility.

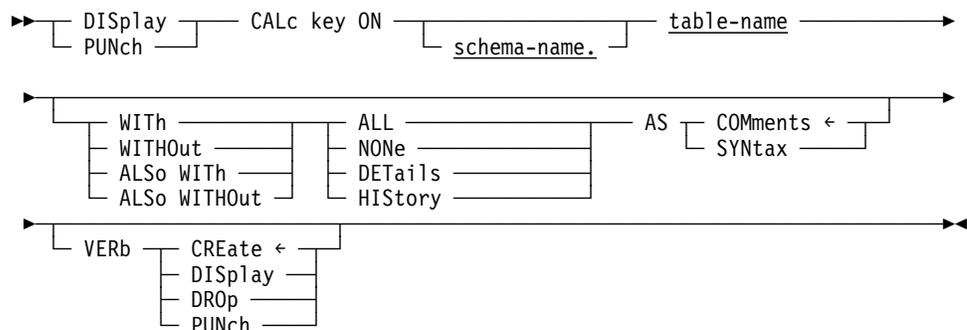
VERB CREATE/ALTER/DROP

Specifies the verb with which the entity statement is to be displayed or punched. For example, if VERB CREATE is specified, the output of the DISPLAY/PUNCH statement is a CREATE statement; if VERB ALTER is specified, the output is an ALTER statement; and so on. The default is VERB CREATE.

5.2.5 DISPLAY/PUNCH CALC KEY

Purpose: Displays or punches a CALC key definition in the dictionary.

Authorization: To issue a DISPLAY/PUNCH CALC KEY statement, you must either own or have the ALTER privilege on the table on which the CALC key is defined.

Syntax**Parameters****schema-name.**

Identifies the schema associated with the named table.

If you do not specify *schema-name*, it defaults to the current schema associated with your SQL session, if the statement is entered through the command facility or executed dynamically.

table-name

Specifies the name of the table on which the CALC key is defined. *Table-name* must be the name of a table defined in the dictionary.

WITH

Lists the requested information, in addition to the information that is always included, such as the entity occurrence name.

WITHOUT

Does *not* list the specified options. *Other* options in effect through the WITH or ALSO WITH clauses in the current DISPLAY statement are displayed.

ALSO WITH

Lists the requested information, in addition to the information requested in previously issued DISPLAY WITH and DISPLAY ALSO WITH statements for the named entity.

Parameters

constraint-name

Specifies the name of the referential constraint, within the current schema associated with your SQL session, to display or punch.

WITH

Lists the requested information, in addition to the information that is always included, such as the entity occurrence name.

WITHOut

Does *not* list the specified options. *Other* options in effect through the WITH or ALSO WITH clauses in the current DISPLAY statement are displayed.

ALSo WITH

Lists the requested information, in addition to the information requested in previously issued DISPLAY WITH and DISPLAY ALSO WITH statements for the named entity.

ALSo WITHOut

Does *not* list the specified options.

ALL

Specifies the display of all of the information associated with the requested entity occurrence.

NONE

Specifies the display of the name of the requested entity occurrence. NONE is meaningful only when the WITH clause is specified.

DETailS

Specifies the display of constraint-specific descriptions.

HIStory

Specifies the display of the date the constraint was created.

AS COMments

Outputs constraint syntax as comments with the characters **+* preceding the text of the statement. AS COMMENTS is the default.

AS SYNtax

Outputs constraint syntax which can be edited and resubmitted to the command facility.

VERB CREate/DISplay/DROp/PUNch

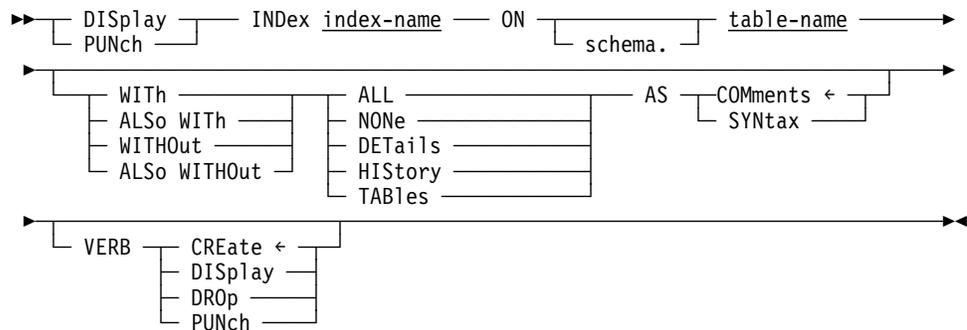
Specifies the verb with which the entity statement is to be displayed or punched. For example, if VERB CREATE is specified, the output of the DISPLAY/PUNCH statement is a CREATE statement; if VERB ALTER is specified, the output is an ALTER statement; and so on. The default is VERB CREATE.

5.2.7 DISPLAY/PUNCH INDEX

Purpose: Displays or punches an index from the dictionary.

Authorization: To issue a DISPLAY/PUNCH INDEX statement, you must either own or have the DISPLAY privilege on the table on which the index is defined.

Syntax



Parameters

index-name

Specifies the name of an index to display or punch. *Index-name* must be the name of an index in the dictionary.

ON table-name

Specifies the table on which the named index is defined.

schema-name.

Identifies the schema associated with the named table.

If you do not specify *schema-name*, it defaults to the current schema associated with your SQL session, if the statement is entered through the command facility or executed dynamically.

WITH

Lists the requested information, in addition to the information that is always included, such as the entity occurrence name.

WITHOuT

Does *not* list the specified options. *Other* options in effect through the WITH or ALSO WITH clauses in the current DISPLAY statement are displayed.

ALSo WITH

Lists the requested information, in addition to the information requested in previously issued DISPLAY WITH and DISPLAY ALSO WITH statements for the named entity.

ALSo WITHOuT

Does *not* list the specified options.

ALL

Specifies the display of all of the information associated with the requested index.

NONE

Specifies the display of the name of the requested index. NONE is meaningful only when the WITH clause is specified.

DEtails

Specifies the display of index-specific descriptions.

HIStory

Specifies the display of the date the index was created.

AS COMments

Outputs index syntax as comments with the characters *+ preceding the text of the statement. AS COMMENTS is the default.

AS SYNTAX

Outputs index syntax which can be edited and resubmitted to the command facility.

VERB CREate/DISplay/DROp/PUNch

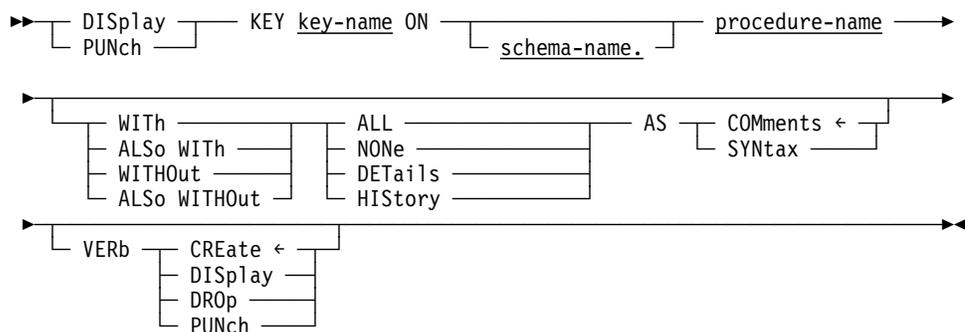
Specifies the verb with which the entity statement is to be displayed or punched. For example, if VERB CREATE is specified, the output of the DISPLAY/PUNCH statement is a CREATE statement; if VERB ALTER is specified, the output is an ALTER statement; and so on. The default is VERB CREATE.

5.2.8 DISPLAY/PUNCH KEY

Purpose: Displays a key definition stored in the dictionary.

Authorization: To issue a DISPLAY KEY statement, you must either own or hold the ALTER privilege on the table procedure on which the key being displayed or punched is defined.

Syntax



Parameters

key-name

Specifies the name of a key on a table procedure.

schema-name.

Identifies the schema associated with the table procedure.

If you do not specify a *schema-name* it defaults to the current schema associated with your SQL session, if the statement is entered through the Command Facility or executed dynamically.

procedure-name

Specifies the name of the table procedure on which the key is defined. The *procedure-name* must identify a table procedure defined in the dictionary.

WITH

Lists the requested information, in addition to the information that is always included, such as the entity occurrence name.

WITHOut

Does *not* list the specified options. *Other* options in effect through the WITH or ALSO WITH clauses in the current DISPLAY statement are displayed.

ALSo WITH

Lists the requested information, in addition to the information requested in previously issued DISPLAY WITH and DISPLAY ALSO WITH statements for the key.

ALSo WITHOut

Does *not* list the specified options.

ALL

Specifies the display of all of the information associated with the requested key.

NONE

Specifies the display of the name of the requested key. NONE is meaningful only when the WITH clause is specified.

DETailS

Specifies the display of key-specific descriptions.

HIStory

Specifies the display of the date the key was created.

AS COMments

Outputs key syntax as comments with the characters **+* preceding the text of the statement. AS COMMENTS is the default.

AS SYNtax

Outputs key syntax which can be edited and resubmitted to the command facility.

VERB CREate/DISplay/DROp/PUNCh

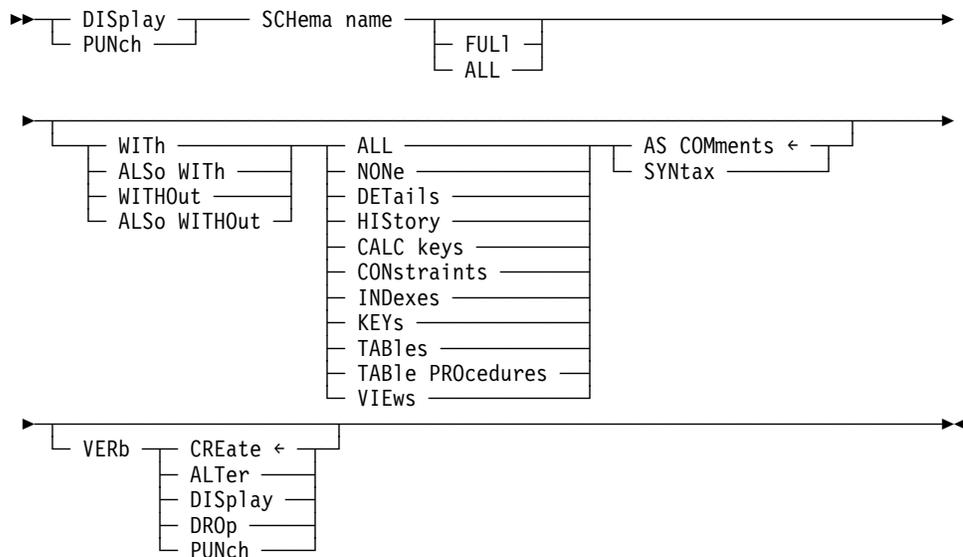
Specifies the verb with which the key statement is to be displayed or punched. For example, if VERB CREATE is specified, the output of the DISPLAY/PUNCH statement is a CREATE statement; if VERB ALTER is specified, the output is an ALTER statement; and so on. The default is VERB CREATE.

5.2.9 DISPLAY/PUNCH SCHEMA

Purpose: Displays or punches an SQL schema in the dictionary.

Authorization: To issue a DISPLAY/PUNCH SCHEMA statement, you must have the DISPLAY privilege on the requested SQL schema.

Syntax



Parameters

schema-name.

Specifies the SQL schema to display or punch.

Schema-name must be the name of the an SQL schema in the dictionary.

FUL1 or ALL

Specifies that you want the SQL schema and all table, view, CALC key, index, and constraint definitions for the requested SQL schema displayed or punched. This option is for SQL-defined schemas only.

WITH

Lists the requested information, in addition to the information that is always included, such as the entity occurrence name.

WITHOut

Does *not* list the specified options. *Other* options in effect through the WITH or ALSO WITH clauses in the current DISPLAY statement are displayed.

ALSo WITH

Lists the requested information, in addition to the information requested in previously issued DISPLAY WITH and DISPLAY ALSO WITH statements for the named entity.

ALSo WITHOut

Does *not* list the specified options.

ALL

Specifies the display of all of the information associated with the requested entity occurrence.

NONe

Specifies the display of the name of the requested entity occurrence. NONE is meaningful only when the WITH clause is specified.

DETAils

Specifies the display of SQL schema-specific descriptions.

HIStory

Specifies the display of the chronological account of an entity's existence, including PREPARED/REVISED BY specifications, date created, and date last updated.

CALC keys

Specifies the display of all CALC keys associated with the requested SQL schema.

CONstraints

Specifies the display of all constraints associated with the requested SQL schema.

INDEXes

Specifies the display of all indexes associated with the requested SQL schema.

KEYs

Specifies the display of all table procedure keys associated with the requested SQL schema.

TABLEs

Specifies the display of all tables associated with the requested SQL schema.

TABLE PROCedures

Specifies the display of all table procedures associated with the requested SQL schema.

VIEWS

Specifies the display of all views associated with the requested SQL schema.

AS COMments

Outputs SQL schema syntax as comments with the characters *+ preceding the text of the statement. AS COMMENTS is the default.

AS SYNtax

Outputs SQL schema syntax which can be edited and resubmitted to the command facility.

VERB CREate/ALTER/DISplay/DROp/PUNch

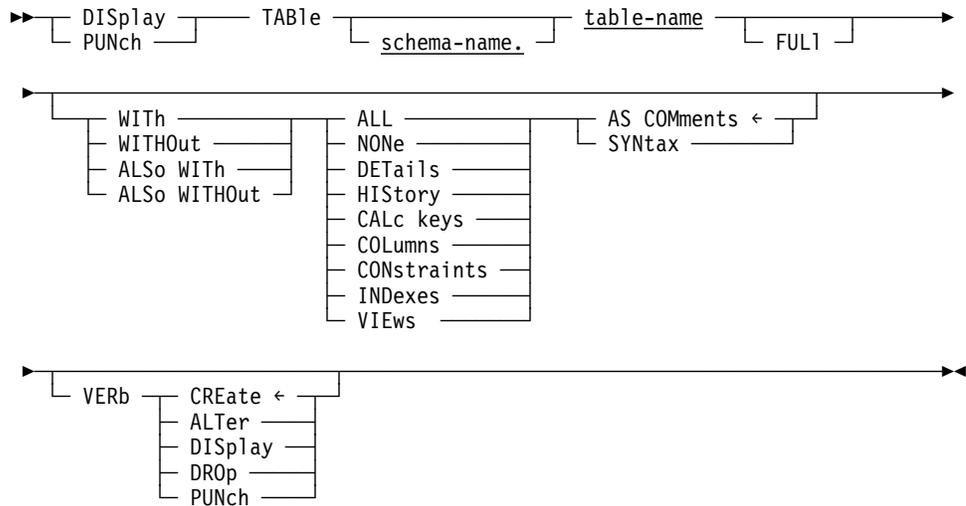
Specifies the verb with which the entity statement is to be displayed or punched. For example, if VERB CREATE is specified, the output of the DISPLAY/PUNCH statement is a CREATE statement; if VERB ALTER is specified, the output is an ALTER statement; and so on. The default is VERB CREATE.

5.2.10 DISPLAY/PUNCH TABLE

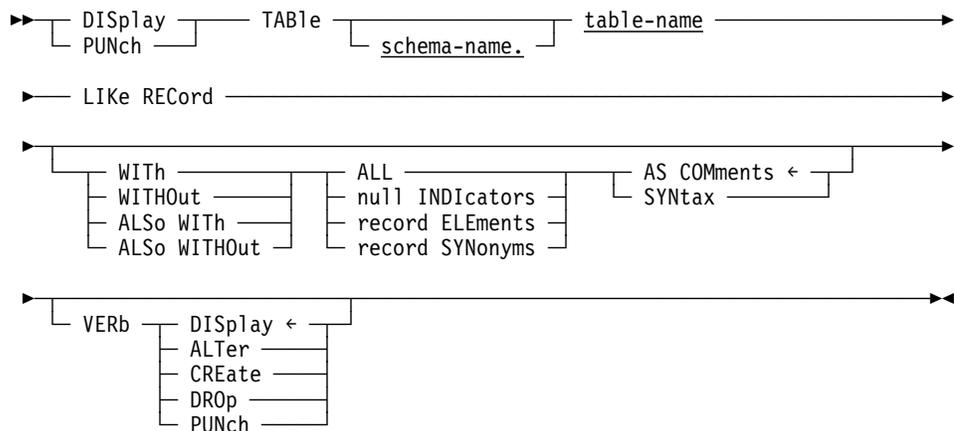
Purpose: Displays or punches the definition of a base table from the dictionary.

Authorization: To issue a DISPLAY/PUNCH TABLE statement, you must either own or have the DISPLAY privilege on the named table.

Syntax



In IDD record format with COBOL elements



Parameters

TABLE table-name

Specifies the name of the table to display or punch. *table-name* must be the name of a table defined in the dictionary.

schema-name.

Identifies the SQL schema associated with the named table.

If you do not specify *schema-name*, it defaults to the current schema associated

with your SQL session, if the statement is entered through the command facility or executed dynamically.

FULI

Specifies that you want syntax for the table and any CALC key, index, and referencing constraint definitions associated with the named table.

WITH

Lists the requested information, in addition to the information that is always included, such as the entity occurrence name.

WITHOut

Does *not* list the specified options. *Other* options in effect through the WITH or ALSO WITH clauses in the current DISPLAY statement are displayed.

ALSo WITH

Lists the requested information, in addition to the information requested in previously issued DISPLAY WITH and DISPLAY ALSO WITH statements for the named entity.

ALSo WITHOut

Does *not* list the specified options.

ALL

Specifies the display of all of the information associated with the requested table.

NONE

Specifies the display of the name of the requested table. NONE is meaningful only when the WITH clause is specified.

DETailS

Specifies the display of table-specific descriptions; for example, the length of a table.

HIStory

Specifies the display of the chronological account of a table's existence, including PREPARED/REVISED BY specifications, date created, and date last updated.

CALc keys

Specifies the display of a CALC key associated with the requested table occurrence.

COLumns

Specifies the display of all columns associated with the requested table occurrence.

CONstraints

Specifies the display of all constraints in which the requested table occurrence has been named.

INDEXes

Specifies the display of all indexes associated with the requested table occurrence.

VIEWS

Specifies the display of all views in which the requested table occurrence participates.

AS COMments

Outputs table syntax as comments with the characters *+ preceding the text of the statement. AS COMMENTS is the default.

AS SYNtax

Outputs table syntax which can be edited and resubmitted to the command facility.

VERB CREate/ALTER/DISplay/DROp/PUNCh

Specifies the verb with which the entity statement is to be displayed or punched. For example, if VERB CREATE is specified, the output of the DISPLAY/PUNCH statement is a CREATE statement; if VERB ALTER is specified, the output is an ALTER statement; and so on. The default is VERB CREATE.

With COBOL elements parameters

LIKe RECord

Specifies that you want IDD RECORD syntax, with its COBOL elements, listed for the named table. For sample uses, see "Usage" later in this section.

null INDicators

Specifies the display of COBOL elements defining NULL indicators for nullable columns.

record ELEments

Specifies the display of elements for the record syntax for the named table.

record SYNonyms

Specifies the display of record synonyms for the record syntax for the named table.

5.2.11 Usage

Using the LIKE RECORD parameter: You can use the LIKE RECORD parameter to produce IDD record syntax for a named table, and then add the record syntax to a dictionary.

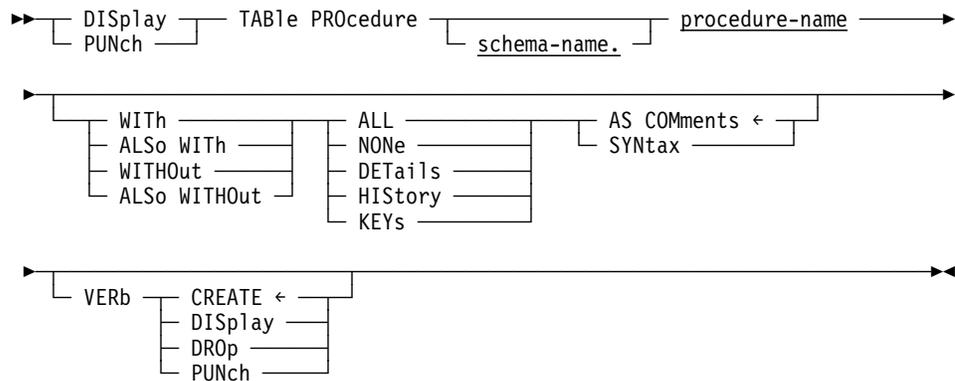
With the IDD record syntax for a table in the dictionary, CA-ADS dialogs can include a work record definition for the table. This same record definition can be included in a map definition.

5.2.12 DISPLAY/PUNCH TABLE PROCEDURE

Purpose: Displays or punches a table procedure.

Authorization: To issue a DISPLAY TABLE PROCEDURE statement, you must have the DISPLAY privilege for the named table procedure.

Syntax



Parameters

schema-name.

Identifies the SQL schema associated with the named table procedure.

If you do not specify *schema-name*, it defaults to the current schema associated with your SQL session, if the statement is entered through the command facility or executed dynamically.

procedure-name

Specifies the name of the table procedure to display or punch. *Procedure-name* must be the name of a table procedure defined in the dictionary.

WITH

Lists the requested information, in addition to the information that is always included, such as the entity occurrence name.

WITHOUT

Does *not* list the specified options. *Other* options in effect through the WITH or ALSO WITH clauses in the current DISPLAY statement are displayed.

ALSO WITH

Lists the requested information, in addition to the information requested in previously issued DISPLAY WITH and DISPLAY ALSO WITH statements for the named entity.

ALSO WITHOUT

Does *not* list the specified options.

ALL

Specifies the display of all of the information associated with the requested entity occurrence.

NONE

Specifies the display of the name of the requested entity occurrence. NONE is meaningful only when the WITH clause is specified.

DETAILS

Specifies the display of entity-specific descriptions; for example, the length of a table.

HIStory

Specifies the display of the chronological account of an entity's existence, including PREPARED/REVISED BY specifications, date created, and date last updated.

KEYs

Specifies the display of all keys associated with the requested table procedure.

AS COMments

Outputs table procedure syntax as comments with the characters *+ preceding the text of the statement. AS COMMENTS is the default.

AS SYNtax

Outputs table procedure syntax which can be edited and resubmitted to the command facility.

VERB CREate/DISplay/DROp/PUNCh

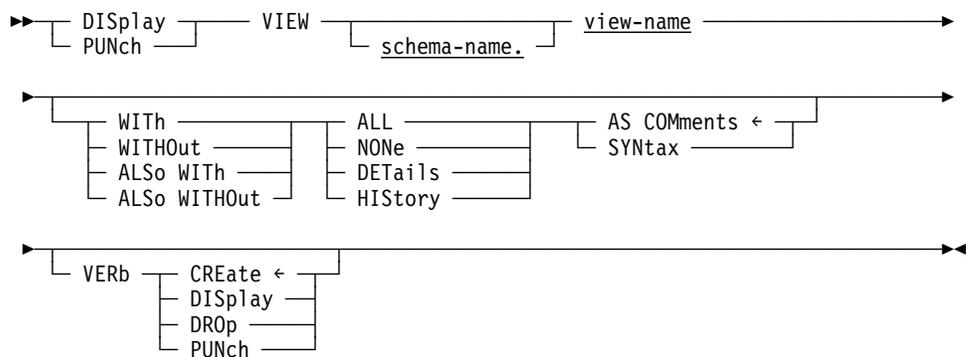
Specifies the verb with which the entity statement is to be displayed or punched. For example, if VERB CREATE is specified, the output of the DISPLAY/PUNCH statement is a CREATE statement; if VERB ALTER is specified, the output is an ALTER statement; and so on. The default is VERB CREATE.

5.2.13 DISPLAY/PUNCH VIEW

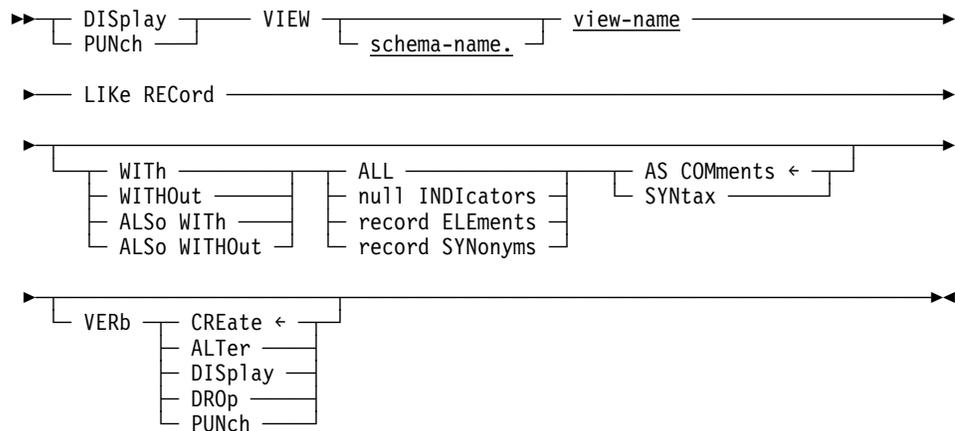
Purpose: Displays or punches a view.

Authorization: To issue a DISPLAY VIEW statement, you must either own the SQL schema in which the view is defined or hold the DISPLAY privilege on the named view.

Syntax



In IDD record format with COBOL elements



Parameters

VIEW view-name

Specifies the name of the view to display or punch.

schema-name.

Identifies the SQL schema associated with the named view.

If you do not specify *schema-name*, it defaults to the current schema associated with your SQL session, if the statement is entered through the command facility or executed dynamically.

WITH

Lists the requested information, in addition to the information that is always included, such as the entity occurrence name.

WITHOut

Does *not* list the specified options. *Other* options in effect through the WITH or ALSO WITH clauses in the current DISPLAY statement are displayed.

ALSo WITH

Lists the requested information, in addition to the information requested in previously issued DISPLAY WITH and DISPLAY ALSO WITH statements for the named entity.

ALSo WITHOut

Does *not* list the specified options.

ALL

Specifies the display of all of the information associated with the requested entity occurrence.

NONE

Specifies the display of the name of the requested entity occurrence. NONE is meaningful only when the WITH clause is specified.

DETailS

Specifies the display of entity-specific descriptions; for example, the length of a table.

HIStory

Specifies the display of the chronological account of an entity's existence, including PREPARED/REVISED BY specifications, date created, and date last updated.

AS COMments

Outputs view syntax as comments with the characters *+ preceding the text of the statement. AS COMMENTS is the default.

AS SYNtax

Outputs view syntax which can be edited and resubmitted to the command facility.

VERB CREate/DISplay/DROp/PUNCh

Specifies the verb with which the entity statement is to be displayed or punched. For example, if VERB CREATE is specified, the output of the DISPLAY/PUNCH statement is a CREATE statement; if VERB ALTER is specified, the output is an ALTER statement; and so on. The default is VERB CREATE.

With COBOL elements parameters

LIKe RECord

Specifies that you want IDD RECORD syntax, with its columns as COBOL elements, listed for the named view. For sample uses, see "Usage" later in this section.

null INDicators

Specifies the display of COBOL elements defining NULL indicators for nullable columns.

record ELEments

Specifies the display of elements for the record syntax for the named view.

record SYNonyms

Specifies the display of record synonyms for the record syntax for the named view.

5.2.14 Usage

Using the LIKE RECORD parameter: The LIKE RECORD parameter produces IDD record syntax for the named table.

You can use this syntax to define a record definition for a view in the dictionary. CA-ADS dialogs can then include it as a work record definition for the view. This same record definition can be included in a map definition.

5.3 Dynamic SQL syntax changes

CA-IDMS Release 14.0 has extended its SQL support to include the following new features:

- You can update or delete the current row of a dynamic cursor
- You can define and manipulate statements and cursors with dynamically-supplied names
- You can define and manipulate global statements and cursors
- You can use dynamic parameters to supply input values for dynamically executed statements

5.3.1 Dynamic positioned UPDATE and DELETE

Positioned UPDATE and DELETE statements may now reference dynamically-defined cursors and may themselves be dynamically PREPARED and EXECUTED. This provides the ability to update or delete the current row in a set of rows, the makeup of which is not known until runtime.

To use this new feature, you do the following:

- PREPARE a SELECT statement that defines the rows to be updated or deleted.
- Define a dynamic cursor that references the PREPARED statement using either a DECLARE CURSOR statement or the new ALLOCATE CURSOR statement.
- OPEN the cursor.
- Position the cursor on the target row using a FETCH statement.
- EXECUTE the positioned UPDATE or DELETE statement in one of the following ways:
 - As a statement embedded in an application program
 - Through the EXECUTE IMMEDIATE statement
 - Through the PREPARE and EXECUTE statements

For more information on this feature, see the UPDATE and DELETE statements later in this chapter.

5.3.2 Dynamically-assigned names

What are they?: Dynamically-prepared statements and their associated cursors may now have names assigned at runtime rather than at compile time. This allows a single set of source statements (OPEN, CLOSE, FETCH, and so on) to reference more than one concurrently-open cursor and it also allows multiple cursors to be associated with a single PREPARED statement. These new capabilities may simplify the coding and design of certain types of applications.

Using dynamically-assigned names: To use this new feature, you specify a host-variable in place of the cursor or statement name. For example, to open a cursor whose name is assigned at runtime, you would code the following:

```
OPEN :cursor-name
```

Before executing this statement, you would move the desired name of the cursor to the *:cursor-name* host variable.

Cursors with dynamically-assigned names must be defined using the new `ALLOCATE CURSOR` statement instead of the `DECLARE CURSOR` statement. The following example creates a new cursor whose name is supplied in *:curs-name* and associates it with the statement whose name is supplied through the *:sel-statemnt* host variable:

```
ALLOCATE :curs-name CURSOR FOR :sel-statemnt
```

For more information on this feature, refer to the expansions of cursor-name and statement-name, as well as the new `ALLOCATE CURSOR` and `DEALLOCATE PREPARE` statements later in this chapter. In addition, refer to the modified description of the following statements to see how these new types of names can be used:

- CLOSE
- DELETE
- DESCRIBE
- EXECUTE
- FETCH
- OPEN
- PREPARE
- UPDATE

5.3.3 Global statements and cursors

What are they?: Dynamically defined statements and cursors may now be created as global objects. This allows them to be referenced by any program executing as part of the transaction in which they were created. This facilitates the use of modular programming by allowing an entity, such as a statement, to be created in one program and referenced, perhaps by an `EXECUTE` statement, in another program.

Defining a global statement or cursor: To define a global statement or cursor, you specify the keyword `GLOBAL` as part of the name when the object is created. To define a global statement, you specify `GLOBAL` on the `PREPARE` statement which defines it:

```
PREPARE GLOBAL :statement-name FOR :statement-text
```

Defining a dynamic global cursor: To define a dynamic global cursor, you use the new `ALLOCATE CURSOR` statement and not the `DECLARE CURSOR` statement:

```
ALLOCATE GLOBAL :cursor-name CURSOR FOR :statement-name
```

Once defined, global statements and cursors are referenced by specifying the GLOBAL keyword as part of the name. For example, to open a global cursor, you code:

```
OPEN GLOBAL :cursor-name
```

For more information on the feature, see the expansions of cursor-name and statement-name, as well as the new ALLOCATE CURSOR and DEALLOCATE PREPARE statements later in this chapter. In addition, see the modified description of the following statements to see how these new types of names can be used:

- CLOSE
- DELETE
- DESCRIBE
- EXECUTE
- FETCH
- OPEN
- PREPARE
- UPDATE

5.3.4 Dynamic parameters

What are they?: Dynamic parameters allow input values to be supplied during the execution of a dynamic SQL statement. This allows a statement, such as an UPDATE or INSERT statement, to be PREPARED once but EXECUTEd multiple times with different input values for each execution. It also allows a SELECT statement to be PREPARED once but to be used with different selection criteria to retrieve different rows.

How are they used?: You indicate the presence of a dynamic parameter by specifying a dynamic parameter marker within the text of the SQL statement being prepared. A dynamic parameter marker is the question mark ("?") symbol. It can be specified anywhere that an input host variable can be specified, except as noted below.

When executing an SQL statement which contains one or more dynamic parameter markers, you supply values to be substituted in place of the markers through the USING clause on the EXECUTE statement. If the prepared SQL statement is a SELECT, the substitution values are supplied through the USING clause on the OPEN statement.

Parameter datatypes: When a statement containing a dynamic parameter marker is prepared, CA-IDMS infers the datatype of the substitution value by examining the context in which the dynamic parameter marker appears. You may use the DESCRIBE statement (or the DESCRIBE option on the PREPARE statement) to determine the assumptions that CA-IDMS has made about the datatypes of the dynamic parameters.

The datatypes of the actual substitution values do not need to be the same as those assumed by CA-IDMS. However, they must be compatible with respect to the assignment operator. That is, the value passed at the time the statement is executed must be capable of being assigned to a variable of the datatype assumed by CA-IDMS.

►►Refer to Comparison, Assignment, Arithmetic and Concatenation Operations in the *CA-IDMS SQL Reference* for more information on the assignment operation.

The following table outlines how CA-IDMS infers the datatype of a dynamic parameter from the context in which it is used.

Context	Datatype of dynamic parameter
Date-time value expressions	
? + date or date + ?	Date duration (DECIMAL(8,0))
? + time or time + ?	Time duration (DECIMAL(6,0))
? + timestamp or timestamp + ?	Time duration (DECIMAL(6,0))
date - ?	Date duration (DECIMAL(8,0))
time - ?	Time duration (DECIMAL(6,0))
timestamp - ?	Time duration (DECIMAL(6,0))
? - date	DATE
? - time	TIME
? + labelled duration or ? - labelled duration	DATE if duration is DAY, MONTH, YEAR; TIME if duration is HOUR, MINUTE, SECOND
v + ? DAY/MONTH/YEAR/ HOUR/MINUTE/SECOND	DECIMAL(31,6)
Other value expressions	
? arithmetic-operator v or v arithmetic-operator ?	Same as v
? v or v ?	VARCHAR (256)
Scalar functions	
CAST (? AS <u>data-type</u>)	<u>data-type</u>
CHAR_LENGTH (?)	VARCHAR (256)
CHARACTER_LENGTH (?)	VARCHAR (256)
COALESCE (v,...?,...)	Same as v (The first entry in the list cannot be a dynamic parameter)
FLOAT (?)	DOUBLE PRECISION

Context	Datatype of dynamic parameter
HEX (?)	VARCHAR (256)
INTEGER (?)	INTEGER
LEFT (?, ?)	First is VARCHAR (256); second is INTEGER
LENGTH (?)	VARCHAR (256)
LOCATE (?, ?, ?)	First and second are VARCHAR (256); third is INTEGER
LOWER (?)	VARCHAR (256)
LTRIM (?)	VARCHAR (256)
POSITION (? IN ?)	Both are VARCHAR (256)
PROFILE (?)	VARCHAR (256)
RTRIM (?)	VARCHAR (256)
SUBSTR (?, ?, ?) or SUBSTRING (? FROM ? FOR ?)	First is VARCHAR (256); second and third are INTEGER
TRIM (? FROM ?)	Both are VARCHAR (256)
UCASE (?)	VARCHAR (256)
UPPER (?)	VARCHAR (256)
VALUE (v,...,?,...)	Same as <i>v</i> (The first entry in the list cannot be a dynamic parameter)
VARGRAPHIC (?)	VARCHAR (256)
Predicates	
? comparison-operator <i>v</i> or <i>v</i> comparison-operator ?	Same as <i>v</i>
? LIKE ? ESCAPE ?	All are VARCHAR (256)
? BETWEEN <i>v1</i> AND <i>v2</i>	Same as <i>v1</i>
<i>v</i> BETWEEN ? AND ?	Both same as <i>v</i>
<i>v1</i> IN (<i>v2</i> ,...,?,...)	Same as <i>v1</i>
? IN (<i>v1</i> , <i>v2</i> , ...)	Same as <i>v1</i>
? comparison-operator ANY/ALL (subquery)	Same datatype as result of subquery.
? comparison-operator (subquery)	Same datatype as result of subquery.
Update values	
UPDATE ... SET <i>column</i> = ?	Same as <i>column</i> .

Context	Datatype of dynamic parameter
INSERT ... VALUES (...?,...)	Same as target column.

Note: Dynamic parameters are always nullable.

Datatype conversion considerations: CA-IDMS uses the above rules to infer a datatype for a dynamic parameter. The actual value of the parameter may have a different datatype provided the two are compatible with regard to the assignment operator. However, in certain cases, compatibility may not be sufficient. For example, if you wish to supply a very long character string as an input value and CA-IDMS has inferred a datatype of VARCHAR(256), the input value may be truncated to a length of 256. To circumvent this, you may always use the CAST function to override the default datatype, as in the following example:

```
UPDATE MY.TEXT
  SET STRING =
    CHAR(CURRENT_TIMESTAMP) || '▶▶' || CAST (? AS VARCHAR(1000))
  WHERE ...
```

As an operand of a concatenate symbol, CA-IDMS would normally assign VARCHAR (256) as the datatype for the dynamic parameter. However, by using a CAST function, the parameter is instead assigned a datatype of VARCHAR (1000).

Restrictions in the use of dynamic parameters: Dynamic parameter markers may not be used in the following contexts:

- Following a unary + or - operator
- By itself as an entry in the *select-list* of a query expression
- As both operands of a dyadic operator (except for the concatenation operator)
- As the first entry in the operand list of the COALESCE and VALUE functions
- As both the first and second or first and third operands of a BETWEEN predicate
- As both the first operand and any entry in the second operand of the IN predicate
- As the operand in the following functions: CHAR, DATE, DAY, DAYS, DECIMAL, DIGITS, MINUTE, MONTH, MICROSECOND, OCTET_LENGTH, SECOND, TIME, TIMESTAMP, YEAR

Tip: The CAST function may be used to assign a datatype to a parameter that otherwise would not be allowed within the desired context. For example, if you wish to use a dynamic parameter as the first operand in the VALUE function, you may embed the parameter in a CAST function in order to assign a default datatype.

New statement options: Refer to new options that make use of dynamic parameters on the following statements:

- DESCRIBE
- EXECUTE

- OPEN
- PREPARE

5.3.5 Dynamic SQL statements and expressions

The rest of this section describes new and modified SQL statements that support dynamic SQL. These statements are:

- Expansions of:
 - Cursor-name
 - Cursor-specification
 - Query-expression
 - Statement-name
- ALLOCATE CURSOR
- CLOSE
- DEALLOCATE PREPARE
- DELETE
- DESCRIBE
- EXECUTE
- FETCH
- OPEN
- PREPARE
- UPDATE

5.3.6 Expansion of cursor-name

Purpose: Represents a cursor.

Syntax

►► static-cursor-name —————►►
 └── extended-cursor-name ─┘

Expansion of static-cursor-name

►► cursor-name —————►►

Expansion of extended-cursor-name

►► ┌── LOCAL ← ─┘ ┌── 'cursor-name' ─┘ —————►►
 └── GLOBAL ─┘ └── :host-variable ─┘

Parameters

cursor-name

Specifies the name of the cursor as an identifier.

'cursor-name'

Specifies the name of the cursor as a literal whose value must conform to the rules for an identifier.

:host-variable

Specifies the name of the cursor as a host-variable whose value must conform to the rules for an identifier.

LOCAL/GLOBAL

Specifies the scope of the associated cursor name:

- LOCAL indicates that the cursor can be referenced only from within the program in which it is defined.
- GLOBAL indicates that the cursor can be referenced from any program executing within the same SQL transaction.

LOCAL is the default.

5.3.7 Usage

Static versus extended cursor names: A static cursor name is one coded as a simple identifier. The following DECLARE CURSOR statement assigns the static name "cursor1" to the cursor being defined:

```
DECLARE cursor1 CURSOR FOR select1
```

Cursors defined by a DECLARE CURSOR statement always have static names. Such cursors may either be dynamic or static, depending on whether the DECLARE CURSOR statement references a dynamically prepared SQL statement, as in the example above, or directly includes a cursor-specification.

An extended cursor name is one coded either as a literal or as a host variable. The following ALLOCATE CURSOR statement assigns the extended name "cursor1" to the cursor being defined:

```
MOVE 'cursor1' to cursor-nam  
ALLOCATE :cursor-nam CURSOR FOR :statement-nam
```

Cursors created by an ALLOCATE CURSOR statement always have extended names and are always dynamic.

If a cursor is defined using a static name, it must be referenced using a static name; similarly, if it is defined using an extended name, it must be referenced using an extended name that has the same scope option as specified on the definition.

Note: An exception to this rule occurs when identifying a cursor within a dynamically prepared UPDATE or DELETE statement. For more information, see the UPDATE and DELETE statements later in this chapter.

Uniqueness of cursor names: Static and extended cursor names do not have to be unique with respect to each other. If two cursors are assigned the same value for a name, they are considered two separate cursors provided that either:

- One of the names is static while the other is extended
- Both of the names are extended, but they have different scopes, as indicated by their LOCAL/GLOBAL parameter.

5.3.8 Example

Example of cursor-name: The following DECLARE CURSOR statement defines a dynamic cursor using a static cursor name of C1. It is referenced within the subsequent OPEN statement:

```
EXEC SQL
  DECLARE C1 CURSOR FOR S1
END-EXEC
EXEC SQL
  OPEN C1
END-EXEC
```

Example of extended cursor name: The following ALLOCATE CURSOR statement defines a local cursor using an extended cursor name of C1. It is then referenced in the subsequent OPEN statement:

```
EXEC SQL
  ALLOCATE 'C1' CURSOR FOR 'S1'
END-EXEC
EXEC SQL
  OPEN 'C1'
END-EXEC
```

Note: Even though C1 is used as the cursor name in both of the above examples, two separate cursors are created: one with a static name of C1 and one with an extended name of C1.

Global extended cursor name: The following ALLOCATE CURSOR statement defines a global cursor using an extended cursor name whose value is not known until runtime. In this case, the value 'C2' is moved to the host variable before the statement is executed and will be the name of the cursor created:

```
      move 'C2' to :cname
EXEC SQL
  ALLOCATE GLOBAL :CNAME CURSOR FOR :SNAME
END-EXEC
```

Since this is a global cursor, it can be referenced in a different program than the one in which the ALLOCATE CURSOR statement appears. For example, the following OPEN statement might be contained in a different program:

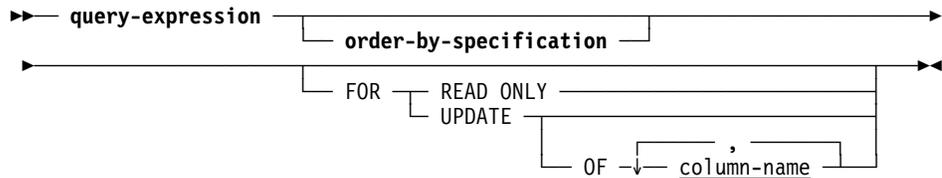
```
EXEC SQL
  OPEN GLOBAL 'C2'
END-EXEC
```

Note: It does not matter that in one case the name of the cursor is supplied through a host-variable and in the other it is specified as a literal. They both refer to the global cursor C2.

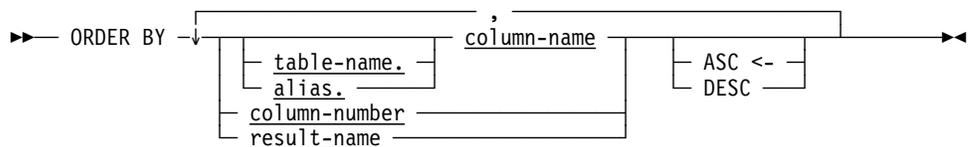
5.3.9 Expansion of cursor-specification

Purpose: Represents the body of a cursor-definition.

Syntax



Expansion of order-by-specification



Parameters

query-expression

Represents a table resulting from the evaluation of a query-expression.

►► For expanded syntax, see Expansion of query-expression in the *CA-IDMS SQL Reference*

order-by-specification

Specifies a sort order for the rows in the result table defined by query-expression.

Expanded syntax for order-by-specification is shown above, immediately following the cursor-specification syntax.

ORDER BY

Sorts the rows in the result table defined by query-expression in ascending or descending order by the values in the specified columns. Rows are ordered first by the first column specified, then by the second column specified with the ordering established by the first column, then by the third column specified, and so on.

column-name

Specifies a sort column by name. *Column-name* must identify a column in the result table of the query expression.

table-name

Specifies the table or view that includes the named column. See "Identifying entities in schemas" in Chapter 2 of the *CA-IDMS SQL Reference* for expanded table-name syntax.

alias

Specifies the alias associated with the table or view that includes the named column. *Alias* must be defined in the FROM parameter of the query specification that makes up the query expression.

column-number

Specifies a sort column by the position of the column in the result table. The first result column is in position 1.

Column-number must be an integer in the range 1 through the number of columns in the result table.

result-name

Specifies the sort column by the result name specified in the AS parameter of the query expression.

ASC

Indicates that the values in the specified column are to be sorted in ascending order. ASC is the default.

DESC

Indicates that the values in the specified column are to be sorted in descending order.

FOR READ ONLY

Specifies that the cursor with which this *cursor-expression* is associated will be used for retrieval operations only. If specified, it prohibits the execution of both positioned UPDATEs and DELETEs that reference the cursor.

FOR UPDATE

Specifies that the cursor will be used for positioned UPDATE operations.

OF column-name

Identifies a column that may be updated through positioned UPDATE statements. If no columns are specified, then all columns in the table may be updated.

5.3.10 Usage

Updatable cursors: A cursor defined by a cursor specification is updatable if the cursor specification:

- Contains an updatable query-expression
- Does not contain an ORDER BY clause
- Does not contain a FOR READ ONLY clause

Updatable cursors may be referenced in positioned DELETE statements.

To reference a cursor in a positioned UPDATE statement, it must be updatable and the FOR UPDATE clause must be specified within the cursor specification.

Tip: To insure optimal performance when processing a cursor that is referenced in a positioned UPDATE statement, you should explicitly identify the columns to be updated rather than specifying FOR UPDATE without naming the columns.

Updatable query-expressions: A query expression is updatable if:

- It consists of a single query specification; that is, the query expression does not include the UNION operator
- The FROM parameter in the query specification specifies only one table, view, or table procedure, and if a view is specified, it is updatable.
- The query specification does not contain DISTINCT, PRESERVE, GROUP BY, or HAVING parameters and does not use aggregate functions in the specification of a result column

5.3.11 Example

Defining a cursor for retrieval-only: The following DECLARE CURSOR statement defines a static cursor by including a cursor-specification directly. In this case, the cursor being defined can only be used to retrieve rows from the database. By coding the FOR READ ONLY option, you ensure that neither positioned UPDATES nor DELETES will be allowed against the cursor:

```
EXEC SQL
  DECLARE EMP_CURSOR CURSOR FOR
    SELECT EMP_ID, DEPT_ID, EMP_LNAME
      FROM EMPLOYEE
      FOR READ ONLY
END-EXEC
```

Defining a dynamic cursor for UPDATE operations: The following set of code defines a dynamic cursor to be used to update the DEPT_ID column of the EMPLOYEE table. The cursor-specification containing the FOR UPDATE clause is first prepared and then an ALLOCATE CURSOR statement is used to create the cursor:

```
MOVE 'SELECT *
      FROM EMPLOYEE FOR UPDATE OF DEPT_ID'
      TO ST-TEXT.

EXEC SQL
  PREPARE 'EMP-STATEMENT' FROM :ST-TEXT
END-EXEC

EXEC SQL
  ALLOCATE 'EMP-CURSOR' CURSOR FOR 'EMP-STATEMENT'
END-EXEC
```

5.3.12 Expansion of statement-name

Purpose: Represents a dynamically-prepared statement.

Syntax

►► static-statement-name
extended-statement-name ►►

Expansion of static-statement-name

►► statement-name ►►

Expansion of extended-statement-name

►► LOCAL ←
GLOBAL 'statement-name'
:host-variable ►►

Parameters

statement-name

Specifies the name of the statement as an identifier.

'statement-name'

Specifies the name of the statement as a literal whose value must conform to the rules for an identifier.

:host-variable

Specifies the name of the statement as a host-variable whose value must conform to the rules for an identifier.

LOCAL/GLOBAL

Specifies the scope of the associated statement name:

- LOCAL indicates that the statement can be referenced only from within the program in which it is prepared.
- GLOBAL indicates that the statement can be referenced from any program executing within the same SQL transaction.

The default is LOCAL.

5.3.13 Usage

Static versus extended statement names: A static statement name is one coded as a simple identifier. The following PREPARE statement assigns the static name "select1" to the statement being prepared:

```
PREPARE select1 from :select1-text
```

An extended statement name is one coded either as a literal or as a host-identifier:

```
MOVE 'SELECT1' to statement-nam
PREPARE :statement-nam FROM :select1-text
```

If a statement is prepared using a static name, it must be referenced using a static name; similarly, if it is prepared using an extended name, it must be referenced using an extended name that has the same scope option.

Uniqueness of statement names: Static and extended names do not have to be unique with respect to each other. If two statements are assigned the same value for a name, they are considered two separate statements provided that either:

- One of the names is static while the other is extended.
- Both of the names are extended, but they have different scopes, as indicated by their LOCAL/GLOBAL parameter.

5.3.14 Example

Static statement name: The following PREPARE statement creates a statement using a static statement name of S1. It is referenced within the subsequent DESCRIBE statement:

```
EXEC SQL
  PREPARE S1 FROM :TEXT
END-EXEC
EXEC SQL
  DESCRIBE S1 USING DESCRIPTOR SQLDA
END-EXEC
```

Extended statement name: The following PREPARE statement creates a local statement using an extended statement name of S1. It is then referenced in the subsequent DESCRIBE statement:

```
EXEC SQL
  PREPARE 'S1' FROM :TEXT
END-EXEC
EXEC SQL
  DESCRIBE 'S1' USING DESCRIPTOR SQLDA
END-EXEC
```

Note: Even though S1 is used as the statement name in both of the above examples, two separate statements are created: one with a static name of S1 and one with an extended name of S1.

Global extended statement name: The following PREPARE statement creates a global statement using an extended statement name whose value is not known until runtime. In this case, the value 'S2' is moved to the host variable before the statement is executed and will be the name of the statement created:

```
MOVE 'S2' TO :SNAME
EXEC SQL
  PREPARE GLOBAL :SNAME FROM :TEXT
END-EXEC
```

Since this is a global statement, it can be referenced in a different program than the one in which the PREPARE statement appears. For example, the following DESCRIBE statement might be contained in a different program:

```
EXEC SQL
  DESCRIBE GLOBAL 'S2'
END-EXEC
```

Note: It does not matter that in one case the name of the statement is supplied through a host-variable and in the other it is specified as a literal. They both refer to the global statement S2.

5.3.15 ALLOCATE CURSOR statement

Purpose: Defines a cursor for a dynamically-prepared statement.

Syntax

```
►►— ALLOCATE extended-cursor-name CURSOR —————►
►— FOR extended-statement-name —————►
```

Parameters

extended-cursor-name

Identifies the name of the cursor being defined. The name must conform to the rules for an identifier and must be unique within the specified scope.

extended-statement-name

Identifies the name of the statement for which the cursor is being defined. A statement with this name and scope must have been prepared within the same SQL transaction as that in which the ALLOCATE CURSOR statement is being executed.

5.3.16 Usage

Updatable cursors: The PREPARED statement referenced in the ALLOCATE CURSOR statement must be a cursor-specification. The cursor created as a result of the ALLOCATE CURSOR statement, is updatable, if the cursor-specification is updatable.

5.3.17 Examples

Creating a local cursor: The following ALLOCATE CURSOR statement creates a local cursor called C1 and associates it with the local statement whose name is passed in :sname:

```
EXEC SQL
  ALLOCATE 'C1' CURSOR FOR :SNAME
END-EXEC
```

Creating a global cursor: The following ALLOCATE CURSOR statement creates a global cursor whose name is passed in :CNAME and associates it with the global statement whose name is passed in :SNAME:

```
EXEC SQL
  ALLOCATE GLOBAL :CNAME CURSOR FOR :SNAME
END-EXEC
```

Sharing a statement definition: The following two ALLOCATE CURSOR statements create two cursors, one of which is local and one of which is global. They are both associated with the same local statement:

```
EXEC SQL
  ALLOCATE 'C1' CURSOR FOR 'S1'
END-EXEC
EXEC SQL
  ALLOCATE GLOBAL CURSOR 'G1' FOR 'S1'
END-EXEC
```

5.3.18 CLOSE statement

Syntax

►► — CLOSE **cursor-name** —————►►

Parameters

cursor-name

Specifies the cursor to be closed. Cursor-name must identify a cursor previously defined by a DECLARE CURSOR statement within the application program or by an ALLOCATE CURSOR statement executed within the same SQL transaction.

5.3.19 Example

Closing a global dynamic cursor: The following statement closes the global cursor whose name is passed in :CNAME:

```
EXEC SQL
  CLOSE GLOBAL :CNAME
END-EXEC
```

5.3.20 DEALLOCATE PREPARE statement

Purpose: Destroys a dynamically-compiled statement and all other dynamically compiled statements that directly or indirectly reference it.

Syntax

►► — DEALLOCATE PREPARE **statement-name** —————►►

Parameters

statement-name

Identifies the statement to be destroyed. It must identify a statement previously created using a PREPARE statement.

5.3.21 Usage

Effect on dependent statements: Upon successful execution of a DEALLOCATE PREPARE statement, the following actions have taken place:

- The target statement is destroyed.
- If the target statement was a cursor-specification, then all cursors that reference the target statement are destroyed. If the cursors were open at the time the DEALLOCATE PREPARE statement was executed, they are first closed.
- If any dynamically compiled positioned UPDATE or DELETE statements reference a cursor being destroyed, they too are destroyed.

5.3.22 Examples

Destroying a prepared statement: The following statement destroys both the local statement named S1 and any cursors that reference the statement. It also destroys any statements that reference the cursors.

```
EXEC SQL
  DEALLOCATE PREPARE S1
END-EXEC
```

5.3.23 DELETE statement

Syntax: The complete syntax of the DELETE statement is shown below. The description for only the CURRENT OF parameter is provided. For a description of the complete syntax, see the *CA-IDMS SQL Reference*.

```

▶▶— DELETE FROM table-name alias
                                     └──┬──┘
                                     WHERE search-condition
                                     └──┬──┘
                                     CURRENT OF cursor-name
                                               └──┬──┘
                                               dynamic-name-clause

```

Expansion of dynamic-name-clause

```

▶▶— LOCAL ← cursor-name
      GLOBAL

```

Parameters

CURRENT OF

Specifies that only the row that corresponds to the current row of the named cursor is to be deleted.

cursor-name

Identifies the cursor whose current row will be deleted. Cursor-name must identify an open cursor previously defined by a DECLARE CURSOR statement within the application program or by an ALLOCATE CURSOR statement executed within the same SQL transaction.

Note: This option may only be used in a DELETE statement embedded in an application program.

dynamic-name-clause

Identifies the cursor whose current row will be deleted.

Note: This option may only be used in a DELETE statement dynamically compiled using a PREPARE or EXECUTE IMMEDIATE statement.

LOCAL

Indicates that the named cursor has a local scope and was defined using either a DECLARE CURSOR statement or an ALLOCATE CURSOR statement. The default is LOCAL.

GLOBAL

Indicates that the named cursor was created by an ALLOCATE CURSOR statement and is global in scope.

cursor-name

Specifies the name of the cursor as an identifier. *Cursor-name* must identify an open cursor previously defined by a DECLARE CURSOR statement within the application program or by an ALLOCATE CURSOR statement executed within the same SQL transaction.

5.3.24 Usage

Dynamic positioned deletes: A dynamic positioned DELETE statement is one that references a dynamic cursor. Such a DELETE statement may either be embedded within an application program or created dynamically using a PREPARE or EXECUTE IMMEDIATE statement.

A positioned DELETE statement embedded in an application program may reference either a static cursor or a dynamic cursor. A positioned DELETE statement created dynamically using a PREPARE or EXECUTE IMMEDIATE statement can only reference a dynamic cursor.

Ambiguous cursor references: When a dynamic positioned DELETE statement is being created by a PREPARE or EXECUTE IMMEDIATE statement, it is possible that CA-IDMS may not be able to determine which cursor is being referenced. This will occur if the application program contains a DECLARE CURSOR statement that defines a cursor having the referenced name and the program has also executed an ALLOCATE cursor statement that creates a cursor with the same name and a local scope. Under these conditions, CA-IDMS cannot determine which of the two cursors is being referenced. To avoid such problems, it is advisable to use different names for cursors that are declared from those that are allocated with a local scope.

5.3.25 Examples

A positioned DELETE referencing a DECLARED cursor: The following statement deletes the current row of the cursor C1. C1 may either be a dynamic or static cursor, but it must have been defined using a DECLARE CURSOR statement:

```
EXEC SQL
  DELETE FROM EMPLOYEE WHERE CURRENT OF C1
END-EXEC
```

A positioned DELETE referencing an ALLOCATED cursor: The following statement deletes the current row of a cursor whose name is specified in the variable CNAME. The referenced cursor must have been defined using an ALLOCATE CURSOR statement:

```
EXEC SQL
  DELETE FROM EMPLOYEE WHERE CURRENT OF :CNAME
END-EXEC
```

A dynamically-compiled positioned DELETE statement: The following statement deletes the current row of local cursor C1. C1 may have been defined using either a DECLARE CURSOR statement or an ALLOCATE CURSOR statement. In either case, the cursor name in the DELETE statement is specified as an identifier rather than as a literal or host variable:

```
EXEC SQL
  EXECUTE IMMEDIATE
  'DELETE FROM EMPLOYEE WHERE CURRENT OF LOCAL C1'
END-EXEC
```

Note: The keyword LOCAL is unnecessary since it is the default. Regardless of whether or not it is specified, if two local cursors named C1 have been defined, one using a DECLARE CURSOR statement and one using an ALLOCATE CURSOR statement, the EXECUTE IMMEDIATE statement will fail on an ambiguous cursor error.

5.3.26 DESCRIBE statement

Syntax

```

▶▶ DESCRIBE [ OUTPUT ← | INPUT ] statement-name →
▶ USING sql DESCRIPTOR descriptor-area-name1 →
▶ [ INPUT | OUTPUT ] USING sql DESCRIPTOR descriptor-area-name2 →

```

Note 1: If DESCRIBE OUTPUT is specified or implied, you may only specify the INPUT USING parameter; similarly, if DESCRIBE INPUT is specified, you may only specify the OUTPUT USING parameter.

Note 2: For compatibility with earlier releases, you can specify "INTO sql descriptor" in place of "USING sql DESCRIPTOR"; however, this is an extension to ANSI standard SQL.

Parameters

INPUT/OUTPUT

Specifies the type of information to be returned in the associated descriptor area. INPUT means that information about dynamic parameters is to be returned in the SQL descriptor area. OUTPUT means that information about output values is to be returned.

statement-name

Specifies the name of the statement being described.

►►See the expansion of statement-name earlier in this chapter for a detailed description.

USING SQL DESCRIPTOR

Specifies the SQL descriptor area in which CA-IDMS is to return information about the named statement.

descriptor-area-name1

Directs CA-IDMS to use the named area as the descriptor area. *Descriptor-area-name1* must identify an SQL descriptor area.

INPUT/OUTPUT USING SQL DESCRIPTOR descriptor-area-name2

Specifies the type of information to be returned in the associated descriptor area. INPUT means that information about dynamic parameters is to be returned in the SQL descriptor area. OUTPUT means that information about output values is to be returned.

Descriptor-area-name2 is the name of the SQL descriptor area.

Note: If DESCRIBE OUTPUT is specified or implied, you may only specify the INPUT USING parameter; similarly, if DESCRIBE INPUT is specified, you may only specify the OUTPUT USING parameter.

►►See Appendix D, "SQL Descriptor Area," in the *CA-IDMS SQL Reference* for the layout of an SQL descriptor area.

5.3.27 Usage

Describing dynamic parameters: The INPUT option is used to return information about dynamic parameters that may be embedded in the SQL statement being described. The SQLD field of the descriptor area indicates the number of dynamic parameters that appear in the statement. If no dynamic parameters are used, this field is zero (0).

If dynamic parameters do appear in the statement, CA-IDMS returns descriptions of the parameters in the descriptor area. The datatype information is derived from the context in which the dynamic parameter appears.

Describing output values: The OUTPUT option is used to return information about values output from CA-IDMS:

- For a SELECT statement, CA-IDMS returns a description of the result table

COBOL: *Host-variable* may be either an elementary item or a non-bulk structure. If a non-bulk structure is specified, each subelement of the structure is counted as a host variable.

:dyn-buff

Identifies the host-variable or bulk-buffer from which CA-IDMS is to retrieve values for the dynamic parameters.

Dyn-buff must identify a host variable previously declared in the host language application program.

The size of *dyn-buff* must be sufficient to hold a complete set of dynamic parameter values for a single execution of the statement. If specified as part of the BULK parameter, *dyn-buff* must be sufficient to hold *row-count-variable* sets of dynamic parameters. The format of the data in *dyn-buff* must conform to the description in the SQL descriptor area specified by *descriptor-area-name*

SQL DESCRIPTOR

Specifies the SQL descriptor area that describes the format of the dynamic parameter values contained in *dyn-buff*.

descriptor-area-name

Directs CA-IDMS to use the named area as the descriptor area. *Descriptor-area-name* must identify an SQL descriptor area.

BULK

Directs CA-IDMS to execute the statement one or more times and to use a contiguous storage area to retrieve input values for the dynamic parameters. The specification of BULK is a CA-IDMS extension of ANSI-standard SQL.

Note: BULK may only be specified if the statement being executed is an INSERT statement.

:bulk-buffer

Identifies a host variable from which CA-IDMS is to retrieve one or more sets of input values. *Bulk-buffer* must identify a host variable previously declared in the host-language application program.

Bulk-buffer must be defined as a multiple-occurring structure having the same number of subelements as there are dynamic parameters in the statement.

bulk-options

Optionally specify the location in *bulk-buffer* for the first row fetched and/or the number of rows to be fetched from the result table associated with the cursor. Expanded syntax for bulk-options immediately follows the statement syntax.

START :start-variable-name

Identifies a host variable containing the relative position within the bulk buffer from which CA-IDMS is to retrieve values for the first row in the named table or view. Values in subsequent entries in the bulk buffer are retrieved sequentially to subsequent new rows in the table or view.

Start-variable-name must be a host variable previously declared in the host-language application program. The value in the host variable must be an integer in the natural range of subscripts for arrays in the language in which the application program is written.

If you do not specify the START parameter, CA-IDMS retrieves the values from the first entry in the bulk buffer.

ROWS :row-count-variable-name

Identifies a host variable that specifies the number of rows CA-IDMS is to retrieve from the bulk buffer.

Row-count-variable-name must be a host variable previously declared in the host-language application program. The value in the host variable must be in the range 1 through the number of rows that will fit in the bulk buffer.

If you do not specify the ROWS parameter, CA-IDMS retrieves rows from the array sequentially until reaching the end of the buffer.

dynamic-bulk-specification

Provides specification for inserting one or more rows into a table.

Expanded syntax for dynamic-bulk-specification appears immediately following the expanded syntax for bulk-options. Descriptions of dynamic-bulk-specification parameters appear above.

►► See Appendix D, "SQL Descriptor Area," of the *CA-IDMS SQL Reference* for the layout of an SQL descriptor area.

5.3.29 Usage

Dynamically-compiled cursor-specifications

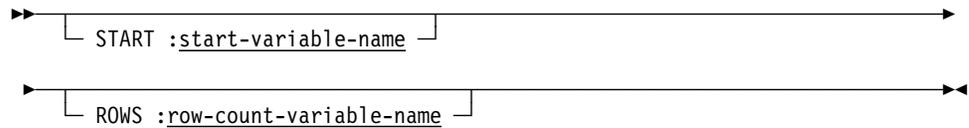
You cannot use the EXECUTE statement with a dynamically-compiled cursor-specification. To retrieve data using a dynamically-compiled cursor-specification, you must define a cursor and use the FETCH statement.

5.3.30 FETCH statement

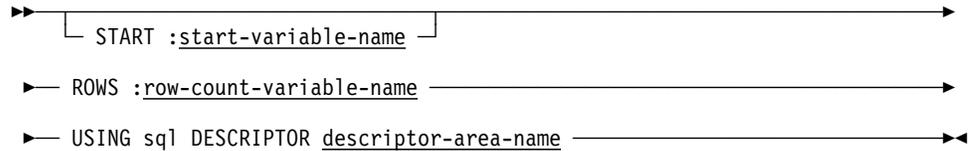
Syntax: The complete syntax of the FETCH statement is shown below. The description for only the cursor-name parameter is provided. For a description of the complete syntax, see the *CA-IDMS SQL Reference*.



Expansion of bulk-options



Expansion of dynamic-bulk-specification



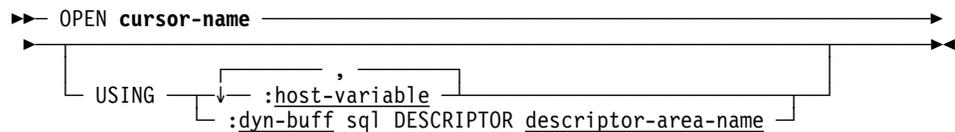
Parameters

cursor-name

Specifies the cursor to be used for retrieving values. Cursor-name must identify an open cursor previously defined by a DECLARE CURSOR statement within the application program or by an ALLOCATE CURSOR statement executed within the same SQL transaction.

5.3.31 OPEN statement

Syntax



Parameters

cursor-name

Specifies the cursor to be opened. Cursor-name must identify a cursor previously defined by a DECLARE CURSOR statement within the application program or by an ALLOCATE CURSOR statement executed within the same SQL transaction.

USING

Supplies values for the dynamic parameters embedded in the text of the dynamically prepared statement with which the cursor is associated.

host-variable

Identifies the host variables from which CA-IDMS is to retrieve values for the dynamic parameters. CA-IDMS assigns the value of the first host variable to the first dynamic parameter, the second host variable to the second dynamic parameter, and so on.

Host-variable must be a host variable declared previously in the host-language application program.

You must specify the same number of host variables in the USING parameter as the number of dynamic parameter markers in the dynamically prepared statement with which the cursor is associated.

COBOL: *Host-variable* may be either an elementary item or a non-bulk structure. If a non-bulk structure is specified, each subelement of the structure is counted as a host variable.

:dyn-buff

Identifies the host-variable from which CA-IDMS is to retrieve values for the dynamic parameters.

Dyn-buff must identify a host variable previously declared in the host language application program.

The size of *dyn-buff* must be sufficient to hold a complete set of dynamic parameter values. The format of the data in *dyn-buff* must conform to the description in the SQL descriptor area specified by *descriptor-area-name*.

SQL DESCRIPTOR

Specifies the SQL descriptor area that describes the format of the dynamic parameter values contained in *dyn-buff*.

descriptor-area-name

Directs CA-IDMS to use the named area as the descriptor area. *Descriptor-name* must identify an SQL descriptor area.

▶▶ See Appendix D, "SQL Descriptor Area," of the *CA-IDMS SQL Reference* for the layout of an SQL descriptor area.

5.3.32 PREPARE statement

Syntax

```
▶▶ PREPARE statement-name FROM [ :statement-text ] →
   [ 'statement-text' ] →
  [ describe-output-expression ] →
  [ describe-input-expression ] →
```

Expansion of describe-output-expression

```
▶▶ DESCRIBE output USING sql DESCRIPTOR descriptor-area-name1 →
  [ INPUT USING sql DESCRIPTOR descriptor-area-name2 ] →
```

Expansion of describe-in-expression

```
▶▶ DESCRIBE INPUT USING sql DESCRIPTOR descriptor-area-name2 →
  [ OUTPUT USING sql DESCRIPTOR descriptor-area-name1 ] →
```

Parameters

statement-name

Specifies the name to be assigned to the compiled statement. It must be unique within its associated scope.

►►For more information, see Expansion of statement-name earlier in this chapter.

FROM

Identifies the statement to be compiled.

:statement-text

Identifies a host variable containing a preparable SQL statement. *:statement-text* must be a host variable previously declared in the application program.

'statement-text'

Specifies a preparable SQL statement enclosed in single quotation marks. Do not include the SQL prefix or terminator within the statement.

DESCRIBE OUTPUT USING SQL DESCRIPTOR descriptor-area-name1

Specifies the SQL descriptor area in which CA-IDMS is to return information about the output values to be returned when the dynamically-compiled statement is executed. *Descriptor-area-name1* is the name of the SQL descriptor area.

INPUT USING SQL DESCRIPTOR descriptor-area-name2

Specifies the SQL descriptor area in which CA-IDMS is to return information about the dynamic parameters used within the statement.

Descriptor-area-name2 is the name of the SQL descriptor area.

DESCRIBE INPUT USING SQL DESCRIPTOR descriptor-area-name2

Specifies the SQL descriptor area in which CA-IDMS is to return information about the dynamic parameters used within the statement.

Descriptor-area-name2 is the name of the SQL descriptor area.

OUTPUT USING SQL DESCRIPTOR descriptor-area-name1

Specifies the SQL descriptor area in which CA-IDMS is to return information about the output values to be returned when the dynamically-compiled statement is executed. *Descriptor-area-name1* is the name of the SQL descriptor area.

►►For more information about the structure of the SQL descriptor area, see Appendix D, "SQL Descriptor Area" of the *CA-IDMS SQL Reference*.

5.3.33 Usage

Preparable statements: The following SQL statements are preparable:

- All authorization and logical data description statements
- COMMIT
- cursor-specification
- DELETE
- EXPLAIN
- INSERT
- RELEASE
- ROLLBACK

- SUSPEND SESSION
- UPDATE

Additionally, all CA-IDMS utility and physical data description statements are preparable.

Specifying dynamic parameters: Dynamic parameters are variables whose values are supplied when the statement is executed, or in the case of a SELECT statement, when its associated cursor is opened.

Dynamic parameters are specified as question marks (?) within the text of the SQL statement. They may appear wherever a host variable is permitted with certain exceptions.

►►For more information on dynamic parameters, see Dynamic parameters earlier in this chapter.

Describing dynamic parameters: The INPUT option is used to return information about dynamic parameters that may be embedded in the SQL statement being described. The SQLD field of the descriptor area indicates the number of dynamic parameter that appear in the statement. If no dynamic parameters are used, this field is zero (0).

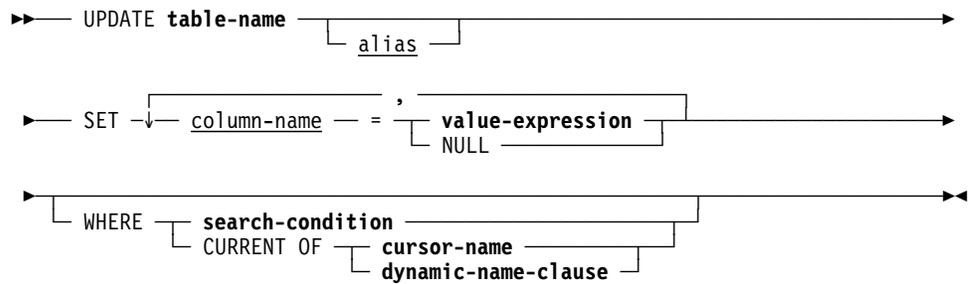
If dynamic parameters do appear in the statement, CA-IDMS returns descriptions of the parameters in the descriptor area. The datatype information is derived from the context in which the dynamic parameter appears.

Describing output values: The OUTPUT option is used to return information about values output from CA-IDMS:

- For a SELECT statement, CA-IDMS returns a description of the result table defined by the statement. The SQLD field of the descriptor area indicates the number of columns in the result table.
- For a statement other than SELECT, CA-IDMS returns the value zero (0) in the SQLD field of the descriptor area.

5.3.34 UPDATE statement

Syntax: The complete syntax of the UPDATE statement is shown below. The description for only the CURRENT OF parameter is provided. For a description of the complete syntax, see the *CA-IDMS SQL Reference*.



Expansion of dynamic-name-clause



Parameters

CURRENT OF

Specifies that only the row that corresponds to the current row of the named cursor is to be updated.

cursor-name

Identifies the cursor whose current row will be updated. Cursor-name must identify an open cursor previously defined by a DECLARE CURSOR statement within the application program or by an ALLOCATE CURSOR statement executed within the same SQL transaction.

Note: This option may only be used in an UPDATE statement embedded in an application program.

dynamic-name-clause

Identifies the cursor whose current row will be updated.

Note: This option may only be used in an UPDATE statement dynamically compiled using a PREPARE or EXECUTE IMMEDIATE statement.

LOCAL

Indicates that the named cursor has a local scope and was defined using either a DECLARE CURSOR statement or an ALLOCATE CURSOR statement. The default is LOCAL.

GLOBAL

Indicates that the named cursor was created by an ALLOCATE CURSOR statement and is global in scope.

cursor-name

Specifies the name of the cursor as an identifier. *Cursor-name* must identify an open cursor previously defined by a DECLARE CURSOR statement within the application program or by an ALLOCATE CURSOR statement executed within the same SQL transaction.

5.3.35 Usage

Dynamic positioned updates: A dynamic positioned UPDATE statement is one that references a dynamic cursor. Such an UPDATE statement may either be embedded within an application program or created dynamically using a PREPARE or EXECUTE IMMEDIATE statement.

A positioned UPDATE statement embedded in an application program may reference either a static cursor or a dynamic cursor. A positioned UPDATE statement created dynamically using a PREPARE or EXECUTE IMMEDIATE statement can only reference a dynamic cursor.

Ambiguous cursor references: When a dynamic positioned UPDATE statement is being created by a PREPARE or EXECUTE IMMEDIATE statement, it is possible that CA-IDMS may not be able to determine which cursor is being referenced. This will occur if the application program contains a DECLARE CURSOR statement that defines a cursor having the referenced name and the program has also executed an ALLOCATE cursor statement that creates a cursor with the same name and a local scope. Under these conditions, CA-IDMS cannot determine which of the two cursors is being referenced. To avoid such problems, it is advisable to use different names for cursors that are declared from those that are allocated with a local scope.

5.3.36 Examples

A positioned UPDATE referencing a DECLARED cursor: The following statement updates the current row of the cursor C1. C1 may either be a dynamic or static cursor, but it must have been defined using a DECLARE CURSOR statement. Furthermore, the cursor-specification on which C1 is based must contain a FOR UPDATE option which either directly or implicitly includes the EMP_LNAME column:

```
EXEC SQL
  UPDATE EMPLOYEE
    SET EMP_LNAME = :emp-name
    WHERE CURRENT OF C1
END-EXEC
```

A positioned UPDATE referencing an ALLOCATED cursor: The following statement updates the current row of a cursor whose name is specified in the variable CNAME. The referenced cursor must have been defined using an ALLOCATE CURSOR statement:

```
EXEC SQL
  UPDATE EMPLOYEE
    SET EMP_LNAME = :emp-name
    WHERE CURRENT OF GLOBAL :CNAME
END-EXEC
```

A dynamically-compiled positioned UPDATE statement: The following statement updates the current row of local cursor C1. C1 may have been defined using either a DECLARE CURSOR statement or an ALLOCATE CURSOR statement. In either case, the cursor name in the UPDATE statement is specified as an identifier rather than as a literal or host variable:

```
EXEC SQL
  EXECUTE IMMEDIATE
    'UPDATE EMPLOYEE SET EMP_STATUS = "T"
     WHERE CURRENT OF LOCAL C1'
END-EXEC
```

Note: The keyword LOCAL is unnecessary since it is the default. Regardless of whether or not it is specified, if two local cursors named C1 have been defined, one using a DECLARE CURSOR statement and one using an ALLOCATE CURSOR statement, the EXECUTE IMMEDIATE statement will fail on an ambiguous cursor error.

5.4 ALTER INDEX support

Release 14.0 supports a new ALTER INDEX statement. This statement provides the ability to change the maximum number of index entries contained in an internal index record (SR8 system record).

Benefit: It is sometimes desirable to change the number of entries in an SR8 system record after an index has been loaded. Prior to this release, there was no way of doing this without dropping and recreating the index, a potentially expensive process for large tables. The new ALTER INDEX statement enables the maximum number of entries to be changed without affecting the existing index structure.

►► For more information on index structure and design considerations, see the *CA-IDMS Database Administration Guide*.

Syntax

```
►► ALTER INDEX index-name ON schema-name. table-identifier
► INDEX BLOCK CONTAINS key-count KEYS
```

Parameters

index-name

Specifies the name of the index being modified. *Index-name* must identify an index defined in the dictionary.

table-identifier

Identifies the table on which the named index is defined.

schema-name

Identifies the SQL schema associated with the named table.

If you do not specify a *schema-name*, the default value is:

- The current schema associated with your SQL session, if the statement is entered through the Command Facility or executed dynamically.
- The SQL schema associated with the access module used at runtime, if the statement is embedded in an application program.

key-count

Establishes the new value for the maximum number of entries in each internal index record (SR8 system record).

Key-count must be an unsigned integer in the range 3 through 8180.

5.4.1 Usage

System tables: You cannot alter an index defined on a table in the SYSTEM schema.

5.4.2 Example

Changing the maximum number of entries in an SR8 record: The following statement changes the characteristics of the JOB-TITLE-INDEX index by specifying that the maximum number of entries in an internal index record is to be 500:

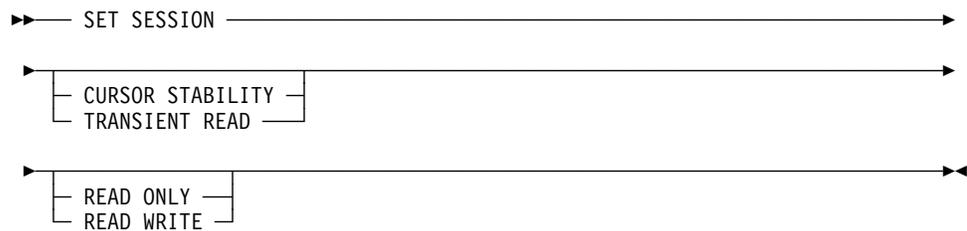
```
ALTER INDEX JOB-TITLE-INDEX ON JOB  
INDEX BLOCK CONTAINS 500 KEYS
```

5.5 Establishing default transaction options

You can now establish default transaction options for an SQL session using the SET SESSION statement. You can establish the default mode in which a database is accessed (READ ONLY or READ WRITE) and specify an isolation level (CURSOR STABILITY or TRANSIENT READ).

If you don't specify either of these options, the defaults are READ WRITE and CURSOR STABILITY, or the settings specified as part of the access module definition for embedded SQL. The default options may be overridden for an individual transaction by using the SET TRANSACTION statement.

Syntax: The syntax for only the new transaction options for the SET SESSION statement is shown below. For a description of the parameters, see the CREATE ACCESS MODULE statement in the *SQL Reference*.



5.6 SQLSTATE field in SQLCA

The SQLCA communications area now contains the SQLSTATE field. SQLSTATE is a status variable first introduced in the SQL2 standard as the method for returning errors to applications.

SQLSTATE is a five-character string in which CA-IDMS returns the status of the last SQL statement executed. It is divided into a two-character class and a three-character subclass. Standard values are associated with each class and subclass, which minimizes the need for vendors to define their own values and makes applications more portable from one environment to another.

5.6.1 SQLSTATE values

The list of SQLSTATE values that CA-IDMS can return appear below. The list is divided into sections based on the class (the first 2 characters of the SQLSTATE value). Each subclass (the last 3 characters of the SQLSTATE value) is listed under its associated class.

ANSI- and ISO-defined values: Class and subclass values beginning with the characters A-H and 0-4 are established by the ANSI and ISO standards organizations.

CA-IDMS-defined values: Class and subclass values beginning with the characters I-Z and 5-9 are vendor-defined; in this case, they are specific to CA-IDMS. (Any subclass value associated with a vendor-defined class is also defined by that vendor.)

SQLSTATE values

```
00 Successful completion
  000 No subclass

01 Warning
  000 No subclass
  004 String data, right truncation
  600 Inconsistent or invalid option
  602 Entity or association already exists
  605 Entity not defined in Catalog
  606 Invalid option for physical DDL
  607 Invalid option for DMCL
  608 Connecting to a dictionary which is missing either or
      or both of DDLCAT/DDLDML areas
  610 Database is inconsistent with request
  638 Warning returned from table procedure

02 No data
  000 No subclass

07 Dynamic SQL error
  000 No subclass
  001 USING clause does not match dynamic parameter specification
  002 USING clause does not match target specification
  003 Cursor specification cannot be executed
  004 USING clause required for dynamic parameters

08 Connection exception
  000 No subclass
  004 SQL-server rejected establishment of SQL-connection
  006 Connection failure
```

21	Cardinality violation
000	No subclass
22	Data Exception
000	No subclass
001	String data, right truncation
002	Null value, no indicator parameter
003	Numeric value out of range
005	Error in assignment
007	Invalid datetime format
008	Datetime field overflow
011	Substring error
012	Division by zero
019	Invalid escape character
23	Constraint violation
000	No subclass
501	Duplicate key violation
24	Invalid cursor state
000	No subclass
25	Invalid transaction state
000	No subclass
006	Read-only SQL-transaction
26	Invalid SQL statement name
000	No subclass
28	Invalid authorization specification
000	No subclass
602	Entity or association already defined
605	Entity or association not previously defined
607	Authorization ids not specified
2C	Invalid character set name
000	No subclass
38	External routine exception
000	No subclass
39	External routine invocation exception
000	No subclass

```
3F Invalid schema name
   000 No subclass

40 Transaction rollback
   000 No subclass
   001 Serialization failure

42 Syntax error or access rule violation
   000 No subclass
   500 Table not found
   501 Column not found
   502 Entity already defined
   503 Authorization failure
   504 Cursor not declared or previously declared
   505 Entity not found
   506 Invalid identifier
   507 Keyword used as identifier
   600 Invalid statement
   601 Statement not valid in this context
   603 Statement not valid for this schema
   604 Invalid data type
   606 Invalid statement option
   607 Missing statement option
   609 Invalid constraint definition
   610 Invalid number of columns

50 CA-defined errors
   000 No subclass
   002 Limit exceeded
   003 Space exceeded
   00B Internal error
   00I Schema mismatch
   00J Invalid entity definition
   00K Uncategorized error
   00L Invalid calling parameters

60 CA-IDMS specific errors
   000 No subclass
   001 Problem with load module or synchronization stamps
   002 Database error
   003 Rollback failed

64 CA-IDMS Physical DDL error
   000 No subclass

6U CA-IDMS Utility error
   000 No subclass
```

5.6.2 SQLSTATE field placement in the SQLCA

The SQLSTATE field appears after the SQLERRM field in the SQLCA. The layout of the SQLCA is shown below.

```
01 SQLCA.
02 SQLCAID          PIC X(8).
02 SQLCODE          PIC S9(8) COMP.
02 SQLCSID          PIC X(8).
02 SQLCINFO.
03 SQLCERC          PIC S9(8) COMP.
03 FILLER           PIC S9(8) COMP.
03 SQLCNRP          PIC S9(8) COMP.
03 FILLER           PIC S9(8) COMP.
03 SQLCSER          PIC S9(8) COMP.
03 FILLER           PIC S9(8) COMP.
03 SQLCLNO          PIC S9(8) COMP.
03 SQLCMCT          PIC S9(8) COMP.
03 SQLCARC          PIC S9(8) COMP.
03 SQLCFJB          PIC S9(8) COMP.
03 FILLER           PIC S9(8) COMP.
03 FILLER           PIC S9(8) COMP.
02 SQLCINF2 REDEFINES SQLCINFO.
03 SQLERRD          PIC S9(8) COMP
                   OCCURS 12.

02 SQLCMSG.
03 SQLCERL          PIC S9(8) COMP.
03 SQLERM           PIC X(256).
02 SQLCMSG2 REDEFINES SQLCMSG.
03 FILLER           PIC X(2).
03 SQLERRM.
04 SQLCERRML        PIC S9(4) COMP.
04 SQLERRMC         PIC X(256).
02 SQLSTATE         PIC X(5).
02 FILLER           PIC X(11).
02 SQLWORK          PIC X(16).
02 SQLCWRK2 REDEFINES SQLWORK.
03 SQLERRP.
04 SQLCVL           PIC X(5).
04 FILLER           PIC X(3).
03 SQLWARN.
04 SQLWARN0         PIC X(1).
04 SQLWARN1         PIC X(1).
04 SQLWARN2         PIC X(1).
04 SQLWARN3         PIC X(1).
04 SQLWARN4         PIC X(1).
04 SQLWARN5         PIC X(1).
04 SQLWARN6         PIC X(1).
04 SQLWARN7         PIC X(1).
```

5.7 Optimization enhancements

Adjustments were made to the costing algorithm for accessing data through an index when statistics are not available. These adjustments may result in improved access strategies under certain conditions. The following effects may be observed:

- Access through a clustering index will be preferred to an area sweep
- Access through single-column indexes will be preferred to multiple-column indexes if WHERE clauses reference only the first columns of the indexes or list access strategies will tend to be avoided for small tables and used more often for larger tables, especially when the underlying rows are clustered through an index

Benefit: These changes should result in improved access strategies in cases where statistics are not available, such as for access to non SQL-defined data or when UPDATE STATISTICS is not used.

5.8 Migration of SQL syntax by CA-IDMS/Dictionary Migrator

You can now use the CA-IDMS/Dictionary Migrator product to migrate logical SQL entities. See Chapter 8, “CA-IDMS Tools” on page 8-1, for more information on using the CA-IDMS/Dictionary Migrator to migrate SQL statements.

Chapter 6. CA-ADS and the Mapping Facility

- 6.1 Overview 6-3
- 6.2 Concurrent checkout of maps and dialogs 6-4
- 6.3 Name and disable dialogs abending on program checks 6-5
- 6.4 Using IDD record syntax for tables and view 6-6

6.1 Overview

This chapter describes general enhancements to CA-ADS and the Mapping Facility.

The following enhancements are new:

- Concurrent checkout for maps and dialogs in multiple dictionaries
- Name and disable abending dialog on program checks
- Using IDD record syntax for table and view definitions

6.2 Concurrent checkout of maps and dialogs

In a multiple dictionary environment, you can now simultaneously check out a map with the same name or dialog with the same name across multiple dictionaries. Previously, if the same map or dialog name was in multiple dictionaries, it could be checked out in only one dictionary at a time.

Benefit: Multiple users can check out a map with the same name or dialog with the same name at the same time in multiple dictionaries.

6.3 Name and disable dialogs abending on program checks

In Release 14.0, when a CA-ADS dialog abends with a program check, the name of the abending dialog is reported and the program check count for the abending dialog is incremented. Therefore, only the abending dialog is disabled when its error threshold is exceeded.

Additionally, the error threshold for ADSOMAIN and ADSORUN1 is always set to zero at startup. After the system is started, you can use the DCMT VARY PROGRAM PROGRAM CHECK THRESHOLD command to change the error threshold for ADSOMAIN and ADSORUN1.

Benefit: This enhancement has these benefits:

- Programmers can readily determine the name of the abending dialog.
- Only the dialog that caused a program check is disabled when its error threshold is exceeded.
- CA-ADS is not disabled due to a dialog logic error or an internal CA-ADS error.

What to do: You may need to adjust the ERROR THRESHOLD parameter for each dialog on its corresponding PROGRAM statement. For more information on modifying these system generation parameters, see Chapter 2, “Upgrading to Release 14.0” on page 2-1.

6.4 Using IDD record syntax for tables and view

The DISPLAY/PUNCH statement for table and view entities includes the LIKE RECORD parameter, which displays or punches IDD record syntax for a table or view.

Using this parameter, you can punch IDD record syntax for a table or view and then use it to define the record or view in a dictionary. Once the record definition is in a dictionary, it can be used in CA-ADS dialogs as a work record definition. This same record definition can be included in a map definition. For more information on the LIKE RECORD parameter, see the DISPLAY/PUNCH TABLE and DISPLAY/PUNCH VIEW statements in Chapter 5, “CA-IDMS SQL Option” on page 5-1.

Chapter 7. CA-OLQ

- 7.1 Overview 7-3
- 7.2 Display report line length 7-4
- 7.3 HOME and LEFTMAX scroll commands 7-5
- 7.4 DISTINCT option in Menu Mode 7-6
- 7.5 Specify report column for aggregate columns 7-7
- 7.6 Override column length on columns with code tables 7-9

7.1 Overview

Most of the enhancements to CA-OLQ Release 14.0 simplify the process of creating and displaying reports in Menu Mode. This chapter presents each of these enhancements in a separate section as follows:

- Display minimum report line width
- Use HOME and LEFTMAX scroll commands
- Select DISTINCT option
- Specify report column for aggregate columns
- Override column length for columns using code tables

7.2 Display report line length

Two new fields appear on the Display Report screen that give you information about the width of a report:

- **Total displayable cols** — The number of columns to be displayed on the report.
- **Formatted line length** — The minimum line width required to print the report. This is the sum of the lengths of the columns.

Benefit: You can choose a more appropriate medium for a wide report, or modify the report so that all of its data can be printed on the current printer. For example, if the formatted line length of a report is 100, you can route the report to a 132-byte width printer.

Sample Display Report screen: The sample Display Report screen below shows the new fields.

```

CA-OLQ Release 14.0                               *** Retrieval Completed ***
→
130000 Select activity and press ENTER key

Number of whole rows . . . . . 57  Total displayable cols . . . . . 20
Total number of records read . . . . . 57  Formatted line length. . . . . 372
Total number of records selected. . . . . 57
Number of data errors . . . . . 0

Select      Command/
Option      → Display/Format Activity <--- Screen Name

X          Display report          DISplay
-          Save report            SAVe
-          Choose the sort sequence of report  SORT
-          Change column headers   HEAdEr
-          Change page header and footer  PAGe HEAdEr
-          Change display format of data ($,commas)  PICTure
-          Format columns (Alignment, sparse)  EDIt
-          Specify summary computations (Totals)  GROUp BY
-          Send the report to a printer      PRInt

1=HELP          3=QUIT          4=MESSAGE          6=MENU

```

7.3 HOME and LEFTMAX scroll commands

The following new commands are available on the Display Report screen to help you easily browse a wide report online:

- **HOME** — Returns you to the top of the report; page 1 column 1
- **LEFTMAX or LMAX** — Returns you to column 1 of the current page of the report

Previously, if you scrolled right 10 times to view an entire report, you needed to scroll left 10 times to return to column 1 before you could view the next page.

7.4 DISTINCT option in Menu Mode

You can now select an option on the Column Select screen to indicate that you don't want duplicate rows (detail lines) displayed on the report. The DISTINCT option is available on the Column Select screen so that you can choose to display only unique rows in a report. The default is N indicating that duplicate rows are displayed.

Benefit You can display a report in Menu Mode that contains only unique rows; duplicate rows are not included in the report.

Sample Column Select Screen The DISTINCT option on the Column Select screen below has been changed to Y to indicate that only unique rows of data are to be displayed in the report.

```

CA-OLQ Release 14.0                                     *** Column Select ***
→                                                    Page 1 Of 2
124000 Select columns, specify selection criteria and press the ENTER key.

Columns Currently Selected: 20      Selection Criteria  Distinct Y Y/N
EMPLOYEE
X 02 EMP-ID-0415                               *
 02 EMP-NAME-0415
X 03 EMP-FIRST-NAME-0415
X 03 EMP-LAST-NAME-0415
 02 EMP-ADDRESS-0415
X 03 EMP-STREET-0415
X 03 EMP-CITY-0415
X 03 EMP-STATE-0415
 03 EMP-ZIP-0415
X 04 EMP-ZIP-FIRST-FIVE-0415
X 04 EMP-ZIP-LAST-FOUR-0415
X 02 EMP-PHONE-0415
X 02 STATUS-0415
Additional Selection Criteria:

Proceed to Selection Criteria Screen? N Y/N
1=QUIT      3=HELP      4=MESSAGE      6=MENU      8=FWD      PA2=REFRESH

```

7.5 Specify report column for aggregate columns

In Menu Mode, you can now specify the column under which you want to display an aggregate field on a report.

Benefit You can position computed fields in an online report where they are most meaningful.

Sample Report Format screen: In the sample screen below, four is specified in the Seq column so that the result of the computed field, START-YEAR-0415-AVE-ALL, appears after the last line of the fourth column on the report.

```

CA-OLQ Release 14.0                               *** Report Format - Group By ***
→                                                    Page 1 OF 1
136000 Specify summary computations and press the ENTER key.

Group by:
EMPLOYEE
X EMP-ID-0415                                     1      -      -      -      -
X EMP-FIRST-NAME-0415                             2      -      -      -      -
X EMP-LAST-NAME-0415                              3      -      -      -      -
X START-YEAR-0415                                  4      -      -      -      -
X START-MONTH-0415                                 5      -      -      -      -
X START-DAY-0415                                   6      -      -      -      -

COMPUTE FIELDS:
X START-YEAR-0415-AVE-ALL=AVE(START-YE           4
AR-0415) GROUP BY ALL LEVEL 1

Skip lines after group 1      Separator character -
Compute:
1=QUIT      3=HELP      4=MESSAGE      6=MENU      10=PICTURE

```

The screen below shows the value of the computed field displayed after the last line of the fourth column, START-YEAR-0415.

7.5 Specify report column for aggregate columns

CA-OLQ Release 14.0 *** Display Report ***
→ Page 6 Line 56
125004 Press ENTER for DISPLAY/FORMAT ACTIVITY selections.

EMPLOYEE REPORT
10/12/95

ID	FIRST NAME	LAST NAME	START YEAR	START MONTH	START DAY
0045	GEORGE	FONRAD	80	APR	14
0321	DANIEL	MOON	78	JAN	03

AVG FOR ALL: 78

END OF REPORT

- 6 -
1=QUIT 3=HELP 4=MESSAGE 6=MENU 7=BWD 10=LEFT 11=RIGHT

7.6 Override column length on columns with code tables

A Display length column now appears on the Report Format screen so that you can override the column length for columns containing a code table. To use this feature in command mode, use the DISPLAY WIDTH command.

For example, suppose a column on a report has a code table assigned to it and the external picture is longer than the report requires. You can override the external picture and assign a shorter, more appropriate length for the report.

Benefit: This allows you to tailor the display width of report columns containing code tables.

Sample Report Format screen: In the example below, the display length of the START-MONTH-0415 column is changed to three so that only a three-character month code is displayed on the report.

```

CA-OLQ Release 14.0                *** Report Format - Edit ***
→                                     Page 1 of 1
135000 Specify edit options and press the ENTER key.

      EMPLOYEE
      X EMP-ID-0415           1  -  -  RIGHT
      X EMP-FIRST-NAME-0415  2  -  -  LEFT
      X EMP-LAST-NAME-0415   3  -  -  LEFT
      X START-YEAR-0415       6  -  -  RIGHT
      X START-MONTH-0415     4  -  -  RIGHT  GWGMONTB  13
      X START-DAY-0415       5  -  -  RIGHT

      COMPUTE FIELDS:
      X START-YEAR-0415-AVE-ALL=AVE(START-YE
        AR-0415) GROUP BY ALL LEVEL 1           -  RIGHT

Compute:
1=QUIT      3=HELP      4=MESSAGE      6=MENU      10=HEADER      11=PICTURE

```

The report is displayed showing the three-character month in the START MONTH column.

7.6 Override column length on columns with code tables

CA-OLQ Release 14.0 *** Display Report ***
→ Page 1 Line 1
125000 Press the ENTER key to go to the next page of the report.

EMPLOYEE REPORT
10/12/95

ID	FIRST NAME	LAST NAME	START MONTH	START DAY	START YEAR
0023	KATHERINE	O'HEARN	MAY	04	78
0472	ROBBY	WILDER	JUL	16	79
0301	BURT	LANCHESTER	FEB	03	75
0027	VLADIMIR	HEAROWITZ	SEP	09	81
0471	THEMIS	PAPAZEUS	SEP	07	78
0007	MONTE	BANK	APR	30	78
0334	CAROLYN	CROW	JUN	17	79
0127	CAROL	MCDOUGALL	JUN	07	80
0019	JULIE	JENSEN	SEP	29	82
0366	ALAN	DONOVAN	OCT	10	81
0476	BETSY	ZEDI	FEB	23	76

- 1 -

1=QUIT	3=HELP	4=MESSAGE	6=MENU	8=FWD	10=LEFT	11=RIGHT
--------	--------	-----------	--------	-------	---------	----------

Chapter 8. CA-IDMS Tools

8.1 Overview	8-3
8.2 CA-IDMS/Dictionary Migrator	8-4
8.2.1 Migrating SQL Entities	8-4
8.2.1.1 Catalog Navigation Report	8-5
8.3 CA-IDMS/ADS Alive	8-6
8.4 CA-IDMS/DB Analyzer, CA-IDMS/DB Audit, CA-IDMS/DB Reorg	8-7
8.4.1 Additional CA-IDMS/DB Reorg features	8-7
8.4.2 Interface to DB/EZReorg	8-7
8.4.3 Specifying a blocking factor for work files	8-8
8.5 Year 2000 support	8-9

8.1 Overview

This chapter describes the new features included in the CA-IDMS tools products that are available with CA-IDMS Release 14.0.

For specific information on CA-IDMS tools, see the appropriate documentation for each tool.

The following CA-IDMS tools include new features, which are available with Release 14.0:

- CA-IDMS/Dictionary Migrator
- CA-IDMS/ADS Alive
- CA-IDMS/DB Analyzer
- CA-IDMS/DB Audit
- CA-IDMS/DB Reprg

Additionally, all CA-IDMS tools now have year 2000 support.

8.2 CA-IDMS/Dictionary Migrator

You can use the CA-IDMS/Dictionary Migrator to extract definitions of logical SQL entities from a source catalog and create a syntax file (BCFUDP) containing these definitions. You can then load the syntax file to a target catalog using the CA-IDMS Batch Command Facility.

Entities you can migrate: You can migrate the following SQL entities:

- Schema
- Table
- Table Procedure
- View

MIGRATOR ASSISTANT updated: The MIGRATOR ASSISTANT has also been updated to generate the parameters to extract logical SQL entities for a batch run.

8.2.1 Migrating SQL Entities

To migrate definitions of logical SQL entities from a source catalog you identify the entities you want to migrate using the logical SQL entity parameters. You also identify the name of the source catalog from which the definitions will be extracted. A new PROCESS statement parameter exists to let you specify that you want to extract the SQL entities as a stand-alone run. These parameters are described next.

SQL entity parameters: Use the following parameters to specify the logical SQL entities you want to migrate.

SQLTable = schema-name.sql-table-name

Specifies the name of a base table in the source catalog whose definition you want to migrate.

Schema-name identifies the name of the schema associated with the table.

SQLView = schema-name.sql-view-name

Specifies the name of a view in the source catalog whose definition you want to migrate.

Schema-name identifies the name of the schema associated with the view.

SQLSchema = sql-schema-name

Specifies the name of a schema in the source catalog that you want to migrate.

SQLProc = schema-name.table-procedure-name

Specifies the name of a table procedure in the source catalog that you want to migrate.

Schema-name identifies the name of the schema associated with the table procedure.

SQLONLY parameter: You can extract SQL entities as a stand-alone run by specifying the SQLONLY parameter on the PROCESS statement or in conjunction with network extractions. This parameter is only valid for a run type of EXPORT.

When you specify the SQLONLY parameter, only the Parameter Verification report and the new Catalog Navigation report are generated.

Specify catalog name on DICTIONARY parameter: You specify the name of the catalog containing the entities you wish to migrate on the DICTIONARY parameter. The default is the system catalog.

Example: The following example shows sample input parameters to extract the DEMO schema from the TEST catalog.

```
PROCESS,
SQLONLY,
DICTIONARY=TEST,
OBJD=TEST,
LEVEL=DIALOG,
RUN=EXPORT
EXTRACT,SQLSCHEMA=DEMO
```

8.2.1.1 Catalog Navigation Report

A new report, Catalog Navigation, is generated when you specify the SQLONLY parameter. It provides information about the entities extracted from the source catalog, and it also highlights error conditions encountered during the extraction process.

The sample report shown below contains information about the DEMO schema specified in the sample input parameters in the above example.

CA-TOOLS	RELEASE	CA-IDMS/DICTIONARY MIGRATOR	DATE	TIME	PAGE
MV9507	R14.0	CATALOG NAVIGATION REPORT	01/31/96	16:03:44	0001
USMS004I	EXTRACT SCHEMA - SCHEMA: DEMO				
USMS002I	EXTRACT TABLE - SCHEMA: DEMO , TABLE: EMPL				
USMS010I	EXTRACT CONSTRAINT - SCHEMA: DEMO , CONSTRAINT: EMP_POS				
USMS002I	EXTRACT TABLE - SCHEMA: DEMO , TABLE: POSITION				
USMS010I	EXTRACT CONSTRAINT - SCHEMA: DEMO , CONSTRAINT: MGR_EMPL				
USMS010I	EXTRACT CONSTRAINT - SCHEMA: DEMO , CONSTRAINT: EMP_DEPS				
USMS002I	EXTRACT TABLE - SCHEMA: DEMO , TABLE: DEPENDENTS				
USMS010I	EXTRACT CONSTRAINT - SCHEMA: DEMO , CONSTRAINT: DEPT_EMPL				
USMS002I	EXTRACT TABLE - SCHEMA: DEMO , TABLE: DEPT				
USMS010I	EXTRACT CONSTRAINT - SCHEMA: DEMO , CONSTRAINT: JOB_POS				
USMS002I	EXTRACT TABLE - SCHEMA: DEMO , TABLE: JOB				
USMS010I	EXTRACT CONSTRAINT - SCHEMA: DEMO , CONSTRAINT: DEPT_BUDGET				
USMS002I	EXTRACT TABLE - SCHEMA: DEMO , TABLE: BUDGET				
USMS011I	EXTRACT TABLE PROCEDURE - SCHEMA: DEMO , PROCEDURE: TESTPROC 1				
USMS012I	EXTRACT KEY - KEYNAME: PROC1_PRIME_KEY , ON SCHEMA: DEMO , PROCEDURE: TESTPROC 1				
USMS012I	EXTRACT KEY - KEYNAME: PROC1_UNIQ_KEY , ON SCHEMA: DEMO , PROCEDURE: TESTPROC 1				
USMS012I	EXTRACT KEY - KEYNAME: PROC1_STD_KEY , ON SCHEMA: DEMO , PROCEDURE: TESTPROC 1				
USMS001I	CATALOG NAVIGATION COMPLETED				

8.3 CA-IDMS/ADS Alive

CA-IDMS/ADS Alive now recognizes tables as records. The RECORD command now lists SQL tables as well as native DML records.

8.4 CA-IDMS/DB Analyzer, CA-IDMS/DB Audit, CA-IDMS/DB Reorg

CA-IDMS/DB Analyzer, CA-IDMS/DB Audit, and CA-IDMS/DB Reorg products now use the following features of CA-IDMS database I/O:

- XA database buffers and control blocks
- ESA dataspace support
- Dynamic database file allocation
- Unrestricted SEGMENT name usage as database names

For more information on these CA-IDMS database features, see *CA-IDMS Database Administration* and *CA-IDMS System Operations*.

Additionally, features of the CA-IDMS database engine and the SYSIDMS PRE-FETCH parameter for read-ahead processing replace the read-ahead processing previously provided by FASTSCAN and EXCP I/O level processing previously defined by the GSDTPARM installation defaults.

8.4.1 Additional CA-IDMS/DB Reorg features

CA-IDMS/DB Reorg offers these two new features:

- An interface to the DB-EZReorg inflight reorganization product to provide a database keys file, which is used to resolve changed database record locations. You can choose to use either CA-IDMS/DB Reorg or CA-IDMS UNLOAD and RELOAD utilities with DB-EZReorg.
- A new BLOCKNUM parameter to specify a blocking factor of up to 32K for most of the work files it generates. This allows you to select an optimum blocking factor for the work files. Previously, the default of 6K was the only option.

8.4.2 Interface to DB/EZReorg

CA-IDMS/DB REeorg creates the interface to the DB/EZReorg file when a DBKEYS dataset is present in the UPDLINK step in the JCL. The dataset is created only when a DBKEYS file is present in the UPDLINK step.

Include these characteristics in the UPDLINK step:

- RECFM=FB
- LRECL=16

Do not specify a block size since CA-IDMS/DB Reorg uses the block size specified in the BLOCKNUM parameter used for all work files. See "Specifying blocking factor," below for more information on specifying block sizes for work files.

Determining space for DBKEYS file: To determine the amount of space to allocate for the DBKEYS file, use the following formula:

(# of records in reorganized areas - index records (SR8s))
x 16 (DBKEYS file LRECL)

Total bytes to allocate for the DBKEYS file

8.4.3 Specifying a blocking factor for work files

You can now specify a blocking factor for most of the CA-IDMS/DB Reorg work files. You cannot specify a blocking factor for the CNTRL1, CNTRL2, and PAGUTIL work files.

BLOCKNUM parameter on the PROCESS statement You specify a blocking factor using the BLOCKNUM parameter on the PROCESS statement. Specify a value in the range from 1 through 32. For work files that will go on tape, use 32. If using disk work files, use a number that represents half track blocking.

8.5 Year 2000 support

All CA-IDMS tools now provide support for the year 2000. For example, when migrating DDL/DML entities with a two-digit date for year, the CA-IDMS/Dictionary Migrator checks if the digits are greater than 69; if they are, it assumes that the century is 19. If the data digits are less than 69, it assumes the century is 20.

Index

Special Characters

- #DEFOPTF macro
 - example 2-15
 - to apply optional APARs 2-15
- #SECR TT macro 2-5
 - changing security definitions 4-7
 - specifying default signon options 4-17
 - syntax 4-18

Numerics

- 24-hour processing 4-6—4-10
 - dynamic lines, terminals, printers 4-6—4-7
 - dynamic resource allocation 4-8—4-10
 - dynamic security refresh 4-7—4-8

A

- ACCESS MODULE, DISPLAY/PUNCH
 - statement 5-12—5-14
- ADSOMAIN program, overriding ERROR THRESHOLD parameter 6-5
- Aggregate columns, CA-OLQ reports 7-7
- ALLOCATE CURSOR statement
 - examples 5-43
 - syntax 5-43
 - updatable cursors 5-43
- ALREADY VERIFIED option 4-28
- ALTER INDEX statement
 - example 5-60
 - overview 5-59
 - syntax 5-59
 - usage 5-60
- APARs, optional
 - applying 2-15
- APPC applications
 - verifying signons for 4-28—4-29

B

- Backend CVs
 - joining/leaving groups 3-14
- BACKUP utility
 - segment support 4-11
 - syntax 4-11
- Buffers, using shared cache 3-20

C

- CA-ADS
 - dialog abends on program check 6-5
 - maps and dialogs, concurrent checkout in dictionary 6-4
 - PROGRAM statement ERROR THRESHOLD parameter, reviewing 2-12
 - using IDD record syntax for tables 6-6
- CA-IDMS
 - Release 14.0 features 4-3—4-39
 - shared cache, defining 3-21
- CA-IDMS Performance Monitor
 - See* Performance Monitor
- CA-IDMS Security
 - See* Security
- CA-IDMS SQL Option
 - See* SQL Option
- CA-IDMS/DC
 - running LE/370-compliant programs 4-31
 - using LE/370-compliant language compilers 4-30—4-33
- CA-IDMS/Dictionary Migrator, changes for Release 14.0 8-4
- CA-OLQ
 - aggregate columns, specifying placement 7-7
 - code tables, overriding column length 7-9
 - DISTINCT option 7-6
 - HOME scroll command 7-5
 - LEFTMAX scroll command 7-5
 - line length, displayed on reports 7-4
- CALC key, DISPLAY/PUNCH statement 5-14—5-15
- Catalog
 - See also* also Catalog conversion utility
 - conversion utility 2-9
 - converting to Release 14.0 2-8
- Catalog conversion utility
 - purpose 2-8
 - running 2-9
- Central versions
 - See* CVs
- CICS-reentrant programs
 - enhancement to 4-39
- Cloned CVs
 - and dynamic database session routing 3-6
 - backend 3-12
 - benefits 3-12
 - overview 3-5

-
- Cloned CVs (*continued*)
 - procedure 3-12
 - startup JCL 3-13
 - system requirements 3-13
 - using cloned backend CVs 3-7
 - CLOSE statement
 - example 5-44
 - syntax 5-44
 - CONSTRAINT, DISPLAY/PUNCH
 - statement 5-15—5-16
 - Continuous processing
 - See* 24-hour processing
 - Coupling Facility
 - defined 3-4
 - shared cache 3-5
 - shared cache, defining 3-20
 - using for shared cache 3-19
 - CVs
 - assigning groups 3-12
 - dynamic lines, terminals, and printer definitions 4-6
 - dynamic resource allocation 4-8
 - dynamic security refresh 4-7
- D**
- Database name table
 - adding groups 3-11
 - defining groups 3-8
 - Database procedures, examining for potential
 - changes 2-6, 2-7, 4-5
 - DB exit 4-34
 - DBGROUP statement
 - examples 3-9
 - syntax 3-8
 - DCMT DISPLAY DBGROUP statement
 - using 3-16
 - DCMT DISPLAY SHARED CACHE statement 3-23
 - DCMT statement, SHARED CACHE option 3-22
 - DCMT SYSGEN REFRESH LINE statement 4-6
 - DCMT VARY DBGROUP statement
 - syntax 3-14
 - to manage dynamic database session routing 3-14
 - DEALLOCATE PREPARE statement
 - examples 5-45
 - syntax 5-44
 - usage 5-45
 - DELETE statement
 - ambiguous cursor references 5-46
 - dynamic deletes, using 5-29
 - dynamic positioned deletes 5-46
 - examples 5-47
 - DELETE statement (*continued*)
 - syntax 5-45
 - DESCRIBE statement
 - describing dynamic parameters 5-48
 - describing output values 5-48
 - syntax 5-47
 - Dialogs
 - concurrent checkout in dictionary 6-4
 - disabling abending on program check 6-5
 - Dictionary record changes 2-14
 - Disabling a CV from a group 3-14
 - DISPLAY/PUNCH statements
 - ACCESS MODULE statement 5-12—5-14
 - ALL 5-5—5-12
 - ALL for security entities 4-19—4-27
 - CALC KEY statement 5-14—5-15
 - CONSTRAINT statement 5-15—5-16
 - date and year 2000 support 4-16, 4-26
 - example of security entities 4-27
 - for SQL entities 5-4, 5-28
 - INDEX statement 5-17—5-18
 - KEY (table procedure) statement 5-18—5-19
 - list of security entities 4-23
 - SCHEMA statement 5-20—5-21
 - security entity syntax
 - TABLE 5-22—5-24
 - TABLE PROCEDURE statement 5-24—5-26
 - using 5-4
 - VIEW 5-26—5-28
 - DISTINCT option, CA-OLQ Menu Mode 7-6
 - DMCL, defining shared cache 3-22
 - DPE COUNT, reviewing primary allocation 2-13
 - Dynamic cursor names
 - examples 5-37
 - static versus extended 5-36
 - syntax 5-35
 - uniqueness 5-37
 - Dynamic cursor specification
 - and EXECUTE statement 5-51
 - examples 5-40
 - syntax 5-38
 - updatable 5-39
 - Dynamic cursors
 - ambiguous cursor references 5-46, 5-57
 - Dynamic database session routing
 - benefits 3-6
 - defined 3-6
 - defining groups 3-8, 3-9
 - enabling and disabling 3-14
 - implementing 3-7—3-12
 - managing 3-14—3-15
-

Dynamic database session routing (*continued*)
 monitoring and tuning 3-15—3-18
 overview 3-5
 planning 3-7
 using 3-6—3-18

Dynamic expressions
 cursor name 5-35—5-37
 cursor specification 5-38—5-40
 described 5-35
 statement name 5-41—5-43

Dynamic global cursor
 defining 5-30

Dynamic lines, terminals, printers 4-6—4-7

Dynamic parameters
 defined 5-31
 describing 5-48, 5-55
 using 5-31

Dynamic positioned delete, examples 5-47

Dynamic positioned update, examples 5-57

Dynamic resource allocation
 reviewing primary allocation in SYSGEN 2-13
 using 4-8—4-10

Dynamic security refresh 4-7—4-8

Dynamic SQL statements
 described 5-35

Dynamic SQL, overview 5-29

Dynamic statement names
 examples 5-42
 static versus extended 5-41
 syntax 5-41
 uniqueness 5-42

Dynamic statements
 destroying 5-44
 positioned deletes 5-46
 positioned updates 5-57

Dynamically-assigned names
 defined 5-29
 using 5-30

E

ERROR THRESHOLD parameter on PROGRAM statement, reviewing 2-12

EXECUTE statement
 dynamically-compiled cursor specifications 5-51
 syntax 5-49

F

FETCH statement
 syntax 5-51

FORMAT utility
 area support for journals 4-12
 syntax 4-13

G

Global statements and cursors
 defined 5-30
 defining 5-30

Groups
 adding to resource name table 3-9
 DBGROUP statement 3-8
 defined 3-8
 defining 3-8, 3-9
 displaying information about 3-17
 enabling and disabling 3-14
 identifying with NODE statement 3-9
 planning 3-8
 sample definitions 3-9, 3-11
 varying status of 3-14

H

HOME OLQ command 7-5

I

IDS MIOX2 DB exit 4-34

INDEX statement
 ALTER 5-59—5-60
 DISPLAY/PUNCH 5-17—5-18

Index, modifying 5-59—5-60

J

JCL
 for cloned CVs 3-13
 for multitasking 4-4

Joining a CV to a group 3-14

K

KEY (on Table procedures), DISPLAY/PUNCH statement 5-18—5-19

L

LE/370, defined 4-30

LE/370-compliant language compilers
 COBOL 370 compiler considerations 4-32
 considerations 4-30
 executing programs under CA-IDMS/DC 4-31

LE/370-compliant language compilers (*continued*)

supported compilers 4-30

supported functions 4-32

unsupported functions 4-32

using 4-30—4-33

using with CA-IDMS/DC 4-30

LEFTMAX (LMAX) command 7-5

LIKE RECORD parameter 5-24, 5-28

M

Mapping Facility, concurrent checkout of maps in dictionary 6-4

Multiple dictionaries, concurrent checkout of maps and dialogs 6-4

Multitasking 4-4—4-5

and user exits and database procedures 2-6

benefits 4-4

implementing 4-4—4-5

startup JCL 4-4

supporting operating systems 4-4

N

NODE statement

example 3-10

identifying groups 3-9

using to identify groups 3-9

O

OPEN statement 5-52

Optional functionality, applying 2-15

P

Parallel Sysplex

benefits 3-3

CA-IDMS features 3-5

Coupling Facility component 3-4

defined 3-4

exploited in CA-IDMS, overview 1-7

Performance Monitor

DBGROUP detail screen 3-17

record and map changes 2-10

shared cache statistics usage 3-25

Preparable statements 5-54

PREPARE statement 5-53

describing dynamic parameters 5-55

describing input and output values 5-55

PRINT SPACE utility

file support 4-14

PRINT SPACE utility (*continued*)

syntax 4-14

Program checks, disable dialogs abending on 6-5

PROGRAM statement ERROR THRESHOLD parameter, reviewing 2-12

PTERM statement

ALREADY VERIFIED option 4-28

Q

Query expression, updatable 5-40

R

RCE COUNT, reviewing primary allocation 2-13

Release 14.0

objectives 1-4—1-8

upgrading to 2-3—2-19

Reports, CA-OLQ

aggregate columns, specifying placement 7-7

code tables, overriding column length 7-9

DISTINCT option 7-6

HOME and LEFTMAX scroll commands 7-5

line length, displayed on reports 7-4

Resource name table, adding groups to 3-9

RESTORE utility

segment support 4-11

syntax 4-11

RHDCOPTF module, re-creating 2-15

RHDCSRTT module, recompiling 2-5

RLE COUNT, reviewing primary allocation 2-13

S

Schema, DISPLAY/PUNCH statement 5-20—5-21

Security

changing security definitions in #SECR TT macro 4-7

DISPLAY/PUNCH ALL statement 4-19—4-27

dynamic refresh 4-7

enhancements 4-17—4-29

specifying default signon options in #SECR TT 4-17

Segments

support in backup, restore, and unlock 4-11

SET SESSION statement, setting default transaction options 5-61

Shared cache 3-19—3-28

and dynamic database session routing 3-6

and XA storage 3-20

benefits 3-20

defined 3-19

defining in CA-IDMS 3-21

defining in Coupling Facility 3-20

Shared cache (*continued*)

- DMCL definition using File override parameter 3-21
- dynamic definition using DCMT command 3-22
- how it works 3-19
- implementing 3-20
- in Coupling Facility 3-5
- monitoring 3-23—3-28
- tuning 3-28
- using 3-23

SHARED CACHE DMCL file override parameter 3-21

SHARED CACHE option, DCMT command 3-22

SQL option

- DISPLAY/PUNCH statements 5-5—5-28
- enhancement for dynamic SQL 5-29—5-58
- enhancements to 5-3
- migrating syntax, using Dictionary Migrator 5-68
- optimizer enhancements 5-67
- SET SESSION statement, setting default transaction options 5-61
- SQLSTATE variable in SQLCA 5-62

SQLCA, SQLSTATE variable 5-62

System generation

- defining groups on NODE statement 3-9, 3-11
- DPE COUNT, reviewing primary allocation 2-13
- dynamic resource allocation 4-8
- PROGRAM statement ERROR THRESHOLD parameter, reviewing 2-12
- RCE COUNT, reviewing primary allocation 2-13
- refreshing SYSGEN 4-6
- reviewing primary allocation of resource control blocks 2-13
- RLE COUNT, reviewing primary allocation 2-13
- verifying signons for 4-28

T

Table procedure, DISPLAY/PUNCH statement 5-24—5-26

Table, DISPLAY/PUNCH statement 5-22—5-24

Task Control Element (TCE), residing in XA storage 2-7

U

UNLOCK utility

- segment support 4-11
- syntax 4-12

Updatable cursors

- and ALLOCATE CURSOR statement 5-43
- specification 5-39

Updatable query expression 5-40

UPDATE statement

- ambiguous cursor references 5-57
- dynamic positioned updates 5-57
- dynamic updates using 5-29
- example using dynamic cursor 5-40
- examples 5-57
- syntax 5-55

UPDATE STATISTICS utility

- nonSQL data support 4-13
- syntax 4-13

User exits

- accessing TCE stack 2-7
- examining for potential changes 2-6, 2-7
- running with MPMODE=ANY 2-6

Utility statement enhancements 4-11, 4-15

V

View, DISPLAY/PUNCH statement 5-26—5-28

X

XA storage

- and shared cache 3-20
- TCE allocated to 2-7

