CA IDMS™

Release Summary

r17

CA Product References

This document references the following CA products:

- CA ADS™
- CA ADS™ Alive
- CA IDMS™/DB
- CA IDMS™/DC (DC)
- CA IDMS™/DC or CA IDMS™ UCF (DC/UCF)
- CA IDMS™ DDS
- CA IDMS™ Dictionary Migrator
- CA IDMS™ Dictionary Migrator Assistant (DMA)
- CA IDMS™ Dictionary Module Editor (CA IDMS DME)
- CA IDMS™ DML Online (CA IDMS DMLO)
- CA IDMS™ Enforcer
- CA IDMS™ Extractor
- CA IDMS™ Journal Analyzer
- CA IDMS™ Masterkey
- CA IDMS™ Online Log Display
- CA IDMS™ SASO
- CA IDMS™ Server
- CA IDMS™ SQL
- CA IDMS™ UCF (UCF)
- CA IDMS™ Visual DBA
- CA OLQ™ Online Query for CA IDMS™ (CA OLQ)
- CA SiteMinder®

Contact Technical Support

For online technical assistance and a complete list of locations, primary service hours, and telephone numbers, contact Technical Support at http://ca.com/support.

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com. If you would like to provide feedback about CA product documentation, complete our short customer survey on CA Support Online.

Contents

Chapter 1. Introduction

The *Release Summary* documents new features and changes to existing features for r17. It also provides information about upgrading to r17.

This chapter summarizes the new features of the release and the conventions used in the presentation of syntax throughout the manual.

This chapter contains the following topics:

1.1 New and Changed Features

This release incorporates many new features and changes to existing features to enhance your use of CA IDMS in the following areas of functionality:

- Non-stop processing
- Performance
- SQL
- TCP/IP
- Administration and Operations
- Application development
- CA IDMS Tools

The following are the new features in CA IDMS r17 and references to detailed descriptions about them.

New r17 Feature	Reference
Non-Stop Processing Enhancements	
A new tracking capability provides a means of making dynamic changes to the database environment of a Central Version (CV) in a fault tolerant manner.	See 3.1, “Change Tracking” on page 46.
Dynamic Journal files provide enhanced 24x7 capabilities by enabling the journal files in use by a CV to be changed while the system remains active.	See 3.2, “Dynamic Journal Files” on page 67.
Two scratch enhancements are provided to improve management and performance of the CA IDMS scratch area as well as increase system availability. These enhancements include: - Scratch above the bar - Extensible scratch	See 3.3, “Scratch Enhancements” on page 75.

New r17 Feature	Reference
Performance Enhancements	
CICS threadsafe support allows threadsafe application programs to use multiple open TCBs while accessing CA IDMS, thereby increasing throughput.	See 4.1, “CICS Threadsafe Support” on page 86.
The FORMAT JOURNAL utility is enhanced to quickly reformat already existing and formatted journal files.	See 4.2, “Fast Journal Format Option” on page 95.
To reduce CPU usage, system mode execution is available for database procedures and other Language Environment (LE) COBOL or PL/I programs.	See 4.3, “LE System Mode Support” on page 96.
CA IDMS's use of 24-bit storage usage is reduced, thereby relieving pressure on storage constrained CA IDMS and CICS systems.	See 4.4, “Reduced 24-bit Storage Usage” on page 97.
CA IDMS is enhanced to exploit zIIP processors on the z9 series for the z/OS operating system.	See 4.5, “zIIP Exploitation” on page 98.
SQL Enhancements	
SQL is now available as a programming language for SQL-invoked procedures and functions.	See 5.1, “SQL Procedural Language Support in Routines” on page 110.
An SQL-invoked procedure can now return result sets in the form of rows of result tables to the procedure invoker.	See 5.2, “Result Sets from SQL-invoked Procedures” on page 161.
The new GET DIAGNOSTICS and GET STATISTICS statements can be used for diagnosing the execution of SQL statements and for returning statistical information about the current transaction.	See 5.3, “Enhanced Diagnostics and Statistics” on page 173.
Join capabilities have been enhanced by adding ANSI/ISO SQL join table support.	See 5.4, “Enhanced ANSI/ISO SQL JOIN Support” on page 182.

New r17 Feature	Reference
The new SET statement provides a simple means of assigning SQL value-expressions to host variables.	See 5.5, “SET Host-variable Assignment” on page 186.
The UPDATE statement has been enhanced to allow a *query-expression* to be assigned to a column.	See 5.6, “Extended Use of query-expression” on page 187.
The SET OPTIONS command facility statement has been extended with a COMMAND DELIMITER option to provide alternate delimiters for separating commands.	See 5.7, “SET OPTIONS COMMAND DELIMITER” on page 188.
SYSCA.SINGLETON_NULL is a pseudo table with only one row and no columns. It can be used for easy evaluation of SQL functions with constant parameters.	See 5.8, “Pseudo Table SYSCA.SINGLETON_NULL” on page 189.
TCP/IP Enhancements	
TCP/IP support is enhanced to provide port number independence, allowing port numbers to be changed without impacting applications or DC/UCF system definitions.	See 6.1, “Port Number Independence” on page 192.
CA IDMS is enhanced to enable the selection of the stacks to be used by socket applications running in the CA IDMS system.	See 6.2, “Enhanced Stack Selection” on page 196.
A new TCP/IP system entity consolidates the definition of the TCP/IP runtime environment and allows multiple socket lines to be active at one time.	See 6.3, “New TCP/IP System Entity” on page 198.
A new TCP_NODELAY socket option enables two consecutive SEND socket requests to be executed without a delay between the sends.	See 6.4, “New TCP_NODELAY Option” on page 214.
CA IDMS now supports the following new socket functions: ■ GETSERVBYNAME ■ GETSERVBYPORT ■ IOCTL	See 6.5, “New Socket Functions” on page 215.

New r17 Feature	Reference
TCP/IP can now be used for CA IDMS DDS communications to improve the performance of database requests to geographically distributed databases.	See 6.6, “DDS Connectivity Using TCP/IP” on page 221.
Administrative and Operational Enhancements	
The linkable RHDCSDEL enhancement allows a user program to clean up security definitions for logically deleted users by linking to RHDCSDEL.	See 7.1, “Callable Security Cleanup” on page 234.
The DCMT DISPLAY SEGMENT command is enhanced to report the number of areas in a segment.	See 7.2, “DISPLAY SEGMENT Enhancement” on page 236.
A number of improvements in the detection and reporting of exceptional conditions facilitate problem diagnosis and correction.	See 7.3, “Enhanced Diagnostic Information” on page 237.
The new EXTIDENT session profile attribute makes the external identity visible to applications and ensures that it can be audited on all CVs that take part in a transaction.	See 7.4, “External Identity Auditing” on page 240.
IDD is enhanced to display only the load modules for a specified type.	See 7.5, “IDD Display Load Modules by Type” on page 243.
Index tuning enhancements are provided in the following areas: ■ The PRINT INDEX utility is enhanced to better determine whether an index needs tuning. ■ The TUNE INDEX utility is enhanced to perform more comprehensive tuning and to improve its ability to tune indexes while they remain available to online applications.	See 7.6, “Index Tuning Enhancements” on page 244.

New r17 Feature	Reference
The Lock Monitor (LOCKMON) system task is enhanced to do the following: ■ Report the area portion of a keep longterm lock ■ Display the longterm lock IDs	See 7.7, “LOCKMON Longterm Lock Display Enhancements” on page 259.
New LOOK functions report on SQL-defined database attributes and converted time stamps.	See 7.8, “LOOK Display Enhancements” on page 264.
A new operand is provided to enable including the volser of the current CA IDMS installation tape in the text of a message.	See 7.9, “New Message Replacement Operand” on page 266.
CA IDMS is enhanced with the following new startup parameters: ■ Multitasking queue depth allows you to set the multitasking queue depth at startup ■ Operating system subpool allows you to specify a different subpool at startup ■ zIIP allows you to control use of zIIP processors in z/OS	See 7.10, “New Startup Parameters” on page 267.
Online Print Log (OLP) usability is enhanced to do the following: ■ Specify seconds in FROM and TO time parameters ■ Flush the data automatically in the log buffer so the most recent data displays	See 7.11, “Online Print Log (OLP) Usability Enhancements” on page 270.
The REORG utility has been enhanced in the following areas: ■ Work file size estimation ■ Reduced record overflows ■ Reduced area sweeps when updating index UP pointers ■ New options for deleting work files	See 7.12, “REORG Enhancements” on page 272.

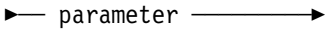
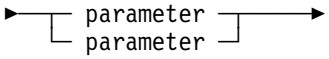
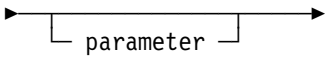
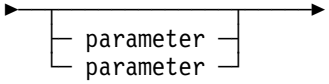
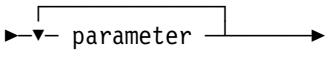
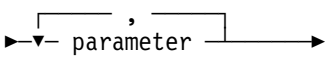
New r17 Feature	Reference
Support of dynamic allocation has been enhanced through changes in the way data set information is recorded in the DMCL.	See 7.13, “Run-time DMCL File Management” on page 281.
New SYSGEN and DCMT facilities are provided to enable control over the generation of system and task SNAPS.	See 7.14, “Snap Enhancements” on page 282.
To more easily handle large volumes of data, CA IDMS now supports large format database files and large and extended format work files for the REORG utility.	See 7.15, “Support for Large and Extended Format Files” on page 291.
Installation of the SVC has been enhanced to avoid inadvertent installation of an unsecured SVC.	See 7.16, “SVC Enhancements” on page 294.
When dynamically allocating a data set on z/OS, and the DSN is in use by another job, you can now request that local jobs and CV startup wait for the DSN rather than fail the request.	See 7.17, “Wait for In-Use Data Set” on page 295.
The IDMSIOX2 DB User Exit supports the following new functionality: ▪ A Pre-Open call is enhanced with a new flag to force the database file to input mode. ▪ A Pre-Write call is enhanced with a new flag to reopen the database file on the next write call.	See 7.18, “Forcing a Database File into Input Mode” on page 296.
Application Development Enhancements	
The ACCEPT database statistics command is enhanced to obtain the extended VIB statistics that are provided as part of the CA IDMS runtime system.	See 8.1, “Accept Extended Database Statistics DML Command” on page 298.
The ACCEPT command is enhanced to retrieve the system ID of the current DC/UCF system.	See 8.2, “Accept System ID DML Command” on page 301.
ADSORPTS is enhanced to report on the columns of SQL tables and an unlimited number of dialogs.	See 8.3, “ADSORPTS Enhancements” on page 302.

New r17 Feature	Reference
Assembler DML programs can make use of the following new functionality: ■ #CHAP allows changing the dispatching priority of the issuing task relative to its current priority. ■ #GETSTG allows requesting storage above the 16-megabyte line.	See 8.4, “Assembler Programming Enhancements” on page 303.
New date-time stamp built-in functions enable CA ADS and CA OLQ applications to convert date-time stamps between their internal and external formats.	See 8.5, “Built-In Functions for Date-Time Stamp Conversions” on page 305.
Debugging line support is provided in COBOL programs so that DML commands can be designated as debugging lines.	See 8.6, “COBOL Compiler Debugging Line Support” on page 316.
The IDMSDMLC precompiler is enhanced to issue a syntax error on a FIND/OBTAIN USING clause that specifies more than one field as a sortkey.	See 8.7, “FIND/OBTAIN WITHIN SET USING SORT KEY DML Statement” on page 317.
A new IDMSIN01 function is added to retrieve runtime environment information.	See 8.8, “IDMSIN01 Environment Information Function” on page 318.
CA IDMS Tools Product Enhancements	
The CA ADS Alive RECORD command is enhanced so that when records to be displayed are subschema built, only those record elements that are contained in the subschema view are displayed. Otherwise, all elements are displayed.	See 9.1, “CA ADS Alive RECORD Command Enhancement” on page 322.
CA IDMS Dictionary Migrator is enhanced to enable optional generation of MODIFY or REPLACE instead of ADD DDDL statements.	See 9.2, “CA IDMS Dictionary Migrator Enhancements” on page 323.

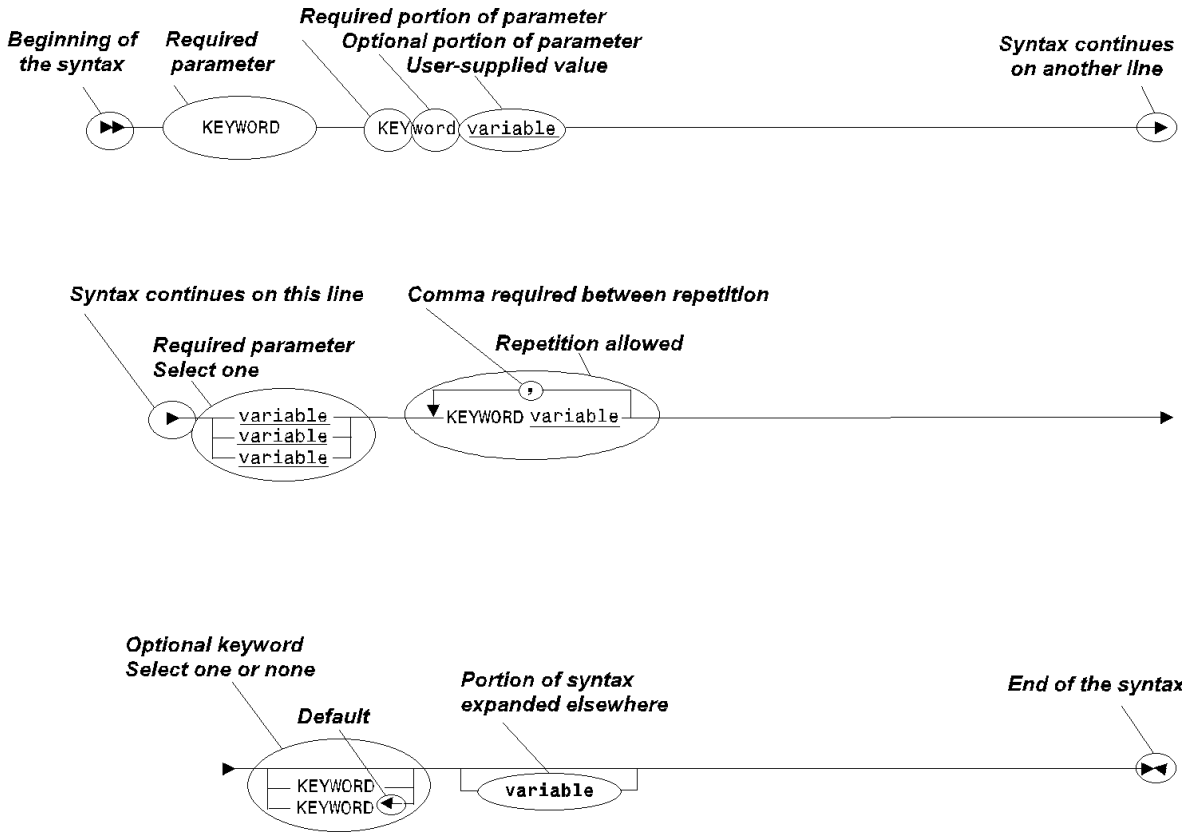
New r17 Feature	Reference
CA IDMS Journal Analyzer is enhanced to do the following: ■ Decompress and display data records that are compressed using custom built Data Characteristic Tables (DCTs). ■ Provide ranking by a cumulative value on the Management Ranking Report.	See 9.3, “CA IDMS Journal Analyzer Enhancements” on page 325.
CA IDMS Online Log Display is enhanced to force the log buffer to be written at task invocation, thereby ensuring the most recent log data is included in the display.	See 9.4, “CA IDMS Online Log Display Enhancement” on page 327.
The CA IDMS Tools Editor is enhanced to include the ECHO command that allows the primary line command to be preserved and redisplayed.	See 9.5, “CA IDMS Tools Editor Enhancement” on page 328.
A new PROKEEP installation parameter enables the automatic deletion of queue records after a specified time. This parameter can be used with the following CA IDMS Tools online products: ■ CA IDMS Dictionary Migrator Assistant ■ CA IDMS DME ■ CA IDMS Enforcer ■ CA IDMS Masterkey ■ CA IDMS SASO	See 9.6, “CA IDMS Tools Queue Record Deletion Enhancement” on page 330.
CA IDMS Tools users can now provide their own site-specific SEGMENT and DBNAME values for each database associated with a CA IDMS Tools product.	See 9.7, “CA IDMS Tools Site-Specific Segment Name and Database Name Enhancement” on page 331.

1.2 Syntax Diagram Conventions

The syntax presented in this guide uses the following notation conventions:

Syntax Convention	Description
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.
underlined lowercase	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.
	Indicates the beginning of a complete piece of syntax.
	Indicates the end of a complete piece of syntax.
	Indicates that the syntax continues on the next line.
	Indicates that the syntax continues on this line.
	Indicates that the parameter continues on the next line.
	Indicates that a parameter continues on this line.
	Indicates a required parameter.
	Indicates a choice of required parameters. You must select one.
	Indicates an optional parameter.
	Indicates a choice of optional parameters. Select one or none.
	Indicates that you can repeat the parameter or specify more than one parameter.
	Indicates that you must enter a comma between repetitions of the parameter.

Sample Syntax Diagram: The following sample explains how the notation conventions are used:



Chapter 2. Upgrading to r17

This chapter describes the considerations and actions that you must take to upgrade from r16 to r17 of CA IDMS. It contains the following topics:

2.1 Overview

You can upgrade to r17 from CA IDMS Releases 10.x, 12.0, 14.0, 14.1, 15.0, or 16.0. The CA IDMS r17 installation tape contains the conversion utilities needed to upgrade from all releases other than 10.x.

The *CA IDMS r17 Release Summary* describes the actions required to upgrade from r16 to r17 of CA IDMS. If you are upgrading to CA IDMS r17 from a release prior to CA IDMS r16, we recommend that you review all the intervening *CA IDMS Release Summary* documents for the cumulative requirements for your upgrade.

The following is a summary of actions required to upgrade the CA IDMS software from r16 to r17:

- Install the software into a new environment.
- Install the new SVC delivered with r17.
- Initialize the journal files using the r17 FORMAT utility before starting an r17 system for the first time.
- Offload the log file using a pre-r17 ARCHIVE LOG utility or initialize the log file before starting an r17 system for the first time.
- CICS users must create new IDMSINTC interface modules before using r17 runtime libraries in their CICS systems. If you are using UCFCICS or UDASCIC, you must create them also using the r17 installed macro library.
- Recompile all user-written programs that reference CA IDMS control blocks or journal files.
- If necessary, increase the size of XA storage and reentrant program pools.
- If necessary, update execution JCL to increase region sizes, reference additional IBM-supplied libraries, and add new SYSIDMS parameters.
- Update the CA IDMS task and program definitions using source members provided on the installation tape.
- Run IDMSDIRL against each dictionary containing the IDMSNTWK schema definition.
- Update each application dictionary with the protocol modules supplied on the installation tape.
- CA IDMS SQL and CA IDMS Visual DBA users should update the SYSCA and SYSTEM schemas in every catalog in which they reside.

- Review the following information to determine if any of these changes impact your environment:
  - The list of deprecated and stabilized features
  - The list of newly reserved words
  - The way in which CV startup manages the run-time DMCL and XA storage pool 255

2.2 Installing the Software

Follow the instructions documented in the CA IDMS installation guide for your operating system. Also, follow any special installation instructions outlined in the cover letter delivered with the installation tape. Be sure to install the r17 software into a new set of installation libraries. If you install r17 into your existing CA IDMS SMP/E or MSHP libraries, the results can be unpredictable.

2.3 Installing the SVC

A new SVC is delivered with r17. It should be used for all r17 systems. The SVC is downward compatible and can be used with Releases 14.1, 15.0, and 16.0 systems.

The procedure for installing the SVC has changed slightly in r17. For more information, see 7.16, “SVC Enhancements” on page 294.

2.4 Formatting Journal Files

The content of journal files is changed slightly in r17. You must initialize the journal files using the r17 FORMAT utility statement before the journal files can be used with an r17 system. At startup, the system verifies that the journal files are in the correct format.

If it is necessary to fall back to an earlier release of the software, the journal files must be reinitialized using the FORMAT utility and runtime libraries from the earlier release, otherwise warmstart fails.

Journal File Changes

The content of disk journal files is changed in r17 in the following ways:

- The JTRSEQ field in DSEG, JSEG, and JSGX records is always binary zeros.
- No TIME record is written to a journal block that contains a DSEG, JSEG, or JSGX record.

The content of archived journal files is also changed in the following ways:

- DSEG, JSEG, and JSGX journal records are no longer offloaded.
- A segment number is added to the TIME record.

These changes may impact user and third party software that directly access journal files.

2.5 Offloading the Log File

The format of the log file's statistics records is unchanged in r17, although the release identifier in these records is updated and contains the string 'R170'. If CA IDMS encounters a log record with an earlier release identifier, the ARCHIVE LOG utility issues the warning message:

```
NON 17.0 RECORD HAS BEEN ENCOUNTERED IN THE LOG, RECORD WILL BE BYPASSED
```

To avoid these messages and to separate logs from prior releases, offload the log file using the ARCHIVE LOG utility before installing r17 or initialize the log file if you do not need the log information.

If it is necessary to fall back to an earlier release of the software, any log file accessed by an r17 system must be offloaded or initialized prior to its use by a pre-r17 system.

2.6 Updating the CICS Interfaces

This section describes the upgrade requirements for the CICS interfaces.

2.6.1 Creating New CICS Interface Modules

Before a CICS system can use the r17 CA IDMS runtime library, you must create new IDMSINTC interface modules and UCF front-ends (if applicable) using the r17 source, macro, and object libraries. However, you do not have to create new IDMSCINT or IDMSCINL modules or relink user applications when upgrading to a CA IDMS r17 system.

The CICS IDMSINTL interface has been stabilized at the r16 level. For the applications that use this interface, we recommend using the IDMSINTC interface instead. You do not have to change or relink the application programs that were using IDMSINTL to use IDMSINTC. You can create an IDMSINTC interface module by compiling a CICSOPT with the same invocation parameters as the IDMSINTL interface being replaced.

2.6.2 Using the New IDMSINTC CICS Interface

Starting with r15, the IDMSINTC CICS interface module is delivered as an object module. It is installed using SMP/E and can be maintained using SMP/E. Various exits and parameterized options are provided so that all users can take advantage of this improved delivery method. The IDMSINTC macro is no longer delivered in source form.

2.6.3 Deprecated Macro Level Support

In compliance with all current releases of CICS, the r17 version of IDMSINTC no longer supports macro level programs. Should you still have such programs, they must be converted to command level before upgrading to the r17 version of the CICS interface.

2.6.4 Updating the CICS System

Install the CA IDMS r17 entity definitions into the CICS CSD. These definitions are contained in the installed source library members CICSCSD and CICSCSD2. Consult the appropriate IBM documentation to ensure that these definitions take precedence over any previously installed definitions for the corresponding entities.

If you issue SQL requests from CICS applications, ensure that the IBM Language Environment runtime support is available in the CICS region.

2.6.5 Upgrading from Versions Earlier than r16

If you are upgrading the CICS interface from versions earlier than r16, additional actions may be required, such as establishing a unique identifier for the CICS system and defining a resynchronization task and program.

Note: For more information, see the *CA IDMS r16 Release Summary* or the *CA IDMS System Operations Guide*.

2.7 Recompiling User-Written Programs

Several control block formats are changed in r17. Although in most cases, the only changes are the addition of new fields, we recommend that you recompile all programs, such as user-written exits, that reference CA IDMS control blocks using the r17 library.

Any program that directly accesses the DMCL run-time structures can be impacted by changes in the way data set information is recorded. For more information, see 7.13, “Run-time DMCL File Management” on page 281.

2.8 Increasing Storage and Program Pools

It may be necessary to increase the size of the XA storage pool (pool 255) due to increased storage requirements. Transactions that use SQL require about 80 KB additional XA storage in pool 255.

2.9 Updating Execution JCL

In most cases, no changes are needed to CA IDMS execution JCL. However, the following conditions may require updating this JCL to successfully upgrade to CA IDMS r17:

- Access to the IBM Language Environment (LE) runtime support is required in the following environments:
  - Central Version
  - Batch job steps that issue CA IDMS SQL statements or that execute IDMSBCF, IDMSDMLC, IDMSDMLP or ADSOBCOM
  - CICS systems in which CA IDMS SQL statements are executed

  If the LE runtime library is not in the linklist, it must be added to the execution JCL if it is not already present. For CV and batch, this means adding the LE SCEERUN load library to the CDMSLIB concatenation if CDMSLIB is used; otherwise, adding it to the STEPLIB concatenation. For CICS, this means enabling the CICS LE runtime support as described in the appropriate IBM documentation.
- It may be necessary to increase the region size to accommodate increased storage requirements.
- In r17, CV and batch jobs attempt to access a new type of data set using a DDNAME that by default begins with the character string "SYSTRK". You do not need to add new DD statements to your JCL to upgrade; however, if these DDNAMEs conflict with ones for your own files, use the SYSTRK_DDNAME_PREFIX SYSIDMS parameter to override the default.

  Note: For more information, see 3.1, “Change Tracking” on page 46.
- Any new SYSIDMS parameters, such as the SYSTRK_DDNAME_PREFIX parameter described previously, and the CV_STARTUP_XA_REGION_MB parameter described in 2.15, “Changes in CV Startup” on page 44, can be added directly to execution JCL or specified in a SYSIDMS load module tailored to your site.

  Note: For more information about creating a SYSIDMS load module, see the *CA IDMS Common Facilities Guide*.

2.10 Updating Task and Program Definitions

New CA-supplied task and program definitions are provided for r17. You should update the system definition using the batch sysgen compiler RHDCSGEN and source members provided on the installation tape. This can be accomplished by taking the following steps:

1. Perform an UPGRADE install to upgrade the definitions for SYSTEM 99.
2. Copy the task and program definitions from SYSTEM 99 to your system definition.

Note: For more information on the UPGRADE install process, see the CA IDMS installation manual for your operating system.

If it is necessary to fall back to an earlier release of the software, you can recreate the earlier versions of the task and program definitions by reinstalling them from the installation tape provided with the earlier release or by restoring the system dictionary from a backup that was taken prior to the upgrade. If returning to r15 or r16, it is not necessary to restore the earlier version of the task and program definitions.

2.11 Updating Dictionary Descriptions

New fields are added to the JOURNAL-1043 record in the IDMSNTWK schema and the IDMSNWKG subschema. You should update the definition of these records in every dictionary containing the IDMSNTWK schema description. To do this, use the IDMSDIRL utility. For more information about executing this utility, see the *CA IDMS Utilities Guide*.

You should also update each of your application dictionaries using the DLOD members appropriate to your environment. For more information, see the installation materials provided with the r17 installation tape.

2.12 Updating Catalogs

CA IDMS Visual DBA and CA IDMS SQL users should take the following steps to update every catalog in which the SYSCA or SYSTEM schemas are defined:

- Convert the catalog to update the SYSTEM schema definition
- Update the SYSCA schema definition

Any catalog, including non-SQL defined catalogs, may require special handling if falling back to an earlier release of CA IDMS. For more information, see 2.12.3, “Fallback Considerations” on page 40.

2.12.1 Updating the SYSTEM Schema

CA IDMS Visual DBA and CA IDMS SQL users should use the CONVERT CATALOG command to update the definitions of system tables in each catalog in which the SYSTEM schema is defined.

2.12.1.1 SYSTEM Schema Changes

When an r16 format catalog is converted, the definitions of the following tables are upgraded to their r17 definition and new columns in associated rows are initialized appropriately:

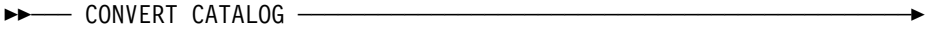
- SYSTEM.JOURNAL
- SYSTEM.TABLE

Changes introduced in earlier releases of the software are applied if they have not already been made. For a description of the changes made in earlier releases of CA IDMS, see the *CA IDMS r16 Release Summary*.

2.12.1.2 Executing the Catalog Conversion Utility

The CONVERT CATALOG utility can be invoked using the online command facility (OCF) or the batch command facility (IDMSBCF). If running in local mode or if converting from Release 12.0 or 12.01, you should back up the target catalog before executing this utility.

To convert a catalog to r17 format, enter the following statement:



After successful execution, CA IDMS issues one of the following informational messages to indicate the status of the conversion:

- If a catalog conversion is performed, the message indicates the number of rows of each type that are changed.
- If a catalog conversion is not required, an appropriate message is issued.

2.12.2 Updating the SYSCA Schema

CA IDMS Visual DBA and CA IDMS SQL users should update the SYSCA schema definition in each catalog in which the SYSCA schema is defined. This process varies slightly depending on your current release of CA IDMS. For more information about the required steps, see the installation materials provided with the r17 installation tape.

The following new procedures are defined to the SYSCA schema for r17 and are downward compatible with prior releases of CA IDMS:

- SYSCA.GET_DIAGNOSTICS
- SYSCA.GET_STATISTICS

2.12.3 Fallback Considerations

The changes implemented by the r17 catalog conversion utility are downward compatible with r14 and later releases of CA IDMS. If it is necessary to fall back to one of these earlier releases, no further action needs to be taken regarding the catalog.

Should it be necessary to reorganize (or unload and reload) a catalog updated by r17 CA IDMS after falling back to an earlier release, it may be necessary to do so using the r17 version of the IDMSCATZ subschema rather than the earlier version. This further action is necessary only if the catalog contains disk journal definitions and one of the following actions took place under r17 of CA IDMS:

- A disk journal file was created or altered
- The catalog was converted

Important: Converted definitions are not downward compatible with Release 12.0 or 12.01 of the software. For this reason, if you are upgrading from either of these releases, you should retain backup files of the catalog before converting it. Should it be necessary to fall back, you must restore the catalog and any database areas containing tables that were created or altered using the r17 version of the software.

2.13 Deprecated and Stabilized Features

This section describes the impact of r17 on the support of features available in earlier releases of CA IDMS.

2.13.1 Agent Technology Support

Support for CA IDMS agent technology has been dropped on all releases of CA IDMS.

2.13.2 BS2000/OSD Support

Support for the Fujitsu-Siemens BS2000/OSD operating system has been stabilized at the r16 level.

2.13.3 CICS IDMSINTL Interface

The CICS IDMSINTL interface has been stabilized at the r16 level. For more information, see 2.6.1, “Creating New CICS Interface Modules” on page 32.

2.13.4 CICS Macro Level Support

The r17 version of IDMSINTC no longer supports macro level programs. For more information, see 2.6.3, “Deprecated Macro Level Support” on page 32.

2.13.5 Optional APARs

Support for optional APAR OPT00224 has been replaced with the SYSIDMS parameters described in 7.17, “Wait for In-Use Data Set” on page 295.

Support for optional APAR OPT00135 has been removed. Retaining the offline status of an area across CV shutdowns can instead be achieved by using the PERMANENT option of the DCMT VARY AREA/SEGMENT command. A warning will be issued at CV startup if CA IDMS detects that this APAR is applied.

2.13.6 SYSIDMS Parameters

Support for SYSIDMS parameters TCP/IP_MAXIMUM_SOCKETS and TCP/IP_MAXIMUM_SOCKETS_PER_TASK is replaced with parameters on the new TCP/IP SYSGEN statement described in 6.3, “New TCP/IP System Entity” on page 198.

2.13.7 DCMT DISPLAY LINE Parameter

The IPINFO option of the DCMT DISPLAY LINE command is no longer supported for a SOCKET LINE. The information can now be obtained using the DCMT DISPLAY TCP/IP ALL command.

Note: For more information, see 6.3.4, “DCMT DISPLAY LINE Command” on page 208.

2.14 New Reserved Profile Attribute

EXTIDENT is now a CA-reserved profile attribute used to represent the external identity for a user session. If you have used this name for a user-defined attribute, you must select another name and update all places in which it is referenced.

Note: For more information about the EXTIDENT profile attribute, see 7.4, “External Identity Auditing” on page 240.

2.15 Changes in CV Startup

The way in which CV startup manages the run-time DMCL structures is changed in r17. In earlier releases, CA IDMS builds the run-time DMCL twice: once before warmstart and again during CV startup. The first run-time DMCL is deleted before the second is built.

In r17, the run-time DMCL is built only once. This occurs prior to warmstart, and the resulting structures are retained in memory throughout CV's execution. This change may impact user and third party software that directly access the run-time DMCL.

This change may also require the use of a new CV_STARTUP_XA_REGION_MB SYSIDMS parameter for CVs whose DMCL is extremely large. This new parameter allows for larger XA storage pool allocation during the initial stages of CV startup to hold the run-time DMCL.

CV_STARTUP_XA_REGION_MB=*nnn*

Specifies the size of the initial XA storage pool acquired during early CV startup.

nnn

Specifies the size in MB (megabytes) of the initial XA storage pool.

Default: 32 MB

Note: The XA storage used for CV startup is internally converted to a dynamic extension of XA storage pool 255 that is fully used. This may impact user and third party software that directly access the IDMS storage management control blocks.

Chapter 3. Non-Stop Processing

This chapter describes the non-stop processing enhancements. It contains the following topics:

3.1 Change Tracking

Change tracking enables changing the database environment of a Central Version (CV) in a fault-tolerant manner. Specifically, it enables the DBA to perform the following actions:

- Vary the data set name of a journal or database file within a CV without introducing the potential for a warmstart failure
- Vary a new version of a DMCL without introducing the potential for a warmstart failure
- Vary the status of an area or segment permanently on a CV without regard to subsequent page range changes
- Change the journal files in use by a CV and coordinate those changes with the associated archive journal jobs

Change tracking also provides an easy way for a local mode job to use the same database environment (definition and data sets) as a CV, even if that environment has been impacted by dynamic modifications.

More Information

- For more information about how CV tracks changes to its database environment, see 3.1.1, “Change Tracking and SYSTRK Files.”
- For more information about the steps necessary to implement change tracking, see 3.1.2, “Implementing Change Tracking” on page 47.
- For more information about how to manage change tracking and its impact on existing DCMT commands, see 3.1.3, “Managing Change Tracking” on page 54.

3.1.1 Change Tracking and SYSTRK Files

To track changes, CV maintains a description of its database environment in a new type of file called a SYSTRK file. The presence of such a file in the execution JCL of a CV triggers change tracking by that CV. A local mode job, such as a journal archive job, can share the description of the CV's database environment by referencing the same SYSTRK file in its execution JCL.

A SYSTRK file holds a description of the database environment most recently in use by the CV. During startup, an image of the current DMCL is written to SYSTRK along with information about database and journal files defined in the JCL. If DCMT commands issued during CV execution cause critical changes to its database environment, SYSTRK is updated to reflect those changes. If the CV fails, the runtime database definition is restored from SYSTRK during restart, ensuring that the files being updated at the time of failure are the ones recovered by warmstart unless explicitly overridden by changes in the JCL used to restart the CV.

Change tracking is optional. If no SYSTRK file is referenced in the execution JCL of the CV, change tracking is not in effect, meaning that the potential for a warmstart failure is introduced when varying in a new copy of a DMCL or dynamically changing the data set name of a file. Additionally, any permanent status established for an area whose page range is changed is lost or may be misapplied to another area whose page range is also changed.

Change tracking can be temporarily inactivated or disabled for a CV to facilitate expansion or replacement of a SYSTRK file. However, doing this impacts the ability to dynamically change the database environment.

- Inactivating change tracking has the effect of disallowing DCMT commands that would otherwise require updating SYSTRK.
- Disabling change tracking allows such commands to be executed, but a warning is issued indicating that manual intervention will be needed to restart CV should it fail before change tracking is re-activated.

3.1.2 Implementing Change Tracking

To implement change tracking for a CV, take the following steps:

1. Create and format two to four SYSTRK files. A minimum of two SYSTRK files are needed because mirroring is used to provide fault tolerance and recoverability in case of file damage.
2. Alter CV execution JCL to reference the SYSTRK files.
3. Alter the JCL for the associated archive journal job to also reference the SYSTRK files and to remove references to the disk journal files.
4. Optionally, change the JCL of other local mode jobs to reference the SYSTRK files and remove explicit references to database files.

Once change tracking has been initiated for a CV, its use should be continued indefinitely; otherwise, permanent area or journal statuses will be lost. Despite this, if you choose to discontinue the use of change tracking, do so as follows:

- Shutdown CV
- Remove all references to the SYSTRK files in the execution JCL of the CV and other jobs
- After restarting CV, use DCMT commands to re-establish permanent area statuses

Note: You can temporarily disable change tracking by issuing a DCMT VARY CHANGE TRACKING command or by overriding the SYSTRK file assignments to reference a dummy file.

More Information

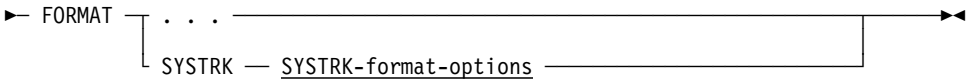
- For more information about sizing and formatting SYSTRK files, see 3.1.2.1, “Formatting SYSTRK Files” on page 48.

- For more information about options for altering CV startup JCL to reference SYSTRK files, see 3.1.2.2, “Referencing SYSTRK Files in Execution JCL” on page 53.

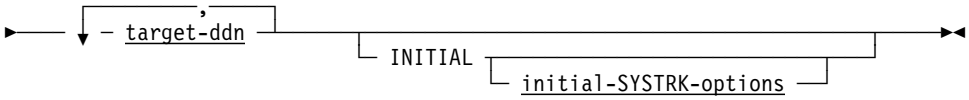
3.1.2.1 Formatting SYSTRK Files

A SYSTRK file must be formatted before it can be used. The FORMAT utility statement has been enhanced for this purpose.

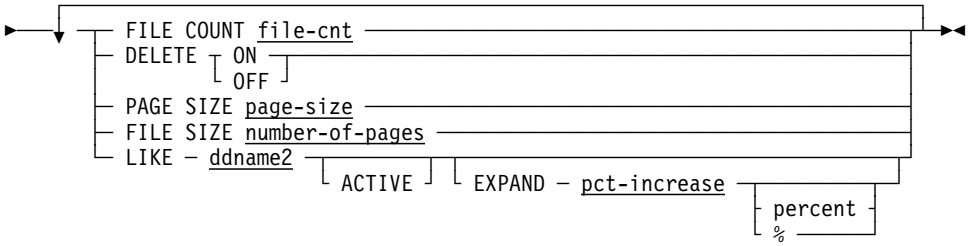
Syntax



Expansion of SYSTRK-format-options



Expansion of initial-SYSTRK-options



Parameters

SYSTRK
Indicates that one or more SYSTRK files are to be formatted.

target-ddn
Specifies the DDname or linkname of a SYSTRK file to be formatted.

Note: The *target-ddn* should *not* begin with the DDname prefix used for referencing SYSTRK files. Otherwise, CA IDMS attempts to use it for building the DMCL definition and fails if it cannot do so.

INITIAL
Indicates this is the first time the SYSTRK file is being formatted. If specified, no check is made to determine whether the file is in use by a CV. INITIAL must be specified the first time a file is formatted. It should not be specified when a file is being re-formatted unless you are sure that the file is not in use by a CV.

file-cnt
Specifies the number of SYSTRK files that are maintained as active mirrors. The *file-cnt* must be an integer in the range 2 through 4.

If *file-cnt* is not specified, the value is taken from the file identified by *ddname2*, if specified, or at run time from the file count currently in use by the DC/UCF system. If *file-cnt* has never been specified on a format for a related SYSTRK file, the first time that a DC/UCF system writes to a set of SYSTRK files, it sets *file-cnt* to be the lesser of the number of files currently allocated and 4. The value can be altered dynamically by a DCMT VARY CHANGE TRACKING command.

DELETE
Specifies whether the DC/UCF system is to automatically delete obsolete SYSTRK files.

ON
(z/OS and z/VM systems only) Enables automatic file deletion.

OFF
Disables automatic file deletion.

If DELETE is not specified, the value is taken from the file identified by *ddname2*, if specified, or at run time from the delete option currently in use by the DC/UCF system. If DELETE has never been specified on a format for a related SYSTRK file, the first time that a DC/UCF system writes to a set of SYSTRK files, it sets the option to OFF. The option can be altered dynamically by a DCMT VARY CHANGE TRACKING command.

page-size
Specifies the size of each page to be written to the SYSTRK file being formatted and must be an integer in the range 4088 through 32764. On z/VM, the value must be 4096. On z/OS, the maximum *page-size* is 32760. Do not specify *page-size* for VSAM files.

number-of-pages
Specifies the number of pages to be written to the SYSTRK file being formatted and must be an integer in the range 10 through 999,999.

ddname2
Specifies the DDname of a file from which the attributes and contents may be taken. If *ddname2* is specified together with either or both *page-size* and *number-of-pages*, the latter values override the respective attributes of the file identified by *ddname2*.

Note: *ddname2* should *not* begin with the DDname prefix used for referencing SYSTRK files unless the identified file contains the DMCL definition to be used during execution of the command facility.

ACTIVE
Indicates that the contents of the file identified by *ddname2* are to be copied to the file being formatted even if the file identified by *ddname2* is currently in-use by a CV.

pct-increase
Specifies the percentage increase in the number of pages written to the file being formatted over the number of pages in the file identified by *ddname2*. The *pct-increase* must be an integer in the range 0 through 1000.

Note: For more information about the DDname prefix used for referencing SYSTRK files, see 3.1.2.2, “Referencing SYSTRK Files in Execution JCL” on page 53.

Usage

Referencing SYSTRK Files During Format: To avoid I/O errors when building the runtime environment in local mode, only previously formatted files should be referenced using a DDname that matches the SYSTRK DDname prefix. For this reason, it is recommended that non-matching DDnames always be used to identify SYSTRK files being formatted.

SYSTRK File Attributes: If a SYSTRK file is being formatted to be added as a mirror of an existing file, the page sizes of the two files must be the same and the file being formatted must have at least as many pages as the existing file. If these criteria are not met the following conditions can occur:

- Any attempt to make the newly formatted file an active mirror of the existing file fails.
- If a LIKE parameter is specified, FORMAT does not copy the contents of the file specified by *ddname2* to the newly formatted file.

If INITIAL is not specified, the page size and number of pages of a file being formatted remain unchanged.

If INITIAL is specified, the number of pages written to the file is determined according to the following precedence rules:

- If a FILE SIZE parameter is specified, then the number of pages is *number-of-pages*.
- If a LIKE parameter is specified, then the number of pages is a value based on the number of pages in the file identified by *ddname2*. The value is calculated as:

  ```
  page-cnt * (100 + pct-increase) / 100
  ```

  page-cnt
  Specifies the number of pages in the file identified by *ddname2*.

  pct-increase
  Specifies the value in the EXPAND parameter, if specified or 0.

- The number of pages is a value based on the size of the current DMCL calculated as:

  ```
  ((DMCL-size + page-size -1) / page-size)*4
  ```

  DMCL-size
  Specifies the size of the DMCL load module.

  page-size
  Specifies the page size of the file being formatted.

In the latter two cases, the number calculated is rounded up to the next larger integer value. If the calculated value is less than the minimum, it is set to the minimum of 10. If the calculated value is larger than the maximum, it is set to the maximum of 999,999.

If INITIAL is specified, the size of the pages written to a non-VSAM file is determined according to the following precedence rules:

- If PAGE SIZE is specified, then the page size is *page-size*.
- If a block size has been assigned (for example, specified in JCL or at the time the file was created), then page size is the block size.
- If a LIKE parameter is specified, then page size is the *page-size* of the file identified by *ddname2*.
- Otherwise, the page size is 7548.

For VSAM files, the page size is the file record size. Any attempt to override this through a PAGE SIZE parameter fails.

Choosing a SYSTRK Page Size: In most cases, the FORMAT utility's default page size for SYSTRK files provides an acceptable tradeoff between memory, I/O, and disk space. Consider overriding the default only if the size of the DMCL is extremely large (500K or more). A larger page size will reduce I/Os and disk space requirements at the expense of slightly increased memory usage for buffers.

Estimating the Minimum Number of Pages for a SYSTRK File: To estimate the minimum number of pages needed for a SYSTRK file, perform the following steps:

1. Take the size of the DMCL load module used by the CV, divide it by the SYSTRK page size and multiply it by 2.5.
2. Multiply the resulting value with a factor to allow for overrides and growth. Overrides require approximately 100 bytes of space each and are generated for:
   - Each database or journal file defined in the execution JCL
   - Each dynamic change in the data set name of a database or journal file
   - Each dynamic change in the permanent status of an area
   - Each dynamic change in the status of a journal file

Copying SYSTRK File Contents: If a LIKE parameter is specified, the contents of the file identified by *ddname2* are copied to the files being formatted unless the file identified by *ddname2* is in use by a CV or the attributes of the two files are incompatible. If the contents of *ddname2* are not copied, a message indicates the reason.

If the file attributes are compatible, specify the keyword ACTIVE to force the copy to occur even if the file identified by *ddname2* is in use by CV. Only do this if you are sure that CV will not update the file while the copy is in progress, otherwise, the contents of the two files may not be the same which can lead to unpredictable results during CV restart. Ensure that a CV does not update its SYSTRK files by varying change tracking inactive before doing the format.

There is normally no need to force the contents of SYSTRK files to be copied. CV automatically updates newly formatted SYSTRK files as part of making them active mirrors.

Note: Formatting SYSTRK files can only be done in local mode.

JCL Considerations

When you submit a FORMAT statement through the batch command facility in local mode to format SYSTRK files, the JCL to execute the facility must include statements to define the SYSTRK files to be processed.

To format a new SYSTRK file, code the following:

```
//anydd   DD  DSN=user.systrkn,DISP=(NEW,disp),
//            UNIT=disk,VOL=SER=nnnnnn,
//            SPACE=(space)
```

To reformat an existing SYSTRK file, code the following:

```
//anydd   DD  DSN=user.systrkn,DISP=(OLD,PASS)
```

anydd
The target DDname specified in the FORMAT SYSTRK statement.

user.systrkn
Data set name of the SYSTRK file.

Note: For more information about generic JCL to execute the batch command facility, see the chapter pertaining to your operating system in the *CA IDMS Utilities Guide*.

Examples

Formatting SYSTRK Files: The following sample IDMSBCF statement instructs the FORMAT utility to format three new SYSTRK files (track01, track02, track03). It directs the utility to format the files to have the default page size of 7548 and contain 60 pages each. CA IDMS will maintain 3 active mirrors.

```
format systrk track01, track02, track03
   initial file count 3
   file size 60;
```

Sample Output

Formatting SYSTRK Files: The following listing was generated after the successful completion of the previous FORMAT SYSTRK example.

```
IDMSBCF                          CA IDMS Batch Command Facility


FORMAT SYSTRK TRACK01, TRACK02, TRACK03
       INITIAL FILE COUNT 3 FILE SIZE 60;

Systrk file TRACK01 page size 7,548 file size 60
        delete NULL file count 3.
Systrk file TRACK02 page size 7,548 file size 60
        delete NULL file count 3.
Systrk file TRACK03 page size 7,548 file size 60
        delete NULL file count 3.
Status = 0        SQLSTATE = 00000

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

3.1.2.2 Referencing SYSTRK Files in Execution JCL

SYSTRK files are referenced in execution JCL using file assignments whose DDname begins with the value specified by the SYSIDMS parameter: SYSTRK_DDNAME_PREFIX. The default value for this parameter is SYSTRK. Because mirroring is used to provide recovery in the event of file damage, multiple SYSTRK files must be used at runtime.

Depending on your operating system, you can reference SYSTRK files in execution JCL by including one of the following:

- A model SYSTRK file assignment
- A file assignment for each SYSTRK file to be used

Referencing SYSTRK files using a model is the recommended approach because it enables the set of active SYSTRK files to be changed without impacting execution JCL.

Using a Model SYSTRK File Assignment: A model SYSTRK file assignment has a DDname that is the SYSTRK_DDNAME_PREFIX. It references a data set whose name is used as the prefix for constructing the names of the real SYSTRK files by appending a numeric suffix ranging from 1 to 9 to the end of the model's data set name.

For example, if the SYSTRK_DDNAME_PREFIX is SYSTRK and a model SYSTRK file assignment references a data set name of DBDC.SYSTEM73.SYSD with a disposition of SHR, CA IDMS attempts to discover through dynamic allocation the data sets shown in the following table:

DDNAME	DSN	DISP
SYSTRK1	DBDC.SYSTEM73.SYSD1	SHR
SYSTRK2	DBDC.SYSTEM73.SYSD2	SHR
"	"	"
SYSTRK9	DBDC.SYSTEM73.SYSD9	SHR

The presence of a file assignment whose DDname is SYSTRK*n* overrides the generated data set name and disposition. If an overriding file assignment refers to a dummied file, the overridden file is not used.

If a model SYSTRK file assignment refers to a dummied file, it is equivalent to not including a model file assignment in the execution JCL.

Note: The data set referenced by a model SYSTRK file assignment is never opened. While the file must exist, its contents are not relevant.

Using Individual SYSTRK File Assignments: The DDNAME for an individual SYSTRK file assignment is the SYSTRK_DDNAME_PREFIX suffixed with a digit from 1 to 9. For example, if the SYSTRK_DDNAME_PREFIX is SYSTRK, the DDNAMEs that can be used for SYSTRK files are SYSTRK1, SYSTRK2, . . . SYSTRK9.

Using this approach to reference SYSTRK files requires that a file assignment be included in the JCL for each SYSTRK file to be used. To change the set of files being used, you must update every set of JCL in which they are referenced.

3.1.3 Managing Change Tracking

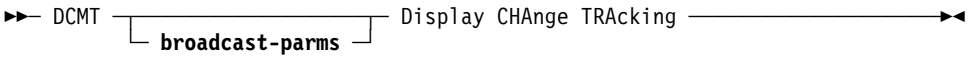
This section describes the facilities that are available for managing change tracking. The following topics are covered in this section:

- The ability to monitor the status of change tracking by issuing a DCMT DISPLAY CHANGE TRACKING command
- The ability to alter the status of change tracking by issuing a DCMT VARY CHANGE TRACKING command
- The ability to increase the size of the SYSTRK files by using the FORMAT utility statement in conjunction with the DCMT VARY CHANGE TRACKING command

3.1.3.1 DCMT DISPLAY CHANGE TRACKING Command

Displays information on the status of change tracking and on the SYSTRK files currently known to the system.

Syntax



Parameters

broadcast-parms

Specifies to execute the DCMT command on all or a list of data sharing group members.

Note: For more information about broadcasting and **broadcast-parms**, see How to Broadcast System Tasks in the *CA IDMS System Tasks and Operator Commands Guide*.

Example

```
DCMT D CHANGE TRACKING
Change Tracking  -  Status   Delete   PageCnt   Target-FileCnt   Actual-FileCnt
                    ACTIVE      OFF        21                4                2

SYSTRK contents              Size   PagCnt   Pct   Last Updated (time zone: UTC)
DMCL + file information      35076       5   24%   2007-06-14-17.06.20.642828
Permanent area statuses          0       0    0%   2007-06-14-17.06.23.349039
Journal status overrides         0       0    0%   2007-06-14-17.06.20.674228
Control information          30192       4   19%   N/A
--------------------------  ------  ------  ----
Total:                       65268       9   43%

File Name   MirrorStat    MODE  ErrStat  PagSize   PagCnt   Fl-Type     DD-Name
SYSTRK1     ACTIVE        Clos        0     7548       21   non-VSAM    SYSTRK1
  DSname:  DBDC.SYSTEM73.SYSTRK1                          DISP=SHR VOLSER:CULL06
     FORMAT  datetime (time zone: UTC) 2007-06-08-13.31.15.587813
     CONTROL datetime (time zone: UTC) 2007-06-14-17.06.15.640034

SYSTRK2     ACTIVE        Clos        0     7548       21   non-VSAM    SYSTRK2
  DSname:  DBDC.SYSTEM73.SYSTRK2                          DISP=SHR VOLSER:CULL06
     FORMAT  datetime (time zone: UTC) 2007-06-08-13.31.15.674543
     CONTROL datetime (time zone: UTC) 2007-06-14-17.06.15.640034
```

Usage

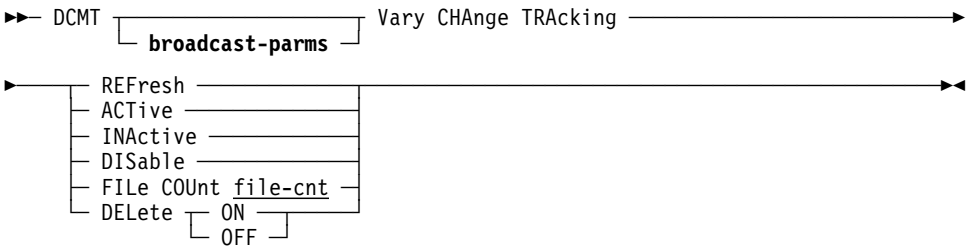
CHAnge TRAcking displays the following attributes:

- Current change tracking status
- Target number of files to be maintained as active mirrors
- Current delete option setting
- Page count in effect for SYSTRK files
- Summary of file content and space utilization
- For each known SYSTRK file
  - DSName, filename, initial format date, page size and file size
  - Mirroring and usage status

3.1.3.2 DCMT VARY CHANGE TRACKING Command

DCMT VARY CHANGE TRACKING changes the status of change tracking.

Syntax



Parameters

broadcast-parms
Specifies to execute the DCMT command on all or a list of data sharing group members.

Note: For more information about broadcasting and **broadcast-parms**, see How to Broadcast System Tasks in the *CA IDMS System Tasks and Operator Commands Guide*.

REFresh
Adds new SYSTRK files to the list of mirrors and terminates use of older or non-existent mirrors. New SYSTRK files are made active before terminating the use of other SYSTRK files. In the process of becoming active mirrors, the contents of new files are brought up-to-date if necessary. Terminated files are deleted if the current delete option is ON. After a successful refresh, the status of change tracking is active.

ACTive
If change tracking is active, this option has no effect. If change tracking is inactive or disabled, this option activates change tracking and enables the

execution of DCMT commands that update information in SYSTRK. An automatic refresh is done as part of activation. Any files with out-of-date contents are brought up-to-date as part of the process of becoming active. The contents of all SYSTRK files are refreshed if change tracking was previously disabled. At least one SYSTRK file must exist and achieve active mirror status before certain DCMT commands can be executed.

Note: For a list of impacted commands, see “DCMT Commands that Require Active Change Tracking” on page 58.

INActive
Deactivates change tracking and prevents the execution of certain DCMT commands. All SYSTRK files are closed and deallocated except those that have encountered an I/O error.

DISable
Disables change tracking but does not prevent the execution of certain DCMT commands. Disabling change tracking should only be used in an emergency situation because the inability to record changes in the SYSTRK files may lead to incorrect recovery during warmstart and incorrect area statuses on system restarts.

file-cnt
Specifies the target number of files to be maintained as active mirrors. *file-cnt* must be an integer in the range 2 through 4. To affect the number of files actually in use while change tracking is active, issue a DCMT VARY CHANGE TRACKING REFRESH command.

DELete
Specifies whether the DC/UCF system automatically deletes obsolete SYSTRK files.

ON
(z/OS and z/VM systems only) Enables automatic file deletion.

OFF
Disables automatic file deletion.

Usage

Refreshing SYSTRK File Use: If the REFresh option is specified or change tracking is activated by specifying the ACTive option, the system replaces existing SYSTRK files with more recently formatted ones. This is useful in expanding the size of SYSTRK files because newer files can have more pages than existing files. To increase the amount of SYSTRK space available, all files must be replaced with files having the larger number of pages.

The following algorithm is used when refreshing SYSTRK file usage:

- A discovery process determines all SYSTRK files that are referenced either directly or indirectly through a model DD statement in the execution JCL.
- Each file is opened and read to determine its characteristics and control information.
  - Any file that cannot be opened, that encounters an I/O error or whose header is invalid, is discarded.
  - Any file whose characteristics are incompatible with the current SYSTRK file characteristics is discarded. To be compatible, its page size must be the same as the current page size and the number of pages must not be less than the current number of pages.
- All out-of-date files are brought up-to-date by copying the content from other files, or by writing new information.
- If the count of active mirrors is greater than the target, then the following actions occur:
  - The use of files is terminated until the count of active mirrors is equal to the target. The next file terminated is the one with the oldest initialization timestamp.
  - For each file whose use is terminated, the following actions occur:
    - If automatic file deletion is enabled, the file is deleted.

      Note: This may take some time if the file is in use by another job.
    - If automatic file deletion is disabled, a message is written indicating that the file is no longer being used and should be deleted manually.
  - If the count of active mirrors is equal to the target, the current number of pages is set to be the smallest of all active files.

DCMT Commands that Require Active Change Tracking: If change tracking is in use for a CV, the following commands are impacted by the status of change tracking:

- DCMT VARY DMCL
- DCMT VARY FILE if it changes the data set name of the file
- DCMT VARY AREA or SEGMENT if it changes the permanent status of an area
- DCMT VARY JOURNAL FILE if it changes the data set name or the permanent status of a journal file

Note: If change tracking is inactive, these commands are prohibited. If it is disabled, a warning is issued if these commands are executed.

3.1.3.3 Expanding SYSTRK Files

A set of SYSTRK files can be expanded by using the FORMAT utility statement. The procedure for expanding files while CV remains active differs depending on whether the SYSTRK files are referenced through a model SYSTRK file assignment or if they are referenced using individual file assignments.

Expanding Files Referenced Through a Model SYSTRK File Assignment:
Assuming two SYSTRK files are in use, take the following steps to increase the size of the SYSTRK files while the CV remains active:

1. Allocate larger SYSTRK files using data set names that conform to the standard established by the model DD statement.
2. Format the larger files by executing a FORMAT utility statement as follows:

   ```
   FORMAT SYSTRK DD1, DD2 INITIAL LIKE DD3 EXPAND 20 PERCENT
   ```

   Where DD1 and DD2 are the DDnames of file assignments referencing the new SYSTRK files, and DD3 is the DDname of a file assignment referencing one of the old SYSTRK files. In this example, the files are being expanded 20 percent over their current size.
3. Replace use of the old files with the new files by issuing the following command:

   ```
   DCMT VARY CHANGE TRACKING REFRESH
   ```

   When the old files are no longer in use, they can be deleted.

Expanding Files Referenced Through Individual File Assignments:
Assuming two SYSTRK files are in use, take the following steps to increase the file size while the CV remains active:

1. Allocate larger SYSTRK files using new data set names
2. Close and deallocate the current set of SYSTRK files by issuing the following command:

   ```
   DCMT VARY CHANGE TRACKING INACTIVE
   ```
3. Format the larger files by executing a FORMAT utility statement as follows:

   ```
   FORMAT SYSTRK DD1, DD2 INITIAL LIKE DD3 EXPAND 20 PERCENT
   ```

   Where DD1 and DD2 are the DDnames of file assignments referencing the new SYSTRK files, and DD3 is the DDname of a file assignment referencing one of the old SYSTRK files. In this example, the files are being expanded 20 percent over their current size.
4. Scratch the old files. Rename the new files to have the same data set names as the old files.
5. Allocate the new files and make change tracking active by issuing the following command:

   ```
   DCMT VARY CHANGE TRACKING ACTIVE
   ```

3.1.4 Change Tracking Impact

This section describes the impact that change tracking has on the execution of existing DCMT commands and the new SYSIDMS parameters related to change tracking.

3.1.4.1 DCMT DISPLAY DATABASE Command

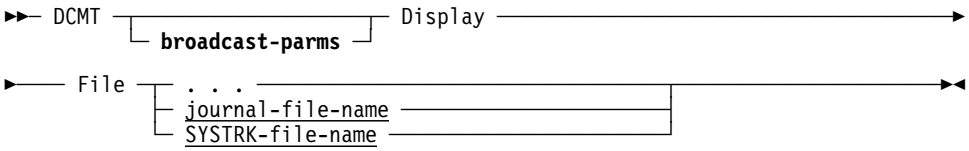
The output of this command now includes the output from a DISPLAY CHANGE TRACKING command so that it includes output from all of the following commands:

- DCMT DISPLAY AREA
- DCMT DISPLAY BUFFER
- DCMT DISPLAY FILE
- DCMT DISPLAY JOURNAL
- DCMT DISPLAY TRANSACTION
- DCMT DISPLAY CHANGE TRACKING

3.1.4.2 DCMT DISPLAY FILE Command

The DCMT DISPLAY FILE command displays information about a specified file. It has been enhanced to support journal and SYSTRK files.

Syntax



Parameters

File

Displays information about one or more files.

journal-file-name

Specifies the name of a disk or archive journal file.

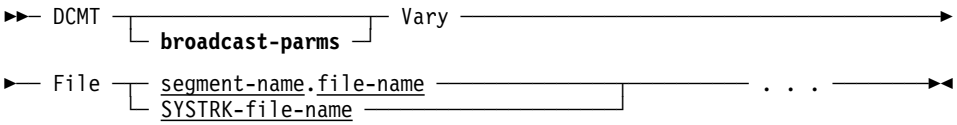
SYSTRK-file-name

Specifies the name of a SYSTRK file.

3.1.4.3 DCMT VARY FILE Command

The DCMT VARY FILE command alters information about a specified file. It has been enhanced to enable altering information about SYSTRK files and to update SYSTRK when changing the data set name of a database or journal file.

Syntax



Parameters

File

Identifies a specific file.

segment-name

Specifies the segment associated with the file.

file-name

Specifies the name of the file.

SYSTRK-file-name

Specifies the name of the SYSTRK file.

ACtive

Enables access to the file and sets its status to something other than 9999 if this is a database file. If the file is not open, it is opened the next time it is accessed. Varying the file active allows suspended transactions that are waiting on the file to resume execution.

If this is a SYSTRK file, its mirror status is changed to active or activating. Before an activating mirror becomes active, its contents are brought up-to-date.

Inactive

Disables access to the file and sets its status to 9999 if this is a database file. The ability to vary database files inactive is provided to simulate I/O errors for the purpose of testing recovery procedures.

If this is a SYSTRK file, its mirror status is changed to inactive. If this is the last active mirror, change tracking is inactivated.

ALlocate

(z/OS and z/VM systems only) Allocates the file dynamically, using its currently assigned data set name and other options specified on its definition.

DSname new-dataset-name

Changes the data set name of a database file in the runtime DMCL to *new-dataset-name*. If the file has not been opened, then only the DSname is changed. If the file has previously been opened, then the DSname is changed, and the DDname is cleared to blanks.

Data set names of SYSTRK files cannot be changed.

Usage

Changing the Data Set Name of a File: The ability to change the data set name of a file through a DCMT VARY FILE command is provided for emergency situations only, such as, when a data set is physically damaged and cannot be recovered using its original name. Data set name changes made through a DCMT VARY FILE command are temporary and are not preserved after a normal shutdown. Furthermore, they introduce the potential for incorrect recovery during warmstart unless change tracking is active or appropriate changes are made to the execution JCL of the system to ensure that the correct data set is referenced.

Note: To make permanent changes to the data set name of a file, change its definition in the dictionary and use a DCMT VARY DMCL command to make the change effective within a DC/UCF system.

To change a data set name through a DCMT VARY FILE command, the following conditions must be met:

- The file must have encountered an I/O error, been varied inactive or its area must be varied offline using a DCMT VARY AREA or SEGMENT command.
- The file must be deallocated, using the FORCE option if necessary.
- If change tracking is in use, it must either be active or disabled.

3.1.4.4 DCMT VARY AREA/SEGMENT Command

The DCMT VARY AREA/SEGMENT command updates information about one or more database areas. It has been enhanced to maintain permanent area statuses in the SYSTRK files if change tracking is in effect for the CV.

Changing the Area Status Permanently: A permanent area status is one that is retained until it is changed by another DCMT VARY command, or until the system journal or SYSTRK files are formatted. The area status is retained across normal shutdowns and across abnormal terminations, provided the warmstart option of the area in the DMCL specifies MAINTAIN CURRENT STATUS.

Note: The permanence of an area status has no effect on physical area locks. It only affects the mode in which the area is accessed when the system is next started. If the DC/UCF system is shut down normally, all physical area locks held by the system are removed, regardless of whether the area status of the system was assigned as UPDATE PERMANENT.

If change tracking is in use for the DC/UCF system, permanent area statuses are recorded in the SYSTRK files. Status entries are identified by area name and are deleted when their associated area is no longer in the runtime DMCL. A vary that affects the permanent status of an area fails if change tracking is inactive and receives a warning if it is disabled.

If change tracking is not in use for the DC/UCF system, permanent status entries are recorded in the system journals and are identified by page group and low-page number. If a page group or low-page number of an area is changed, an existing permanent status entry cannot be matched against the area. If this happens, the usage mode of the area defaults to the usage-mode specified in the DMCL and the orphaned status entry for the area remains in the journals until they are formatted. It is also possible for an orphaned status entry to be misapplied to a new area with a matching low page number and page group.

3.1.4.5 DCMT VARY DMCL Command

The DCMT VARY DMCL command updates the database environment in use by the CV to reflect the current DMCL load module. It has been enhanced to maintain an up-to-date description of that environment in the SYSTRK files if change tracking is in effect for the CV.

Using a New Copy of the Database Load Module: DCMT VARY DMCL NEW COPY allows programs running under the DC/UCF system to benefit from changes made to the database definition without having to recycle the system. For example, an area can be added to an existing segment while the system remains active.

When a DCMT VARY DMCL NEW COPY command is issued, CA IDMS applies changes to the database definition that have been made by the following DDL statements:

- CREATE/ALTER/DROP AREA
- CREATE/ALTER/DROP BUFFER
- CREATE/ALTER/DROP FILE
- CREATE/ALTER/DROP SEGMENT
- CREATE/ALTER/DROP DISK JOURNAL

Additionally, CA IDMS loads a new copy of the database name table. The system must be recycled to implement changes made to the journal buffer, to VARY in a DMCL generated under a release level that is different from that of the current DMCL, or to remove or replace all active disk journal files at the same time.

Impact of Change Tracking: If change tracking is in use, a DCMT VARY DMCL NEW COPY command can only be issued if change tracking is active or disabled. We recommend that change tracking be active in systems in which new copies of DMCLs are to be varied online.

Note: For more information, see “Recovery Considerations and DMCL Changes” on page 64.

What DC/UCF Does in Response to a New Copy Command: In response to a DCMT VARY DMCL NEW COPY command, CA IDMS performs the following actions:

- Compares the contents of the runtime DMCL with the new DMCL load module, identifying entities that have been added, changed or removed. Changes to entities are detected by comparing their timestamps.
- Displays all of the changes to the user.
- Unless NOPROMPT was specified, issues the following prompt: 'Continue with Vary DMCL Yes or No?'. Specifying No negates the changes and allows the system to run as before. Specifying Yes causes the changes to be incorporated into the runtime DMCL as described in the following steps.
- Quiesces those areas and disk journals that have been removed or impacted by a change.
- Updates the runtime DMCL to reflect the new DMCL load module.
- If change tracking is active, writes an image of the new runtime DMCL to the SYSTRK files.
- Swaps to a new active journal file and writes the timestamp from the new DMCL load module to the active journal file.
- Reopens the disk journals, buffers, files, and areas using the definitions contained in the new runtime DMCL. New areas are opened in the mode specified in the DMCL and existing areas are opened in the mode they were in prior to the vary operation.

When quiescing access to impacted entities, the following actions are taken:

- Areas that have been dropped or modified are varied offline.
- Their associated files are closed and de-allocated.
- Buffers that have been dropped or modified or whose associated files are changing are closed.
- Disk journals that have been dropped or modified are varied offline.

Note: If areas or disk journals must be varied offline, the vary operation could have a lengthy completion time. Before responding Yes to the prompt, note the areas affected by the change and the transactions that are accessing those areas. If disk journals are being changed, look for transactions that may depend on those journal files for recovery. Look especially for long-running transactions that do not issue frequent commits.

Recovery Considerations and DMCL Changes: If change tracking is active when a DCMT VARY DMCL NEW COPY is issued, CA IDMS ensures that any subsequent warmstart uses the correct data sets and DMCL definition by recording the new definition in the SYSTRK files. If a failure occurs prior to writing the new DMCL to SYSTRK, the system restarts using the old DMCL definition and data sets. Otherwise, the system restarts using the new definition and data sets. If the write to SYSTRK fails because of an I/O error or

out-of-space condition, the vary operation continues but change tracking is varied inactive, and manual intervention is needed to restart the CV in the event of a failure. Therefore, you should correct the cause of the failure and vary change tracking active as soon as possible. If the CV fails before these corrective actions are taken, specify IGNORE_SYSTRK_DMCL=ON in the SYSIDMS file when restarting the system, and ensure that the execution JCL does not reference obsolete data sets. If IGNORE_SYSTRK_DMCL=ON is not specified, warmstart fails due to a mismatch between the timestamp in the DMCL and that recorded on the journal files.

If change tracking is disabled or not in use when a DCMT VARY DMCL NEW COPY is issued, manual intervention may be necessary to ensure correct recovery in the event that a subsequent warmstart is needed. The necessary actions depend on when the failure occurs:

- If the failure occurs before the timestamp of the new DMCL was recorded in the journal files, warmstart fails due to a mismatch between the timestamp in the DMCL load module and the timestamp recorded in the journal. The old DMCL load module must be restored, and the system restarted with JCL that reflects the data sets in use at the time of the failure.
- If the failure occurs after the new DMCL timestamp was recorded in the journal files, no timestamp mismatch occurs. However, before restarting, the JCL may need to be adjusted so that obsolete DD statements do not override files whose data set names were changed by the DCMT VARY DMCL command.
- In either case, if change tracking was disabled at the time of the failure, IGNORE_SYSTRK_DMCL=ON must be specified in the SYSIDMS file when restarting the system.

3.1.4.6 New SYSIDMS Parameters

New SYSIDMS parameters have been added in support of change tracking.

IGNORE_SYSTRK_DMCL=ON|OFF
Specifies whether to disable building the runtime DMCL from the SYSTRK file and instead force the use of the DMCL load module.

ON
Specifies to disable use of SYSTRK for building the runtime DMCL.

OFF
Specifies to build the runtime DMCL from SYSTRK if the CV was not previously shutdown normally.

Default: OFF

SYSTRK_DDNAME_PREFIX=*xxxxxxx*
Specifies the DDname prefix to be used for referencing SYSTRK files in execution JCL.

xxxxxxx
Specifies the 1- to 7-character DDname prefix.

Default: SYSTRK

3.2 Dynamic Journal Files

Dynamic Journal files provide enhanced 24x7 capabilities by enabling the journal files in use by a CV to be changed while the system remains active. Journal files can be added, removed, or replaced without processing interruption. If change tracking is employed, the following additional functionality is provided:

- Use of a journal file can be permanently disabled. The journal file remains unused until explicitly re-enabled instead of being automatically re-enabled when the CV is restarted after a normal shutdown.
- Changes made to CV journals can be automatically shared with the jobs that archive those journals.

To implement this feature, a data set name can be specified during journal file definition and if present, is used to dynamically allocate the file at runtime. New options on the DCMT VARY JOURNAL command enable changing a journal's data set name and varying journal files on and offline. Journal files can be dynamically added, replaced or removed by varying a new copy of the DMCL.

To dynamically change the data set name of a journal file or remove a journal file, it must be "quiesced." A journal file reaches a quiesced state when no active transaction is dependent on it for recovery and its contents have been archived. The ability to quiesce use of a journal file is new for r17.

3.2.1 Dynamically Adding or Removing a Journal File

During a DCMT V DMCL NEW COPY command, the system automatically quiesces use of journal files that have been removed or whose definition has had critical changes.

A new FILE OFFLINE option on the DCMT V JOURNAL command enables the DBA to quiesce use of a specific journal file. When a CV restarts after an abnormal termination, an offline journal remains offline. A PERMANENT option on the VARY command enables the DBA to specify that the offline status should be retained across a shutdown and restart. The PERMANENT option is only available if change tracking is active.

To Dynamically Add a Journal File to a CV

1. Define the new disk journal and specify its data set name.

   ```
   CREATE/ALTER DISK JOURNAL dmcl-name.journal-name DSNAME 'data-set-name'
   ```

2. Generate, punch, and link edit the affected DMCL load module.
3. Allocate and format the new journal file using the newly-created DMCL.
4. Activate the change in the CV by issuing a DCMT VARY DMCL NEW COPY command.

To Dynamically Remove a Journal File

1. Delete the disk journal.

   ```
   DROP DISK JOURNAL dmcl-name.journal-name
   ```

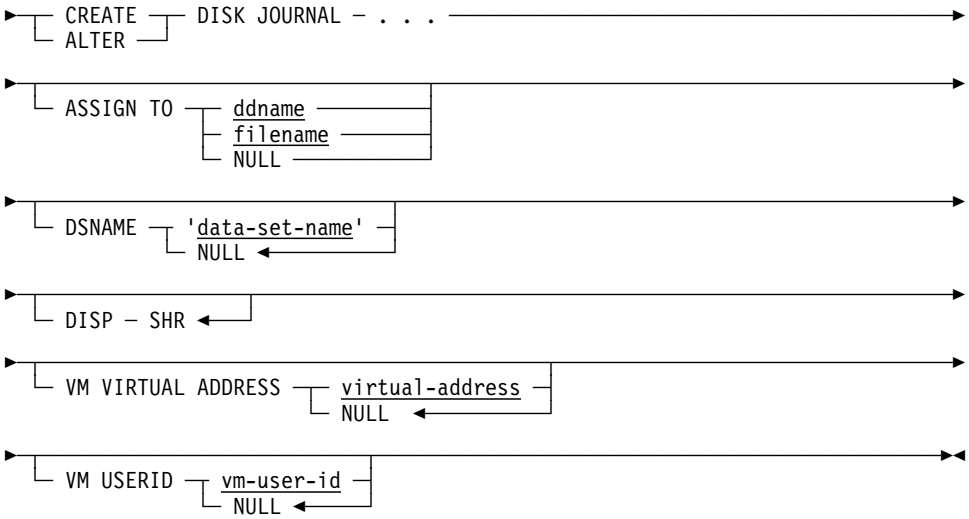
2. Generate, punch, and link edit the affected DMCL load module.
3. Activate the change in the CV by issuing a DCMT VARY DMCL NEW COPY command.

Note: It is not possible to apply critical changes to all active journal files at one time when varying in a new copy of the DMCL. To make such changes without recycling CV, repeat the above process and replace or remove only some of the journal files each time.

3.2.2 CREATE/ALTER DISK JOURNAL: New Parameters

To enable dynamic allocation of journal files, new parameters have been added to the CREATE DISK JOURNAL and ALTER DISK JOURNAL DDL statements:

Syntax



Parameters

ASSIGN TO

Specifies an external file name. Every external file name in a DMCL definition must be unique. In z/VSE without DYNAM/D, an external file name must be specified. In other environments, if the external file name is not specified, a data set name or VM virtual address must be specified.

ddname

(z/OS and z/VM systems only) Specifies the external name for the file. *ddname* must be a 1- through 8-character value that follows operating system conventions for ddnames.

filename

(z/VSE systems only) Specifies the external name for the file. *filename* must be a 1- through 7-character value that follows operating system conventions for file names.

NULL

Sets the external file name to blanks. It is equivalent to not specifying an external file name for a file. This option is not valid under z/VSE unless DYNAM/D is being used.

DSNAME data-set-name

Specifies the name of the data set to be used when dynamically allocating the journal file for z/OS, z/VSE, and OS-format data sets under z/VM.

data-set-name must conform to host operating system rules for forming data set names.

A *data-set-name* that includes embedded periods must be enclosed in single or double quotation marks.

Under z/VM, the DSNAME parameter or VM VIRTUAL ADDRESS and USERID parameters, or both can be specified.

NULL

Sets the data set name to blanks. This is equivalent to not specifying a data set name for a file.

DISP

(z/OS and z/VM systems only) Specifies the disposition to be assigned when the file is dynamically allocated.

SHR

Indicates that the data set used for the file is available to multiple DC/UCF systems and local mode applications at a time.

Under z/VM, DISP SHR causes a link with an access mode of multiple read (MR).

SHR is the default when you do not include the DISP parameter in a CREATE JOURNAL FILE statement.

VM VIRTUAL ADDRESS 'virtual-address'

(z/VM systems only) Specifies the virtual address of the minidisk used for the journal file. *virtual-address* is a hexadecimal value in the range X'01' to X'FFFF'.

NULL

Sets the virtual address to blanks. On CREATE statements, this is equivalent to not specifying a virtual address for a file. On ALTER statements, it removes any previous virtual address specification for the file.

VM USERID vm-user-id

(z/VM systems only) Identifies the owner of the minidisk used for the journal file. *vm-user-id* is a 1- to 8-character value.

A user ID for an OS-format data set must be specified. The user ID is optional for CMS-format files.

If a user ID is not specified for a CMS-format file, CA IDMS assumes that the owner of the minidisk is the user ID of the virtual machine in which it is running.

NULL
On CREATE statements, this is equivalent to not specifying a minidisk owner for a file. On ALTER statements, removes any previous minidisk owner specification for the file.

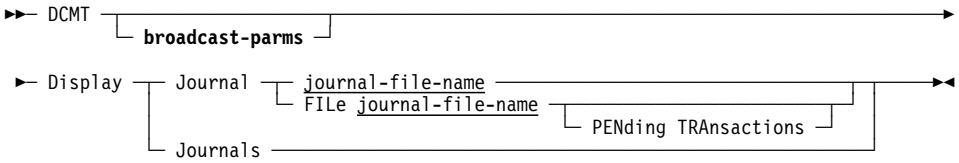
Usage

Dynamic Allocation of Journal Files: Dynamic allocation of files is operating system and file-type dependent. For information about allocating files dynamically, see FILE Statements in the *CA IDMS Database Administration Guide*

3.2.3 DCMT DISPLAY JOURNAL Command

The DCMT DISPLAY JOURNAL command displays information about the journals. It has been enhanced to display information about a specific journal file.

Syntax



Parameters

FILe
Displays information about a disk journal.

journal-file-name
Specifies the name of the disk journal.

PENding TRAnsactions
Outputs information about pending transactions.

Usage

Displaying Pending Transactions of a Disk Journal File: A pending transaction is a transaction that is active and might need the named disk journal if the transaction has to be backed out. Pending transactions prevent the disk journal from reaching an offline status.

Example

DCMT DISPLAY JOURNAL FILE journal-file-name PENDING TRANSACTIONS

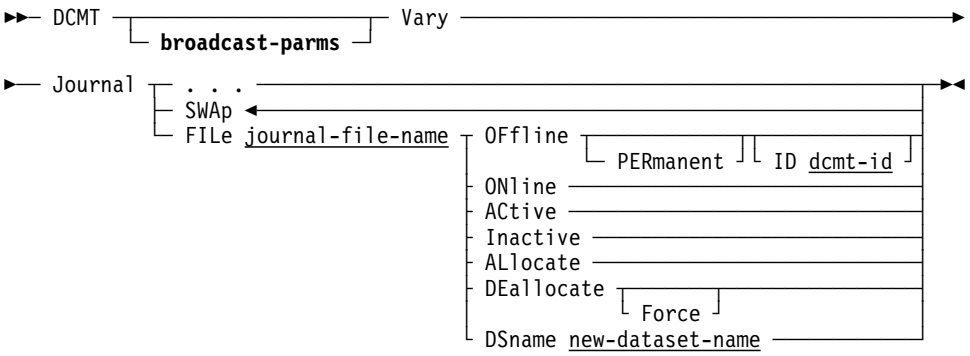
```
 DCMT DISPLAY JOURNAL FILE SYSJRNL2 PENDING TRANSACTION
Task / LTE   Trans-ID Pri Orig Module   SS/AM    St Stat        Date:Time
        57        165 100  LOC LOCKTEST EMPSS01  RW   H 2007-03-16-08.52.45.6079
```

3.2.4 DCMT VARY JOURNAL Command

The DCMT VARY JOURNAL command varies journaling-related options and switches the active journal from one file to another. It has been enhanced to enable changing the status and attributes of an individual journal file.

- Switch the active disk journal from one file to another
- Disable or enable use of a disk journal file
- Change the data set name or disposition of a disk journal file
- Allocate or deallocate a disk journal file
- Change the values assigned to the journal fragment interval
- Assign a value to the journal transaction level

Syntax



Parameters

SWAp
Directs CA IDMS to switch the active journal file from one file to another.

FILe journal-file-name
Specifies the name of the disk journal to be varied.

OFfline
Makes the specified disk journal file inaccessible to the system.

PERmanent
Specifies that the OFFLINE status of the journal file is permanent. The status remains in effect until it is changed by another DCMT VARY command or the SYSTRK files are formatted.

The ability to record a status as permanent requires that change tracking be active. If change tracking is not active, the OFFLINE status is not made permanent and a warning message is issued.

dcmt-id
Specifies the identifier that is assigned to this vary operation. It must be a 1- to 8-alphanumeric character string that is unique across all outstanding DCMT operations originating on this DC/UCF system.

If no *dcmt-id* is specified, the vary operation is assigned an internally generated identifier.

The identifier can subsequently be used to monitor or terminate the vary using DCMT DISPLAY ID and DCMT VARY ID commands.

ONline
Makes the specified disk journal file accessible to the system.

ACtive
Enables access to the journal file and sets its status to something other than 9999.

Varying the file active allows suspended transactions that are waiting on the journal file to resume execution.

Inactive
Disables access to the journal file and sets its status to 9999. No new journal images are written to the file, but it can still be read for recovery purposes.

The ability to vary journal files inactive is provided to simulate I/O errors for the purpose of testing recovery procedures.

ALlocate
Dynamically allocates the journal file using its currently assigned data set name.

DEallocate
Dynamically closes and deallocates the named file.

Force

Directs DC/UCF to mark the file as deallocated and closed, even though these actions were not taken. This option is useful for certain types of error conditions for which a close cannot be completed.

DSname new-dataset-name

Changes the DSname temporarily in the runtime DMCL to *new-dataset-name*. If the file has previously been opened, the DDname is cleared to blanks.

To change the DSname of a disk journal, it must be offline.

Usage

Forcing a Journal SWAP: The SWAP option of the DCMT VARY JOURNAL command directs the DC/UCF system to switch the active journal file from one disk journal file to another. If only one journal file is online and usable, the contents of the journal file must be offloaded before the command can complete and journaling resume. This causes a delay in the execution of all update transactions until the swap completes.

Varying a Specific Journal File: The DCMT VARY JOURNAL FILE command is intended for solving disk journal problems such as I/O errors while DC/UCF remains active. Before issuing any DCMT VARY JOURNAL FILE command, see Recovery Procedures from Journal File I/O Errors in the *CA IDMS Database Administration Guide*.

To successfully issue a VARY JOURNAL FILE command, the target journal file must not be the active journal file. Additionally, the following restrictions apply depending on the nature of the change:

- To vary the data set name of the journal file or to allocate or deallocate the file, it must be offline or inactive or have encountered an I/O error.
- To vary a journal file whose status is permanently offline to an active or online state, requires that change tracking be active.

Varying a Disk Journal File Offline: When varying a disk journal file offline, the system quiesces use of the journal file before marking it as offline. While the journal file is quiescing, the following is true:

- No further journal images are written to the journal file.
- The journal file remains available for recovery operations until all transactions having journal images on the disk journal have been terminated.
- The journal file remains in a pending offline state until all journal images contained on the file have been offloaded by an ARCHIVE JOURNAL utility statement.

Note: Once the journal file reaches the quiesced state, it is closed.

The DCMT DISPLAY JOURNAL FILE command is used to determine which transactions may have journal images on the target file.

Dynamically Allocating and Deallocating Journal Files: The ability to dynamically allocate and deallocate journal files is operating system and file-type dependent. The restrictions are the same as those for database files.

Note: For more information about allocating and deallocating files, see DCMT VARY FILE Command in the *CA IDMS System Tasks and Operator Commands Guide*.

Example

DCMT VARY JOURNAL FILE journal-file-name OFFLINE

```
 DCMT VARY JOURNAL FILE SYSJRNL2 OFFLINE
Journal SYSJRNL2                    OFFLINE
Disk Journal   Segno  LoRBN  HiRBN  NxRBN Ful Act Rcv Arc Stat  DsRBN DsINTV Tql
SYSJRNL2       OFFLINE
------ Journal File ------- MODE Stat Pg-Size  Fl-Type M-Cache S-Cache DD-Name
SYSJRNL2                    Clos    0   2932  non-VSAM    No      No   SYSJRNL2
                                                                      VOLSER: CULL06
  DSname: (JCL)... DBDC.SYSTEM73.SYSJRNL2                             DISP=SHR
```

3.3 Scratch Enhancements

The CA IDMS scratch area is a virtual database whose content disappears when a job step ends. To eliminate all I/O to the associated file, the scratch area can be maintained in memory. Scratch enhancements for r17 include the following:

- Scratch above the bar allows direct memory access to 64-bit storage and allows the size of the scratch area to be significantly larger when it is memory resident.
- Extensible scratch allows you to dynamically extend the scratch area.

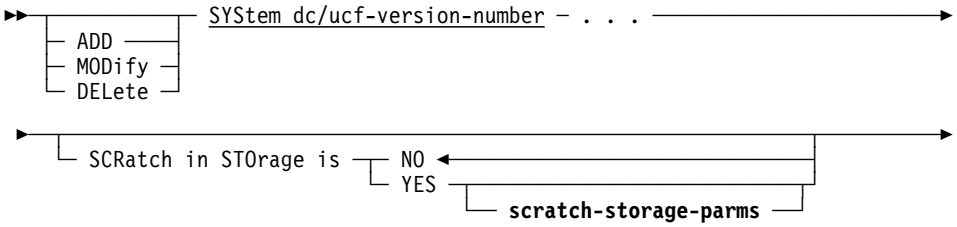
You can control scratch in memory using the following methods:

- For CV, through new parameters on the system generation SYSTEM statement or through a new DCMT VARY SCRATCH command.
- In batch local mode, through new SYSIDMS parameters.

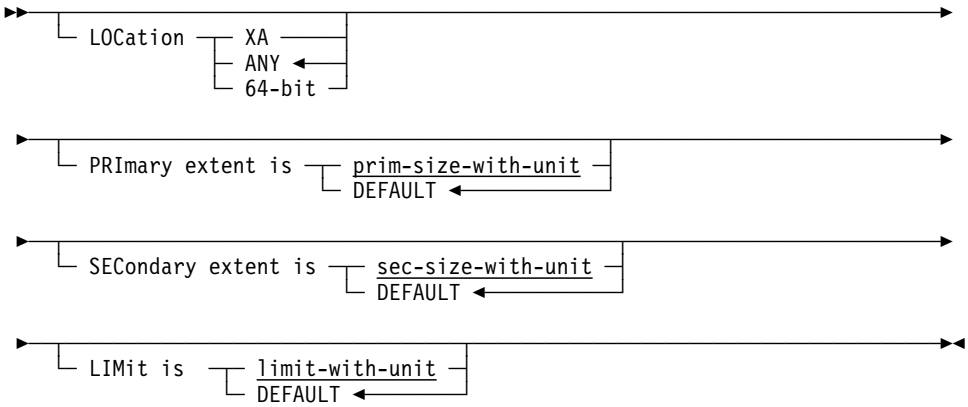
3.3.1 System Generation SYSTEM Statement

Use the system generation SYSTEM statement to control the scratch area.

Syntax



Expansion of scratch-storage-parms



Parameters

SCRatch in STOrage is

Specifies whether scratch information resides in memory.

NO

Specifies that the scratch information is not memory-resident.

YES

Specifies that the scratch information is memory-resident.

LOCation

Controls where memory for the scratch information is allocated with the following options:

ANY|XA|64-bit

Determines the storage location. The storage needed for scratch processing is allocated directly from the operating system and not from the CA IDMS storage pools.

ANY

Acquires 64-bit storage, if possible. If the request to allocate 64-bit storage fails, XA storage is acquired.

XA

Acquires 31-bit storage.

64-bit

Acquires 64-bit storage. If the request to allocate 64-bit storage fails, no attempt to acquire XA storage is done.

Notes:

- SCRATCH IN XA STORAGE IS YES is synonymous to SCRATCH IN STORAGE IS YES LOCATION XA.
- Usage of 64-bit storage is controlled by the MEMLIMIT parameter of the JOB or EXEC JCL statement.

PRImary extent is

Specifies the primary scratch allocation size.

prim-size-with-unit

Specifies the size of the initial storage area acquired for scratch use. Enter a number in the range 1-32767 followed by a unit of KB (Kilobyte: 2**10), MB (Megabyte: 2**20), GB (Gigabyte: 2**30), TB (Terabyte: 2**40), or PB (Petabyte: 2**50).

DEFAULT

Specifies the system default value. If the DMCL contains a scratch area definition, the default value is the number of pages in the area multiplied by the page size. If no scratch area is defined in the DMCL, the system default value is 1 MB.

SECondary extent is

Specifies the secondary scratch allocation size.

sec-size-with-unit

Specifies the amount of additional storage acquired when all existing scratch storage is in use. Enter a number in the range 1-32767 followed by a unit of KB (Kilobyte: 2**10), MB (Megabyte: 2**20), GB (Gigabyte: 2**30), TB (Terabyte: 2**40), or PB (Petabyte: 2**50).

DEFAULT

Specifies the system default value. The size of the secondary allocation is equal to the size of the primary allocation.

LIMit is

Specifies the maximum scratch allocation size.

limit-with-unit

Specifies the maximum amount of scratch storage. The system continues to allocate more storage for scratch processing until the sum of all allocations reaches the value specified by *limit-with-unit*. Enter a number in the range 1-32767 followed by a unit of KB (Kilobyte: 2**10), MB (Megabyte: 2**20), GB (Gigabyte: 2**30), TB (Terabyte: 2**40), or PB (Petabyte: 2**50).

DEFAULT

Specifies the system default value. If the DMCL contains a scratch area definition, the default value is the number of pages in the area multiplied by the page size. If no scratch area is defined in the DMCL, the system default value is the size of the primary allocation plus 99 times the size of the secondary allocation.

More Information

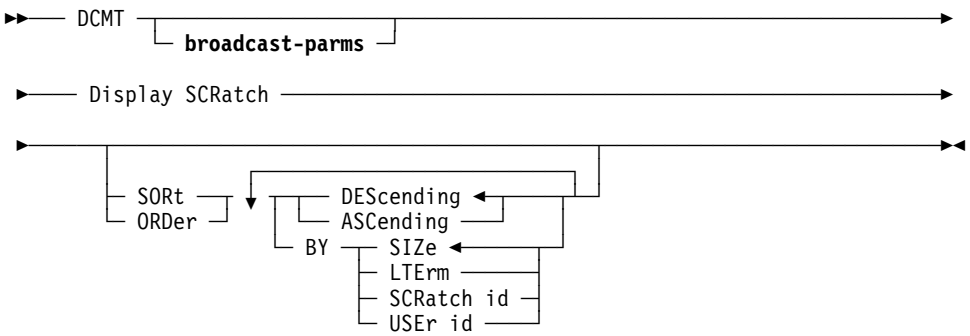
For more information about the System Generation SYSTEM statement, see the *CA IDMS System Generation Guide*.

3.3.2 DCMT DISPLAY SCRATCH Command

A new DCMT DISPLAY SCRATCH command has been added to obtain the following information about scratch usage:

- Definition-related information, such as number of pages, page size, and location
- Global statistics and high-water marks
- Detailed information

Syntax



Parameters

broadcast-parms
Indicates to execute the DCMT command on all or a list of data sharing group members.

Note: For more information about broadcasting and **broadcast-parms**, see How to Broadcast System Tasks in the *CA IDMS System Tasks and Operator Commands Guide*.

Display SCRatch
Displays global statistics, definition-related, and detailed information about scratch.

SORt or ORDer
Requests sorted output.

DEScending
Specifies to display the higher values first in the sorted output. This is the default.

ASCending
Specifies to display the lower values first in the sorted output.

BY
Identifies the sort criterion.

SIZe
Specifies to sort by the scratch area size. This is the default.

LTErm
Specifies to sort by the logical terminal name.

SCRatch id
Specifies to sort by the scratch area ID.

USEr id
Specifies to sort by the user ID.

Usage

Obtaining Output with SORT BY: SORT BY allows you to obtain the output that is most relevant for a given situation as shown in the following examples:

- SORT DESCENDING BY SIZE—Identifies the largest scratch users.
- SORT BY USER ID—Identifies how much scratch a user owns.
- SORT BY SCRATCH ID—Identifies an application program's usage based on scratch area ID.

Example

DCMT DISPLAY SCRATCH SORT DESCENDING BY SIZE

```
DCMT DISPLAY SCRATCH SORT DESCENDING BY SIZE
Total   number of pages         391     Location                    ANY Storage
Page size                     2676     Storage address      00000001 00900000
Primary   extent #pages         391     Primary   extent size             1 MB
Secondary extent #pages         783     Secondary extent size             2 MB
Storage   limit  #pages       11363     Storage   limit  size            29 MB
PUT    scratch requests         223     Scratch Areas active                9
GET    scratch requests          91     Scratch Areas created              12
DELETE scratch requests          44     HWM concurrent Scr. Areas           9
Pages in use                    240     HWM pages in use                  241
Pages in use percentage          61%    HWM pages in use  percentage        61%
Buffers                         N/A     HWM pages in use  for 1 S.A.       141
Pages found in buffer           N/A     HWM pages found   for 1 S.A.       N/A
Pages written                   N/A     HWM pages written for 1 S.A.       N/A
Pages read                      N/A     HWM pages read    for 1 S.A.       N/A

Scratch Area ID     Size: Pages / %     LTERM      User id
OCF*FSEO                   140    35   VL72001    USER02
DDDLFSEI                    83    21   VL72002    USER01
SSCHFSEO                     6     1   VL72001    USER02
OCF*FSEI                     3    <1   VL72001    USER02
SSCHFSEI                     3    <1   VL72001    USER02
DDDLFSEO                     2    <1   VL72002    USER01
DDDLFSEC                     1    <1   VL72002    USER01
OCF*FSEC                     1    <1   VL72001    USER02
SSCHFSEC                     1    <1   VL72001    USER02
```

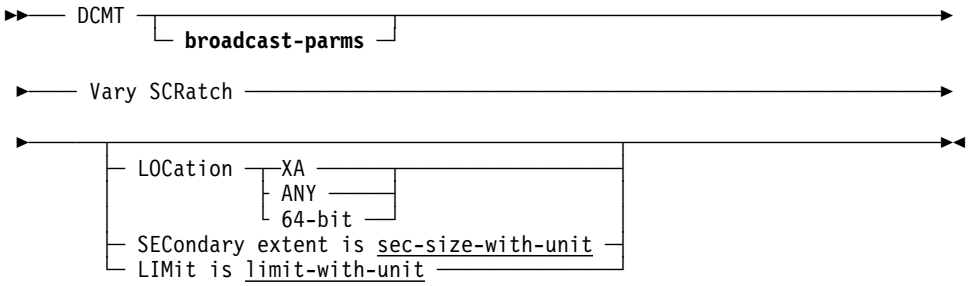
More Information

For more information about DCMT commands, see the *CA IDMS System Tasks and Operator Commands Guide*.

3.3.3 DCMT VARY SCRATCH Command

A new DCMT VARY SCRATCH command has been added to dynamically alter the scratch area control parameters.

Syntax



Parameters

broadcast-parms
Indicates to execute the DCMT command on all or a list of data sharing group members.

Note: For more information about broadcasting and **broadcast-parms**, see How to Broadcast System Tasks in the *CA IDMS System Tasks and Operator Commands Guide*.

Vary SCRatch
Specifies the size of the secondary allocation, maximum amount of storage, and storage location.

LOCation
Specifies where memory for the scratch information is allocated with the following options:

ANY|XA|64-bit
Determines the storage location. The storage needed for scratch processing is allocated directly from the operating system and not from the CA IDMS storage pools.

ANY
Acquires 64-bit storage if possible. If the request to allocate 64-bit storage fails, XA storage is acquired.

XA
Acquires 31-bit storage.

64-bit
Acquires 64-bit storage. If the request to allocate 64-bit storage fails, no attempt to acquire XA storage is done.

SECondary extent is

Specifies the secondary scratch allocation size.

sec-size-with-unit

Specifies the amount of additional storage acquired when all existing scratch storage is in use. Enter a number in the range 1-32767 followed by a unit of KB (Kilobyte: 2**10), MB (Megabyte: 2**20), GB (Gigabyte: 2**30), TB (Terabyte: 2**40), or PB (Petabyte: 2**50).

LIMit is

Specifies the maximum scratch allocation size.

limit-with-unit

Specifies the maximum amount of scratch storage. The system continues to allocate more storage for scratch processing until the sum of all allocations reaches the value specified by *limit-with-unit*. Enter a number in the range 1-32767 followed by a unit of KB (Kilobyte: 2**10), MB (Megabyte: 2**20), GB (Gigabyte: 2**30), TB (Terabyte: 2**40), or PB (Petabyte: 2**50).

Usage

Changing Scratch Parameters: The following information should be taken into consideration when changing scratch parameters:

- A change in scratch location can be done only if scratch is in storage.
- A change in scratch location only affects the location of future secondary allocations. Current allocations are not relocated.
- Decreased values for *sec-size-with-unit* and *limit-with-unit* are honored at the time a secondary extent becomes empty.

  Example: *prim-size-with-unit*=10 MB; *sec-size-with-unit*=5 MB; *limit-with-unit*=50 MB; three secondary extents are allocated (25 MB of storage is in use). DCMT VARY SCRATCH LIMIT 20 MB is issued. A secondary allocation is freed only when it becomes entirely unused.

Examples

DCMT VARY SCRATCH SECONDARY EXTENT 1 MB

```
      V SCR SECONDARY EXTENT 1 MB
IDMS DC293001 V71 USER:JSMITH  Scratch Secondary extent changed to 1 MB
```

DCMT VARY SCRATCH LIMIT 10 MB

```
      V SCR LIMIT 10 MB
IDMS DC293001 V71 USER:JSMITH  Scratch Limit changed to 10 MB
```

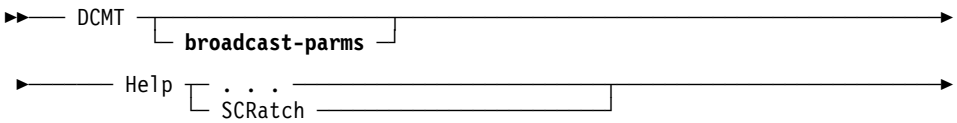
More Information

For more information about DCMT commands, see the *CA IDMS System Tasks and Operator Commands Guide*.

3.3.4 DCMT Help Command

The DCMT HELP command has been enhanced to display a help screen of the DCMT DISPLAY SCRATCH and DCMT VARY SCRATCH syntax.

Syntax



Parameters

Help
Displays syntax for the HELP command.

SCRatch
Displays the scratch help screen.

More Information

For more information about the DCMT HELP command, see the *CA IDMS System Tasks and Operator Commands Guide*.

3.3.5 SYSIDMS Parameters

Use the following SYSIDMS parameters to control the scratch processing:

SCRATCH_IN_STORAGE=ON|OFF|ANY|XA|64-bit

Enables storage allocation from the operating system for scratch processing.

ON

Specifies the same as SCRATCH_IN_STORAGE=ANY.

OFF

Specifies to allocate the scratch area as defined in the DMCL.

ANY

Acquires 64-bit storage if possible. If the request to allocate 64-bit storage fails, XA storage is acquired.

XA

Acquires 31-bit storage.

64-bit

Acquires 64-bit storage. If the request to allocate 64-bit storage fails, no attempt to acquire XA storage is done.

Default: OFF

Note: Usage of 64-bit storage is controlled by the MEMLIMIT parameter of the JOB or EXEC JCL statement.

SCRATCH_LIMIT=*limit-with-unit*

Specifies the maximum amount of scratch storage. The system continues to allocate more storage for scratch processing until the sum of all allocations reaches the value specified by *limit-with-unit*. Enter a number in the range 1-32767 followed by a unit of KB (Kilobyte: 2**10), MB (Megabyte: 2**20), GB (Gigabyte: 2**30), TB (Terabyte: 2**40), or PB (Petabyte: 2**50).

The default value is determined as follows:

- If the DMCL contains a scratch area definition, the default is the number of pages in the area multiplied by the page size.
- If no scratch area is defined in the DMCL, the default is the size of the primary allocation plus 99 times the size of the secondary allocation.

SCRATCH_PRIMARY_EXTENT=*prim-size-with-unit*

Specifies the primary scratch allocation size. Enter a number in the range 1-32767 followed by a unit of KB (Kilobyte: 2**10), MB (Megabyte: 2**20), GB (Gigabyte: 2**30), TB (Terabyte: 2**40), or PB (Petabyte: 2**50).

The default value is determined as follows:

- If the DMCL contains a scratch area definition, the default value is the number of pages in the area multiplied by the page size.

- If no scratch area is defined in the DMCL, the system default value is 1 MB.

SCRATCH_SECONDARY_EXTENT=*sec-size-with-unit*

Specifies the amount of storage to allocate when all existing scratch storage is in use. Enter a number in the range 1-32767 followed by a unit of KB (Kilobyte: 2**10), MB (Megabyte: 2**20), GB (Gigabyte: 2**30), TB (Terabyte: 2**40), or PB (Petabyte: 2**50).

The default size of the secondary allocation is equal to the size of the primary allocation.

More Information

For more information about SYSIDMS parameters, see the *CA IDMS Common Facilities Guide*.

Chapter 4. Performance

This chapter describes the performance enhancements. It contains the following topics:

4.1 CICS Threadsafe Support

CA IDMS is enhanced with CICS threadsafe support that allows threadsafe application programs to use multiple open TCBs while accessing CA IDMS.

4.1.1 Threadsafe Concepts

CICS Transaction Server for z/0S (CTS) provides a method for multiple CTS transactions to run simultaneously on separate TCBs. Application programs that are eligible to run in this mode are described as threadsafe. For more information about the CTS threadsafe operation, see the appropriate IBM documentation. A brief overview of this IBM feature as it relates to the CA IDMS interface is described in the next section.

Historically, all CICS application programs ran on the same TCB, which allowed only one program task to execute at any given instant. While multiple tasks could be active, only one task could execute instructions on a CPU. Under CTS, IBM has introduced the concept of threadsafe application programs that can be run on open TCBs, thus allowing multiple programs to execute simultaneously on different CPUs.

A program that is declared with the CONCURRENCY(THREADSAFE) attribute is considered to be eligible to run on an open TCB, but this attribute alone is not enough to cause the program to do so. Various conditions exist that cause a threadsafe program to execute on an open TCB. Three of the most common cases are the following:

- Define an application program with the API(OPENAPI) attribute. This attribute is only available in CTS Version 3.1 and later.
- Invoke a Task Related User Exit (TRUE) that has been enabled with the API(OPENAPI) attribute.
- Access a DB2 database using DB2 Version 6 or later. This is a special instance of the previous case because DB2 executes as a TRUE exit.

When a task begins to run on an open TCB, it continues to run until one of the following occurs:

- A non-threadsafe command is executed
- An EXEC CICS RETURN is made to a non-threadsafe program
- A particular point is reached during CICS task termination processing

A threadsafe command is one which can be executed on an open TCB. A program defined as threadsafe can issue a non-threadsafe command. However, issuing a non-threadsafe command causes the task to be switched to run on the QR (single-threaded) TCB. This can cause performance degradation, particularly if a lot of TCB switching is done.

If a threadsafe program defined with API(CICSAPI) is switched to the QR TCB, it stays there unless another OPENAPI TRUE exit is invoked. If a threadsafe program defined with API(OPENAPI) is switched to the QR TCB, it switches back to the open TCB when control is returned to the application program after execution of the non-threadsafe command.

4.1.2 CA IDMS Support for Threadsafe Applications

The CA IDMS interface modules that run in a CTS region have been enhanced to be threadsafe. Threadsafe application programs, that is, programs defined with the CONCURRENCY(THREADSAFE) attribute can use this enhancement to obtain increased throughput.

An application program that is running on an open TCB can access CA IDMS without switching to the single-threaded QR TCB. A new option, TRUEAPI, is provided to allow the first CA IDMS access by a task to force a switch to an open TCB. If the interface has been called by a program defined as threadsafe, the program continues to run on the open TCB after return from the CA IDMS call. For more information about the TRUEAPI option, see 4.1.3, “IDMSINTC Interface Considerations” on page 88.

CTS has rules and guidelines on whether an application program can or should be defined with CONCURRENCY(THREADSAFE) or API(OPENAPI) attributes or both. Before defining your own programs as THREADSAFE or OPENAPI, be sure to consult the appropriate IBM documentation.

The use of CA IDMS with an otherwise threadsafe program does not cause integrity problems and can provide significant performance improvement. Depending on the nature of the application, however, it may not improve performance and could conceivably cause performance degradation. In addition, if a client-written application program is declared to be threadsafe and the program itself violates the rules for threadsafe programs, the results are unpredictable.

Note: A few cases exist where the CA IDMS interface issues CICS commands that force a switch to the QR TCB. For more information, see 4.1.3, “IDMSINTC Interface Considerations” on page 88.

4.1.3 IDMSINTC Interface Considerations

You can enter the IDMSINTC interface program using one of the following methods:

- Through the PLT or by invocation of a transaction that starts the IDMSINTC interface. By invoking IDMSINTC in this way, it is not threadsafe, so the program cannot be defined as THREADSAFE. See the sample definition of PROGRAM(IDMSINTC) in member CICSCSD in the installed CA IDMS source library.
- Through a branch entry from the application program using the IDMSCINT stub program. This entry functions as an extension of the calling program with the same program attributes. This includes the THREADSAFE and OPENAPI attributes.

Except for a few cases, discussed in Non-threadsafe Instructions, the IDMSINTC interface does not issue any non-threadsafe commands. Therefore, if IDMSINTC is entered on an open TCB, it stays on the open TCB throughout its execution and return to the application program.

When a task makes its first CA IDMS call, IDMSINTC invokes the CA IDMS TRUE exit. This exit is always enabled with the THREADSAFE attribute. The TRUEAPI=OPEN parameter is provided on the CICSOPT macro that causes the exit to also be enabled with the OPENAPI attribute. For more information about the TRUEAPI parameter, see 4.1.9, “New CICSOPT Parameters” on page 93.

If TRUEAPI=OPEN is specified, the first CA IDMS call in each task causes a switch to an open TCB. If the application program is defined as threadsafe, the interface continues to execute on that open TCB through its return to the application program.

Non-threadsafe Instructions

A few cases exist where invocation of the CA IDMS interface from an application program causes a non-threadsafe instruction to be issued. If the application program is defined as THREADSAFE, but not OPENAPI, the interface continues to execute on the QR TCB through return to the application program. If the application program is defined as OPENAPI, CICS switches to the QR TCB during execution of the non-threadsafe instruction and back to the open TCB after completion of the instruction. The interface continues to run on the open TCB through return to the application program.

The following cases can cause a non-threadsafe instruction to be issued:

- If the CICSOPT macro specifies a value other than IMMEDIATE on the TIMEOUT parameter, an EXEC CICS START TRANSACTION is issued at task termination. This is not an important performance consideration

because there will be no return to an application program, and CICS always switches to the QR TCB at some point during task termination.

- If a ROLLBACK command is issued when AUTOCMT=ON is in effect, an EXEC CICS SYNCPOINT ROLLBACK is issued.
- The EXEC CICS WRITEQ TD command can produce certain error conditions, resulting in a message to be written. These conditions are rare in a production system.
- If DEBUG=ON is specified in the CICSOPT macro or IDMSDBUG=ON is specified as a SYSIDMS runtime parameter, various information is written using the EXEC CICS WRITEQ TD command. These options are usually used only in special situations when CA Technical Support personnel need diagnostic information to resolve a problem. For more information about using an alternative parameter value of DEBUG=QTS, see 4.1.9, “New CICSOPT Parameters” on page 93.

4.1.4 UCF Front-end (#UCFCICS) Considerations

The UCF Front-end program does not violate any threadsafe rules and can be declared as a THREADSAFE or OPENAPI program. It does, however, issue various non-threadsafe commands, such as terminal I/O commands. Therefore, for best performance, it should be defined with CONCURRENCY(QUASIRENTRANT).

4.1.5 Distributed Processing with #UDASCIC Considerations

The distributed processing program created with the #UDASCIC macro rarely issues a non-threadsafe command and is a good candidate to declare as a THREADSAFE or OPENAPI program. However, this program does issue a non-threadsafe command when a 1473 Error-Status is received from the CA IDMS interface because of a MAXERUS condition on the CA IDMS Central Version. In this case, the program waits by continuing to issue EXEC CICS DELAY INTERVAL(1) commands until the condition is alleviated or 100 attempts have been made.

4.1.6 CICS Abort Session Program Considerations

The CICS Abort Session Program is created by compiling the #UCFCICZ program and can be declared as a THREADSAFE and OPENAPI program.

The #UCFCICZ macro generates some code that is not compliant with the recommended usage with CICS Transaction Server. This code can cause problems with applications that are associated with a bridge facility. To provide compatibility with previous methods of calling #UCFCICZ from CICS error programs, two new parameters are added to the #UCFCICZ macro: PASSVAL and BRIDGE. To prevent the #UCFCICZ macro from issuing non-threadsafe commands, we recommend that you compile it with the PASSVAL=TERMID parameter.

Note: For more information, see 4.1.7, “CICS Abort Session Program” on page 90.

4.1.7 CICS Abort Session Program

The #UCFCICZ macro is enhanced to provide compatibility with previous methods of calling #UCFCICZ from CICS error programs with two new parameters: PASSVAL and BRIDGE.

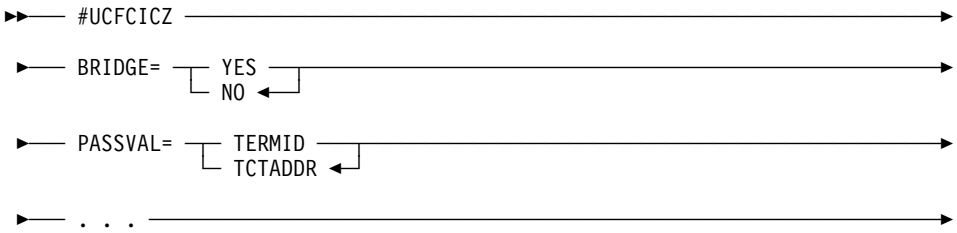
The #UCFCICZ macro can be assembled to create an abort program to request UCF to abort the session for any terminal that disconnects or goes out of service. You can call the abort program by any combination of the following methods:

- The CICS terminal error program DFHTEP
- The node error program DFHZNEP
- The bridge facility global exit XFAINTU

By using #UCFCICZ, you can assure the timely release of back-end resources when a front-end abort occurs. You can also prevent the following scenario from occurring:

A user signs onto CICS through a bridge facility or onto a VTAM terminal through a multisession manager. During a terminal-read request from UCF, the user loses the connection or terminates the CICS session from the multisession manager. A second user simultaneously connects and is assigned to the same terminal identifier. The second user invokes the UCF front-end program and is placed in the middle of the session started by the first user.

Syntax



Parameters

BRIDGE

Specifies whether the module generated by this #UCFCICZ macro will be called from a program invoked by the bridge facility exit point XFAINTU. If NO is specified, the UCF abort session program assumes that the aborted session is associated with a permanent terminal. If UCFCICS had modified the UCTRANST value associated with the terminal, the UCF abort session program attempts to restore the original UCTRANST value. Therefore, this parameter has no effect if the associated UCFCICS macro specifies UCTRAN=TCT. The default is NO.

PASSVAL=TCTADDR/TERMID

Specifies the format and value of the COMMAREA parameter that is passed to the UCF abort session program. PASSVAL=TCTADDR indicates that the COMMAREA contains a fullword address pointing to the Terminal Control table. PASSVAL=TERMID indicates that the COMMAREA contains the 4-byte identifier of the terminal or bridge facility associated with the aborted session.

How to Use the UCF CICS Abort Session Program: One or two UCF CICS abort sessions are needed for each UCFCICS program created with a #UCFUFT macro that specifies the corresponding NTID. One program is needed for persistent terminals. A separate one may be needed for sessions associated with a bridge facility. A single program can be used if both the following conditions are true:

- All callers pass the same format COMMAREA to the UCF CICS abort session program as defined by the PASSVAL parameter. PASSVAL=TERMID is recommended.

  Note: The default is PASSVAL=TCTADDR for compatibility with previous site-created versions of DFHZNEP or DFHTEP.

- The associated #UCFCICS macro specifies UCTRAN=TCT.

For each UCF CICS abort session program you create, perform the following steps:

1. Assemble the #UCFCICZ macro with the appropriate parameters and link the resulting program with a unique name
2. Add an entry to the CICS CSD for each session abort program as follows:

```
DEFINE   PROGRAM(ucfcicz)
         GROUP(groupnam) LANGUAGE(ASSEMBLER) CEDF(NO)
         EXECKEY(CICS)
```

3. Modify DFHTEP, DFHZNEP, and XFAINTU to call the appropriate versions of the program

Modify DFHTEP/DFHZNEP/XFAINTU to link to UCFCICZ: Modify the error programs or bridge facility tidy up program or both to link to the appropriate UCF session abort programs.

Note: For more information about DFHTEP, DFHZNEP, and XFAINTU, refer to the CICS system documentation.

The following examples illustrate one approach to the modification of error and tidy up programs.

For DFHTEP and DFHZNEP, insert the instructions immediately before the DFHTEP/DFHZNEP exit. The logic states that if the error action codes indicate that the application task (if any) is to abend, a link is made to two UCF CICS session abort programs.

DFHTEP instructions when PASSVAL=TERMID: The following statements add instructions to DFHTEP when the #UCFCICZ macro specifies PASSVAL=TERMID:

```
            TM    TCTLEECB+1,X'04'           ABEND TASK?
            BZ    NOCICZ                     NO
            LA    10,TCTLEPTE                POINTER TO TCTTE
            L     10,0(,10)                  TCTTETI
            EXEC  CICS LINK PROGRAM('UCFCICZ1')
                           COMMAREA( 0(10) )
                           LENGTH( 4).
            EXEC  CICS LINK PROGRAM('UCFCICZ2')
                           COMMAREA( 0(10) )
                           LENGTH( 4).
    NOCICZ  DS    0H
```

DFHTEP instructions when PASSVAL=TCTADDR: The following statements add instructions to DFHTEP when the #UCFCICZ macro specifies PASSVAL=TCTADDR. This method is provided for compatibility with earlier versions. PASSVAL=TERMID is recommended.

```
            TM    TCTLEECB+1,X'04'           ABEND TASK?
            BZ    NOCICZ                     NO
            LA    10,TCTLEPTE                POINTER TO TCTTE
            EXEC  CICS LINK PROGRAM('UCFCICZ1')
                           COMMAREA( 0(10) )
                           LENGTH( 4).
            EXEC  CICS LINK PROGRAM('UCFCICZ2')
                           COMMAREA( 0(10) )
                           LENGTH( 4).
    NOCICZ  DS    0H
```

DFHZNEP instructions when PASSVAL=TERMID: The following statements add instructions to DFHZNEP when the #UCFCICZ macro specifies PASSVAL=TERMID:

```
            TM    TWAROPT2,TWAOAT            ABEND TASK?
            BZ    NOCICZ                     NO
            L     7,TWATCTA
            EXEC  CICS LINK PROGRAM('UCFCICZ1')
                           COMMAREA(0(7))
                           LENGTH( 4).
*
            EXEC  CICS LINK PROGRAM('UCFCICZ2')
                           COMMAREA(0(7))
                           LENGTH( 4).
    NOCICZ  DS    0H
```

DFHZNEP instructions when PASSVAL=TCTADDR: The following statements add instructions to DFHZNEP when the #UCFCICZ macro specifies PASSVAL=TCTADDR. This method is provided for compatibility with earlier versions. PASSVAL=TERMID is recommended.

```
           TM    TWAROPT2,TWAOAT             ABEND TASK?
           BZ    NOCICZ                      NO
           EXEC  CICS LINK PROGRAM('UCFCICZ1')
                          COMMAREA(TWATCTA)
                          LENGTH( 4).
*
           EXEC  CICS LINK PROGRAM('UCFCICZ2')
                          COMMAREA(TWATCTA)
                          LENGTH( 4).
    NOCICZ  DS    0H
```

XFAINTU instructions when UEPFAREQ=UEPFATU: The following statements illustrate how to modify XFAINTU. The code should be executed only if UEPFAREQ contains the value UEPFATU on entry to XFAINTU.

```
           EXEC  CICS LINK PROGRAM('UCFCICZ1')
                          COMMAREA(UEPFANAM)
                          LENGTH( 4).
*
           EXEC  CICS LINK PROGRAM('UCFCICZ2')
                          COMMAREA(UEPFANAM)
                          LENGTH( 4).
```

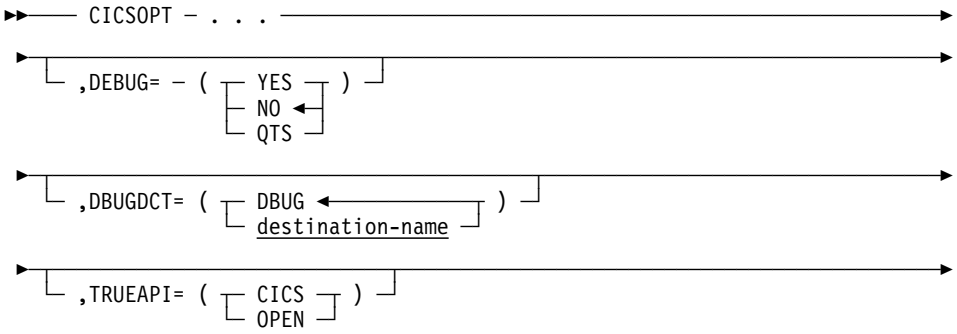
4.1.8 IDMSRSYN Resynchronization Program Considerations

IDMSRSYN is threadsafe, but because it issues non-threadsafe commands, it should not be defined as OPENAPI.

4.1.9 New CICSOPT Parameters

This section describes the new and enhanced CICSOPT parameters.

Syntax



Parameters

DEBUG

Specifies whether IDMSINTC produces extra debugging information about internal processing.

YES

Specifies that IDMSINTC produces extra information. This information is written using WRITEQ TD to the destination specified on the DBUGDCT parameter.

NO

Specifies that IDMSINTC does not produce debugging information. This is the default. You should always use DEBUG=NO unless otherwise requested by CA Technical Support to resolve a system problem.

QTS

Specifies that IDMSINTC produces extra information. This information is written using WRITEQ TS to the queue specified on the DBUGDCT parameter.

DBUGDCT

Identifies the CICS transient data or temporary storage destination to use as the target for error messages produced if DEBUG=YES or DEBUG=QTS is specified.

destination-name

The default *destination-name* is DBUG. Use another destination if you want to route diagnostic messages to another CICS destination. If DEBUG=YES is specified, the DCT entry should be defined with a variable length record of at least 136 characters. We recommend that you use the values provided in source library member CICSCSD2.

TRUEAPI

Specifies whether to enable the IDMS TRUE exit with the OPENAPI attribute.

CICS

Specifies to enable the exit with the THREADSAFE attribute.

OPEN

Specifies to enable the exit with the THREADSAFE and OPENAPI attributes.

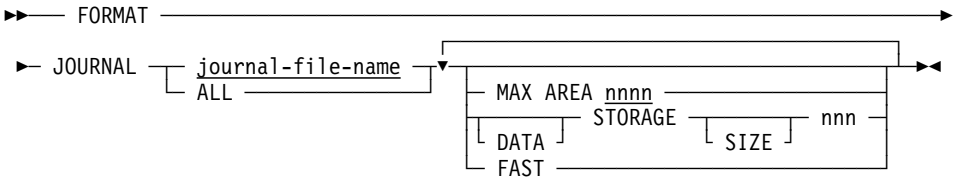
More Information

For more information about the CICSOPT macro, see IDMSINTC in the "TP-Monitor Considerations" chapter in the *CA IDMS System Operations Guide*.

4.2 Fast Journal Format Option

A new FAST parameter on the FORMAT JOURNAL utility statement provides a method of quickly reformatting already existing and formatted journal files rather than completely reinitializing entire journal files.

Syntax



Parameters

FAST

Formats only the journal header blocks of already existing and formatted journal files. If MAX AREA is specified, the number of JHDA entries are recalculated and the number formatted may change. If the STORAGE clause is specified, the number of JHD2 entries are recalculated and the number formatted may change.

More Information

For more information about the FORMAT utility statement, see the *CA IDMS Utilities Guide*.

4.3 LE System Mode Support

CA IDMS is enhanced to allow database procedures, SQL-invoked routines, and TCP/IP generic listener programs to execute in system mode if they are written in COBOL or PL/I and compiled with an LE-compliant compiler.

Note: The CA IDMS stack usage increases when the system is extended with applications that run in system mode. You might need to increase the system generation STACKSIZE parameter, depending on its current value. These applications include numbered exits, database procedures, SQL-invoked routines (defined with SYSTEM MODE), and TCP/IP generic listeners (defined with MODE IS SYSTEM).

4.3.1 Database Procedure

In prior releases, a database procedure written in COBOL or PL/I had to be invoked using a stub module as described in Coding Database Procedures in the "Writing Database Procedures" chapter in the *CA IDMS Database Administration Guide*. This requirement disappears if the database procedure is compiled with an LE-compliant COBOL or PL/I compiler. If no stub module is used and standard program linking is done, the database procedure executes in system mode, resulting in better performance.

Note: For more information about database procedures, see the *CA IDMS Database Administration Guide*.

4.3.2 SQL-invoked Routine

To execute an SQL-invoked routine in system mode, define it with the SYSTEM MODE attribute.

Note: For more information about SQL-invoked routines, see the *CA IDMS SQL Reference Guide*.

4.3.3 TCP/IP Generic Listener

To execute a TCP/IP generic listener program in system mode, define it with the SYSTEM MODE attribute.

Note: For more information about how to define a TCP/IP generic listener, see the *CA IDMS System Generation Guide*.

4.4 Reduced 24-bit Storage Usage

With this release, more IDMS structures and programs can reside in XA storage. This increases XA storage requirements, but it frees up 24-bit storage for other uses. For example, the amount of file related storage that must reside below the line has been reduced. This will benefit 24-bit storage constrained systems that access a large number of database files. The need for below the line storage is further reduced by enabling IDMSDBIO to reside in XA storage.

The IDMSINTC interface is also enhanced to allocate RCA and additional IDMS control blocks above the line.

4.5 zIIP Exploitation

CA IDMS is enhanced to exploit zIIP processors on the z9 series and above for the z/OS operating system. This feature enables offloading computing cycles to zIIPs, thereby increasing overall CPU throughput at lower operational costs. The zIIP feature is not dependent on any other CA IDMS feature, including multitasking.

z/OS software feature HBB7709 is required to use the zIIP feature.

The default mode of operation is to not use zIIP processors unless specifically requested at runtime. A new ZIIP startup parameter is available to enable or disable the use of these processors by CA IDMS. To facilitate analysis of the potential benefit, the feature can be enabled even if no zIIP processors are available. For more information, see 7.10, “New Startup Parameters” on page 267.

If this feature is enabled, CA IDMS uses Workload Manager to create a dependent enclave for each OS task capable of servicing work type IDMS (see DCMT DISPLAY SUBTASK in the *CA IDMS System Tasks and Operator Commands Guide*.) It then schedules a separate preemptable SRB into each such enclave.

The following sections discuss topics related to zIIP exploitation:

- Eligibility requirements
- zIIP-related DCMT commands and displays, and DCPROFIL system task
- Steps for evaluating the zIIP feature benefits

Note: On systems utilizing zIIP processors, CPU time in CA IDMS statistics includes time on the zIIP processor normalized to standard processor speed.

4.5.1 zIIP Eligibility

Most CA IDMS system code is eligible to run on a zIIP processor. However, user exits, database procedures, SQL-invoked routines, and application programs are not eligible to run on a zIIP processor. CA IDMS runtime processing ensures that a non-zIIP processor is selected to run non-eligible routines.

To ensure that only eligible modules are selected to be run on a zIIP processor, some load modules must be loaded from one of the following secured locations:

- An authorized load library named in the STEPLIB concatenation or in the CDMSLIB concatenation. A library is authorized by adding it to the list of APF-authorized libraries in the appropriate PROG*xx* or IEAAPF*xx* member in SYS1.PARMLIB.

- The Link Pack Area, which includes the following modules:
  - Dynamic LPA modules, as specified in PROG*xx* members in SYS1.PARMLIB
  - Fixed LPA (FLPA) modules, as specified in IEAFIX*xx* members
  - Modified LPA (MLPA) modules, as specified in IEALPA*xx* members
  - Pageable LPA (PLPA) modules, loaded from libraries specified in LPALST*xx* or PROG*xx* members
  - A library in the linklist, as specified in PROG*xx* and LNKLST*xx* members.

Note: For more information about authorized libraries, the LPA, and the linklist, see the IBM documentation.

The specific rules for load module residence for zIIP processing are as follows:

- The load module that is executed to start the CA IDMS CV must reside in an authorized library in the STEPLIB concatenation or in a linklist library. This module is RHDCOMVS or the startup routine. For more information about the startup routine, see Step 1: Link Edit the Startup Routine in the System Startup chapter in the *CA IDMS System Operations Guide*.
- CA IDMS nucleus modules, including all line drivers and service drivers, must be loaded from an authorized load library in the CDMSLIB concatenation or from the LPA. The IBM Language Environment library (usually CEE.SCEERUN) must be authorized if it is included in the CDMSLIB concatenation. Alternatively, the following modules must reside in the LPA: CEEPIPI, CEEPLPKA, and CEEEV003.
- z/OS Callable Services library (SYS1.CSSLIB) must be in the linklist or it must be authorized and included in the STEPLIB concatenation.
- Use of the LPA or linklist for modules supplied during the CA IDMS installation is not generally recommended. Maintenance of such modules is difficult to manage and can lead to the inadvertent use of a module with one release of CA IDMS when the module was created for a different release.

Modules that consist of non-executable code or code that is never eligible to run on a zIIP processor do not have to come from a secured location. Most modules which are supplied by a client or which are modifiable at a client site are in this category. This category includes the following:

- Client-written code, including application programs, CA ADS dialogs, table procedures, database procedures, and SQL routines
- Stand-alone load modules for DC exits WTOEXIT or WTOREXIT
- RHDCUXIT and stand-alone load modules for numbered exits
- DMCL load modules
- Database name tables

- Control blocks or tables that contain no executable code

The IDMSDBIO load module can be modified at a client site by linking a DB user exit with it. It is a nucleus module containing executable code, and it must be loaded from a secured location for zIIP eligibility regardless of whether it is modified.

Individual nucleus members in a load library do not have to be authorized and should **not** be linked with SETCODE AC(1). The startup module (RHDCOMVS or site-linked startup module) must be linked with SETCODE AC(1) if and only if the AUTHREQ parameter is specified for the CA IDMS SVC. For more information about the AUTHREQ parameter, see Generating the SVC for z/OS in the Setting Up Interpartition Communication and the SVC chapter in the *CA IDMS System Operations Guide*.

Note that not every load library in the CA IDMS startup STEPLIB and CDMSLIB needs to be authorized; only those libraries from which nucleus modules are loaded must be authorized. Appropriate startup error messages are provided to assist in this effort.

To ensure that all nucleus modules are loaded from an authorized library, it is recommended that one of the following actions be taken:

- Authorize the SMP/E target load library created during the installation of CA IDMS.
- Maintain two separate but identical SMP/E target zones except that one contains an authorized load library and the other contains a non-authorized load library.
- Manually copy all modules in the SMP/E target load library to an authorized library to be used by CV startup. Recopy all modules in this library whenever maintenance is applied.

When CA IDMS is used with a batch program, no modules are made zIIP-eligible. There are, however, considerations that arise from the use of an authorized load library. The z/OS operating system enforces certain rules for programs that are loaded from a set of authorized load libraries. In particular, any program that is linked with the RENT attribute cannot be modified at runtime. If this rule is violated, an S0C4 program check occurs. Application programs linked with the CA IDMS interface module will be modified at runtime by CA. Therefore, the batch STEPLIB concatenation should contain at least one non-authorized load library, or such user programs should be linked without the RENT attribute.

CA-supplied application programs (such as IDDSDDDL) are linked appropriately in the SMP/E target load library, so no special action is required for these programs.

4.5.2 DCMT DISPLAY SUBTASK Command

The DCMT DISPLAY SUBTASK command is enhanced to produce additional output when zIIP support is activated.

4.5.2.1 zIIP-Enabled Example Without a zIIP Processor

The following example illustrates a CA IDMS system running in multitasking mode with zIIP support enabled. The display was obtained on hardware that contained two CPs and no zIIP.

DCMT DISPLAY SUBTASK 0003

```
*** Display Subtask details ***
                          Name  SUBT0002
                        Number  03
                        Status  BUSY
                     Work type  IDMS
                 Count wakeups  80,836,576
         Count task dispatches  96,549,679
            User mode CPU time  00:00:00.0251
          System mode CPU time  00:17:10.3946
         CPU effectiveness (%)  27
       Count times fast posted  10,451,388
         Count times OS posted  00
       Count found work pass 1  96,256,979
       Count found work pass 2  292,700
 Count times POSTEXIT resumed  80,639,015
                      *** Enclave Info ***
                    zIIP time  00:00:00.0000
               zIIP on CP time  00:05:39.9737
         CPU effectiveness (%)  41
           Count swap attempts  60,356
            Count actual swaps  60,336
```

4.5.2.2 zIIP-Enabled Examples with a zIIP Processor

The following series of examples illustrate a CA IDMS system running in multitasking mode with zIIP support enabled. The displays were obtained on hardware that contained five CPs and one zIIP.

DCMT DISPLAY SUBTASK 0001

```
*** Display Subtask details ***
                          Name  MAINTASK
                        Number  01
                        Status  IDLE
                     Work type  IDMS
                  Count wakeups  1,445
          Count task dispatches  1,597
             User mode CPU time  00:00:00.0000
           System mode CPU time  00:00:01.3376
          CPU effectiveness (%)  14
        Count times fast posted  21
          Count times OS posted  00
        Count found work pass 1  1,478
        Count found work pass 2  119
 Count times POSTEXIT resumed    1,445
                       *** Enclave Info ***
                     zIIP time  00:00:00.0304
                zIIP on CP time  00:00:00.0000
          CPU effectiveness (%)  173
            Count swap attempts  3,397
             Count actual swaps  3,397
```

DCMT DISPLAY SUBTASK 0006: The following example illustrates the additional information provided for the preferred subtask:

```
*** Display Subtask details ***
                          Name  SUBT0005
                        Number  06
                        Status  BUSY
                     Work type  IDMS
                  Count wakeups  11,308,085
          Count task dispatches  30,029,342
             User mode CPU time  00:00:00.0137
           System mode CPU time  00:05:15.0039
          CPU effectiveness (%)  57
        Count times fast posted  9,261,458
          Count times OS posted  00
        Count found work pass 1  29,728,572
        Count found work pass 2  300,770
 Count times POSTEXIT resumed    11,234,399
                       *** Enclave Info ***
                     zIIP time  00:01:44.4525
                zIIP on CP time  00:00:00.1209
          CPU effectiveness (%)  113
            Count swap attempts  35,029
             Count actual swaps  35,008
```

DISPLAY SUBTASK EFFECTIVENESS: The following example illustrates whether zIIP support is active by subtask. It includes CPU statistics for each subtask and associated SRB, and percentage comparison of CPU effectiveness.

```
 DISPLAY SUBTASK EFFECTIVENESS
 *** Subtask display ***
Subtask          Elapsed time                Total CPU time          % CPU  SRB
 Name         TCB            SRB           TCB            SRB        TCB SRB
-------- -------------- -------------- -------------- -------------- --- ---
MAINTASK  00:00:08.7635  00:00:00.0182  00:00:01.3060  00:00:00.0316  14 173  Y
SUBT0001  00:00:00.0069  00:00:00.0003  00:00:00.0053  00:00:00.0002  76  66  Y
SUBT0002  00:00:00.0067  00:00:00.3827  00:00:00.0060  00:00:00.4328  89 113  Y
SUBT0003  00:00:00.0117  00:00:18.2373  00:00:00.0117  00:00:20.5911 100 112  Y
SUBT0004  00:00:00.1378  00:01:16.6358  00:00:00.0723  00:01:26.6610  52 113  Y
SUBT0005  00:00:00.2610  00:04:38.2176  00:00:00.1506  00:05:14.8743  57 113  Y
-------- -------------- -------------- -------------- -------------- --- ---
Totals    00:00:09.1876  00:06:13.4919  00:00:01.5519  00:07:02.5910  16 113
```

4.5.2.3 Usage

DCMT DISPLAY SUBTASK 000n: (z/OS systems only) Displays the following CPU statistics under Enclave Info when zIIP support is active:

Field	Value
zIIP time	The CPU time consumed while physically executing on a zIIP processor.
zIIP on CP time	The CPU time used on a CP, such as the time of scheduling the zIIP processor use and contention for a zIIP processor.
CPU effectiveness	The percentage comparison of CPU time to wall-clock time while the subtask was executing. A subtask is considered to be executing if it has not been put into a WAIT state by the CA IDMS system. An executing subtask can lose effective CPU time due to paging or to other tasks being given a higher priority by the operating system. Reported CPU effectiveness can exceed 100% due to pro-rating techniques used by the operating system to compensate for relative speed differences between the CP and zIIP.

DCMT DISPLAY SUBTASK EFFECTIVENESS: Displays whether zIIP support is active by subtask and displays the following fields for each TCB and SRB:

Field	Value
Name	The name of each subtask.
Elapsed time	The length of time the subtask or SRB has been running.
Total CPU time	The amount of CPU time the subtask or SRB has used.
CPU effectiveness	The percentage comparison of CPU time to wall-clock time while the subtask was executing. A subtask is considered to be executing if it has not been put into a WAIT state by the CA IDMS system. An executing subtask can lose effective CPU time due to paging or to other tasks being given a higher priority by the operating system. Reported CPU effectiveness can exceed 100% due to pro-rating techniques used by the operating system to compensate for relative speed differences between the CP and zIIP.

4.5.2.4 More Information

For more information about the DCMT DISPLAY SUBTASK command, see the *CA IDMS System Tasks and Operator Commands Guide*.

4.5.3 DCPROFIL System Task

The OPERATING SYSTEM display of the DCPROFIL system task is enhanced to display whether the CA IDMS system is eligible to run on a zIIP processor.

Example

```
TAPE:                volser          NUMBER OF SCTS:        0008

TOOLS TAPE:          volser

SYSTEM TRACE:          YES           OPERATING SYSTEM:      z/OS   ZIIP=N

CWA SIZE:        0000000504          DMCL TABLE:        CVDMCL

                                     PRIMARY STORAGE
SCRATCH HWM      0000000176          PROTECT KEY:             04

SIZE OF XA                           ACTIVE TRANSACTION
STORAGE AREA:    0049381376          COUNT:                 0009

QUEUE AREA
LOW PAGE:        0007999951
HIGH PAGE:       0008001950

DC VERSION ID:         0210          SVC NUMBER:            173

NUMBER OF USER                       GETMAIN SUBPOOL:       001
TRACE BUFFERS:         0250



             PAGE 00001 - NEXT PAGE:
```

More Information

For more information about the DCPROFIL system task, see the *CA IDMS System Tasks and Operator Commands Guide*.

4.5.4 Evaluating the zIIP Feature Benefits

Evaluation of the zIIP feature requires neither zIIP processors nor even hardware that is capable of supporting zIIP processors.

Several easy steps are used to determine the benefits that can be achieved by using the zIIP feature as follows:

1. Run the system with ZIIP=N using your preferred performance test stream.
2. Record the results of DCMT DISPLAY SUBTASK EFFECTIVENESS. Using a UCFBATCH program is a good method for obtaining this information.
3. Run the system with ZIIP=Y using your preferred performance test stream.
4. Record the results of DCMT DISPLAY SUBTASK EFFECTIVENESS. Using a UCFBATCH program is a good method for obtaining this information.
5. Compare the TCB column from Step 2 with that from Step 4. The difference is proportional to the potential reduction in both the total CPU use and the Total Cost of Ownership (TCO) that can be achieved by using the zIIP feature.

   The SRB column from Step 4 is proportional to the number of MIPs of zIIP processing power that will be required to achieve these cost reductions

Examples

The following displays indicate that each run used between 93 and 98 CPU seconds of total normalized CPU. The second run shows that 91.1222 CPU seconds out of a total of 93.5108 CPU seconds were offloaded to an SRB. Approximately one third of this SRB CPU time can be offloaded to a zIIP processor. For this particular application mix, this means that approximately one third of the total CV CPU could be offloaded to zIIP processors.

Since a zIIP processor was present, the actual offloaded CPU can then be confirmed from the JES LOG Step End messages, IEF374I, which, in this case, indicates a total CPU reduction of 31.68 seconds.

Step 2 Output with ZIIP=N

```
*** Subtask display ***
Subtask          Elapsed time                 Total CPU time         % CPU  SRB
 Name         TCB            SRB            TCB            SRB       TCB SRB
-------- -------------- -------------- -------------- -------------- --- ---
MAINTASK  00:00:14.0505  00:00:00.0000  00:00:02.3699  00:00:00.0000  16 N/A  N
SUBT0001  00:00:00.0122  00:00:00.0000  00:00:00.0104  00:00:00.0000  85 N/A  N
SUBT0002  00:00:00.0175  00:00:00.0000  00:00:00.0131  00:00:00.0000  74 N/A  N
SUBT0003  00:00:00.2348  00:00:00.0000  00:00:00.0398  00:00:00.0000  16 N/A  N
SUBT0004  00:00:00.2175  00:00:00.0000  00:00:00.0240  00:00:00.0000  11 N/A  N
SUBT0005  00:01:42.0081  00:00:00.0000  00:01:35.8538  00:00:00.0000  93 N/A  N
-------- -------------- -------------- -------------- -------------- --- ---
Totals    00:01:56.5406  00:00:00.0000  00:01:38.3110  00:00:00.0000  84 N/A
```

JES LOG Step End Message

```
IEF374I STEP/DCV     /STOP  2008242.0425 CPU    1MIN 39.53SEC SRB
0MIN 11.73SEC VIRT  7840K SYS   552K EXT   56072K SYS   11460K
```

Step 4 Output with ZIIP=Y

```
*** Subtask display ***
Subtask          Elapsed time                 Total CPU time         % CPU  SRB
 Name         TCB            SRB            TCB            SRB       TCB SRB
-------- -------------- -------------- -------------- -------------- --- ---
MAINTASK  00:00:12.9116  00:00:00.0434  00:00:02.1387  00:00:00.0525  16 120  Y
SUBT0001  00:00:00.0125  00:00:00.0000  00:00:00.0111  00:00:00.0000  88 N/A  Y
SUBT0002  00:00:00.0118  00:00:00.0002  00:00:00.0103  00:00:00.0000  87  00  Y
SUBT0003  00:00:00.1643  00:00:00.0014  00:00:00.0624  00:00:00.0016  37 114  Y
SUBT0004  00:00:00.0276  00:00:00.0004  00:00:00.0239  00:00:00.0008  86 200  Y
SUBT0005  00:00:00.3849  00:01:17.1037  00:00:00.1422  00:01:31.0673  36 118  Y
-------- -------------- -------------- -------------- -------------- --- ---
Totals    00:00:13.5127  00:01:17.1491  00:00:02.3886  00:01:31.1222  17 118
```

JES LOG Step End Message

```
IEF374I STEP/DCV     /STOP  2008242.0400 CPU    1MIN 01.60SEC SRB
0MIN 17.98SEC VIRT  7840K SYS   552K EXT   56080K SYS   11500K
```

More Information

For more information about the DCMT DISPLAY SUBTASK command, see the *CA IDMS System Tasks and Operator Commands Guide*.

Chapter 5. SQL

This chapter describes the new SQL enhancements. It contains the following topics:

5.1 SQL Procedural Language Support in Routines

This new feature adds SQL as a programming language for SQL-invoked procedures and functions. Earlier releases of CA IDMS provided support for Cobol, PL/I, Assembler, and CA ADS.

The new SQL language elements, syntax, and terminology are fully compliant with the ISO standards, except where noted.

The SQL language for SQL routines includes syntax to perform the following:

- Direct the flow of control
- Assign the result of expressions to variables and parameters
- Specify condition handlers to process various conditions
- Signal and resignal conditions
- Declare local cursors
- Declare local variables

The advantages of writing SQL routines in the SQL language include:

- The ability to use a readable, simple yet powerful programming language
- A single language to access and process data
- Native support for all the SQL data types making manipulation of VARCHAR, DATE, TIME, and TIMESTAMP data easier
- Built-in NULL support that eliminates the need for defining and manipulating NULL indicators for data such as table columns and parameters of SQL routines
- Flexible handlers able to process SQL events easily
- A single development and test platform fully integrated in all the CA IDMS supported environments

5.1.1 New Terminology

This feature introduces the following new and changed terminology for routines invoked through SQL:

SQL-invoked routine
Specifies a routine that is allowed to be invoked only from within SQL. An SQL-invoked routine can be defined in the SQL catalog as a procedure, function, or table procedure.

SQL-invoked procedure
Specifies an SQL-invoked routine defined as a procedure in the SQL catalog.

SQL-invoked function
Specifies an SQL-invoked routine defined as a function in the SQL catalog.

SQL routine
Specifies an SQL-invoked routine whose language attribute is SQL. Because table procedures cannot be written in the SQL language, an SQL routine is necessarily defined as a procedure or a function.

SQL procedure
Specifies an SQL routine defined in the SQL catalog as a procedure with language attribute SQL.

SQL function
Specifies an SQL routine defined in the SQL catalog as a function with language attribute SQL.

5.1.2 Implementing SQL Routines

To define an SQL routine, use a CREATE FUNCTION or CREATE PROCEDURE statement and specify LANGUAGE SQL. In the same statement, also specify the SQL statements that make up the body of the routine. These are the statements that are executed when the routine is invoked.

After successful creation of an SQL routine, it can be invoked immediately. No additional steps, such as creating an access module are needed.

Routines written in the SQL language are implemented internally as CA ADS When an SQL routine is successfully created, it results in the creation of the following objects:

- A mapless dialog whose name is the external name specified on the routine definition.
- A process module whose name is PREMAP-*dialog-name*. This premap process has a builder code of "Q" and identifies the owning routine in its description.
- Zero or more SQL tables used to represent the local variables defined in compound statements within the routine. These tables have names of the form *schema-name*.SQLLOCnnnn*dialog-name*.
- An access module whose name is the same as that of the dialog.

The following considerations apply when creating SQL routines:

- The dictionary to which the SQL session is connected must include DDLDML and DDLDCLOD areas and these areas must be updatable.
- SQL routines require more space in both the catalog and the DDLDML and DDLDCLOD areas than SQL-invoked routines written in other languages.
- The following attribute settings are required for SQL routines. You do not need to specify these when defining an SQL routine, but if you do, their values must be as indicated:

  - The protocol must be ADS
  - The mode of the SQL routine must be SYSTEM
  - Transaction sharing must be ON

- When using the command facility to define an SQL routine whose body contains multiple statements, it is necessary to change the command delimiter to distinguish termination of the routine definition from termination of the SQL statements that make up the routine body. The following example illustrates the use of the character string "++" as a command delimiter:

```
set options command delimiter '++';
create procedure DEFJE01.TIF1
  ( TITLE   varchar(10) with default
  , P_LEFT    integer
  , P_RIGHT   real
  , RESULT  varchar(30)
  )
    EXTERNAL NAME TIF1 LANGUAGE SQL
Label_200:
begin not atomic
 /*
 ** Compare an integer value with a real value
 */
  if (P_LEFT > P_RIGHT)
    then set RESULT = 'p_left > p_right';
  elseif (P_LEFT = P_RIGHT)
    then set RESULT = 'p_left = p_right';
  elseif  (P_LEFT < P_RIGHT)
    then set RESULT = 'p_left < p_right';
  else   set RESULT = 'p_left and/or p_right NULL !';
  end if;
end
++
```

Note: For more information about changing the command delimiter, see 5.7, “SET OPTIONS COMMAND DELIMITER” on page 188.

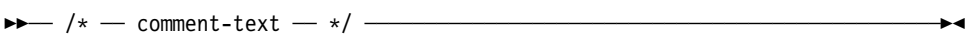
5.1.3 Statement Components

The new SQL statement components are described in this section.

5.1.3.1 Bracketed Comment

The comment capabilities within SQL have been extended to support bracketed comments. This enables multiple statements to be designated as comments simply by enclosing them within comment delimiters.

Syntax



Parameters

comment-text
Specifies the text to be designated as a comment.

Usage

Where bracketed comments can be used: Bracketed comments are only allowed in the routine body of an SQL routine. Outside of this context, they are not recognized.

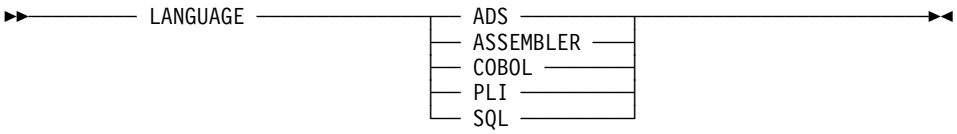
Coding conventions: The bracket introducer '/*' and terminator '*/' strings cannot be split over two lines. They can be specified wherever a separator or space is allowed.

When defining an SQL routine using the command facility tools OCF, IDMSBCF, or the command console in CA IDMS Visual DBA, the comment introducer '/*' cannot be placed in column 1, because a '/*' is interpreted as an end of file on the input by the command processor.

5.1.3.2 Expansion of language-clause

The *language-clause* specifies the programming language of an SQL-invoked routine. This clause is required for SQL routines written in SQL. For others, it is documentational only. The *language-clause* can be used on a CREATE or ALTER PROCEDURE and on a CREATE or ALTER FUNCTION statement.

Syntax



Parameters

ADS
Specifies that the SQL routine is written in the CA ADS language.

ASSEMBLER
Specifies that the SQL routine is written in the assembler language.

COBOL
Specifies that the SQL routine is written in the COBOL language.

PLI
Specifies that the SQL routine is written in the PL/I language.

SQL
Specifies that the SQL routine is written in the SQL language.

Note: The ability to specify ADS or ASSEMBLER as a language is a CA IDMS extension.

Usage

Specifying Language SQL: If LANGUAGE SQL is specified when creating a routine, the following routine attributes are established by default and any attempt to override them to other values will fail:

- Protocol is ADS
- Mode is SYSTEM
- Transaction sharing is ON

If no Language is Specified:: If the language is not specified when a routine is created, it is treated as null. There is no default.

Note: In the ISO standard, the default for LANGUAGE is SQL.

Example

The TLANG1 function defined in the schema USER01 returns the edited name, given the first and last names.

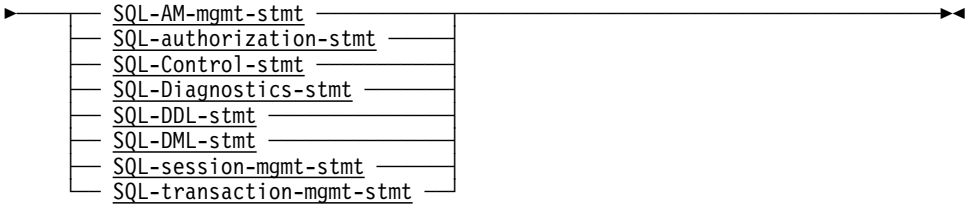
```
create function USER01.TLANG1
  ( P_FNAME      char(20)
  , P_LNAME      char(20)
  ) returns   varchar(41)
    external name TLANG1 language SQL
    return trim(P_FNAME) || ' ' || trim(P_LNAME);

select USER01.TLANG1('James    ', 'Last   ')
  from SYSCA.SINGLETON_NULL;

*+
*+ USER_FUNC
*+ ---------
*+ James Last
```

5.1.3.3 Expansion of procedure-statement

Defines the SQL statements that can be included in the body of an SQL routine or in an SQL Control statement.

Syntax

Expansion of procedure-statement



Parameters

SQL-AM-mgmt-stmt
Specifies a statement from the Access Module Management Statements category.

SQL-authorization-stmt
Specifies a statement from the Authorization Statements category.

SQL-Control-stmt
Specifies a statement from the Control Statements category.

SQL-Diagnostics-stmt
Specifies a statement from the Diagnostics Statements category.

SQL-DDL-stmt
Specifies a statement from the Data Description Statements category.

SQL-DML-stmt
Specifies a statement from the Data Manipulation Statements category.

SQL-session-mgmt-stmt
Specifies a statement from the Session Management Statements category.

Note: The ability to include a RELEASE, SUSPEND, or RESUME statement in an SQL routine is a CA IDMS extension.

SQL-transaction-mgmt-stmt
Specifies a statement from the Transaction Management Statements category.

Note: The ability to include a COMMIT or ROLLBACK statement in an SQL routine is a CA IDMS extension.

Usage

Grouping procedure statements into a single statement: Multiple procedure statements can be grouped together as a compound statement. A compound statement is a control statement and therefore is also a procedure statement.

More Information

- For more information about Control Statements, see 5.1.5, “Control Statements” on page 132.
- For more information about Diagnostics Statements, see 5.3, “Enhanced Diagnostics and Statistics” on page 173.
- For more information about the other categories, see Statement Categories in the "Statements" chapter in the *CA IDMS SQL Reference Guide*.

5.1.3.4 Local Variables

Local variables are new entities introduced in support of SQL routines.

A local variable is a variable that is defined in an SQL routine. You use local variables to temporarily store and retrieve values as needed in the logic of the routine. Local variables are used for such things as:

- Retrieving data from a CA IDMS database by specifying them on the INTO clause of a SELECT statement
- Passing data to and from other SQL-invoked routines by specifying them as arguments on the routine invocation
- Holding computational values by specifying them as a target of a SET statement or as values within expressions.

Local variables can only be referenced within the body of the SQL routine in which they are defined.

Declaring Local Variables

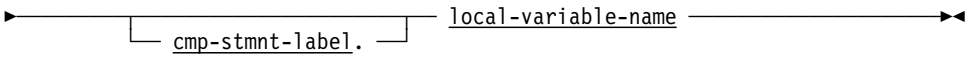
A local variable is defined by a **variable-declaration** statement that is included in a compound statement within an SQL routine body. The declaration of a local variable consists of the specification of its name, data type, and optionally its initial value. For more information about declaring local variables, see 5.1.5.3, “Compound Statement” on page 136.

5.1.3.5 Expansion of Local-variable

Identifies a local variable declared in a compound statement.

Syntax

Expansion of local-variable



Parameters

cmp-stmnt-label
Specifies the label of the compound statement that contains the definition of **local-variable**.

local-variable-name
Identifies the local variable of an SQL routine.

Usage

Referencing Local Variables: A local variable can only be referenced from within the compound statement that contains its declaration or from within a compound statement contained in the compound statement that contains its declaration.

Avoiding Ambiguous References: The name of a local variable of an SQL routine can be the same as the name of another local variable, a routine parameter, a column, or another schema-defined entity such as a table. To avoid ambiguity when referencing these objects, qualification can be used as follows:

- A local variable can be qualified with the label of the compound statement in which it is declared.
- A routine parameter can be qualified with its associated schema and routine name.
- A column can be qualified with its schema and table name.
- Other schema-defined objects can be qualified with the name of the schema in which they are defined.

Resolving Ambiguous References: If a name is not qualified and more than one object has the specified name, CA IDMS uses the following precedence rules to resolve the ambiguous reference:

- If a local variable with a matching name has been declared within the compound statement in which the reference occurs, the reference is to the local variable. If more than one such variable is declared, the reference is to the variable declared in the innermost compound statement containing the reference.

- If a parameter of the routine in which the reference occurs has a matching name, the reference is to the routine parameter.
- Otherwise, the reference is treated as a reference to a schema-defined object. For information on how such a reference is resolved, see Resolving References to Entities in Schemas in "Identifiers" in the *CA IDMS SQL Reference Guide*.

Note: In the ISO standard, an unqualified reference would be to the object with innermost scope.

Example

In the following SQL procedure, two local variables, FNAME and LNAME are defined. The references are qualified in the SELECT statement with the label of the compound statement that holds the definition of the local variables. The SET statement uses unqualified references.

```
set options command delimiter '++';
create procedure SQLROUT.LOCALVAR
  ( TITLE     varchar(10) with default
  , P_EMP_ID  NUMERIC(4)
  , P_NAME    varchar(25)
  )
    external name LOCALVAR language SQL
L_MAIN: begin not atomic
 /*
** Count number of employees with equal Firstname using REPEAT
*/
 declare FNAME   char(20);
 declare LNAME   varchar(20);

 select EMP_FNAME, EMP_LNAME
   into L_MAIN.FNAME, L_MAIN.LNAME
   from DEMOEMPL.EMPLOYEE
  where EMP_ID = P_EMP_ID;

  set P_NAME = FNAME || LNAME;
end L_MAIN
++
*+ TITLE       P_EMP_ID  P_NAME
call SQLROUT.LOCALVAR('LOCALVAR',2010)++
*+

*+ -----       --------  -------------
*+ LOCALVAR    2010      Cora    Parke
```

5.1.3.6 Routine Parameter

Routine parameters are new entities introduced in support of SQL routines.

A routine parameter is a parameter of an SQL routine. You use routine parameters to perform the following:

- Pass values to and from the SQL routine

- Store and retrieve values as needed by the routine logic
- Pass values to other SQL-invoked routines

Routine parameters can only be referenced within the body of the SQL routine in which they are defined.

Defining Routine Parameters

A routine parameter is defined through a **parameter-definition** clause of the CREATE PROCEDURE or CREATE FUNCTION statements. The definition includes the specification of the name, the data type, and optional WITH DEFAULT attribute.

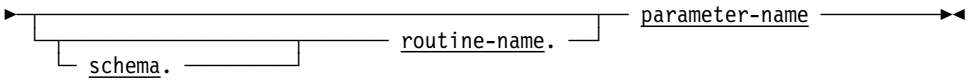
Note: For more information about defining routine parameters, see CREATE PROCEDURE and CREATE FUNCTION statements in the *CA IDMS SQL Reference Guide*.

5.1.3.7 Expansion of Routine-parameter

Identifies a routine parameter of an SQL routine.

Syntax

Expansion of routine-parameter



Parameters

schema
Specifies the schema with which the SQL routine identified by *routine-name* is associated.

routine-name
Specifies the name of the SQL routine in which the routine parameter identified by **routine-parameter** is defined.

parameter-name
Identifies a parameter of an SQL routine.

Usage

Referencing Routine Parameters: Routine parameters can only be referenced within the body of the SQL routine in which they are defined. A routine parameter is global to the SQL routine. It can be referenced anywhere in the body of the routine.

Avoiding Ambiguous References: The name of a routine parameter can be the same as the name of a local variable, a column, or another schema-defined entity such as a table. To avoid ambiguity when referencing these objects, qualification can be used as follows:

- A local variable can be qualified with the label of the compound statement in which it is declared
- A routine parameter can be qualified with its associated schema and routine name
- A column can be qualified with its schema and table name
- Other schema-defined objects can be qualified with the name of the schema in which they are defined.

Resolving Ambiguous References: If a name is not qualified and more than one object has the specified name, CA IDMS uses the following precedence rules to resolve the ambiguous reference:

- If a local variable with a matching name has been declared within the compound statement in which the reference occurs, the reference is to the local variable. If more than one such variable is declared, the reference is to the variable declared in the innermost compound statement containing the reference.
- If a parameter of the routine in which the reference occurs has a matching name, the reference is to the routine parameter.
- Otherwise, the reference is treated as a reference to a schema-defined object. For information on how such a reference is resolved, see Resolving References to Entities in Schemas in "Identifiers" in the *CA IDMS SQL Reference Guide*.

Note: In the ISO standard, an unqualified reference would be to the object with innermost scope.

Example

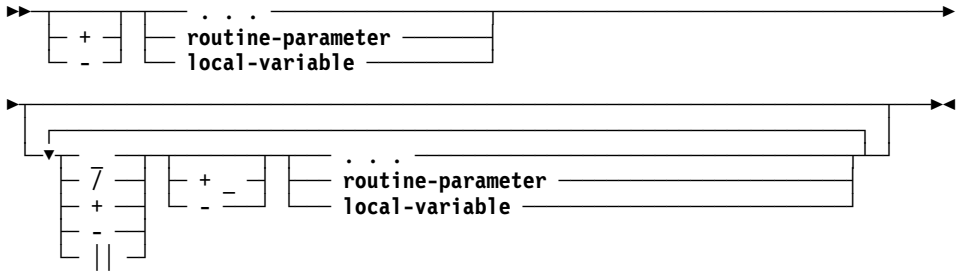
In the following SQL procedure, three routine parameters, TITLE, P_EMP_ID, and P_LAST_NAME are defined. The references are to P_EMP_ID and P_LAST_NAME in the SELECT statement are qualified. The SET statement uses an unqualified reference to TITLE.

```
.
set options command delimiter '++';
create procedure SQLROUT.GETLNAME
  ( TITLE     varchar(10) with default
  , P_EMP_ID  NUMERIC(4)
  , P_LAST_NAME    varchar(25)
  )
    external name GETLNAME language SQL
L_MAIN: begin not atomic

 select EMP_FNAME
   into SQLROUT.GETLNAME.P_LAST_NAME
   from DEMOEMPL.EMPLOYEE
  where EMP_ID = GETLNAME.P_EMP_ID;

  set TITLE = 'Success';
end L_MAIN
++

call SQLROUT.GETLNAME  ('?',2010)++
**
** TITL:EP_EMP_ID   P_LAST_NAME
** -----       --------   -----------
** Success         2010   Cora
```

5.1.3.8 Expansion of value-expression

The expansion of value-expression has been enhanced to enable referencing parameters and local variables of SQL routines.

Syntax



Parameters

routine-parameter

Specifies a parameter of an SQL routine to be used as a single operand in the value expression. For information about expanded **routine-parameter** syntax, see 5.1.3.7, “Expansion of Routine-parameter” on page 119.

local-variable

Specifies a local variable of an SQL routine to be used as a single operand in the value expression. For information about expanded **local-variable** syntax, see 5.1.3.5, “Expansion of Local-variable” on page 117.

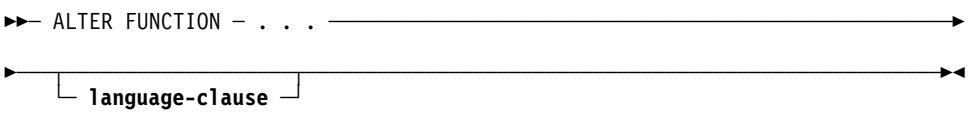
5.1.4 Enhanced Data Description Statements

This section contains data description statements that have been enhanced in support of SQL routines.

5.1.4.1 ALTER FUNCTION

The ALTER FUNCTION statement has been enhanced with the addition of the *language-clause* which is used to change the language of the function.

Syntax



Parameters

language-clause
Specifies the programming language of the function.

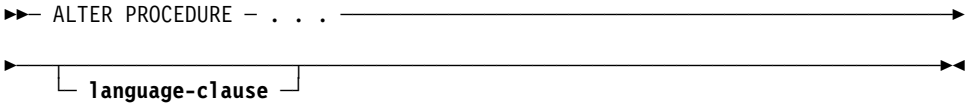
Usage

Changing the language of a function: A function with language SQL cannot be changed to any other language and a function whose language is not SQL cannot be changed to language SQL.

5.1.4.2 ALTER PROCEDURE

The ALTER PROCEDURE statement has been enhanced with the addition of the *language-clause* which is used to change the language of the procedure.

Syntax



Parameters

language-clause
Specifies the programming language of the procedure.

Usage

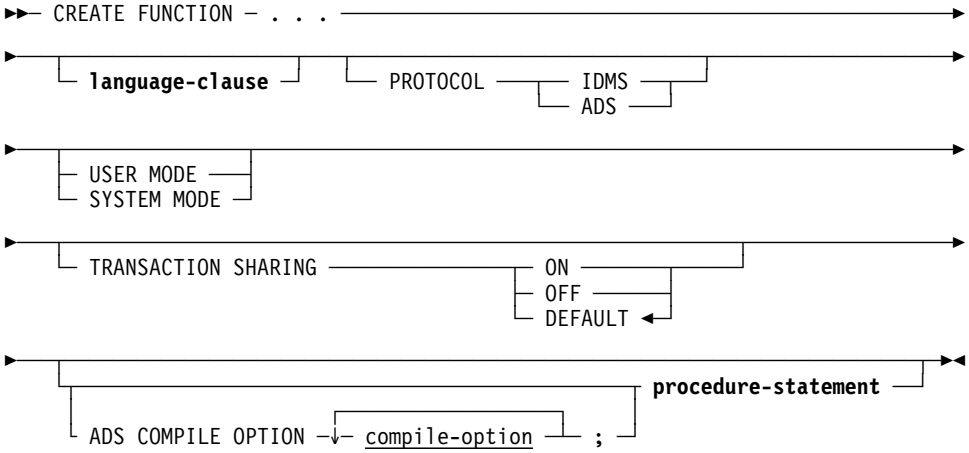
Changing the language of a procedure: A procedure with language SQL cannot be changed to any other language and a procedure whose language is not SQL cannot be changed to language SQL.

5.1.4.3 CREATE FUNCTION

The CREATE FUNCTION statement has been enhanced to enable the definition of functions written in the SQL language. The CREATE FUNCTION statement is a data description statement that stores the definition of a function in the SQL catalog. You can then invoke the function in any value-expression of an SQL statement except in the search condition of a table's check constraint. The function invocation results in CA IDMS calling the corresponding routine. Such routines can perform any action and return a single scalar value. Use the formal parameters of a function definition to specify the data type and format of the data to be passed to the function. Similarly, the data type of the return value is specified in the function definition.

Functions can be defined with a language of SQL, in which case, the routine actions written as SQL statements are specified and stored together with the function definition in the SQL catalog.

Syntax



Parameters

language-clause

Specifies the programming language of the function. This clause is required for functions written in SQL. For others, it is documentational only. If the language is not specified, it is treated as null.

PROTOCOL

Specifies the protocol with which the function is invoked. This specification is required except with language SQL. If LANGUAGE SQL is specified, PROTOCOL must be ADS or the clause must not be specified.

IDMS

Use IDMS for functions that are written in COBOL, PL/I, or Assembler.

ADS

Use ADS for functions that are written in CA ADS. The name of the dialog that is loaded and executed when the function is invoked is

specified by the external-routine-name in the EXTERNAL NAME clause. ADS is the default if LANGUAGE SQL is specified.

USER MODE
Specifies that the function should execute as a user-mode application program within CA IDMS. This cannot be specified with language SQL or protocol ADS. For other languages and protocols, it is the default.

SYSTEM MODE
Specifies that the function should execute as a system-mode application program. SYSTEM MODE is the default if language is SQL or protocol is ADS.

To execute as SYSTEM MODE, the program must be one of the following:

- A fully reentrant Assembler program
- A Language Environment (LE) COBOL or PL/I program
- A mapless dialog

TRANSACTION SHARING
Specifies whether transaction sharing should be enabled for database sessions started by the function. If transaction sharing is enabled for a function's database session, it shares the current transaction of the SQL session. If language SQL is specified, TRANSACTION SHARING must be ON or the clause must not be specified.

ON
Specifies that transaction sharing should be enabled. ON is the default if language is SQL.

OFF
Specifies that transaction sharing should be disabled.

DEFAULT
Specifies that the transaction sharing setting in effect when the function is invoked should be retained. DEFAULT is the default for languages other than SQL.

compile-option
Specifies a CA ADS option to be used when compiling the dialog associated with an SQL function. The options that can be specified and the syntax to use are given in the *CA ADS Reference Guide*, Appendix D.2.6 Dialog-expression. *Compile-option* can be specified only if language is SQL.

Note: The ability to specify the ADS COMPILE OPTION clause is a CA IDMS extension.

procedure-statement
Specifies the actions taken in the function. **Procedure-statement** is required if language is SQL. It cannot be specified otherwise.

Usage

Language SQL: If LANGUAGE SQL is specified, the following attribute settings are established by default and must not be overridden to a different value:

- Protocol is ADS
- Mode is SYSTEM
- Transaction sharing is ON

Functions whose language is SQL are implemented through an automatically generated CA ADS dialog whose name is *external-routine-name*.

An error while parsing **procedure-statement** or an error while compiling the associated CA ADS dialog causes termination of the CREATE FUNCTION statement with a warning instead of a statement error. This allows the erroneous **procedure-statement** syntax to be saved in the catalog for later correction using the DISPLAY FUNCTION command. The CA ADS dialog and associated access module are not created.

Specifying CA ADS Compile Options: If LANGUAGE SQL is specified, you can specify one or more compile options to be used when the associated dialog is compiled. Specifying compile options can be useful for debugging purposes to enable tracing and the use of online debugging facilities. Compile options can also be used to include additional work records and SQL tables which can be referenced in native CA ADS code included in the routine body.

Some useful compile options include:

- SYMBOL TABLE IS YES - to allow the use of symbols by the TRACE command and the online debug facilities
- ADD RECORD record-name - to enable manipulation of elements from the specified record
- ADD SQL TABLE table-name - to enable manipulation of columns or parameters of the specified SQL table-like object.

Example

```
set options command delimiter '++';
drop function USER01.TCNTEQNAME++
commit++
create function USER01.TCNTEQNAME
  ( TITLE     varchar(40) with default
  , P_FNAME   char(20)
  , P_COUNT     integer
  , RESULT    varchar(10)
  ) RETURNS   varchar(20)
    EXTERNAL NAME TCNTEQN LANGUAGE SQL
```

```
Label_700:
begin not atomic
 /*
 ** Count number of employees with equal Firstname
 */
  declare FNAME       char(20);
  declare LNAME       varchar(20);
  declare P_COUNT_SAV integer default 0;

  declare EMP1 CURSOR FOR
        Select EMP_FNAME, EMP_LNAME
          From DEMOEMPL.EMPLOYEE
         where EMP_FNAME = P_FNAME;
   open EMP1;
  fetch EMP1 into FNAME, LNAME;

  fetching_loop:

  loop
      if (SQLSTATE < > '00000')
        then leave fetching_loop;
      end if;
      set P_COUNT = P_COUNT + 1;
      fetch EMP1 into FNAME, LNAME;
  end loop fetching_loop;
  set RESULT = SQLSTATE;
  close EMP1;
  if (P_COUNT < = P_COUNT_SAV)
    then return null;
    else return 'Res: ' || cast(P_COUNT as char(5));
  end if;
end
++
```

5.1.4.4 CREATE PROCEDURE

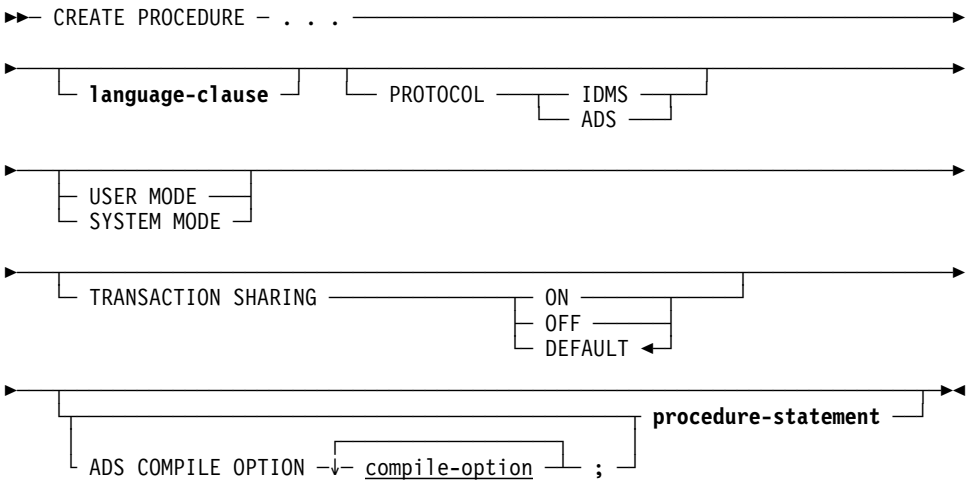
The CREATE PROCEDURE statement has been enhanced to enable the definition of procedures written in the SQL language.

The CREATE PROCEDURE statement is a data description statement that stores the definition of a procedure in the SQL catalog. You can refer to the procedure in an SQL CALL statement or in an SQL SELECT statement just as you would a table procedure. These references result in CA IDMS calls to the corresponding routine. Such routines can perform any action, such as manipulating data stored in some other organization (for example, in a non SQL-defined database or in a set of VSAM files). You can also use them to implement business logic.

Procedures can be defined with a language of SQL. The routine actions, written as SQL statements, are specified and stored together with the procedure definition in the SQL catalog.

The formal parameters of a procedure definition can be used like columns of a table during a procedure invocation to pass values to and from the procedure.

Syntax



Parameters

language-clause

Specifies the programming language of the procedure. This clause is required for procedures written in SQL. For others, it is documentational only. If the language is not specified, it is treated as null.

PROTOCOL

Specifies the PROTOCOL with which the procedure is invoked. This specification is required except with language SQL. If LANGUAGE SQL is specified, PROTOCOL must be ADS or the clause must not be specified.

IDMS

Use IDMS for procedures that are written in COBOL, PL/I, or Assembler.

ADS

Use ADS for procedures that are written in CA ADS. The name of the dialog that is loaded and executed when the procedure is invoked is specified by the *external-routine-name* in the EXTERNAL NAME clause. ADS is the default if LANGUAGE SQL is specified.

USER MODE

Specifies that the procedure should execute as a user-mode application program within CA IDMS. This cannot be specified with language SQL or protocol ADS. For other languages and protocols, it is the default.

SYSTEM MODE

Specifies that the procedure should execute as a system-mode application program. SYSTEM MODE is the default if language is SQL or protocol is ADS.

To execute as SYSTEM MODE, the program must be one of the following:

- A fully reentrant Assembler program
- A Language Environment (LE) COBOL or PL/I program

- A mapless dialog

TRANSACTION SHARING

Specifies whether transaction sharing should be enabled for database sessions started by the procedure. If transaction sharing is enabled for a procedure's database session, it shares the current transaction of the SQL session. If language SQL is specified, TRANSACTION SHARING must be ON or the clause must not be specified.

ON

Specifies that transaction sharing should be enabled. ON is the default if language is SQL.

OFF

Specifies that transaction sharing should be disabled.

DEFAULT

Specifies that the transaction sharing setting in effect when the procedure is invoked should be retained. Default is the default for languages other than SQL.

compile-option

Specifies a CA ADS option to be used when compiling the dialog associated with an SQL procedure. The options that can be specified and the syntax to use are given in the *CA ADS Reference Guide*, Appendix D.2.6 Dialog-expression. *Compile-option* can be specified only if language is SQL.

Note: The ability to specify the ADS COMPILE OPTION clause is a CA IDMS extension.

procedure-statement

Specifies the actions taken in the procedure. **Procedure-statement** is required if language is SQL. It cannot be specified otherwise.

Usage

Language SQL: If LANGUAGE SQL is specified, the following attribute settings are established by default and must not be overridden to a different value:

- Protocol is ADS
- Mode is SYSTEM
- Transaction sharing is ON

Procedures whose language is SQL are implemented through an automatically generated CA ADS dialog whose name is *external-routine-name*.

An error while parsing **procedure-statement** or an error while compiling the associated CA ADS dialog causes the CREATE PROCEDURE statement to terminate with a warning instead of a statement error. This allows the erroneous **procedure-statement** syntax to be saved in the catalog for later correction

using the DISPLAY PROCEDURE command. The CA ADS dialog and associated access module are not created.

Specifying CA ADS Compile Options: If LANGUAGE SQL is specified, you can specify one or more compile options to be used when the associated dialog is compiled. Specifying compile options can be useful for debugging purposes to enable tracing and the use of online debugging facilities. Compile options can also be used to include additional work records and SQL tables which can be referenced in native CA ADS code included in the routine body.

Some useful compile options include:

- SYMBOL TABLE IS YES - to allow the use of symbols by the TRACE command and the online debug facilities
- ADD RECORD record-name - to enable manipulation of elements from the specified record
- ADD SQL TABLE table-name - to enable manipulation of columns or parameters of the specified SQL table-like object

Example

The procedure USER01.TSELECT1 uses the given employee ID to retrieve the first and last name. It returns the edited name in the RESULT parameter.

```
create procedure USER01.TSELECT1
  ( TITLE       varchar(10) with default
  , P_EMP_ID    numeric(4)
  , RESULT      varchar(20)
  )
    EXTERNAL NAME TSELECT1 LANGUAGE SQL
 select trim(EMP_FNAME) || ' ' || trim(EMP_LNAME)
   into RESULT
   from DEMOEMPL.EMPLOYEE
  where EMP_ID = P_EMP_ID
;

call user01.tselect1('TSIGNAL3', 1003);
*+
*+ TITLE       P_EMP_ID  RESULT
*+ -----       --------  ------
*+ TSIGNAL3        1003  Jim Baldwin
```

5.1.4.5 DISPLAY/PUNCH FUNCTION

The DISPLAY/PUNCH FUNCTION statement has been enhanced to display the SQL statements that make up the body of a function written in SQL. The DISPLAY/PUNCH FUNCTION statement lets you display or punch a function. For functions with language SQL, the statement also displays the SQL routine body from the dictionary.

5.1.4.6 DISPLAY/PUNCH PROCEDURE

The DISPLAY/PUNCH PROCEDURE statement has been enhanced to display the SQL statements that make up the body of a procedure written in SQL. The DISPLAY PROCEDURE statement displays or punches the definition of a procedure. For procedures with language SQL, the statement also displays the SQL routine body from the dictionary.

5.1.4.7 DROP FUNCTION

The DROP FUNCTION statement has been enhanced to delete the additional components created in support of a function written in SQL. The DROP FUNCTION statement is a data description statement that deletes the definition of the referenced function from the dictionary. For functions with language SQL, the statement removes the SQL routine body from the dictionary and the associated entities: access module(AM), relational command module (RCM), ADS premap process code, and dialog load module.

If in the same SQL session the DROP of an SQL function is followed by a CREATE of an SQL routine with an external name identical to that of the dropped function, a COMMIT should follow the DROP to avoid deadlocks on the load module resources.

5.1.4.8 DROP PROCEDURE

The DROP PROCEDURE statement has been enhanced to delete the additional components created in support of a procedure written in SQL. The DROP PROCEDURE statement is a data description statement that deletes the definition of the referenced procedure from the dictionary. For procedures with language SQL, the statement removes the SQL routine body from the dictionary and the associated entities: access module(AM), relational command module (RCM), ADS premap process code, and dialog load module.

If in the same SQL session the DROP of an SQL procedure is followed by a CREATE of an SQL routine with an external name identical to that of the dropped procedure, a COMMIT should follow the DROP to avoid deadlocks on the load module resources.

5.1.4.9 DROP SCHEMA

The DROP SCHEMA statement has been enhanced to delete the additional components created in support of SQL routines. The DROP SCHEMA statement is a data description statement that deletes a schema definition from the dictionary. The DROP SCHEMA statement is a CA IDMS extension of ANSI-standard SQL.

Usage

Effect of the CASCADE Parameter: When you specify CASCADE in a DROP SCHEMA statement, CA IDMS deletes the following:

- For functions and procedures with language SQL, the statement removes the SQL routine body from the dictionary and the associated CA ADS entities and program structures: access module(AM), relational command module (RCM), ADS premap process code and dialog load module.

5.1.5 Control Statements

This is a new category of CA IDMS SQL statements that allow you to define the flow of control in an SQL routine and assign values to routine parameters or local variables.

SQL Control Statements

Statement	Purpose
CALL	Invokes a procedure or a table procedure. **Note:** The CALL statement has been available in earlier releases of CA IDMS, where it has been categorized as a DML-statement.
CASE	Determines the execution flow by the evaluation of one or more value-expressions.
Compound	Specifies a grouping of statements, with optional definitions of local variables, cursors, and handlers.
EXEC ADS	Starts a block of CA ADS code.
IF	Determines by evaluation of a search-condition, which block of statements are executed.
ITERATE	Begins a new iteration in a programmatic loop.
LEAVE	Exits a programmatic loop.
LOOP	Defines a programmatic loop.
REPEAT	Defines a programmatic loop with an end condition.
RESIGNAL	Raises an SQL exception in a handler.
RETURN	Exits an SQL-invoked routine or compound statement.
SET Assignment	Assigns a value to a routine parameter, local variable, or host variable. **Note:** This statement can also be embedded in any SQL client program.

Statement	Purpose
SIGNAL	Raises an SQL exception.
WHILE	Defines a programmatic, conditional loop.

5.1.5.1 CALL

The CALL statement has been enhanced to enable the invocation of SQL procedures.

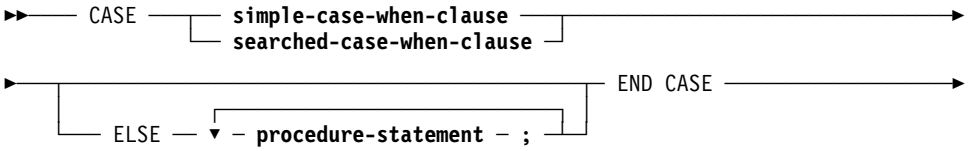
Usage

Calling an SQL Procedure: An SQL procedure is an SQL-invoked procedure with language SQL. Any transaction started by this procedure is shared with the transaction of the caller. After returning from an SQL procedure, any session opened by the procedure is automatically released except for sessions that have result sets. Such sessions are released when their last result set has been processed and the associated received cursor has been closed.

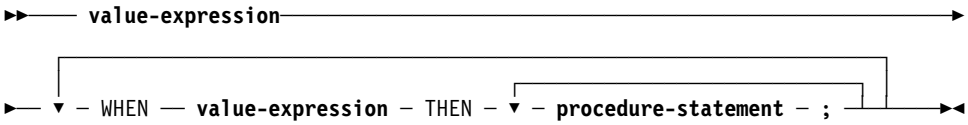
5.1.5.2 CASE

The CASE statement selects different execution paths depending on the evaluation of one or more *value-expressions*.

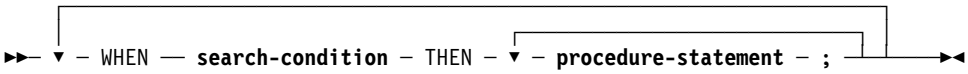
Syntax



Expansion of simple-case-when-clause



Expansion of searched-case-when-clause



Parameters

Simple Case:

CASE value-expression
Specifies the value expression whose outcome is compared to the outcomes of the **value-expression**s in the WHEN clauses.

WHEN value-expression
Specifies a value expression whose outcome is compared to the outcome of the CASE **value-expression**. If the two values are equal, the group of statements specified in the corresponding THEN is executed.

THEN procedure-statement
Identifies the group of statements to be executed when the value expressions of the CASE and WHEN clauses are equal.

Searched Case:

CASE WHEN
Identifies the CASE as a searched case.

WHEN search-condition
Specifies the search condition whose outcome, if true, results in the execution of the group of statements specified by the THEN clause.

THEN procedure-statement
Identifies the group of statements executed when the **search-condition** in the corresponding WHEN clause evaluates to true.

ELSE procedure-statement END CASE
Specifies the group of statements to be executed when none of the THEN group of statements has been executed because of the evaluation and comparison of the **value-expression**'s and **search-condition**'s. This clause can be specified for both simple and searched case statements.

Usage

SQL Exceptions: If an ELSE clause is not specified and none of the THEN group of statements has been executed because of the outcome of evaluation of the value expressions and search conditions, an SQL exception is raised.

Examples

The first example demonstrates the use of a **simple-case-when-clause**.

```
set options command delimiter '++';
create function USER01.TCASE1
  ( TITLE     varchar(40) with default
  , P_EMP_ID  unsigned numeric(4)
  ) RETURNS   varchar(30)
    external name TCASE1 language SQL
begin not atomic
  /*
  ** Function selects an employee with the given EMP_ID and swaps
  ** the first_name value 'James' with 'Jim'.
  ** Returns a message text with the outcome of the execution
  */
  declare MY_STATUS varchar(30);
  declare LOC_FNAME char(20) default ' ';

  select EMP_FNAME into LOC_FNAME
    from DEMOEMPL.EMPLOYEE
```

```
   where EMP_ID = P_EMP_ID;

  case LOC_FNAME
    when 'James'
      then update DEMOEMPL.EMPLOYEE set EMP_FNAME = 'Jim'
            where EMP_ID = P_EMP_ID;
           set MY_STATUS = 'James->JIM';
    when 'Jim'
      then update DEMOEMPL.EMPLOYEE set EMP_FNAME = 'James'
            where EMP_ID = P_EMP_ID;
           set MY_STATUS = 'Jim->James';
    when 'Thomas'
      then update DEMOEMPL.EMPLOYEE set EMP_FNAME = 'Thomas'
            where EMP_ID = P_EMP_ID;
           set MY_STATUS = 'Dummy update';
    else set MY_STATUS = 'No Changes';
  end case;
  return MY_STATUS;
end
++

select USER01.TCASE1('TCASE1', 1034)from SYSCA.SINGLETON_NULL++
*+
*+ USER_FUNC
*+ ---------
*+ Jim->James
```

The second example demonstrates the **searched-case-when-clause**. It is functionally equivalent with the example of **simple-case-when-clause**.

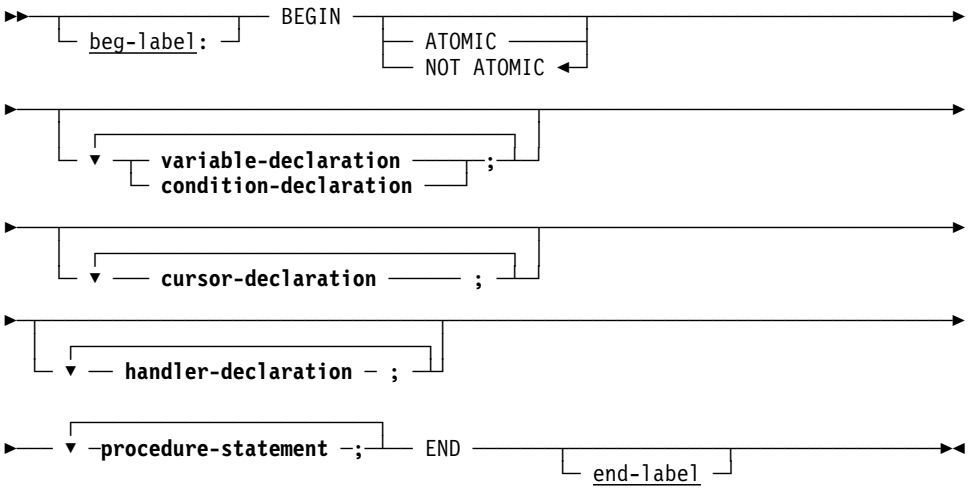
```
set options command delimiter '++';
create function USER01.TCASESR1
  ( TITLE     varchar(40) with default
  , P_EMP_ID  unsigned numeric(4)
  ) RETURNS   varchar(30)
    external name TCASESR1 language SQL
begin not atomic
  /*
  ** Function selects an employee with the given EMP_ID and
  ** does some conditional updates.
  ** Returns a message text with the outcome of the execution
  */
  declare MY_STATUS varchar(30);
  declare LOC_FNAME char(20) default ' ';
  declare LOC_LNAME char(20) default ' ';

  select EMP_FNAME, EMP_LNAME into LOC_FNAME, LOC_LNAME
    from DEMOEMPL.EMPLOYEE where EMP_ID = P_EMP_ID;


  case
    when LOC_FNAME = 'James'
      then update DEMOEMPL.EMPLOYEE set EMP_FNAME = 'Jim'
            Where EMP_ID = P_EMP_ID;
           set MY_STATUS = 'James->JIM';
    when LOC_FNAME = 'Jim' and LOC_LNAME = 'Gallway'
      then update DEMOEMPL.EMPLOYEE set EMP_FNAME = 'James'
            Where EMP_ID = P_EMP_ID;
```

```
          set MY_STATUS = 'Jim->James';
    when LOC_LNAME = 'Van der Bilck'
      then update DEMOEMPL.EMPLOYEE set EMP_LNAME = 'Vanderbilck'
            Where EMP_ID = P_EMP_ID;
          set MY_STATUS = 'Van der Bilck->Vanderbilck';
    else set MY_STATUS = 'No Changes';
  end case;

  return MY_STATUS;
end
++
```

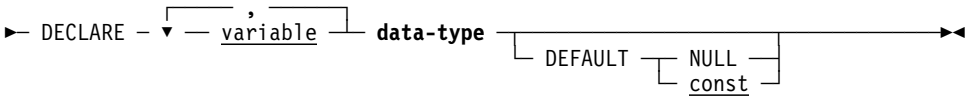
5.1.5.3 Compound Statement

The Compound statement defines a block of related SQL statements and can include the definition of local variables, condition names, cursors, and condition handlers.

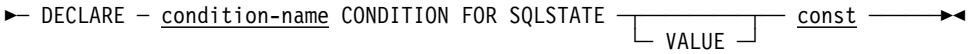
Syntax



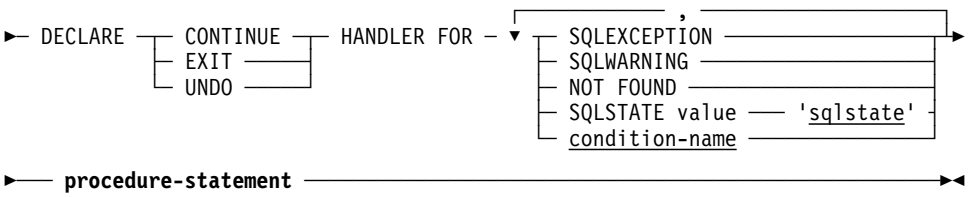
Expansion of variable-declaration



Expansion of condition-declaration



Expansion of handler-declaration



Parameters

beg-label:
Specifies a 1- through 32-character SQL identifier that labels the compound statement. The value must be different from any other label used in the compound statement.

ATOMIC
Specifies that an unhandled exception raised while executing the compound statement causes a rollback of the effects of the compound statement.

NOT ATOMIC
Specifies that an unhandled exception raised while executing the compound statement does not cause a rollback of the effects of the compound statement. This is the default.

variable-declaration
Defines a local variable.

variable
Specifies the name of the local variable. *Variable* must be a 1- through 32-character name that follows the conventions for SQL identifiers. The names of all local variables declared within a compound statement must be unique.

DEFAULT
Specifies the initial value of the local variable.

NULL
Initializes the local variable to NULL.

const
Initializes the local variable to the value of *const*. *Const* must be a literal whose value is compatible for assignment to the local variable.

Note: If DEFAULT is not specified, the local variable is not initialized.

DECLARE condition-name **FOR CONDITION SQLSTATE**
Defines a name for a condition. This name can be used in other statements to refer to the condition.

VALUE
Specifies an optional keyword without semantic meaning.

const

Specifies the value of SQLSTATE that constitutes the condition. **const** is a 5-character string-literal that consists of only digits (0-9) and capital alphabetic characters (A-Z). **const** can not be '00000', the value of SQLSTATE for sucessful completion.

cursor-declaration

Defines a local cursor for use within the compound statement. For a description of this clause, see 5.2.6, “DECLARE CURSOR” on page 168.

procedure-statement

Defines an SQL procedure statement to be included in the compound statement. *Procedure-statement* may be any statement defined by 5.1.3.3, “Expansion of procedure-statement” on page 115 except a compound statement.

handler-declaration

Defines a handler routine for SQL exception or completion conditions. A handler routine receives control when the execution of an SQL statement fails or terminates with a condition for which the handler has been defined. The three types of handlers (CONTINUE, EXIT, UNDO) and the conditions under which they are invoked are as follows:

CONTINUE

After executing the handler action, a CONTINUE handler returns control to the statement following the one that caused the event. If this statement is contained in an IF, CASE, LOOP, WHILE, or REPEAT statement, control is returned to the statement following the IF, CASE, LOOP, WHILE, or REPEAT statement.

EXIT

After executing the handler action, an EXIT handler returns control to the statement following the compound statement. If there is no statement following the compound statement, control is returned to the invoker of the routine.

UNDO

Before executing the handler action, an UNDO handler will rollback the database changes caused by the execution of the compound statement that caused the handler to be activated. After the handler actions have been executed, control is returned to the statement following the compound statement. If there is no statement after the compound statement, control is returned to the invoker of the routine. An UNDO handler requires its defining compound statement to be ATOMIC.

SQLEXCEPTION

Specifies that the handler is to be activated for all events except those of classes "Successful completion" (SQLSTATE = '00xxx'), "Completed with Warning" (SQLSTATE ='01xxx'), and "Completed with No Data" (SQLSTATE = '02xxx').

SQLWARNING
Specifies that the handler is to be activated for events of the class, "Completed with Warning" (SQLSTATE = '01xxx').

NOT FOUND
Specifies that the handler is to be activated for events of the class, "Completed with No Data" (SQLSTATE = '02xxx').

'sqlstate'
Specifies a value of SQLSTATE for which the handler is activated. *'Sqlstate'* must be a 5-character string-literal that consists of only digits (0-9) and capital alphabetic characters (A-Z). *'Sqlstate'* cannot be '00000', the value of SQLSTATE for successful completion.

condition-name
Specifies the name of a condition for which the handler is activated. *Condition-name* must identify a condition declared in the compound statement.

procedure-statement
Defines the SQL procedure statement that is to be executed when the handler routine is invoked.

end-label
Specifies an SQL identifier that labels the end of the compound statement. If specified, a *beg-label* must also have been specified and both labels must be equal.

Usage

Variables, Parameters, and Column Names: When ambiguity exists in referencing local variables, parameters and column names, qualification is required to resolve the ambiguity.

Note: For more information, see 5.1.3.5, “Expansion of Local-variable” on page 117 and 5.1.3.7, “Expansion of Routine-parameter” on page 119.

Nesting of Compound Statement: A compound statement cannot contain other compound statements with the exception of handlers. A handler, which necessarily is contained in a compound statement, can have a compound statement as its procedure statement *procedure-statement*.

Handlers: When both a generic class handler (a handler for SQLEXCEPTION or SQLWARNING) and a specific handler cover the same event, the more specific handler is invoked when the event occurs.

Only one handler for a specific event can be defined.

Handlers cannot be defined with duplicate conditions.

If an SQL exception occurs in a compound statement for which there is no handler defined, control returns to the statement following the compound

statement that caused the exception and an implicit RESIGNAL is executed. The exception is passed in the SQLSTATE. Database changes made by compound statements defined as ATOMIC will be rolled back before control returns.

Atomic Compound Statements: Compound statements defined as ATOMIC cannot contain the transaction management statements, COMMIT and ROLLBACK, or the session management statement, RELEASE.

Cursor state upon exiting from a compound statement: When execution of a compound statement ends, all cursors defined within the compound statement that are still open are automatically closed, except for returnable cursors. For more information about returnable cursors, see 5.2.6, “DECLARE CURSOR” on page 168.

Example

The procedure USER01.TCOMP01 retrieves an employee for a given EMP_ID and returns a formatted name. An exit handler for NOT FOUND handles the NOT FOUND condition. An exit handler for SQLEXCEPTION handles generic database errors.

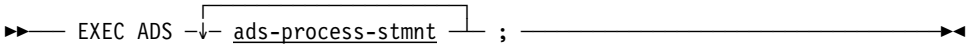
```
set options command delimiter '++';
create procedure USER01.TCOMP01
  ( P_ID     numeric(4)
  , P_NAME   char(30)
  , RESULT   varchar(30)
  )
    external name TCOMP01 language SQL

Label_400:
 /*
 ** Return formatted name of employee with given EMP_ID
 */
begin not atomic
  declare  L_FNAME   char(50);
  declare  L_LNAME   char(50);

  declare exit handler for SQLEXCEPTION
    label_8888:
      begin not atomic
        set RESULT = 'Unexpected SQLSTATE: ' || SQLSTATE;
        set P_NAME = '** Error **';
      end;

  declare exit handler for NOT FOUND
    set RESULT = 'No employee for EMP_ID: '
              || cast(P_ID as char(4));

  set RESULT = ' ';
  set P_NAME = ' ';

   select EMP_FNAME, EMP_LNAME into L_FNAME, L_LNAME
    from DEMOEMPL.EMPLOYEE
   where EMP_ID = P_ID;
```

```
  set P_NAME = trim(L_FNAME)  || ' ' || trim(L_LNAME);

  set RESULT = 'All OK';
End label_400
++

call user01.TCOMP01(1003);
**
**   P_ID  P_NAME                RESULT
**   ----  ------                ------
**   1003  Jim Baldwin           ALL OK

call user01.TCOMP01(9);
**
**   P_ID  P_NAME                RESULT
**   ----  ------                ------
**      9                        NO EMPLOYEE FOR EMP_ID: 9
call user01.TCOMP01(-2000);
**
**   P_ID  P_NAME                RESULT
**   ----  ------                ------
**  -2000  ** ERROR **           UNEXPECTED SQLSTATE: 22005
```

5.1.5.4 EXEC ADS

The EXEC ADS statement is a CA IDMS extension that enables inserting CA ADS code in SQL routines.

Syntax



Parameters

ads-process-stmnt
Specifies a CA ADS statement to be executed.

Usage

Allowable CA ADS statements: Only CA ADS statements that are allowed in a mapless dialog can be included in the body of an SQL routine.

Care should be taken in coding SQL transaction and session management statements because a ROLLBACK or COMMIT breaks the atomicity of a compound statement containing the EXEC ADS statement.

Referencing SQL-defined data: SQL-defined data can be referenced by respecting the mapping rules for identifiers and data types between SQL and CA ADS:

- Underscore characters are mapped to dashes.
- VARCHAR data are structures that start with a smallint field that holds the length of the character data, followed by the character data itself. The name of the structure is the mapped SQL identifier. The name of the length field is the mapped SQL identifier suffixed with "-LEN". The name of the data field is the mapped SQL identifier suffixed with "-TEXT".
- Nullable SQL data must have their NULL indicators managed properly. All SQL parameters and local variables are nullable.
- Date, time, and timestamp data types must be correctly processed.

Example

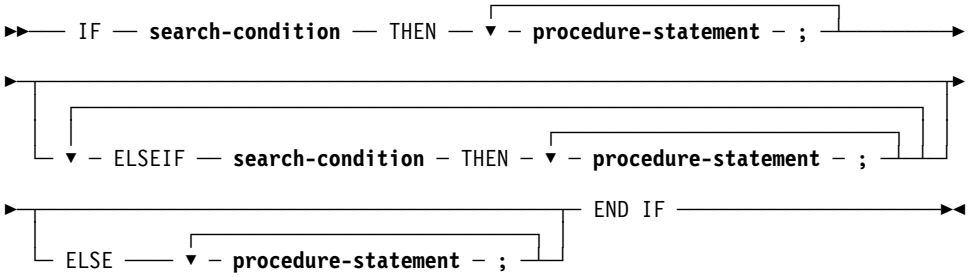
The SQL function USER01.TEXECADS2 returns the LTERM ID of the LTERM on which the function is being executed.

```
set options command delimiter '++';
create function USER01.TEXECADS2
  ( P_DUMMY   char(1)
  ) returns   char(8)
    external name TEXECAD2 language SQL
begin not atomic
 /*
 ** SQL Function to return LTERM ID using EXEC ADS
 */
 declare L_LTERMID char (8) default ' ';
 exec ads
          ACCEPT LTERM ID INTO L-LTERMID. ;
 return L_LTERMID;
end
++

select USER01.TEXECADS2()
  from SYSCA.SINGLETON_NULL;
*+
*+ USER_FUNC
*+ ---------
*+ VL71001
```

5.1.5.5 IF

The IF statement selects different execution paths depending on the evaluation of one or more truth value expressions, given as SQL search conditions.

Syntax



Parameters

IF search-condition
Specifies the truth value expression to be evaluated. The outcome of the evaluation determines the execution path.

THEN procedure-statement
Specifies the statements to be executed if the immediately preceding search condition is true.

ELSEIF search-condition
Specifies the truth value expression to be evaluated if the outcomes of all previously evaluated search conditions are false.

ELSE procedure-statement
Specifies the statements to be executed if all search conditions are false.

Usage

If no alternative execution path is given, execution continues with the next statement outside the IF.

Example

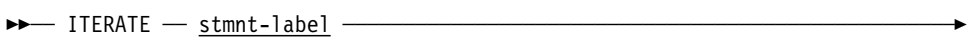
```
set options command delimiter '++';
create procedure USER01.TIF1
  ( TITLE   varchar(10) with default
  , P_LEFT    integer
  , P_RIGHT   real
  , RESULT  varchar(30)
  )
    EXTERNAL NAME TIF1 LANGUAGE SQL
Label_200:
begin not atomic
 /*
 ** Compare an integer value with a real value
 */
  if (P_LEFT > P_RIGHT)
```

```
    then set RESULT = 'p_left > p_right';
  elseif (P_LEFT = P_RIGHT)
    then set RESULT = 'p_left = p_right';
  elseif  (P_LEFT < P_RIGHT)
    then set RESULT = 'p_left < p_right';
  else   set RESULT = 'p_left and/or p_right NULL !';
  end if;
end
++
call user01.TIF1('Test IF >', 4, 2)++
**
** TITLE            P_LEFT         P_RIGHT  RESULT
** -----            ------         -------  ------
** Test IF >             4   2.0000000E+00  P_LEFT >  P_RIGHT
call user01.TIF1('Test IF <', 4, 9)++
**
** TITLE            P_LEFT         P_RIGHT  RESULT
** -----            ------         -------  ------
** Test IF <             4   9.0000000E+00  P_LEFT <  P_RIGHT

call user01.TIF1('Test IF =', 2, 2)++
**
** TITLE            P_LEFT         P_RIGHT  RESULT
** -----            ------         -------  ------
** Test IF =             2   2.0000000E+00  P_LEFT = P_RIGHT

call user01.TIF1('Test IF ', 4)++
**
** TITLE            P_LEFT         P_RIGHT  RESULT
** -----            ------         -------  ------
** Test IF               4          <null>  P_LEFT AND/OR P_RIGHT
NULL !
```

5.1.5.6 ITERATE

The ITERATE statement terminates execution of the current iteration of an iterated statement, such as LOOP, REPEAT or WHILE. If the iteration condition is true, a new iteration starts; otherwise, the statement following the iterated statement is executed.

Syntax



Parameters

stmnt-label

Specifies the begin label of the iterated statement.

Usage

Statements that may be iterated: The labeled statement referred in the ITERATE must be a LOOP, REPEAT, or WHILE statement that contains the ITERATE statement.

Example

The procedure USER01.TITERATE1 retrieves all rows of the DEMOEMPL.EMPLOYEE table three times. The first loop uses a WHILE, the second uses a REPEAT, and the third uses a LOOP statement.

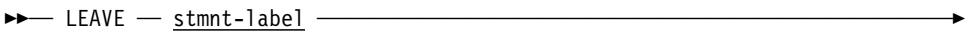
```
set options command delimiter '++';
create procedure USER01.TITERATE1
  ( TITLE     varchar(10) with default
  , P_FNAME   char(20)
  , P_COUNT     integer
  , RESULT    varchar(10)
  )
    EXTERNAL NAME TITERATE LANGUAGE SQL

Label_600:
begin not atomic
  declare FNAME   char(20);
  declare LNAME   varchar(20);
  declare EMP1 CURSOR FOR
        Select EMP_FNAME, EMP_LNAME
          From DEMOEMPL.EMPLOYEE;
 /*
 ITERATE in WHILE
 */
  set RESULT = '?????';
  open EMP1;

  while_loop:
  while (9 = 9)
     do
      fetch EMP1 into FNAME, LNAME;
      if (SQLSTATE = '00000')
        then
           set P_COUNT = P_COUNT + 1;
           iterate while_loop;
      end if;

      if (SQLSTATE = 'abcde')
        then
           iterate while_loop;
      end if;

      set RESULT = SQLSTATE;
      leave while_loop;
  end while while_loop;

  close EMP1;

 /*
 ITERATE in REPEAT
```

```
 */
  set RESULT = '?????';
  open EMP1;

  repeat_loop:
  repeat
      fetch EMP1 into FNAME, LNAME;
      if (SQLSTATE = '00000')
        then
           set P_COUNT = P_COUNT + 1;
           iterate repeat_loop;
      end if;

      set RESULT = SQLSTATE;
      leave repeat_loop;
  until (9 = 0)
  end repeat repeat_loop;

  close EMP1;
 /*
 ITERATE in LOOP
 */
  set RESULT = '?????';
  open EMP1;

  loop_loop:
  loop
      fetch EMP1 into FNAME, LNAME;
      if (SQLSTATE = '00000')
        then
           set P_COUNT = P_COUNT + 1;
           iterate loop_loop;
      end if;

      set RESULT = SQLSTATE;
      leave loop_loop;
  end loop loop_loop;

  close EMP1;
end
++

call USER01.TITERATE1('TITERATE1','James  ',0,'U')++
++
++ TITLE       P_FNAME                   P_COUNT  RESULT
++ -----       -------                   -------  ------
++ TITERATE1   James                         165  02000
```

5.1.5.7 LEAVE

The LEAVE statement continues execution with the statement that immediately follows the specified labeled statement.

Syntax



Parameters

stmnt-label

Specifies the begin label of a statement that contains the LEAVE statement, and identifies the statement that needs to be left.

Usage

Statements that may be left: The labeled statement referred in the LEAVE must be a LOOP, REPEAT, WHILE or compound statement that contains the LEAVE statement.

Example

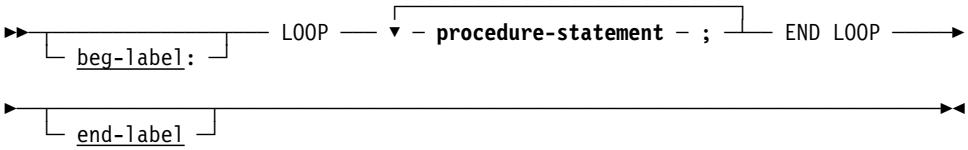
```
set options command delimiter '++';
create procedure USER01.TLEAVE1
  ( TITLE     varchar(10) with default
  , P_FNAME   char(20)
  , P_COUNT   integer
  , RESULT    varchar(25)
  )
    EXTERNAL NAME TLEAVE1 LANGUAGE SQL
Label_700:
 /*
 ** Count number of employees with equal Firstname
 */
begin not atomic
  declare FNAME   char(20);
  declare LNAME   varchar(20);
  declare EMP1 CURSOR FOR
    Select EMP_FNAME, EMP_LNAME
      From DEMOEMPL.EMPLOYEE
     where EMP_FNAME = P_FNAME;

  open EMP1;
  fetch EMP1 into FNAME, LNAME;
  fetching_loop:
  loop
    if (SQLSTATE < > '00000')
      then leave fetching_loop;
    end if;
    set P_COUNT = P_COUNT + 1;
    fetch EMP1 into FNAME, LNAME;
  end loop fetching_loop;

  set RESULT = 'SQLSTATE: ' || SQLSTATE;
```

```
  close EMP1;
end
++

call USER01.TLEAVE1('TLEAVE1','Martin',0)
**
** TITLE       P_FNAME                  P_COUNT  RESULT
** -----       -------                  -------  ------
** TLEAVE1     Martin                         3  SQLSTATE: 02000
```

5.1.5.8 LOOP

The LOOP statement repeats the execution of a statement or a group of statements.

Syntax



Parameters

beg-label:
Specifies a 1- through 32-character SQL identifier that labels the LOOP statement. The value must be different from any other label used in the compound statement if the LOOP statement is contained in a compound statement.

LOOP procedure-statement END LOOP
Specifies a statement or group of statements that are repeatedly executed.

end-label
Specifies an SQL identifier that labels the end of the LOOP statement. If specified, a *beg-label* must also have been specified and both labels must be equal.

Usage

How execution of a LOOP statement ends: To end the repeated execution of the *procedure-statement*s contained in a LOOP statement, a LEAVE statement can be used or an exit handler can be driven.

Example

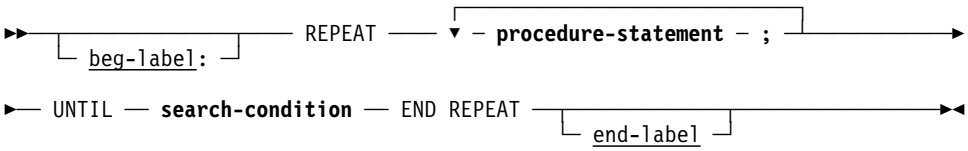
See the example for the LEAVE statement. The procedure USER01.TLOOP1, is similar to USER01.TLEAVE1 but it uses an exit handler to terminate the LOOP.

```
set options command delimiter '++';
create procedure USER01.TLOOP1
  ( TITLE     varchar(10) with default
  , P_FNAME   char(20)
  , P_COUNT   integer
  , RESULT    varchar(30)
  )
    EXTERNAL NAME TLOOP1 LANGUAGE SQL
Label_700:
 /*
 ** Count number of employees with equal Firstname
 */
begin not atomic
  declare FNAME   char(20);
  declare LNAME   varchar(20);
  declare EMP1 CURSOR FOR
        Select EMP_FNAME, EMP_LNAME
          From DEMOEMPL.EMPLOYEE
         where EMP_FNAME = P_FNAME;
  declare exit handler for SQLEXCEPTION, SQLWARNING, NOT FOUND
      set RESULT = 'SQLSTATE: ' || SQLSTATE;

 /*
 ** Count number of employees with equal Firstname
 */
  open EMP1;
  fetch EMP1 into FNAME, LNAME;

  fetching_loop:
  loop
      set P_COUNT = P_COUNT + 1;
      fetch EMP1 into FNAME, LNAME;
    end loop fetching_loop;
end
++


call USER01.TLOOP1('TLOOP1','Martin ',0,'U')
*+
*+ TITLE       P_FNAME                   P_COUNT
*+ -----       -------                   -------
*+ TLOOP1      Martin                          3
*+
*+ RESULT
*+ ------
*+ SQLSTATE: 02000
```

5.1.5.9 REPEAT

The REPEAT statement repeats the execution of a statement or a group of statements until a condition is met.

Syntax



Parameters

beg-label:
Specifies a 1- through 32-character SQL identifier that labels the REPEAT statement. The value must be different from any other label used in the compound statement if the REPEAT statement is contained in a compound statement.

REPEAT procedure-statement
Specifies the statement or group of statements that are repeatedly executed.

UNTIL search-condition
Specifies the search condition that is evaluated after each iteration. If the outcome is true, the statement following the REPEAT statement is executed. Otherwise, a new iteration starts.

end-label
Specifies an SQL identifier that labels the end of the REPEAT statement. If specified, a *beg-label* must also have been specified and both labels must be equal.

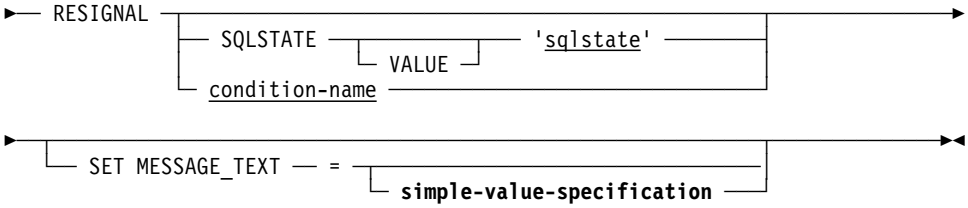
Example

```
set options command delimiter '++';
create procedure USER01.TREPEAT1
  ( TITLE     varchar(10) with default
  , P_FNAME   char(20)
  , P_COUNT   integer
  , RESULT    varchar(25)
  )
    EXTERNAL NAME TREPEAT1 LANGUAGE SQL
Label_700:
 /*
 ** Count number of employees with equal First name using REPEAT
 */
begin not atomic
  declare FNAME   char(20);
  declare LNAME   varchar(20);
  declare EMP1 CURSOR FOR
        Select EMP_FNAME, EMP_LNAME
          From DEMOEMPL.EMPLOYEE
         where EMP_FNAME = P_FNAME;

  open EMP1;

  fetching_loop:
  repeat
    fetch EMP1 into FNAME, LNAME;

    if (SQLSTATE = '00000')
      then set P_COUNT = P_COUNT + 1;
    end if;
  until SQLSTATE < > '00000'
  end repeat fetching_loop;
  set RESULT = 'SQLSTATE: ' || SQLSTATE;

  close EMP1;
end
++

call USER01.TREPEAT1('TREPEAT1','Martin',0,'U')
++
** TITLE       P_FNAME                   P_COUNT  RESULT
** -----       -------                   -------  ------
** TREPEAT1    Martin                          3  SQLSTATE: 02000
```

5.1.5.10 RESIGNAL

The RESIGNAL statement resignals an SQL event or exception condition in a handler for the next higher level scope.

Syntax



Parameters

condition-name

Specifies the name of a condition whose SQLSTATE value is to be resignaled. *Condition-name* must identify a condition defined by a *condition-declaration* in a compound-statement containing the RESIGNAL statement. if more than one such *condition-declaration* has the specified *condition-name*, the one with the innermost scope is raised.

'sqlstate'

Specifies the value for SQLSTATE that is to be resignaled. *'Sqlstate'* is a 5-character string-literal that consists of only digits (0-9) and capital alphabetic characters (A-Z). *'Sqlstate'* cannot be '00000', the value of SQLSTATE for successful completion.

simple-value-specification

Specifies a character value to be added to the information item MESSAGE-TEXT. The data type of **simple-value-specification** must be a character.

Usage

Propagating the SQL Condition: The RESIGNAL statement can only be used in a handler to propagate an SQL condition to the scope that encloses the exception handler's scope. If the RESIGNAL is issued in a handler of a top level compound statement, control returns to the invoker of the SQL-invoked routine.

FLOW of CONTROL: If in the outer scope a handler exists for the raised exception or SQL event, the handler acquires control. After execution of the handler, control returns as with any other statement that causes a handler to activate.

SQLSTATE: There are no restrictions on the values that can be set for SQLSTATE, other than compliance with the syntactic rules for SQLSTATE values. We recommend using values in accordance with the classification of SQLSTATE values.

Note: For more information, see SQLSTATE Values in the "SQL Communication Area" appendix in the *CA IDMS SQL Reference Guide*.

MESSAGE_TEXT: This is an information item of CHAR type with no defined maximum length.

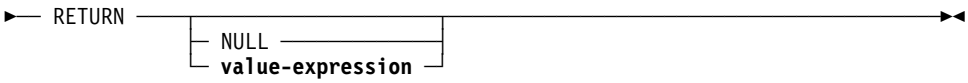
Example

```
set options command delimiter '++';
drop procedure USER01.RESIGNAL1++
commit++
create procedure USER01.RESIGNAL1
  ( TITLE     varchar(10) with default
  , RESULT    varchar(120)
  )
    EXTERNAL NAME RESIGNA1 LANGUAGE SQL
Label_400:
 /*
 ** Resignal show case
 */

begin not atomic
  declare DEAD_LOCK condition for SQLSTATE '12000';
  declare NOT_FOUND condition for SQLSTATE '02000';
  declare exit handler for NOT FOUND
    begin not atomic
        set RESULT = RESULT || ' Not Found';
        resignal SQLSTATE '38607';
    end;

  set RESULT = 'Signal trace:';
  signal NOT_FOUND;

end label_400
++

call user01.resignal1('Signal')

*+ Status = -4       SQLSTATE = 38000        Messages follow:
*+ DB001075 C-4M321: Procedure RESIGNA1 exception 38607
```

5.1.5.11 RETURN

The RETURN statement returns a value for an SQL function. As an extension to the ISO standard, a RETURN without parameters can also be used to exit a compound statement.

Syntax



Parameters

NULL
Specifies that the function return value is NULL.

value-expression
Specifies the function return value.

Usage

Compatible Data Types: The data type of the *value-expression* and the data type of the function return value named in the CREATE FUNCTION statement must be compatible for assignment.

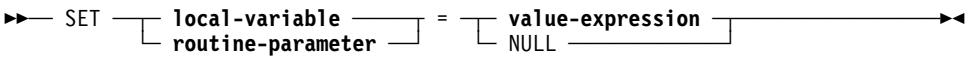
Example

See 5.1.4.3, “CREATE FUNCTION” on page 124.

5.1.5.12 SET Assignment

The SET Assignment statement assigns values to parameters and variables used in SQL routines.

Syntax



Parameters

local-variable
Identifies the local variable that is the target of the SET assignment statement. **local-variable** must be the name of a local variable defined within the compound statement containing SET statement.

routine-parameter
Identifies the SQL routine parameter that is the target of the SET assignment statement. **Routine-parameter** must be the name of a parameter of the routine containing the SET assignment statement.

value-expression
Specifies the value to be assigned to the target of the SET assignment statement.

NULL
Specifies that the null value is to be assigned to the target of the SET assignment statement.

Usage

Valid assignments: The rules for assignment are provided in Comparison, Assignment, Arithmetic, and Concatenation Operations in the "Data Types and Null Values" chapter in the *CA IDMS SQL Reference Guide*.

Example

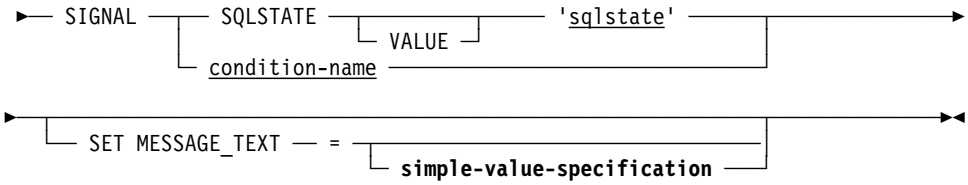
The procedure TSET3 creates a combined, edited name from a given first and last name. If the first or last name is null, or if the length of the last name is 0, the null value is returned for the edited name.

```
set options command delimiter '++';
create procedure SQLROUT.TSET3
  ( P_FNAME   varchar(20)
  , P_LNAME   varchar(20)
  , P _NAME   varchar(41)
  )
    EXTERNAL NAME TSET3 LANGUAGE SQL
 /*
 ** Return an edited name from the given Firstname and Lastname
 */
 if (LENGTH(P_LNAME) <= 0)
   then set P_NAME = null;
   else set P_NAME = trim(P_FNAME)  || ' ' || trim(P_NLNAME) ;
 end if
++
set options command delimiter default++

call SQLROUT.TSET3('James    ', 'Last   ');
*+
*+ P_FNAME               P_LNAME
*+ -------               -------
*+ James                 Last
*+
*+ P_NAME
*+ ------
*+ James Last

call SQLROUT.TSET3('James    ', '');
*+
*+ P_FNAME               P_LNAME
*+ -------               -------
*+ James
*+
*+ P_NAME
*+ ------
*+ <null>
```

5.1.5.13 SIGNAL

The SIGNAL statement raises and signals an SQL event or exception condition.

Syntax



Parameters

condition-name
Specifies the name of a condition whose SQLSTATE value is to be signaled. *Condition-name* must identify a condition defined by a condition declaration in a compound statement containing the SIGNAL statement. If more than one such condition declaration has the specified condition name, the one with the innermost scope is raised.

'sqlstate'
Specifies the value for SQLSTATE that is to be signaled. *'sqlstate'* is a 5-character string-literal value that consists of only digits (0-9) and capital alphabetic characters (A-Z). *'Sqlstate'* cannot be '00000', the value of SQLSTATE for successful completion.

simple-value-specification
Specifies a character value to be added to the information item MESSAGE-TEXT. The data type of **simple-value-specification** must be a character.

Usage

FLOW of CONTROL: If a handler exists for the raised exception or SQL event, the handler acquires control. After execution of the handler, control returns as with any other statement that causes activation of a handler.

If no handler is activated, control goes to the end of the compound statement that contains the signal. If the signal is not in a compound statement of an exit handler, control returns to the invoker of the SQL routine. Otherwise, it returns to the statement after the SIGNAL statement, just as if a continue handler had been activated.

SQLSTATE: There are no restrictions on the values that can be set for SQLSTATE, other than compliance with the syntactic rules for SQLSTATE values. We recommend that values are used in accordance with the classification of SQLSTATE values. See SQLSTATE Values in the "SQL Communication Area" appendix in the *CA IDMS SQL Reference Guide*.

MESSAGE_TEXT: This is an information item of CHAR type with no defined maximum length.

Example

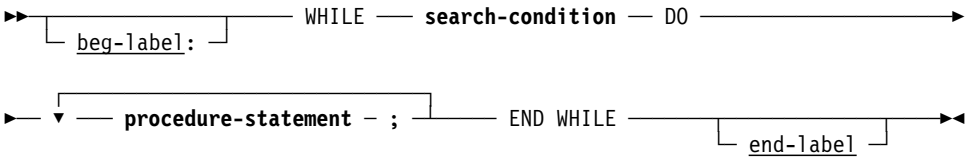
```
set options command delimiter '++';
create procedure USER01.TSIGNAL5
  ( TITLE     varchar(10) with default
  , RESULT    varchar(120)
  )
    EXTERNAL NAME TSIGNAL5 LANGUAGE SQL
Label_400:
 /*
 ** Trace execution of consecutive signal statements
 */

begin not atomic
  declare DEAD_LOCK condition for SQLSTATE '12000';
  declare NOT_FOUND condition for SQLSTATE '02000';

  declare continue HANDLER for SQLWARNING
    LABEL_9999:
      begin not atomic
        set RESULT = RESULT || ' Sqlwarning';
      end;
  declare continue handler for SQLEXCEPTION
    Label_8888:
      begin not atomic
        set RESULT = RESULT || ' Sqlexception';
      end;
  declare continue handler for SQLSTATE '23800'
        set RESULT = RESULT || ' 23800';
  declare continue handler for DEAD_LOCK
    LABEL_6666:
      begin not atomic
        set RESULT = RESULT || ' Deadlocked';
      end;
  declare continue handler for NOT FOUND
        set RESULT = RESULT || ' Not Found';
  set RESULT = 'Signal trace:';

  signal SQLSTATE '23800';
  signal NOT_FOUND;
  signal SQLSTATE '01200';
  signal SQLSTATE '72300';
  signal DEAD_LOCK;

end label_400
++
call user01.tsignal5('Signal')
```

```
**
** TITLE
** -----
** Signal
**
**
** RESULT
** ------
** Signal trace: 23800 Not Found Sqlwarning Sqlexception
Deadlocked
**
```

5.1.5.14 WHILE

The WHILE statement repeats the execution of a statement or a group of statements while a condition is met.

Syntax



Parameters

beg-label:
Specifies a 1- through 32-character SQL identifier that labels the WHILE statement. The value must be different from any other label used in the compound statement if the WHILE statement is contained in a compound statement.

WHILE search-condition
Specifies the search condition to be evaluated. If the outcome is false, the statement after the WHILE statement is executed. Otherwise, an iteration of the group of statements enclosed by DO and END WHILE is started.

DO procedure-statement END WHILE
Specifies the statement or group of statements that are repeatedly executed.

end-label
Specifies an SQL identifier that labels the end of the WHILE statement. If specified, a *beg-label* must also have been specified and both labels must be equal.

Example

```
set options command delimiter '++';
create procedure USER01.TWHILE2
  ( TITLE     varchar(10) with default
  , P_FNAME   char(20)
  , P_COUNT   integer
  )
    EXTERNAL NAME TWHILE2 LANGUAGE SQL
Label_700:
begin not atomic
 /*
 ** Count number of employees with equal first name
 */
  declare FNAME   char(20);
  declare LNAME   varchar(20);
  declare EMP1 CURSOR FOR
        Select EMP_FNAME, EMP_LNAME
          From DEMOEMPL.EMPLOYEE
         where EMP_FNAME = P_FNAME;
  set P_COUNT = 0;
  open EMP1;
  fetch EMP1 into FNAME, LNAME;
  fetching_loop_non_SQL:
  while (SQLSTATE = '00000')
    do
      set P_COUNT = P_COUNT + 1;
      fetch EMP1 into FNAME, LN&AME;
    end while fetching_loop_non_SQL;

  close EMP1;
end
++

call USER01.TWHILE2('TWHILE2','Martin  ')
;
**
** TITLE       P_FNAME                   P_COUNT
** -----       -------                   -------
** TWHILE2     Martin                          3
```

5.2 Result Sets from SQL-invoked Procedures

An SQL-invoked procedure can return results to the caller by assigning values to one or more parameters of the procedure. With r17, it is now possible for an SQL-invoked procedure to return result sets in the form of rows of result tables.

To exploit result sets returned by an SQL-invoked procedure, an application must consist of at least an SQL-invoked procedure and a caller of that procedure. The caller can be an SQL client program or another SQL-invoked routine. The SQL-invoked procedure that returns the result sets can be an external procedure (Cobol, PL/I, Assembler or CA ADS) or an internal SQL-invoked procedure written in SQL.

For an SQL-invoked procedure to return result sets to its caller, it must be defined with a positive integer value for the new *Dynamic Result Sets* attribute.

A cursor declared or dynamically allocated in the SQL-invoked procedure becomes a potential returned result set if its definition contains *With Return* as the value for the new returnability attribute. Such a cursor is called a **returnable cursor**. It becomes a returned result set if it is in the open state when the SQL-invoked procedure terminates.

An SQL-invoked procedure can return multiple result sets up to the number specified by the Dynamic Result Sets attribute of the procedure. The list of returned result sets are sequenced in the order of the open of the cursors. If the procedure starts multiple sessions, then returned result sets are grouped by session and the sessions are sequenced in the order of the connects. After a procedure CALL, the new SQLCA field SQLCNRRS contains the number of result sets returned by the procedure.

The caller of an SQL-invoked procedure accesses returned result sets by allocating a dynamic cursor and associating it with the procedure through an ALLOCATE CURSOR FOR PROCEDURE statement. Such a cursor is called a **received cursor**.

A successful ALLOCATE CURSOR FOR PROCEDURE statement associates the received cursor with the first result set from the sequence of returned result sets and places the cursor in the open state. The cursor position is the same as it was when the SQL-invoked procedure terminated and the associated returned result set is removed from the list of returned result sets.

The caller of the procedure can access the next in the sequence of returned result sets by either allocating another cursor for the procedure or by closing the previously allocated received cursor. If the close is successful and the list of remaining returned result sets is not empty, the received cursor is automatically placed in the open state and associated with the result set that is now first in the list. The newly associated result set is also removed from the list. This process can be repeated until the list of returned result sets is empty.

A new invocation of the SQL-invoked procedure automatically destroys all the returned result sets from the previous invocation.

The received cursors, allocated by the caller and associated with returned result sets, are necessarily dynamic. Unless the program knows the returned columns and their data type, a DESCRIBE CURSOR statement is needed to retrieve the description of the returned result set in an SQL descriptor area (SQLDA).

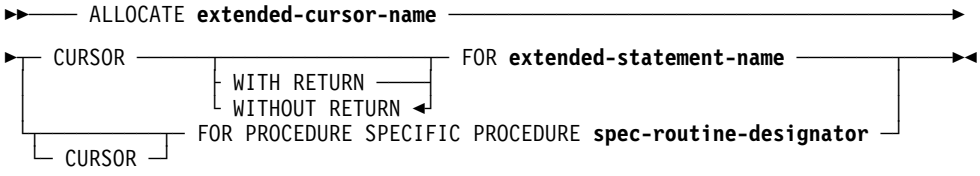
Only the immediate caller of an SQL-invoked procedure can process returned result sets. There is no mechanism for the caller to return returned result sets to its caller.

5.2.1 ALLOCATE CURSOR

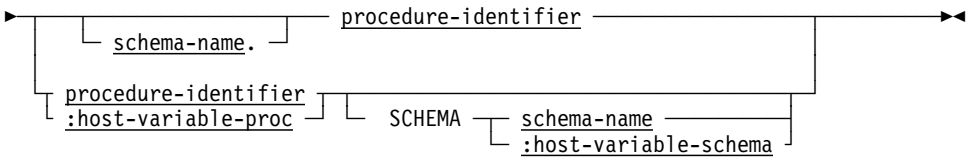
The ALLOCATE CURSOR statement has been enhanced with the FOR PROCEDURE and WITH/WITHOUT RETURN clauses.

The ALLOCATE CURSOR statement defines a cursor for a dynamically-prepared statement or for a result set returned from a previously invoked procedure.

Syntax



Expansion of spec-routine-designator



Parameters

WITH RETURN
Defines the cursor as a returnable cursor. If a returnable cursor is allocated in an SQL-invoked procedure and is in the open state when the procedure terminates, a result set is returned to the caller.

WITHOUT RETURN
Specifies that the cursor is not a returnable cursor. This is the default.

FOR PROCEDURE SPECIFIC PROCEDURE
Specifies that the cursor is to be allocated for a result set returned by the invocation of the identified procedure. This type of cursor is called a received cursor.

spec-routine-designator
Identifies the SQL-invoked procedure.

schema-name
Specifies the schema with which the procedure identified by *procedure-identifier* is associated.

procedure-identifier
Identifies a procedure defined in the dictionary.

:host-variable-proc
Identifies a host variable, routine parameter, or local variable containing the name of the previously invoked procedure.

If *:host-variable-proc* is a routine parameter or local variable, the colon must not be coded.

SCHEMA
Qualifies the procedure name with the name of the schema with which it is associated. This option is an extension to the ISO standard.

schema-name
Specifies the schema with which the procedure identified by *procedure-identifier* or *host-variable-proc* is associated.

:host-variable-schema
Identifies a host variable, routine parameter, or local variable containing the name of the schema with which the procedure identified by *procedure-identifier* or *host-variable-proc* is associated.

If *:host-variable-schema* is a routine parameter or local variable, the colon must not be coded.

Note: For more information about using a schema name to qualify a procedure, see Identifying Entities in Schemas in the "Identifiers" chapter in the *CA IDMS SQL Reference Guide*.

Usage

Allocating a Received Cursor for a Result Set: If the ALLOCATE statement is used for a result set, then the procedure identified by **spec-routine-designator** must have been previously invoked by an SQL CALL or SELECT statement in the same transaction as that in which the ALLOCATE CURSOR statement is executed.

The result sets that the SQL-invoked procedure returns, form a list ordered in the sequence in which the cursors were opened by the procedure. When a received cursor is allocated, the following actions are taken:

- the new cursor is associated with the first result set in the list of returned result sets
- the result set is removed from the list
- the cursor is placed in the open state

- the cursor is positioned at the same point at which the corresponding returnable cursor was left by the procedure

If an SQL-invoked procedure has started multiple sessions, the sequence of returned result sets is by session, in the order in which the sessions were connected. Within each session, the result sets are sequenced by the order in which their cursors were opened.

A received cursor cannot be used to return a result set nor can it be referenced in a positioned update or delete statement.

Note: For more information, see 5.2.7, “DESCRIBE CURSOR” on page 169.

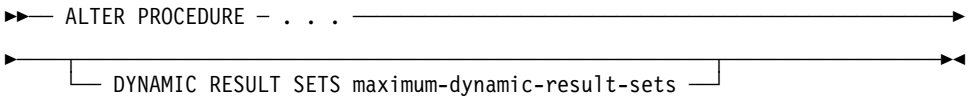
Example

```
exec sql
      call GET_EMPLOYEE_INFO(1003)
end-exec
exec sql
      allocate 'RECEIVED_CURSOR_GET_EMPG' for procedure specific
      procedure GET_EMPLOYEE_INFO
end-exec
```

5.2.2 ALTER PROCEDURE

This is an existing statement that is enhanced with the DYNAMIC RESULT SETS clause. Using the ALTER PROCEDURE statement you can change the number of dynamic result sets that a procedure can return to the caller.

Syntax



Parameters

DYNAMIC RESULT SETS
Defines the maximum number of result sets that a procedure invocation can return to its caller. A result set is a sequence of rows specified by a *cursor-specification*, created by the opening of a cursor and ranged over that cursor.

maximum-dynamic-result-sets
Defines an integer in the range 0-32767 specifying the maximum number of result sets a procedure can return.

Note: For more information, see 5.2.5, “CREATE PROCEDURE” on page 167.

5.2.3 CALL

The CALL statement has been enhanced to provide a warning if the procedure returns more result sets than specified in the DYNAMIC RESULT SETS attribute of the procedure.

Note: For a comprehensive example illustrating the basic coding techniques to use dynamic result sets in an application, see the CALL statement in the "Statements" chapter in the *CA IDMS SQL Reference Guide*.

Usage

Calling an SQL-invoked Procedure Returning Result Sets: After a CALL of an SQL-invoked procedure that has been defined with a positive value for the Dynamic Result Sets attribute the number of actual returned results sets is available in the field SQLCNRRS of the SQLCA. The number of returned result sets can also be determined by issuing a GET DIAGNOSTICS statement to retrieve the IDMS_RETURNED_RESULT_SETS information item.

The successful execution of a CALL statement may result in one of two warning conditions:

0100C SQL invoked procedure returned result sets
Indicates that the number of result sets returned by the procedure is less than or equal to the value of the procedure's DYNAMIC RESULT SETS attribute.

0100E Attempt to return too many result sets
Indicates that the procedure attempted to return more result sets than permitted by its DYNAMIC RESULT SETS attribute. The actual number of result sets is reduced to the value of the DYNAMIC RESULT SETS attribute.

A call of a procedure destroys any result sets left over from a previous invocation of the same procedure.

Note: For more information, see 5.2.1, “ALLOCATE CURSOR” on page 162.

5.2.4 CLOSE CURSOR

The CLOSE statement has been enhanced to associate a received cursor with the next in the sequence of result sets returned by an SQL-invoked procedure.

The CLOSE statement places a specified cursor in the closed state or disassociates a received cursor from the current returned result set and associates it with the next result set returned by the procedure. Use this statement only in SQL that is embedded in a program.

Usage

Closing a Received Cursor: A received cursor is a dynamically allocated cursor used to process one or more result sets returned by an SQL-invoked procedure. Returned result sets are maintained in an ordered list. An ALLOCATE CURSOR statement associates the cursor with the first result set in the list and removes it from the list.

If the list of returned result sets is not empty when a received cursor is closed, the CLOSE statement causes the following actions to be taken:

- disassociates the cursor from its current result set
- associates the cursor with the first result set in the list of returned result sets
- removes the result set from the list
- positions the cursor at the same point at which the corresponding returnable cursor was left by the procedure
- returns a warning "Additional result sets returned" (SQLSTATE "0100D)

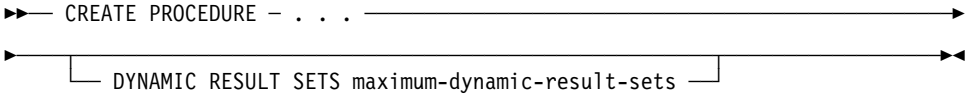
Closing the cursor associated with the last result set of a session started by the called procedure, releases that session.

Note: For more information, see the FOR PROCEDURE clause in 5.2.1, “ALLOCATE CURSOR” on page 162.

5.2.5 CREATE PROCEDURE

The CREATE PROCEDURE statement has been enhanced with the DYNAMIC RESULT SETS clause.

Syntax



Parameters

DYNAMIC RESULT SETS

Defines the maximum number of result sets that a procedure invocation can return to its caller. A result set is a sequence of rows specified by a *cursor-specification*, created by the opening of a cursor and ranged over that cursor.

maximum-dynamic-result-sets

Defines an integer in the range 0-32767 specifying the maximum number of result sets a procedure can return. The default is 0.

Usage

Dynamic Result Sets: An SQL invoked procedure can return one or more result sets to its caller, up to the maximum number specified by its dynamic result sets attribute. A result set is returned for each returnable cursor that is still open when the procedure returns control to its caller.

For information on defining a returnable cursor, see 5.2.1, “ALLOCATE CURSOR” on page 162 or 5.2.6, “DECLARE CURSOR” on page 168. For information on how the caller processes the returned result sets, see 5.2.1, “ALLOCATE CURSOR” on page 162.

Example

The GET_EMPLOYEE_INFO procedure uses the given employee ID, to construct two result set cursors:

- A static declared cursor RET_COVERAGE returns a cursor with the data from the COVERAGE table.
- The allocated dynamic cursor RET_BENEFITS to return the data from the BENEFITS data.

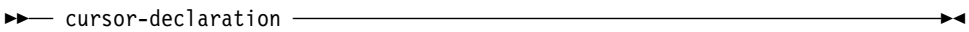
```
set options command delimiter '++';
create procedure SQLROUTE.GET_EMPLOYEE_INFO
  ( TITLE       varchar(10) with default
  , P_EMP_ID    numeric(4)
  , RESULT      varchar(20)
  )
    EXTERNAL NAME GETEMPIN LANGUAGE SQL
    DYNAMIC RESULT SETS 2
begin not atomic
  declare STMNT_NAME    char(10) default 'DYN_STMNT1';
  declare STMNT_BUF     char(80) default ' ';
  declare RET_COVERAGE cursor with return for
    select * from DEMOEMPL.COVERAGE
     where EMP_ID = P_EMP_ID;
  open RET_COVERAGE;
  set STMNT_BUF = 'select * from DEMOEMPL.BENEFITS'
                || 'where EMP_ID = ' || P_EMP_ID;
  prepare STMT_NAME from STMT_BUF;
  allocate 'RET_BENEFITS' cursor with return for STMT_NAME;
  open 'RET_BENEFITS';
  set RESULT = '2 returned result sets';
end
set options command delimiter default ++
```

5.2.6 DECLARE CURSOR

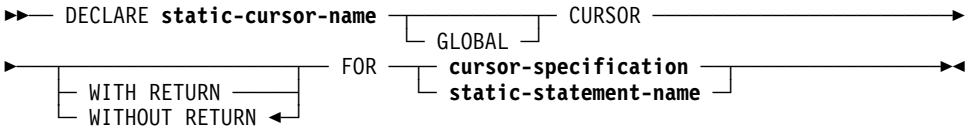
The DECLARE CURSOR statement has been enhanced with the cursor returnability characteristic.

The DECLARE CURSOR statement is a data manipulation statement that defines a cursor for a specified result table. Use this statement only in SQL that is embedded in a program.

Syntax



Expansion of cursor-declaration



Parameters

WITH RETURN

Defines the cursor as a returnable cursor. If a returnable cursor is declared in an SQL-invoked procedure and is in the open state when the procedure returns to its caller, a result set is returned to the caller.

WITHOUT RETURN

Specifies that the cursor is not a returnable cursor. This is the default.

Usage

Defining Returnable Cursors: While any cursor can be defined as a returnable cursor using WITH RETURN, it only makes sense to do so in programs that are invoked as SQL-invoked procedures and that are defined with a non-zero dynamic result set attribute.

The invoker must use the ALLOCATE CURSOR statement to associate returned result sets with received cursors for further processing. For more information about how the caller processes returned result sets, see 5.2.1, “ALLOCATE CURSOR” on page 162.

Example

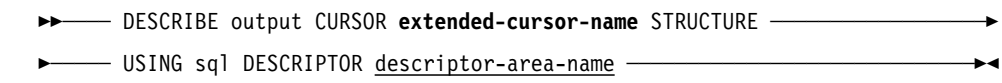
The following DECLARE CURSOR statement is specified in an SQL-invoked procedure written in SQL. The cursor RET_COVERAGE returns a result set consisting of the rows of the table DEMOEMPL.COVERAGE for which the column EMP_ID equals the value of the parameter P_EMP_ID. To effectively return the result set, the cursor must be left open on the return from the procedure.

```
declare RET_COVERAGE cursor with return for
    select * from DEMOEMPL.COVERAGE
     where EMP_ID = P_EMP_ID;
```

5.2.7 DESCRIBE CURSOR

The DESCRIBE CURSOR is a new data manipulation statement that describes the result set associated with a received cursor. It directs CA IDMS to return information about the result set associated with a received cursor into an SQL descriptor area. Use this statement only in SQL that is embedded in a program.

Syntax



Parameters

extended-cursor-name
Specifies the name of the cursor whose result set is to be described. The cursor must have been previously associated with a returned result set using the ALLOCATE CURSOR statement.

USING sql DESCRIPTOR
Specifies the SQL descriptor area where CA IDMS is to return information about the result set with which the cursor is associated.

descriptor-area-name
Directs CA IDMS to use the named area as the descriptor area. *descriptor-area-name* must identify an SQL descriptor area.

Example

The GET_EMPLOYEE_INFO procedure returns two result sets for a given EMP_ID:

- One for COVERAGE info
- One for BENEFITS info

Note: For more information about how to define this procedure, see the examples in 5.2.5, “CREATE PROCEDURE” on page 167.

* Invocation of the procedure GET_EMPLOYEE_INFO.

```
exec sql
      call GET_EMPLOYEE_INFO(1003)
end-exec
```

* The dynamic cursor 'RECEIVED_CURSOR' is associated with the first result set.

* The received cursor is in the open state.

```
exec sql
      allocate 'RECEIVED_CURSOR' for procedure specific procedure
      GET_EMPLOYEE_INFO
end-exec
```

* The 'RECEIVED_CURSOR' cursor info is being described.

```
exec sql
      describe cursor 'RECEIVED_CURSOR' structure
      using sql descriptor SQLDA-AREA
end-exec
```

* The COVERAGE info is being processed.

* The statement is executed in a loop until the SQLSTATE indicates *NO MORE DATA*.

```
 exec sql
       fetch 'RECEIVED_CURSOR' into :BUFFER-COVER
       using descriptor SQLDA-AREA
 end-exec
```

* The dynamic cursor 'RECEIVED_CURSOR' is associated with the second result set.

* The received cursor is in the open state.

```
 exec sql
       close 'RECEIVED_CURSOR'
 end-exec
 . .
```

* The 'RECEIVED_CURSOR' info is being described.

```
exec sql
      describe cursor 'RECEIVED_CURSOR' structure
      using sql descriptor SQLDA-AREA
end-exec
. . .
```

* The BENEFITS info is being processed.

* The statement is executed in a loop until the SQLSTATE indicates *NO MORE DATA*.

```
exec sql
      fetch 'RECEIVED_CURSOR' into :BUFFER-BENEF
      using descriptor SQLDA-AREA
end-exec
```

* Close the cursor.

```
exec sql
      close 'RECEIVED_CURSOR'
end-exec
```

5.2.8 SQL Communication Area

SQLCA

After a CALL or SELECT of a procedure, the new SQLCNRRS field holds the actual number of result sets that an SQL-invoked procedure returned.

SQLSTATE

The following new SQLSTATE values are associated with processing returned result sets.

SQLSTATE	Description
0100C	SQL-invoked procedure returned result sets
0100D	Additional result sets returned
0100E	Attempt to return too many result sets

5.2.9 Catalog Extensions

The SYSTEM.TABLE and SYSTEM.SYNTAX catalog tables have been enhanced.

SYSTEM.TABLE

TYPE Column

The new type value of 'L' identifies this as an internal table whose columns represent the local variables of an SQL routine.

LANGUAGE Column

The new LANGUAGE column identifies the language in which the SQL routine is written. The data type of this column is CHAR(3) and its allowable values are: ADS, ASM, COB, PLI, or SQL.

DYNRESULTSETS column

The new DYNRESULTSETS column specifies the maximum number of result sets that can be returned by a procedure. The data type of this column is SMALLINT and its allowable values range from 0-32767.

SYSTEM.SYNTAX

TYPE Column

The new type value of 'S' identifies this as a row of text contained in the body of an SQL routine.

5.3 Enhanced Diagnostics and Statistics

The new SQL Diagnostic statements category is used for diagnosing the execution of SQL statements and for returning statistical information for the current transaction.

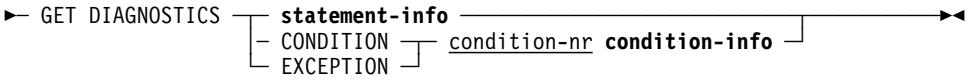
Note: These statements can be used as embedded SQL, including embedding in an SQL-invoked routine. The GET STATISTICS statement can also be used in the command facility and the CA IDMS Visual DBA command console.

Statement	Purpose
GET DIAGNOSTICS	Diagnoses the execution of the last executed SQL statement.
GET STATISTICS	Returns statistical information for the current transaction.

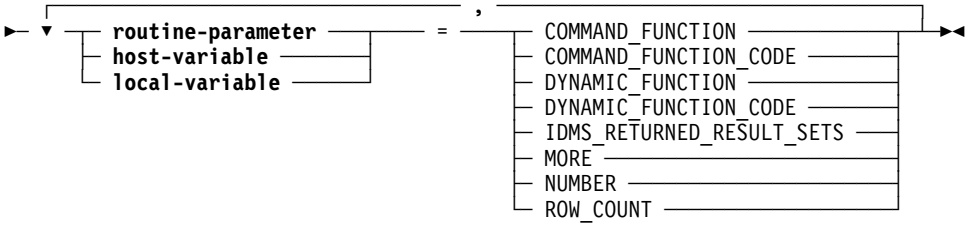
5.3.1 GET DIAGNOSTICS

The GET DIAGNOSTICS statement extracts information on exception or completion conditions from the diagnostics area and returns it to the issuer. Use this statement in SQL that is embedded in a program.

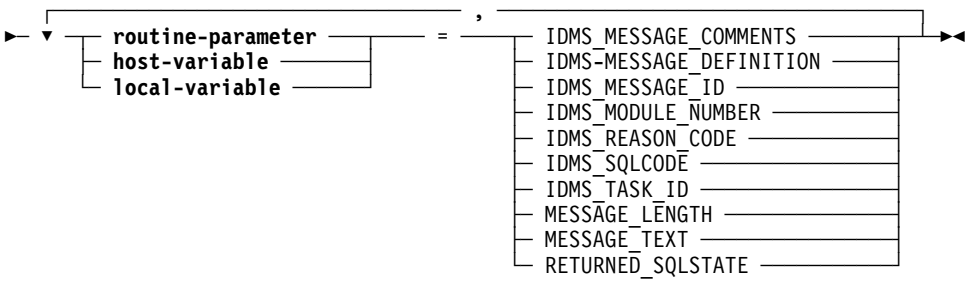
Syntax



Expansion of statement-info



Expansion of condition-info



Parameters

routine-parameter

Identifies an SQL routine parameter that is to receive the value of the specified diagnostics item. **Routine-parameter** must be a parameter of the current SQL routine and must be compatible for assignment with the specified diagnostic item.

For information about expanded syntax, see 5.1.3.7, “Expansion of Routine-parameter” on page 119.

host-variable

Identifies a host variable that is to receive the value of the specified diagnostics item. **Host-variable** must be a host variable previously declared in the application program and must be compatible for assignment with the specified diagnostic item.

For information about expanded syntax, see Expansion of Host-variable in "Values and Value Expressions" in the *CA IDMS SQL Reference Guide*.

local-variable

Identifies a local variable of an SQL routine that is to receive the value of the specified diagnostics item. **local-variable** must be a local variable declared in the current SQL routine and must be compatible for assignment with the specified diagnostic item.

For information about expanded syntax, see 5.1.3.5, “Expansion of Local-variable” on page 117.

statement-info

Identifies the type of statement information to be extracted and returned. **Statement-info** names that begin with 'IDMS_' are extensions to the ISO standard.

COMMAND_FUNCTION

Returns a value with data type varchar(64) indicating the type of SQL command that was last executed. The values that may be returned are listed under the Statement Type column in Table Procedure Requests in the "Defining and Using Table Procedures" chapter in the *CA IDMS SQL Reference Guide*.

COMMAND_FUNCTION_CODE

Returns a value with data type integer indicating the type of SQL command that was last executed. The values that may be returned are listed under the Command Number column in Table Procedure Requests in the "Defining and Using Table Procedures" chapter in the *CA IDMS SQL Reference Guide*.

DYNAMIC_FUNCTION

Returns a value with data type varchar(64) indicating the type of SQL command that was prepared or dynamically executed by the last command. The values that may be returned are listed under the Statement Type column in Table Procedure Requests in the "Defining and Using Table Procedures" chapter in the *CA IDMS SQL Reference Guide*.

DYNAMIC_FUNCTION_CODE

Returns a value with data type integer indicating the type of SQL command that was prepared or dynamically executed by the last command. The values that may be returned are listed under the Command Number column in Table Procedure Requests in the "Defining and Using Table Procedures" chapter in the *CA IDMS SQL Reference Guide*.

IDMS_RETURNED_RESULT_SETS

Returns a value with data type integer indicating the number of result sets returned by a procedure invoked by the last command. This value is only valid if the diagnosed statement is a call of an SQL invoked procedure.

MORE

Returns a value with data type char(1). A value of 'Y' indicates that the execution of the previous SQL statement caused more conditions than have been set in the diagnostics area. A value of 'N' means that the diagnostics area contains information on all the completion and exception conditions.

NUMBER

Returns a value with data type integer indicating the number of the exceptions or completion conditions set by the execution of the previous SQL statement for which information is available in the diagnostics area.

ROW_COUNT

Returns a value with data type DEC(31). The value depends on the type of the previously executed statement:

- INSERT - Number of rows inserted
- DELETE - Number of rows deleted
- UPDATE - Number of rows updated
- BULK FETCH - Number of rows fetched
- FETCH - 1 or 0

CONDITION

Requests diagnostic information for a condition.

EXCEPTION
Specifies a synonym for CONDITION. While it is part of the current ISO standard, its use is discouraged because it will not be in future ISO standards.

condition-nr
Specifies the number of the completion or exception condition for which diagnostics information is being requested. An exception is raised if *condition-nr* does not refer to a valid condition number.

condition-info
Identifies the type of condition-related information to be extracted and returned. **Condition-info** names that begin with 'IDMS_' are extensions to the ISO standard.

IDMS_MESSAGE_COMMENTS
Returns a value with data type varchar(4000) containing the comments in the message dictionary for the message associated with the condition.

IDMS_MESSAGE_DEFINITION
Returns a value with data type varchar(4000) containing the definition in the message dictionary of the message associated with the condition.

IDMS_MESSAGE_ID
Returns a value with data type char(8) containing the message ID in the message dictionary of the message associated with the condition.

IDMS_MODULE_NUMBER
Returns a value with data type integer containing the number of the module that detected the condition.

IDMS_REASON_CODE
Returns a value with data type integer containing the reason code of the condition.

IDMS_SQLCODE
Returns a value with data type integer containing the SQLCODE value associated with the condition. See the "SQL COMMUNICATION Area" appendix in the *CA IDMS SQL Reference Guide*.

IDMS_TASK_ID
Returns a value with data type integer containing the IDMS task ID of the task that encountered the condition.

MESSAGE_LENGTH
Returns a value with data type integer indicating the length of the message associated with the specified condition.

MESSAGE_TEXT
Returns a value with data type varchar(256) containing the message text associated with the specified condition.

RETURNED_SQLSTATE
Returns a value with data type char(5) indicating the SQLSTATE associated with the specified condition.

Example

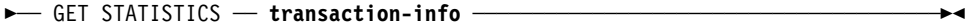
The procedure TGETDIAG1 executes a SELECT statement that causes a number of string truncations. The first GET DIAGNOSTICS returns the number of conditions that the SELECT statement raised. A WHILE LOOP containing the second GET DIAGNOSTICS concatenates the message texts of all the raised conditions to the RESULT parameter of the procedure.

```
set options command delimiter '++';
drop procedure SQLROUT.TGETDIAG1++
commit++
create procedure SQLROUT.TGETDIAG1
  ( TITLE    varchar(10) with default
  , P_NAME   char(18)
  , P_NUMBER integer
  , RESULT   varchar(512)
  )
    EXTERNAL NAME TGETDIAG LANGUAGE SQL
begin not atomic
  declare L_NUMBER  integer      default 1;
  declare L_MESSAGE varchar(256) default ' ';
  select NAME into P_NAME from system.schema
   where cast(NAME as char(12)) = P_NAME;
  /* retrieve the number of conditions raised */
  get diagnostics P_NUMBER = NUMBER;
  while (L_NUMBER < = P_NUMBER)
    do
      /* retrieve the message text of the raised condition */
      get diagnostics condition L_NUMBER
        L_MESSAGE = MESSAGE_TEXT
      set RESULT = RESULT || ' ' || L_MESSAGE;
      set L_NUMBER = L_NUMBER + 1;
    end while;
end
++

call SQLROUT.TGETDIAG1('TGETDIAG1', 'SYSTEM');
**
** TITLE       P_NAME                 P_NUMBER
** -----       ------                 --------
** TGETDIAG1   SYSTEM                        4
**
** RESULT
** ------
** DB001043 T171 C1M322: String truncation DB001043 T171 C1M322:
** String truncation DB001043 T171 C1M322: String truncation
** DB001043 T171 C1M322: String truncation
```

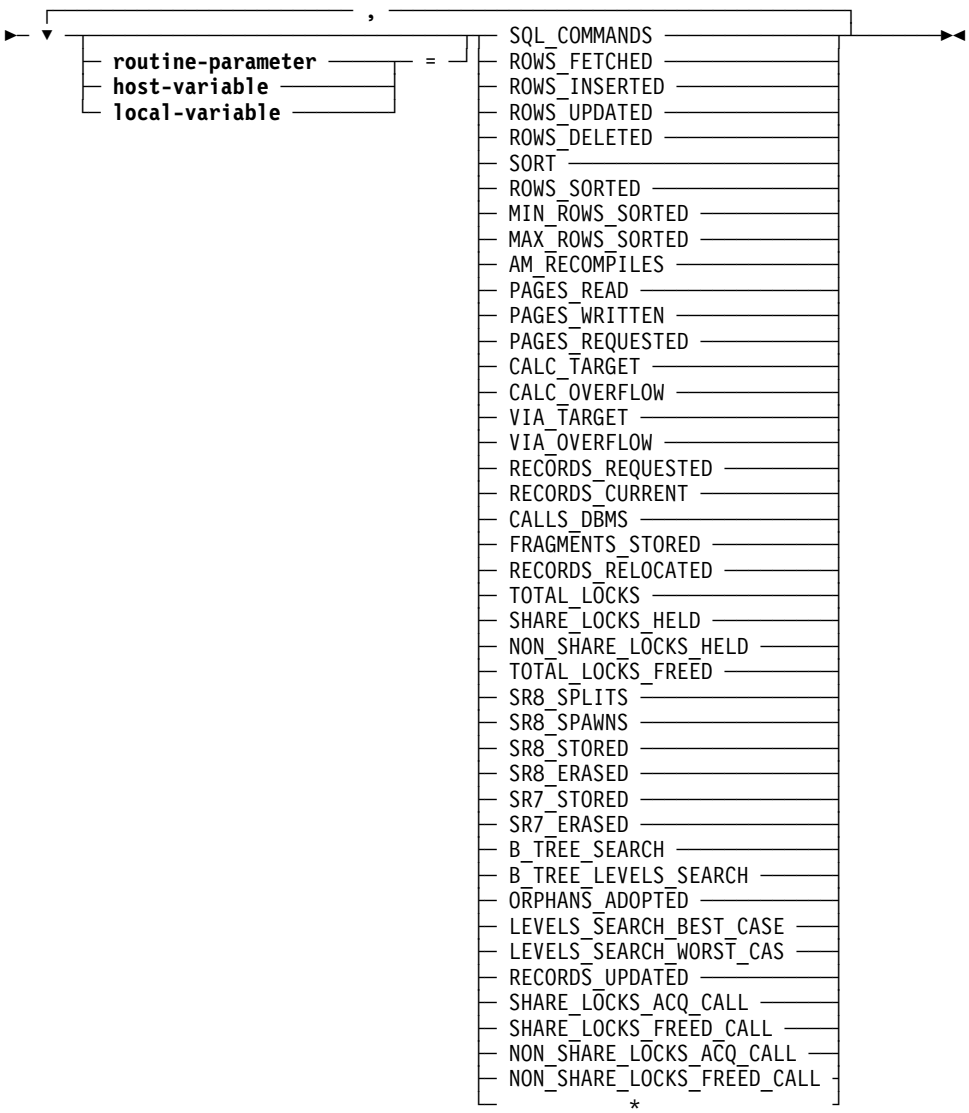
5.3.2 GET STATISTICS

The GET STATISTICS statement returns statistical information for the current transaction. It is a CA IDMS extension to the ISO SQL standard. Use this statement in SQL that is embedded in a program, in the SQL command facility, and in the command console of CA IDMS Visual DBA.

Syntax



Expansion of transaction-info



Parameters

routine-parameter

Identifies an SQL routine parameter that is to receive the value of the specified statistics item. **Routine-parameter** must be a parameter of the current SQL routine and must be compatible for assignment with the specified statistics item.

For information about expanded syntax, see 5.1.3.7, “Expansion of Routine-parameter” on page 119.

host-variable

Identifies a host variable that is to receive the value of the specified statistics item. **Host-variable** must be a host variable previously declared in the application program and must be compatible for assignment with the specified statistics item.

For information about expanded syntax, see Expansion of Host-variable in "Values and Value Expressions" in the *CA IDMS SQL Reference Guide*.

local-variable

Identifies a local variable of an SQL routine that is to receive the value of the specified statistics item. **local-variable** must be a local variable declared in the SQL-invoked routine and must be compatible for assignment with the specified statistics item.

For information about expanded syntax, see 5.1.3.5, “Expansion of Local-variable” on page 117.

Note: A routine-parameter, host-variable or local-variable must be specified for each transaction-info when the statement is embedded in a program. Otherwise, these must not be specified.

transaction-info

Identifies the type of transaction information that is to be returned. Each item has an integer data type and represents statistical information for the current transaction. For more information about these items, see the DCMT DISPLAY STATISTICS SYSTEM and DCMT DISPLAY TRANSACTION commands in the *CA IDMS System Tasks and Operator Commands Guide*.

*****

Requests that all **transaction-info** items to be retrieved. This is not allowed in combination with the specification of a **routine-parameter**, **host-variable**, or **local-variable** and therefore cannot be used in a program.

Example

The SQL procedure TGETSTA1 counts the number of rows of one of four tables:

- SYSTEM.TABLE
- SYSTEM.COLUMN

- SYSTEM.SCHEMA
- DEMOEMPL.EMPLOYEE

The actual table is selected through the value of the TITLE parameter. Besides returning the count of rows, the procedure also returns the values of a number of statistical information items for the transaction:

- SQL_COMMANDS
- PAGES_REQUESTED
- PAGES_READ
- CALLS_DBMS
- TOTAL_LOCKS

```
set options command delimiter '++';
drop procedure   SQLROUT.TGETSTA1++
commit++
create procedure SQLROUT.TGETSTA1
  ( TITLE             char(8) with default
  , P_COUNT           integer
  , P_SQL_COMMANDS    integer
  , P_PAGES_REQUESTED integer
  , P_PAGES_READ      integer
  , P_CALLS_DBMS      integer
  , P_TOTAL_LOCKS     integer
  )
    EXTERNAL NAME TGETSTA1 LANGUAGE SQL
Lab1: begin not atomic
 case TITLE
   when 'TABLE'
     then  select count(*) into P_COUNT
           from SYSTEM.TABLE;
   when 'COLUMN'
     then  select count(*) into P_COUNT
           from SYSTEM.COLUMN;
   when 'SCHEMA'
     then  select count(*) into P_COUNT
           from SYSTEM.SCHEMA;
   when 'EMPLOYEE'
     then  select count(*) into P_COUNT
           from DEMOEMPL.EMPLOYEE;
  end case;

 get statistics
    P_SQL_COMMANDS    = sql_commands
  , P_PAGES_REQUESTED = pages_requested
  , P_PAGES_READ      = pages_read
  , P_CALLS_DBMS      = calls_dbms
  , P_TOTAL_LOCKS     = total_locks;
end
++
set options command delimiter default ++

call sqlrout.TGETSTA1('TABLE');
++
```

```
*+ TITLE         P_COUNT  P_SQL_COMMANDS  P_PAGES_REQUESTED  P_PAGES_READ
*+ -----         -------  --------------  -----------------  ------------
*+ TABLE             808               2                836             9
*+
*+ P_CALLS_DBMS  P_TOTAL_LOCKS
*+ ------------  -------------
*+          813           1673

call sqlrout.TGETSTA1('COLUMN');
*+
*+ TITLE         P_COUNT  P_SQL_COMMANDS  P_PAGES_REQUESTED   P_PAGES_READ
*+ -----         -------  --------------  -----------------   ------------
*+ COLUMN           6450               3               8953           1068
*+
*+ P_CALLS_DBMS  P_TOTAL_LOCKS
*+ ------------  -------------
*+         8071           8300

call sqlrout.TGETSTA1('SCHEMA');
*+
*+ TITLE         P_COUNT  P_SQL_COMMANDS  P_PAGES_REQUESTED   P_PAGES_READ
*+ -----         -------  --------------  -----------------   ------------
*+ SCHEMA             56               4                 59              2
*+
*+ P_CALLS_DBMS  P_TOTAL_LOCKS
*+ ------------  -------------
*+           61            130

call sqlrout.TGETSTA1('EMPLOYEE');
*+
*+ TITLE         P_COUNT  P_SQL_COMMANDS  P_PAGES_REQUESTED   P_PAGES_READ
*+ -----         -------  --------------  -----------------   ------------
*+ EMPLOYEE           55               5                 58              2
*+
*+ P_CALLS_DBMS  P_TOTAL_LOCKS
*+ ------------  -------------
*+           60            128
```

5.4 Enhanced ANSI/ISO SQL JOIN Support

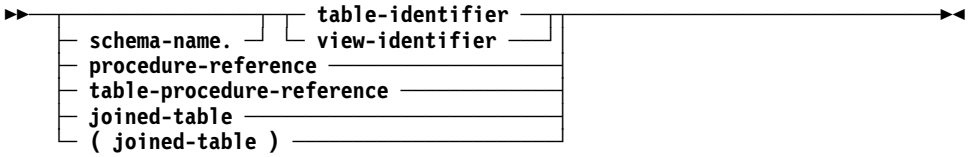
ANSI/ISO SQL join support extends the CA IDMS SQL language with the following <joined table> SQL language element components:

- <cross join> forms the Cartesian product of two tables.
- <qualified join> allows users to specify an inner join or a left, right or full outer join in a structured way.
- <union join> combines two tables where values of columns originating from the other table are set to NULL.

5.4.1 Expansion of Table-reference

The SQL language element **table-reference** is extended with the new **joined-table** parameter as follows:

Syntax



Parameter

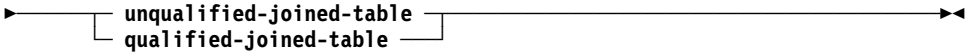
joined-table
Defines a joined table.

5.4.1.1 Expansion of Joined-table

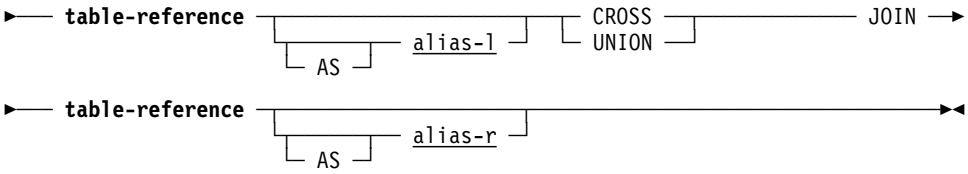
Joined-table represents a table that is derived from joining two specified tables. The different types of join operations are specified through the following join types:

- CROSS
- UNION
- INNER
- LEFT OUTER
- RIGHT OUTER
- FULL OUTER

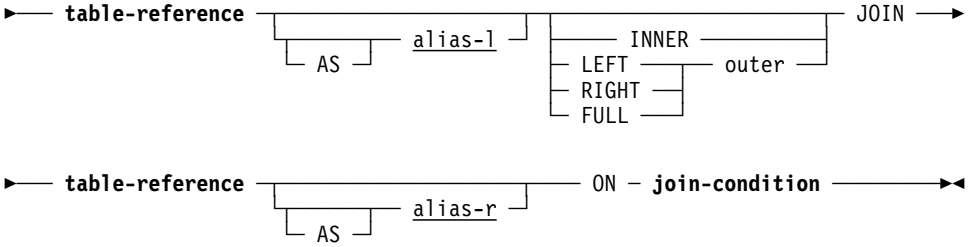
Syntax



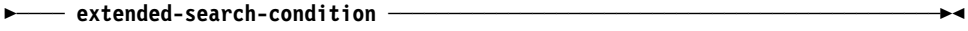
Expansion of unqualified-joined-table



Expansion of qualified-joined-table



Expansion of join-condition



Parameters

unqualified-joined-table
Specifies a **joined-table** where the join operation is a cross or union.

qualified-joined-table
Specifies a **joined-table** where the join operation is an inner, left outer, right outer, or full outer.

table-reference
Represents a table-like object. In a **joined-table** specification, a left and a right **table-reference** are required to define the left and right operands of the join operation.

AS alias-l
Defines a new name used to identify the left table-like object within the **joined-table** specification. **Alias-l** must be a 1-through 18-character name that follows the conventions for SQL identifiers.

AS alias-r
Defines a new name used to identify the right table-like object within the **joined-table** specification. **Alias-r** must be a 1-through 18-character name that follows the conventions for SQL identifiers.

CROSS
Specifies a cross join. A cross join is the cross product of the left and right table.

UNION
Specifies a union join. A union join is equivalent to a full outer join where the **join-condition** always evaluates to false.

INNER
Specifies an inner join. In an inner join, the cross product of the left and right table-like objects is made, and only the rows for which **join-condition** evaluates to true are kept in the result. This is the default.

LEFT outer
Specifies a left outer join. In a left outer join, the cross product of the left and right table-like objects is made, and the rows for which **join-condition** evaluates to true are kept. The result is extended with all the missing rows from the left table, and the values of the columns in the result row, derived from the right table, are set to NULL.

RIGHT outer
Specifies a right outer join. In a right outer join, the cross product of the left and right table-like objects is made, and the rows for which **join-condition** evaluates to true are kept. The result is extended with all the missing rows from the right table, and the values of the columns in the result row, derived from the left table, are set to NULL.

FULL outer
Specifies a full outer join. In a full outer join, the cross product of the left and right table-like objects is made, and the rows for which **join-condition** evaluates to true are kept. The result is extended with all the missing rows from the left table, and the values of the columns in the result row, derived from the right table, are set to NULL. The result is further extended with all the missing rows from the right table, and the values of the columns in the result row, derived from the left table, are set to NULL.

join-condition
Represents the condition for joining two table-like objects in a qualified join. Expanded syntax for **join-condition** appears immediately after the **joined-table** syntax.

extended-search-condition
Specifies the condition used for joining two table-like objects in a qualified join.

Note: For more information about expanded syntax, see "Expansion of Extended-search-condition" in the *CA IDMS SQL Reference Guide*.

Usage

Effect on updatability: A query expression that contains a **joined-table** is not updatable.

Nesting of joined-table expressions: Joined-table expressions can be nested. The table reference in either the left or right operand of a joined-table expression can be another **joined-table**. The default order of evaluation of nested joined-table expressions is left to right. You can use parenthesis for clarity and to alter the default evaluation order.

Restrictions on the use of **set-specification***:* Only the **join-condition** of the inner-most join can contain a **set-specification**, since both the left and right table references in a **set-specification** must be to base tables of a non-SQL-defined database.

Examples

Selecting all Departments and Employees in Department: The following examples list all the departments and the employees of the department. The two statements give identical results.

```
select d.*, e.*
 from DEMOEMPL.DEPARTMENT d left join DEMOEMPL.EMPLOYEE e
   on  d.dept_id = e.dept_id
select d.*, e.*
 from DEMOEMPL.EMPLOYEE e  right join DEMOEMPL.DEPARTMENT d
   on  d.dept_id = e.dept_id
```

Selecting all Departments and Employees in Department With or Without Position: The following examples show nesting of joined tables. The two statements give identical results.

```
select d.*, e.*, p.*
    from DEMOEMPL.DEPARTMENT d left join
         (DEMOEMPL.EMPLOYEE  e left join DEMOEMPL.POSITION p
                                    on p.EMP_ID  = e.EMP_ID )
                                    on e.DEPT_ID = d.DEPT_ID;
select d.*, e.*, p.*
    from DEMOEMPL.DEPARTMENT d left join
         (DEMOEMPL.POSITION p right join DEMOEMPL.EMPLOYEE e
                                    on p.EMP_ID  = e.EMP_ID )
                                    on e.DEPT_ID = d.DEPT_ID;
```

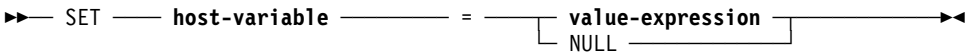
5.4.2 More Information

For more information about Expansion of Table-reference, see the *CA IDMS SQL Reference Guide*.

5.5 SET Host-variable Assignment

A new SET statement enables directly assigning the results of an SQL value expression to a host variable. This statement can only be used in embedded SQL.

Syntax



Parameters

host-variable
Identifies a **host-variable** that is to receive the value of the specified value expression or null. **Host-variable** must be a host variable previously declared in the application program.

value-expression
Specifies the value to be assigned to the destination or receiving field of the assignment statement.

NULL
Specifies that **host-variable** is set to the NULL value.

Usage

The rules for assignment are provided in Comparison, Assignment, Arithmetic, and Concatenation Operations in the "Data Types and Null Values" chapter in the *CA IDMS SQL Reference Guide*.

Example

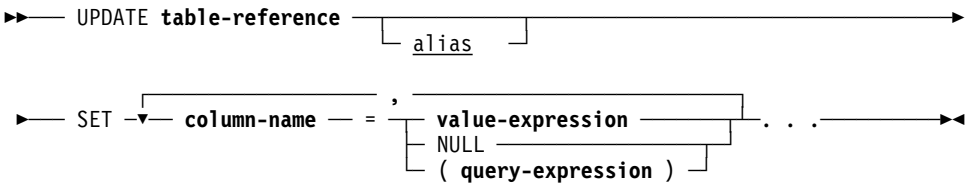
The **host-variable** COMB-NAME is constructed from the values in the **host-variable**s FIRST-NAME and LAST-NAME.

```
EXEC SQL
   set:COMB-NAME=trim(:FIRST-NAME) ||' '|| trim(:LAST-NAME);
END-EXEC
```

5.6 Extended Use of query-expression

The UPDATE statement has been extended to allow a **query-expression** as a value for a column. This permits the use of UNION (ALL) in the specification of value expression.

Syntax



Parameter

(query-expression)

Represents a value to be assigned to a column in an UPDATE statement. The **query-expression** must return at most one row and the result table of the **query-expression** must consist of a single column.

Note: For more information about expanded **query-expression** syntax, see the chapter "Query Specifications, Subqueries, Query Expressions, and Cursor Specifications" in the *CA IDMS SQL Reference Guide*.

Usage

Using a ***query-expression*** as a Source Value*:* If a query expression, used as the value to be assigned to a column, returns no rows, then the column is set to the NULL value. If the column does not allow NULLs, an exception is raised.

5.7 SET OPTIONS COMMAND DELIMITER

The SET OPTIONS statement of the command facility (OCF and IDMSBCF) has been enhanced to support the specification of a character string different from the default command terminator, the semi-colon (;).

The use of an alternate command delimiter is required when entering multi-statement SQL routine bodies using the CREATE PROCEDURE or CREATE FUNCTION commands. According to the SQL procedural language, multiple SQL statements must be separated by the semi-colon. However, using the semi-colon also as the command terminator would truncate the CREATE command after the first semi-colon, and any statements thereafter would erroneously be interpreted as new commands for the command facility and not as statements that make up the rest of the SQL routine body.

Note: Specifying a command terminator string replaces the one that is currently in effect, which by default is the semi-colon. The specification of a command delimiter remains in effect until a new SET OPTIONS COMMAND DELIMITER is issued or until the end of the command facility session.

Syntax

Parameters

COMMAND DELIMITER

Specifies the character string whose value is used to delimit a command facility statement.

'delimiter'

Specifies the character string literal to be used as a delimiter. *'Delimiter'* must be a 1- to 32-character string.

DEFAULT

Specifies that the default of a semicolon (;) be used as a delimiter.

Example

```
set options command delimiter '++';
drop procedure PRODUCTION.PROCESS ++
commit++
create procedure PRODUCTION.PROCESS
  (PROC_TYPE integer,PROC_VALUE char(20))
    external name DPROCESS language SQL
begin
  set PROC_TYPE = 12;
  set PROC_VALUE = 'High';
end
++
set options command delimiter DEFAULT ++
```

5.8 Pseudo Table SYSCA.SINGLETON_NULL

The SYSCA.SINGLETON_NULL is a pseudo table that can be used to return the results of expressions whose parameters are constants. It has one row and no columns. This table is a pseudo table because it does not exist in the catalog. It can be queried through a SELECT statement. This table is used internally by CA IDMS, and it is also useful when evaluating SQL functions and other expressions with constant parameters.

Example

```
select USER01.TLANG1('James    ', 'Last   ')
  from SYSCA.SINGLETON_NULL;

*+
*+ USER_FUNC
*+ ---------
*+ James Last
```

Chapter 6. TCP/IP

This chapter describes the new TCP/IP enhancements. It contains the following topics:

6.1 Port Number Independence

In r17, CA IDMS provides the ability for DC/UCF systems and applications that execute within those systems to use logical service names in place of port numbers. The use of logical names allows port numbers to be changed without impacting application code or system definitions.

Port number independence is provided through the following facilities:

- A CA IDMS services resolver that translates a service name into a port number or vice versa.
- The ability to specify a service name in place of a port number on LISTENER and DDSTCPIP PTERM SYSGEN statements.
- The ability to specify or change a service name using a DCMT VARY PTERM statement.
- The ability to display the service name for a PTERM using a DCMT DISPLAY PTERM statement.
- New socket functions for returning a port number associated with a service name or the service name(s) associated with a port number.

The new socket functions are described in 6.5.1, “GETSERVBYNAME” on page 215 and 6.5.2, “GETSERVBYPORT” on page 217. The remainder of the new facilities are described below.

6.1.1 CA IDMS Services Resolver

The CA IDMS services resolver is a new component of r17 responsible for translating service names to and from port numbers. It does this using the contents of a services file that can be included in the execution JCL of a DC/UCF system. The DDNAME or filename of the services file is specified in the new TCP/IP SYSGEN entity.

The services file contains a list of service names together with the port number, protocol and optional alias names associated with each one. This information is used when servicing a GETSERVBYNAME or GETSERVBYPORT socket function to retrieve the port number associated with a service name or the service name associated with a port number.

On z/OS, the default services data set is cataloged as <*tcpip-hlq*>.ETC.SERVICES. You can use this data set, a customized data set containing the definitions of the services that you want to use, or a combination of the two by concatenating the two files together. To ensure that the customized entries take precedence, include the customized data set before the system default data set.

During the initialization of the TCP/IP environment in a CA IDMS system, the services file is read and its contents converted to an internal structure that is

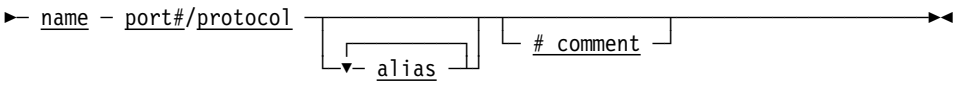
stored in memory for efficient access. The contents of the in-core structure can be refreshed from the services file using the new DCMT VARY TCP/IP command.

Note: To allow updating the services file while it is in use by a CV, make it a member of a partitioned data set rather than a sequential file.

Each record within a services file defines a service name entry in character format.

Syntax

The syntax of the services data set records accepted by the CA IDMS services resolver follows the general standards for such files. The format is as follows:



Parameters

name
Identifies the official Internet service name.

port#
Identifies the port number. Port number is a positive number between 1 and 65535.

protocol
Identifies the protocol used for the service.

alias
Identifies an unofficial name for the service.

 # comment
Specifies comment text that continues until the end of the record.

Notes

- Record size must be between 56 and 256.
- Items in a record are separated by spaces or tabs.
- *name*, *protocol,* and *alias* are case sensitive.
- *name* must start in column 1.
- Records with duplicate service names are ignored. The first definition of service name takes precedence.
- Maximum length for *name* and *alias* is normally 32 characters, but longer names are accepted on all systems.
- Comments begin with the number sign (#) character and continue until the end of record.

- The only protocols accepted in CA IDMS are TCP and UDP, in any combination of uppercase and lowercase.

The following illustrates a standard TCPIP.ETC.SERVICES data set delivered with the operating system:

```
#
# Network services, Internet style
#
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp             21/tcp
. . . . . .
```

More Information

- For more information about the GETSERVBYNAME and GETSERVBYPORT socket functions, see 6.5, “New Socket Functions” on page 215.
- For more information about the new TCP/IP SYSGEN entity, see 6.3, “New TCP/IP System Entity” on page 198.
- For more information about the DCMT VARY TCP/IP command, see 6.3.5, “DCMT VARY TCP/IP Command” on page 209.

6.1.2 Service Name for LISTENER and DDSTCPIP PTERMS

To allow DC/UCF system definitions to be independent of port numbers, you can now specify a service name in place of a port number on LISTENER and DDSTCPIP PTERM SYSGEN statements and alter a service name using a DCMT VARY PTERM statement. Additionally the DCMT DISPLAY PTERM statement has been enhanced to report on the service name associated with a PTERM.

The PORT (for a LISTENER PTERM) and TARGET PORT (for a DDSTCPIP PTERM) parameters can now accept a service name in place of a port number value. The service name must be the name of a service in the services file with an associated protocol of TCP. Its length is limited to 32 characters.

When the corresponding PTERM is varied online, the CA IDMS services resolver tries to retrieve the port number associated with the service name. Regardless of the TCP/IP CASE option in effect for the system, CA IDMS always attempts to find the matching entry using the CASE SENSITIVE option

first and if not found will then search using the CASE INSENSITIVE option. The same method is followed for the TCP protocol.

The output of a DCMT DISPLAY PTERM command displays the service name associated with a LISTENER or DDSTCPIP PTERM as illustrated in the following example.

Example

DCMT DISPLAY PTERM SY71CA31

```
Logical Term ID SY71CA31
         Physical Term ID SY71CA31
         Physical Line ID TCPIP
       Physical Term Type DDS TCP/IP
      Physical Term Model
     Physical Term Status InSrv
      Logical Term Status InSrv
            IP stack name *none (*DEFAULT)
              Target node SYSTEM71
              Target host USILCA31
              Target port  3771/DDS-TO-SYSTEM71
               Port range   OFF
            Idle interval    60
    Permanent connections     3
      Maximum connections   OFF
       Active connections     1
     Connections rejected     0
          Number of Reads 0000000
         Number of Writes 0000000
    Number of Read Errors 0000000
   Number of Write Errors 0000000
```

More Information

- For more information about the PORT and TARGET port parameters, see 6.6.2, “System Generation PTERM Statement” on page 221.
- For more information about varying the service name for a PTERM, see 6.6.3, “DCMT VARY PTERM Command” on page 225.
- For more information about the TCP/IP CASE option, see 6.3.1, “System Generation TCP/IP Statement” on page 198.

6.2 Enhanced Stack Selection

In a multiple TCP/IP stack environment, this feature lets you control or limit the stacks that are usable by the socket applications running in the CA IDMS system. This enhancement is primarily for CA IDMS systems running on z/OS with CINET active and on z/VM. It is useful in an environment where certain applications need to use secured sockets or some TCP/IP stacks are for testing only.

You can control or limit the TCP/IP stacks using the following methods:

- At startup, through a new system generation TCP/IP statement
- At startup, through new SYSIDMS parameters
- Dynamically, through a new DCMT VARY TCP/IP command

The new SYSIDMS parameters are described below. For more information about the two other methods, see 6.3, “New TCP/IP System Entity” on page 198.

6.2.1 SYSIDMS Parameters

Use the new SYSIDMS parameters to create an INCLUDE list or an EXCLUDE list of TCP/IP stack names to use at runtime. You can use these parameters to do the following:

- (z/OS only) Create an INCLUDE or an EXCLUDE list of stack names to override the list of stacks provided by the operating system
- (z/OS and z/VM only) Create an INCLUDE or EXCLUDE list of stack names to override the list of stacks defined through SYSGEN
- (z/VM only) Include a new stack

Parameter Descriptions

EXCLUDE_TCP/IP_STACK

(z/OS and z/VM only) Creates an EXCLUDE list of up to eight TCP/IP stack names to override the list of stacks supplied by the operating system or SYSGEN.

EXCLUDE_TCP/IP_STACK and INCLUDE_TCP/IP_STACK are mutually exclusive parameters and support wildcards.

INCLUDE_TCP/IP_STACK

(z/OS and z/VM only) Creates an INCLUDE list of up to eight TCP/IP stack names to override the list of stacks supplied by the operating system or SYSGEN.

EXCLUDE_TCP/IP_STACK and INCLUDE_TCP/IP_STACK are mutually exclusive parameters and support wildcards.

Examples

The following examples illustrate how to specify SYSIDMS parameters to create an INCLUDE list or an EXCLUDE list of TCP/IP stacks.

The SYSIDMS parameter that is processed first determines whether an INCLUDE list or an EXCLUDE list is built; any parameters from the other group are rejected.

The following group of parameters specifies a list of TCP/IP stacks to include in the INCLUDE list.

```
INCLUDE_TCP/IP_STACK=TCPIP31
INCLUDE_TCP/IP_STACK=TCPIP31A
INCLUDE_TCP/IP_STACK=TCPIP31B
```

The following group of parameters specifies an EXCLUDE list of parameters to reject from the system list, the TCPIP31X stack and all the stacks starting with the RUNT pattern:

```
EXCLUDE_TCP/IP_STACK=TCPIP31X
EXCLUDE_TCP/IP_STACK=RUNT*
```

More Information

For more information about SYSIDMS, see the *CA IDMS Common Facilities Guide*.

6.3 New TCP/IP System Entity

Support of TCP/IP in CA IDMS has been enhanced by enabling the TCP/IP runtime environment to be defined independently from a socket line. This consolidates the definition of the TCP/IP attributes that were previously specified in the SOCKET line definition and various SYSIDMS parameters and enables them to be displayed and varied using new DCMT DISPLAY TCP/IP and DCMT VARY TCP/IP commands.

The new TCP/IP entity provides the following benefits:

- Multiple socket lines can be active at the same time, allowing specific listeners to be associated with different lines.
- Client programs can use TCP/IP services even if no SOCKET line is defined.
- New system-wide attributes can be specified and altered dynamically.

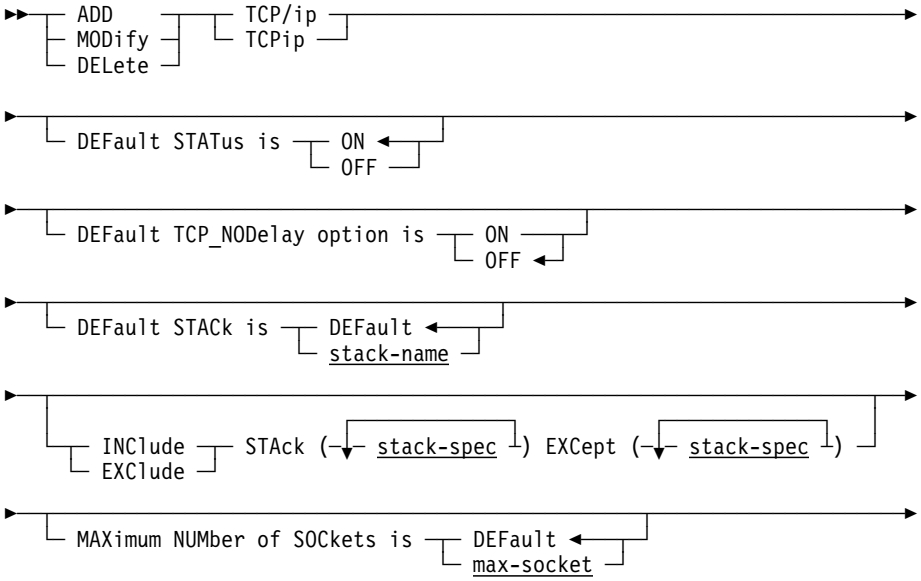
You define the TCP/IP runtime environment for a DC/UCF system through the new TCP/IP system generation statement or through the new DCMT VARY TCP/IP command. Both methods are described in the following sections. Enhancements to related SYSGEN and DCMT commands are also described.

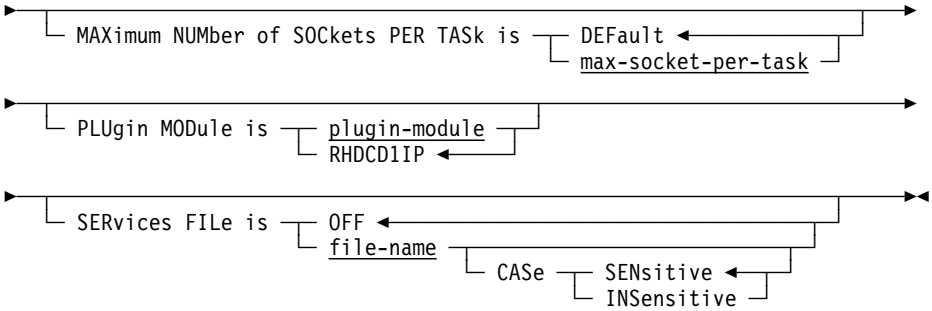
6.3.1 System Generation TCP/IP Statement

The TCP/IP statement is used to define the TCP/IP runtime environment of a DC/UCF system.

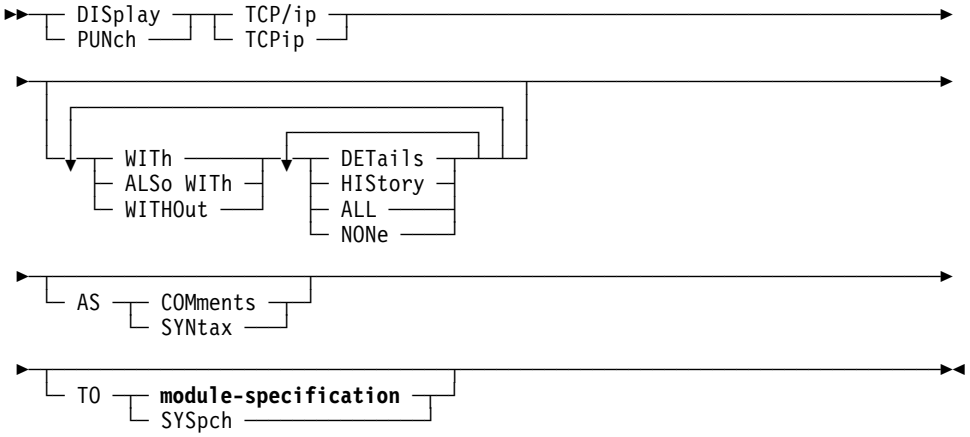
Syntax

ADD/MODIFY/DELETE TCP/IP Statement





DISPLAY/PUNCH TCP/IP Statement



Parameters

TCP/ip and TCPip

These are equivalent keywords that can be used the same way everywhere. They can be specified in SYSGEN definitions or DCMT commands.

DEFault STATus is

Specifies the status of TCP/IP support in CA IDMS. OFF disables TCP/IP support for CA IDMS at startup. ON enables support. The default is ON if an ADD TCP/IP statement is coded. If the TCP/IP entity is not defined to the system, the default is OFF.

Note: For compatibility with earlier releases, if the TCP/IP entity is not defined to SYSGEN, but the same system contains the definition of an enabled SOCKET line, TCP/IP is automatically enabled during startup. A warning message is displayed to the log recommending the use of the TCP/IP SYSGEN entity.

Note: The default TCP/IP status value defined to SYSGEN can be overwritten at startup by the new TCP/IP_STATUS=ON/OFF SYSIDMS parameter, and/or it can be changed dynamically at runtime using the new DCMT VARY TCP/IP command.

DEFault TCP_NODelay option is

Specifies the default value for the NO_DELAY SOCKET option. Unless overridden for a specific socket connection, this is the value that will be used for all communication. The default value is OFF, meaning that a delay may be experienced between consecutive sends in order to optimize overall data movement.

DEFault STACk is

Specifies the default stack to be used by the DC/UCF system.

stack-name

Identifies the name of the stack to be used as a default.

DEFault

Specifies that the CA IDMS assigned default stack is to be used. DEFAULT is the default.

The default stack for a DC/UCF system varies by operating system.

- On z/OS, the operating system assigns a specific stack as the default stack. Unless that stack is explicitly excluded from use by SYSGEN or SYSIDMS parameters, CA IDMS uses the operating-system assigned default stack. If the default stack has been excluded, CA IDMS chooses the first active stack from the list of stacks as the default.
- On z/VM, the default stack is the first stack in the list of stacks.

INClude/EXClude STack stack-spec

Controls or limits the stacks that can be used by the socket applications running in the CA IDMS system. This option is useful only in a multiple stack environment where multiple TCP/IP stacks run concurrently; it is ignored on systems where only one TCP/IP stack is active. It is used differently depending on the operating system:

- On z/OS, the INCLUDE or EXCLUDE list is used to customize the default stacks list returned by the operating system when CINET is active. Both lists are mutually exclusive. If no INCLUDE or EXCLUDE list is specified, all stacks in the list returned by the operating system are included.

  If specified, the resulting list of stacks depends on the type of list being defined:

  – INCLUDE List—This list is built by excluding all the stack names that are not present in the SYSGEN INCLUDE list.

  – EXCLUDE List—This list is built by excluding all the stack names that are present in the SYSGEN EXCLUDE list.

  Wildcards can be used as special names for *stack-spec* to define groups of stack names starting with the same pattern. When wildcards are used in the INCLUDE or EXCLUDE list, the EXCEPT list can be used to refine the set of included or excluded stacks by excluding specific stacks.

Wildcards can also be used for *stack-spec* in the EXCEPT list if they represent a sub-group of names from a larger group declared in the INCLUDE or EXCLUDE list. See examples at the end of this section.

- On z/VM, only an INCLUDE list is available. Use the INCLUDE list to define the full list of stacks to use in the CA IDMS system. Wildcards are not accepted.

  This list can be used to replace the r16 definitions using the SYSTCPD file and the existing SYSIDMS parameters; these definitions are ignored when the stacks are defined through SYSGEN.

An empty list can be specified for the INCLUDE, EXCLUDE or EXCEPT list in order to remove all entries from the corresponding list. Duplicate names are ignored when specified within the same list of stacks.

MAXimum NUMber of SOCkets is max-socket

Specifies the maximum number of sockets that can be created globally in the DC/UCF system. *max-socket* is a positive number between 1 and 65535. If DEFAULT is specified, a default value is assigned at startup. This default value depends on the operating system: 65535 on z/OS, 8000 on z/VSE, and 512 on z/VM.

The maximum number of sockets that can be created in one address space can also be limited by the operating system, for example, through USS definitions under z/OS.

MAXimum NUMber of SOCkets per TASk is max-socket-per-task

Specifies the maximum number of sockets that can be created by a single task in the DC/UCF system. The maximum value and the default value for this parameter are both equal to the value assigned at runtime to *max-socket*. If the *max-socket-per-task* value is greater than *max-socket*, it is truncated.

PLUgin MODule is plugin module

Specifies the name of the plug-in module that implements support for specific TCP/IP stack implementations. The only plug-in module name that is accepted is RHDCD1IP; this is also the default value.

SERvices FILe is

Defines the file to be used for translating service names to port numbers and vice versa.

file-name

Specifies the ddname (z/OS and z/VM) or the file name (z/VSE) of the services file.

If the data set or file corresponding to *file-name* cannot be found at runtime (DD card not specified in the startup JCL or data set not cataloged), an error message is written to the log file. Subsequent calls to the GETSERVBYNAME or GETSERVBYPORT socket function returns a specific ERRNO code.

OFF
Indicates that no services file is available and port number/service name resolution is not supported. OFF is the default.

CASe
Indicates whether the service name specified on input to a GETSERVBYNAME socket function is case-sensitive or case-insensitive. The default value is case-sensitive.

Example

Including or Excluding TCP/IP Stacks: This example illustrates a list of INCLUDE and EXCLUDE TCP/IP stack definitions and the TCP/IP stacks generated from them.

Assume the special system call on z/OS returns the following list of TCP/IP stacks as defined to CINET:

```
TCPSY100 - TCPSY110 - TCPSY200 - RUNTCP10 - RUNTCP11 - TESTTCP
```

The following SYSGEN definitions illustrate how to specify the TCP/IP stacks to include or exclude in the CA IDMS system:

- This statement

  ```
  MOD TCPIP
   INCLUDE STACK (TCPSY*,RUNTCP*) EXCEPT (TCPSY2*,RUNTCP11).
  ```

  produces the following list of stacks:

  ```
  TCPSY100 - TCPSY110 - RUNTCP10
  ```

- This statement

  ```
  MOD TCPIP
   INCLUDE STACK (*) EXCEPT (TCPSY*,TESTTCP).
  ```

  produces the following list of stacks:

  ```
  RUNTCP10 - RUNTCP11
  ```

- This statement

  ```
  MOD TCPIP
   EXCLUDE STACK (TCP*) EXCEPT (TCPSY200).
  ```

  produces the following list of stacks:

  ```
  TCPSY200 - RUNTCP10 - RUNTCP11 - TESTTCP
  ```

More Information

For more information about the system generation statements, see the *CA IDMS System Generation Guide*.

6.3.2 System Generation SOCKET LINE Statement

In prior releases, the name of the plug-in module was specified on the MODULE is plug-in clause of the SOCKET LINE SYSGEN statement. While this clause is still supported for upward compatibility, it is no longer required and the name of the plug-in module should now be specified using the PLUGIN MODULE clause of the new system generation TCP/IP statement.

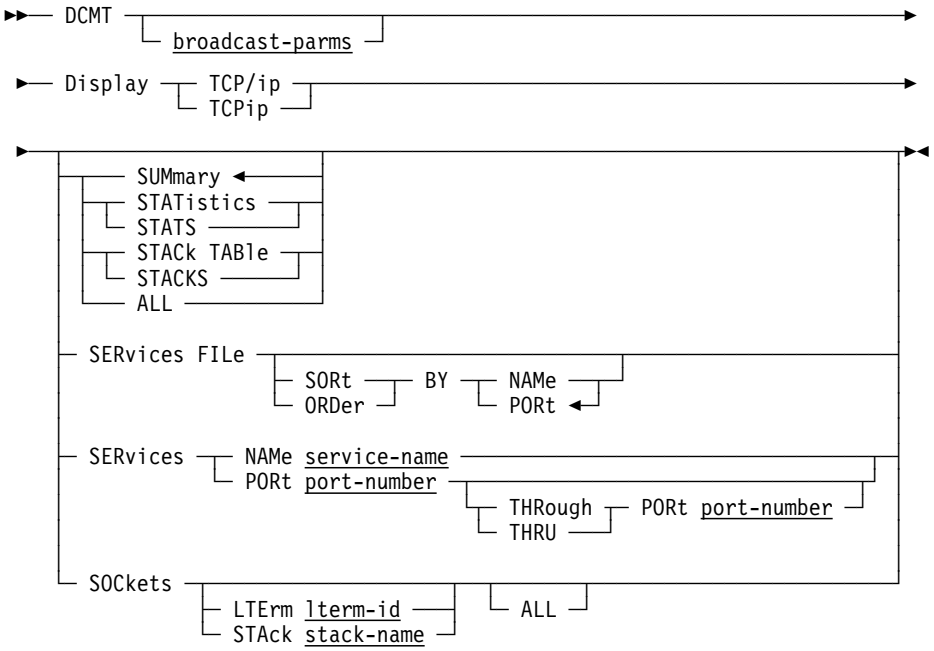
Note: For more information about the PLUGIN MODULE clause, see

6.3.1, “System Generation TCP/IP Statement” on page 198.

6.3.3 DCMT DISPLAY TCP/IP Command

The DCMT DISPLAY TCP/IP command displays information about the TCP/IP runtime environment of a DC/UCF system. In addition to current attribute settings, it can also display TCP/IP-related statistics and a list of all the TCP/IP stacks and their corresponding status.

Syntax



Parameters

broadcast-parms

Specifies to execute the DCMT command on all or a list of data sharing group members.

Note: For more information about broadcasting and **broadcast-parms**, see How to Broadcast System Tasks in the *CA IDMS System Tasks and Operator Commands Guide*.

SUMmary
Displays summary information about this system's TCP/IP environment. This is the default if no option is specified.

STATistics
Displays statistics information.

STACk TABle
Displays the TCP/IP stack table containing the name of all the stacks defined in the system. The output table contains five columns that provide the following information:

- Hostname
- IP address
- Name of the stack (job name), designated with (D) if it is the default stack
- Flag indicating the following values:
  - Y—If stack is active
  - N—If stack is not active
  - Excl-D—If stack is excluded by DCMT Command
  - Excl-G—If stack is excluded by SYSGEN
  - Excl-I—If stack is excluded by SYSIDMS
  - New—If stack is new in the list, after the execution of a DCMT VARY TCP/IP STACK TABLE REFRESH command
- Flag indicating if the stack supports IPv6

ALL
Displays all the information provided by the SUMMARY, STATISTICS, and STACK TABLE options.

SERvices FILe
Displays the contents of the services file, if one is in use. The output table contains three columns that provide the following information:

- Port numbers
- Protocol names
- Service names

  Aliases, if present, are displayed on secondary lines in the service name column.

The output table can be sorted by the service name or by the port number. By default, it is sorted by the port number.

SERvices NAMe or SERvices PORt
Displays the contents of the services file, if one is in use but restricts the output to specific service names or specific port numbers.

service-name

Specifies the name of a specific service or a wildcard that displays all the services with a name starting with the same pattern.

When using the SERVICES PORT clause, you can specify a specific port number or a range of ports.

port-number

Specifies a *port-number*. *port-number* is a positive number between 1 and 65535. If the THROUGH PORT sub-clause is specified, the second *port-number* value must be greater than or equal to the first one.

SOCkets LTErm or SOCkets STAck

Displays information about all LTERM's owning sockets in the system. The output table contains six columns (no ALL option) or ten columns (with ALL option) that display the following information:

- Without the ALL option: the LTERM name, the PTERM name, the PTERM type, the current stack affinity, the current socket function, and the total number of sockets owned by the LTERM.
- With the ALL option specified: the LTERM name, the PTERM name, the PTERM type, and for each socket descriptor currently owned by the LTE, the stack affinity, the socket function, the socket descriptor, the socket domain, an indicator telling whether the TCP_NODELAY socket option applies, and the socket timeout value.

  Note: When the ALL option is specified and the current socket function is SELECT or SELECTX, the name of the function is displayed for the first socket descriptor only.

Examples

DCMT DISPLAY TCP/IP SUMMARY

```
SYSGEN definitions                      Run-time information
=================================       =================================
Default status                    ON    TCP/IP status                Active
Default TCP_NODELAY option       OFF    TCP_NODELAY option              OFF
Max number sockets              9999    Max number sockets             9999
Max number sockets per task      999    Max number sockets per task     999
Plugin module               RHDCD1IP
Services file               SERVICES
Services file case          Sensitive   Services file case        Sensitive
Default stack                DEFAULT    Default stack             TCPIP31
Include stack list           TCP*

SYSIDMS parameters
=================================
EXCLUDE_TCP/IP_STACK        TCPIP31V
```

DCMT DISPLAY TCP/IP STATISTICS

```
Statistics
=====================================================
Number of sockets currently open                    10
Number of sockets created                           11
HWM of concurrent open sockets (global)             11
HWM of concurrent open sockets (1 LTERM)             1
Number of socket reads                              98
Number of socket writes                             64
Number of accepted connections rejected              0
Number of DDS      connections rejected              0
Number of listener connections rejected              0
```

DCMT DISPLAY TCP/IP STACK TABLE

```
Hostname  IP address       Job name    Active  IPv6
========  ===============  ========== ======  ====
HOSTCA31  111.111.111.111  TCPIP31 (D)    Y      Y
HOSTCA32  222.222.222.222  TCPIP32        Y      Y
                           TCPIP33        N
                           RUNTCP      Excl-G
                           TCPIP31V    Excl-I
```

DCMT DISPLAY TCP/IP SERVICES FILE

```
Services file                SERVICES
Services file case          Sensitive

Port#  Protocol  Service name or alias
=====  ========  ====================
    7     tcp    echo
    7     udp    echo
   13     tcp    daytime
   13     udp    daytime
   15     tcp    netstat
   19     tcp    chargen
                 ttytst
                 source
   19     udp    chargen
                 ttytst
                 source
   21     tcp    ftp
   23     tcp    telnet
   ..     . .    . . . .
```

DCMT DISPLAY TCP/IP SERVICE NAME nameserv*

```
Services file                SERVICES
Services file case          Sensitive

Port#  Protocol  Service name or alias
=====  ========  ====================
   42     tcp    nameserver
   53     tcp    nameserver
   53     udp    nameserver
```

DCMT DISPLAY TCP/IP SERVICE PORT 10 THROUGH 20

```
Services file                SERVICES
Services file case          Sensitive

Port#  Protocol  Service name or alias
=====  ========  ====================
   13     tcp    daytime
   13     udp    daytime
   15     tcp    netstat
   19     tcp    chargen
                 ttytst
                 source
   19     udp    chargen
                 ttytst
                 source
```

DCMT DISPLAY TCP/IP SOCKETS

```
Lterm-ID Pterm-ID Type Stack    Socket-call     Count
======== ======== ==== ======== =============== =====
LD000001 *No-PTE* FRST TCPIP31                       1
SY71CA31 SY71CA31 DTCP TCPIP31  RECV   (async)       2
TCLJSRV  TCPJSRV  LIST TCPIP31  ACCEPT (async)       1
TCPLIS01 TCPLIS01 LIST TCPIP31  ACCEPT (async)       1
VL72002  VP72002  3279 TCPIP31  ACCEPT               2
```

DCMT DISPLAY TCP/IP SOCKETS ALL

```
Lterm-ID Pterm-ID Type Stack    Socket-call     Socket-desc Dom NDL Timeout
======== ======== ==== ======== =============== =========== === === =======
LD000001 *No-PTE* FRST TCPIP31                            0 IN   N  Forever
SY71CA31 SY71CA31 DTCP TCPIP31  RECV                      0 IN   Y  Forever
                       TCPIP31                            1 IN   N  Forever
TCLJSRV  TCPJSRV  LIST TCPIP31  ACCEPT (async)            0 IN6  N  Forever
TCPLIS01 TCPLIS01 LIST TCPIP31  ACCEPT (async)            0 IN6  N  Forever
VL72002  VP72002  3279 TCPIP31                            0 IN   N      300
                       TCPIP31  ACCEPT                    1 IN   N      300
```

More Information

For more information about DCMT commands, see the *CA IDMS System Tasks and Operator Commands Guide*.

6.3.4 DCMT DISPLAY LINE Command

The DCMT DISPLAY LINE IPINFO option is no longer supported for a SOCKET LINE. The corresponding output can now be displayed using the DCMT DISPLAY TCP/IP ALL command.

Example

DCMT DISPLAY LINE TCPIP

```
 *** Physical Line Display ***
PLine-ID TCPIP
  Status InSrv
  Opened 2007-04-15-05.55.20.092194
  Module IP
LTerm-ID PTerm-ID Type/M Status  Port Target-host
SY71CA31 SY71CA31 DTCP   OutSrv 00000 USILCA31
SY71CA11 SY71CA11 DTCP   OutSrv 03771 USILCA11
TCPLIS01 TCPLIS01 LIST   OutSrv 01234
TCLJSRV  TCPJSRV  LIST   InSrv  03772
TCPIPB01 TCPIPB01 BULK   Discon
TCPIPB02 TCPIPB02 BULK   Discon
```

Note: If a PTERM has a service name assigned to it, and the PTERM status is out-of-service, the Port column shows the value 00000. You must issue an explicit DCMT DISPLAY PTERM command to display the corresponding service name.

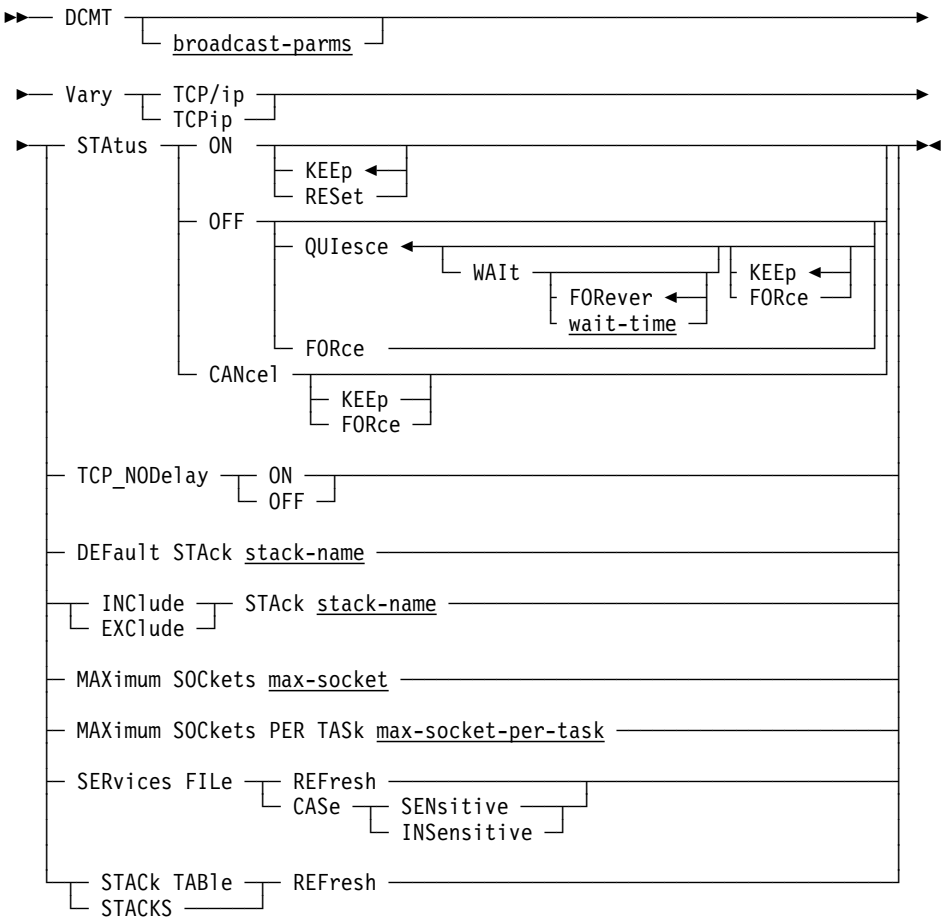
More Information

- For more information about the DCMT DISPLAY LINE command, see the *CA IDMS System Tasks and Operator Commands Guide*.
- For more information about the new DCMT DISPLAY TCP/IP command, see 6.3.3, “DCMT DISPLAY TCP/IP Command” on page 203.

6.3.5 DCMT VARY TCP/IP Command

The DCMT VARY TCP/IP command enables all the parameters that are defined in the system generation TCP/IP statement to be altered dynamically at runtime.

Syntax



Parameters

broadcast-parms

Specifies to execute the DCMT command on all or a list of data sharing group members.

Note: For more information about broadcasting and **broadcast-parms**, see How to Broadcast System Tasks in the *CA IDMS System Tasks and Operator Commands Guide*.

STAtus
Switches the status of the TCP/IP support ON or OFF in the DC/UCF system.

ON KEEp
Enables or reenables TCP/IP support in the DC/UCF system. If reenabling TCP/IP support in the system, the latest value of each option is kept.

ON RESet
Enables or reenables TCP/IP support in the DC/UCF system. If reenabling TCP/IP support in the system, the value of each option is set to its original value.

OFF QUIesce
Prevents the creation of any new sockets, but allows executing applications using sockets to finish processing. All the LISTENER and DDSTCPIP PTERM's are closed. QUIesce is the default option for a DCMT VARY TCP/IP STATUS OFF command. By default, the QUIesce command waits indefinitely until all the socket descriptors are closed.

WAIt wait-time
Sets a maximum time interval the QUIesce command should wait for all socket descriptors to close. *wait-time* is a positive number between 1 and 32767. When this time interval is exhausted or when the quiesce request is canceled, the following occurs, depending on the KEEp or FORce option specified on the WAIt clause:

- If KEEp is specified (default value), TCP/IP is reenabled in the same way as using a DCMT VARY TCP/IP STATUS ON KEEP command.
- If FORCe is specified, TCP/IP is disabled in the same way as using a DCMT VARY TCP/IP STATUS OFF FORCE command.

OFF FORce
Immediately terminates TCP/IP support in the DC/UCF system. All the LISTENER and DDSTCPIP PTERM's are closed, including all active sockets. Applications using sockets receive an error code on their next socket function call.

CANcel
Cancels an outstanding DCMT VARY TCP/IP STATUS OFF QUIESCE command. The KEEp or FORCe option overwrites the KEEp or FORCe option specified on the DCMT VARY TCP/IP STATUS OFF QUIESCE command.

TCP_NODelay
Switches the TCP_NODELAY socket global option ON or OFF.

DEFault STAck stack-name
Overwrites the default stack assigned by the system. Changing the default stack dynamically has no effect on the existing sockets. Only the newly

created sockets that use the default stack affinity are affected. This option is useful only in a multiple stack environment.

INClude STAck stack-name
Includes (activates) a TCP/IP stack in the DC/UCF system. *stack-name* is the job name of a TCP/IP stack and is limited to eight characters. This option is used differently depending on the operating system:

- On z/OS, *stack-name* must be the name of a stack that belongs to the CINET list. That is, it appears in the list of stacks displayed by the DCMT DISPLAY TCP/IP STACK TABLE command.

  If *stack-name* is active in the operating system, it becomes active in the CA IDMS system; if not, it remains inactive in the DC/UCF system.

- On z/VM, *stack-name* can be the name of any stack that is active in the operating system.

EXClude STAck stack-name
Excludes a TCP/IP stack that is included (active) in the DC/UCF system. *stack-name is the* job name of a TCP/IP stack. The *stack-name* is limited to eight characters.

MAXimum SOCkets max-socket
Specifies the maximum number of sockets that can be created globally in the DC/UCF system. *max-socket* is a positive number between 1 and 65535. The maximum number of sockets that can be created in one address space can also be limited by the operating system, for example, through USS definitions under z/OS.

MAXimum SOCkets PER TASk max-socket-per-task
Specifies the maximum number of sockets that can be created by a single task in the DC/UCF system. The maximum value and the default value for this parameter are both equal to the value assigned at runtime to *max-socket*. If the *max- socket-per-task* value is greater than *max-socket*, it is truncated.

SERvices FILe REFresh
Refreshes the internal copy of the services file in memory after the services file has been updated.

Note: To make updates to the services file while the data set is currently defined in the startup JCL with the DISP=SHR option, the file should be allocated as a member from a PDS.

SERvices FILe CASe
Changes the case sensitivity that applies to the services names specified on the GETSERVBYNAME function calls.

STACk TABle REFresh
(z/OS only) Refreshes the list of stacks currently defined to CINET without the need to stop the TCP/IP support in the DC/UCF system. This command is accepted only when the TCP/IP status is ON.

If a new stack has been added to the list, it will not be activated in the DC/UCF system automatically. You must issue an explicit DCMT VARY TCP/IP INCLUDE STACK command to activate it in the DC/UCF system. The DCMT DISPLAY TCP/IP STACK TABLE shows the value New in the Active column from the corresponding entry.

Usage

Specifying new socket values: New values can be assigned to *max-sockets* and *max-socket-per-task* when TCP/IP is currently enabled in the DC/UCF system, only if the new value is lower than the corresponding value at the time TCP/IP was enabled. In the other case, TCP/IP must be recycled. That is, disabled first and then reenabled.

The checks on the maximum number of sockets allowed are always done when a new socket is created. No sockets are forcibly closed if the maximum number of sockets is set to a lower value.

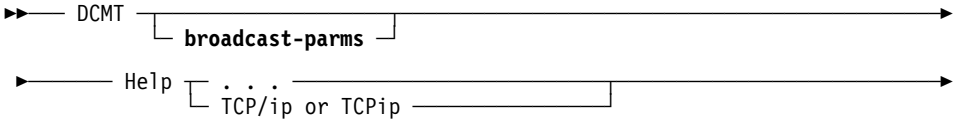
More Information

For more information about DCMT commands, see the *CA IDMS System Tasks and Operator Commands Guide*.

6.3.6 DCMT Help Command

The DCMT HELP command has been enhanced to display a help screen of the DCMT DISPLAY TCP/IP and DCMT VARY TCP/IP syntax.

Syntax



Parameters

Help
Displays syntax for the HELP command.

TCP/ip or TCPip
Displays the TCP/IP help screen.

More Information

For more information about the DCMT HELP command, see the *CA IDMS System Tasks and Operator Commands Guide*.

6.4 New TCP_NODELAY Option

CA IDMS is enhanced with a new TCP/IP socket option, TCP_NODELAY. The TCP_NODELAY socket option disables the Nagle's algorithm, enabling two consecutive SEND socket functions to be executed without any delay between the two sends.

You can set the TCP_NODELAY socket option using the following methods:

- At startup, through a new system generation TCP/IP statement
- Dynamically, through a new DCMT VARY TCP/IP command
- At the user application level (assembler, COBOL, PL/I, and CA ADS), through a new option on the SETSOCKOPT socket function

The SETSOCKOPT socket function is described below. The two other methods are described in 6.3, “New TCP/IP System Entity” on page 198.

6.4.1 SETSOCKOPT Socket Function

Use the SETSOCKOPT socket function to explicitly set the TCP_NODELAY socket option for a user application, thereby overriding the default value.

EQUate Symbol	Field Name	Description
TCP@NODL	SOCKET-SOCKOPT-NODELAY	TCP_NODELAY option

This section describes only the new option. For more information about the SETSOCKOPT socket function, see the *CA IDMS Callable Services Guide*.

Notes

- The EQUate symbol is generated by the #SOCKET TCPIPDEF macro call and the field names are located in the SOCKET-MISC-DEFINITIONS record.
- For PL/I programs, the SOCKET_MISC_DEFINITIONS is used and the dashes are replaced by underscores.

6.5 New Socket Functions

CA IDMS is enhanced to support the following socket functions:

- GETSERVBYNAME
- GETSERVBYPORT
- IOCTL

These new socket functions provide the following capabilities:

- **GETSERVBYNAME and GETSERVBYPORT socket functions**—Let you retrieve the port number associated with a service name, or retrieve the service name, with all its aliases, associated with a specific port number.

  These new functions allow a user application to be independent of a specific port number. A user application program can make use of a service name and dynamically retrieve the corresponding port number to use at runtime using the GETSERVBYNAME socket function. The services file identified by the TCP/IP SYSGEN statement is used to resolve this mapping.

- **IOCTL socket function**—Lets you control or query the Application Transparent Transport Layer Security (AT-TLS), a facility on z/OS that allows the use of secured connections between different applications without the need to change the code of the application to encrypt and decrypt the data that is exchanged.

6.5.1 GETSERVBYNAME

GETSERVBYNAME takes a service name and a protocol and tries to resolve them using the services file. If successful, it returns the information in a SERVENT structure.

Assembler

```
Label    #SOCKET GETSERVBYNAME,
               RETCODE=return-code,
               ERRNO=errno,
               RSNCODE=reason-code,
               SERVNAME=service-name,
               SERVNAML=service-name-length,
               PROTNAME=protocol-name,
               PROTNAML=protocol-name-length,
               SERVENTP=serventp,
               PLIST=parameter-list-area,
               RGSV=(rgsv)
```

List of USING Parameters

```
SOCKET-FUNCTION-GETSERVBYNAME,
return-code,
errno,
reason-code,
service-name,
service-name-length,
protocol-name,
protocol-name-length,
serventp
```

6.5.1.1 Parameters

Parameter	Description
service-name	The name of an area containing the name of the service to resolve.
service-name-length	The name of a fullword field containing the length of *service-name*. *service-name-length* can be specified as an absolute expression. The maximum value for this parameter is 256.
protocol-name	The name of an area containing the name of the protocol to use.
protocol-name-length	The name of a fullword field containing the length of *protocol-name*. *protocol-name-length* can be specified as an absolute expression. The maximum value for this parameter is 256.
serventp	The name of a fullword field where the system returns the address of a SERVENT structure containing the information about the service.

6.5.1.2 Notes

- The services socket functions are supported by CA IDMS's internal services resolver. For more information, see 6.1.1, “CA IDMS Services Resolver” on page 192.
- The SERVENT structure area is allocated by the system and associated with a CA IDMS task. It is freed at task termination. It is reused by subsequent calls to a services function: GETSERVBYNAME or GETSERVBYPORT.

  Note: For more information about the SERVENT structure, see 6.5.4, “Socket Structure Description” on page 220.
- When the CASE sub-clause in the SERVICES FILE clause is defined as SENSITIVE, then the *service-name* and the *protocol-name* must be specified exactly as they are defined in the services file.

If it is defined as INSENSITIVE, the internal services resolver always tries to first retrieve the *service-name* and *protocol-name* as they are coded in the socket function call. If they are not found, the first entry where the uppercase versions of the service names and protocol names match are returned. In all cases, all the strings returned in the SERVENT structure are always coded as they appear in the services file.

6.5.2 GETSERVBYPORT

GETSERVBYPORT takes a port number and a protocol number and tries to resolve them using the services file. If successful, it returns the information in a SERVENT structure.

Assembler

```
Label    #SOCKET GETSERVBYPORT,
               RETCODE=return-code,
               ERRNO=errno,
               RSNCODE=reason-code,
               PORT=port-number,
               PROTNAME=protocol-name,
               PROTNAML=protocol-name-length,
               SERVENTP=serventp,
               PLIST=parameter-list-area,
               RGSV=(rgsv)
```

List of USING Parameters

```
SOCKET-FUNCTION-GETSERVBYPORT,
return-code,
errno,
reason-code,
port-number,
protocol-name,
protocol-name-length,
serventp
```

6.5.2.1 Parameters

Parameter	Description
port-number	The name of a fullword field containing the *port-number* to resolve.
protocol-name	The name of an area containing the name of the protocol to use.
protocol-name-length	The name of a fullword field containing the length of protocol-name. *protocol-name-length* can be specified as an absolute expression. The maximum value for this parameter is 256.

Parameter	Description
serventp	The name of a fullword field where the system returns the address of a SERVENT structure containing the information about the service.

6.5.2.2 Notes

- The services socket functions are supported by CA IDMS's internal services resolver. For more information, see 6.1.1, “CA IDMS Services Resolver” on page 192.
- The SERVENT structure area is allocated by the system and associated with a CA IDMS task. It is freed at task termination. It is reused by subsequent calls to a services function: GETSERVBYNAME or GETSERVBYPORT.

  Note: For more information about the SERVENT structure, see 6.5.4, “Socket Structure Description” on page 220.
- When the CASE sub-clause in the SERVICES FILE clause is defined as SENSITIVE, then the *service-name* and the *protocol-name* must be specified exactly as they are defined in the services file.

  If it is defined as INSENSITIVE, the internal services resolver always tries to first retrieve the *service-name* and *protocol-name* as they are coded in the socket function call. If they are not found, the first entry where the uppercase versions of the service names and protocol names match are returned. In all cases, all the strings returned in the SERVENT structure are always coded as they appear in the services file.

6.5.3 IOCTL

IOCTL controls certain characteristics of a socket. Depending on the command, it can retrieve or set control information.

Assembler

```
Label    #SOCKET IOCTL,
               RETCODE=return-code,
               ERRNO=errno,
               RSNCODE=reason-code,
               SOCK=socket-descriptor,
               COMMAND=command,
               ARGUMENT=argument,
               ARGUMENL=argument-length,
               PLIST=parameter-list-area,
               RGSV=(rgsv)
```

List of USING Parameters

```
SOCKET-FUNCTION-IOCTL,
return-code,
errno,
reason-code,
socket-descriptor,
command,
argument,
argument-length
```

6.5.3.1 Parameters

Parameter	Description
socket-descriptor	The name of a fullword field containing the *socket-descriptor* to process.
command	The name of a fullword field containing the *command* to perform on the socket. *command* can be specified as an absolute expression.
argument	The name of a fullword field containing the address of the *argument* area that is used by the corresponding *command*. The *argument* area usually contains input and output fields.
argument-length	The name of a fullword field that contains the length of the *argument* area.

The different commands and arguments allowed usually depend on the operating system where the CA IDMS system is running. For a full description of these parameters, see the corresponding socket API manual.

6.5.3.2 Notes

The following table lists the commands that can be specified. The EQUate symbol is generated by #SOCKET macro and the field names are associated with the SOCKET-MISC-DEFINITIONS-2 record.

EQUate Symbol	Field Name	Description
IO@NREAD	SOCKET-IOCTL-FIONREAD	Sets or resets socket in non-blocking mode
IO@NBIO	SOCKET-IOCTL-FIONBIO	Retrieves the number of readable bytes available
IO@CTTLS	SOCKET-IOCTL-SIOCTTLSCTL	Allows an application to query or control AT-TLS

PL/I programs: The SOCKET_MISC_DEFINITIONS_2 is used and the dashes are replaced by underscores.

More Information

For more information about socket functions, see the *CA IDMS Callable Services Guide*.

6.5.4 Socket Structure Description

6.5.4.1 SERVENT Structure

The SERVENT structure is returned by the GETSERVBYNAME and GETSERVBYPORT function calls.

Field	Description
Service name	Address of a service name (null-terminated string).
Aliases	Address of a zero-terminated array of pointers to aliases, which are null-terminated strings.
Port	Port number associated with a service.
Protocol	Address of the protocol associated with a service (null-terminated string).

6.6 DDS Connectivity Using TCP/IP

CA IDMS is enhanced to provide a new access method for CA IDMS DDS using the TCP/IP protocol through the SOCKET interface. This enhancement improves the performance of database requests to geographically distributed databases.

You can define the DDS connectivity using the following methods:

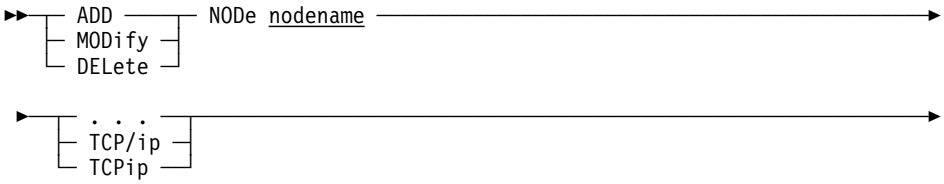
- Through a new parameter on the NODE SYSGEN statement and a new DDSTCPIP type on the PTERM statement from a SOCKET line
- Dynamically, through new parameters on the DCMT VARY PTERM command

6.6.1 System Generation NODE Statement

Use the system generation NODE statement to specify the use of TCP/IP to access a target node in the DC/UCF communications network.

Syntax

ADD/MODIFY/DELETE NODE Statement



Parameters

TCP/ip or TCPip
(DDS users only) Specifies that the TCP/IP protocol is used to access the named node.

More Information

For more information about the system generation NODE statement, see the *CA IDMS System Generation Guide*.

6.6.2 System Generation PTERM Statement

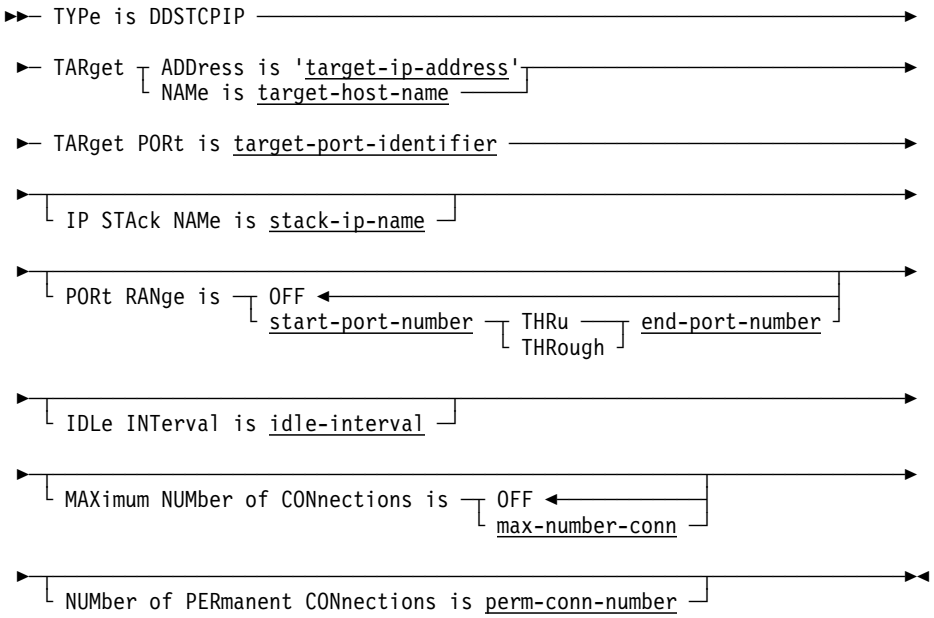
In support of DDS communications through TCP/IP, a new DDSTCPIP type of PTERM can now be associated with a SOCKET line and the maximum number of connections can be specified for a LISTENER PTERM.

In support of port number independence, the LISTENER PTERM statement has also been enhanced to allow the specification of a service name in place of a port number.

6.6.2.1 DDSTCPIP PTERM Statement

Use the DDSTCPIP PTERM statement in a SOCKET line to define the remote system where the target node is running.

Syntax



Parameters

TARget

The *target-ip-address* and *target-host-name* parameters are mutually exclusive. You must specify at least one of these parameters in the definition of a DDSTCPIP type PTERM.

target-ip-address

Specifies the IP address of the target system enclosed in single quotes. The IP address limit depends on whether IPv4 or IPv6 is used: IPv4 is 15 characters; IPv6 is 45 characters.

target-host-name

Specifies the host name of the target system. The maximum host name length is 64 characters.

target-port-identifier

Specifies the number of the target port or a service name. If *target-port-identifier* is a port number, it must be a positive number, between 1 and 65535. If *target-port-identifier* is a service name, it is limited to 32 characters and must be the name of a service in the services file with an associated protocol of TCP.

stack-ip-name

Specifies the job name of the TCP/IP stack to use in the local system. The job name is limited to 8 characters. Specify an empty string value (two single-quotes) to remove an IP STACK NAME definition.

start-port-number and end-port-number

Defines a range of port numbers that are used to BIND the local sockets explicitly. Each time a new connection is established, the first free port from the range is selected and associated (bound) with the corresponding socket. If no free port is found, the request is aborted.

The default value is OFF, indicating that the operating system will select a free port from the pool and bind the socket implicitly during the connect processing. *start-port-number* and *end-port-number* are positive numbers between 1 and 65535. *start-port-number* must be lower than or equal to *end-port-number*.

idle-interval

Defines the time interval a non-permanent connection stays in an idle state after the corresponding DDS request has finished. This allows the same connection to be reused if a new DDS request comes in before the timeout expires.

idle-interval is a positive number between 0 and 32767. The default value is 0.

max-number-conn

Defines the maximum number of active connections allowed from the local system.

max-number-conn is a positive number between 1 and 65535. The default value is OFF, indicating that the maximum number of connections is unlimited.

Note: The maximum number of connections depends on the number of free BULK PTERMs in the SOCKET line on the target (remote) system.

perm-conn-number

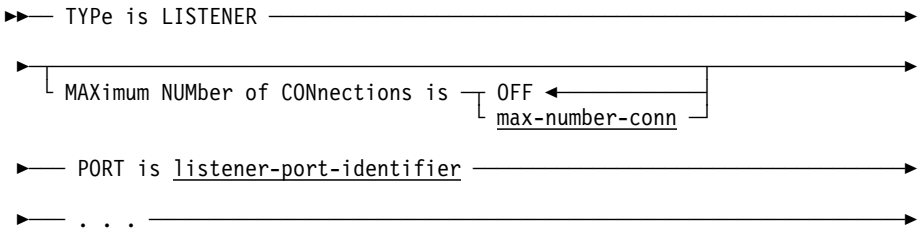
Defines the number of permanent connections that can exist between the host and the target systems.

perm-conn-number is a positive number between 0 and 65535. The default value is 0, indicating that permanent connections are not needed. In this case, the connections are always established dynamically when a new DDS request arrives.

6.6.2.2 LISTENER PTERM Statement

Use the LISTENER PTERM statement in a SOCKET line to control the number of active BULK PTERMs that can be started from a specific LISTENER PTERM and to alternatively specify a service name instead of a port number.

Syntax



Parameters

max-number-conn

Defines the maximum number of active connections that can be started from the corresponding listener program, that is, the maximum number of active BULK PTERMs allocated by the specific LISTENER. When the number of connections reaches the value specified for *max-number-conn*, any new connection accepted by the listener program will be rejected.

max-number-conn is a positive number between 1 and 65535. The default value is OFF, indicating that the maximum number of connections is unlimited.

listener-port-identifier

Specifies the number of the listener port or a service name. If *listener-port-identifier* is a port number, it must be a positive number between 1 and 65535. If *listener-port-identifier* is a service name, it is limited to 32 characters and must be the name of a service in the services file with an associated protocol of TCP.

6.6.2.3 More Information

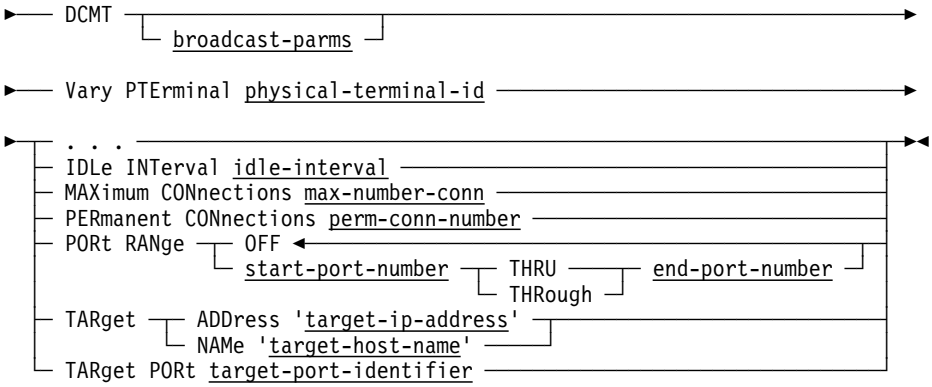
For more information about the system generation PTERM statement, see the *CA IDMS System Generation Guide*.

6.6.3 DCMT VARY PTERM Command

The DCMT VARY PTERM command has been enhanced to allow altering the attributes specific to a DDSTCPIP PTERM.

Note: Only a few of the parameters can be changed without requiring the corresponding PTERM to be varied OFFLINE first. For more information, see “Usage” on page 226.

Syntax



Parameters

IDLe INTerval idle-interval

Defines the time interval a non-permanent connection stays in an idle state after the corresponding DDS request has finished. This allows the same connection to be reused if a new DDS request comes in before the timeout expires.

idle-interval is a positive number between 0 and 32767. The default value is 0.

MAXimum CONnections max-number-conn

For a DDSTCPIP type PTERM, defines the maximum number of active connections allowed from the local system. For a LISTENER type PTERM, defines the maximum number of active BULK PTERM that can be started from that listener.

max-number-conn is a positive number between 1 and 65535. The default value is OFF, indicating that the maximum number of connections is unlimited.

Note: The maximum number of connections depends on the number of free BULK PTERMs in the SOCKET line on the target (remote) system.

PERmanent CONnections perm-conn-number

Defines the number of permanent connections that can exist between the host and the target systems.

perm-conn-number is a positive number between 0 and 65535. The default value is 0, indicating that permanent connections are not needed. In this

case, the connections are always established dynamically when a new DDS request arrives.

PORt RANge start-port-number and end-port-number
Defines a range of port numbers that are used to BIND the local sockets explicitly. Each time a new connection is established, the first free port from the range is selected and associated (bound) with the corresponding socket. If no free port is found, the request is aborted.

The default value is OFF, indicating that the operating system will select a free port from the pool and bind the socket implicitly during the connect processing. *start-port-number* and *end-port-number* are positive numbers between 1 and 65535. *start-port-number* must be lower than or equal to *end-port-number*.

TARget ADDress target-ip-address
Specifies the IP address of the target system enclosed in single quotes. The IP address limit depends on whether IPv4 or IPv6 is used: IPv4 is 15 characters; IPv6 is 45 characters.

TARget NAMe target-host-name
Specifies the host name of the target system. The maximum host name length is 64 characters.

TARget PORt target-port-identifier
Specifies the number of the target port or a service name. If *target-port-identifier* is a port number, it must be a positive number between 1 and 65535. If *target-port-identifier* is a service name, it is limited to 32 characters and must be the name of a service in the services file with an associated protocol of TCP.

Usage

LISTENER or DDSTCPIP type PTERM OFFLINE Requirements: The following table contains the parameters that are accepted for a LISTENER or DDSTCPIP type PTERM. The last column indicates if the owning PTERM must be OFFLINE to allow the corresponding parameter to be changed dynamically.

	LISTENER PTERM	DDSTCPIP PTERM	PTERM OFFLINE
BACKLOG	X		X
IDLE INTERVAL		X	
MAXIMUM CONNECTIONS	X	X	
MODE SYSTEM/USER	X		X
PARM	X		X

	LISTENER PTERM	DDSTCPIP PTERM	PTERM OFFLINE
PERMANENT CONNECTIONS		X	
PORT	X		X
PORT RANGE OFF		X	
PORT RANGE <range>		X	*
TARGET ADDRESS		X	X
TARGET NAME		X	X
TARGET PORT		X	X
TASK	X		X
TCP/IP ADDRESS	X		X
TCP/IP NAME	X		X
TCP/IP STACK	X	X	X

***** If the corresponding PTERM is ONLINE, the <range> value can be changed dynamically only if the port range parameter was not assigned to OFF at the time the PTERM was opened.

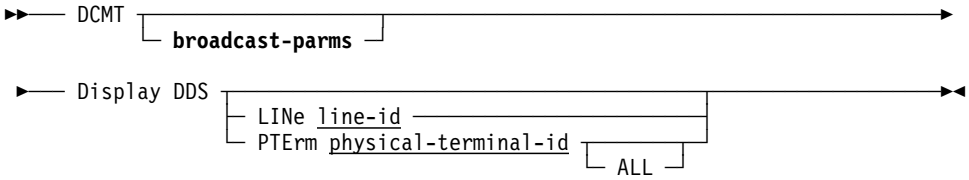
More Information

For more information about the DCMT VARY PTERM command, see the *CA IDMS System Tasks and Operator Commands Guide*.

6.6.4 DCMT DISPLAY DDS Command

Use the DCMT DISPLAY DDS command to display general information about the DDS network or about a particular DDS line or physical terminal. It can now display a DDSTCPIP type PTERM, if present.

Syntax



Parameters

PTErm

Displays information for the specified DDS physical terminal.

physical-terminal-id

The ID of a physical terminal defined on the system generation PTERM statement.

ALL

(DDSTCPIP type PTERM only) Displays a list of all TCP/IP connections with its owning LTERM, the corresponding expiration time (if the connection is in the idle list only), and the local port used.

Examples

DCMT DISPLAY DDS

```
   ***     Display DDS     ***
  Line     PTerm          Node Name  Weight    BLKSIZE
DDSVTAM   PDDSVT99                        30       8192
          PDDSVT73                        12      28000
          PDDSVT74                        12      16500
          PDDSVX73                        12      28000
          PDDSVX74                        12      16500
          PDDSVX71                        12       8176
          PDDSVT71        SYSTEM71        12       8192

  Line     PTerm          Node Name
SOCKET    SY71CA31        SYSTEM71
          SY71CA11
```

DCMT DISPLAY DDS PTERM ddstcpip-pterm-id ALL

```
PTERM definitions                         Run-time information
======================================    ======================================
PTERM name                      SY71CA31  Target IDMS node              SYSTEM71
LTERM name                      SY71CA31  Number connections requested        54
Line name                       SOCKET    Number connections created           4
IP stack name                   *DEFAULT  Number connections active            4
Target host                     USILCA31  HWM connections in-use               4
Target port                         3771  Number connections found in
Port range                           OFF    * permanent list                  45
Maximum connections                  OFF    * idle list                        5
Permanent connections                  1  Number retry for free port           0
Idle interval                         60  Number connections rejected
                                            * max connection                   0
                                            * no free port                     0
                                            * socket error                     0

TCP/IP connections  Owning LTERM  Expiration time  Local port
==================  ============  ===============  ==========
Control connection    SY71CA31         n/a            2138
In-use list           LD000001         n/a            2152
Permanent list        SY71CA31         n/a            2161
Idle list             SY71CA31          45            2165
```

Usage: DCMT DISPLAY DDS PTERM ddstcpip-pterm-id ALL displays global information and statistics about a specific DDSTCPIP type PTERM. The display includes the following PTERM definitions and run-time and ALL option information:

Field	Value
PTERM Definitions	
PTERM name	Name of the DDS physical terminal
LTERM name	Name of the DDS logical terminal
Line name	Name of the line with which the physical terminal is associated
IP stack name	Job name of the TCP/IP stack in the local system
Target host	Host name of the target system
Target port	Target port number or service name
Port range	Range of port numbers
Maximum connections	Maximum number of active connections allowed from the local system
Permanent connections	Number of permanent connections between the host and the target systems

Field	Value
Idle interval	Time interval that the non-permanent connection stays in an idle state after the corresponding DDS request has finished
Run-time Information	
Target IDMS node	Name of the CA IDMS node in the target (remote) system
Number connections requested	Number of DDS requests that have already been processed to the target system. Each DDS request is processed through one TCP/IP connection.
Number connections created	Number of connections that have been created to satisfy all the DDS requests
Number connections active	Number of connections currently active between the client system and the remote system
HWM connections in-use	Maximum number of connections that are processing DDS requests concurrently
Number connections found in permanent list/idle list	Number of times a free connection could be found in the permanent list or idle list to process a DDS request. A small number in these fields in comparison with the value displayed for the Number connections created field indicates that you may want to increase the definitions for NUMber of PERmanent CONnections or IDLe INTerval parameters in SYSGEN.
Number retry for free port	Number of times the system had to retry to find a free port number from the port range defined at the PTERM level. This occurs only when a port from the port range is in use by another application in the system.
Number connections rejected	Number of times the creation of a connection has been rejected. A rejection is caused by one of the following: ■ The maximum number of active connections was reached ■ No free port could be found in the port range ■ A socket call error (usually returned after an error at the remote system)

Field	Value
ALL Option Information	
TCP/IP connections	A type or list owning the connection as follows: ■ Control connection always describes the control connection between the local and remote systems. It is reserved for the system. ■ In-use list indicates that the corresponding LTERM is currently processing a DDS request. ■ Permanent list indicates that the corresponding connection is free and thus ready to be assigned to a LTERM to process a DDS request. ■ Idle list indicates that the corresponding connection has been freed and remains in the list for the number of seconds currently displayed in the Expiration time column. When the time has expired, the connection is closed.
Owning LTERM	Name of the LTERM owning the connection
Expiration time	This field applies only to connections belonging to the Idle list. It indicates the remaining time, in seconds, before the corresponding connection is closed. The maximum value for this field is the value assigned to the IDLe INTerval parameter in SYSGEN.
Local port	Port number used at the local side of the connection

More Information

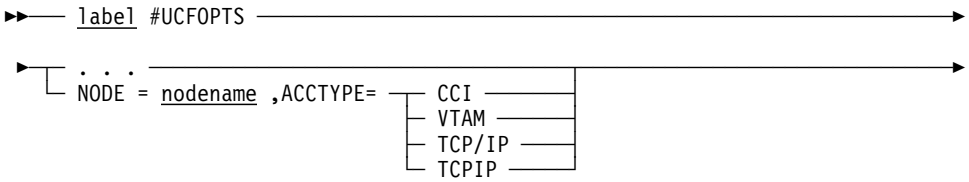
For more information about the DCMT DISPLAY DDS command, see the *CA IDMS System Tasks and Operator Commands Guide*.

6.6.5 DC Front-end System

The UCF DC front-end enables a terminal on one DC system to execute tasks on a second DC system.

The ACCTYPE parameter on the #UCFOPTS macro can now accept TCP/IP parameters.

Syntax



Parameters

NODE=/ACCTYPE=

Identifies the back-end when access is via DC/DDS:

- NODE=nodename specifies the one- to eight-character name of a system defined to the DC/DDS communication network.
- ACCTYPE=CCI/VTAM/TCPIP specifies the type of DDS communication that is to be used for the UCF connection.

Note: This parameter applies only to DDS.

More Information

For more information about UCF support for a DC front-end, see the *CA IDMS System Operations Guide*.

Chapter 7. Administrative and Operational Enhancements

This chapter describes administrative and operational enhancements. It contains the following topics:

7.1 Callable Security Cleanup

The linkable RHDCSDEL enhancement allows a user program to clean up security definitions for logically deleted users by linking to RHDCSDEL. To use this feature, you must write a user program that links to RHDCSDEL.

Note: Securing the SDEL task code does not secure usage of the RHDCSDEL program. If you want to limit the use of RHDCSDEL, that program must be secured.

RHDCSDEL LINK Statement

The calling program links to program RHDCSDEL, passing the addresses DICTNAME, RETCODE, and OUTAREA as parameters:

```
#LINK PGM='RHDCSDEL',PARMS=(DICTNAME,RETCODE,OUTAREA)
```

Parameters

DICTNAME

Specifies the dictionary name of the DDLDML and DDLCAT areas to be scanned for security definitions associated with logically deleted users.

This is an 8-character field, left-justified, and padded with blanks. If DICTNAME is set to blanks, DC/UCF processes the DDLDML and DDLCAT areas of the default dictionary for the system. If DICTNAME is set to CL8'*ALL', all updatable DDLDML and DDLCAT areas in the DMCL are processed.

RETCODE

Specifies a fullword in which RHDCSDEL provides a return code. The possible return codes are as follows:

00

Specifies processing was successful. The OUTAREA contains informational messages DC048004 and DC048008.

04

Specifies processing was successful but contains warnings. The possible causes are as follows:

- There were no logically deleted users to process. The OUTAREA contains informational message DC048002.
- The OUTAREA is too small to contain all output messages.

08

Specifies a processing error. The possible causes are as follows:

- The DICTNAME is invalid. The outarea contains error message DC048001
- An unexpected database error was encountered. The OUTAREA contains error message DC048003.

- A BIND failed. The OUTAREA contains error message DC048004 or DC048006.

12

Specifies the fatal error, the DMCL module is invalid. The OUTAREA contains error message DC048007.

OUTAREA

Specifies an area where RHDCSDEL puts messages. The first fullword of the area must be initialized to the area length, which also includes the first full word. Upon return, the first fullword contains the size of the messages. Each message is in the following format:

```
AL1(L'message),C'message'
```

Note: The RETCODE is set to 04 if the output area is too small, unless a more severe error occurred.

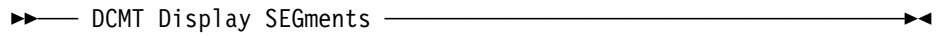
More Information

- For more information about the LINK statement, see the *CA IDMS DML Reference Guide* for the language of the calling program.
- For more information about securing a program, see the *CA IDMS Security Administration Guide*.

7.2 DISPLAY SEGMENT Enhancement

The DCMT DISPLAY SEGMENT command is enhanced to include the number of areas in the segment in the output display.

Syntax



Example

The following example shows the number of areas for each of the segments displayed.

DCMT DISPLAY SEGMENTS

```
Segment-Name   Schema-Name     Type  #areas  Pg-Grp  Radix      Datetime-stamp
  AAA                       Network       1      25      8    2005-03-29-10.07.59
  DAR                           SQL       3       0      8    2005-03-29-10.07.59
  DBCR                      Network       2      15      8    2005-03-29-10.07.59
  EMPDEMO                   Network       3       0      8    2005-03-29-10.07.59
  ETOT                      Network       1      32      8    2005-03-29-10.07.59
  KJM                       Network      30      35      8    2005-03-29-10.07.59
  LRD                       Network       1      30      8    2005-03-29-10.07.59
  QADICT                    Network       2       0      8    2005-03-29-10.07.59
  QAMISC                    Network       1       0      8    2005-03-29-10.07.59
  R120DICT                  Network       2       0      8    2005-03-29-10.07.59
  SYSDAR                        SQL       3       0      8    2005-03-29-10.07.59
  SYSDEF                    Network       5       0      8    2005-03-29-10.07.59
  SYSDICT                   Network       2       0      8    2005-03-29-10.07.59
  SYSLOCAL                  Network       1       1      8    2005-03-29-10.07.59
  SYSMSG                    Network       1       0      8    2005-03-29-10.07.59
  SYSSQL                        SQL       3       0      8    2005-03-29-10.07.59
  SYSUSER                   Network       1       0      8    2005-03-29-10.07.59
  USERDB                        SQL       3       0      8    2005-03-29-10.07.59
  USERDB2                       SQL       3       2      8    2005-03-29-10.07.59
  VSAMT                     Network       6       0      8    2005-03-29-10.07.59
V74  ENTER NEXT TASK CODE:        CA IDMS release nn.n tape volser node SYSTEM74
```

More Information

For more information about the DCMT DISPLAY SEGMENT command, see the *CA IDMS System Tasks and Operator Commands Guide*.

7.3 Enhanced Diagnostic Information

This section describes a number of improvements in the detection and reporting of exceptional conditions in order to facilitate problem diagnosis and correction.

7.3.1 Display Data at the PSW

The 32 bytes of data at the PSW (16 bytes before and 16 bytes after) is included in the #ACEDS (TCE ACE) and is displayed for system and task snaps when the abend control element (ACE) control block is formatted and snapped, and the PSW address is a valid storage address.

7.3.2 GETMAIN Failure Message for Buffers

A GETMAIN command can fail at startup when the system is unable to acquire storage for the database buffers from operating system storage. A new message is issued indicating that the database buffers are being allocated from IDMS storage pools. The message has the following format:

DC205029 Unable to allocate buffer in OPSYS storage. Trying IDMS storage pools.

7.3.3 Identification of Program Filling Journal

A new message is issued identifying the transaction that is filling the journal files without issuing commits. This message displays the transaction's program name, subschema name, and the number of BFOR image bytes written on behalf of the transaction. The message has the following format:

DC205030 LID=<*Local-Transaction-id*> PROG=<*Program-Name*> SUBS=<*Subschema-Name*> BFOR=<*BFOR-Journal-Space-Usage*>

It displays when the existing messages are displayed:

DC205003 Disk Journal is FULL. Submit ARCHIVE JOURNAL for <*journal-file-name*>

DC205024 Journal Write waiting on full Journal

Note: For more information, see the *CA IDMS Messages and Codes Guide*.

7.3.4 IDMSINTC CWADISP ABND Message

The IDMSINTC interface is enhanced to issue a new ABNDK007 error message when the CWADISP value is greater than the CWASIZE value. Previously, unpredictable results occurred.

7.3.5 IDMSINTC Maximum Run Units ABND Message

The IDMSINTC interface is enhanced to issue a new ABNDK214 error message when the number of active run units exceeds the value specified in the CICSOPTS USERCNT parameter. Previously, an AKEA abend occurred.

7.3.6 Journal Warning Message at Startup

A new warning message is issued at startup when the number of journal area entries is within 10% of the maximum number of areas that the JHDA headers can accommodate. This message identifies the number of used area entries and the number of available area entries that the journal can hold. The message has the following format:

DC205031 Warning - <*number of area entries in JHDAs*> area entries in journal header nearing max of <*total entries*>

7.3.7 Validation and Shutdown Sysplex Messages

The following messages are now issued in a data sharing environment to track certain events:

- DB347052 message while doing area validation
- DC200*nnn* messages during CV shutdown

7.3.8 VTAM Enhanced Error Reporting

For improved VTAM error reporting, the VTAM feedback code and return code have been added to DC error message DC084109 as follows:

DC084109 PTERM <*pterm-id*> ON LINE <*line-id*> : SIMLOGON TO VTAM NODE <*vtam-node-name*> MODEENT <*vtam-modeent-name*> FAILED, FDBK <*vtam-feedback*> SENSE: <*vtam-sense-code*>

The additional data is returned for all VTAM-SNA lines that fail to open, resulting in the DC084109 error message.

Note: For more information, see the *CA IDMS Messages and Codes Guide*.

7.3.9 XCF and XES Messages Written to Log

Messages relating to XCF/XES macro requests issued by CA IDMS are now written to the DC log after it has been opened, in addition to being written to the console/system log.

The DC215999 message has the following format:

DC215999 <*macro*> RC=<*retcode*> Reason=<*reason*> name=<*sname*>

This message indicates that the DC/UCF system has issued the macro request identified in the message text to the IBM Parallel Sysplex system. The IBM Parallel Sysplex system processes the request and returns a return code and reason code for the named structure also identified in the message text.

Note: For more information, see the *CA IDMS Messages and Codes Guide*.

7.4 External Identity Auditing

An external identity represents the end user of an application that uses a generic internal user id to sign on to CA IDMS. The actual end user id is commonly stored as a value but not used to authenticate access to the database. This technique is often used in web applications. The CA IDMS Server r16.1 JDBC driver supports an external identity audit feature that records the external identity in the journal when a database record is updated on the entry CV.

CA IDMS r17 extends this feature by setting the external identity as a session profile attribute and broadcasting it on remote database connections. This makes the external identity visible to customer applications and ensures that it can be audited on all CVs that take part in a transaction.

Note: The external identity is not propagated to or from an r16 CV. All CV's must be r17 or later to successfully audit external identities on remote databases.

The external identity can be set in the following ways:

- The CA IDMS JDBC driver obtains the identity of the user from the SiteMinder Application Server Agent for J2EE applications managed by CA SiteMinder.
- Standalone Java applications use a CA IDMS extension to the JDBC API to set the external identity.
- Online CA IDMS applications set the external identity as a session profile attribute.

Journal reports are used to audit external user identities.

7.4.1 Profile Attribute Key

The external user identity is set in the EXTIDENT session profile attribute. This attribute is treated as if it were a built-in system profile attribute. EXTIDENT is now a reserved attribute name.

As a profile attribute, the value of EXTIDENT can be established through a signon or user profile or interactively by issuing a DCUF SET command. However, it is most usefully established programmatically in one of the ways described below.

Usage

Java Applications: The session attribute is set by the JDBC driver for Java applications that use CA IDMS Server. This is described in the *CA IDMS Server r16.1 User Guide*.

CA IDMS/DC Applications: CA IDMS/DC applications can use the IDMSINO1 SETPROF function to set the session profile attribute and can use the IDMSINO1 GETPROF function to get the current value of the attribute.

When this attribute is set in the current user session profile it is also sent to all remote systems that are associated with the user session. The return code is set to the highest error encountered. A nonzero return code indicates that the external identity may not have been set on one or more CVs.

The DBA can disable this feature, either at the system or user level, by setting the EXTIDENT attribute to blank and specifying that it cannot be overridden.

SQL Applications: SQL applications can use the PROFILE scalar function to retrieve the current value of the attribute. SQL called procedures can use the IDMSINO1 GETPROF function to get the current value of the attribute.

7.4.2 Journal Reports

The following journal reports are used to audit external user identities.

7.4.2.1 Journal Analyzer Chronological Event Report

The area of the report for the BGIN record includes the external user identity. If present, it is included on the line that reports the user id.

```
---------EVENT--------- --------IDENT-------- ---QUIESCE LVL/USER/EXT
ID-- . . . .

TIME     TYPE  DURATION RUN UNIT      PROGRAM     . . .

hh:mm:ss BGIN                1633  JAVAPROG ONL X . . .

                                          <user id>         <external id>
```

7.4.2.2 JREPORT 000

JREPORT 000 supports a REC card for the external identity field to allow the use the external identity as selection criteria when running existing JREPORTs.

7.4.2.3 JREPORT 008

JREPORT 008 displays the external identity information when reporting the BGIN record. The external id heading and value are only printed when available.

```
BGIN   TECHDC30  12/27/05  20.07.13.56         529276      170604 ... ... .

       USER ID                          EXTERNAL ID

       USER01                           USER2007
```

7.4.2.4 JREPORT 010

JREPORT 010 lists the user id, external identity, date, time, program name, and run unit id. This information provides a customer with enough information to run 'JREPORT 008' with SELECT criteria to provide details on all activity involving a particular user.

```
REPORT NO. 10                       IDMS JOURNAL REPORTS                      R16.0
JREPORT 010                  EXTERNAL USER IDENTITY JOURNAL REPORT
  USER                 EXT                TRANSACT  PROGRAM    LOCAL     LOCAL
   ID                   ID                  IDX      NAME      DATE      TIME


 USER01               USER2006                 5  IDMSJDBC  12/22/05  14.19.29.66
 USER ID NOT CAPTURED EXT ID NOT CAPTURED      6  IDMSDDDL  12/22/05  14.19.29.67
 USER01               EXT ID NOT CAPTURED      7  RHDCRUAL  12/22/05  14.19.29.68
 NO USER SIGNON                                8  RHDCRUAL  12/22/05  14.19.29.69
 USER01               USER123                  9  JAVAPROG  12/22/05  14.19.29.70

C750009 RECORDS WRITTEN FOR REPORT 10 --      8
```

7.5 IDD Display Load Modules by Type

IDD is enhanced to be able to do a DISPLAY ALL LOAD MODULES WHERE TYPE IS '*load-module-type*'. This displays only the load modules for the specified type.

Example

The following example shows only the load modules for load module type SUBSCHEMA:

```
DISPLAY ALL LOAD MODULES WHERE TYPE IS 'SUBSCHEMA' .
*+   DISPLAY LOAD MODULE NAME IS KJMTEST3 VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS KJMTEST VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS QA120SS2 VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS QA120SS1 VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS EMPSS01 VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS SS1 VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS VSUB01 VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS DBCRSSC1 VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS TOM2 VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS TOM VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS RXMLRF05 VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS RXMLRF04 VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS EMPL1 VERSION IS 1 .
*+   DISPLAY LOAD MODULE NAME IS JPD VERSION IS 1 .
000003*+ I DC601157  NO MORE ENTITY OCCURRENCES FOUND
```

Note: For more information, see the *CA IDMS IDD DDDL Reference Guide*.

7.6 Index Tuning Enhancements

Index tuning is enhanced in the following areas:

- **Index Structure Reporting**—The PRINT INDEX utility now provides the ability to better determine whether an index structure needs tuning. Additionally, you can now report on a specific occurrence of a user-owned index, thus providing an additional debugging aid.
- **Index Tuning**—The TUNE INDEX utility now provides the ability to perform more comprehensive index tuning as follows:
  - Eliminate orphans at all levels within an index
  - Relocate the top-level SR8 to its optimal location
  - Optionally, rebalance an index with temporary values for IBC and PAGE RESERVE
  - Optionally, resequence an index with temporary values for IBC and PAGE RESERVE

  TUNE INDEX is also enhanced in its ability to tune indexes while they remain available to online applications.

7.6.1 PRINT INDEX

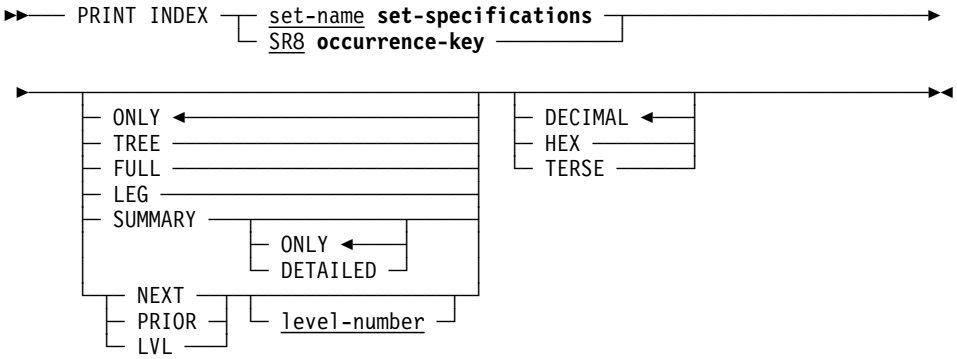
The PRINT INDEX utility reports on the structure of an indexed set. Using the PRINT INDEX utility, you can review:

- The number of levels in an index.
- The contents of the fixed and variable portions of one or more SR8 records in an index.
- The amount of available space on the page containing each SR8 in an index.
- The index structure efficiency.

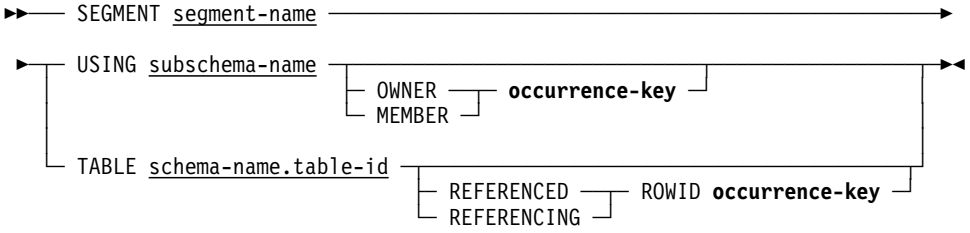
Authorization:

To	You Need This Privilege	On
Report on an index	DBAREAD	The area containing the index and the area(s) containing records referenced by the index

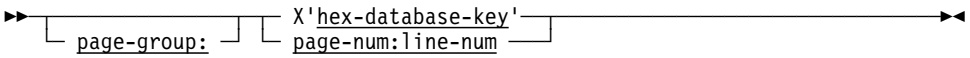
Syntax



Expansion of set-specifications



Expansion of occurrence-key



Parameters

SUMMARY

Requests a summary report for the target index. A summary report consists of three parts:

- Part 1 (header) provides general information on the index definition.
- Part 2 (main body) provides information on index owner occurrence(s). A system-owned index contains a single index owner; a user-owned index can contain more than one index owner.
- Part 3 (index overview) provides global statistical information for a user-owned index only.

A summary report on a system-owned index contains parts 1 and 2.

A summary report on a user-owned index always contains parts 1 and 3. Part 2 is included only in a detailed summary report.

ONLY

Requests a summary report with parts 1 and 3 for the target user-owned index. This parameter is ignored for a system-owned index. ONLY is the default.

DETAILED
Requests a summary report with parts 1, 2, and 3 for the target user-owned index. This parameter is ignored for a system-owned index.

REFERENCED ROWID
For the named table, directs the PRINT INDEX utility to report on the index occurrence whose owner is the referenced row identified by *occurrence-key*.

REFERENCING ROWID
For the named table, directs the PRINT INDEX utility to report on the index occurrence containing the row ID of the referencing row identified by *occurrence-key*.

7.6.1.1 Usage

How to submit the PRINT INDEX statement: You submit the PRINT INDEX statement by using the batch command facility or the online command facility.

When to use PRINT INDEX: The PRINT INDEX utility can help you determine whether an index needs to be rebuilt. For example, you should consider rebuilding an index when the PRINT INDEX utility report on the index indicates one of the following:

- The number of index levels is greater than anticipated for the original index structure.
- Twenty-five percent or more of the member records are orphans.

An index can be rebuilt using MAINTAIN INDEX or TUNE INDEX. For more information about index rebuilding and indexing in general, see the *CA IDMS Database Administration Guide*.

Note: The output of PRINT INDEX without the SUMMARY parameter is proportional to the number of index members that are being reported. If PRINT INDEX is run online or in batch through CV, the output is buffered in scratch. If the scratch area cannot contain all the output, PRINT INDEX fails with a task abend.

Hexadecimal display of symbolic keys: The HEX parameter of the SET/SR8 statement is useful when the symbolic key for the index is a non-displayable data type, such as binary or packed.

7.6.1.2 Examples

Printing a summary report of an index: The following example directs the PRINT utility to report on the DEPT_EMPL index using the SUMMARY option.

```
PRINT INDEX DEPT_EMPL SEGMENT USERDB TABLE DEMO.DEPT SUMMARY;
```

Printing a REFERENCING ROWID summary report of an index: The following example directs the PRINT utility to report on the index occurrence containing the row ID of the referencing row identified by X'0013D401'.

```
PRINT INDEX DEPT_EMPL SEGMENT USERDB TABLE DEMO.DEPT
   REFERENCING ROWID X'0013D401'  SUMMARY;
```

7.6.1.3 Sample Output

Printing a summary report of an index: The report below illustrates the use of the SUMMARY option to request the printing of a user-owned index.

```
  PRINT INDEX DEPT_EMPL SEGMENT USERDB TABLE DEMO.DEPT SUMMARY;
SET Name: DEPT_EMPL
   IBC 3                                Displacement        2
   Sort option SORTED SYM ASC           Key length          10
   Duplicates FIRST                     Compression         No
OWNER: DEPT
   AREA USERDB.ORG_AREA                 Low page            5051
   Page size 1024                       High page           5100
MEMBER: EMPL                            Set membership      Optional Manual
   Located VIA index No                 Index is            Linked
   AREA USERDB.EMP_AREA                 Low page            5001
   Page size 1024                       High page           5050

Index overview
   Nr of owner occurrences                      5
   Nr of empty owners                           1      20.0%
   Nr of displaced top level SR8s               1      20.0%
   Nr of SR8s:              Total              62
                          Average            12.4
                          Highest              59      Owner X'0013D401'
   Min. nr of SR8s:         Total              49
                          Average             9.8
                          Highest              46      Owner X'0013D401'
   Nr of levels:          Average             1.6
                          Highest              5      Owner X'0013D401'
   Min. nr of levels:     Average             1.6
                          Highest              5      Owner X'0013D401'
   Nr of pages:           Average             2.2
                          Highest              8      Owner X'0013D401'
   Min. nr of pages:      Average             1.8
                          Highest              6      Owner X'0013D401'
   Nr of occurrences with orphans              1
   Nr of Orphans:           Total              42      27.8%
                          Highest              42      Owner X'0013D401'
   Total size of all SR8s                    5784
   Size of largest SR8                        104
```

```
Distribution of Index Levels
       ....+....20...+....40...+....60...+....80...+....
    6+|                                                   0    0.0%
    5 |**********                                         1   20.0%
    4 |                                                   0    0.0%
    3 |                                                   0    0.0%
    2 |                                                   0    0.0%
    1 |******************************                     3   60.0%
    0 |**********                                         1   20.0%

Distribution of Minimum Index Levels
       ....+....20...+....40...+....60...+....80...+....
    6+|                                                   0    0.0%
    5 |**********                                         1   20.0%
    4 |                                                   0    0.0%
    3 |                                                   0    0.0%
    2 |                                                   0    0.0%
    1 |******************************                     3   60.0%
    0 |**********                                         1   20.0%

Distribution of Number of SR8s
       ....+....20...+....40...+....60...+....80...+....
   60+|                                                   0    0.0%
   56 |**********                                         1   20.0%
   52 |                                                   0    0.0%
   48 |                                                   0    0.0%
   44 |                                                   0    0.0%
   40 |                                                   0    0.0%
   36 |                                                   0    0.0%
   32 |                                                   0    0.0%
   28 |                                                   0    0.0%
   24 |                                                   0    0.0%
   20 |                                                   0    0.0%
   16 |                                                   0    0.0%
   12 |                                                   0    0.0%
    8 |                                                   0    0.0%
    4 |                                                   0    0.0%
    1 |******************************                     3   60.0%
    0 |**********                                         1   20.0%

Distribution of Number of Index Members
       ....+....20...+....40...+....60...+....80...+....
   90+|                                                   0    0.0%
   85 |**********                                         1   20.0%
   80 |                                                   0    0.0%
   75 |                                                   0    0.0%
   70 |                                                   0    0.0%
   65 |                                                   0    0.0%
   60 |                                                   0    0.0%
   55 |                                                   0    0.0%
   50 |                                                   0    0.0%
   45 |                                                   0    0.0%
   40 |                                                   0    0.0%
   35 |                                                   0    0.0%
   30 |                                                   0    0.0%
   25 |                                                   0    0.0%
   20 |                                                   0    0.0%
   15 |                                                   0    0.0%
   10 |                                                   0    0.0%
    5 |                                                   0    0.0%
    1 |******************************                     3   60.0%
    0 |**********                                         1   20.0%
```

```
Distribution of Estimated IOs for Sequential Bottom Level Access Using 1 Buffer
       ....+....20...+....40...+....60...+....80...+....
   28+|                                                    0    0.0%
   27 |**********                                          1   20.0%
   26 |                                                    0    0.0%
   25 |                                                    0    0.0%
   24 |                                                    0    0.0%
   23 |                                                    0    0.0%
   22 |                                                    0    0.0%
   21 |                                                    0    0.0%
   20 |                                                    0    0.0%
   19 |                                                    0    0.0%
   18 |                                                    0    0.0%
   17 |                                                    0    0.0%
   16 |                                                    0    0.0%
   15 |                                                    0    0.0%
   14 |                                                    0    0.0%
   13 |                                                    0    0.0%
   12 |                                                    0    0.0%
   11 |                                                    0    0.0%
   10 |                                                    0    0.0%
    9 |                                                    0    0.0%
    8 |                                                    0    0.0%
    7 |                                                    0    0.0%
    6 |                                                    0    0.0%
    5 |                                                    0    0.0%
    4 |                                                    0    0.0%
    3 |                                                    0    0.0%
    2 |                                                    0    0.0%
    1 |******************************                      3   60.0%
    0 |**********                                          1   20.0%

Distribution of Nr of Pages with Intermediate Level SR8s
       ....+....20...+....40...+....60...+....80...+....
    7+|                                                    0    0.0%
    6 |**********                                          1   20.0%
    1+|                                                    0    0.0%
    0 |****************************************            4   80.0%

Distribution of Minimum Nr of Pages with Intermediate Level SR8s
       ....+....20...+....40...+....60...+....80...+....
    4+|                                                    0    0.0%
    3 |**********                                          1   20.0%
    1+|                                                    0    0.0%
    0 |****************************************            4   80.0%

Distribution of % Displaced Intermediate Level SR8s
       ....+....20...+....40...+....60...+....80...+....
   48+|**********                                          1   20.0%
    1+|                                                    0    0.0%
    0 |****************************************            4   80.0%

Distribution of Nr of Pages with Bottom Level SR8s
       ....+....20...+....40...+....60...+....80...+....
    7+|                                                    0    0.0%
    6 |**********                                          1   20.0%
    5 |                                                    0    0.0%
    4 |                                                    0    0.0%
    3 |                                                    0    0.0%
    2 |                                                    0    0.0%
    1 |******************************                      3   60.0%
    0 |**********                                          1   20.0%
```

```
Distribution of Minimum Nr of Pages with Bottom Level SR8s
       ....+....20...+....40...+....60...+....80...+....
    4+|                                                    0    0.0%
    3 |**********                                          1   20.0%
    2 |                                                    0    0.0%
    1 |******************************                      3   60.0%
    0 |**********                                          1   20.0%

Distribution of % Displaced Bottom Level SR8s
       ....+....20...+....40...+....60...+....80...+....
    1+|                                                    0    0.0%
    0 |**************************************************  5  100.0%
Status = 0        SQLSTATE = 00000
```

Printing a REFERENCING ROWID summary report of an index: The report below illustrates the use of the REFERENCING ROWID option to request the printing of the index occurrence containing the row ID of the referencing row identified by X'0013D401'.

```
  PRINT INDEX DEPT_EMPL SEGMENT USERDB TABLE DEMO.DEPT
     REFERENCING ROWID X'0013D401'  SUMMARY;
SET Name: DEPT_EMPL
   IBC 3                              Displacement        2
   Sort option SORTED SYM ASC         Key length          10
   Duplicates FIRST                   Compression         No
OWNER: DEPT
   AREA USERDB.ORG_AREA               Low page            5051
   Page size 1024                     High page           5100
MEMBER: EMPL                          Set membership      Optional Manual
   Located VIA index No               Index is            Linked
   AREA USERDB.EMP_AREA               Low page            5001
   Page size 1024                     High page           5050

OWNER X'0013D401' on page 5076
   Top level SR8  on page 5079        utilization    100.0%
Intermediate Level
   Nr of SR8s                                26                17 Minimum
   Nr of pages with SR8s                      6                 3 Minimum
   Nr of displaced SR8s                      15      57.6%
   Nr of entries in use                      58      74.3%
   Nr of Orphans                             18      31.0%
   Total size of all SR8s                  2704
Bottom       Level
   Nr of SR8s                                33                29 Minimum
   Nr of pages with SR8s                      6                 3 Minimum
   Nr of displaced SR8s                       0       0.0%
   Nr of entries in use                      87      87.8%
   Nr of Orphans                             24      27.5%
   Total size of all SR8s                  2844
Index occurrence totals
   Nr of members                             87
   Nr of levels                               5                 5 Minimum
   Size of largest SR8                      104
   Nr of SR8s                                59                46 Minimum
   Nr of pages with SR8s                      8                 5 Minimum
   Nr of displaced SR8s                      15      25.4%
   Nr of entries in use                     145      81.9%
   Nr of Orphans                             42      28.9%
   Total size of all SR8s                  5548

Nr of Buffers versus Estimated IOs for Sequential Bottom Level access
-------------       -------------
     1                         27
     2                         21
     3                         15
     4                         10
     5                          8
     6  - 20                    6
Status = 0        SQLSTATE = 00000
```

Report Output Description

- **Part 1—Header**

  The report header provides general information on the index definition, the index owner record or SQL table, and the index member record or SQL table.

- **Part 2—Details for each index occurrence**

  A detailed report on index run-time data per index occurrence is always output for a system-owned index. For a user-owned index, it is output only when explicitly requested using SUMMARY DETAILED. The report provides the following:

  – The DBKEY of the index owner record occurrence and its page number.

  – The page number of the first (top level) SR8. Ideally, the top level SR8 should reside on the same page as the index owner, except for an index with only one level and a non-zero index displacement.

  – At the intermediate and bottom level (output only if the index occurrence has more than 1 level):

    ♦ Number of SR8's and its computed minimum value

    ♦ Number of pages with SR8's and its computed minimum value

    ♦ Number of displaced SR8's and as a percentage of SR8's

    ♦ Number of entries in use and as a percentage of available entries

    ♦ Number of orphans and as a percentage of used entries

    ♦ Total size of all SR8's

  – Index occurrence totals:

    ♦ Number of levels in the index and its computed minimum value

    ♦ Number of members in the index

    ♦ Size of the largest SR8

    ♦ Number of SR8's and its computed minimum value

    ♦ Number of pages with SR8's and its computed minimum value

    ♦ Number of displaced SR8's and as a percentage of SR8's

    ♦ Number of entries in use and as a percentage of available entries

    ♦ Number of orphans and as a percentage of used entries

    ♦ Total size of all SR8's

  – Estimated IO's versus number of database buffers for sequential bottom level access indicates the physical "sequentiality" of the index. Ideally, the number of I/O's should not vary with the number of buffers and should be equal to the number of pages with bottom level SR8's.

A displaced SR8 is a bottom level SR8 located within the index displacement or a non-bottom level SR8 located outside the index displacement.

A computed minimum value is obtained by using the current number of entries in the index, filling SR8's to 100% using the current value of INDEX BLOCK CONTAINS for the index, and assuming that all space on a database page is available to hold the index owner and the associated SR8's.

- **Part 3—Index overview and distribution diagrams for a user-owned index**
  - Index overview

    An index overview provides the following information:

    - Number of owner occurrences
    - Number of empty owners and as a percentage of owner occurrences
    - Number of displaced (not on same page as owner) top level SR8's
    - Total, average, and highest value of the number of SR8's
    - Total, average, and highest value of the computed minimum number of SR8's
    - Average and highest value of the index level
    - Average and highest value of the computed minimum level
    - Average and highest value of the number of pages
    - Average and highest values of the computed minimum number of pages
    - Number of index occurrences with orphans
    - Number of orphans: total and as a percentage of the number of entries and highest plus its owner DBKey
    - Total size of all SR8's
    - Size of largest SR8
  - Distribution diagrams

    A distribution diagram provides the number and percentage of index occurrences for a certain property in both a numeric and a pseudo-graphical way. Properties for which a distribution diagram is output are:

    - Index level
    - Minimum index level
    - Number of SR8's
    - Number of members in the index occurrence

- Estimated IOs using 1 buffer for sequential bottom level access
- Number of pages with intermediate level SR8's
- Minimum number of pages with intermediate level SR8's
- Percentage displaced intermediate level SR8's
- Number of pages with bottom level SR8's
- Minimum number of pages with bottom level SR8's
- Percentage displaced bottom level SR8's

7.6.1.4 More Information

For more information about the PRINT INDEX utility, see the *CA IDMS Utilities Guide*.

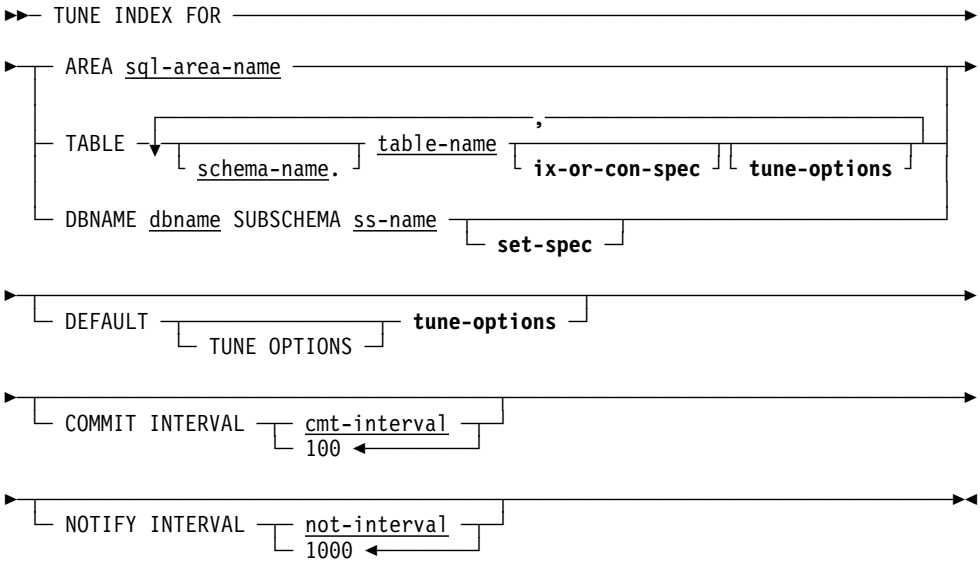
7.6.2 TUNE INDEX

The TUNE INDEX utility performs the following functions:

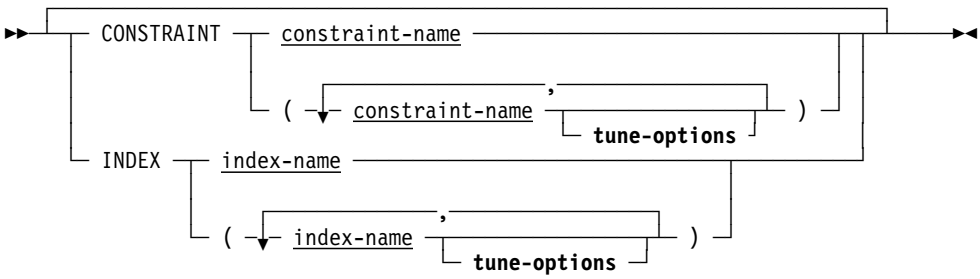
- Adopts orphans in an index structure. An *orphaned indexed record* is a record whose index pointer does not point back to the index record (SR8) that contains the record's index entry. Orphans occur as the result of splitting an existing SR8 into two records to accommodate a new entry. As part of the split, some of the entries are moved to a new SR8, but the index pointer in their associated records is not adjusted to reflect the change, resulting in "orphaned" records. By eliminating orphans, runtime database performance is improved when traversing from an indexed record to its associated index entry.
- Moves the top level SR8 to its optimal location.
- Optionally rebalances the index structure. Rebalancing ensures that the resulting index structure is a balanced tree and has a minimal number of levels and SR8's. You can temporarily override the index block contains value of the index and the page reserve value of the area that contains the index structure. Using these overrides allows tuning the index while allowing for future growth.
- Optionally resequences the index structure. Resequencing puts the SR8 records in physical sequence. By resequencing the index structure, database performance is improved when accessing the index structure sequentially at the bottom level.

Authorization: A user must have DBAWRITE authority on all areas processed by the utility.

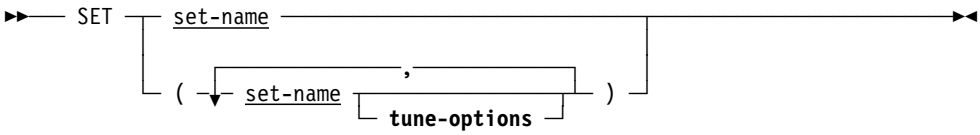
Syntax



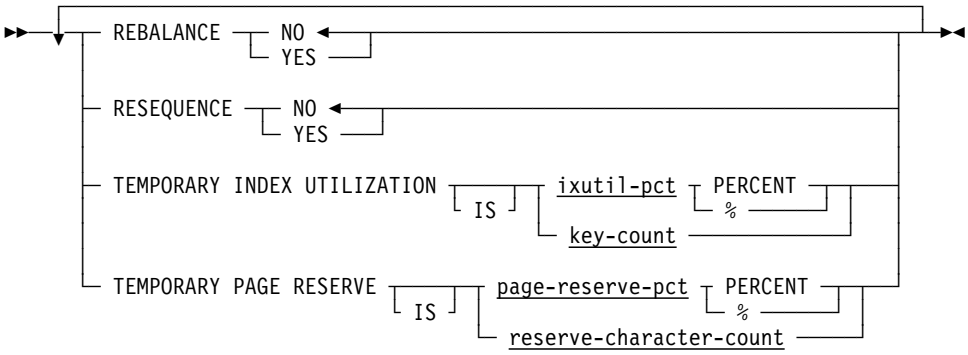
Expansion of ix-or-con-spec



Expansion of set-spec



Expansion of tune-options



Parameters

REBALANCE

Specifies whether to rebalance the index. A well-balanced index has the minimum number of index levels and best performance if the index is frequently accessed vertically from top to bottom.

YES

Rebalances the index.

Note: Rebalancing an index can be resource-intensive.

NO

No rebalancing is done.

RESEQUENCE

Specifies whether to resequence the index. A properly sequenced index is important only if the index is frequently accessed sequentially at the bottom level.

YES

Resequences the index for optimum performance.

Note: Resequencing an index can be resource-intensive.

NO

No resequencing is done.

TEMPORARY INDEX UTILIZATION

Specifies a temporary override for the operation. If not specified, the current run-time value for INDEX BLOCK CONTAINS is used and index blocks are used at 100%.

ixutil-pct

Specifies the percentage of the maximum number of entries that each index block should contain after tuning is complete. *ixutil-pct* is an integer in the range 10 through 100. The number of entries of an index block is computed as *index-block-contains* * *ixutil-pct* / 100.

key-count

Specifies the maximum number of entries that each index block should contain after tuning is complete. *key-count* is an integer in the range 3 through 8180.

TEMPORARY PAGE RESERVE

Specifies a temporary override of the page reserve for the area in which the index resides. If not specified or specified as NULL, the page reserve of the area in which the index resides is used.

page-reserve-pct

Specifies the percentage of each page to leave as free space if it contains a portion of an index being tuned. *page-reserve-pct* is an integer in the range 0 through 30. The page reserve of the area is computed as *area-page-size* * *page-reserve-pct* / 100.

reserve-character-count

Specifies the number of characters to reserve on each page to accommodate increases in the length of records or rows stored on the page if it contains a portion of an index being tuned. *reserve-character-count* is an integer with a value not larger than 30% of the page size.

7.6.2.1 Usage

General Considerations: The TUNE INDEX utility has the following usage considerations:

- To use the TUNE INDEX utility, you must specify one of the following:
  - One or more tables whose indexed constraints are to be tuned
  - One or more areas containing tables whose indexed constraints are to be tuned
  - A subschema and DBNAME and optionally a list of indexed sets to be tuned
- If multiple indexes and/or multiple tables are processed in the same area, increasing the number of buffers further improves performance.
- Index tuning is a resource-intensive operation consisting mostly of CPU and I/O.

Operating modes: You can execute the TUNE INDEX utility both online (through the online command facility) and in batch through central version or batch local. When index tuning is executed by a central version, TUNE INDEX tries to minimize impact on other online tasks as follows:

- When a record or area lock conflict occurs with other applications, TUNE INDEX takes the following actions:
  - For record lock conflicts, TUNE INDEX commits the updates done so far.
  - For area lock conflicts, TUNE INDEX finishes its database session and starts a new one.
- TUNE INDEX lowers its priority to one below its normally assigned priority. If a deadlock occurs, the default deadlock selection algorithm selects the task with the lowest priority as the deadlock victim. The TUNE INDEX utility can recover from a deadlock by restarting the index tuning process of the current index occurrence.

Commit interval: You can specify a commit interval that determines the frequency with which the utility will commit. The interval specifies the number of updates that can take place before a commit is issued. You can disable committing and automatic restart by specifying a 0-commit interval. Regardless of the commit interval specified, the utility always issues a COMMIT ALL at the end of the tune process of an index occurrence to release all record locks. It also issues a COMMIT ALL if it detects that another task is waiting on a record lock that it holds and it issues a FINISH if it detects that another task is waiting on an area lock that it holds.

Notify interval: You can specify a time interval in minutes. Each time this interval expires, a message is written indicating the index tuning progress. The message is written to the job log and the operator's console if TUNE INDEX runs in local mode; otherwise, it is written to the IDMS LOG and console. You can disable notification by specifying a 0-notify interval.

7.6.2.2 Examples

The following example directs the TUNE INDEX utility to adopt orphaned index records, rebalance and resequence the index. It also shows how to temporarily override the DMCL or subschema values for PAGE RESERVE and INDEX BLOCK CONTAINS.

```
TUNE INDEX FOR DBNAME EMPDEMO SUBSCHEMA EMPSS01
           SET (EMP-NAME-NDX)
           DEFAULT TUNE OPTIONS
              REBALANCE  YES
              RESEQUENCE YES
              TEMPORARY INDEX UTILIZATION IS 80 %
              TEMPORARY PAGE RESERVE IS 15 PERCENT
           NOTIFY INTERVAL 1000;
```

7.6.2.3 Sample Output

The following is a sample of a report produced by the TUNE INDEX utility.

```
TUNE INDEX FOR DBNAME EMPDEMO SUBSCHEMA EMPSS01
           SET (EMP-NAME-NDX)
           DEFAULT TUNE OPTIONS
              REBALANCE  YES
              RESEQUENCE YES
              TEMPORARY INDEX UTILIZATION IS 80 %
              TEMPORARY PAGE RESERVE IS 15 PERCENT
           NOTIFY INTERVAL 1000;
Status = 0        SQLSTATE = 00000        Messages follow:
DB002994 C0M333: IDMSTUNE  -  processing started
DB002994 C0M333: IDMSTUNE  -  Indexes selected for processing:
DB002994 C0M333: IDMSTUNE  -  EMP-NAME-NDX (IBC=32) in area EMPDEMO.EMP-DEMO-REGION (PGRSV=644)
DB002994 C0M333: IDMSTUNE  -  Statistics for area EMP-DEMO-REGION
DB002994 C0M333: IDMSTUNE  -  Orphan adoption read 34 records (of which 6 SR8s)
DB002994 C0M333: IDMSTUNE  -  Orphan adoption adopted 20 index orphans
DB002994 C0M333: IDMSTUNE  -  Rebalancing read 65 records
DB002994 C0M333: IDMSTUNE  -  Resequencing read 35 records
DB002994 C0M333: IDMSTUNE  -  134 total records read
DB002994 C0M333: IDMSTUNE  -  20 total index orphans adopted
DB002994 C0M333: IDMSTUNE  -  1 indexes/sets processed
DB002994 C0M333: IDMSTUNE  -  processing completed
```

7.6.2.4 More Information

- For more information about the TUNE INDEX utility, see the *CA IDMS Utilities Guide*.
- For more information about indexed constraints, see the *CA IDMS SQL Reference Guide*.
- For more information about indexing, dbnames, subschemas and sets, see the *CA IDMS Database Administration Guide*.

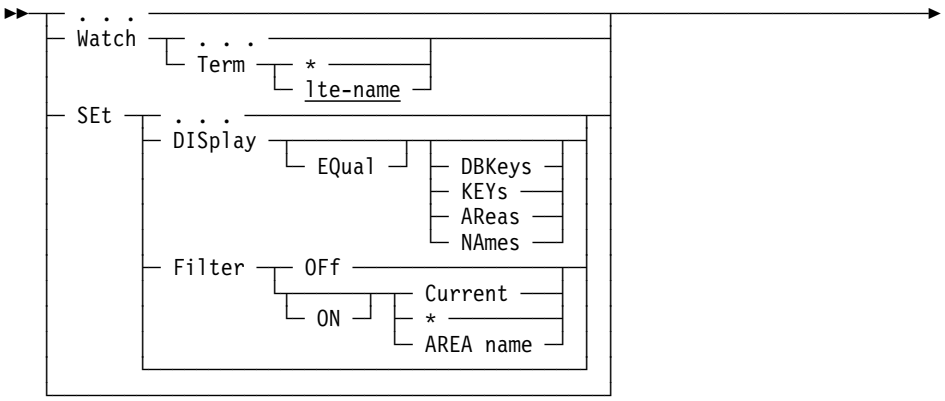
7.7 LOCKMON Longterm Lock Display Enhancements

The Lock Monitor (LOCKMON) system task includes the following enhancements:

- Reports the area portion of a keep longterm lock. This makes it easier to relate longterm lock problems back to possible sources of contention for data and potential deadlocks and bottlenecks.
- Displays the longterm lock IDs. This helps to determine which tasks and LTEs (logical terminal elements) are holding locks on which area and on which dbkey.

Syntax

LOCKMON commands



Parameters

Term
Displays logical terminals holding longterm locks.

*****
Displays all logical terminals holding longterm locks.

lte-name
Specifies the name of the logical terminal or a mask that identifies one or more logical terminals. The display can be formatted in dbkeys or area names.

DISplay EQual
Changes the display format for the terminal detail displays.

DBKeys
Displays locked dbkeys in the detail information.

KEYs
A synonym for DBKeys.

AReas
Displays area names with locked dbkeys in the detail information.

NAmes
A synonym for AReas.

Filter
Specifies a filter change.

Note: Filters are sticky items.

OFf
Turns off filtering.

ON
Turns on filtering.

Current
Uses the current filter specified in the previous filter command.

*****
Uses all filters previously set.

AREA name
Sets the filter to an area name or mask, resulting in using one or more area names as the current filter.

7.7.1 DISPLAY Commands

This section shows the area name format and DBKey format displays generated by the new WATCH TERMINAL command.

WATCH TERMINAL (Area name format) command: Enter this command to display a report of the terminals holding longterm locks, the longterm lock ids, and the area names for which locks are being held. For each area, a count of the notify, share and exclusive locks is reported.

```
CA IDMS DB/DC Lock Monitor Version nn.n     LTE: *                Tape: volser
Longterm_Lock_ID   Segment.Area_Name__________           Notfy   Share    Excl
Terminal: LTEnnnn    User: USER01
LOCK ID 1          EMPDEMO.EMP-DEMO-REGION                  0       0       2
LOCK ID 2          EMPDEMO.INS-DEMO-REGION                  0       0       2
LOCK ID 3          EMPDEMO.EMP-DEMO-REGION                  1       1       0
                   EMPDEMO.INS-DEMO-REGION                  1       1       0
LOCK1              EMPDEMO.EMP-DEMO-REGION                  0       0       2
LOCK2              EMPDEMO.INS-DEMO-REGION                  0       0       2
LOCK3              EMPDEMO.EMP-DEMO-REGION                  1       1       0
                   EMPDEMO.ins-DEMO-REGION                  1       1       0









CA IDMS DB/DC V300                                                Time: hh:mm:ss
```

WATCH TERMINAL (DBKey format) command: Enter this command to display a report of the terminals holding longterm locks, the longterm lock ids, the DBKeys associated with the longterm lock id, and the locking level of the lock held for each DBKey.

```
CA IDMS DB/DC Lock Monitor Version nn.n     LTE: *                Tape: volser
Longterm_Lock_ID   PgGrp  Lock Mode/DBKey(s)......
Terminal: LTEnnnn    User: USER01
LOCK ID 1          00000  EXCL  75007:001
LOCK ID 2          00000  EXCL  75106:001
LOCK ID 3          00000  NTFY  75106:001  75050:004
LOCK1              00000  EXCL  75007:001
LOCK2              00000  EXCL  75106:001
LOCK3              00000  NTFY  75106:001  75050:004









CA IDMS DB/DC V300                                          Time: hh:mm:ss
```

7.7.2 Miscellaneous Commands

This section shows the INFO command screen with the new longterm lock information displayed.

INFO command: Enter this command in the Lock Monitor command field to display information about the version of LOCKMON that you are running.

```
CA IDMS DB/DC Lock Monitor Version nn.n     Info/Status Details   Tape: volser



System Information
CV Number: 300                             Generation ID: TECHDC30

Task Information
Task Code: LOCKMON                         Program Name: LOCKMON

Program Information
Module Name: LOCKMON   nn.n                Assembled: mm/dd/yy @ hh:mm

Current Execution Information
Task ID:            76                     Line:        VTAM
Loaded at:    2324CC00                     PTerm:       PTEnnnn
Size:         00009F90                     LTerm:       LTEnnnn
Refresh Interval:    5                     DCMT status: Usable
Longterm Lock Displays:  Format:           Area Names
                         Filter Status: Off
                         Filter:        *

CA IDMS DB/DC V300                                              Time: hh:mm:ss
```

7.7.3 More Information

For more information about the Lock Monitor, see the *CA IDMS System Tasks and Operator Commands Guide*.

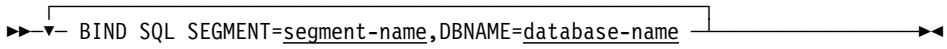
7.8 LOOK Display Enhancements

New LOOK functions report on SQL-defined database attributes and converted time stamps.

7.8.1 SQL-Defined Database Attributes

A new BIND SQL SEGMENT function has been added to the batch IDMSLOOK utility and online LOOK system task to report on logical and physical attributes for areas, tables, constraints, and indexes for a segment of an SQL-defined database. The output is similar to that of the BIND SUBSCHEMA function.

Syntax



Parameters

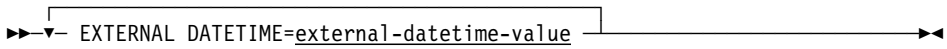
segment-name
Specifies the segment that contains the SQL database areas.

database-name
Specifies the database name that contains the segment where the catalog for the SQL definitions reside.

7.8.2 Converted Date/Time Stamps

A new EXTERNAL DATETIME function has been added to the batch IDMSLOOK utility and online LOOK system task to report on the internal value of an external date/time stamp.

Syntax



Parameters

external-datetime-value
The 26 characters that make up the external representation of the date/time stamp. The format is *yyyy-mm-dd-hh.mm.ss.ffffff*.

- *yyyy* specifies the year. *yyyy* must be an integer in the range 0001 through 9999.
- *mm* specifies the month within the year. *mm* must be an integer in the range 01 through 12.
- *dd* specifies the day within the month. *dd* must be an integer in the range 01 through 31.

- *hh* specifies the hour on a 24-hour clock. *hh* must be an integer in the range 00 through 23.
- *mm* specifies the number of minutes past the hour. *mm* must be an integer in the range 00 through 59.
- *ss* specifies the number of seconds past the minute. *ss* must be an integer in the range 00 through 59.
- *ffffff* specifies the number of millionths of a second past the specified second.

Example

Input

```
EXTERNAL DATETIME=2006-05-09-11.21.53.677107
```

Output

```
Internal datetime stamp=X'0165A2E9FD1A54F3'
```

7.8.3 More Information

- For more information about the batch IDMSLOOK utility, see the *CA IDMS Utilities Guide*.
- For more information about the online LOOK system task, see the *CA IDMS System Tasks and Operator Commands Guide*.

7.9 New Message Replacement Operand

A new operand is provided to enable including the volser of the current CA IDMS installation tape in the text of a message.

This section describes only the new operand. For more information, see the *CA IDMS IDD DDDL Reference Guide*.

Message occurrence structure

Operand	Replacement value
&$9	*CA IDMS tape volser*

7.10 New Startup Parameters

The following startup parameters have been added to CA IDMS and can be coded as freeform or positional parameters:

- **Multitasking queue depth**—Lets you set the multitasking queue depth at startup.
- **Operating system subpool**—Lets you specify a different subpool to be used at startup.
- **zIIP**—Lets you force or forbid the use of zIIP processors.

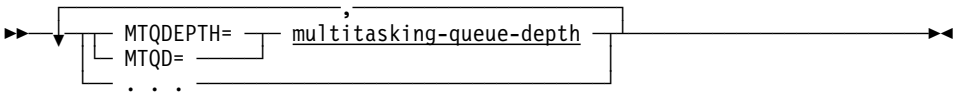
7.10.1 Coding Options as Freeform Parameters

This section describes the multitasking queue depth, operating system subpool, and zIIP freeform parameters.

7.10.1.1 Multitasking Queue Depth

In earlier releases, to override the default multitasking queue depth of 2, you had to issue a DCMT VARY MT command after a DC/UCF system had started. You can now set the desired queue depth at startup by using the MTQDEPTH parameter.

Syntax



Parameter

MTQDEPTH|MTQD=multitasking-queue-depth

(z/OS systems only) Specifies the multitasking queue depth.

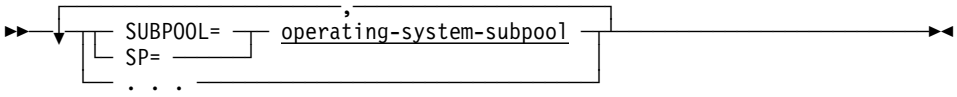
The optimum value for the MT queue depth is dependent on factors outside the control of DC, such as other work on the CPUs, operating system dispatcher parameters, paging rate, etc. Therefore, it is advised to experiment with the value and watch the results. The value must be in a range of 0 to 255; however, the advised value is in a range of 0 to 9. The default is 2.

Note: Specifying a low value causes more usage of subtasks. A too-low value causes subtasks to wake up and go back to sleep again without doing any work because the queue was already emptied by another subtask. A too-high value disables multitasking, and most if not all work is processed by only one subtask.

7.10.1.2 Subpool Usage

If you want CA IDMS to use an operating system subpool other than subpool 1 when the GETMAIN requests build the CA IDMS system at startup, you can now specify a different subpool to be used at startup with a new SUBPOOL parameter.

Syntax



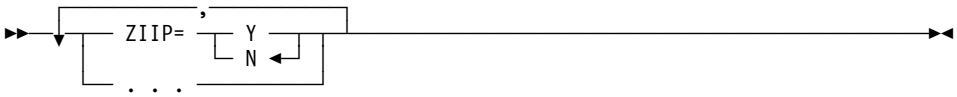
Parameter

SUBPOOL|SP=operating-system-subpool
(z/OS systems only) Specifies the operating system subpool to use for GETMAIN requests. See your operating system documentation for information on operating system subpools. The valid values for operating system subpools are from 1 to 127. The default is 1.

7.10.1.3 zIIP

You can now control use of zIIP processors in z/OS through the new zIIP startup parameter.

Syntax



Parameter

ZIIP=Y|N
(z/OS systems only) Specifies the type of zIIP support to provide in z/OS.

Valid values are the following:

- Y specifies to use zIIP processors if present.
- N specifies not to use zIIP processors. This is the default.

7.10.2 Coding Options as Positional Parameters

This section describes the positional parameter positions for the operating system subpool, multitasking queue depth, and zIIP values.

```
          Column
          0         1         2         3
          12345678901234567890123456789012345678

PARM='S=sys#prompt              tnnsclll###submqdz
```

Parameters

sub

Columns 32-34—(z/OS systems only) Specifies the operating system subpool value to use for GETMAIN requests. See your operating system documentation for information on operating system subpools. The valid values for operating system subpools are from 1 to 127. The default is 1.

mqd

Columns 35-37—(z/OS systems only) Specifies the multitasking queue depth value.

The optimum value for the MT queue depth is dependent on factors outside the control of DC, such as other work on the CPUs, operating system dispatcher parameters, paging rate, etc. Therefore, it is advised to experiment with the value and watch the results. The value must be in a range of 0 to 255; however, the advised value is in a range of 0 to 9. The default is 2.

Note: Specifying a low value causes more usage of subtasks. A too-low value causes subtasks to wake up and go back to sleep again without doing any work because the queue was already emptied by another subtask. A too-high value disables multitasking, and most if not all work is processed by only one subtask.

z

Column 38—(z/OS systems only) Specifies the type of zIIP support to provide in z/OS.

Valid values are the following:

- Y specifies to use zIIP processors if present.
- N specifies not to use zIIP processors. This is the default.

7.10.3 More Information

- For more information about the DCMT VARY MT command, see the *CA IDMS System Tasks and Operator Commands Guide*.
- For more information about startup parameters, see the appendix "Specifying Runtime Options" in the *CA IDMS System Operations Guide*.
- For more information about zIIP exploitation, see 4.5, “zIIP Exploitation” on page 98.

7.11 Online Print Log (OLP) Usability Enhancements

OLP enhancements include the following:

- You can specify seconds in FROM and TO time parameters.
- Upon initial entry, OLP flushes the data in the log buffer so that the most recent data will be included in the display.

FRom/TO

Specifies the log messages to be displayed according to the time when the messages were issued.

OLP displays the current FROM/TO times and dates. The following partial screen shows what you would see if you were searching for log records issued between 11:00 and 11:56 p.m. on 1/13/07:

```
  FROM        ON       TO       ON      COL PRT SKIP   LOG TYPES ROLL STATUS
23:00:00 2007-01-13 23:56:00 2007-01-13 001 OFF 0000 (WT/TR/DU/ ) 040
```

begin-time

Specifies the time of the *first* log message to be displayed.

You can specify *begin-time* using any one of these formats (where *hh* specifies hours based on a 24-hour clock, *mm* minutes, and *ss* seconds):

- *hh:mm:ss*—For example, 13:04:07
- *hhmm*—For example, 1304
- *hh:mm*—For example, 4:23
- *hh*—For example, 12

The following defaults are defined for *begin-time*:

- **00:00:00** is the default time if you specify FROM without a time.
- **30 minutes before the session began** is the default time if you do not specify FROM at all.

end-time

Specifies the time of the *last* log message to be displayed.

You can specify *end-time* using any one of these formats (where *hh* specifies hours based on a 24-hour clock, *mm* minutes, and *ss* seconds):

- *hh:mm:ss*—For example, 13:04:07
- *hhmmss*—For example, 130407
- *hhmm*—For example, 1304
- *hh:mm*—For example, 4:23
- *hh*—For example, 12

The following defaults are defined for *end-time*:

- **24:00:00** is the default time if you specify TO without a time.

- **The time at which the session began** is the default time if you do not specify TO at all.

More Information

For more information about the OLP command, see the *CA IDMS System Tasks and Operator Commands Guide*.

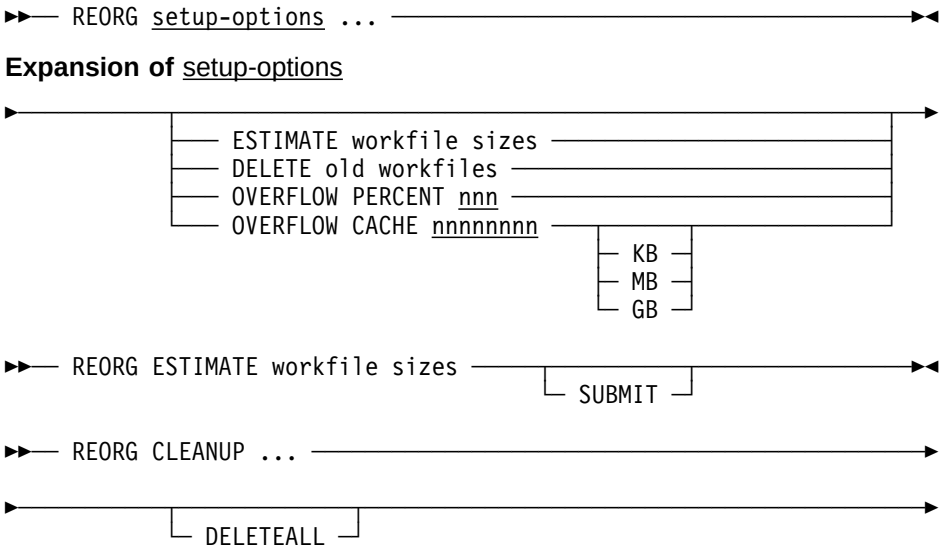
7.12 REORG Enhancements

The REORG utility has been enhanced in the following areas:

- **Work file size estimation**—REORG now estimates the size of all work files after unloading the data, or if requested during a separate pass of the data. These sizes are reported in the Work File Summary Report. They are also used to generate a SPACE parameter when dynamically creating a work file if no primary space is specified in its DSMODEL.
- **Reduced record overflows**—If a database record will not fit on its intended page, REORG now saves the record in a memory cache rather than writing it immediately to the database. This allows more records to be written to their intended pages because they are not displaced by earlier overflow records. New statistics are reported to help determine cache effectiveness.
- **Reduced area sweeps when updating index UP pointers**—REORG now makes only one pass of the database when updating pointers in a record, even if that record has multiple indexes defined to it.
- **New options for deleting work files**—The CLEANUP phase normally only deletes work files created by the current REORG operation. The new DELETEALL option allows CLEANUP to delete a work file even if it was created by another operation. The new DELETE OLD WORKFILES option directs REORG to delete all old work files prior to allocating new files during its setup phase.

Syntax

The following new syntax options have been added:



Parameters

ESTIMATE workfile sizes

Directs REORG to estimate the size of work files by gathering statistics in a separate pass of the database. This option generates estimates for both UNLOAD and RELOAD work files.

By default, statistics are collected during the unload phase that can be used for sizing RELOAD work files only. UNLOAD work files must be sized manually.

If specified together with *setup-options*, statistics gathering starts as soon as the setup phase is complete and processing stops after the statistics have been gathered. Additional jobs are automatically submitted to help gather statistics and process the database by UNLOAD slice and index.

If specified independently of *setup-options*, it must be specified in a separate execution of the REORG utility that occurs between the setup and UNLOAD phases. Multiple jobs are submitted to gather statistics only if the SUBMIT option is specified. Processing stops when file estimation is complete.

When file estimation is complete, processing must be restarted by specifying a new STOP AFTER point.

File size estimates are automatically used when dynamically allocating work files if no primary space value is specified for the file's DSMODEL.

DELETE old workfiles

Directs REORG setup to delete work files that may be left from a previous run. All existing work files that match the name of a new work file are deleted, including DBKEYS files.

By default, old work files are reused.

OVERFLOW PERCENT nnn

Specifies the percentage used to estimate the size of SYSOF2 and SYSOF8 work files and all output work files for the two overflow tasks: RELOAD2 and RELOAD5. The size of these files cannot be predicted, so they are estimated to be a percentage of related work files.

By default, 10% is used.

OVERFLOW CACHE nnnn

Specifies a limit on the size of the overflow cache used to temporarily hold records that do not fit on their target page.

nnn is the maximum size of the overflow cache specified in bytes, Kilobytes (2**10), Megabytes (2**20), or Gigabytes (2**30), depending on whether it is followed by no qualifier or by KB, MB, or GB respectively.

A value larger than 2**31-1 is limited to 2**31-1 bytes.

By default, 32 KB is used.

DELETEALL

Directs an explicit cleanup job to delete all work files associated with the current control file, including those that were not dynamically allocated by the current REORG operation. This option does not apply to DBKEYS files.

By default, only files dynamically allocated by the current REORG operation are deleted.

7.12.1 Usage

This section describes enhancements in the use of the REORG utility.

7.12.1.1 Using the REORG Utility

Efficiency of the reorganized database: The database resulting from a REORG operation should be as efficient as one reorganized through UNLOAD and RELOAD even though the two may not be identical.

A database processed by RELOAD is loaded back to front. CALC and VIA records that overflow are usually written to pages that have already been loaded and have room. A database processed by REORG is loaded from front to back. CALC and VIA records that overflow are saved in a memory cache so that they do not displace records targeting later pages. If the cache is not large enough, the records are written to an overflow file and loaded in a later step.

The resulting databases should be similar in terms of the number of records that are stored on their intended target page and the number of records that overflow, but the two databases will not be exactly the same.

You can use the IDMSDBAN utility to obtain a report of the number of page changes needed to traverse all occurrences of each CALC and VIA set. By executing this utility before and after reorganization, you can determine the effect that REORG has had on these statistics and therefore the relative efficiency of the resulting database.

7.12.1.2 Special Database Considerations

DCMT VARY PERMANENT considerations: If you run the REORG utility to change an area's low page number and Change Tracking is not used, it is recommended that you remove any permanent status on the affected area before making the new DMCL active within the CV.

When Change Tracking is not used, the PERMANENT feature is implemented by carrying the area's low page number in the journals across cycles of the CV. Changing an area's low-page prohibits future cycles of the CV from properly identifying the area once the new page range is implemented.

If a DCMT VARY SEGMENT/AREA PERMANENT command is still in effect when the new page range is implemented, the area's usage-mode at startup is

determined by the value specified in the DMCL. The entry in the journals for the old area's page range remains until the next format of the journals.

The journal entry for the old starting page can be removed without formatting the journal by doing a non-permanent DCMT VARY AREA command against the area prior to changing the DMCL definition in the CV.

7.12.1.3 Work Files

Work file creation: REORG can create work files dynamically, or you can manually create them prior to beginning the UNLOAD and/or RELOAD phases of REORG execution. Regardless of how the files are created, it is a good idea to halt execution after setup to determine what work files are needed by examining the report produced by REORG.

If using dynamic work file creation, you must specify the attributes of the work files using one or more CREATE DSMODEL statements. REORG creates the files as directed by the CREATE WORKFILE clause or at the time they are first accessed. Dynamically created work files, other than DBKEYS files, are deleted automatically during the cleanup phase.

If you want to use REORG's size estimates to create a file, code a DSMODEL without a primary SPACE allocation. You can code a SPACE parameter with just a unit type (TRK, CYL, or blksize) and no value for primary allocation.

You must code a primary space allocation value or delay creating work files until estimates are available. This means, for example, that you cannot direct REORG to create RELOAD work files during the setup phase unless the DSMODEL contains a primary allocation value.

If you code a zero primary space allocation value and a non-zero secondary value, the secondary value is replaced by a value derived from the estimated primary value.

Work file deletion: By default during the cleanup phase, REORG deletes all work files created during the current operation other than DBKEYS files. It deletes only those files for which a matching data set model was specified at setup or when the file was created. If a matching model is detected, REORG attempts to dynamically allocate the file, thereby determining its data set name which may be derived from the model or overridden by a DD statement in the JCL. Regardless of how the data set name is determined, if the file is created during the execution of a REORG statement, it is deleted during the cleanup phase of the operation unless the file is subsequently overridden with a different data set name in some later job. You can determine which files cleanup will delete by examining the work file summary sections of the REORG Status Report.

Should it be necessary to restart a REORG operation from the beginning, you should first execute a REORG statement that specifies CLEANUP in order to

delete any work files created by the interrupted operation. If you do not do this, none of the work files created during the first operation execution will be deleted by REORG even if they are reused during the second execution.

If REORG is restarted without first cleaning up the old work files, you can still direct REORG to delete them using one of the following methods:

- You can allow REORG to reuse the old files and then after REORG has ended normally, run a REORG CLEANUP job with the DELETEALL option. This option directs REORG to delete all work files, other than DBKEYS files, whether they were created by the most recent REORG operation or not
- You can delete the old work files during the setup phase of the restarted run. Specify DELETE OLD WORKFILES during setup and REORG checks for and deletes any files it finds that match the file names it plans to use. This includes DBKEYS files. This approach is preferable when the old files may be too small to be reused.

Sizing work files: The simplest way to size work files is to let REORG do it for you. REORG automatically estimates the size of all files after making a pass of the database and gathering statistics. This occurs by default when the data is unloaded. While this does not require an extra pass of the data, the estimates that are generated can only be used for allocating RELOAD work files because the UNLOAD files have already been created and used.

To use REORG-generated estimates for allocating UNLOAD work files, use the ESTIMATE WORKFILE SIZES option. This directs REORG to make a preliminary pass of the data without opening or writing to any work files. The generated file estimates are stored in the control file and can be used to allocate both UNLOAD and RELOAD work files when the REORG operation is resumed.

For REORG to more accurately estimate the size of overflow files, it may be necessary to specify an OVERFLOW PERCENT parameter. REORG assumes that 10% of the records to be stored in the database will overflow. If this assumption is not valid for a particular database, you may need to specify a different overflow percent value so that REORG can generate better estimates for the size of overflow files.

While REORG attempts to accurately estimate the size needed for work files, it may not always be able to do so. In certain cases, it may be necessary to manually estimate the size needed for one or more work files.

Estimating the size needed for work files is difficult for two reasons: many classes of files contain different types of records, and the number of records written to each file within a class varies depending on how REORG chooses to divide the page ranges into slices. Consequently, there are no simple formulas that can be used to estimate work file sizes. There are however, some techniques that can be used to facilitate file sizing and allocation.

In planning for a reorganization, do one or more trial runs to determine the actual file sizes needed for a given n-WAY value. This can be done during a non-critical time against a copy of the source database. This is the easiest way to obtain accurate size estimates. If it is impractical to do a trial reorganization on a full copy of the database, do it on a reasonably-sized sample that is representative of the original and then scale up the work file sizes proportionately. Be sure that the sample database is large enough relative to the n-WAY parameter so that the slicing algorithm does not reduce the number of slices due to their small size. Size estimates determined using a sample database are not as accurate as those determined using a full copy of the database and so should be increased to account for this.

The following is a list of additional techniques that may prove helpful in allocating work files:

- Specify a relatively modest primary allocation and a large secondary allocation. This allows the file to extend to handle large amounts of information without wasting unneeded space.
- Use IBM's Storage Management Subsystem (SMS) to determine where files are to be allocated rather than trying to manually place files on specific volumes. SMS can automatically spread the files across volumes wherever there is available space. Use the DATACLAS and STORCLAS options to allow files to extend across multiple volumes.
- If SMS is not available, allow the files to be allocated across multiple volumes by specifying multiple volume serial numbers. There must be enough space across all listed volumes to satisfy the space needs of all of the work files. The order of the entries in the list is not important.
- Consider using extended format data sets for large work files. Although this adds 32 bytes of overhead to each block, extended format data sets can have more extents across more volumes than a basic format data set.

7.12.1.4 REORG Processing Details

REORG tasks and phases: REORG processing is divided into tasks and grouped into phases. Each phase processes a type of work needed to unload or reload a database or rebuild an index. Each task processes a slice or index group within a phase.

For example, if a database were divided into two UNLOAD slices and had three index groups, the UNLOAD phase could have up to five tasks: one for each slice and possibly one for each index group. All UNLOAD tasks must successfully complete before REORG can begin to process tasks in the RELOAD or REBUILD phases.

RELOAD contains six phases numbered 1 through 6. These phases reload slices, rebuild user indexes, and reconnect pointers. Each RELOAD task in a given phase must successfully complete before processing can move to the next RELOAD phase.

REBUILD can contain up to three phases numbered 1 through 3. Each index group has one task per phase, but each index group can run independently of the others and independently of RELOAD phases 3 through 5.

If a system index related page range conflict exists between tasks in RELOAD and REBUILD or between REBUILD tasks, updates of the page range are serialized.

7.12.1.5 RELOAD Processing Phases

There is no longer a REBUILD4 phase. The updating of owner and UP pointers for system owned indexes has been merged with the RELOAD6 phase to reduce the number of passes of the database.

Eliminating the REBUILD4 phase also eliminates the SYSX10 and SYSX11 class of work files. Instead, the REBUILD3 phase generates additional SYS010 files to be processed by RELOAD6.

7.12.2 Sample Output

This section contains the REORG enhancements made to the sample output.

REORG Status Report - Section 1: The following example shows the addition of the Overflow Percentage value to the Status Report:

```
 ****************************************************************************
 *                                                                          *
 *                 REORG Status Report                                      *
 *     Identifying time stamp: 2005-11-17-13.44.06.388141      *
 *  Unload subschema=EMPTSS01  Unload segment=EMPDEMO   Unload DMCL=EMPTDMCL  *
 *  Reload subschema=EMPTSS01  Reload segment=EMPBDEMO  Reload DMCL=EMPBDMCL  *
 *                                                                          *
 ****************************************************************************
                     Options in effect
                     -----------------
Divide processing 3 ways    Notify interval=1    Job submission=YES  SHARE=YES
STOP AFTER CLEANUP  CREATE ALL WORKFILES     reuse workfiles=NO   SORTEXIT=NO
Overflow Percentage=10       Concurrent jobs=5       GENERATE DBKEYS FILES=YES
                      Current status
                      --------------
SETUP=completed    UNLOAD=completed    RELOAD=completed    CLEANUP=completed
     total tasks=23     tasks completed=23     reload areas are not locked
```

REORG Status Report - Section 6: The following example shows the general-purpose files used during the UNLOAD and RELOAD phases. Other sections show index files, sorted data files, and DBKEYS files.

All work file summary reports show the estimated size of each file and the attributes used in determining those sizes. BPT is Blocks Per Track; TPC is Tracks Per Cylinder; and *3390* indicates that attributes were based on a generic 3390 device, as opposed to a specific volume.

An asterisk (*) following a DDNAME indicates that the file was created by a REORG job executing as part of the current operation and therefore will be deleted automatically during cleanup.

```
                    Unload/Reload work file summary
                    -------------------------------
DDname    DSN                                           Estimated Size
------    ---                                           --------------
WU00001 * USERA01.EMPDEMO.WORKFILE.WU00001                       1 Trk
          Estimate based on: VOLSER=*3390*  BLKSIZE=27998  BPT=02  TPC=15
          UNIT=SYSDA   SPACE=Trk  PRI=1  SEC=1
WU00002 * USERA01.EMPDEMO.WORKFILE.WU00002                       1 Trk
          Estimate based on: VOLSER=*3390*  BLKSIZE=27998  BPT=02  TPC=15
          UNIT=SYSDA   SPACE=Trk  PRI=1  SEC=1
⋮
WU00039 * USERA01.EMPDEMO.WORKFILE.WU00039                       1 Trk
          Estimate based on: VOLSER=*3390*  BLKSIZE=27998  BPT=02  TPC=15
          UNIT=SYSDA   SPACE=Trk  PRI=1  SEC=1
Total primary space:          0 Cyl         39 Trk          0 Blk
```

Database Load Statistics Report: The following example shows standard database statistics as returned from an ACCEPT DATABASE STATISTICS command. The values reflect the processing done by the current RELOAD1 or RELOAD2 task.

```
UT005002 DATABASE LOAD STATISTICS
 DATABASE LOADED ON 01/15/08 AT 21:24:53
PAGES READ ...................            62
PAGES WRITTEN ................            59
PAGES REQUESTED ..............            86
CALC RCDS IN TARGET PAGE .....           268
CALC RCDS OVERFLOWED .........             0
VIA RCDS IN TARGET PAGE ......         3,869
VIA RCDS OVERFLOWED ..........             0
LINES REQUESTED BY IDMS ......         1,742
RCDS MADE CURRENT OF R/U .....         4,137
CALLS TO IDMS ................         4,144
FRAGMENTS STORED .............             0
RECORDS RELOCATED ............             0
```

RELOAD Statistics Report: The following example shows additional statistics produced by REORG when reloading a database. The values reflect the processing done by the current RELOAD1 or RELOAD2 task.

```
 RELOAD STATS
RCDS READ ....................        4,380
RCDS STORED IN DATABASE ......        4,137
RCDS STORED ON TARGET PAGE ...        3,998
RCDS STORED TARGET PERCENT ...           96
RCDS WRITTEN TO OVERFLOW .....          243
RCDS CACHED ..................          357
RCDS STORED FROM CACHE .......          139
RCDS OVERFLOW FROM CACHE .....          218
CACHE STORAGE USED ...........       16,384
CACHE STORAGE LIMIT ..........       16,384
```

The fields reported are the following:

RCDS READ
The total database records read from a work file. This does not include pointer and other work records

RCDS STORED IN DATABASE
The number of database records stored in the database.

RCDS STORED ON TARGET PAGE
The number of database records stored on the intended target page.

RCDS STORED TARGET PERCENT
The percentage of database records stored on their target page.

RCDS WRITTEN TO OVERFLOW
The number of database records written to a SYSOF2 file, to be processed by the RELOAD2 overflow task

RCDS CACHED
The number of database records written to the memory cache.

RCDS STORED FROM CACHE
The number of database records that were written to memory cache and then stored in the database.

RCDS OVERFLOW FROM CACHE
The number of database records that were written to memory cache and then written to the overflow work file.

CACHE STORAGE USED
The amount of allocated cache storage at the end of task. If this is the same as the storage limit, the cache reached its limit and records may have been written to the overflow file.

CACHE STORAGE LIMIT
The maximum storage allowed for the overflow cache. This can be changed using the OVERFLOW CACHE option.

7.13 Run-time DMCL File Management

The internal run-time DMCL file structures have been enhanced. Storage for information related to dynamic allocation is only allocated when needed. This storage can also be added with DCMT commands even if the DMCL previously contained no dynamic allocation information. Any DSN specified in the DMCL definition or through the JCL is saved when changed with a DCMT VARY FILE command, and is reported by a DCMT DISPLAY FILE command. Additionally, these changes enable dynamic allocation of journal files.

To implement these changes, dynamic allocation information has been removed from the file control block (FCB) and moved to a new FDSA control block. There can be multiple FDSAs chained off the FCB: one representing the data set as it is defined in the JCL, one representing the data set as it is defined in the DMCL, and one representing the data set as defined by a DCMT command.

7.14 Snap Enhancements

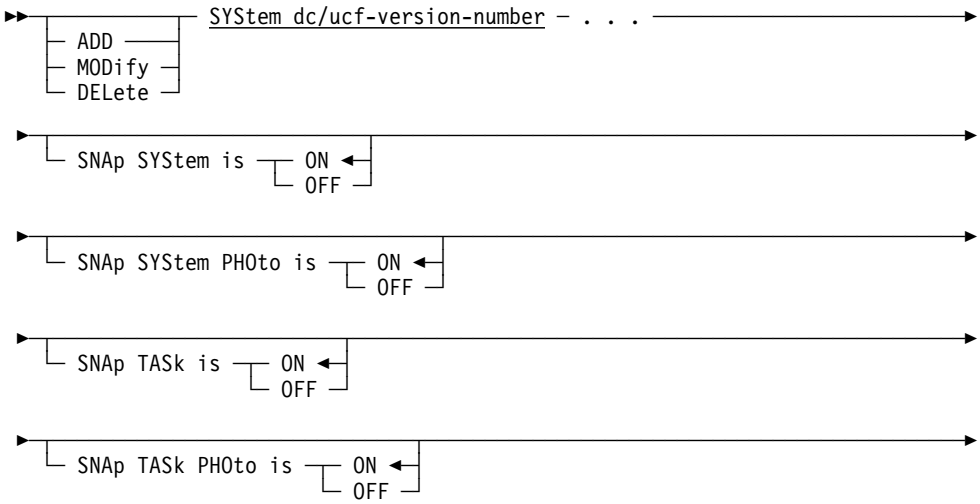
Snap enhancements include the following:

- New parameters are provided on the system generation SYSTEM statement that let you disable or enable snaps for system or task abends.
- New parameters are provided on the DCMT VARY PROGRAM and DCMT VARY TASK commands that let you dynamically enable snaps for an individual program or task or disable snaps which have been dynamically enabled.
- DCMT DISPLAY SNAP command displays are enhanced.

7.14.1 System Generation SYSTEM Statement

Use the system generation SYSTEM statement to specify whether to write system or task snap dumps or system or task photo snaps for system or task abends on a DC/UCF system.

Syntax



Parameters

SNAp SYStem is

Specifies whether to write a system snap dump to the DC/UCF log file. A system snap dump writes a formatted display of the resources allocated to all active tasks.

ON

Enables the writing of a system snap dump. This is the default for the ADD SYSTEM statement.

OFF

Disables the writing of a system snap dump.

SNAp SYStem PHOto is
Specifies whether to write a system photo snap to the DC/UCF log file. A system photo snap provides a summary of resources for all active tasks.

ON
Enables the writing of a system photo snap. This is the default for the ADD SYSTEM statement.

OFF
Disables the writing of a system photo snap.

SNAp TASk is
Specifies whether to write a task snap dump to the DC/UCF log file. A task snap dump writes a formatted display of the resources allocated to the task being snapped.

ON
Enables the writing of a task snap dump. This is the default for the ADD SYSTEM statement.

OFF
Disables the writing of a task snap dump.

SNAp TASk PHOto is
Specifies whether to write a task photo snap to the DC/UCF log file. A task photo snap provides a summary of the resources for the task being snapped.

ON
Enables the writing of a task photo snap. This is the default for the ADD SYSTEM statement.

OFF
Disables the writing of a task photo snap.

Usage

Task Snap Dump and System Snap Dump: A task snap dump can provide useful information when developing and debugging user programs. Typically, a task snap dump includes the following areas at a minimum:

- System Registers Before #SNAP
- Most Recent User Mode Registers
- Task's TCE (Task Control Element)
- TCE (Task Control Element) Stack
- Task's Signon Storage
- Task's DCE (Dispatch Control Element)
- Task's LTE (Logical Terminal Element)
- LTE's UAB (User Attribute Block, if a user is signed on to the LTE)
- Task's PTE (Physical Terminal Element)

- Task's PLE (Physical Line Element)
- Task's Storage Chain
- PDE (Program Definition Element) and Program Text
- Last 33 Messages/Codes
- Options (Startup Options Table)
- CCE (Central Control Element)
- SVC Parms
- CSA (CA IDMS Common System Area)

If a task photo snap is requested, the following information is included in the beginning of the task snap dump:

- Task Code
- Task ID
- Dispatching Priority of the Task
- Program Name and LTERM Name

For each allocated resource, the following information is included:

- RCE (Resource Control Element) Address
- RCE (Resource Control Element) TYPE
- Resource Name
- Resource Address and Resource Information

A system snap dump provides the same information for all active tasks that a task snap dump provides for a single task. In addition to those areas, a system snap dump includes information for the following areas at a minimum:

- ESE (External Service Element) Area
- ERE (External Request Element) Area
- CSA (CA IDMS Common System Area)
- TCA (Task Control Area) Header
- DCE (Dispatch Control Element) Area
- TCE (Task Control Element) Area
- RCA (Resource Control Area) Header
- RLE (Resource Link Element) Area
- RCE (Resource Control Element) Area
- DPE (Deadlock Prevention Element) Area
- ILE (Interval Lock Element) Area

- LMGR LKM (Lock Manager Main Control Block)
- LMGR (Lock Manager) Area
- LTERM (Logical Terminal) Table
- TASK (Task Definition) Table
- QUEUE Table Index
- DEST (Destination) Table
- SLL and PDE (System Library List and Program Definition Element) Area
- SYS-RU-TAB (System Run Unit Table)
- DMCL TABLE
- OXXX Module (Operating System Dependent)
- DBIO Module (IDMSDBIO Module)
- DBMS Module (IDMSDBMS Module)

If a system photo snap is requested, the following information is included in the beginning of the system snap dump for each active task:

- Task Code
- Task ID
- Dispatching Priority of the Task
- Program Name and LTERM name

For each allocated resource, the following information is included:

- RCE (Resource Control Element) Address
- RCE (Resource Control Element) TYPE
- Resource Name
- Resource Address and Resource Information

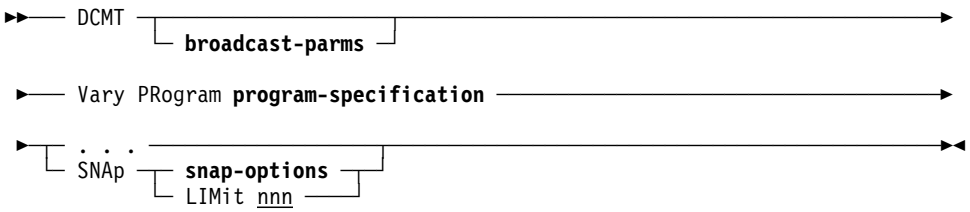
More Information

- For more information about the system generation SYSTEM statement, see the *CA IDMS System Generation Guide*.
- For more information about reading dumps, see the *CA IDMS Navigational DML Programming Guide*.
- For more information about how the system logs errors, see the *CA IDMS System Operations Guide*.

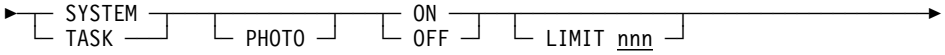
7.14.2 DCMT VARY PROGRAM Command

Use the DCMT VARY PROGRAM command to dynamically enable system or task snap dumps for a DC/UCF program or to disable snaps which have been dynamically enabled.

Syntax



Expansion of snap-options



Parameters

SNAp snap-options

Specifies the type of snap dump or photo snap to write to the DC/UCF log file.

Valid values are the following:

SYSTEM

Specifies whether to write a system snap dump for the specified program. A system snap dump writes a formatted display of the resources allocated to all active tasks.

ON

Enables the writing of a system snap dump.

OFF

Disables the writing of a system snap dump.

SYSTEM PHOTO

Specifies whether to write a system photo snap for the specified program. A system photo snap provides a summary of resources for all active tasks.

ON

Enables the writing of a system photo snap.

OFF

Disables the writing of a system photo snap.

TASK

Specifies whether to write a task snap dump for the specified program. A task snap dump writes a formatted display of the resources allocated to the task being snapped.

ON

Enables the writing of a task snap dump.

OFF

Disables the writing of a task snap dump.

TASK PHOTO

Specifies whether to write a task photo snap for the specified program. A task photo snap provides a summary of the resources for the task being snapped.

ON

Enables the writing of a task photo snap.

OFF

Disables the writing of a task photo snap.

LIMIT nnn

Specifies the total snaps allowed for the specified program. When the snap limit is reached, snaps are disabled for the program. The maximum snap limit value is 999.

Example

DCMT VARY PROGRAM program-id SNAP TASK ON LIMIT 5

```
   V PROGRAM ADSOMAIN SNAP TASK ON LIMIT 5
IDMS DC262015 V210 USER:JBC  TASK SNAP VARIED ON FOR PROGRAM
IDMS DC262016 V210 USER:JBC  SNAP LIMIT FOR PROGRAM VARIED FROM 000 TO 005
```

More Information

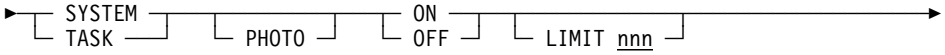
- For more information about the DCMT VARY PROGRAM command, see the *CA IDMS System Tasks and Operator Commands Guide*.
- For more information about reading dumps, see the *CA IDMS Navigational DML Programming Guide*.
- For more information about how the system logs errors, see the *CA IDMS System Operations Guide*.

7.14.3 DCMT VARY TASK Command

Use the DCMT VARY TASK command to dynamically enable system or task snap dumps for a DC/UCF task or to disable snaps which have been dynamically enabled.

Syntax

Expansion of snap-options



Parameters

SNAp snap-options

Specifies the type of snap dump or photo snap to write to the DC/UCF log file.

Valid values are the following:

SYSTEM

Specifies whether to write a system snap dump for the specified task. A system snap dump writes a formatted display of the resources allocated to all active tasks.

ON

Enables the writing of a system snap dump.

OFF

Disables the writing of a system snap dump.

SYSTEM PHOTO

Specifies whether to write a system photo snap for the specified task. A system photo snap provides a summary of resources for all active tasks.

ON

Enables the writing of a system photo snap.

OFF

Disables the writing of a system photo snap.

TASK

Specifies whether to write a task snap dump for the specified task. A task snap dump writes a formatted display of the resources allocated to the task being snapped.

ON

Enables the writing of a task snap dump.

OFF

Disables the writing of a task snap dump.

TASK PHOTO

Specifies whether to write a task photo snap for the specified task. A task photo snap provides a summary of the resources for the task being snapped.

ON

Enables the writing of a task photo snap.

OFF

Disables the writing of a task photo snap.

LIMIT nnn

Specifies the total snaps allowed for the specified task. When the snap limit is reached, snaps are disabled for the task. The maximum snap limit value is 999.

Example

DCMT VARY TASK ADS SNAP SYSTEM ON LIMIT 3

```
   V TASK ADS SNAP SYSTEM ON LIMIT 3
IDMS DC261020 V209 USER:JBC  SYSTEM SNAP VARIED ON FOR TASK
IDMS DC261021 V209 USER:JBC  SNAP LIMIT FOR TASK VARIED FROM 000 TO 003
```

More Information

- For more information about the DCMT VARY TASK command, see the *CA IDMS System Tasks and Operator Commands Guide*.
- For more information about reading dumps, see the *CA IDMS Navigational DML Programming Guide*.
- For more information about how the system logs errors, see the *CA IDMS System Operations Guide*.

7.14.4 DCMT DISPLAY SNAP Command

The DCMT D SNAPS command output is modified to include the status of all active snap overrides entered for TASK or PROGRAMS.

Examples

DCMT DISPLAY SNAP

```
D SNAPS
            *** DISPLAY SNAP REQUEST ***
     SYSTEM SNAP STATUS IS OFF (DISABLED)
     SYSTEM SNAP PHOTO STATUS IS OFF (DISABLED)
     TASK SNAP STATUS IS OFF (DISABLED)
     TASK SNAP PHOTO STATUS IS OFF (DISABLED)

Snap Overrides
Pgm/Task   Type   Limit   Task   Task Photo   System   System Photo
JBC1       ASM       12     x         x
ADSOMAIN   ASM        3                          x
RHDCD0EV   ASM                       x           x          x
JBCABORT   ADS        3     x                                x
JBCTASK2   TSK      999     x
```

DCMT DISPLAY SNAP With No Overrides Found

```
      D SNAPS
            *** DISPLAY SNAP REQUEST ***
     SYSTEM SNAP STATUS IS ON (ENABLED)
     SYSTEM SNAP PHOTO STATUS IS ON (ENABLED)
     TASK SNAP STATUS IS ON (ENABLED)
     TASK SNAP PHOTO STATUS IS ON (ENABLED)

No Program/Task Overrides Found
```

More Information

- For more information about the DCMT DISPLAY SNAP command, see the *CA IDMS System Tasks and Operator Commands Guide*
- For more information about reading dumps, see the *CA IDMS Navigational DML Programming Guide*.
- For more information about how the system logs errors, see the *CA IDMS System Operations Guide*.

7.15 Support for Large and Extended Format Files

CA IDMS now supports large format single volume files in z/OS for database and journal files and both large and extended format for work files dynamically created by the REORG utility. Both large and extended format files can have more than 65,535 tracks on a single volume. For more information about large and extended format files, see the IBM documentation.

This feature requires z/OS version 1.7 or later.

7.15.1 Large Format Database and Journal Files

The ability to allocate large format database and journal files means that fewer files need to be defined, referenced, and managed leading to a reduction in the administrative effort associated with large database environments.

To use this feature, define a database or journal file whose size in tracks exceeds 65,535. When creating the file, specify DSNTYPE=LARGE in your JCL and be sure to allocate the file on a single volume.

If you want to combine existing smaller files into fewer larger files, use the following procedure:

1. Back up by area.
2. Change the file definitions and regenerate the DMCL.
3. Delete the existing files.
4. Reallocate with the new file sizes.
5. Format and then restore by area.

7.15.2 Large and Extended Format Work Files

The ability for the REORG utility to dynamically create large and extended format files makes work file allocation easier when reorganizing large databases.

To direct REORG to create large or extended format work files, specify the new DSNTYPE parameter as described below in one or more associated DSMODEL statements.

CREATE DSMODEL Statement

The CREATE DSMODEL utility statement has been extended in r17 to allow the specification of the DSNTYPE parameter as a data set attribute.

Syntax

Expansion of dataset-attribute-spec

Parameters

DSNTYPE
Specifies the type attribute for new SMS-managed data sets. DSNTYPE overrides the DSNTYPE defined in the data class of a new data set. If SMS is not active, DSNTYPE is ignored.

EXTREQ
Specifies extended format. This option is valid for VSAM and sequential data sets only.

EXTPREF
Specifies extended as the preferred format. The data set will be allocated as extended if it is either VSAM or sequential; otherwise, it will be allocated as a basic format data set.

LARGE
Specifies large format. This option is valid for sequential non-VSAM data sets only.

BASIC
Specifies basic format. This option is valid for sequential non-VSAM data sets only.

Usage

Extended and large format files: Large and extended format data sets can have more than 65,535 tracks on a single volume, making them particularly useful for storing large amounts of data. For more information about the characteristics and limitations associated with extended and large format files, refer to the appropriate IBM documentation.

7.15.3 More Information

For more information about the REORG and CREATE DSMODEL utility statements, see the *CA IDMS Utilities Guide*.

7.16 SVC Enhancements

This section describes the enhancements made to the SVC.

7.16.1 Default to the Secured SVC

This feature prevents the unintentional installation of the unsecured version of the SVC on z/OS systems only.

The CVKEY is now a mandatory parameter with no default value in both the SVC and the install procedure. In addition to the existing valid values of 1 through 15, a value of * can be used to indicate that the unsecured SVC should be generated.

The following steps are required prior to performing the install procedure and any subsequent rebuilds of the SVC:

1. Study the "CA IDMS z/OS System Integrity Statement" in SAMPJCL.
2. Study and understand the CVKEY operand explanations in the *CA IDMS Installation and Maintenance Guide—z/OS* and *CA IDMS System Operations Guide*.
3. Ensure the system startup module is now located in an authorized library.
4. Update SYS1.PAMLIB(SCHEDnn) to indicate the key chosen for CVKEY.
5. We recommend that AUTHREQ=YES also be coded.

7.16.2 Load the SVC Using CAIRIM

Module IDMSMSVA has been added to the list of modules required to be present when CAIRIM is run to load an r17 SVC.

CAIRIM can now be used to refresh a copy of IDMSMSVA or RHDCSSFM without replacing the SVC by using the following form of the REFRESH parameter:

```
PRODUCT(CA IDMS) VERSION(GJH0) INIT(GJH0INIT) PARM(REFRESH(module-name))
```

Note: When a module is refreshed, all CVs and batch IDMS jobs that are using the module being refreshed must be ended prior to the REFRESH. Once the module has been refreshed, CVs and batch jobs can be restarted.

The r17 version of RHDCSSFM is downward compatible. If you are using an earlier release of CA IDMS, the previous version of this module may not have been refreshed when the r17 SVC was installed. This module should be explicitly refreshed.

Note: For more information, see the *CA IDMS Installation and Maintenance Guide—z/OS*.

7.17 Wait for In-Use Data Set

When dynamically allocating a data set on z/OS, local jobs and CV startup will now optionally wait for the DSN if it is in use by another job. To enable waiting, you must specify one of the following new SYSIDMS parameters with the appropriate option selected:

DYNALLOC_WAIT=ON|OFF

Specify ON to force dynamic allocation to do an ENQ wait for the DSN until it becomes available.

Specify OFF to allow the dynamic allocation request to fail when a DSN is not available.

When the DYNALLOC_WAIT_SECONDS option is specified with a non-zero value, this option is ignored.

This option applies to local mode jobs and to CV startup.

DYNALLOC_WAIT_SECONDS=*nnn*

Specifies the number of seconds that dynamic allocation waits when a DSN is unavailable. After waiting, the request is retried and if the DSN is still unavailable, the process repeats until it is successful or the job is cancelled.

Specify zero seconds to allow the dynamic allocation request to fail. If specified, the DYNALLOC_WAIT option can override this option.

If this option is specified with a non-zero value, it overrides the DYNALLOC_WAIT option.

This option applies to local mode jobs and to CV startup.

Limits: 255 seconds

Note: The IDMSMSVA module is required to use this feature. It must be loaded with the r17 SVC. See 7.16.2, “Load the SVC Using CAIRIM” on page 294 for a description of CAIRIM changes.

Note: For more information about SYSIDMS, see the *CA IDMS Common Facilities Guide*.

7.18 Forcing a Database File into Input Mode

The IDMSIOX2 Pre-Open exit call now supports forcing the database file to input mode by setting a new IOX2INPUT flag. If the Pre-Open exit sets this flag, the database file is opened in input mode and does not get reopened in update mode on a WRITE. If the IDMSIOX2 Pre-Write exit does not intercept a write request, an error is issued.

The IDMSIOX2 Pre-Write exit also supports a new IOX2REOPEN flag that causes the database file to be reopened on the next write call. The file open mode does not change for the current write. If the IDMSIOX2 Pre-Open exit does not force the file to input mode on the reopen, it is opened in update to satisfy the write. This allows the exit to control the open mode for a database file.

The IOX2INPUT and IOX2REOPEN flags are documented in the #IOX2DS copy book.

The JCL LABEL=(,,,IN) option is now honored for EXCP files. When specified, the file is opened in input mode. Any writes to this file result in a write error. This option is not supported by dynamic allocation or preserved if a file is deallocated. This option is independent of the IDMSIOX2 open mode support and overrides options set by the exit.

Chapter 8. Application Development

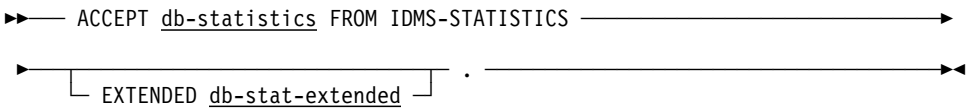
This chapter describes the application development enhancements. It contains the following topics:

8.1 Accept Extended Database Statistics DML Command

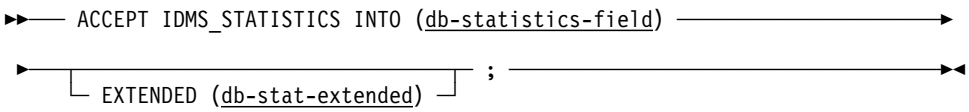
The ACCEPT database statistics command is enhanced with a new parameter that lets you acquire the extended VIB statistics provided as part of the CA IDMS runtime system. You can code this new parameter in a COBOL or PL/I program or in a CA ADS dialog.

If you manually coded the expansion of the ACCEPT database statistics verb in a program and hard coded the extra parameter on the command to acquire the extended numbers, you can remove this code and replace it with the appropriate new DML command. In addition, you can replace the hard coded record structure with a copy of the CA IDMS-provided record from the data dictionary. Implementing either one of these options requires that you precompile the program, and compile and link the appropriate load modules.

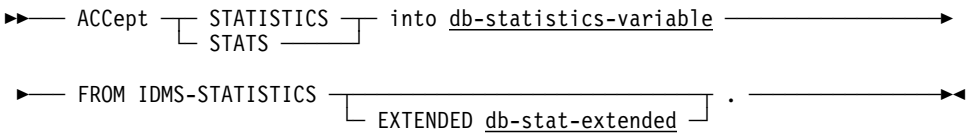
COBOL Syntax



PL/I Syntax



CA ADS Syntax



Parameter

db-stat-extended

Specifies the name of a fullword-aligned 100-byte field in program variable storage.

The data copied from IDMS-STATISTICS to *db-stat-extended* is formatted as follows for COBOL and CA ADS:

```
01  DB-STAT-EXTENDED
  03  SR8-SPLITS                 PIC S9(8)  COMP.
  03  SR8-SPAWNS                 PIC S9(8)  COMP.
  03  SR8-STORES                 PIC S9(8)  COMP.
  03  SR8-ERASES                 PIC S9(8)  COMP.
  03  SR7-STORES                 PIC S9(8)  COMP.
  03  SR7-ERASES                 PIC S9(8)  COMP.
  03  BINARY-SEARCHES-TOTAL      PIC S9(8)  COMP.
  03  LEVELS-SEARCHED-TOTAL      PIC S9(8)  COMP.
  03  ORPHANS-ADOPTED            PIC S9(8)  COMP.
  03  LEVELS-SEARCHED-BEST       PIC S9(4)  COMP.
  03  LEVELS-SEARCHED-WORST      PIC S9(4)  COMP.
  03  FILLER                     PIC X(60).
```

For COBOL, you can copy this record layout from the data dictionary by coding the following statement in program variable storage:

```
COPY IDMS DB-STAT-EXTENDED.
```

For CA ADS, the record DB-STAT-EXTENDED is defined in the dictionary when CA IDMS is installed and can be included as a dialog work record.

The data copied from IDMS_STATISTICS to *db-stat-extended* is formatted as follows for PL/I:

```
DECLARE
  01  DB_STAT_EXTENDED,
    03  SR8_SPLITS               FIXED BINARY(31),
    03  SR8_SPAWNS               FIXED BINARY(31),
    03  SR8_STORES               FIXED BINARY(31),
    03  SR8_ERASES               FIXED BINARY(31),
    03  SR7_STORES               FIXED BINARY(31),
    03  SR7_ERASES               FIXED BINARY(31),
    03  BINARY_SEARCHES_TOTAL    FIXED BINARY(31),
    03  LEVELS_SEARCHED_TOTAL    FIXED BINARY(31),
    03  ORPHANS_ADOPTED          FIXED BINARY(31),
    03  LEVELS_SEARCHED_BEST     FIXED BINARY(15),
    03  LEVELS_SEARCHED_WORST    FIXED BINARY(15),
    03  FILLER___1               CHARACTER(60);
```

You can copy this record layout from the data dictionary by coding the following statement in program variable storage:

```
INCLUDE IDMS (DB_STAT_EXTENDED).
```

Extended Statistics Fields

Field	Description
SR8 splits	The number of SR8 records that were split during the life of the run-unit.

Field	Description
SR8 spawns	The number of times that a new level of an index was created due to the splitting of the index's top level SR8.
SR8 stores	The number of SR8 records of all levels that were stored into the database.
SR8 erases	The number of SR8 records of all levels that were erased from the database.
SR7 stores	The number of SR7 records stored into the database.
SR7 erases	The number of SR7 records erased from the database.
Total binary searches	The total number of times the DBMS initiated a binary search against an index.
Total levels searched	Incremented every time that the DBMS goes down a level during a binary search throughout the life of the entire run-unit across all accessed indexes.
Orphans adopted	The number of orphaned user records that were adopted back to their referencing level-0 SR8.
Fewest levels searched (best)	The fewest number of levels walked during a binary search throughout the life of the run-unit.
Most levels searched (worst)	The greatest number of levels walked during a binary search throughout the life of the run-unit.

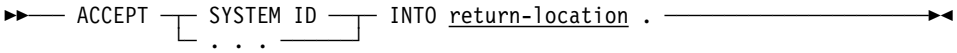
More Information

- For more information about using the ACCEPT database statistics command in COBOL programs, see the *CA IDMS DML Reference Guide for COBOL*.
- For more information about using the ACCEPT database statistics command in PL/I programs, see the *CA IDMS DML Reference Guide for PL/I*.
- For more information about using the ACCEPT database statistics command in CA ADS, see the *CA ADS Reference Guide*.

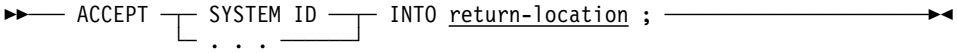
8.2 Accept System ID DML Command

The ACCEPT command is enhanced to include a new SYSTEM ID parameter that lets you retrieve the system ID of the current DC/UCF system. You can code the SYSTEM ID parameter in a COBOL or PL/I program or in a CA ADS dialog.

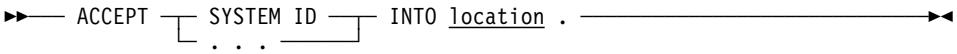
COBOL Syntax



PL/I Syntax



CA ADS Syntax



Parameter

SYSTEM ID

Specifies the 8 character name (nodename) by which the DC/UCF system is known to other nodes in the DC/UCF communications network.

More Information

- For more information about using the ACCEPT command in COBOL programs, see the *CA IDMS DML Reference Guide for COBOL*.
- For more information about using the ACCEPT command in PL/I programs, see the *CA IDMS DML Reference Guide for PL/I*.
- For more information about using the ACCEPT command in CA ADS, see the *CA ADS Reference Guide*.

8.3 ADSORPTS Enhancements

ADSORPTS enhancements include the following:

- SQL table expansion
- Unlimited dialog reporting

8.3.1 SQL Table Expansion

ADSORPTS is enhanced to report on SQL tables as it reports on records. When a RECORD report is requested, ADSORPTS now provides column names and descriptions for SQL tables in the same format that it provides field names and descriptions for records.

8.3.2 Unlimited Dialog Reporting

ADSORPTS is enhanced to report on an unlimited number of dialogs and eliminate the message: Dialog Name Table too small. This message was displayed in the previous release when the total number of dialog names, masks, and ranges that were requested in a single ADSORPTS job step exceeded a limit of 200.

You no longer need to break migrations into multiple steps of less than 200 requests each.

8.3.3 More Information

For more information about ADSORPTS, see the *CA ADS Reference Guide*.

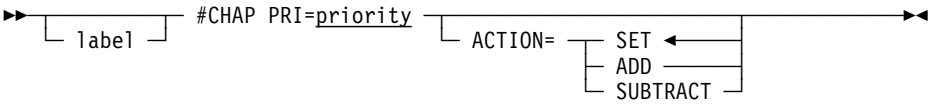
8.4 Assembler Programming Enhancements

This section describes the enhancements that are usable in Assembler programs.

8.4.1 #CHAP

The #CHAP DML statement is enhanced to change the dispatching priority of the issuing task relative to its current priority.

Syntax



Parameters

PRI=

Specifies a new dispatching priority for the issuing task.

priority

A register that contains the *priority* in the low-order byte, the symbolic name of a user-defined field that contains the *priority*, or an absolute expression in the range 0 through 240.

ACTION=

Specifies the meaning of the *priority* value using one of the following options:

SET

The *priority* is an absolute value. SET is the default.

ADD

The *priority* is a relative value and is added to the task's current *priority*.

SUBTRACT

The *priority* is a relative value and is subtracted from the task's current *priority*.

Example: The following example lowers the dispatching priority to one less than the current dispatching priority:

```
#CHAP PRI=1,ACTION=SUBTRACT
```

Status Codes: The change-priority request is unconditional; any return code other than X'00' results in an abend of the task.

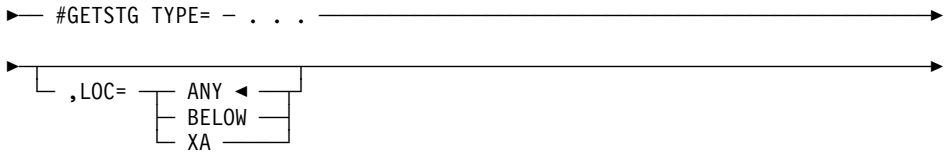
More Information

For more information about the #CHAP DML statement, see the *CA IDMS DML Reference Guide for Assembler*.

8.4.2 #GETSTG

The #GETSTG DML statement is enhanced to include a new parameter that requests the system to allocate storage above the 16-megabyte line.

Syntax



Parameters

LOC=
Indicates where the system allocates storage.

ANY
(Default); indicates that storage can be allocated anywhere in the region.

BELOW
Requests that the system allocate storage below the 16-megabyte line.

XA
Requests that the system allocate storage above the 16-megabyte line. This option is ignored if the system has no XA storage pools defined or if it is not XA-enabled.

More Information

For more information about the #GETSTG DML statement, see the *CA IDMS DML Reference Guide for Assembler*.

8.5 Built-In Functions for Date-Time Stamp Conversions

CA IDMS is enhanced to enable CA ADS dialogs and CA OLQ procedures to call the date-time functions of IDMSIN01 through new date-time stamp built-in functions.

8.5.1 CA ADS Built-In Functions

This section describes the new date-time stamp built-in functions that you can code into existing or new CA ADS dialogs.

8.5.1.1 Date-Time Stamp Functions

Date-time stamp built-in functions convert external date-time stamps to internal date-time stamps. Conversely, internal date-time stamps can be converted to external date-time stamps. The date-time stamp built-in functions call the date-time functions of IDMSIN01.

Note: For more information about IDMSIN01, see the *CA IDMS Callable Services Guide*.

In the following table, 8-byte binary fields are defined as PIC 9(16) COMPUTATIONAL fields, while display fields are defined with PIC X definitions.

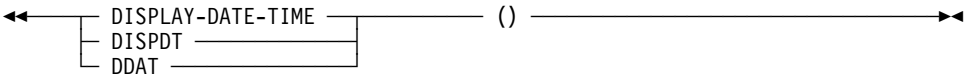
Note: For more information about the date-time stamp formats used by the date-time stamp functions, see the chapter "Representation of Date/Time Values" in the *CA IDMS SQL Reference Guide*.

Function	Keyword	What it does
External Date	DATEEXT	Returns a 10-byte external date stamp as an 8-byte internal binary date stamp
Internal Date	DATEINT	Returns an 8-byte internal binary date stamp as a displayable 10-byte date stamp
Display Date Time	DISPDT	Returns the current date-time stamp as a 26-byte displayable date-time stamp
External Date-Time	DATETIMX	Returns a 26-byte external date-time stamp as an 8-byte internal binary date-time stamp
Internal Date-Time	DTINT	Returns an 8-byte internal date-time stamp as a 26-byte displayable date-time stamp
External Time	TIMEEXT	Returns an 8-byte displayable time stamp as an 8-byte binary time stamp
Internal Time	TIMEINT	Returns an 8-byte internal time stamp as a displayable 8-byte time stamp

8.5.1.2 DISPDT

Purpose: Returns the current date-time stamp as a 26-byte displayable date-time stamp. The returned value is in the format CCYY-MM-DD-HH.MM.SS.NNNNNN.

Syntax



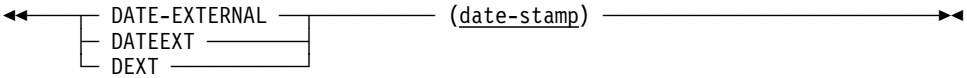
Example: In the following example, the DISPDT function is used to move the current date-time stamp to the field DATE-TIME-FIELD. The DISPDT function is executed on August 1, 2007 at approximately 3:37 p.m.

```
Statement:
    MOVE  DISPDT()  TO  DATE-TIME-FIELD

Returned string:
    '2007-08-01-15.37.11.876526'
```

8.5.1.3 DATEEXT

Purpose: Returns a 10-byte external date stamp as an 8-byte internal binary date stamp.

Syntax



Parameter

date-stamp

Specifies the 10-byte representation of the *date-stamp* in the format CCYY-MM-DD.

date-stamp can be one of the following:

- A string literal enclosed in single quotation marks
- A name of a user-defined variable data field containing the date string

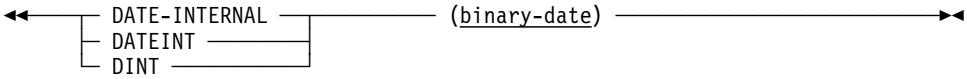
Example: In the following example, the DATEEXT function is used to convert a 10-byte string date, with the format CCYY-MM-DD, to an 8-byte binary value:

```
Initial value:
    DATE-FIELD: '2007-08-01'

Statement:
    MOVE DATEEXT (DATE-FIELD) TO DATE-BINARY

Returned value:
    x0165DB0000000000
```

8.5.1.4 DATEINT

Purpose: Returns an 8-byte internal binary-date stamp as a displayable 10-byte date stamp. The returned value is in the format CCYY-MM-DD.

Syntax



Parameter

binary-date

Specifies the user-defined variable that contains an 8-byte internal *binary-date* stamp.

binary-date must be the name of a user-defined variable that contains an 8-byte internal binary date stamp.

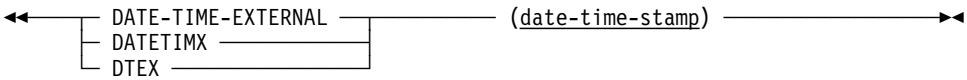
Example: In the following example, the DATEINT function converts an 8-byte internal date stamp to a 10-byte displayable value. The returned value is in the format CCYY-MM-DD.

```
Initial value:
    DATE-STAMP-BINARY: x0165DB0000000000

Statement:
    MOVE DATEINT(DATE-STAMP-BINARY) TO DATE-FIELD

Returned string:
    '2007-08-01'
```

8.5.1.5 DATETIMX

Purpose: Returns a 26-byte external date-time stamp as an 8-byte date-time internal binary stamp.

Syntax



Parameter

date-time-stamp

Specifies the 26-byte date-time-stamp to convert to an 8-byte binary date-time-stamp.

date-time-stamp can be one of the following:

- A string literal enclosed in quotation marks in the format CCYY-MM-DD-HH.MM.SS.NNNNNN
- The name of a user-defined variable containing the date-time string

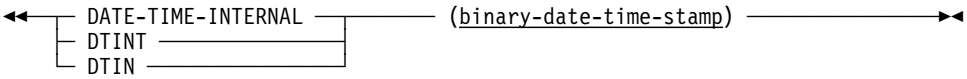
Example: In the following example, the DATETIMX function converts a 26-byte string date-time, with the format CCYY-MM-DD-HH.MM.SS.NNNNNN, to an 8-byte binary value:

```
Initial value:
    DT-FIELD: '2007-08-01-15.37.11.876526'

Statement:
    MOVE DATETIMX(DT-FIELD)  TO   DATE-TIME-BINARY

Returned value:
    x0165DB0DBA7D5FEE
```

8.5.1.6 DTINT

Purpose: Returns an 8-byte internal binary date-time stamp as a 26-byte displayable date-time stamp.

Syntax



Parameter

binary-date-time-stamp

Specifies the user-defined variable that contains an 8-byte internal binary date-time stamp.

binary-date-time stamp must be the name of a user-defined variable that contains an 8-byte internal binary date-time stamp.

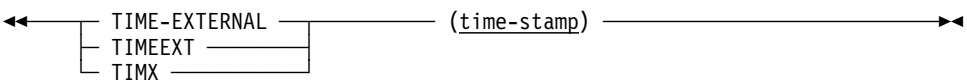
Example: In the following example, the DTINT function converts an 8-byte internal date-time stamp to a 26-byte displayable date-time stamp. The returned value is in the format CCYY-MM-DD-HH.MM.SS.NNNNNN.

```
Initial value:
    DATE-TIME-STAMP-BINARY: x0165DB0DBA7D5FEE

Statement:
    MOVE DTINT(DATE-TIME-STAMP-BINARY) TO DT-FIELD

Returned string:
    '2007-08-01-15.37.11.876526'
```

8.5.1.7 TIMEEXT

Purpose: Returns an 8-byte displayable time as an 8-byte internal binary time stamp.

Syntax



Parameter

time-stamp

Specifies the 8-byte displayable time stamp to be converted into an 8-byte binary internal time stamp.

time-stamp can be one of the following:

- A string literal enclosed in quotation marks in the format HH.MM.SS
- The name of a user-defined variable that contains the 8-byte time stamp string

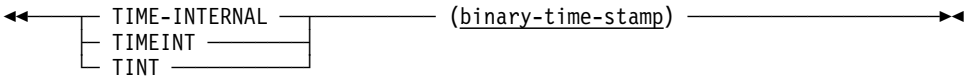
Example: In the following example, the TIMEEXT function is used to convert an 8-byte displayable time stamp to an 8-byte binary internal time stamp in the format HH.MM.SS:

```
Initial value:
    TIME-FIELD: '17.08.09'

Statement:
    MOVE TIMEEXT(TIME-FIELD) TO TIME-BINARY

Returned value:
    x0000000F0F900000
```

8.5.1.8 TIMEINT

Purpose: Returns an 8-byte internal binary time stamp as a displayable 8-byte time stamp.

Syntax



Parameters

binary-time-stamp

Specifies the 8-byte internal binary time stamp.

binary-time-stamp must be the name of a user-defined variable that contains an 8-byte internal binary time stamp.

Example: In the following example, the TIMEINT function converts an 8-byte internal binary time stamp to an 8-byte displayable time stamp in the format HH.MM.SS:

```
Initial value:
    TIME-BINARY: x0000000F0F900000

Statement:
    MOVE TIMEINT (TIME-BINARY) TO TIME-FIELD

Returned string:
    '17.08.09'
```

8.5.1.9 More Information

For more information about built-in functions, see the *CA ADS Reference Guide*.

8.5.2 CA OLQ Procedures

This section describes the new date-time stamp built-in functions that you can code into existing or new CA OLQ procedures. Date-time stamp built-in functions convert external date-time stamps to internal date-time stamps. Conversely, internal date-time stamps can be converted to external date-time stamps.

8.5.2.1 Invoking Built-In Functions

You can invoke the built-in functions by specifying an invocation name. These functions call the date-time functions of IDMSIN01.

Note: For more information about IDMSIN01, see the *CA IDMS Callable Services Guide*.

Note: For more information about the date-time stamp formats used by the date-time stamp functions, see the chapter "Representation of Date/Time Values" in the *CA IDMS SQL Reference Guide*.

Table 1 (Page 1 of 2). CA OLQ Date-Time Stamp Built-In Functions

Function	Invocation	Example
Return a 10-byte external date as an 8-byte internal binary date stamp	DATE-EXTERNAL DATEEXT DEXT	COMPUTE DATE-BINARY = DATEEXT(DATE-FIELD)
Return an 8-byte internal binary date stamp as a displayable 10-byte date stamp	DATE-INTERNAL DATEINT DINT	COMPUTE DATE-FIELD = DATEINT(DATE-STAMP-BINARY)
Return a 26-byte external date-time stamp as an 8-byte internal binary date-time stamp	DATE-TIME-EXTERNAL DATETIMX DTEX	COMPUTE DATE-TIME-BINARY = DATETIMX(DT-FIELD)
Return an 8-byte internal binary date-time stamp as a 26-byte displayable date-time stamp	DATE-TIME-INTERNAL DTINT DTIN	COMPUTE DT-FIELD = DTINT(DATE-TIME-STAMP)

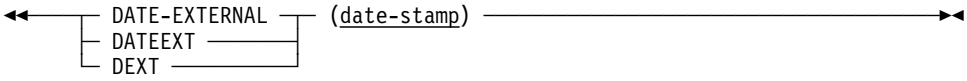
Table 1 (Page 2 of 2). CA OLQ Date-Time Stamp Built-In Functions

Function	Invocation	Example
Return an 8-byte displayable time as an 8-byte internal binary time stamp	TIME-EXTERNAL TIMEEXT TIMX	COMPUTE TIME-BINARY = TIMEEXT(TIME-FIELD)
Return an 8-byte internal binary time stamp as a displayable 8-byte time stamp	TIME-INTERNAL TIMEINT TINT	COMPUTE TIME-FIELD = TIMEINT(TIME-BINARY)

8.5.2.2 DATEEXT

Purpose: The date external function returns a 10-byte external date as an 8-byte internal binary date stamp.

Syntax



Invocation names:

```
DATE-EXTERNAL
DATEEXT
DEXT
```

Parameter

date-stamp

Represents the 10-byte date stamp in the format CCYY-MM-DD.

date-stamp can be one of the following:

- A string literal enclosed in single quotation marks
- A name of a user-defined variable data field containing the date string

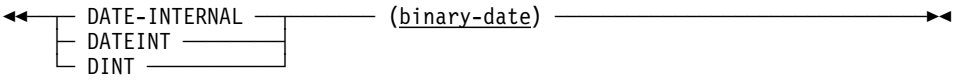
Example: This example uses the DATEEXT function to convert a 10-byte string date, with the format CCYY-MM-DD, to an 8-byte binary value.

```
Initial value:
    DATE-FIELD: '2007-08-01'

Statement:
    COMPUTE DATE-BINARY = DATEEXT(DATE-FIELD).

Returned value:
    x0165DB0000000000
```

8.5.2.3 DATEINT

Purpose: The date internal function returns an 8-byte internal binary date stamp as a displayable 10-byte date stamp. The returned value is in the format CCYY-MM-DD.

Syntax



Invocation names:

```
DATE-INTERNAL
DATEINT
DINT
```

Parameter

binary-date

Specifies the name of a user-defined variable that contains an 8-byte internal binary date.

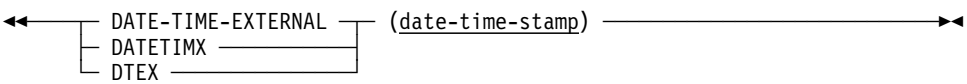
Example: This example uses the DATEINT function to convert an 8-byte internal date stamp to a 10-byte displayable value. The returned value is in the format CCYY-MM-DD.

```
Initial value:
    DATE-STAMP-BINARY: x0165DB0000000000

Statement:
    COMPUTE DATE-FIELD = DATEINT(DATE-STAMP-BINARY).

Returned string:
    '2007-08-01'
```

8.5.2.4 DATETIMX

Purpose: The date-time external function returns a 26-byte external date-time stamp as an 8-byte date-time internal binary stamp.

Syntax



Invocation names:

```
DATE-TIME EXTERNAL
DATETIMX
DTEX
```

Parameter

date-time-stamp

Specifies the 26-byte date-time stamp to convert to an 8-byte binary date-time stamp.

date-time-stamp can be one of the following:

- A string literal enclosed in quotation marks in the format CCYY-MM-DD-HH.MM.SS.NNNNNN
- The name of a user-defined variable containing the date-time string

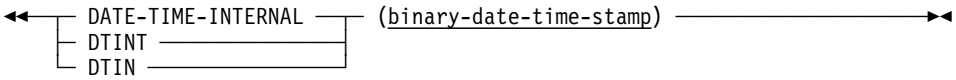
Example: This example uses the DATETIMX function to convert a 26-byte string date, with the format CCYY-MM-DD-HH.MM.SS.NNNNNN, to an 8-byte binary value.

```
Initial value:
    DT-FIELD: '2007-08-01-15.37.11.876526'

Statement:
    COMPUTE DATE-TIME-BINARY = DATETIMX(DT-FIELD)

Returned value:
    x0165DB0DBA7D5FEE
```

8.5.2.5 DTINT

Purpose: The date internal function returns an 8-byte internal binary date-time stamp as a 26-byte displayable date-time stamp.

Syntax



Invocation names:

```
DATE-TIME-INTERNAL
DTINT
DTIN
```

Parameter

binary-date-time-stamp

Specifies the name of a user-defined variable that contains an 8-byte internal binary date-time stamp.

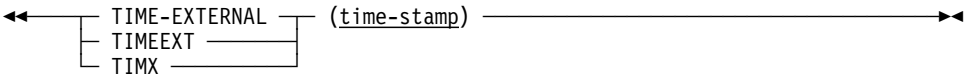
Example: This example uses the DTINT function to convert an 8-byte internal date-time stamp to a 26-byte displayable date-time stamp. The returned value is in the format CCYY-MM-DD-HH.MM.SS.NNNNNN.

```
Initial value:
    DATE-TIME-STAMP-BINARY: x0165DB0DBA7D5FEE

Statement:
    COMPUTE DT-FIELD = DTINT(DATE-TIME-STAMP-BINARY)

Returned string:
    '2007-08-01-15.37.11.'
```

8.5.2.6 TIMEEXT

Purpose: The time external function returns an 8-byte displayable time as an 8-byte internal binary time stamp.

Syntax



Invocation names:

```
TIME-EXTERNAL
TIMEEXT
TIMX
```

Parameter

time-stamp

Specifies the 8-byte displayable time stamp to be converted into an 8-byte binary internal time stamp.

time-stamp can be one of the following:

- A string literal enclosed in quotation marks in the format HH.MM.SS
- The name of a user-defined variable that contains the 8-byte time stamp string

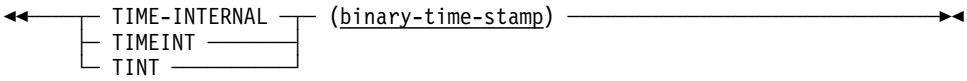
Example: This example uses the TIMEEXT function to convert an 8-byte displayable time stamp to an 8-byte binary internal time stamp in the format HH.MM.SS.

```
Initial value:
    TIME-FIELD: '17.08.09'

Statement:
    COMPUTE TIME-BINARY = TIMEEXT(TIME-FIELD)

Returned Value:
    x0000000F0F900000
```

8.5.2.7 TIMEINT

Purpose: The time internal function returns an 8-byte internal binary time stamp as a displayable 8-byte time stamp.

Syntax



Invocation names:

```
TIME-INTERNAL
TIMEINT
TINT
```

Parameters

binary-time-stamp

Specifies the 8-byte internal binary time stamp.

binary-time-stamp must be the name of a user-defined variable that contains an 8-byte internal binary time stamp.

Example: This example uses the TIMEINT function to convert an 8-byte internal binary time stamp to an 8-byte displayable time stamp in the format HH.MM.SS.

```
Initial value:
    TIME-BINARY: x0000000F0F900000

Statement:
    COMPUTE TIME-FIELD = TIMEINT(TIME-BINARY)

Returned string:
    '17.08.09'
```

8.5.2.8 More Information

For more information about built-in functions, see the *CA OLQ Reference Guide*.

8.6 COBOL Compiler Debugging Line Support

Debugging line support is provided in COBOL programs so that DML commands can be designated as debugging lines.

A debugging line is a statement that is compiled only when the compile-time switch is activated through COBOL syntax. If the debugging switch is activated, the COBOL compiler compiles the debugging lines into the object. If not activated, the debugging lines are treated as comments.

If the debugging line is also a CA IDMS DML statement, the COBOL precompiler propagates the “D” in column 7 on all generated lines, so that it is passed to the COBOL compiler and processed properly. If a DML statement is continued over multiple lines, you should only specify the “D” in column 7 of the first line of the DML statement.

8.7 FIND/OBTAIN WITHIN SET USING SORT KEY DML Statement

A COBOL program containing the FIND/OBTAIN WITHIN SET USING SORT KEY DML statement might compile with a syntax error although it compiled successfully on a prior release. CA IDMS now ensures compliance with the following rules when processing a FIND/OBTAIN WITHIN SET USING SORT KEY DML statement:

- You cannot specify multiple field names as the sort key in the USING clause.
- You must terminate the DML statement with a period or semicolon after specifying the sort key in the USING clause, unless the statement is followed by an ON clause.

The IDMSDMLC precompiler is enhanced to detect extra parameters and issue a syntax error at precompile time.

Note: For more information about the FIND/OBTAIN WITHIN SET USING SORT KEY DML statement, see the *CA IDMS DML Reference Guide for COBOL*.

8.8 IDMSIN01 Environment Information Function

An application program can call IDMSIN01 with a new function to receive a block of runtime information, such as the mode the CA IDMS application is running under.

When calling IDMSIN01 from a COBOL program, the first two parameters passed are always the address of an RPB block and the address of the function REQUEST-CODE and RETURN-CODE fields. The rest of the parameters depend on what service is being called.

COBOL programs can use standard calling conventions. The following is an example of calling IDMSIN01 to retrieve a block of runtime information:

```
  ENVIRONMENT DIVISION.
  DATA DIVISION.
  WORKING-STORAGE SECTION.


*****************************************************************
*     The following is the 1st parameter on all IDMSIN01 calls
*****************************************************************
 01  RPB.
     02  FILLER             PIC X(36).

*****************************************************************
*     The following is the 2nd parameter on all IDMSIN01 calls
*****************************************************************
 01  REQ-WK.
     02  REQUEST-CODE      PIC S9(8) COMP.
         88  IN01-FN-TRACE            VALUE 00.
         88  IN01-FN-NOTRACE          VALUE 01.
         88  IN01-FN-GETPROF          VALUE 02.
         88  IN01-FN-SETPROF          VALUE 03.
         88  IN01-FN-GETMSG           VALUE 04.
         88  IN01-FN-GETDATE          VALUE 05.
         88  IN01-FN-GETUSER          VALUE 08.
         88  IN01-FN-SYSCTL           VALUE 10.
         88  IN01-FN-TRINFO           VALUE 16.
         88  IN01-FN-TXNSON           VALUE 28.
         88  IN01-FN-TXNSOFF          VALUE 29.
         88  IN01-FN-RRSCTX           VALUE 30.
         88  IN01-FN-STRCONV          VALUE 34.
         88  IN01-FN-ENVINFO          VALUE 36.
     02  REQUEST-RETURN     PIC S9(8) COMP.

*****************************************************************
*     The following work fields are used by a variety of
*     IDMSIN01 calls
*****************************************************************
 01  WORK-FIELDS.
     02  WK-DTS-FORMAT          PIC S9(8) COMP VALUE 0.
     02  LINE-CNT               PIC S9(4) COMP.
     02  WK-DTS                 PIC X(8).
     02  WK-CDTS                PIC X(26).
     02  WK-KEYWD               PIC X(8).
```

```
     02  WK-VALUE               PIC X(32).
     02  WK-DBNAME              PIC X(8).
     02  WK-USERID              PIC X(32).
     02  WK-SYSCTL              PIC X(8).
     02  WK-TIME-INTERNAL       PIC X(8).
     02  WK-TIME-EXTERNAL       PIC X(8).
     02  WK-DATE-INTERNAL       PIC X(8).
     02  WK-DATE-EXTERNAL       PIC X(10).
     02  WK-RRS-FAKE-FUNCTION   PIC S9(4) COMP.
         88  IN01-FN-RRSCTX-GET       VALUE 01.
         88  IN01-FN-RRSCTX-SET       VALUE 02.
     02  WK-RRS-FUNCTION-REDEF    REDEFINES WK-RRS-FAKE-FUNCTION.
         03  WK-RRS-FAKE-FILLER PIC X.
         03  WK-RRS-FUNCTION    PIC X.
     02  WK-RRS-CONTEXT         PIC X(16).
     02  WK-STRING-FUNCTION     PIC X(4).
         88  CONVERT-EBCDIC-TO-ASCII  VALUE 'ETOA'.
         88  CONVERT-ASCII-TO-EBCDIC  VALUE 'ATOE'.
     02  WK-STRING              PIC X(17)
                                      VALUE 'String to convert'.
     02  WK-STRING-LENGTH       PIC S9(8) COMP VALUE 17.
*****************************************************************
*     The following group item is only used by the call that
*     returns runtime environment information.
*****************************************************************
 01  EVBLOCK.
     02  EV$SIZE                PIC S9(4) COMP VALUE +31.
     02  EV$MODE                PIC X.
     02  EV$TAPE#               PIC X(6).
     02  EV$REL#                PIC X(6).
     02  EV$SPACK               PIC X(2).
     02  EV$DMCL                PIC X(8).
     02  EV$NODE                PIC X(8).

*********************************************************
  PROCEDURE DIVISION.
*********************************************************


*********************************************************
*  Call IDMSIN01 to request that it return runtime environment
*  information.
*
*    Parm 1 is the address of the RPB.
*    Parm 2 is the address of the REQUEST-CODE and RETURN-CODE.
*    Parm 3 is the address of the ENVINFO return area.
*********************************************************

     SET IN01-FN-ENVINFO TO TRUE.
     CALL 'IDMSIN01' USING RPB REQ-WK EVBLOCK.
     DISPLAY 'Runtime mode is  ' EV$MODE.
     DISPLAY 'CA/IDMS tape volser is  ' EV$TAPE#.
     DISPLAY 'CA/IDMS release number is  ' EV$REL#.
     DISPLAY 'CA/IDMS service pack number is  ' EV$SPACK.
     DISPLAY 'DMCL name is  ' EV$DMCL.
     DISPLAY 'System node name is  ' EV$NODE.
```

Note: For more information about IDMSIN01, see the *CA IDMS Callable Services Guide*.

Chapter 9. CA IDMS Tools

This chapter describes the enhancements to the CA IDMS Tools products. It contains the following topics:

9.1 CA ADS Alive RECORD Command Enhancement

The CA ADS Alive RECORD command is enhanced so that when records to be displayed are subschema built, only those record elements that are contained in the subschema view are displayed. Otherwise, all elements are displayed. The dictionary navigation logic is updated to use the SSR-032 record as a dictionary entry point if present for the record access.

Note: For more information about the RECORD command, see the *CA ADS Alive User Guide*.

9.2 CA IDMS Dictionary Migrator Enhancements

CA IDMS Dictionary Migrator is enhanced to include two new options:

- MODIFY verb—This option can be used rather than the ADD verb when creating syntax for dictionary entities.
- REPLACE verb—This option can be used rather than the MODIFY or ADD verbs when creating syntax for dictionary entities.

By default the MODIFY syntax is generated for DDDL entities in a CHANGEONLY migration. If an entity does not exist in the target dictionary, the following message is displayed: DC601014 xyz is not in the dictionary.

If the REPLACE verb is used and the entity does not exist, the entity is added to the dictionary, and the following message is displayed: DC601210 Replace changed to Add.

You can set the MODIFY and REPLACE options by changing the installation parameters. However, you do not have to change the options from their default values. The options are not mutually exclusive; one or both or neither of them can be altered from their default values. If required, you can adjust the options during installation or at a later time by altering the source member USMTPARM and reassembling and relinking a new USMTPARM module as documented in Usermod UMOD1.

The two new installation parameters are as follows:

DDLMOD Parameter

```
DDLMOD = Y or N (defaults to DDLMOD=N)
```

If set to Y, the syntax for the schema and subschema compiler entities is created using the MODIFY verb rather than the ADD verb. In addition, if DDLMOD = Y, the schema and subschema delete syntax file statements are commented out in the SCHMDEL and SUBSDEL files.

If set to N, the syntax for the schema and subschema entities is created using the ADD verb. This is the default and is consistent with behavior in prior releases.

For z/OS users, the value for this parameter is set by the WIMVPV49 VARBLIST variable.

DDDLREP Parameter

```
DDDLREP = Y or N (defaults to DDDLREP=N)
```

If set to Y, syntax for all DDDL entities is created using the REPLACE verb.

If set to N, the syntax for DDDL entities is created using the ADD or MODIFY verbs. This is the default and is consistent with behavior in prior releases.

For z/OS users, the value for this parameter is set by the WIMVPV50 VARBLIST variable.

More Information

- For more information about CA IDMS Dictionary Migrator, see the *CA IDMS Dictionary Migrator User Guide*.
- For more information about changing installation parameters, see the CA IDMS installation guide for your operating system.

9.3 CA IDMS Journal Analyzer Enhancements

CA IDMS Journal Analyzer enhancements include the following:

- Enhanced decompression support
- Management Ranking Report enhancement

9.3.1 Enhanced Decompression Support

CA IDMS Journal Analyzer is enhanced to allow data records that are compressed using custom Data Characteristic Tables (DCTs) to be uncompressed and displayed in the JNLA Display output. To decompress records using custom DCTs, the appropriate DCT load modules must be available in the JNLA loadlib concatenation.

Only the highest version number schema record for a given schema within the OOAK-S set is processed. This is consistent with behavior in prior releases.

Note: CA IDMS Journal Analyzer already supports decompression of records that use the generic or BUILTIN compression.

When a named DCT cannot be located, the following message is displayed: DICT - NO PRESSPACK DCT TABLE NAME PROVIDED.

9.3.2 Management Ranking Report Enhancement

The CA IDMS Journal Analyzer Management Ranking Report is enhanced to provide cumulative values for all like named program executions for the following attributes:

- LOCKS REQUESTED
- PAGES READ
- PAGES WRITTEN
- PAGES WRITTEN + PAGES READ (TOTAL I/O's)
- RECORDS UPDATED

When any of these attributes are specified, an extra line is added on the Management Ranking Report (ABSOLUTE value type). This line provides a cumulative value for the specified attribute by program name.

Example

The following example shows the cumulative value of the PAGES READ and WRITTEN for all program executions by program name.

```
ID                    RELEASE                    CA IDMS JOURNAL ANALYZER              DATE            TIME          PAGE
                     Rnn.nn                       MANAGEMENT RANKINGS                mm/dd/yy        hh:mm:ss         23
                                            mm/dd/yy hh:mm - mm/dd/yy hh:mm

                          ALL RUN UNITS RANKED HIGHEST (TO LOWEST) BY PAGES READ + WRITTEN
                          RANK   RUN UNIT     PROGRAM            START             VALUE
                          ----   --------   ------------   -----------------   ------------
                             1     2090437   IDMSDDDL ONL   mm/dd/yy hh:mm:ss         1,113
                                  TOTALS     IDMSDDDL                           1,113
                             2     2092894   ADSOGEN1 ONL   mm/dd/yy hh:mm:ss           403
                             3     2092734   ADSOGEN1 ONL   mm/dd/yy hh:mm:ss           389
                             4     2090152   ADSOGEN1 ONL   mm/dd/yy hh:mm:ss           379
                             5     2092617   ADSOGEN1 ONL   mm/dd/yy hh:mm:ss           375
                                  TOTALS     ADSOGEN1                           1,546
                             6     2090589   RHDCSGEN ONL   mm/dd/yy hh:mm:ss           366
                             7     2090664   RHDCSGEN ONL   mm/dd/yy hh:mm:ss           361
                             8     2090509   RHDCSGEN ONL   mm/dd/yy hh:mm:ss           358
                             9     2090365   RHDCSGEN ONL   mm/dd/yy hh:mm:ss           355
                            10     2090269   RHDCSGEN ONL   mm/dd/yy hh:mm:ss           351
                                  TOTALS     RHDCSGEN                           1,791
```

9.3.3 More Information

For more information, see the *CA IDMS Journal Analyzer User Guide*.

9.4 CA IDMS Online Log Display Enhancement

CA IDMS Online Log Display forces the current DDLDCLOG buffer to be written at initial task invocation. This ensures that the most recent log data is included in the display.

Note: For more information about CA IDMS Online Log Display, see the *CA IDMS Online Log Display User Guide*.

9.5 CA IDMS Tools Editor Enhancement

The CA IDMS Tools Editor is enhanced to include the ECHO command. The ECHO command preserves the primary command line when this option is set on. The Profile command used to display all the environmental settings will include the ECHO setting.

The ECHO command is available for the following CA IDMS Tools online products in both browse and edit modes:

- CA ADS Alive
- CA IDMS Extractor
- CA IDMS Dictionary Migrator Assistant
- CA IDMS DME
- CA IDMS DMLO
- CA IDMS Enforcer
- CA IDMS SASO

In addition, if CA IDMS DME is entered using the ADSC Interface, the ECHO command is also available.

9.5.1 ECHO Command

Use the ECHO command to preserve the primary command line. If ECHO is turned on, the last command entered on the command line is preserved and redisplayed. If ECHO is turned off, the last command entered is not preserved. The ECHO setting is maintained in the Editor profile for the signed on CA IDMS/DC userid. The syntax for the ECHO command is the following:

```
ECHO {ON
      OFF}
```

Default: OFF

This command is available in edit and browse modes.

9.5.2 More Information

For more information about any of the CA IDMS Tools online products, see the following guides:

- *CA ADS Alive User Guide*
- *CA IDMS Extractor User Guide*
- *CA IDMS Dictionary Migrator User Guide*
- *CA IDMS Dictionary Module Editor User Guide*

- *CA IDMS DML Online User Guide*
- *CA IDMS Enforcer User Guide*
- *CA IDMS SASO User Guide*

9.6 CA IDMS Tools Queue Record Deletion Enhancement

CA IDMS is enhanced with a new PROKEEP installation parameter to allow queue records to be deleted after a specified time for the following CA IDMS Tools online products:

- CA IDMS Dictionary Migrator Assistant
- CA IDMS DME
- CA IDMS Enforcer
- CA IDMS Masterkey
- CA IDMS SASO

Queue records are used to save settings that are used in the product's operation. These settings include: module names, PF key values, caps, and tabs. A number of unused queue records can accumulate over time.

PROKEEP=*nnn*

This installation parameter lets you specify a retention date for the CA IDMS Tools online products when a retention date is not already provided.

nnn specifies the number of days to retain profile queue records in the dictionary. You must specify an integer between 0 and 255. A value of 255 retains the queue records indefinitely.

More Information

For more information about any of the CA IDMS Tools online products and specifying installation parameters, see the following guides:

- *CA IDMS Dictionary Migrator User Guide*
- *CA IDMS Dictionary Module Editor User Guide*
- *CA IDMS Enforcer User Guide*
- CA IDMS installation guide for your operating system
- *CA IDMS Masterkey User Guide*
- *CA IDMS SASO User Guide*

9.7 CA IDMS Tools Site-Specific Segment Name and Database Name Enhancement

This installation enhancement allows CA IDMS Tools users to specify their own segment name and database name values for each database associated with a CA IDMS Tools product. By specifying these names as installation parameters, you no longer have to alter the generated installation JCL to change the CA-supplied segment name and database name values to match the existing names at your site.

To use this feature, override the default segment name and database name values using the installation parameters listed in the following table.

When performing an upgrade installation, you need to know the existing segment name and database name values. You can obtain them in one of the following ways:

- If the installer is not a database administrator, the values can be obtained from the DBA staff.
- Log on to the appropriate CA IDMS system and display the values by doing the following:
  - Enter the DCMT DISPLAY SEGMENTS command to display information about the segment names.
  - Enter the DCMT DISPLAY DBTABLE command to display information about the database names.
- Log on to the appropriate CA IDMS system and enter the online LOOK DMCL system task command to display information about the DMCL. The DMCL contains information about the segment names and database names.
- Run a batch job to execute the IDMSLOOK utility to report on the DMCL. The batch report contains the same information as the online LOOK DMCL system task command but can be printed for easier reference.

The following are the new installation variables for the CA IDMS Tools installation that allow you to specify site-specific segment names and database names for each database associated with a CA IDMS Tools product:

Product Name	VARBLIST Segment Name Variable	Default Segment Name	VARBLIST Database Name Variable	Default Database Name
CA IDMS Dictionary Migrator Assistant	DMA#SEG	DMA	DMA#DBN	DMA
CA IDMS DMLO	DMLO#SEG	DMLO	DMLO#DBN	DMLO
CA IDMS Enforcer	ENFR#SEG	ENFORCER	ENFR#DBN	ENFORCER
CA IDMS Extractor	DBX#SEG	DBX	DBX#DBN	DBX
CA IDMS Masterkey	MAST#SEG	MASTRKEY	MAST#DBN	MASTRKEY
CA IDMS SASO	SDS#SEG and SDC#SEG	SASOSTR and SASODOC	SASO#DBN	SPG

Note: CA IDMS SASO defines 2 segments, SASODOC and SASOSTR. Both should be included in a single database name.

Note: For more information about installation parameters, see the *CA IDMS Installation and Maintenance Guide—z/OS*.

Chapter 10. DCMT Command Codes

The following command codes apply to new and revised DCMT commands:

Code	DCMT Command
N024	VARY TASK
N024024	SNAP snap-options
N025	VARY PROGRAM
N025020	SNAP snap-options
N029	VARY PTERM
N029010	tcp/ip-parameters
N035	HELP
N035047	SCRATCH
N035048	TCP/IP
N035049	CHANGE TRACKING
N104	DISPLAY/VARY SCRATCH
N104001	DISPLAY SCRATCH
N104002	VARY SCRATCH
N105	VARY CHANGE TRACKING
N105002	FILE COUNT nnn/DELETE ON/OFF/REFRESH/ACTIVE/INACTIVE/DISABLE/ENABLE
N106	DISPLAY CHANGE TRACKING
N107	DISPLAY TCP/IP
N107001	DISPLAY TCP/IP ALL
N107002	DISPLAY TCP/IP SUMMARY
N107003	DISPLAY TCP/IP STATISTICS
N107004	DISPLAY TCP/IP STACK TABLE
N107005	DISPLAY TCP/IP SERVICES
N107006	DISPLAY TCP/IP SOCKETS
N108	VARY TCP/IP
N108001	VARY TCP/IP STATUS
N108002	VARY TCP/IP TCP_NODELAY
N108003	VARY TCP/IP DEFAULT STACK
N108004	VARY TCP/IP INCLUDE/EXCLUDE STACK
N108005	VARY TCP/IP MAXIMUM SOCKETS
N108006	VARY TCP/IP MAXIMUM SOCKETS PER TASK
N108007	VARY TCP/IP SERVICES FILE
N108008	VARY TCP/IP STACK TABLE
N109	VARY JOURNAL FILE
N10900	VARY JOURNAL FILE journal-file-name ACTIVE
N10901	VARY JOURNAL FILE journal-file-name INACTIVE
N10905	VARY JOURNAL FILE journal-file-name ALLOCATE
N10906	VARY JOURNAL FILE journal-file-name DEALLOCATE
N10907	VARY JOURNAL FILE journal-file-name DEALLOCATE FORCE
N10910	VARY JOURNAL FILE journal-file-name DSNAME
N10913	VARY JOURNAL FILE journal-file-name ONLINE
N10914	VARY JOURNAL FILE journal-file-name OFFLINE (PERmanent)

Note: For more information about using command codes to secure DCMT commands, see the *CA IDMS Security Administration Guide*.

The following command code has been removed:

Code	DCMT Command
N048005	VARY LINE WEIGHT

+++ Imbedding (bmgr bmaster)
+++ Imbedding (IDMSMH00 bmaster)
+++ Imbedding (IDMSPH00 bmaster)
+++ imbedding (framewrk BMASTER)
+++ Imbedding (rrights bmaster)
+++ Imbedding (tradmark bmaster)
+++ Imbedding (CHAP01 idfegh00)
+++ Imbedding (CHAP02 idfegh00)
+++ Imbedding (CHAP03 idfegh00)
+++ Imbedding (CHAP04 idfegh00)
+++ Imbedding (CHAP05 idfegh00)
+++ Imbedding (CHAP06 idfegh00)
+++ Imbedding (CHAP07 idfegh00)
+++ Imbedding (CHAP08 idfegh00)
+++ Imbedding (CHAP09 idfegh00)
+++ Imbedding (CHAP10 idfegh00)
+++ Imbedding (bmgr bmaster)
+++ Imbedding (IDMSMH00 bmaster)
+++ Imbedding (IDMSPH00 bmaster)
+++ imbedding (framewrk BMASTER)
+++ Imbedding (rrights bmaster)
+++ Imbedding (tradmark bmaster)
+++ Imbedding (CHAP01 idfegh00)
+++ Imbedding (CHAP02 idfegh00)
+++ Imbedding (CHAP03 idfegh00)
+++ Imbedding (CHAP04 idfegh00)
+++ Imbedding (CHAP05 idfegh00)
+++ Imbedding (CHAP06 idfegh00)
+++ Imbedding (CHAP07 idfegh00)
+++ Imbedding (CHAP08 idfegh00)
+++ Imbedding (CHAP09 idfegh00)
+++ Imbedding (CHAP10 idfegh00)
+++ Imbedding (bmgr bmaster)
+++ Imbedding (IDMSMH00 bmaster)
+++ Imbedding (IDMSPH00 bmaster)
+++ imbedding (framewrk BMASTER)
+++ Imbedding (rrights bmaster)
+++ Imbedding (tradmark bmaster)
+++ Imbedding (CHAP01 idfegh00)
+++ Imbedding (CHAP02 idfegh00)
+++ Imbedding (CHAP03 idfegh00)
+++ Imbedding (CHAP04 idfegh00)
+++ Imbedding (CHAP05 idfegh00)
+++ Imbedding (CHAP06 idfegh00)
+++ Imbedding (CHAP07 idfegh00)
+++ Imbedding (CHAP08 idfegh00)
+++ Imbedding (CHAP09 idfegh00)
+++ Imbedding (CHAP10 idfegh00)
+++ Imbedding (bmgr bmaster)
+++ Imbedding (IDMSMH00 bmaster)
+++ Imbedding (IDMSPH00 bmaster)
+++ imbedding (framewrk BMASTER)
+++ Imbedding (rrights bmaster)
+++ Imbedding (tradmark bmaster)
+++ Imbedding (CHAP01 idfegh00)
+++ Imbedding (CHAP02 idfegh00)
+++ Imbedding (CHAP03 idfegh00)
+++ Imbedding (CHAP04 idfegh00)
+++ Imbedding (CHAP05 idfegh00)
+++ Imbedding (CHAP06 idfegh00)
+++ Imbedding (CHAP07 idfegh00)
+++ Imbedding (CHAP08 idfegh00)
+++ Imbedding (CHAP09 idfegh00)
+++ Imbedding (CHAP10 idfegh00)