CA IDMS™

Release Notes

Version 18.0.00

CA Technologies Product References

This document references the following CA Technologies products:

- CA ADS ™ Alive
- CA ADS™ Trace
- CA IDMS™ Culprit
- CA IDMS™ DML Online (CA IDMS DMLO)
- CA IDMS™ Log Analyzer
- CA IDMS™ Task Analyzer
- CA IDMS™ Server
- CA Mainframe Software Manager (CA MSM)

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

Contents

Chapter 1: New Features

This section contains the following topics:

CA Mainframe Software Manager

CA MSM is an application that simplifies and unifies the management of CA Technologies mainframe products on z/OS systems.

CA MSM provides services that make it easier for you to do the following:

- Acquire, install, and deploy products
- Automatically obtain and apply maintenance

These services enable you to easily manage your software based on industry accepted best practices. A web-based interface makes the look and feel of the environment friendly and familiar, enabling you to install and maintain your products faster and with less chance of error.

You can acquire CA MSM from the CA Support website.

Note: For more information, see your product's installation instructions and the CA Mainframe Software Manager online help.

Installation Guide

The *Installation Guide* is been restructured for CA IDMS Version 18.0, and describes the following methods of installing CA IDMS:

- CA MSM—CA MSM simplifies and unifies the management of CA mainframe products on z/OS systems. The services provided by CA MSM acquire, install, deploy, and maintain products in a common way.
- Pax-Enhanced Electronic Software Delivery (ESD)—This utility enables you to download and install CA's mainframe software and maintenance electronically to your own disk.
- Tape—You can use the sample JCL to install CA IDMS from tape using the SMP/E RECEIVE-ACCEPT-APPLY method.

CA HTML Bookshelf

This release contains the CA HTML bookshelf, which is an HTML help system that provides access to all deliverables in the product documentation set in both HTML and PDF. HTML provides robust online viewing and search capabilities, while PDF provides a print-friendly option.

The HTML bookshelf features include:

- A single help screen that displays all documentation for this release.
- An all-in-one search tool that searches the entire documentation set and returns matches found in both the HTML and PDF formatted documentation, without the need for a specialized .PDX index file.
- Additional links for using the bookshelf, downloading Acrobat Reader, and contacting CA Technologies.

Note: You must have Adobe Reader 8 or above to view the PDF files in the bookshelf.

Access the Guides

If your documentation set has a large number of guides, the initial display may not list the guides.

To access the guides through the bookshelf

1. Click Show All in the upper right corner above the All Documentation section.

   All the guides are listed on the bookshelf.

   Note: If you click Hide All, the list of documents disappears. You can select from the letter bar to list only those documents whose titles begin with the letter you select.

2. Click the HTML or PDF link next to the document you want to open.

   The document opens.

Search the Bookshelf

The bookshelf includes a search facility that helps you locate information throughout the set.

To search the bookshelf

1. Enter your search criteria in the Search field in the upper right corner of the bookshelf and press Enter.

   The search returns HTML results listed by topic and PDF results listed by guide. The results are sorted by date so that the most recently updated topics or PDFs appear at the top of the list. To find a topic in a PDF, open the PDF and view the list of topics within the PDF that match the search criteria.

2. (Optional) Click Sort by Relevance.

   The list is reordered so that the HTML topics or PDFs that contain the most matches appear at the top of the list.

Chapter 2: Upgrading to Version 18.0

This chapter describes the considerations and actions that you must take to upgrade from CA IDMS r17 to Version 18.0.

Note: This CA IDMS documentation details Version 18.0 on the z/OS operating system only. We have not yet updated the references to the z/VSE and z/VM operating systems for Version 18.0.

This section contains the following topics:

Overview

You can upgrade CA IDMS to Version 18.0 from previous releases 10.x, 12.0, 14.0, 14.1, 15.0, 16.0 or 17.0. The Version 18.0 installation delivers the conversion utilities that you need to upgrade from all releases except from release r10.x.

This *Release Notes* guide describe the actions required to upgrade from r17 to Version 18.0. If you upgrade CA IDMS to Version Version 18.0 from a release previous to r17, review all the intervening CA IDMS Release Notes documents for the cumulative requirements for your upgrade.

The following list summarizes the actions required to upgrade from r17 to Version 18.0:

- Install the software into a new environment.
- Install the new SVC delivered with Version 18.0.
- CICS users must create new interface modules and update their CICS PPT before using Version 18.0 runtime libraries in their CICS systems.
- CICS users of CA IDMS DML Online must relink this application before using Version 18.0 runtime libraries in their CICS systems.
- UCF and UDAS users must create new interface modules using the Version 18.0 installed macro library.
- Update the CA IDMS task and program definitions using source members provided during installation.
- 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 must update the SYSCA and SYSTEM schemas in every catalog in which they reside.
- CA IDMS Task Analyzer and CA IDMS Masterkey users must include new SYSIDMS parameters in their CV startup JCL
- Recompile all user-written programs that reference CA IDMS control blocks.
- Offload the log file using the ARCHIVE LOG utility from your current release or initialize the log file before starting a Version 18.0 system for the first time.
- Increase the size of XA storage and reentrant program pools, if necessary.

- Update execution JCL to increase region sizes, reference new libraries, and add new SYSIDMS parameters, if necessary.
- Create a product intent module (optional).
- Review the following information to determine if any of these changes affect your environment:
  - The list of deprecated and stabilized features
  - Changes in CV startup

Software Installation and Configuration

Follow the instructions in the *Installation Guide* for your operating system and review the chapter "Installation and Configuration" for changes that impact the way you configure your IDMS environment.

In z/OS, you can install, deploy, and configure CA IDMS Version 18.0 using CA MSM. This web-based application simplifies the management of the mainframe operating environment. Use of CA MSM significantly reduces the effort required to install and maintain CA IDMS in z/OS.

Further changes in the way you customize CA IDMS 18.0 make installation and maintenance easier in all operating systems. By adhering to CA's new software delivery standards and separating user customizations from CA-distributed code, subsequent maintenance can be applied without re-implementing customizations.

More information:

Installation and Configuration (see page 29)

SVC Installation

This release delivers a new SVC that must be used with all 18.0 systems. The SVC is downward compatible and can be used with Release 14.1, 15.0, 16.0, and 17.0 systems.

Change in Log File format

The log file statistics record format is changed in 18.0. These records are marked with the **18.00** release identifier. If CA IDMS encounters a log record with an earlier release identifier, the ARCHIVE LOG utility issues the following warning message:

NON 18.0 RECORD HAS BEEN ENCOUNTERED IN THE LOG, RECORD WILL BE BYPASSED

To avoid these messages and to separate logs from previous releases, before starting a central version using 18.0 software offload the log using the ARCHIVE LOG utility for your current release or initialize the log 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 18.0 system must be offloaded or initialized prior to its use by a pre-18.0 system.

Updating the CICS Interfaces

This section describes the upgrade requirements for the CICS interfaces.

New CICS Interface Modules

Before a CICS system can use the 18.0 CA IDMS runtime library, create new IDMSINTC and IDMSINTL interface modules and UCF front-ends (if applicable) using the 18.0 libraries. You do not need to create new IDMSCINT or IDMSCINL modules or relink user applications when upgrading to a CA IDMS 18.0 system. However, 18.0 has changed the link-edit parameters for creating the IDMSINTC and IDMSINTL interface modules.

The IDMSINTL interface module is no longer delivered in source form. If you previously modified this program to change request routing, use the new IDMSINTL OPTIXIT capability instead.

If either two-phase commit or CA IDMS VSAM Transparency is used at your site, relink the CA IDMS CICS resynchronization routine.

More Information

Interface Configuration Enhancements (see page 42)

Relink the CICS Front-end version of CA IDMS DML Online

Before using the Version 18.0 or newer runtime libraries in a CICS system in which you execute CA IDMS DML Online (DMLO), relink the DMLO stub module with the appropriate IDMSCINT module.

More information:

DMLO in a CICS Front-End (see page 168)

Update the CICS System

Install the CA IDMS Version 18.0 entity definitions into the CICS CSD. The installed source library member CICSCSD contains these definitions. Consult the appropriate IBM documentation to verify that these definitions take precedence over any previously installed definitions for the corresponding entities.

More information:

Interface Configuration Enhancements (see page 42)

Update the UDAS and UCF Interfaces

Before issuing a UCF or UDAS request in a front end that is using 18.0 runtime libraries, create new UCF and UDAS front-end interface modules using the 18.0 libraries.

Recompile User-Written Programs

In 18.0, several control block formats are changed. Although most changes are for 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 18.0 macro library.

Update the CA-supplied Task and Program Definitions

New CA-supplied task and program definitions are provided for 18.0. You can update all system definitions using the batch SYSGEN compiler (RHDCSGEN) and source members provided during installation.

Important! This must be done before starting a CA IDMS system using 18.0 libraries.

To accomplish this task, do the following:

- Perform an UPGRADE configuration to upgrade the definitions for SYSTEM 99.
- Copy the task and program definitions from SYSTEM 99 to your system definition.

For more information about the UPGRADE configuration process, see the *Installation Guide* 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 earlier release or by restoring the system dictionary from a backup that was taken prior to the upgrade. If returning to an r15 or later release, it is not necessary to restore the earlier version of the task and program definitions.

Update Dictionary Definitions

New fields are added to the SYS-041, SYSMO-170, and CVGDEFS-142 records in the IDMSNTWK schema and the IDMSNWKG subschema. Update the definition of these records in every dictionary containing the IDMSNTWK schema description. Use the IDMSDIRL utility to perform the updates.

Note: For more information about executing this utility, see the *Utilities Guide*.

You should also update each of your application dictionaries using the DLOD members appropriate to your environment.

Note: For more information, see the *Installation Guide* for your operating system.

Update Catalogs

CA IDMS Visual DBA and CA IDMS SQL users must update every catalog in which the SYSCA or SYSTEM schemas are defined.

Any catalog, including non-SQL defined catalogs, may require special handling if falling back to an earlier release of CA IDMS.

More information:

Fallback Considerations (see page 21)

Update the SYSTEM Schema

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

Executing the CONVERT CATALOG utility ensures that all changes introduced in previous releases of the software are applied. When an r17 format catalog is converted, the definition of SYSTEM.TABLE is upgraded to its Version 18.0 definition and new column in associated rows is initialized appropriately.

Changes introduced in earlier releases of the software are applied if they have not already been made.

More information:

- For a description of the changes made in earlier releases of CA IDMS, see the *r17 Release Summary*.
- For information about how to execute the CONVERT CATALOG utility, see the *Utilities Guide*.

Update the SYSCA Schema

CA IDMS Visual DBA and CA IDMS SQL users must update the SYSCA schema definition when upgrading from a release previous to r17. This update must be done in each catalog in which the SYSCA schema is defined. This process varies slightly depending on your current release of CA IDMS.

More information: Review member VIEWDOC in the CAGJSAMP, or equivalent, data set for complete instruction on how the update your database files.

Fallback Considerations

The changes implemented by the 18.0 catalog conversion utility are downward compatible through r14. If you need to fall back to one of these previous releases, you do not need to take any action regarding the catalog definitions.

Should it be necessary to reorganize (or unload and reload) a catalog updated by 18.0 CA IDMS after falling back to a release previous to r17, you may need to use the 18.0 version of the IDMSCATZ subschema rather than the earlier version. You need to use the 18.0 version of IDMSCATZ only if the catalog contains disk journal definitions and one of the following actions took place under 18.0 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 CA IDMS. For this reason, if you are upgrading from either of these releases, you must retain backup files of the catalog before converting it. If you need to fall back, you must restore the catalog and any database areas containing tables that were created or altered using the 18.0 version of the software.

Storage Pool Size

You may need to increase the size of the XA storage pool (pool 255) due to increased storage requirements.

Execution JCL Changes

In most cases, no changes are required to CA IDMS execution JCL. However, the following conditions require updating JCL to upgrade to CA IDMS 18.0 successfully:

- You may need to increase the region size to accommodate increased storage requirements.
- To enable CA IDMS Task Analyzer and CA IDMS Masterkey for a CV, you must include new SYSIDMS parameters in your startup JCL. These replace the need for adding CA-supplied exit routines for these products to your RHDCUXIT module.
- You may need to change the STEPLIB and CDMSLIB library concatenations to include additional data sets or to update the data set names to reference the 18.0 libraries.
  - All customized load modules created during configuration are placed in a dataset referred to as your custom load library *[your.custom.loadlib]*. This dataset must be placed in the library concatenation list before the standard CA IDMS deployed load library *[yourHLQ.CAGJLOAD]* in all execution JCL.
  - Uppercase modules are deployed into a library separate from other CA IDMS load modules. If you need to use the uppercase character set, the uppercase load library *[yourHLQ.CAGJLMDU]* must be placed in the concatenation list before the standard CA IDMS deployed load library *[yourHLQ.CAGJLOAD]* in all execution JCL.

Note: For a complete list of libraries that may be referenced in JCL, see References to Libraries in JCL (see page 177)

Changes in CV Startup

l CVs and DC/UCF systems now share a common CA-delivered startup module: RHDCOMVS in z/OS.

Installation assigns the aliases IDMSDC and IDMSCV to the startup module named RHDCOMVS. You can refer to the startup module in your EXEC statement using either alias (IDMSDC or IDMSCV) or the standard name (RHDCOMVS). However, to attach a CV in z/OS, you must use the name IDMSCV to invoke it.

You no longer link your own startup modules to specify parameters or include user exits. The following startup changes have been made for 18.0:

- In 18.0, all startup parameters are specified through the PARM field on the EXEC statement. The #DCPARM macro and sample RHDCPARM program source are no longer distributed.
- User exits identified by name, such as WTOEXIT, are either enabled using the PARM field in your startup JCL or are linked with IDMSUXIT. They are no longer linked with the DC/UCF startup module.

More information:

Named User Exit Enablement (see page 39)

Deprecated and Stabilized Features

The following sections describe the impact of 18.0 on the support of features available in previous releases of CA IDMS.

CICS IDMSINTL Source

Macro Source Containing Executable Code

Prior to Version 18.0, several CA IDMS macros containing executable code were distributed to clients. In release 18.0, each of those macros contains only an options table. The options table is linked with a small stub module that will rarely if ever need maintenance.

Almost all the corresponding executable code is distributed in object form and is installed automatically at all client sites.

The new procedure simplifies maintenance since you will normally never need to recompile or relink your option tables after applying maintenance. It also makes it easier for CA Technologies technical support to provide assistance if a problem arises.

The following macros are affected:

- #UCFBTCH
- #UCFCICS
- #UCFCICZ
- UCFTSO
- #UDASBCH
- #UDASCIC
- IDMSINTL

A special version of each of these macros can be obtained by contacting CA IDMS Technical Support. The special version of each macro will not include any new parameters introduced after Version 17.0, but full support will be provided for each such macro in Version 18.0. CA Technologies does not warrant that the macro will be supported in future releases. Also note that any client modifications are the responsibility of the client site.

It is recommended that sites using any of these macros consult with CA Technologies about an alternative that will allow use of the standard Version 18.0 macro.

More information:

Interface Configuration Enhancements (see page 42)

Tools SVC

The SVC used by CA IDMS Task Analyzer to write statistics to the SMF file is no longer used and is no longer distributed. CA IDMS Task Analyzer now uses the standard IDMS SVC to perform this function.

IMS UCF Interface

The ability to use IMS as a UCF or UDAS front end is stabilized. A UCFIMS front-end created under an earlier release can be used to access a Version 18.0 back end. The #UCFIMS macro and supporting CA IDMS macros are installed into the CAGJMAC macro library during a standard install, but there is no automatic configuration process. Procedures used in previous releases can be used to create an interface program.

The distributed macros will not support any new features, but they will be compatible with existing applications.

#DGTBGEN Macro

The #DGTBGEN has been renamed to #GTABGEN. The parameters are unchanged.

Optional APARs

Optional APARs are no longer used to tailor CA IDMS functionality. The functionality provided through optional APARs in previous releases is now provided in one of the following three ways:

- By changes to the software that eliminate the need for an option.
- New syntax that implements the optional behavior.
- New numbered options that trigger the alternative behavior if the corresponding option flag is set in the RHDCOPTF module.

More information:

Optional APAR Replacement (see page 53)

Deprecated Numbered Options

The following numbered options have been deprecated:

- Numbered option 25 (option flag OPT00025) to override the ARCHIVE JOURNAL warning percent is replaced with a new SYSIDMS parameter. For more information, see ARCHIVE JOURNAL Warning Threshold (see page 146).
- Numbered option 38 (option flag OPT00038) to override the line length for DC reports is replaced with a new system definition SYSTEM statement parameter. For more information, see System Definition Enhancements (see page 148).
- Numbered option 76 (option flag OPT00076) to override the CA ADS comment delimiter is replaced with a new system definition ADSO statement parameter. For more information, see ADS Comment Delimiter (see page 157).
- Numbered option 208 (option flag OPT00208) to trigger century validation is replaced with new system definition and SYSIDMS parameters. For more information, see Century Validation (see page 150).
- Numbered option 246 (option flag OPT00246) to control the inclusion of trace information in snap dumps is replaced with new system definition and DCMT SNAP TASK parameters. For more information, see Enhanced System Tracing (see page 79).

LOOK OPTIONAL APARS Parameter

The LOOK system task and the IDMSLOOK utility no longer accept the OPTIONAL APARS parameter. Instead, use the new OPTION FLAGS parameter to obtain a list of the flags that have been set in the current RHDCOPTF module.

System Tracing Controls

The following parameters and commands for controlling system tracing are no longer supported and are replaced with other facilities in 18.0:

- The SYSIDMS DB_TRACE_TABLE parameter is no longer supported. This parameter is replaced with the ADJUNCT_TRACE_TABLE parameter for consistency with new SYSGEN syntax.
- The SYSGEN SYSTRACE ENTRIES parameter is replaced with a new SYSGEN and DCMT SNAP TASK parameter. For upward compatibility the system definition value for the new TRACE LIMIT parameter is set to the value of the deprecated SYSTRACE ENTRIES parameter.

- The DCMT VARY SYSTRACE ENTRIES parameter is no longer supported. The size of the system trace table is now specified using the new DCMT VARY TRACE command.
- The DCMT DISPLAY DBTRACE and DCMT VARY DBTRACE commands are no longer supported and have been replaced with the new DCMT DISPLAY TRACE and DCMT VARY TRACE commands.

More information:

Enhanced System Tracing (see page 79)

TCAM Support

Support for the TCAM line driver is no longer available.

BTAM Support in z/OS

Support for the BTAM line driver is no longer available in z/OS.

Chapter 3: Installation and Configuration

This section contains the following topics:

Installation Overview

In Version 18.0, what has traditionally been referred to as installation is now separated into the following four distinct operations:

- Acquisition
- Installation
- Deployment
- Configuration

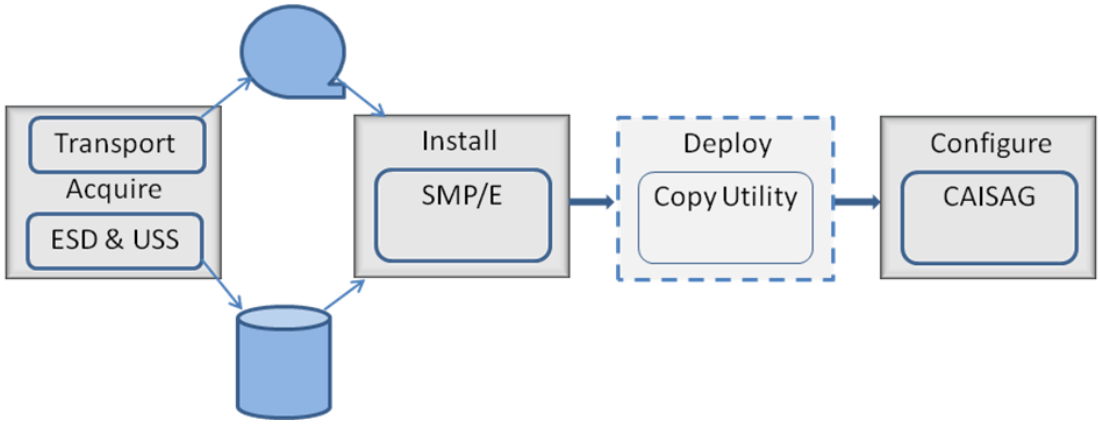
These operations can be performed using the same tools as in previous releases, such as Electronic Software Delivery (ESD) to download a PAX file and the CAISAG program to generate a configuration job stream. Now, with Version 18.0, you can perform all four operations for z/OS using CA Mainframe Software Manager (CA MSM).

Regardless of the tools used, a clear distinction is made between installing the software and configuring it to meet the needs of an operational environment. With Version 18.0, no customization takes place during installation and product selection now occurs during configuration. As a result, the contents of the installed libraries are identical at all customer sites except for a few operating system and optional interface differences.

After the software is installed, you can deploy it to one or more target systems within your environment. You can perform deployment by copying files or by using CA MSM in z/OS. Deployment is required only if you use CA MSM for configuration; otherwise, deployment is optional.

Configuration is the creation of an executable CA IDMS instance tailored to the needs of a specific operational environment. For example, you may create test and production configurations that have different characteristics to meet their different operational requirements. In Version 18.0, configuration can be done using a CAISAG generated job stream or CA MSM in z/OS. In either case, all customizations are kept separate from CA-distributed modules.

The following diagram illustrates the installation process without the use of CA MSM:



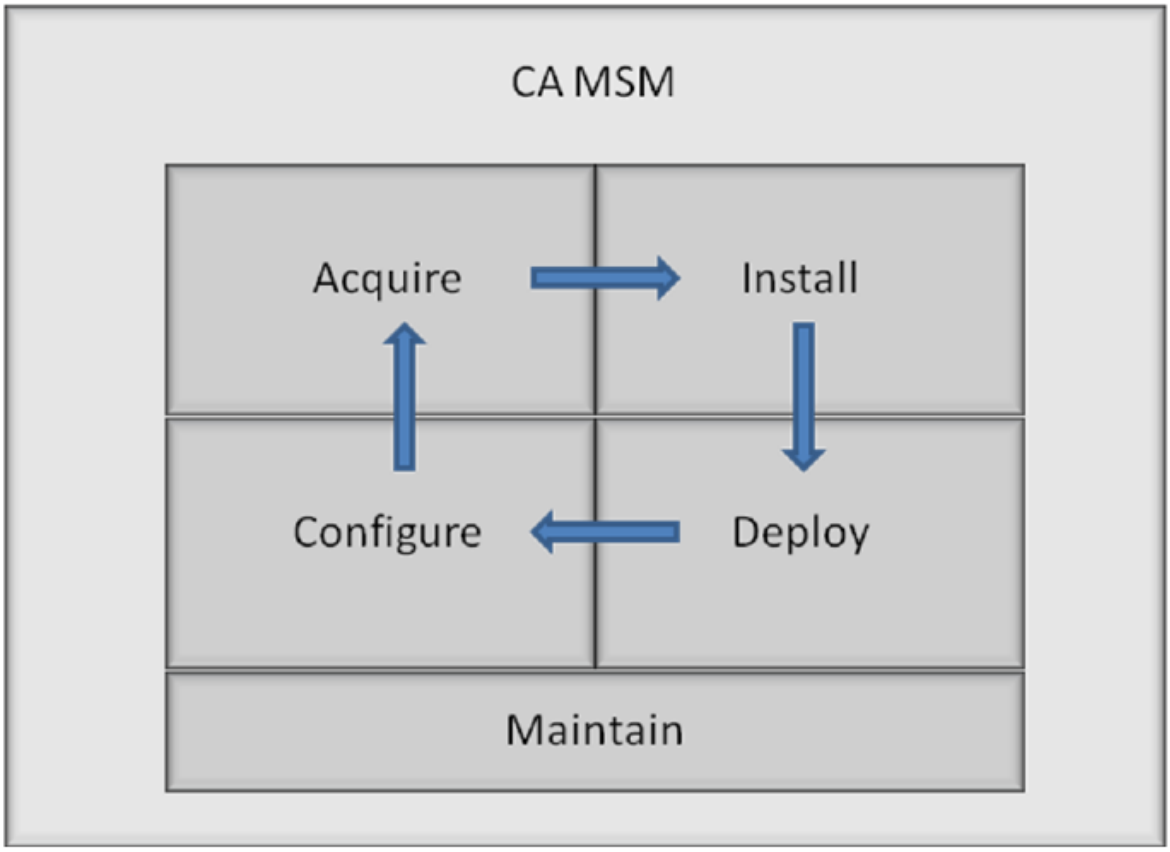
CA Mainframe Software Management Use in z/OS

z/OS users can acquire, install, deploy, configure, and maintain CA IDMS using CA Mainframe Software Manager (MSM). CA MSM is a web-based application that simplifies the management of the mainframe environment.

Use of CA MSM provides the following benefits:

- Simpler installation and maintenance through an intuitive graphical user interface.
- A paradigm that can be applied across all CA products.
- Faster installation.
- Automatic configuration changes driven by maintenance and release upgrades.

The following diagram shows the installation process using CA MSM:



Key Changes from Previous Releases

CA IDMS Version 18.0 introduces several enhancements in the installation and configuration process to conform to CA's software delivery standards. These enhancements both simplify the installation process and make applying maintenance and upgrading to new releases easier. However, these new enhancements may also have an impact on some of the procedures you have used in the past to configure CA IDMS.

The following enhancements apply to all CA IDMS Version 18.0 installations regardless of operating system or the tools used to do installation and configuration:

- The entire CA IDMS family of products is installed at all sites. Identification of the specific products to be used in an operational environment takes place during configuration. For more information about how products are identified, see Product Intent Declaration (see page 36).
- Site-specific customizations such as option modules and exits are either linked independently or with stub modules that will rarely, if ever, require maintenance updates. For more information about what this enhancement means and how it impacts your configuration procedures, see Separation of User-configured Modules (see page 32).
- Tailored load and source modules created during configuration are placed in libraries separate from those libraries that are populated during installation and deployment. These libraries are referred to as *custom libraries*. For more information about CA IDMS libraries and their use, see the *Installation Guide*.
- Optional APARs are no longer used to tailor CA IDMS behavior. In Version 18.0, optional APARs are replaced either with syntax or numbered options implemented through flag settings in the RHDCOPTF module. For a list of the optional APARs that have been eliminated and how to achieve equivalent Version 18.0 functionality, see Optional APAR Replacement (see page 53).

The preceding changes make applying maintenance and upgrading to new releases easier because most customizations need not be redone. If maintenance or release upgrade impact a customization, the preceding strategy makes reimplementing the customization easier.

Separation of User-Configured Modules

Starting with CA IDMS Version 18.0, user-configured modules and CA-distributed code are completely separated. To achieve this separation, management of the following items have been changed:

- Option modules
- Security modules
- User exits
- CA ADS built-in functions

The CA IDMS SVC is the only load module that still contains both CA-distributed common SVC code and user-configured code consisting of a customized SVCOPTS module and an optional user exit routine. Therefore you may need to reconfigure any instances of the IDMS SVC if maintenance is applied to the CA-supplied code.

Separation of Option Modules

In Version 18.0, option modules are no longer linked with the code that uses them. Instead, they are either dynamically loaded or linked with stub modules that rarely require maintenance updates.

Note: Option modules define a set of runtime options for a CA IDMS product or component. These modules are typically created by assembling a set of source code and linking the result with the code for which the options apply.

The Separation of Option Modules from CA-delivered code impacts the following situations:

- All CICS and UCF option modules are now linked with stub modules rather than the interface itself. For more information see Interface Configuration Enhancements (see page 42).
- The startup parameter module (RHDCPARM) is no longer used, eliminating the need for linking a customized CA IDMS startup routine.
- All CA IDMS Tools options modules are now dynamically loaded. For more information see Installation and Configuration Enhancements (see page 167).
- The CA IDMS exit routines are no longer linked with RHDCUXIT, eliminating the need for linking these CA modules with your customized RHDCUXIT routine. The CA ADS Alive exit feature is still supported in v18 and will be used if CA ADS Alive has been installed in the CV. No further action is required to use CA ADS Alive. For more information see Simplified Tools Exit Management (see page 169).
- The CICS version of DMLO is now linked in a way that separates the DMLO application from the IDMSCINT interface module. See the section DMLO in a CICS Front-End (see page 168).
- The COBOL and PL/I precompilers now dynamically load their parameter module. For more information see Precompiler Parameter Module (see page 33).
- CA Culprit for CA IDMS now dynamically loads its parameter module. For more information see CA Culprit for CA IDMS Profile Module (see page 34).

Precompiler Parameter Module

In Version 18.0, the COBOL and PL/I precompilers dynamically load a parameter module. This module lets you tailor the behavior of the precompilers without having to relink the precompilers when maintenance is applied.

In previous releases, you tailored default precompiler behavior by assembling an EDBPPARM module and linking the result with the precompiler. In Version 18.0, you create a parameter module in a similar way, but link the result into your custom load library as a stand-alone module called IDMSPPRM. The COBOL and PL/I precompilers dynamically load this module at runtime.

For more information about precompiler parameters, see the *DML Reference – COBOL Guide* or *DML Reference - PL/I Guide*.

z/OS assemble and link

To create a precompiler parameter module in z/OS, execute the z/OS Assemble and Link-edit JCL (see page 178) substituting the name of your IDMSPPRM source member and inserting the following binder statements:

```
SETOPT PARM(AMODE=31,RMODE=ANY)
NAME IDMSPPRM(R)
```

CA Culprit for CA IDMS Profile Module

In Version 18.0, Culprit dynamically loads a profile module. As a result, you can tailor the behavior of Culprit without having to relink modules when applying maintenance..

In previous releases, you tailored default Culprit behavior by assembling a CULXPROF module using the CULMPROF macro and linking the result with various Culprit CUSTLIB programs. In Version 18.0, you create a profile module using the same macro, but link the result into your custom load library as a stand-alone module named CULPPROF. Culprit dynamically loads this module at runtime.

Note: For more information about profile parameters, see the *CA Culprit for CA IDMS Reference Guide.*

z/OS assemble and link

To create a Culprit profile module in z/OS, execute the z/OS Assemble and Link-edit JCL (see page 178) substituting the name of your CULPPROF source member and inserting the following binder statements.

```
SETOPT PARM(AMODE=24,RMODE=24)
NAME CULPPROF(R)
```

Separation of Security Classification Modules

The security classification modules that are used to assign activity numbers to application functions are now dynamically loaded instead of being linked with CA-distributed code. To classify application functions by activity in Version 18.0, you assemble and independently link the following stand alone modules into your custom loadlib as follows:

- **IDMSCTAB** – To assign activity numbers to DCMT commands. This module is created by compiling the #CTABGEN macro. RHDCCTAB is no longer a user-configurable module. All DCMT discrete security is specified in the module IDMSCTAB.
- **IDMSGTAB** – To assign activity numbers to debugger commands. This module is created by compiling the #GTABGEN macro. The parameters are the same as for the old #DGTBGEN macro, which is no longer used.
- **IDMSUTAB** – To assign activity numbers to utility statements. This module is created by compiling the #UTABGEN macro.

Note: For more information about how to assemble these modules for assigning security classes to activities, see the *Security Administration Guide*.

Separation of User Exits

User exits are no longer linked with the code that invokes them. This change impacts the following exits:

- Exits invoked in the CICS environment are now linked with the IDMSINTC or IDMSINTL stub module instead of the interface. For more information, see Interface Configuration Enhancements (see page 42).
- The CA IDMS DML Online exit (USDMLXIT) is no longer linked with DMLO, and it is dynamically loaded at run time. For more information, see CA IDMS Tools Installation and Configuration Enhancements (see page 167).
- Other named user exits, such as IDMSCALC and IDMSIOX2, are no longer linked with the CA modules that invoke them. Instead, they are linked with the IDMSUXIT vector module. For more information about this new facility, see Named User Exit Enablement (see page 39).

CA ADS Built-in Function Management

In previous releases, some sites improved CA ADS performance by tailoring the link of ADSOMAIN to include their own or CA-supplied built-in functions. The management of CA ADS BIFs has been enhanced in Version 18.0 so there is no longer a need for taking this action.

More information:

Improved BIF Management (see page 155)

Product Intent Declaration

This release of CA IDMS has simplified software delivery by always installing all CA IDMS related products. This process makes it easier to do the following activities:

- Install and maintain products with fewer choices and a simplified process.
- Install the software once and use different mixes of products in different environments.
- Try out a new product.

Legal use of CA products is still subject to your license agreement and verified through LMP key checking. As with previous releases, if a required LMP key is not installed, any attempt to use the associated product results in warning messages.

You can proactively prevent unauthorized product use by declaring your intended product use. A product intent module is a list of the products that are intended to be used in a given environment; you can assemble one and place it in your custom load library.

If you do not have a product intent module, all products are treated as if they are intended for use. When you have a product intent module only the identified products are eligible for use. An attempt to use a product that is not identified by your product intent module results in a task failure.

Note: During the CA-provided configuration process, a product intent module is created automatically and placed in your custom load library. The set of products identified for use is based on information you provide during configuration.

Create a Product Intent Module

The easiest way to create a product intent module is to begin with the sample RHDCPINT source module installed with CA IDMS. After creating the product intent module, you need to assemble and link the module into to your custom load library.

To create a product intent module and link it to your custom load library

1. Access the sample RHDCPINT module.
2. Uncomment the lines for the products you intend to use.
3. Save it to your custom source library.
4. Execute the z/OS Assemble and Link-edit JCL.

   Substitute the name of your source member and insert the following binder statements:

   ```
   ENTRY PINTEP1
   NAME RHDCPINT(R)
   ```

Example: Product Intent Module Creation

The example product intent module authorizes the use of these CA products:

- CA IDMS/DB
- CA IDMS Server
- CA IDMS/DC
- CA IDMS Performance Monitor
- CA IDMS Presspack
- CA ADS
- CA IDMS SQL
- CA IDMS Culprit for CA IDMS
- CA IDMS Dictionary Module Editor
- CA IDMS DML Online

```
         TITLE 'RHDCPINT - Product Intent Table'
* RHDCPINT
*   CA IDMS Core Products
*   ---------------------
        #DEFPINT ADS             CA ADS
*       #DEFPINT ADSB            CA ADS Batch
*       #DEFPINT APPC            CA ADS APPC
*       #DEFPINT CMS             CA IDMS CMS Option
*       #DEFPINT DDS             CA IDMS DDS
*       #DEFPINT DICTLODR        CA IDMS Dictionary Loader
*       #DEFPINT EDPAUDIT        CA EDP Auditor
*       #DEFPINT ICMS            CA ICMS
        #DEFPINT IDMSCULP        CA Culprit for CA IDMS
        #DEFPINT IDMSDB          CA IDMS/DB
        #DEFPINT IDMSDC          CA IDMS/DC
*       #DEFPINT OLQ             CA OLQ Online Query for CA IDMS
*       #DEFPINT OTPMON          CA IDMS/TP Monitor
        #DEFPINT PERFMON         CA IDMS Performance Monitor
        #DEFPINT PRESPACK        CA IDMS Presspack
        #DEFPINT SERVER          CA IDMS Server
        #DEFPINT SQL             CA IDMS SQL
*       #DEFPINT UCF             CA IDMS UCF

*   CA IDMS Transparency Options
*   ----------------------------
*       #DEFPINT DBOMP/T         CA IDMS DBOMP Transparency
*       #DEFPINT DL1/T           CA IDMS DLI Transparency
*       #DEFPINT TOTAL/T         CA IDMS TOTAL Transparency
*       #DEFPINT VSAM/T          CA IDMS VSAM Transparency

*   CA IDMS Tools Products
*   ----------------------
*       #DEFPINT ADSALIVE        CA ADS Alive
*       #DEFPINT ADSTRACE        CA ADS Trace
*       #DEFPINT DBANALYZ        CA IDMS/DB Analyzer
*       #DEFPINT DBAUDIT         CA IDMS/DB Audit
*       #DEFPINT DBEXTRCT        CA IDMS Extractor
*       #DEFPINT DBREORG         CA IDMS/DB Reorg
*       #DEFPINT DICTMIGR        CA IDMS Dictionary Migrator
        #DEFPINT DMLO            CA IDMS DML Online
        #DEFPINT DME             CA IDMS Dictionary Module Editor
*       #DEFPINT DQF             CA IDMS Dictionary Query Facility
*       #DEFPINT ENFORCER        CA IDMS Enforcer
*       #DEFPINT JRNLANLZ        CA IDMS Journal Analyzer
*       #DEFPINT LOGANALZ        CA IDMS Log Analyzer
*       #DEFPINT MASTRKEY        CA IDMS Masterkey
*       #DEFPINT ONLINLOG        CA IDMS Online Log Display
*       #DEFPINT SASO            CA IDMS SASO
*       #DEFPINT SCHEMAPR        CA IDMS Schema Mapper
*       #DEFPINT TPSORT          CA IDMS/DC Sort
*       #DEFPINT TSKANALZ        CA IDMS Task Analyzer

*   CA Endevor/DB for IDMS
*   ----------------------
*       #DEFPINT ENDEVOR/DB      CA Endevor/DB for IDMS
        #DEFPINT TYPE=GENERATE
        END
```

Display Product Intent

The DCPROFIL system task now shows which products you intend to use.

```
                    * Product Intent Status *
CA IDMS Core Products                 CA IDMS Tools Products
---------------------                 ----------------------
CA ADS                       YES   CA ADS Alive                NO
CA ADS Batch                  NO   CA ADS Trace                NO
CA ADS APPC                   NO   CA IDMS/DB Analyzer         NO
CA IDMS CMS Option            NO   CA IDMS/DB Audit            NO
CA IDMS DDS                  YES   CA IDMS/DB Extractor         NO
CA IDMS Dictionary Loader     NO   CA IDMS/DB Reorg             NO
CA EDP Auditor                NO   CA IDMS Dictionary Migrator  NO
CA ICMS                       NO   CA IDMS DML Online           YES
CA CULPRIT for CA IDMS        NO   CA IDMS Dictionary Module Editor  YES
CA IDMS/DB                   YES   CA IDMS Dictionary Query Facility  NO
CA IDMS/DC                   YES   CA IDMS Enforcer             NO
CA OLQ                        NO   CA IDMS Journal Analyzer     NO
CA IDMS/TP Monitor            NO   CA IDMS Log Analyzer         NO
CA IDMS Performance Monitor  YES   CA IDMS Masterkey            NO
CA IDMS Presspack            YES   CA IDMS Online Log Display   NO
CA IDMS Server               YES   CA IDMS SASO                 NO
CA IDMS SQL                  YES   CA IDMS Schema Mapper        NO
CA IDMS UCF                  YES   CA IDMS/DC Sort              NO
                                CA IDMS Task Analyzer         NO

CA IDMS Transparency Options          CA Endevor/DB for IDMS
----------------------------          ----------------------
CA IDMS DBOMP Transparency    NO   CA Endevor/DB for CA IDMS    NO
CA IDMS DLI Transparency      NO
CA IDMS TOTAL Transparency    NO
CA IDMS VSAM Transparency     NO
```

Change the Product Intent

To change your intended product use, you need to create a new product intent module and use the DCMT VARY NUCLEUS command to make the new RHDCPINT module available in each of your CA IDMS systems.

Named User Exit Enablement

To facilitate applying system maintenance, user-written exits are now linked separately from the code that invokes the exit.

With Version 18.0, all named user-written exits are linked with IDMSUXIT, which is a list of VCONs that address the exits to be invoked. Exits invoked in the CICS environment and exits that are loaded dynamically are not linked with IDMSUXIT. To enable the use of other named exits, you need to link your exit module with IDMSUXIT and place it in your custom load library, and then include this in your STEPLIB concatenation for batch jobs and in your CDMSLIB concatenation for Central Version startup.

You can determine the named exits that are enabled within a DC/UCF system using the DCPROFIL system task.

More information:

Interface Configuration Enhancements (see page 42)
Enhanced DCPROFIL Output (see page 101)

Entry Point Names

The following table identifies the exits that can be linked with IDMSUXIT and the name of the entry point that must be used in each case.

Exit	Entry Point Name
User ID exit for batch jobs using the 10.2 services interface	BTCIDXIT
RELOAD error exit	DBLUEREX
IDD batch compiler exit *	IDDEXITB
IDD online compiler exit *	IDDEXITO
ARCHIVE JOURNAL exit	IDMSAJNX
IDMSCALC exit	IDMSCLCX
IDMSDBIO duplex exit	IDMSDPLX
IDMSDBIO IO statistics exit	IDMSIOXT
IDMSDBIO I/O exit	IDMSIOX2
IDMSDBIO journal exit	IDMSJNL2
OLQ DML exit	OLQDMLX
Schema batch compiler exit*	SCHEXITB
Schema online compiler exit*	SCHEXITO
Subschema batch compiler exit *	SUBEXITB
Subschema online compiler exit *	SUBEXITO
SYSGEN batch compiler exit *	SGNEXITB
SYSGEN online compiler exit *	SGNEXITO
Ticker exit	TCKREXIT
User ID exit for batch jobs	USRIDXIT
Wait exit	WAITEXIT
Write to operator exit **	WTOEXIT

Exit	Entry Point Name
Write to operator reply exit **	WTOREXIT

* See the section Compiler Exits and Entry Points (see page 41)
**See the section Enabling WTOEXIT and WTOREXIT (see page 41)

Compiler Exits and Entry Points

Each compiler calls separate exit entry points in each of the batch and online environments. If you have separate exit modules for each compiler and each environment, you must change their entry points as indicated in the table in Entry Point Names. There are two ways you can change an entry point:

- Change the entry point name in the source code and recompiling the program.
- Use appropriate binder directives to change the entry point name when including the exit module in the link of IDMSUXIT.

We recommend changing the program source code and recompiling.

If you use the same exit module for more than one compiler or in more than one environment, you can continue to do so by changing the exit module to include entry points for each of the compilers and environments in which the module is to be used. Alternatively, you can use binder directives to change the name of one or more external references to match the name of your module's entry point in IDMSUXIT. We recommend changing your exit program.

WTOEXIT and WTOREXIT Implementation

There are three ways you can implement WTO and WTOR exits:

1. By linking your exit routines with IDMSUXIT.
2. By specifying the name of the WTO and WTOR exits using DC/UCF system startup parameters.
3. In z/VSE, by linking your exit routines as phases WTOEXIT and/or WTOREXIT.

Note: If the second or third technique is used, CA IDMS dynamically loads the exit modules during startup and ignores any corresponding exit routine linked with IDMSUXIT.

For more information on using an OPS/MVS API rule as a replacement for the WTOEXIT, see the Appendix Sample OPS/MVS API rule (see page 181). This can lead to improved performance when using the CA IDMS zIIP feature.

Note: You must link your exit modules with IDMSUXIT if you want to invoke the WTO and WTOR exits in batch jobs.

IDMSUXIT Module Creation

To create an IDMSUXIT module in z/OS, execute the z/OS Link-edit JCL using the appropriate binder input statements to identify the exits that you want to enable.

The following binder input represents the minimum statements that you need to link-edit IDMSUXIT. This binder input disables all user exits, so it is equivalent to the installed version of the module:

```
ORDER IDMSUXIT
INCLUDE CAGJLOAD(IDMSUXIT)
ENTRY UEXITS
SETOPT PARM(REUS=NONE)
NAME IDMSUXIT(R)
```

Example: Binder statements needed to enable IDMSCALC and ARCHIVE JOURNAL user exits

```
ORDER IDMSUXIT
INCLUDE CAGJLOAD(IDMSUXIT)
INCLUDE CUSTLIB(idmsclcx) IDMSCALC user exit
INCLUDE CUSTLIB(idmsajnx) ARCHIVE JOURNAL exit
ENTRY UEXITS
SETOPT PARM(REUS=NONE)
NAME IDMSUXIT(R)
```

idmsclcx

Specifies the name of your IDMS CALC exit module.

idmsajnx

Specifies the name of your ARCHIVE JOURNAL exit module.

Interface Configuration Enhancements

CA IDMS Version 18.0 minimizes the need for relinking an interface due to release upgrades and maintenance. This enhancement reduces and simplifies the time and effort involved in these activities. The following interfaces were enhanced to simplify installation and maintenance:

- CICS interfaces
- CICS Resyncronization programs
- UCF interfaces
- UDAS interfaces

CICS Interfaces

In this release of CA IDMS, you link user-generated interface code only with stub modules. As a result, you will typically need to regenerate your interfaces only when new business requirements require changes in options or exit routines. This differs from previous releases where you had to regenerate as a result of applying maintenance or upgrading to a new release.

For Version 18.0, the CA IDMS CICS interface consists of three components:

- A stub module generated by the IDMSCINT or IDMSCINL macro that is linked with application programs requiring CA IDMS services. This component is unchanged from prior releases.
- A stub module that is linked with a site-specific CICSOPT options module and exit routines.
- A new site-independent interface module containing only CA-supplied code.

Both the standard IDMSINTC and the IDMSINTL interfaces conform to this new architecture.

Note: Before using the Version 18.0 runtime libraries in your CICS system, reassemble your CICSOPT module or IDMSINTL module and link it with the appropriate stub module.

Create an IDMSINTC Interface Program

To create an IDMSINTC interface program under z/OS:

1. Create a CICSOPT source module as follows:

```
      GBLC & MODNAME
&MODNAME SETC CICSOPT'
      CICSOPT cicsopt-parameters
      END
```

2. Save the CICSOPT source module in your custom source library.
3. Assemble and link the module into your custom load library by executing the z/OS Assemble and Link-Edit JCL (see page 178).

4. Substitute the name of your CICSOPT source member and insert the following binder statements:

```
ORDER DFHEAI
INCLUDE CUSTLIB(optixit)  (Optional)
INCLUDE CUSTLIB(optiqxit)  (Optional)
INCLUDE CUSTLIB(usridxit)  (Optional)
INCLUDE CAGJLOAD(IDMSCSTB)
ENTRY STARTUP
SETOPT PARM(AMODE=31,RMODE=24,REUS=NONE)
NAME idmsintc(R)
```

optixit

Specifies the name of your OPTIXIT exit routine.

optiqxit

Specifies the name of your OPTIQXIT exit routine.

usridxit

Specifies the name of your USRIDXIT exit routine.

idmsintc

Specifies the name of your IDMSINTC interface load module.

To create an IDMSINTC interface program under z/VSE:

1. Assemble and catalog a CICSOPT options table using the sample JCL in z/VSE Assemble JCL.

   Modify the JCL by substituting the following in place of the *Assembler input statements*:

```
        PUNCH 'CATALOG cicsopts.OBJ REPLACE=YES'
        CICSOPTS TITLE 'CA IDMS CICS OPTIONS MODULE'
        GBLC &MODNAME
&MODNAME. SETC 'cicsopts'
        CICSOPT cicsopt-parameters
        END
```

2. Link the IDMSINTC interface program using the sample JCL in z/VSE Link JCL.

   Modify the JCL by substituting the following statements in place of the *Linkage editor control statements*:

```
PHASE idmsintc,*
INCLUDE DFHEAI
INCLUDE optiexit  (Optional)
INCLUDE optiqxit  (Optional)
INCLUDE usridxit  (Optional)
INCLUDE usridxit  (Optional)
INCLUDE cicsopts
INCLUDE IDMSLST6
INCLUDE DFHEAI0
MODE    AMODE(31),RMODE(24)
ENTRY   STARTUP
```

   cicsopts

   Specifies the name you choose for your IDMSINTC options table.

   Note: You can have more than one options table with different names.

   idmsintc

   Specifies the name you choose for your IDMSINTC stub program.

   Note: You can have more than one stub program with different names.

   optixit

   Specifies the name of your OPTIXIT exit routine.

   optiqxit

   Specifies the name of your OPTIQXIT exit routine.

   usridxit

   Specifies the name of your USRIDXIT exit routine.

   idmsintc

   Specifies the name of your IDMSINTC interface load module.

Create an IDMSINTL Interface Program

To create an IDMSINTL interface program

1. Create an IDMSINTL source module as described in the *System Operations Guide*.
2. Save the IDMSINTL source module in your custom source library.
3. Assemble and link it into your custom load library by executing the z/OS Assemble and Link-Edit JCL (see page 178)

4. Substitute the name of your IDMSINTL source member and insert the following binder statements:

```
ORDER DFHEAI
INCLUDE CUSTLIB(optixit)   (Optional)
INCLUDE CAGJLOAD(IDMSLSTB)
ENTRY STARTUP
SETOPT PARM(REUS=NONE,AMODE=31,RMODE=24)
NAME idmsintl(R)
```

optixit

Specifies the name of your OPTIXIT exit routine.

idmsintl

Specifies the name of your IDMSINTL interface load module.

Create the Resynchronization Program

Linking the IDMSCSYN module with an IDMSCINT module creates the resynchronization program. A separate resynchronization program must be created for each version of the CA IDMS interface module (IDMSINTC) that is used within a given CICS system.

In order to simplify future maintenance, the IDMSCSYN module contains minimal executable code. It simply passes control to module IDMSCCSY which contains most of the executable code to perform resynchronization between CICS and IDMS. If maintenance is applied to IDMSCCSY, it is not necessary to re-link your resynchronization program.

Resynchronization Program Link Edit (z/OS)

To link a resynchronization program for a CICS environment in z/OS, execute the z/OS Link-Edit JCL inserting the following binder statements.

```
ORDER   DFHEAI,IDMSCSYN
INCLUDE CAGJLOAD(IDMSCSYN)
INCLUDE CUSTLIB(idmscint)
ENTRY   DFHEAI
SETOPT PARM(AMODE=31,REUS(REFR),RMODE=24)
NAME idmsrsyn(R)
```

idmscint

Specifies the name of your IDMSCINT interface module.

UCF Interfaces

With Version 18.0, the UCF interfaces have been configured to make installation and maintenance easier by separating user-generated code from the main UCF interface routines. Because of these changes, you must regenerate all of your UCF interfaces when upgrading to Version 18.0.

The remainder of this section describes changes in the way you link your UCF interfaces.

For information on all of the necessary steps to generate a UCF interface, see the *System Operations Guide*.

UCF CICS Interface

Prior to Version 18.0, UCF CICS front-end and abort session programs contained all of the executable code for UCF front-end and abort session processing, respectively. Now these programs only contain an options table and a small stub module that calls the main processing routines.

Create a UCF CICS Front-end Program

To create a UCF CICS front-end program

1. Create a UCFCICS source module as described in the *System Operations Guide*.
2. Save the UCFCICS source module in your custom source library.
3. Assemble and link it into your custom load library by executing the z/OS Assemble and Link-Edit JCL (see page 178)
4. Substitute the name of your UCFCICS source member and insert the following binder statements:

```
INCLUDE CAGJLOAD(UCFCICX0)
INCLUDE CUSTLIB(ucffet)
INCLUDE CUSTLIB(idmscint)
INCLUDE CUSTLIB(ucfprint)  (Optional)
INCLUDE CUSTLIB(usridxit)  (Optional)
SETOPT PARM(AMODE=31,RMODE=24)
ENTRY ucfcicse
NAME ucfcics(R)
```

ucffet

Specifies the name of your UCF front-end table.

idmscint

Specifies the name of your IDMSCINT interface module.

ucfprint

Specifies the name of your UCFPRINT exit routine.

usridxit

Specifies the name of your USRIDXIT exit routine.

ucfcicse

Specifies the entry point name of your UCFCICS options module:

Note: Use UCFCXEP1 or the name specified as the label on the assembler instruction that invokes the #UCFCICS macro. The assembler label is optional. You cannot use UCFCXEP1 as the assembler label.

ucfcics

Specifies the name of your UCF CICS front-end load module.

5. Define the ucfcics program in the CICS CSD.

   For a template, see the definition of PROGRAM(UCFCICS) in the sample CICSCSD. Note that the program must be defined as RESIDENT.

Create a UCF CICS Abort Session Program

To create a UCF CICS abort session program

1. Create a UCFCICSZ source module as described in the System Operations Guide.
2. Save the UCFCICSZ source module in your custom source library.
3. Assemble and link it into your custom load library by executing the z/OS Assemble and Link-edit JCL (see page 178)
4. Substitute the name of your UCFCICSZ source member and insert the following binder statements:

```
INCLUDE CAGJLOAD(UCFCIZX0)
INCLUDE CUSTLIB(idmscint)
ENTRY CICZEP
SETOPT PARM(AMODE=31,RMODE=24)
NAME ucfcicz(R)
```

idmscint

Specifies the name of your IDMSCINT interface module.

ucfcicz

Specifies the name of your UCF CICS abort session load module.

5. Define the ucfcicz program in the CICS CSD.

   For a template, see the definition of PROGRAM(UCFCICZ) in the sample CICSCSD. Note that the program must be defined as RESIDENT.

UCF Batch Interface

Prior to Version 18.0, UCF batch front-end and print support programs contained all the executable code for UCF front-end processing. Now these programs only contain an options table and a small set of executable code that calls the main processing routines that reside in separate modules.

Create a UCF Batch Front-end Program

To create a UCF batch front-end program

1. Create a UCFBTCH source module as described in the *System Operations Guide*.
2. Save the UCFBTCH source module in your custom source library.
3. Assemble and link it into your custom load library by executing the z/OS Assemble and Link-edit JCL (see page 178)
4. Substitute the name of your UCFBTCH source member and insert the following binder statements:

```
INCLUDE CUSTLIB(idmsopti)  (Optional)
ENTRY ucfbtche
SETOPT PARM(REUS=NONE,AMODE=31,RMODE=24 )
NAME dcucfbtc(R)
```

idmsopti

Specifies the name of your IDMSOPTI module.

ucfbtche

Specifies the entry point name of your UCF Batch front end load module.

Note: Use UCFBTCHX or the name specified as the label on the assembler instruction that invokes the #UCFBTCH macro.

dcucfbtc

Specifies the name of your UCF batch front-end load module.

Create a UCF Batch Print Support Program

To create a UCF batch print support program

1. Create a UCFBTCH source module as described in the *System Operations Guide*.
2. Save the UCFBTCH source module in your custom source library.
3. Assemble and link it into your custom load library by executing the z/OS Assemble and Link-Edit JCL (see page 178)

4. Substitute the name of your UCFBTCH source member and insert the following binder statements:

```
INCLUDE CUSTLIB(idmsopti)  (Optional)
ENTRY ucfbatpe
SETOPT PARM(REUS=NONE,AMODE=31,RMODE=24)
NAME ucfbatp(R)
```

idmsopti

Specifies the name of your IDMSOPTI module.

ucfbatpe

Specifies the entry point name of your UCF Batch print support load module.

Note: Use UCFBTCHX or the name specified as the label on the assembler instruction that invokes the #UCFBTCH macro.

ucfbatp

Specifies the name of your UCF batch printer support load module.

UCF DC Interface

Prior to Version 18.0, UCF DC front-end programs contained all the executable code for UCF front-end processing. Now these programs only contain an options table and a small set of executable code that calls the main processing routines that reside in separate modules.

Create a UCF DC Front-end Program

To create a UCF DC front-end program

1. Create a UCFOPTS source module as described in the *System Operations Guide*.
2. Save the UCFOPTS source module in your custom source library.
3. Assemble and link it into your custom load library by executing the z/OS Assemble and Link-edit JCL (see page 178)
4. Substitute the name of your UCFOPTS source member and insert the following binder statements:

```
INCLUDE CAGJLOAD(RHDCDBDC)
INCLUDE CUSTLIB(ucffet)
SETOPT PARM(AMODE=31)
ENTRY DBDCEP1
NAME ucfdbdc(R)
```

ucffet

Specifies the name of your UCF front-end table.

ucfdbdc

Specifies the name of your UCF DC front-end load module.

UCF TSO Interface

Prior to Version 18.0, UCF TSO front-end programs contained all the executable code for UCF front-end processing. Now these programs only contain an options table and a small set of executable code that calls the main processing routines that reside in separate modules.

Create a UCF TSO Front-end Program

To create a UCF TSO front-end program

1. Create a UCFTSO source module as described in the *System Operations Guide*.
2. Save the UCFTSO source module in your custom source library.
3. Assemble and link it into your custom load library by executing the z/OS Assemble and Link-Edit JCL (see page 178)
4. Substitute the name of your UCFTSO source member and insert the following binder statements:

```
INCLUDE CUSTLIB(idmsopti)  (Optional)
ENTRY UCFTSOX
SETOPT PARM(REUS=NONE,AMODE=31,RMODE=24)
NAME ucftso(R)
```

idmsopti

Specifies the name of your IDMSOPTI module.

ucftso

Specifies the name of your UCF TSO front-end load module.

UDAS Interfaces

With Version 18.0, the UDAS interfaces have been configured to make installation and maintenance easier by separating user-generated code from the main UDAS interface routines. Because of these changes, you must regenerate all of your UDAS interfaces when upgrading to Version 18.0.

The remainder of this section describes changes in the way you link your UDAS interfaces.

For information on all of the necessary steps to generate a UDAS interface, see the *System Operations Guide*.

UDAS CICS Interface

Prior to Version 18.0, UDAS CICS front-end programs contained all the executable code for UDAS CICS front-end processing. Now these programs only contain an options table and a small stub module that calls the main processing routines that reside in separate modules.

Create a UDAS CICS Front-end Program

To create a UDAS CICS front-end program

1. Create a UDASCIC source module as described in the *System Operations Guide*.
2. Save the UDASCIC source module in your custom source library.
3. Assemble and link it into your custom load library by executing the z/OS Assemble and Link-Edit JCL (see page 178)
4. Substitute the name of your UDASCIC source member and insert the following binder statements:

```
ORDER DFHEAI
INCLUDE CAGJLOAD(UDSCICX0)
INCLUDE CUSTLIB(idmscint)
ENTRY udascice
SETOPT PARM(AMODE=31,RMODE=24)
NAME udascic(R)
```

idmscint

Specifies the name of your IDMSCINT interface module.

udascice

Specifies the entry point name of your UDASCIC options module.

Note: Use UDSCXEP1 or the name specified as the label on the assembler instruction that invokes the #UDASCIC macro.

udascic

Specifies the name of your UDAS CICS front-end load module.

UDAS Batch Interface

Prior to Version 18.0, UDAS batch front-end programs contained all the executable code for UDAS batch front-end processing. Now these programs only contain an options table and a small stub module that calls the main processing routines that reside in separate modules.

Create a UDAS Batch Front-end Program

To create a UDAS batch front-end program

1. Create a UDASBCH source module as described in the *System Operations Guide*.
2. Save the UDASBCH source module in your custom source library.
3. Assemble and link it into your custom load library by executing the z/OS Assemble and Link-Edit JCL (see page 178)
4. Substitute the name of your UDASBCH source member and insert the following binder statements:

```
INCLUDE CUSTLIB(idmsopti)  (Optional)
ENTRY udasbche
SETOPT PARM(REUS=NONE,AMODE=31,RMODE=24)
NAME udasbchi(R)
```

idmsopti

Specifies the name of your IDMSOPTI module

udasbche

Specifies the entry point name of your UDAS Batch front-end load module:

Note: Use UDASBCH or the name specified as the label on the assembler instruction that invokes the #UDASBCH macro.

udasbchi

Specifies the name of your UDAS batch front-end load module

Optional APAR Replacement

Optional APARs are no longer used to tailor CA IDMS behavior. In Version 18.0, optional APARs are replaced either with syntax or numbered options that are implemented through flag settings in the RHDCOPTF module.

The following table lists the optional APARs that were eliminated in Version 18.0 and indicates how to achieve equivalent functionality using standard CA IDMS facilities:

APAR	Description	Version 18.0 Replacement
CI98234	CA IDMS Culprit: override DDNAME for SYS004 file	Profile parameter: OUTDD
CS82781	CA IDMS DC/UCF: Override the default queue retention	SYSGEN: SYSTEM Queue Retention parameter
CS98817	CA Online Query: Allow blocked OLQ batch SYSIPT file	Numbered option 283

APAR	Description	Version 18.0 Replacement
GS09836	CA IDMS: DBCS-compatible menu box character	Numbered option 284
GS23359	CA IDMS Culprit: Suppress 750009 message	Profile parameter: RECMSG
GS26164	CA IDMS: Allow recursive IDD program associations	Numbered option 287
GS27525	CA IDMS: Override CV Retry message codes	SYSIDMS: CVRETRY_MSG_CODES
GS28742	CA IDMS DC/UCF: Override message buffer size for debugging	SYSGEN: Debug Message Buffers
GS33482	CA Online Query: Suppress password reporting in OLQ batch	Numbered option 286
GS48843	CA IDMS: Protect ASF/IDB signon fields	Numbered option 289
GS48844	CA IDMS: Protect ASF/IDB signon fields	Numbered option 289
GS53980	CA IDMS: Allow CV to run attached	Invoke it using a name of IDMSCV (alias for RHDCOMVS)
GS81247	CA IDMS Performance Monitor: Allow setting refresh interval on a UCF terminal	Numbered option 288
LS17909	CA IDMS: Prevent cancelling driver tasks	Obsolete. Drivers cannot be cancelled.
LS19767	CA IDMS Culprit: Specify SSN formatting	Profile parameter: SSN
LS25619	CA IDMS DC/UCF: DBCS-compatible pageable map definition in MAPC	Numbered option 307
LS25622	CA IDMS: DBCS-compatible editing for ASF initial values	Numbered option 307
LS36498	CA IDMS Masterkey: Enable tailored PFKEY defaults	PFKEY table in SSKTPARM option module
LS38822	CA ADS: Override default ADS comment delimiter	SYSGEN: ADSO Comment Delimiter

APAR	Description	Version 18.0 Replacement
LS39409	CA IDMS SQL Option: Generate uppercase AMC messages	Obsolete. Uppercase is used if necessary based on RHDCCODE.
LS39411	CA IDMS Culprit: DBCS-extended character recognition	Profile parameter: HD
LS40898	CA IDMS DC/UCF: Override report line length for online reports	SYSGEN: SYSTEM Overriding Report Line Length
LS49906	CA IDMS: Force use of 10.2 interface to avoid CSV002I	Obsolete. Fixed in code.
LS58334	CA IDMS Dictionary Migrator: Avoid duplication of EDIT/CODE tables.	USMTPARM: TABNULL forces "EXCLUDE VALUES NULL"
LS63145	CA ADS: Force use of 4-digit years.	Numbered option 285
LS63259	CA IDMS: Override year to be treated as base for 20nn	SYSGEN: SYSTEM EVAL Base Year SYSIDMS: EVAL_BASE_YEAR
LS71078	CA ADS Alive: Disable use of NOSTOP	Numbered option 303
LS71683	CA ADS Alive: Allow special characters in record display	USGTPARM: CHARTAB table defines valid display characters
LS78113	CA IDMS DML Online: User-tailored message from USDMLXIT	USDTPARM: Defines text of messages
LS78489	CA ADS: Upward compatibility for ADS ZERO test	Numbered option 290
LS80233	CA IDMS: Set OPER refresh to 10 seconds	Numbered option 291
LS80772	CA IDMS: Make IDD USAGE RETRIEVAL the default.	Numbered option 292
LS85112	CA ADS: Support debugging dialogs > 64K	Numbered option 293
LS88822	CA IDMS Dictionary Migrator: Generate MODIFY verb instead of ADD	USMTPARM: EXPMOD=YES forces generation of MODIFY
LS88828	CA IDMS: Override default of 1 sort passes to catch more errors	SYSIDMS: DBAN_SORT_PASSES

APAR	Description	Version 18.0 Replacement
LS88839	CA IDMS: Override ARCHIVE JOURNAL default 10% free space warning threshold	SYSIDMS: ARCHIVE_JOURNAL_WARNING_ PERCENT
LS88860	CA IDMS: Set low and high centuries for validation	SYSGEN: SYSTEM century validation SYSGEN: SYSTEM EVAL Low Century SYSGEN: SYSTEM EVAL High Century SYSIDMS: CENTURY_VALIDATION SYSIDMS: EVAL_LOW_CENTURY SYSIDMS: EVAL_HIGH_CENTURY
LS89002	CA IDMS Culprit: Override year to be treated as base for 20nn	Profile parameter: US12YR
LS94973	CA IDMS DML Online: Use BIND instead of BIND ALL	Numbered option 302
LS95989	CA IDMS: Disable SVC screening	SYSIDMS: DISABLE_SVC_SCREEN
LS96927	CA ADS Alive: Avoid displaying dialog selection list unless dialog name supplied	Numbered option 304
QO38508	Force task snaps	SYSGEN: SNAP TASK … TRACE DCMT: SNAP TASK … TRACE
QS00476	CA Endevor/DB for CA IDMS: Preserve map datetime stamp during migration	Numbered option 294
QS00477	CA Endevor/DB for CA IDMS: Retain action CLEs during compress	Numbered option 295
QS00481	CA Endevor/DB for CA IDMS: Make current date the default start date in online management facility.	Numbered option 296
QS00482	CA Endevor/DB for CA IDMS: Generate DCMT V PRO ENABLE before V NC	Numbered option 297
QS00514	CA Endevor/DB for CA IDMS: Don’t generate SIGNON statement	Numbered option 298
QS00515	CA Endevor/DB for CA IDMS: Don’t generate DCUF SET DICTNAME	Numbered option 299

APAR	Description	Version 18.0 Replacement
QS00516	CA Endevor/DB for CA IDMS: Don't generate SIGNON statements	Numbered option 300
QS13727	CA IDMS DML Online: Increase stack from 4 to 16K	USDTPARM: STKSIZE
QS15800	CA IDMS Database Extractor: Specify target DMCL	USVTPARM: SYSIDMS_DMCL
QS16472	CA IDMS Journal Analyzer: Selectively bypass run units	BYPASS parameter
QS21024	CA IDMS Task Analyzer: Disable exits	SYSIDMS: TASK_ANALYZER_EXITS
QS29629	CA Endevor/DB for CA IDMS: Don't route messages to console or system log	Numbered option 301
QS39434	CA IDMS: Force RC of 0 if PRINT/ARCHIVE LOG has nothing to to print.	Numbered option 282
QS56063	CA IDMS: Override default of 1 sort pass to catch more errors	SYSIDMS: DBAN_SORT_PASSES
QS68666	CA IDMS DC/UCF: Override retention for message queues	SYSGEN: SYSTEM Message Retention
QS80740	Avoid SKTHRED BLDL failure under SAS 9	Obsolete. Fixed in code
QS90515	CA Endevor/DB for CA IDMS: Protect fields on signon screen	Numbered option 306

For a complete list of the numbered options that are supported in Version 18.0, see PIB RI29610.

Chapter 4: Non-Stop Processing

This chapter describes the enhancements that enable CA IDMS to support continuous operations.

This section contains the following topics:

Expanded Statistics Fields

Several fields used to collect CA IDMS run-time statistics have been enlarged to accommodate the larger values that can be expected with faster processors and non-stop operations.

The field size has been increased from single to double words for statistics related to storage management and CPU utilization. These changes affect the following DSECTs:

- #SCADS—Subtask control area
- #SCTDS—Storage control table
- #SMTDS—Storage management table

The following system tasks and operator commands now display larger values:

- OPER WATCH STORAGE SUBPOOL
- OPER WATCH STORAGE POOL USAGE <nn>
- DCMT DISPLAY ACTIVE STORAGE <pool number>
- DCMT DISPLAY MPMODE
- DCMT DISPLAY STATISTICS SYSTEM
- DCMT DISPLAY SUBTASKS
- DCMT DISPLAY SUBTASK <nn>

Where space is limited on the output screen, large values are displayed in scientific notation.

Interval Roll for Statistics

System-wide statistics and histograms can be now automatically written to the log and cleared at a specified time and day. This process ensures that the values of these statistics are synchronized by avoiding overflow. It also makes analysis easier since statistics are gathered over a standard time interval.

A new interval roll time specifies the time of day and day frequency when the statistics are written and reset. When the interval roll time is reached, current system-wide statistics and histograms are written to the CA IDMS log, and then cleared in preparation for the next collection interval.

You can set the interval roll time and day frequency using the new parameters on the SYSGEN SYSTEM statement and the DCMT VARY STATISTICS command. The DCMT DISPLAY STATISTICS command has been enhanced to display the interval roll time, and the DCMT WRITE STATISTICS command provides the ability to both write and clear the statistics.

The following new fields have been added to the #STLDS DSECT that describes the header portion of the statistics records written to the log:

- The time at the start of the interval, in internal timestamp format
- The time when the DC/UCF system started, in internal timestamp format
- The job name of the DC/UCF system
- The Central Version (CV) number

Several statistics reports have been changed to handle the new statistics record layout and to account for the collection of statistics on an interval basis. The affected reports are: SRPT000, SRPT003, SRPT005, SRPT006, SRPT007, SRPT008, SRPT009, SRPT010, SRPT011, SRPT012, SRPT013, SRPT014, SRPT015, SRPT016 and SRPT021.

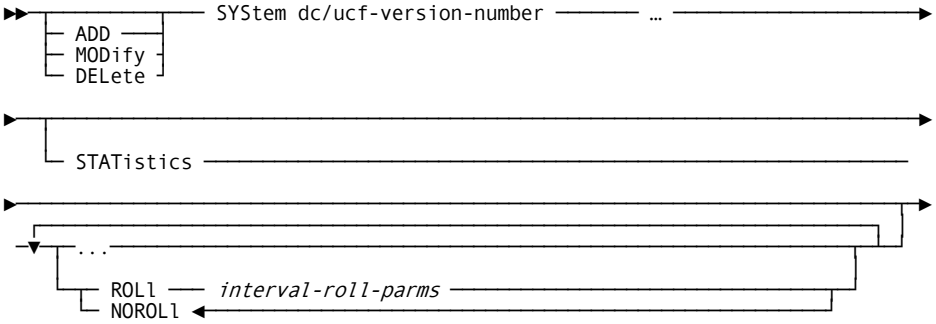
For more information about statistics and how they are managed, see the *System Operations Guide*.

SYSGEN SYSTEM Statement for Interval Roll

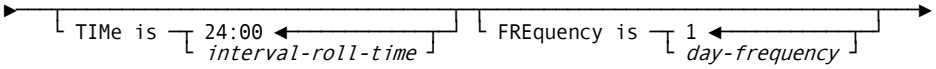
The SYSGEN SYSTEM statement is enhanced to let you specify the time of day and day frequency when statistics are to be written and reset.

SYSGEN SYSTEM Statement Syntax for Interval Roll

The following syntax shows updates to the SYSGEN SYSTEM statement.



Expansion of interval-roll-parms



SYSGEN SYSTEM Statement Parameter for Interval Roll

This section describes the new parameter for the SYSGEN SYSTEM statement:

ROLl

Directs the system to write system statistics and histograms to the log file and roll them out at specified time and day interval.

TIMe is *interval-roll-time*

Specifies the time of day in twenty-four hour format (HH:MM) at which statistics are to be written and reset.

Default: 24:00

FREquency is *day-frequency*

Specifies the day frequency at which system statistics and histograms are to be written and reset.

Range: 1–999

Default: 1

NOROLl

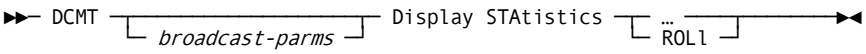
Directs the system not to perform statistics interval roll.

DCMT DISPLAY STATISTICS Command

The DCMT DISPLAY STATISTICS command has been enhanced to enable reporting on the current statistics interval roll time.

DCMT DISPLAY STATISTICS Syntax

The following syntax shows updates to the DCMT DISPLAY STATISTICS command.



DCMT DISPLAY STATISTICS Parameters

This section describes the new parameters for the DCMT DISPLAY STATISTICS command:

ROLl

Displays the time date stamp of the last performed interval roll and the issuer of the request – either RHDCSROL system task, or the user by DCMT WRITE STATISTICS ROLL command, Displays the time of day in twenty-four hour format (HH:MM) and day interval at which statistics are written to the log and reset.

Example: Displaying the interval roll time

The following example shows that the statistics are written to the log and reset at 02:40 45 a.m. each seventh day.

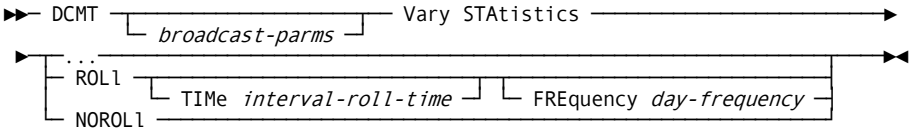
```
DCMT D STATISTICS ROLL
*** Display Statistics Interval Roll details ***

 Last Statistics Roll  2010-08-23-02.45.00.812033
     Last Issued By  SYSTEM TASK
  Interval Roll Time  02:45
     Day Frequency  7
```

DCMT VARY STATISTICS Command

The DCMT VARY STATISTICS command has been enhanced to let you vary the time of day and day interval when statistics are written and reset.

DCMT VARY STATISTICS Syntax

The following syntax shows updates to the DCMT VARY STATISTICS command.



DCMT VARY STATISTICS Parameters

This section describes the new parameters for the DCMT VARY STATISTICS command:

ROLl

Varies the interval at which statistics are written to the log file and reset.

TIMe *interval-roll-time*

Specifies time of day in twenty-four hour format (HH:MM) format at which statistics are written to the log and reset.

FREQuency *day-frequency*

Specifies the day frequency at which statistics are written to the log and reset.

NOROLl

Varies the system not to perform statistics interval roll.

Example: Changing Interval Roll Time and Day Frequency

The following example shows how the roll interval was changed to 8:30 p.m., each seventh day:

```
DCMT V STATISTICS ROLL TIME 20:30 FREQUENCY 7
STATISTICS INTERVAL ROLL WAS OFF
CHANGED TO 20:30 FREQUENCY IS 7  DAY(S)
```

The following example shows how the day frequency was changed to 10 days:

```
DCMT V STATISTICS ROLL FREQUENCY 10
STATISTICS INTERVAL ROLL TIME WAS 20:30 INTERVAL WAS 7  DAY(S)
CHANGED TO 20:30 FREQUENCY IS 10  DAY(S)
```

The following example shows how the roll interval was changed to 10:00 p.m., day frequency is unchanged:

```
DCMT V STATISTICS ROLL TIME 22:00
STATISTICS INTERVAL ROLL TIME WAS 20:30 INTERVAL WAS 10  DAY(S)
CHANGED TO 22:00 FREQUENCY IS 10  DAY(S)
```

The following example shows how to disable interval roll:

```
DCMT V STATISTICS NOROLL
STATISTICS INTERVAL ROLL TIME WAS 22:00 INTERVAL WAS 10  DAY(S)
CHANGED TO NOROLL
```

The following example shows how to enable interval roll with the last used values:

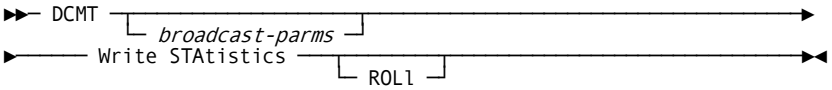
```
DCMT V STATISTICS ROLL
STATISTICS INTERVAL ROLL WAS OFF
CHANGED TO 22:00 FREQUENCY IS 10  DAY(S)
```

DCMT WRITE STATISTICS Command

The DCMT WRITE STATISTICS command provides a new option to clear statistics after they are written.

DCMT WRITE STATISTICS Syntax

The following syntax shows updates to the DCMT WRITE STATISTICS command.



DCMT WRITE STATISTICS Parameter

This section describes the new parameters for the DCMT WRITE STATISTICS command:

ROLl

Writes the current system-wide statistics and histograms to the DC/UCF log file and resets their values.

Example: Writing to the DC/UCF log files

This example shows the message displayed after the statistics are written to the log and reset.

```
DCMT W STATISTICS ROLL
IDMS DC275916 V74 STATISTICS WRITTEN TO THE LOG AND ROLLED OUT BY USER
```

Chapter 5: SQL

This chapter describes the SQL related enhancements for CA IDMS Version 18.0. These new features make it easier and faster to change the structure of an existing database and provide new options for defining SQL databases.

This section contains the following topics:

Alter and Drop Column Support

With this release, you can alter and drop columns in SQL tables. This ability lets you change the structure of a table without unloading and reloading data.

You can now make the following types of column changes using the ALTER TABLE statement:

- Change a column's data type or null attribute
- Drop or change a column's default clause
- Rename a column
- Drop a column

In addition, the syntax for adding columns lets you include an optional COLUMN keyword following the ADD keyword. Inclusion of the COLUMN keyword makes the syntax consistent with the SQL standard and with other column-related clauses.

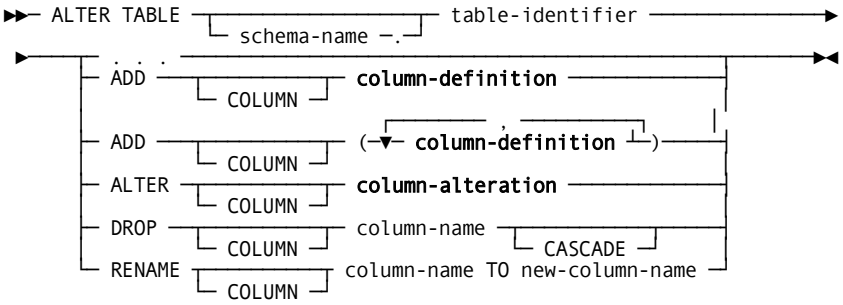
ALTER TABLE Statement Changes for Columns

New parameters were added to the ALTER TABLE statement to let you change attributes and drop columns.

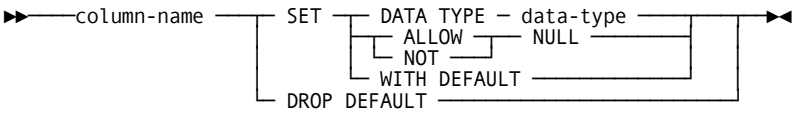
For more information about the full ALTER TABLE statement, see the *SQL Reference Guide*.

Syntax

The following diagram shows the enhanced ALTER TABLE syntax:



Expansion of column-alteration



Parameters

This section describes the parameters that support the ALTER TABLE statement:

ALTER COLUMN column-alteration

Specifies the changes to be made to the attributes of a column.

Expanded syntax for column-alteration is shown after the ALTER TABLE syntax. Descriptions of column-alteration parameters follow the description of ALTER TABLE parameters.

DROP COLUMN *column-name*

Identifies the column to be removed from the table. Column-name must be the name of a column in the table.

Note: You cannot drop columns that are part of a CALC key of a populated table or that are named in a check constraint.

CASCADE

Drops the following entities:

- The CALC key if it includes the column.
- All referential constraints in which the named column is a referenced or foreign key column.
- All linked constraints in which the named column is a sort column.
- All indexes in which the named column is an indexed column.
- All views in which the column is named.

If CASCADE is not specified, the column must not participate in a referential constraint or index, or be named in a view.

RENAME COLUMN *column-name*

Identifies the column name to be changed. Column-name must be the name of a column in the table.

Note: You cannot rename a column if the column is named in a check constraint or in a view.

TO *new-column-name*

Specifies the new name for the identified column.

Limits: new-column-name must be a 1-32 character value that follows the conventions for SQL identifiers.

Note: It must be distinct from the name of any existing column in the table.

Parameters for Expansion of column-alteration

column-name

Identifies the column whose attributes are to be changed. Column-name must be the name of a column in the table.

DATA TYPE data-type

Defines the new data type for the named column. The specified data type must be compatible for assignment with the column's existing data type.

Note: For expanded data-type syntax, see the section "Expansion of Data-type" in the *SQL Reference Guide*.

You cannot change the data type of a column that is part of a CALC key of a populated table or that is a referenced or foreign key column in a constraint.

ALLOW NULL

Indicates that the column can contain null values.

You cannot change the null attribute of a column that is part of a CALC key of a populated table or a referenced key.

NOT NULL

Indicates that the column cannot contain null values.

You cannot change the null attribute of a column that is part of a CALC key of a populated table or a referenced key.

WITH DEFAULT

Sets the column's value to a default if no value for the column is specified when a row is inserted.

DROP DEFAULT

Does not set the column's value to a default when a row is inserted.

The following parameter describes the Default Index enhancement to the ALTER TABLE statement

ADD DEFAULT INDEX

Creates a default index for the named table.

The table must not already have a default index associated with it.

ALTER TABLE Usage for Columns

The following considerations apply when using the Alter and Drop Column parameters to the ALTER TABLE statement:

Add a Default to a Column

Allowing a column to have a default value affects only the table's definition; existing table rows are not affected.

Remove a Column's Default

If the table is populated and the column does not allow null values, every existing row must contain a value in the changed column. To ensure this, each row is *accessed and updated* if it does not contain a value for the column.

Rename a Column

A column that is named in a check constraint or a view cannot be renamed.

The definition of all referential constraints, sort keys, CALC keys and indexes in which the column participates *are updated* to show the new column name.

Drop a Column

Every row in the table is updated to remove the column value.

If a column is named in a check constraint or is part of the CALC key of a populated table, you cannot drop the column.

If you do not specify CASCADE, the column must not be one of the following types of columns:

- A column in a CALC key
- A referenced or foreign key column in a referential constraint
- An indexed column
- A sort column of a linked constraint
- Named in a view

If you specify CASCADE, how the column is used determines what other items are dropped:

- Dropping a CALC key column also drops the CALC key
- Dropping a referenced or foreign key column in a referential constraint also drops the constraint
- Dropping an indexed column also drops the index
- Dropping a sort column of a linked constraint also drops the constraint
- Dropping a column named in a view also drops the view

Change a Column's Null Attribute

The following situations apply when you change a column's null attribute:

- When the column is part of a CALC key of a populated table, or is a referenced column in a constraint, the ALTER statement fails.
- When you change a null attribute, every row in the table is updated to add or remove the null attribute byte for that column.
- When the changed column is a sort column, every index and linked indexed constraint is automatically rebuilt.
- When disallowing nulls and the value of the column is null for a row in the table, the ALTER statement fails.

Change a Column's Data Type

The following situations apply when you change a column's data type:

- When the column is part of a CALC key of a populated table, or is a referenced column in a constraint, the ALTER statement fails.
- When changing a column's data type, the new data type you enter must be compatible for assignment with the original data type.
- Every row in the table is restructured to convert the column value to the new type. This might involve increasing or decreasing the length of the row.
- The ALTER statement will fail if a loss of data (such as truncation of a non-blank character or numeric overflow) would occur as part of the conversion.
- When you change data type, every index and linked indexed constraint in which the column is a sort column is rebuilt.

Example: Alter Table Statement Parameters

The following examples illustrate the Alter and Drop Column Support enhancement to the ALTER TABLE statement parameters:

Add a new column to an existing table:

```
alter table demo.empl
    add column phone character(10);
```

Change a column's data type:

```
alter table demo.empl
    alter column city
    set data type varchar(20);
```

Drop a column using the CASCADE option:

```
alter table demo.empl
    drop column status cascade;
```

Rename a column:

```
alter table demo.empl
    rename column proj_id to project_id;
```

Alter Index Extensions

CA IDMS Version 18.0 supports changing additional index characteristics. This support lets you change the structure and location of an index without dropping and recreating it.

Additions to the ALTER INDEX statement let you change the following attributes:

- Displacement
- Uniqueness
- Area association

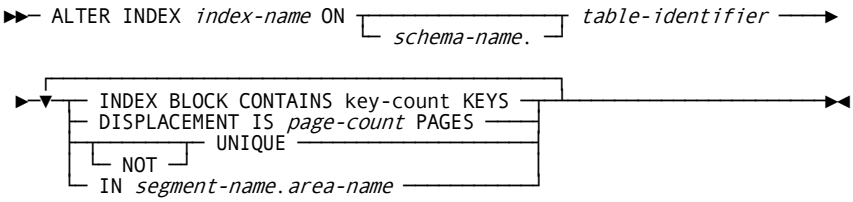
ALTER INDEX Statement

The ALTER INDEX was enhanced to support additional types of changes to an index.

For more information on the full ALTER INDEX statements, see the *SQL Reference Guide*.

ALTER INDEX Syntax

The following diagram shows the syntax for the new parameters for the ALTER INDEX statement:



ALTER INDEX Parameters

This section describes the parameters for the ALTER INDEX statement:

***page-count* PAGES**

Specifies how far away from the index owner the bottom-level index records are stored.

If the value of page-count is zero (0), the bottom-level internal index records are not displaced from the index owner.

Limit: An unsigned integer from 0–32,767.

UNIQUE

Specifies that the index-key value in any given row of the table on which the index is defined must be different from the index-key value in all other rows of the table. The table cannot contain any duplicate index-key values.

If you specify UNIQUE and the table contains duplicate index-key values, the alter statement will fail.

NOT UNIQUE

Removes the restriction that all values of the index-key within the table must be unique.

When the UNIQUE restriction is used to ensure uniqueness of a referenced key in some constraint, you cannot remove it from an index unless another index or CALC key can be used in its place.

IN

Requests a change in the location of the named index.

area-name

Identifies a new area with which the index is to be associated. Area-name must identify an area defined in the dictionary.

segment-name

Identifies the segment associated with the area.

Example: Alter the EMP_LNAME Index

In this example, the EMP_LNAME index is moved from its current location to the DEMO.EMPAREA area. Each internal index record will have a maximum of 30 keys and the bottom-level index records will be displaced 40 pages from the top of the index.

```
alter index emp_lname (last_name) on emp.benefits
    displacement is 40 pages
        index block contains 30 keys
    in area demo.emparea;
```

Alter Constraint Support

You can now alter certain characteristics of a linked index referential constraint. This ability lets you make changes to the constraint without dropping and recreating it and, potentially, without recreating the constraint's associated tables.

The new ALTER CONSTRAINT statement lets you change the following constraint attributes:

- Index displacement
- Uniqueness

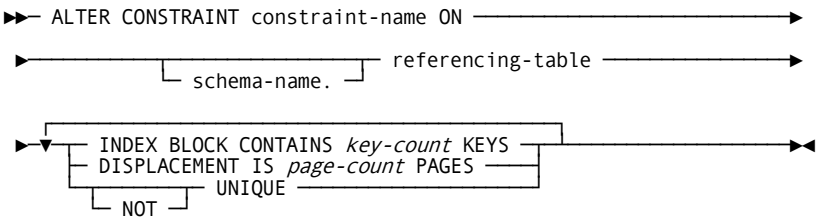
ALTER CONSTRAINT Statement

The ALTER CONSTRAINT statement changes the characteristics of an existing referential constraint. This statement is a CA IDMS extension to the SQL standard.

Note: To issue an ALTER CONSTRAINT statement you must own or hold the ALTER privilege on the table on which the constraint is defined.

ALTER CONSTRAINT Syntax

The following diagram shows the new ALTER CONSTRAINT statement:



ALTER CONSTRAINT Parameters

This section describes the ALTER CONSTRAINT parameters:

constraint-name

Identifies the referential constraint to be changed. Constraint-name must be the name of a constraint on the table identified in the ON clause.

referencing-table

Specifies the name of the referencing table in the constraint to be changed.

schema-name

Identifies the schema associated with the referencing table.

Default: The default varies depending on where the statement is encountered.

If you do not specify schema-name, it defaults to:

- The current schema associated with your SQL session when the statement is entered through the Command Facility or executed dynamically.
- The schema associated with the access module used at runtime when the statement is embedded in an application program.

key-count **KEYs**

Establishes a new value for the maximum number of entries in each internal index record (SR8 system record).

Limits: Key-count must be an unsigned integer in the range 3 through 8180.

page-count **PAGES**

Specifies how far away from the referenced row the bottom-level index records are stored.

If the value of page-count is zero (0), the bottom-level internal index records are not displaced from the referenced row.

Limits: Page-count must be an unsigned integer in the range 0 through 32,767.

UNIQUE

Specifies that the sort-key value in any given row of the referencing table must be different from the sort-key value in all other rows that have the same non-null referencing key value.

NOT UNIQUE

Removes the restriction that all values of the sort-key with the same non-null foreign key value must be unique.

Example: Alter the DEPT_EMPL Constraint

In this example, the physical characteristics of the DEPT_EMPL constraint are changed. Each internal index record will have a maximum of 10 keys and the bottom level index records will be displaced 50 pages from the associated referenced row:

```
alter constraint dept_empl on emp.empl
    displacement is 50 pages
    index block contains 10 keys;
```

Adding a Default Index

A new option in the ALTER TABLE statement enables you to add a default index to an existing table. Adding a default index can improve performance when accessing the table or making additional definitional changes.

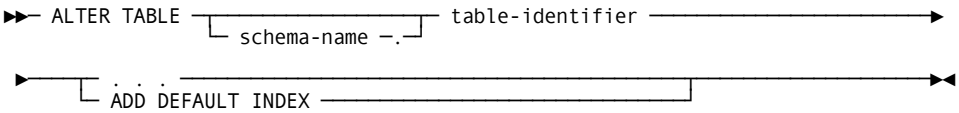
ALTER TABLE Statement for Adding a Default Index

The ADD DEFAULT INDEX parameter has been added to the ALTER TABLE statement to enable adding a default index.

For more information about default indexes, see the section CREATE TABLE in the SQL Reference Guide.

ALTER TABLE Syntax for Adding a Default Index

The following diagram shows the syntax placement of the new parameter for the Default Index enhancement to the ALTER TABLE statement:



ALTER TABLE Parameters for Adding a Default Index

The following parameter describes the Default Index enhancement to the ALTER TABLE statement

ADD DEFAULT INDEX

Creates a default index for the named table.

The table must not already have a default index associated with it.

Example: Add a Default Index

The following statement adds a default index to the EMP.DEPT table:

```
alter table emp.dept
    add default index;
```

Additional Columns in Foreign Key Indexes

With previous versions of CA IDMS, an index used to enforce the integrity of an unlinked referential constraint contains only columns that make up the foreign key. With Version 18.0, you can add additional columns to your foreign key indexes. This can potentially reduce disk space requirements and improve your overall performance.

Extending foreign key indexes with additional columns allows you to use one index for multiple purposes. For example, the primary key in a table is often a concatenation of one of its foreign keys with additional columns that together form a unique identifier for each table row. You can now use a single index to enforce both the integrity of the referential constraint and the uniqueness of the primary key. By eliminating a second index, you reduce disk space requirements and the overhead associated with index maintenance.

Including extra columns in a foreign key index can also potentially improve access efficiency by enabling the use of more index scans to identify rows that match your selection criteria. Using an index scan can significantly reduce the number of I/Os necessary to satisfy a query.

To take advantage of this feature, define an index so that the foreign key columns precede any additional columns in your index key. As in earlier releases, the order of the foreign key columns in the index key must match the order of the referenced columns in some unique index or CALC key on the referenced table.

Large Key Support

For CA IDMS running in a z/OS or z/VSE environment, the maximum key length in an SQL database has been increased from 256 to 2000 bytes. This enhancement allows CA IDMS to support a richer set of applications.

You can now define CALC, index, sort, foreign, and referenced keys that are up to 2000 bytes in length. To use this feature, define a key with a length greater than 256 bytes.

When using large keys, consider the following restrictions:

- You must use the REORG utility instead of the UNLOAD and RELOAD utilities to reorganize a database that contains keys longer than 256 bytes.
- You cannot use the FOR CALC option of the PRINT PAGE utility for locating the target page of a large key value, because you can only specify key values that are less than or equal to 256 bytes

Improved DDL Performance

CA IDMS now considers statistics when processing SQL DDL statements. Considering statistics may result in fewer I/Os and faster execution when executing the following DDL statements that access all rows of a table:

- CREATE INDEX, ALTER INDEX
- CREATE TABLE, ALTER TABLE
- ALTER CONSTRAINT, DROP CONSTRAINT

When CA IDMS must access all table rows during DDL statement execution, it uses either a DBKEY or clustering index if one is available. However, if no such index exists, CA IDMS must decide whether to access rows through some other index or through an area sweep. CA IDMS now chooses between these alternatives based on table statistics. If there are fewer rows than half the number of pages in the area, the table is accessed through the index; otherwise, it is accessed through an area sweep.

Chapter 6: Serviceability Aids

This section contains the following topics:

Enhanced System Tracing

Enhanced tracing provides improved diagnostic tools for faster problem resolution by technical support. The following table lists the enhanced capabilities:

- Optional merging of system and extended (formerly called DBTRACE) trace information for easier determination of the events leading up to a problem.
- The ability to save trace data in a persistent data store (either a new trace area or the log area) so that more information is available for problem diagnosis.
- The ability to report on saved trace information and chronologically merge information from multiple data sharing members.
- Syntactic control over the inclusion of trace information in task snaps, which eliminates the need for optional APAR flag 246.

Trace Tables and Entry Types

Starting with Version 18.0 you can allocate two types of system-wide trace tables: a system trace table and an adjunct trace table. The two types of tables are used as follows:

- The system trace table records the standard system trace (SYSTRACE) entries. As in prior releases, standard system tracing is enabled using either the SYSGEN SYSTEM statement SYSTRACE parameter or the DCMT VARY SYSTRACE command. If system tracing is enabled dynamically and no system trace table exists, one is allocated automatically.
- Extended trace entries are variable in length and are generated only if certain CSA flags are turned on using either SYSIDMS parameters or the DCMT VARY CSAFLAGS command. Extended trace entries are recorded in the adjunct trace table if one exists; otherwise, they are recorded in the system trace table. If neither table exists, extended trace entries are not generated even if the CSA flags are on.

Saving Trace Data

Saving system trace data increases the information available for problem diagnosis. When saving trace data, CA IDMS periodically writes trace entries to a persistent data store: either a trace area or a log area.

A trace area is a new type of system area that can be defined in Version 18.0. It is similar to a log area and, to be usable, it must either be included in the SYSTEM segment or be accessible through the SYSTEM DBNAME. It differs from a log area in that its name is DDLDCTRC instead of DDLDCLOG. The new trace area is archived and printed using the new ARCHIVE TRACE and PRINT TRACE utility statements. Just as for a log area, a job to archive the trace area can be automatically submitted using the WTO exit.

When you enable saving trace data, consider the following:

- If a trace area is defined in the run time DMCL, trace entries are written to the trace area rather than the log area. Trace information is never written to a sequential log file.
- If large volumes of trace data are to be saved, you should define a trace area to minimize the impact on system throughput.
- If an adjunct table exists, only its contents are saved.
- If a system trace table exists without an adjunct table, the contents of the system trace table are saved. This can generate high volumes of trace output.

Specify Trace Options

Tracing options include the sizes of the system and adjunct trace tables, and whether trace information is saved or not. You can specify these options using SYSGEN syntax, DCMT commands, and SYSIDMS parameters.

Note: Options specified through SYSIDMS parameters generally override those specified on the SYSGEN statement. Options specified through DCMT commands dynamically change the attributes currently in effect for a CV.

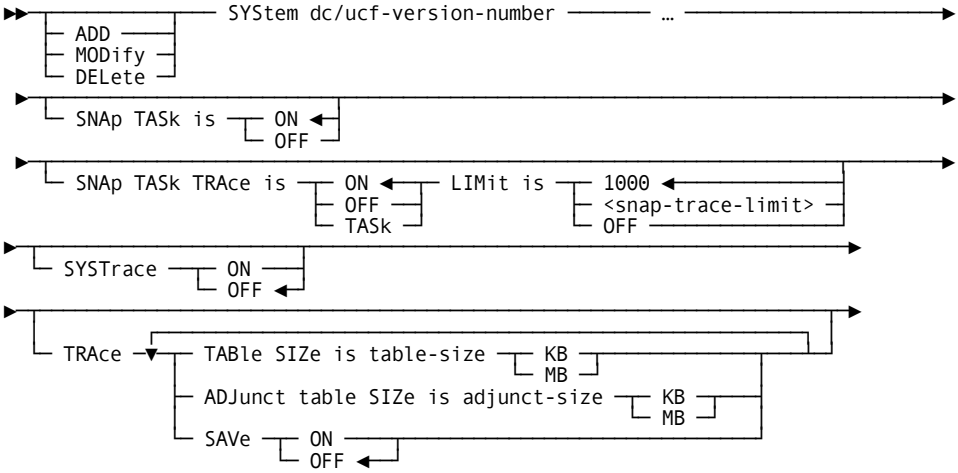
SYSGEN SYSTEM Statement for Trace Options

To support trace options, the SYSGEN SYSTEM statement has been enhanced in the following ways:

- The new TRACE and TRACE LIMIT options on the SNAP TASK parameter control the inclusion of trace information in task snaps. The TRACE option replaces numbered option 246. The TRACE LIMIT option replaces the SYSTRACE ENTRIES parameter. For upward compatibility when migrating from a prior release, the value in effect for the SYSTRACE ENTRIES parameter becomes the trace limit value.
- The ENTRIES sub-parameter of the SYSTRACE parameter is no longer supported. If specified, a warning is issued.
- A new TRACE parameter lets you specify tracing options.

SYSGEN SYSTEM Syntax for Trace Options

The following diagram shows the syntax updates to the SYSGEN SYSTEM statement:



SYSGEN SYSTEM Parameters for Trace Options

This section describes the new parameters for the SYSGEN SYSTEM statement:

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

OFF

Disables the writing of a task snap dump.

Default: ON

TRAce is ON|OFF|TASk

Controls the inclusion of trace information in task snaps.

ON

Includes formatted trace information for all tasks in a task snap.

OFF

Includes no trace information in a task snap.

TASk

Includes only trace information for the task for which the snap is being issued.

Default: ON

LIMit is snap-trace-limit|OFF

Limits the number of trace entries reported in a task snap.

snap-trace-limit

Specifies the maximum number of trace entries that are reported in a task snap.

Limit: 0–32767

Default: 1000

Note: A value of 0 is the same as specifying OFF.

OFF

Indicates that there is no limit to the number of trace entries included in a task snap.

SYSTrace ON|OFF

Enables or disables basic system tracing.

ON

Enables basic system tracing.

OFF

Disables basic system tracing.

Default: OFF

TRAce

Specifies system trace options.

TABle SIZe table-size KB|MB

Specifies the size of the system trace table in kilobytes (KB) or megabytes (MB).

Limit: 0–9999

Default: 4 MB

Note: Two copies of the system trace table are allocated on all z/OS systems.

ADJunct table SIZe adjunct-size KB|MB

Specifies the size of the adjunct trace table in kilobytes (KB) or megabytes (MB).

Limit: 0–9999

Default: 0

SAVe ON|OFF

Controls whether trace information is saved for future reporting.

ON

Enables saving of trace information.

- Adjunct trace table entries are saved if an adjunct trace table has been allocated. If no adjunct trace table has been allocated system trace table entries are saved.
- Trace entries are written to the trace area if one is defined in the run time DMCL, or they are written to the log area if one is defined. If neither area is defined no trace information is saved.

OFF

Disables saving of trace information.

Default: OFF

DCMT Changes for Trace Options

To support trace options, some of the DCMT commands have been enhanced in the following ways:

- The new DCMT VARY TRACE command alters the tracing options.
- The new DCMT DISPLAY TRACE command displays the current trace options in effect for a CV.
- The new TRACE and TRACE LIMIT parameters on the DCMT VARY SNAP command control the inclusion of trace information in task snaps.
- The ENTRIES parameter of the DCMT VARY SYSTRACE command is no longer supported.
- The DCMT DISPLAY LOG command has been enhanced to indicate when log statistics were last reset.

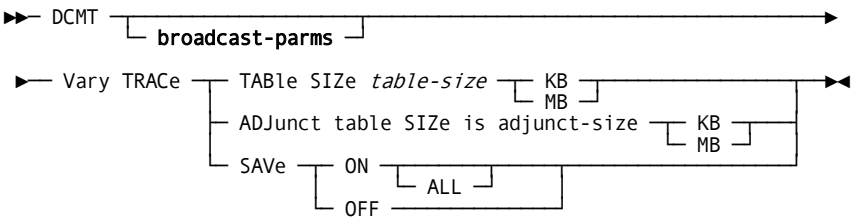
Note: The DCMT DISPLAY DBTRACE and DCMT VARY DBTRACE commands are no longer supported. Those commands have been replaced by the DCMT DISPLAY TRACE and DCMT VARY TRACE commands, respectively.

DCMT VARY TRACE Command

The DCMT VARY TRACE command alters the tracing options currently in effect for your system.

DCMT VARY TRACE Syntax

The following diagram shows the syntax for the DCMT VARY TRACE command:



DCMT VARY TRACE Parameters

This section describes the parameters for the DCMT VARY TRACE command:

TABle SIZe *table-size* KB|MB

Specifies the size of the system trace table in kilobytes (KB) or megabytes (MB).

Limits: 0–9999.

ADJunct table SIZe *adjunct-size* KB|MB

Specifies the size of the adjunct trace table in kilobytes (KB) or megabytes (MB).

Limits: 0–9999.

SAVe

Controls whether trace information is saved for future reporting.

ON

Saves trace information.

Note: If an adjunct trace table has been allocated, only its contents are saved; otherwise, the contents of the system trace table are saved.

Trace information is written to the trace area if one is defined in the runtime DMCL; otherwise, it is written to the log area if one is defined. If the DMCL contains neither area, no trace information is saved.

ALL

Saves both the current and future contents of the trace table. If ALL is not specified, only future entries are saved.

OFF

Specifies that trace information is not saved for future reporting.

Example: Changing the Size of the Trace Table

The following example changes the size of the system trace table to 1 MB:

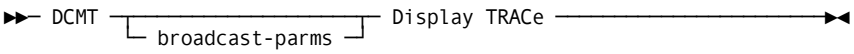
```
DCMT V TRACE TABLE SIZE 1 MB
System tracing (SYSTRACE):   ON
     Trace table size:    1 MB     Address: 36605000
   Adjunct table size:    0 KB     Address: 00000000


Save: OFF       Driver: INACTIVE       Area: DDLDCTRC
```

DCMT DISPLAY TRACE Command

The DCMT DISPLAY TRACE command displays the tracing options currently in effect for your system.

DCMT DISPLAY TRACE Syntax

The following diagram shows the syntax for the DCMT DISPLAY TRACE command:



DCMT DISPLAY TRACE Parameters

This section describes the parameters for the DCMT DISPLAY TRACE statement:

broadcast-parms

Executes the DCMT command on all or a list of data sharing group members.

For more information about broadcasting and broadcast-parms syntax, see "How to Broadcast System Tasks" in the *System Tasks and Operator Commands Guide.*

Example: DCMT DISPLAY TRACE outputs

The following example illustrates the output from a DCMT DISPLAY TRACE command when trace information is not being saved.

```
DCMT DISPLAY TRACE
System tracing (SYSTRACE):    ON
                Trace table size:    20 MB       Address: 39A40000
              Adjunct table size:    10 MB       Address: 36603000
Save: OFF         Driver: INACTIVE         Area: DDLDCLOG
```

The following example illustrates the output from a DCMT DISPLAY TRACE command when trace information is being saved to a DDLDCTRC area.

```
DCMT DISPLAY TRACE
System tracing (SYSTRACE):    ON
      Trace table size:     4 KB       Address: 39B65000
    Adjunct table size:     8 MB (S)   Address: 36603000

 Save: ON          Driver: ACTIVE          Area: DDLDCTRC     0% FULL
 ------------Trace service driver statistics-------------
Driver started.....2009-12-08-12.23.21.151167
Number of save requests.....44
Number of times entries missed.....2
Bytes/hour.....1067733
Pages/hour.....300
Number of reads.....14
Number of writes.....9
Number of read waits.....1
Number of write waits.....0
Number of page range resets.....1
Number of area full waits.....0
Number of errors.....0
% of waits to I/Os.....0
Number of RUs.....8
Number of look aheads.....5
% of look aheads to RUs.....63
```

System Tracing

Potential values are as follows:

- ON—System tracing is enabled.
- OFF—System tracing is disabled.

Trace table size

The size of the system trace table in kilobytes (KB) or megabytes (MB).

If the characters “(S)” follow table size, it indicates that the contents of the system trace table are being saved.

Address

The address of the system trace table.

Adjunct table size

The size of the adjunct trace table in kilobytes (KB) or megabytes (MB).

If the characters “(S)” follow table size, it indicates that the contents of the adjunct trace table are being saved.

Address

The address of the adjunct trace table.

Save

Potential values are as follows:

- ON—Trace saving is enabled.
- OFF—Trace saving is disabled.
- REQUESTED—Trace saving has been requested but is not yet fully enabled.

Driver

Potential values are as follows:

- ACTIVE—Trace service driver is active.
- INACTIVE—Trace service driver is inactive.
- PENDING—Trace service driver is starting up.

Area

Potential values are as follows:

- DDLDCLOG—Trace information is written to the log area.
- DDLDCTRC—Trace information is written to the trace area.

% Full

The percentage of space used in the area.

Trace service driver statistics

A header for statistics that are displayed only if trace saving is enabled.

Driver started

The date and time at which the trace service driver was started.

Statistics reset

The date and time when the driver statistics were reset due to overflow.

Number of save requests

The number of requests made to save trace information.

Number of times entries missed

The number of times one or more trace entries were not saved because they had been overlaid before they could be written.

Bytes/hour

The rate at which trace information is being written, specified as bytes per hour.

Pages/hour

The rate at which pages are written to the log or trace area, specified as pages per hour.

Number of reads

The number of pages read from the log or trace area.

Number of writes

The number of pages written to the log or trace area.

Number of read waits

The number of times the driver had to wait for a read to complete.

Number of write waits

The number of times the driver had to wait for a write to complete.

Number of page range resets

The number of times the driver had to recalculate the range of pages into which it can write information.

Number of area full waits

The number of times the driver had to wait for the contents of the log or trace area to be archived.

Number of errors

The number of I/O errors encountered.

% of waits to I/Os

The percent of waits to I/O requests.

Number of RUs

The number of run units currently in use.

Number of look aheads

The number of look ahead reads in effect.

% of look aheads to RUs

The percent of run units being used for look ahead reads.

How to Reduce the Number of Missed Entries

Eliminating missed trace entries can be difficult; however, there are steps you can take to reduce the number of missed entries. In the trace information output, if the value for number of times entries missed is large compared to the value for number of save requests, consider taking one or more of the following actions:

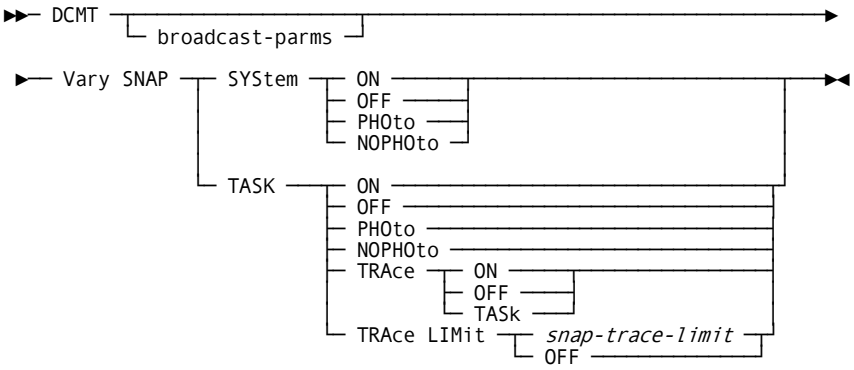
- Save trace information to the trace area rather than the log area.
- Reduce the amount of trace information being saved. If only extended trace information is of interest, be sure to allocate an adjunct table so only extended trace information is saved.
- Increase the size of the trace or adjunct table.
- Ensure that the appropriate archive utility is executed often enough that the trace area does not fill. The easiest way to do this is to automate the submission of the archive job using a WTO exit.

DCMT VARY SNAP Command Enhancements

The new TRACE and TRACE LIMIT parameters on the DCMT VARY SNAP command control the inclusion of trace information in task snaps.

DCMT VARY SNAP Syntax

The following syntax shows the enhancements to the DCMT VARY SNAP command:



DCMT VARY SNAP Parameters

This section describes the new parameters for the DCMT VARY SNAP command:

broadcast-parms

Executes the DCMT command on all or a list of data sharing group members.

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

SYStem

Applies the VARY SNAP command to system snaps.

ON

Enables the writing of system snap dumps to the DC/UCF log file.

OFF

Disables the writing of system snap dumps to the DC/UCF log file.

PHOto

Enables the writing of system snap photos to the DC/UCF log file.

NOPHOto

Disables the writing of system snap photos to the DC/UCF log file.

TASK

Applies the VARY SNAP command to task snaps.

ON

Enables the writing of task snap dumps to the DC/UCF log file.

OFF

Disables the writing of task snap dumps to the DC/UCF log file.

PHOto

Enables the writing of task snap photos to the DC/UCF log file.

NOPHOto

Disables the writing of task snap photos to the DC/UCF log file.

TRAce

Controls the inclusion of system trace information in task snaps.

ON

Includes system trace information for all tasks in a task snap.

OFF

Includes no system trace information in a task snap.

TASk

Includes only system trace information for the task for which the snap is being issued.

TRAce LIMit

Limits the number of trace entries reported in a task snap.

snap-trace-limit

Specifies the maximum number of trace entries that are reported in a task snap.

Limit: 0–32767

Note: A value of 0 (zero) is the same as specifying OFF.

OFF

Includes an unlimited number of trace entries in a task snap.

Note:

- For more information about displaying current snap attributes, see the section DCMT DISPLAY SNAP.
- For more information about dynamically controlling snaps at the program or task level, see DCMT VARY PROGRAM and DCMT VARY TASK.
- For more information about snap dumps and snap photos, see the *CA IDMS Navigational DML Programming Guide*.

  For more information about setting snaps at the system level, see documentation of the SYSTEM statement in the *CA IDMS System Generation Guide*.

Example: Change the Trace Limit for Task Snaps

The following example shows how to change the trace limit for task snaps to 500.

```
DCMT V SNAP TASK TRACE LIMIT 500
IDMS DC278011 V73 USER:***  TASK SNAP TRACE LIMIT SET
```

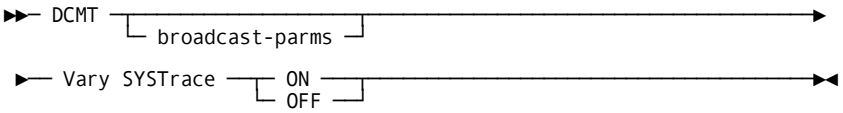
DCMT VARY SYSTRACE Command Change

In Version 18.0, the DCMT VARY SYSTRACE command no longer supports an ENTRIES parameter. Use the DCMT VARY SNAP TASK TRACE LIMITS parameter to limit the number of trace entries in a task snap and use the DCMT VARY TRACE TABLE SIZE parameter to specify the size of the system trace table.

Note: A warning is issued if you specify the ENTRIES parameter.

DCMT VARY SYSTRACE Syntax

The following diagram shows the syntax change to the DCMT VARY SYSTRACE command:



DCMT VARY SYSTRACE Parameters

This section describes the new parameters for the DCMT VARY SYSTRACE command:

broadcast-parms

Executes the DCMT command on all or a list of data sharing group members.

For more information on broadcasting and broadcast-parms syntax, see the section "How to Broadcast System Tasks" in the *System Tasks and Operator Commands Guide*.

ON

Enables standard system tracing.

OFF

Disables standard system tracing.

More Information

- For more information about displaying the status and size of the SYSTRACE table, see the section DCMT DISPLAY SYSTRACE.

  For more information about defining the system trace table, see the *System Generation Guide*.

DCMT DISPLAY LOG Output Changes

The output of the DCMT DISPLAY LOG command has been enhanced to indicate when log statistics were last reset if they needed to be reset to avoid an overflow of the read count.

SYSIDMS Parameters for Trace Options

New SYSIDMS parameters have been added to support tracing enhancements:

- The new TRACE_TABLE_SIZE parameter lets you specify the size of the system trace table.
- The new SYSTRACE parameter controls whether basic system tracing is enabled or not.
- The new ADJUNCT_TRACE_TABLE parameter replaces the existing DB_TRACE_TABLE parameter.

New SYSIDMS parameters have been added to support tracing enhancements:

TRACE_TABLE_SIZE=*table-size* KB | MB

Specifies the size of the system trace table in kilobytes (KB) or megabytes (MB).

Limit: 0–9999

Default: 0

Note: If basic system tracing is enabled by the SYSIDMS SYSTRACE parameter and the table size is 0, the table size is changed to 4 MB.

Any non-zero table size established by a SYSIDMS parameter overrides the trace table size specified in the system definition.

SYSTRACE=ON|OFF

Controls whether basic system tracing is enabled.

ON

Enables basic system tracing.

Note: If basic system tracing is enabled by the SYSIDMS SYSTRACE parameter, it remains enabled for a system even if its system definition indicates that SYSTRACE is OFF.

OFF

Does not enable basic system tracing.

ADJUNCT_TRACE_TABLE=*table-size* KB|MB

Specifies the size of the adjunct trace table in kilobytes (KB) or megabytes (MB).

Limit: 0–9999

Default: 0

Note: Any non-zero adjunct table size established by a SYSIDMS parameter overrides the adjunct trace table size specified in the system definition.

Printing and Archiving Trace Information

Starting with Version 18.0, the following enhancements facilitate the printing and archiving of trace information:

- Trace entries in snap dumps now include additional information.
- New and modified utilities let you the print and archive saved trace information.
- The sample WTOEXIT program supports automatic archiving of trace information.
- The supplied WTOEXIT checks for message DC050004 (LOG FULL) in addition to DC050001 (% free halved).

Format of Trace Entries in Snap Dumps

The format of trace entries in snap dumps has been enhanced to show additional information for system trace entries and to display extended trace entries, if they exist in the system trace table.

Entries now appear in newest to oldest order rather than oldest to newest as in prior releases. Each entry now includes the UTC time and date it was generated, the address of the primary LTE associated with the task, and the TOD clock value at the time it was generated.

```
012610 10.09.37 TRACE ENTRIES, ORDERED NEWEST TO OLDEST.
                TASKID/
 TOD TIME/DATE SCA  A(LTE)  MOD  MAC  CALL   R11/R2  R12/R3  R13/R4  R14/R5  R15/R6   R0/R7   R1/R8      TOD CLOCK
15.10.01.809756 00      12   9  240   1 000000C8 3836B0B8 36FAF750 B836B0CC 3836B0B8 371675B8 36FAF678 C5722F6CF3F5C00B
   2010/01/26   00267D30 SNAP #GETSTK    00085009 00000000 383A6FF2 00085019 383C86B8 383A6EAC 383A68B8
15.10.01.809755 00      12  11  29  13 36FAF678 383A58B8 36FAF724 B83A6C8C 3836B0B8 371675B8 36FAF678 C5722F6CF3F5BC8B
   2010/01/26   00267D30 WTL  SNAPEP1    00085009 00000000 383A6FF2 00085019 383C86B8 383A6EAC 383A68B8
15.10.01.809754 00      12  10  27  32 36FAF764 38368404 36FAF798 B8368690 38375CB8 002674C8 36FAF764 C5722F6CF3F5A80B
   2010/01/26   00267D30 MISC RMGREP1    36FAF764 37166988 B8376B28 37318CDC 00000000 37176780 B83685F4
15.10.01.809754 00      12  10  12  34 36FAF764 38368404 36FAF798 B8368708 38376888 383684A6 36FAF764 C5722F6CF3F5A58B
   2010/01/26   00267D30 MISC RMGREP2    37167108 37318F3C 00000001 37318CDC 00027734 37176780 B83685F0
15.10.01.809754 00      12  10 240  29 00000000 38368404 36FAF768 B8368414 38368404 00000002 38366F3C C5722F6CF3F5A38B
   2010/01/26   00267D30 MISC #GETSTK    37318B18 37318F3C 00000001 37318CDC 00000000 B836611A 37B49708
15.10.01.809754 00      12  10  62   5 00000000 B83660A6 36FAF754 B836613E 38368404 00000002 38366F3C C5722F6CF3F5A08B
   2010/01/26   00267D30 MISC SNGLEP1    37318B18 37318F3C 00000001 37318CDC 00000000 B836611A 37B49708
15.10.01.809752 00      12  14 240   1 0000003C 383A3CB8 36FAF780 B83A3CCC 383A3CB8 00000008 00000040 C5722F6CF3F5808B
   2010/01/26   00267D30 TIMP #GETSTK    37318B18 37318F3C 00000001 37318CDC 00000000 00000006 37B49708
15.10.01.809751 00      12  10   9   6 00000000 B83660A6 36FAF754 B8366620 383A3CB8 00000008 00000040 C5722F6CF3F57C8B
   2010/01/26   00267D30 MISC TIMPGET    37318B18 37318F3C 00000001 37318CDC 00000000 00000006 37B49708
15.10.01.809748 00      12  10  27  31 36FAF77C 38368404 36FAF7B0 B8368650 38375CB8 002674C8 36FAF77C C5722F6CF3F5488B
   2010/01/26   00267D30 MISC RMGREP1    37167108 FFFFC00F 00000001 00085019 00027734 37176780 B83684D2
.
.
.
```

Enhanced Utility Support

The following new utilities manipulate saved trace information:

- The PRINT TRACE utility reports on trace information that resides in the DDLDCTRC area, the DDLDCLOG area, or a trace or log archive file. You can use the PRINT TRACE utility to report chronologically merged trace information from multiple data sharing members.

  Note: The ARCHIVE LOG utility archives trace information written to the DDLDCLOG area, but will not print it. You must use the PRINT TRACE utility for that purpose.

- The ARCHIVE TRACE utility archives and optionally reports on the contents of the DDLDCTRC area.

PRINT TRACE Utility

The PRINT TRACE utility prints all or selected trace entries that reside in a DDLDCTRC area, a DDLDCLOG area, or one or more archive files created by the ARCHIVE TRACE or ARCHIVE LOG utilities.

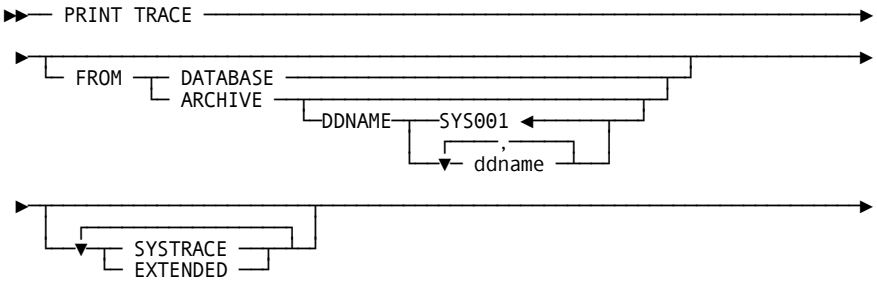
You can select entries for printing based on the following:

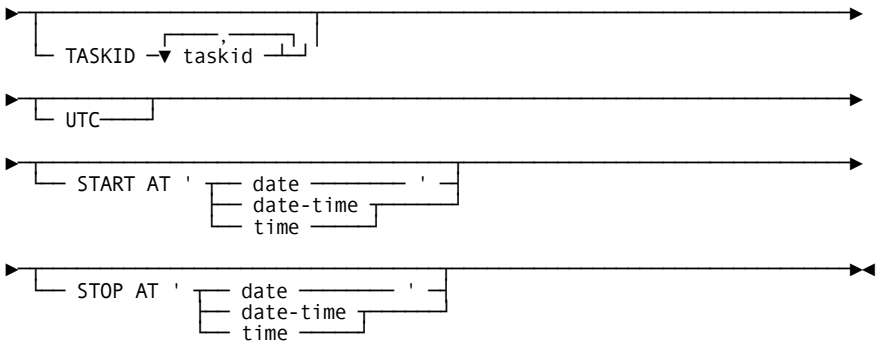
- Type of information
- Date and time
- Taskid

Note: To print trace information you need the DBAREAD privilege on the SYSTEM.DDLDCTRC or SYSTEM.DDLDCLOG area.

PRINT TRACE Syntax

The following diagram shows the syntax for the PRINT TRACE utility statement:





PRINT TRACE Parameters

This section describes the PRINT TRACE command parameters:

FROM

Indicates the location from which to print trace information: a database area or an archive file.

DATABASE

Prints trace information from either the DDLDCTRC or the DDLDCLOG area. If the DMCL contains a DDLDCTRC area, only trace information in that area is printed; otherwise, only trace information in the DDLDCLOG area is printed.

ARCHIVE

Prints trace information from one or more archive files.

DDNAME

Identifies the ddname of one or more archive files whose contents are to be printed. If more than one ddname is specified, trace information is merged and displayed in chronological sequence.

ddname

Specifies the ddname of an archive file.

Default: SYS001

Limit: 32 ddnames

Note: Printing trace information from multiple archive files is only available to z/OS users. z/VSE users need to consolidate multiple archive files into a single file (in the order in which they were created) and use this file with the PRINT TRACE utility.

SYSTRACE

Includes basic system trace information in the output.

EXTENDED

Includes extended trace information in the output.

Note: If you specify SYSTRACE or EXTENDED, only the specified type of information is printed. If you do not specify either option, both types of information are printed.

TASKID

Prints trace information for selected tasks.

taskid

Specifies the taskid of a task whose trace information is to be printed.

Default: Trace information associated with all tasks is printed.

Limit: 32 task identifiers.

START AT *date-time*

Prints only trace information recorded at or after the specified time.

Default: Prints information from the beginning of the database area or archive file.

STOP AT *date-time*

Prints only trace information recorded at or before the specified date and time.

Default: Prints information recorded in the database area or archive file (starting at the time specified in the START AT parameter, if any).

Expansion of date-time

date

Specifies a date, in one of the following formats:

- yyyy-mm-dd
- mm/dd/yyyy

The date components are:

- yyyy specifies the year.

  Limit: 0001–9999 (leading zeros are optional)

- mm specifies the month within the year.

  Limit: 01–12 (leading zeros are optional)

- dd specifies the day within the month.

  Limit: 01–31 (leading zeros are optional)

Note: The combined values of yyyy, mm, and dd must represent a valid date. For example, 1988-02-29 is a valid date but 1989-02-29 is not.

date-time

Specifies a date and time in one of the following formats:

- yyyy-mm-dd-hh.mm.ss.ffffff
- yyyy-mm-dd-hh.mm.ss

The date components are the same as those that can be specified for *date*.

The time components are:

- hh specifies the hour on a 24-hour clock.

  Limit: 00-23 (leading zeros are optional)

- mm specifies the number of minutes past the hour.

  Limit: 00-59 (leading zeros are optional)

- ss specifies the number of seconds past the minute.

  Limit: 00-59 (leading zeros are optional)

- ffffff specifies the number of millionths of a second past the specified second.

  Limit: 000000-999999 (trailing zeros are optional)

  Default: 000000

time

Specifies a time in the following format:

- hh:mm:ss

The time components are the same as those that can be specified for *date-time*.

Note: The date is assumed to be the current date.

UTC

Specifies that Start and Stop times are interpreted as UTC times instead of local times.

Examples

PRINT TRACE Output for System Trace Entries

The following items are included in the PRINT TRACE output for system trace entries.

TOD TIME/DATE

The UTC time and date the trace entry was generated.

SCA

The relative SCA number of the subtask that generated the trace entry.

TASKID

Taskid of the task that generated the trace entry.

A(LTE)

The address of the generating task's LTE.

MOD

The internal module number and four-character identifier of the module generating the trace entry.

MAC

The issuing macro number or entry point.

CALL

The relative macro expansion within the issuing program.

R11/R2-R1/R8

The contents of registers 11 through 1 and 2 through 8 at the time the entry was generated.

TOD CLOCK

The contents of the TOD clock at the time the entry was generated.

<SYSTEM73>

Node or data sharing member name of issuing system.

PRINT TRACE Output for Extended Entries

The following PRINT TRACE statement requests printing of all SYSTRACE and EXTENDED trace entries from the archive file starting from March 02,2011 at 4:12 p.m. until just before 4:30 p.m. on March 02,2011 using the archive file with DDNAME SYS001.

```
PRINT TRACE FROM ARCHIVE
   DDNAME SYS001
   START AT '2011-03-02-21.12.00'
   STOP AT  '2011-03-02-21.30.00';
```

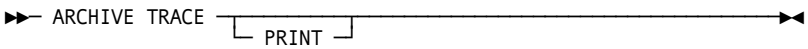
ARCHIVE TRACE Utility

The ARCHIVE TRACE utility offloads the contents of the DC/UCF trace area to an archive file.

Note: To archive trace information you need DBAWRITE privilege on the SYSTEM.DDLDCTRC area.

ARCHIVE TRACE Syntax

The following diagram shows the syntax for the ARCHIVE TRACE utility statement:



ARCHIVE TRACE Parameter

This section describes the parameters for the ARCHIVE TRACE utility:

PRINT

Prints a copy of the contents of the archived trace information. If you do not specify PRINT, a copy of the archived trace information is not printed.

Note: For a description of the output produced by the PRINT option of the ARCHIVE TRACE utility, see the PRINT TRACE Utility (see page 95).

Example: Archive the Contents of a Trace Area

The following example writes the contents of a system trace area (DDLDCTRC) to an archive file and empties the area.

```
ARCHIVE TRACE
```

Sample WTO Exit

The sample WTO exit distributed with CA IDMS has been updated to automate the offloading of trace information from the DDLDCTRC area. It now performs the following activities:

- Searches for DC050024 messages indicating that the trace area has reached a certain percent full
- Submits the contents of the file whose ddname is PTRCJOB to the internal reader if the percent is 25 or more.

To use this sample exit, you must add a PTRCJOB DD statement to your DC/UCF system startup JCL. The contents of the associated file should include a command facility job stream that executes the ARCHIVE TRACE utility.

Enhanced DCPROFIL Output

The DCPROFIL task is enhanced to display additional information about a DC/UCF system, which makes it easier for you to determine what options are in effect.

The DCPROFIL output now includes the following information:

- The external security system being used and the way in which signon is secured.
- An indication if trace information is being saved and the area to which it is written.
- The named user exits that are enabled.
- The ADS comment delimiter in effect.
- The product intent status for all licensable products.

Examples: DCPROFIL output

These examples show the new information contained in the first page of DCPROFIL output, in the Named User Exit page, and in the revised ADSO Control Block display:

```
TAPE:              GJI00B          NUMBER OF SCTS:      0005

TOOLS TAPE:        -NONE-          OPERATING SYSTEM:    z/OS  ZIIP=N

SYSTEM TRACE:      YES             TRACE SAVE:  OFF  (DDLDCTRC)

CWA SIZE:      0000000000          DMCL TABLE:      CVDMCL

                        PRIMARY STORAGE
SCRATCH HWM    0000000185          PROTECT KEY:         08

SIZE OF XA                     ACTIVE TRANSACTION
STORAGE AREA:  0044793856          COUNT:           0011

QUEUE AREA                     SECURITY
LOW PAGE:      0000040001          SECURITY SYSTEM: CA TOP SECRET
HIGH PAGE:     0000041000          SIGNON SECURITY: ON

DC VERSION ID:     0073        SVC NUMBER:        172

USER TRACE BUFFERS:  0500      GETMAIN SUBPOOL:     001

                  PAGE 00001 - NEXT PAGE:
```

```
                    * Named User Exits *
EXIT              ENTRY          EXIT              ENTRY
NAME      DEFINED    POINT          NAME      DEFINED    POINT

BTCIDXIT   NO        00000000        DBLUEREX   NO        00000000
IDDEXITB   NO        00000000        IDDEXITO   NO        00000000
IDMSAJNX   NO        00000000        IDMSCLCX   NO        00000000
IDMSDPLX   NO        00000000        IDMSIOXT   NO        00000000
IDMSIOX2   YES       3B095E00        IDMSJNL2   NO        00000000
OLQDMLX    NO        00000000        SCHEXITB   NO        00000000
SCHEXITO   NO        00000000        SGNEXITB   NO        00000000
SGNEXITO   NO        00000000        SUBEXITB   NO        00000000
SUBEXITO   NO        00000000        TCKREXIT   NO        00000000
USRIDXIT   NO        00000000        WAITEXIT   NO        00000000
WTOEXIT    NO        00000000        WTOREXIT   NO        00000000
                  PAGE 00004 - NEXT PAGE:
```

```
                    * ADSO CONTROL BLOCK *

MENU PROC. PRG. TASK     ADS          MAIN PROC. TASK CODE     ADS2
AUTO DIALOG NAME                   PRIMARY REC. BUFF SZ     0000004084
SECOND. REC. BUFF SZ     0000004084     INTRNL. TASK CDE TCF     ADS2T
FAST MODE THRESHOLD      NO           THRESHOLD FOR SCRTCH     0000000000
MAX. LINK LEVEL          0010         STAT. DEF. REC. VRSN     0001
USER MENU ONLY           YES          KEEP MENU IN SCRATCH     NO
AUTOSTATUS = YES         YES          AUTOSTATUS MANDATORY     NO
STAT. DEF. MANDATORY     NO           BYPASS DIAG. SCREEN      NO
MAPOUT NEWPAGE REQ.      NO           RELOCATABLE STORAGE      NO
ACTIVITY LOGGING         NO           DIAGLOG STATS. COLL.     NO
SELECTED DIALOG STAT     NO           COBOL MOVE               NO
STATS. CHKPOINT CNT.     0000         STORAGE MODE CALC.       YES
COMMENT DELIMITER        !
STATUS DEF. REC NAME     ADSO-STAT-DEF-REC


          PAGE 00005 - NEXT PAGE:
```

Display Memory Enhancements

The DCMT DISPLAY MEMORY command was enhanced to give you the following capabilities:

- Display DMCL-related entities.
- Support indirect addressing for identifying an area to be displayed.

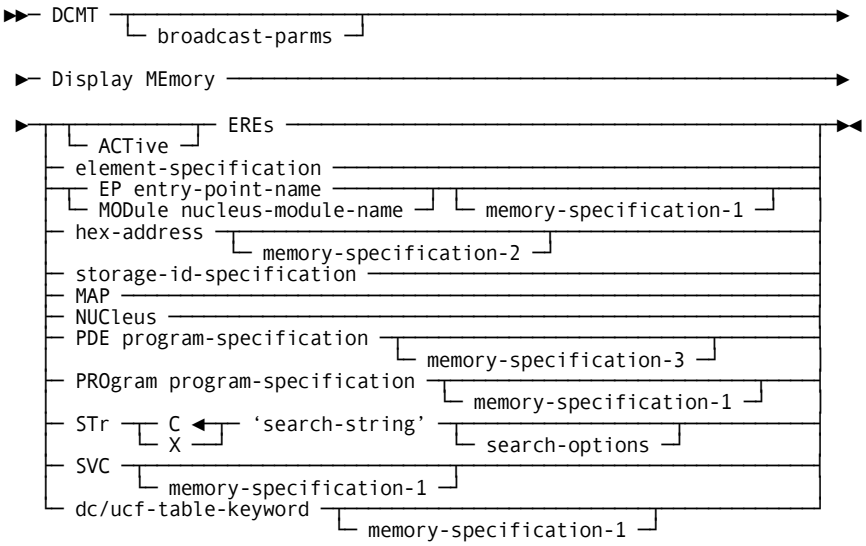
These capabilities make it easier for you to locate areas of interest when diagnosing a problem. The enhancements also facilitate the inclusion of DCMT DISPLAY MEMORY commands in UCF batch scripts because they can be independent of the absolute address of the area you need to display.

DCMT DISPLAY MEMORY Command

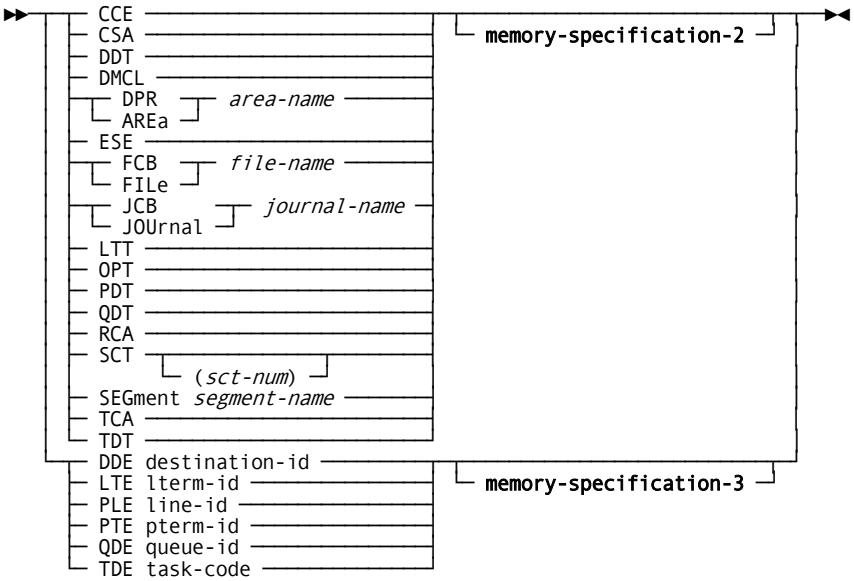
The DCMT DISPLAY MEMORY command was enhanced to display portions of the DMCL and support indirect addressing.

DCMT DISPLAY MEMORY Syntax

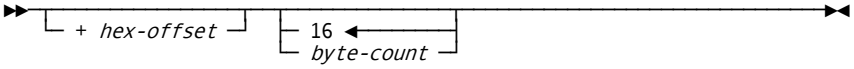
The following diagram shows the complete syntax for the enhanced DCMT DISPLAY MEMORY command:



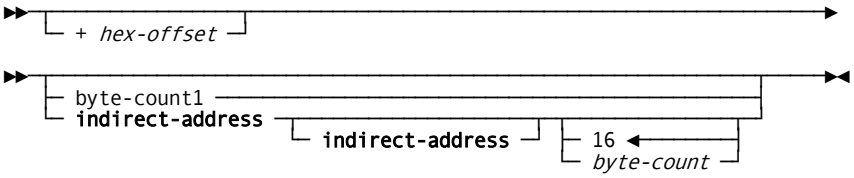
Expansion of element-specification



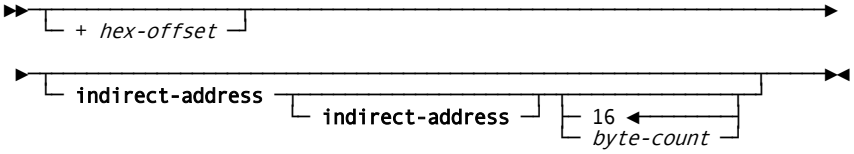
Expansion of memory-specification-1



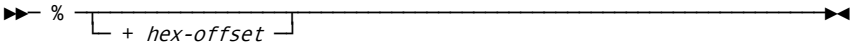
Expansion of memory-specification-2



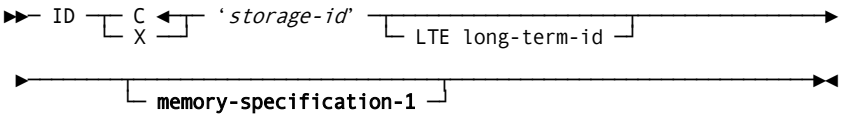
Expansion of memory-specification-3



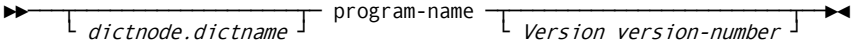
Expansion of indirect-address



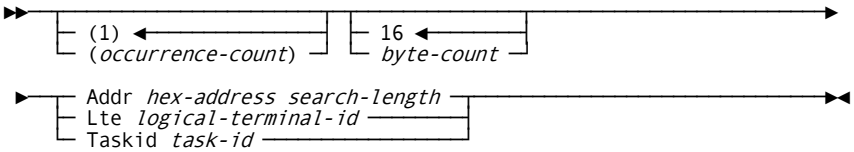
Expansion of storage-id-specification



Expansion of program-specification



Expansion of search-options



DCMT DISPLAY MEMORY Parameters

This section describes the new parameters that support the enhancements to the DCMT DISPLAY MEMORY command:

broadcast-parms

Executes the DCMT command on all or a list of data sharing group members.

For more information on broadcasting and broadcast-parms syntax, see the section "How to Broadcast System Tasks" in the *System Tasks and Operator Commands Guide*.

element-specification

Identifies an area of memory to display.

PROgram *program-specification*

Specifies a program or a nucleus module as the base location of the memory to display. program-specification must identify a program or a module residing in the DC/UCF address space. To identify a program that was loaded from an alternate data dictionary, specify DICTNODE or DICTNAME as described in the Expansion of program-specification.

memory-specification-1

Provides additional information about the location and length of the memory to display.

memory-specification-2

Provides additional information about the location and length of the memory to display.

memory-specification-3

Provides additional information about the location and length of the memory to display.

Expansion of element-specification

DMCl

Displays memory content beginning at the start of the DMCL's DM58 control block.

DPR|AREa

Displays memory content beginning at the start of the DPR (PR60) control block for the named area. Both keywords are synonymous and give the same result.

segment.area-name

Identifies the area whose DPR control block is to be displayed. segment.area-name must identify a physical area included in the DMCL.

FCB|FILe

Displays memory content beginning at the start of the FCB (FC59) control block for the named file. Both keywords are synonymous and give the same result.

segment.file-name

Identifies the file whose FCB control block is to be displayed. segment.file-name must identify a file included in the DMCL.

JCB|JOUrnal

Displays memory content beginning at the start of the JCB (JD62) control block for the named journal. Both keywords are synonymous and give the same result.

journal-name

Identifies the journal whose JCB control block is to be displayed. journal-name must identify a journal included in the DMCL.

SEGment

Displays memory content beginning at the start of the SEG (SG49) control block for the named segment.

segment

Identifies the segment whose SEG control block is to be displayed. segment must identify a segment included in the DMCL.

Expansion of memory-specification-1

+ hex-offset

Displays memory content beginning at the specified hexadecimal offset from the requested starting location.

Default: 0 (zero)

byte-count

Specifies the number of bytes to be displayed, rounded to the next multiple of 4.

Default: 16 bytes

Expansion of memory-specification-2

+ hex-offset

Identifies a location in memory as a hexadecimal offset from the requested starting location.

Default: 0 (zero).

If an **indirect-address** is not specified, the offset identifies the start of the memory to display.

If an **indirect-address** is specified, the offset identifies a location whose content is an address.

byte-count1

If no indirect-address is specified, byte-count1 is the number of bytes to be displayed, rounded to the next multiple of 4.

Default: The length of the specified control block.

indirect-address

Indicates that the location in memory identified by the preceding parameters contains an address that is to be used in identifying the memory to be displayed.

The last **indirect-address** specified identifies the start of the memory to be displayed.

byte-count

Specifies the number of bytes to be displayed, rounded to the next multiple of 4.

Default: 16 bytes.

Expansion of memory-specification-3

+ hex-offset

Identifies a location in memory as a hexadecimal offset from the requested starting location.

Default: 0 (zero).

If an **indirect-address** is not specified, the offset identifies the start of the memory to be displayed.

If an **indirect-address** is specified, the offset identifies a location whose content is an address.

indirect-address

Indicates that the location in memory identified by the previous parameters contains an address that is to be used in identifying the memory to display.

The last indirect-address specified identifies the start of the memory to display.

byte-count

Specifies the number of bytes to be displayed, rounded to the next multiple of 4.

Default: 16 bytes.

Expansion of indirect-address

%

Indicates that the location in memory identified by the preceding parameters is an address of a location of memory.

+ hex-offset

Identifies a location in memory as a hexadecimal offset from the indirectly addressed location.

Expansion of program-specification

dictnode

Specifies the DDS node that controls the data dictionary from where the named program was loaded.

dictname

Specifies the alternate data dictionary from where the named program was loaded.

Note: Although *dictnode* and *dictname* are both optional parameters, if *dictnode* is specified and *dictname* is not specified, a "." (period) must be included to represent the missing *dictname* parameter.

program-name

Identifies the name of a program or nucleus module that resides in the DC/UCF address space.

Version *version-number*

Specifies the version number of the named program.

Limits: 1 - 9999

Default: 1

Example: Display Memory Outputs

The following example illustrates displaying the DPR control block for an area.

```
DCMT D MEM AREA EMPDEMO.EMP-DEMO-REGION
 <Addr>  <Offset>              <Hex>                 <Character>
36F11598  00000000  D7D9F6F0 1080C5D4 D7C4C5D4 D64BC5D4  *PR60..EMPDEMO.EM*
36F115A8  00000010  D760C4C5 D4D660D9 C5C7C9D6 D5404040  *P-DEMO-REGION   *
36F115B8  00000020  40016632 8F8AA1D5 C0000000 000124F9  * ......N{......9*
36F115C8  00000030  0001255C 0000FF08 00000000 0000084A  *...*...........¢*
36F115D8  00000040  36F11778 36F116F8 36F116C8 36F11168  *.1...1.8.1.H.1..*
36F115E8  00000050  36F11778 00000000 0000FF03 00010000  *.1..............*
36F115F8  00000060  00000000 00000000 00000000 36F11598  *.............1.q*
36F11608  00000070  36F11598 03030000 0000C008 36F116F8  *.1.q......{...1.8*
36F11618  00000080  36F116C8 00000000 00000000 00000000  *.1.H............*
36F11628  00000090  00000000 00000000 00000000 00000000  *................*
36F11638  000000A0  00000000 00000000 00000000 00000000  *................*
36F11648  000000B0  00000000 00000000 00000000 40800000  *............ ...*
36F11658  000000C0  00000000 00000000 00000000 00000000  *................*
36F11668  000000D0  00000000 00000000 00000000 00000000  *................*
36F11678  000000E0  00000000 00000000 00000000 00000000  *................*
36F11688  000000F0  00000000 00000000 00000000 00000000  *................*
36F11698  00000100  00000000 C5D4D760 C4C5D4D6 60D9C5C7  *....EMP-DEMO-REG*
36F116A8  00000110  C9D6D540 40400000 40000000 00000000  *ION   .. .......*
36F116B8  00000120  00000000 00000000 00000000 00000000  *................*
```

The following example illustrates the use of indirect addressing to display the first file associated with an area:

```
DCMT D MEM AREA EMPDEMO.EMP-DEMO-REGION +44 % +10 % 200
 <Addr>  <Offset>              <Hex>                 <Character>
36F111A8  00000000  C6C3F5F9 0190C5D4 D7C4C5D4 D64BC5D4  *FC59..EMPDEMO.EM*
36F111B8  00000010  D7C4C5D4 D6404040 40404040 40404040  *PDEMO          *
36F111C8  00000020  40016632 8F8AA1D5 C0C5D4D7 C4C5D4D6  * ......N{EMPDEMO*
36F111D8  00000030  40000000 000010B4 00000064 40404040  * ...........    *
36F111E8  00000040  40404040 40404040 40404040 40400000  *              ..*
36F111F8  00000050  36F112F8 36F112F8 36F116F8 36F0CD48  *.1.8.1.8.1.8.0..*
36F11208  00000060  00000000 36F11168 00000000 00000000  *.....1.........*
36F11218  00000070  00000000 00000000 00000000 00000000  *................*
36F11228  00000080  00000000 00000000 36F112A8 00000000  *.........1.y....*
36F11238  00000090  00000000 00000000 00000000 00000000  *................*
36F11248  000000A0  36F116F8 00000000 00000000 36F112F8  *.1.8.........1.8*
36F11258  000000B0  00000000 08180000 00000001 00000000  *................*
36F11268  000000C0  00000000 36F1D490                    *.....1M.
```

Vary Memory Enhancements

The DCMT VARY MEMORY command was enhanced to give you the following new capabilities:

- Use relative addressing in identifying the area to vary.
- Enable verification of the contents of the area to vary.

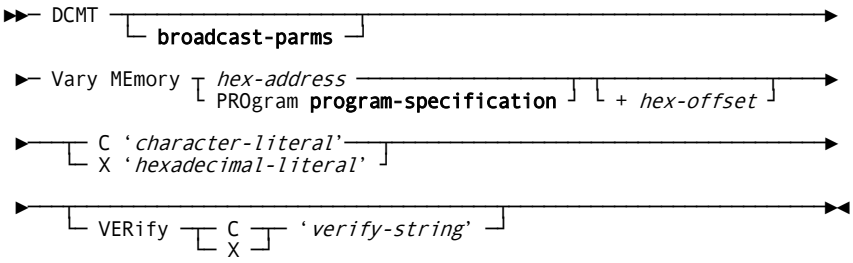
These two capabilities make changing memory contents easier and less error-prone. Because these capabilities can be independent of the absolute address of the area to be varied, these enhancements facilitate the inclusion of DCMT VARY MEMORY commands in UCF batch scripts.

DCMT VARY MEMORY Command

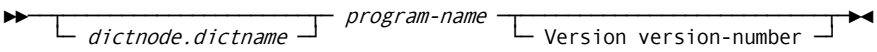
The DCMT VARY MEMORY command has been enhanced to validate memory contents and support offsets.

DCMT VARY MEMORY Syntax for memory enhancements

The following diagram shows the complete syntax for the enhanced DCMT VARY MEMORY command:



Expansion of program-specification



DCMT VARY MEMORY Parameters for memory enhancements

This section describes the parameters for the DCMT VARY MEMORY command.

broadcast-parms

Executes the DCMT command on all or a list of data sharing group members.

For more information on broadcasting and broadcast-parms syntax, refer to the section "How to Broadcast System Tasks" in the *System Tasks and Operator Commands Guide*.

hex-address

Specifies an address as the base location of the memory to be varied. Hex-address must be a 1 – 8-digit hexadecimal value identifying a location in memory within the DC/UCF address space. You can omit leading zeros.

PROgram program-specification

Specifies a program as the base location of the memory to be varied. Program-specification must identify a program or nucleus module residing in the DC/UCF address space.

Note: To identify a program that was loaded from an alternate data dictionary, specify DICTNODE or DICTNAME as described under Expansion of program-specification.

+ hex-offset

Specifies the relative offset of the memory to be varied from the base location. Hex-offset must be a valid hexadecimal value.

Default: 0

C *'character-literal'*

Specifies the value to which the identified memory is to be changed. character-literal must be a valid character string.

Limits: 1-32 character string

X *'hexadecimal-literal'*

Specifies the value to which the identified memory is to be changed. hexadecimal-literal must be a valid hexadecimal value.

Limits: 1-32 digits

VERify

Requests verification of the current memory content; if the verification fails, the command returns an error and does not change the contents of memory.

C

Indicates that the verify string is in character format.

X

Indicates that the verify string is in hexadecimal format.

'verify-string'

Specifies the string to use to verify the current memory content. If the memory content does not match the specified string, verification fails.

Limits:

- 1-32 character value if in character format.
- 1-32 digit (16-byte) hexadecimal value if in hexadecimal format.

More Information

- For more information about displaying memory contents, see the section DCMT DISPLAY MEMORY.

Expansion of program-specification

dictnode

Specifies the DDS node that controls the data dictionary from where the named program was loaded.

dictname

Specifies the alternate data dictionary from where the named program was loaded.

Note: Although *dictnode* and *dictname* are both optional parameters, if *dictnode* is specified and *dictname* is not specified, a "." (period) must be included to represent the missing *dictname* parameter.

program-name

Identifies the name of a program or nucleus module that resides in the DC/UCF address space.

Version *version-number*

Specifies the version number of the named program.

Limits: 1 - 9999

Default: 1

Examples: Vary Memory Commands

DCMT VARY MEMORY

The following example illustrates the use of relative addressing from the start of a program to identify the location of memory to vary. The example also ensures that the offset is correct by verifying its contents before allowing the operation to proceed.

```
DCMT V MEMORY PROGRAM RHDCMTME +60 C 'CA TEST' VERIFY C 'CA-IDMS'
Program: RHDCMTME  Loadlib: CDMSLIB
 <Addr>  <Offset>          <Hex>              <Character>
38BEE860  00000000  C3C140E3 C5E2E340 41C0F000 4180C800  *CA TEST .{0...H.*
```

DCMT VARY MEMORY with a failed verify

The following example illustrates what happens if the contents of memory do not match the verification value.

```
DCMT V MEMORY PROGRAM RHDCMTME +60 C 'CA TEST' VERIFY C 'CA-IDMS'
Program: RHDCMTME  Loadlib: CDMSLIB
IDMS DC269903 V73 VERIFY FAILED - VARY MEMORY NOT DONE
 <Addr>  <Offset>          <Hex>              <Character>
38BEE860  00000000  C3C140E3 C5E2E340 41C0F000 4180C800  *CA TEST .{0...H.*
```

CICS Wait Information

When the standard CICS interface (IDMSINTC) enters a CICS wait, the type of wait is now identified. This information can be useful in diagnosing unexplained processing delays.

The IDMSINTC interface specifies a NAME parameter when it issues an EXEC CICS WAIT. The value of this parameter identifies the type of wait being done. The values that are assigned to the NAME parameter and their meanings are:

IDMSDL1T

Task is waiting for CA IDMS DLI Transparency processing.

IDMSSYNC

Task is waiting for two-phase commit synchronization to complete.

IDMS EXT

Task is waiting for completion of a UCF or DML request to a CA IDMS back end.

Note: You can display the type of wait being done using CICS facilities such as CEMT INQ TASK.

Enhanced Module Identification

To comply with the Mainframe 2.0 serviceability standards, most assembler modules distributed by CA now contain the following identification and state information:

- Module release level
- Date and time of assembly
- Function identifier (FMID) with which the module is associated
- Required maintenance identifier (RMID) indicating the module's maintenance level.

You can display this information using the following CA IDMS facilities.

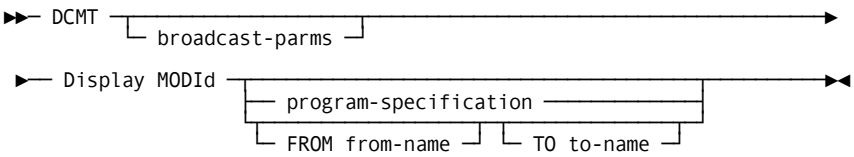
- The new DCMT DISPLAY MODID command displays detailed information about an individual program and summary information for a range of programs.
- LOOK DATES and LOOK PROGRAM now additionally display the FMID and RMID for the components of a specific program.

DCMT DISPLAY MODID Command

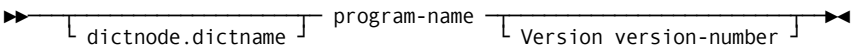
The DCMT DISPLAY MODID command displays identifying information for components of one or more programs.

DCMT DISPLAY MODID Syntax

The following diagram shows the syntax for the DCMT DISPLAY MODID command:



Expansion of program-specification



DCMT DISPLAY MODID Parameters

This section describes the DCMT DISPLAY MODID command parameters:

broadcast-parms

Executes the DCMT command on all or a list of data sharing group members.

For more information about broadcasting and broadcast-parms syntax, see the *System Tasks and Operator Commands Guide*

program-specification

Identifies the program for which information is to be displayed. program-specification must identify a program or nucleus module residing in the DC/UCF address space.

To identify a program that was loaded from an alternate data dictionary, specify DICTNODE or DICTNAME as described in Expansion of program-specification.

FROM from-name

Identifies the first program for which information is to be displayed. Information is displayed for all programs and nucleus modules whose name is greater than or equal to from-name.

Default: Spaces if TO to-name is specified.

TO to-name

Specifies the name of the last program for which information is to be displayed. Information is displayed for all programs and nucleus modules whose name is less than program-name or begins with to-name.

Default: Z if FROM from-name is specified.

Note: If program-specification, from-name and to-name are not specified, information is displayed for all programs and nucleus modules residing in the DC/UCF address space.

Expansion of program-specification

program-name

Specifies the name of a program or nucleus module that resides in the DC/UCF address space.

dictnode

Specifies the DDS node that controls the data dictionary from which the named program was loaded.

dictname

Specifies the alternate data dictionary from which the named program was loaded.

Note: Although dictnode and dictname are both optional parameters, if dictnode is specified and dictname is not specified, you must include a . (period) to represent the missing dictname parameter.

Version version-number

Specifies the version number of the named program. version-number must be an integer in the range 1 through 9999.

Default:1

Examples: Displaying module information

The following example shows how to use the DCMT command to display module information about all components of one program, IDMSDBMS.

```
DCMT D MODID IDMSDBMS
  Module Name     IDMSDBMS
  Assembly Date   20100111
  Assembly Time   16.41
  Product Name    CA IDMS
  Product Release 18.0
  RMID            CAGJI00
  Component ID    CAGJI00
  Copyright (C)   1972-2010 CA.   ALL RIGHTS RESERVED.


  Module Name     IDMSDBM2
  Assembly Date   20100111
  Assembly Time   16.41
  Product Name    CA IDMS
  Product Release 18.0
  RMID            CAGJI00
  Component ID    CAGJI00
  Copyright (C)   1972-2010 CA.   ALL RIGHTS RESERVED.
```

The following example shows how to use the DCMT command to display module information for a range of programs from IDMSDB to IDMSDD.

```
DCMT D MODID FROM IDMSDB TO IDMSDD
Module      Date     Time  FMID     RMID
IDMSDBIO
  IDMSDBIC  20100111 16.41 CAGJI00  CAGJI00
  IDMSDBIB  20100111 16.41 CAGJI00  CAGJI00
  IDMSDBIJ  20100111 16.41 CAGJI00  CAGJI00
  IDMSDBID  20100111 16.41 CAGJI00  CAGJI00
  IDMSDBIM  20100111 16.41 CAGJI00  CAGJI00
  IDMSDBIO  20100111 16.41 CAGJI00  CAGJI00
  IDMSDBIV  20100111 16.41 CAGJI00  CAGJI00
  IDMSDBIX  20100111 16.41 CAGJI00  CAGJI00
  IDMSDBIT  20100111 16.41 CAGJI00  CAGJI00
IDMSDBMS
  IDMSDBMS  20100111 16.41 CAGJI00  CAGJI00
  IDMSDBM2  20100111 16.41 CAGJI00  CAGJI00
IDMSDCLI    20100111 16.42 CAGJI00  CAGJI00
IDMSDCOM    20100111 16.42 CAGJI00  CAGJI00
IDMSDDAM
  IDMSDDAM  20100111 16.42 CAGJI00  CAGJI00
  IDMSDDAT  20100111 16.42 CAGJI00  CAGJI0
```

Enhanced LOOK Output

The LOOK system task and the IDMSLOOK utility program now display the FMID and RMID of all component modules when you use the DATES and PROGRAM functions.

Example: Displaying Component Module Information

The following LOOK command displays information about the component modules of program IDMSCHDC.

```
LOOK DATES=IDMSCHDC
IDMSLOOK  -  OPSYS=z/OS       Release 18.0 Service pack 0   tape GJI00B
DATES=IDMSCHDC

      was loaded From CDMSLIB DSN --> IDMSNDV.MOTM.IDMS.BASE.P2.LOADLIB
Entry Point Offset +0     -  Reentrant     -  AMODE 31  -  RMODE ANY
    31,016 Bytes in Load Module IDMSCHDC loaded at 38D1CA00

       Module   Offset  Date   Time   FMID     RMID

      IDMSFSED  +18     100111  1645  CAGJI00  CAGJI00
      IDDSFEDC  +6080   100111  1637  CAGJI00  CAGJI00
      IDMSCHPT  +7408   100111  1640  CAGJI00  CAGJI00
      IDMSDATE  +7878   100111  1712  CAGJI00  CAGJI00
```

Standardized Dump Title

Dump titles are produced by products when there is an unexpected abend. The dump title format has been standardized for all CA mainframe products. Standardization makes it easier to determine the nature of the abend and identify the failing module.

Example: CA Standard Dump Title

The following example shows the CA standard dump title:

```
13.53.56 JOB30322  +CCSR010E RHDCOESA S0C4 at 0000DB26 LMOD RHDCOMVS CSECT RHDCOESA +006B26 SYSTEM72 TESTDCV DCV
13.53.56 JOB30322  +CCSR021I OWNER = CA IDMS 18.0.0
13.53.56 JOB30322  +CCSR022I MODULE = RHDCOESA FMID = CAGJI00 RMID = CAGJI00
13.53.56 JOB30322  +CCSR061I PSW: 00000000 00000000 078D1F00 8000DB26
13.53.56 JOB30322  +CCSR062I ILC: 02 INTERRUPT CODE: 04 REASON CODE: 00000004
13.53.56 JOB30322  +CCSR063I TRANSLATION EXCEPTION ADDRESS: 00000000_00018000
13.53.56 JOB30322  +CCSR064I DATA AT PSW 0000DB1E : 00181610 980FC178 10110101
13.53.56 JOB30322  +CCSR065I HOME = 0084  PRIMARY = 0084  SECONDARY = 0084
13.53.56 JOB30322  +CCSR070I GR0  - GR1   00000000_C0000000 00000000_C4004000
13.53.56 JOB30322  +CCSR070I GR2  - GR3   00000000_00006CE0 00000000_367D5B54
13.53.56 JOB30322  +CCSR070I GR4  - GR5   00000000_00000000 00000000_3790E47E
13.53.56 JOB30322  +CCSR070I GR6  - GR7   00000000_00000000 00000000_0002ECE0
13.53.56 JOB30322  +CCSR070I GR8  - GR9   00000000_3790B954 00000000_0020B7A0
13.53.56 JOB30322  +CCSR070I GR10 - GR11  00000000_00027068 00000000_8000EBEC
13.53.56 JOB30322  +CCSR070I GR12 - GR13  00000000_00000000 00000000_3657C274
13.53.56 JOB30322  +CCSR070I GR14 - GR15  00000000_00FF4950 00000000_00000004
13.53.56 JOB30322  +CCSR071I AR0  - AR3   007FE030 00000000 00000000 00000000
13.53.56 JOB30322  +CCSR071I AR4  - AR7   00000000 00000000 00000000 00000000
13.53.56 JOB30322  +CCSR071I AR8  - AR11  00000000 00000000 00000000 00000000
13.53.56 JOB30322  +CCSR071I AR12 - AR15  00000000 00000000 00000000 00000000
13.53.56 JOB30322  IEA794I SVC DUMP HAS CAPTURED:  455
  455          DUMPID=123 REQUESTED BY JOB (SYSTEM72)
  455          DUMP TITLE= CCSR010E RHDCOESA S0C4 at 0000DB26 LMOD RHDCOMVS CSECT RHDCOESA +006B26 SYSTEM72 TESTDCV DCV
```

Chapter 7: Administrative and Operational Enhancements

This section contains the following topics:

Automatic Tuning

A new auto-tuning capability lets CA IDMS set the startup values for certain DC/UCF system configuration parameters based on historical information. The values are automatically adjusted over time in response to changes in workload. Auto-tuning relieves the DBA from having to monitor and adjust the parameter values manually.

The following system definition parameters are eligible for automatic tuning:

- RLE count
- RCE count
- DPE count
- SYSLOCKS

New clauses on the system generation SYSTEM statement enable or disable automatic tuning for each of these parameters.

Notes:

- To use auto-tuning, change tracking must be active for the DC/UCF system, since statistical information needed for tuning is captured in the SYSTRK files.
- New DCMT commands display and reset values used in tuning.

While the use of auto-tuning can be beneficial in many situations, it is not recommended under the following conditions:

- If a system is normally active for only short durations (less than 24 hours).
- If a parameter must be set to accommodate a rarely-executed workload.

How Automatic Tuning Works

When auto-tuning is enabled, CA IDMS begins to collect high water mark (HWM) information for the parameters being tuned. This information is collected on 24-hour intervals from the time the DC/UCF system is started. At the end of each interval, the statistics needed for tuning are updated and written to the AUTOTUNE member in SYSTRK and each HWM value is set to the current value of its associated parameter.

Statistical information is also collected at shutdown, but only if the system has been active for at least eight hours.

After five sets of statistics have been gathered and at the end of each interval thereafter, CA IDMS calculates a new value for each of the parameters being tuned. These values are used the next time the DC/UCF system is started after a normal shutdown. If the system is started after an abnormal termination, the parameter values from the previous execution are used. If you change the SYSGEN value of a tuned parameter, the new SYSGEN value is used the next time the system is started, and all historical information for the parameter is discarded.

New parameter values are recalculated at the end of each interval using the most recent HWM and information gathered from prior intervals. A smoothing algorithm is used so that abnormal conditions, such as runaway tasks, do not skew the results. Tuned values can both increase and decrease; however, they decrease more slowly than they increase.

You can use the DCMT DISPLAY AUTOTUNE command to see the current and new values for each of the tuned parameters and obtain graphs of HWM values over the last 32 time intervals.

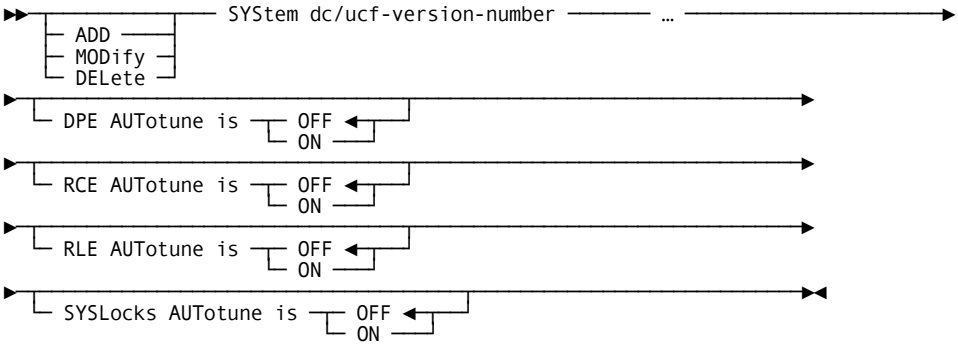
SYSGEN SYSTEM Statement Enhancements for Automatic Tuning

The SYSGEN SYSTEM statement has been enhanced to enable the automatic tuning of certain parameters. You can now enable automatic tuning for the following parameters:

- DPE count
- RCE count
- RLE count
- SYSLOCKS

SYSGEN SYSTEM Statement Syntax for Automatic Tuning

The following syntax contains enhancements to the SYSGEN SYSTEM statement:



SYSGEN SYSTEM Statement Parameters for Automatic Tuning

This section describes the new parameters that support the Auto-Tuning enhancement to the SYSGEN SYSTEM statement:

DPE AUTotune is

Indicates if CA IDMS should automatically tune the DPE count value using execution statistics.

OFF

Turns off automatic tuning of DPE count.

ON

Turns on automatic tuning of DPE count.

Default: **OFF**

RCE AUTotune is

Indicates if CA IDMS should automatically tune the RCE count value using execution statistics.

OFF

Turns off automatic tuning of RCE count.

ON

Turns on automatic tuning of RCE count.

Default: **OFF**

RLE AUTotune is

Indicates if CA IDMS shouldautomatically tune the RLE count value using execution statistics.

OFF

Turns off automatic tuning of RLE count.

ON

Turns on automatic tuning of RLE count.

Default: **OFF**

SYSLocks AUTotune is

Indicates if CA IDMS shouldautomatically tune the SYSLOCKS parameter value using execution statistics.

OFF

Turns off automatic tuning of SYSLOCKS.

ON

Turns on automatic tuning of SYSLOCKS.

DCMT Enhancements for Automatic Tuning

This release provides the following new DCMT command enhancements specific to automatic tuning:

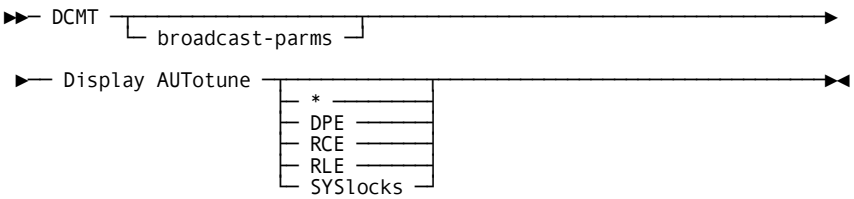
- The new DCMT DISPLAY AUTOTUNE command shows statistics used in automatic tuning.
- The new DCMT VARY AUTOTUNE command disables automatic tuning or clears the statistics so that tuning calculations are based on future information.
- The DCMT DISPLAY CHANGE TRACKING output now includes the new AUTOTUNE member if auto-tuning is enabled.

DCMT DISPLAY AUTOTUNE Command

The DCMT DISPLAY AUTOTUNE command shows statistics related to automatic tuning.

DCMT DISPLAY AUTOTUNE Syntax

The following diagram shows the syntax for the DCMT DISPLAY AUTOTUNE command:



DCMT DISPLAY AUTOTUNE Parameters

This section describes the parameters for the DCMT DISPLAY AUTOTUNE command:

broadcast-parms

Executes the DCMT command on all or a list of data sharing group members. For more information about the broadcasting and broadcast-parms syntax, see How to Broadcast System Tasks in the *System Tasks and Operator Commands Guide*.

*****

Displays graphs and summary information for all parameters for which automatic tuning is enabled.

DPE

Displays the graph and summary information for the DPE count parameter.

RCE

Displays the graph and summary information for the RCE count parameter.

RLE

Displays the graph and summary information for the RLE count parameter.

SYSlocks

Displays the graph and summary information for the SYSLOCKS parameter.

Note: If no parameter type is specified, summary information for all parameters is displayed.

Example: Automatic Tuning Results

The following command displays summary information about all parameters being automatically tuned:

```
DCMT D AUTOTUNE
*** Display Autotune ***

 AUTOTUNE last save time (time zone: UTC): 2009-12-21-15.10.12.528239

  Parameter|Lowest  |Highest |SYSGEN  |Current | Next
  Name     | HWM    | HWM    | Value  | Value  |Value
-----------+--------+--------+--------+--------+--------
  SYSLOCKS |  140   |  160   |  600   |  600   |  454
-----------+--------+--------+--------+--------+--------
  RLE      |  334   |  353   | 5000   | 5000   | 3841
-----------+--------+--------+--------+--------+--------
  RCE      |  303   |  312   | 5000   | 5000   | 3833
-----------+--------+--------+--------+--------+--------
  DPE      |  497   |  517   |  600   |  600   |  588
-----------+--------+--------+--------+--------+--------
```

Last save time

Indicates the date and time that tuning information was last saved. The value is in the UTC timezone.

Parameter Name

Indicates the name of the parameter being tuned.

Lowest HWM

Indicates the lowest high-water mark recorded in the last 32 collection intervals.

Highest HWM

Indicates the highest high-water mark recorded in the last 32 collection intervals.

Current HWM

Indicates the high-water mark for the current time interval.

SYSGEN Value

Indicates the value assigned to the parameter in the system definition when the DC/UCF system was last started.

Current Value

Indicates the value assigned to the parameter when the system was last started.

This value will be assigned to the parameter when the system is next started if it terminated abnormally.

Next Value

Indicates the value that will be assigned to the parameter when the system is next started if it terminated normally.

The following command displays automatic tuning information for only the SYSLOCKS parameter.

```
DCMT D AUTOTUNE SYSLOCKS
*** Display Autotune ***


  AUTOTUNE last save time (time zone: UTC): 2009-12-21-15.06.12.460692

  Parameter name:      SYSLOCKS

  160| __        _
     |_##________#___________
     |#######################_________
     |################################
     |################################
     |################################
     |################################
     |################################
     |################################
    0L#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_#_

  Summary
  -------
  Original SYSGEN value:  600          Highest HWM:        160
  Current value:          600          Lowest HWM:         120
  Next value:             454          Current HWM:        120
```

In addition to the fields shown in a summary display, a detail display includes the following information.

Graph

Shows the high-water mark recorded for the parameter in each of the last 32 time intervals.

The horizontal axis represents time intervals, in oldest to newest order.

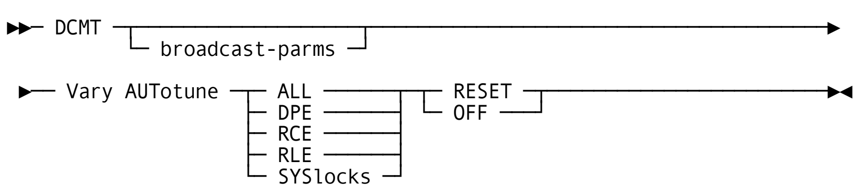
The vertical axis represents HWM values. The scale adjusts to accommodate the range of HWM values being displayed.

DCMT VARY AUTOTUNE Command

Use the DCMT VARY AUTOTUNE command to reset auto-tuning statistics and disable auto-tuning.

DCMT VARY AUTOTUNE Syntax

The following diagram shows the syntax for the new DCMT VARY AUTOTUNE command:



DCMT VARY AUTOTUNE Parameters

This section describes the parameters for the DCMT VARY AUTOTUNE command:

broadcast-parms

Executes the DCMT command on all or a list of data sharing group members.

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

ALL

Varies auto-tuning for all tuned parameters.

DPE

Varies auto-tuning of the DPE count parameter.

RCE

Varies auto-tuning of the RCE count parameter.

RLE

Varies auto-tuning of the RLE count parameter.

SYSlocks

Varies auto-tuning of the SYSLOCKS parameter.

RESET

Resets historical information for the specified parameter so that tuning is based on future values only.

OFF

Disables automatic tuning for the specified parameters. Auto-tuning is disabled only for the current execution of the DC/UCF system.

Note: To permanently disable automatic tuning, you must change your system definition.

Example: Reseting Statistics

The following command resets statistics for auto-tuning SYSLOCKS:

```
DCMT V AUTO SYSLOCKS RESET
   SYSLOCKS  has been reset
```

DCMT DISPLAY CHANGE TRACKING Command Output

Output from the DCMT DISPLAY CHANGE TRACKING command shows the amount of space consumed by automatic tuning.

```
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     36964        5   24%   2009-12-21-12.25.59.212961
Permanent area statuses         0        0    0%   2009-12-21-12.26.00.543069
Journal status overrides        0        0    0%   2009-12-21-12.25.59.234894
Autotune overrides           1400        1    5%   2009-12-21-15.14.15.539878
Control information         30192        4   19%   N/A
--------------------------- ------ ------ ----
Total:                      68556       10   48%

File Name   MirrorStat    MODE  ErrStat  PagSize   PagCnt   Fl-Type     DD-Name
SYSTRK2     ACTIVE        Clos        0     7548       21   non-VSAM    SYSTRK2
  DSname:  DBDC.SYSTEM73.SYSTRK2                    DISP=SHR VOLSER:CULL05
    FORMAT  datetime (time zone: UTC) 2009-12-19-14.06.15.881502
    CONTROL datetime (time zone: UTC) 2009-12-21-12.25.56.579801

SYSTRK1     ACTIVE        Clos        0     7548       21   non-VSAM    SYSTRK1
  DSname:  DBDC.SYSTEM73.SYSTRK1                    DISP=SHR VOLSER:CULL05
    FORMAT  datetime (time zone: UTC) 2009-12-19-14.06.15.853321
    CONTROL datetime (time zone: UTC) 2009-12-21-12.25.56.579801

  Next value:            454           Current HWM:          14
```

SYSLOCKS High Water Mark

CA IDMS now tracks and displays a high-water mark for the SYSLOCKS parameter.

SYSLOCKS is an estimate of the maximum number of database locks that can be held at one time. It is used to determine the amount of storage that the lock manager should initially allocate. Because the amount of storage needed to represent a lock is not a fixed value, the high-water mark for SYSLOCKS is a calculated value based on the high-water mark of lock storage in use. CA IDMS tracks this value and includes it in the output of the DCMT DISPLAY LOCK STATISTICS command and the STATUS display of the LOCKMON command.

The following output from the DCMT DISPLAY LOCK STATISTICS command shows both the current and HWM values for the SYSLOCKS parameter.

```
DCMT D LOCK STAT
*** Transaction Lock Statistics ***
                Local Trans     Local Page    Global Proxy Global Resource
Lock Requests           452              0              0              0
Locks Held               12              0              0              0
                  - - - -  Notify/Longterm Stats - - - -
                       Notify       Longterm Share      Longterm Update
Acquired                    0                   0                   0
Held                        0                   0                   0
                  - - - - - Storage Management - - - - -
SYSLOCKs value:           454
SYSLOCKs HWM:             160
               # Times Ovfl   # Ovfl Getstg  Curr Ovfl Size   Ovfl Size HWM
Overall:                  0              0              0              0
  Session:                0              0              0              0
    Class:                0              0              0              0
 Resource:                0              0              0              0
 XES Reqs:                0              0              0              0
    Proxy:                0              0              0              0
                  - - - - - -  Miscellaneous - - - - - -
Upgrade Reqs:             0    In Place:         0    Denied:          0
Cleanup Calls:            0    Compression Calls:                    0
            PAGE 00001 - NEXT PAGE:
```

The following output from the STATUS display of the LOCKMON task also shows both the current and HWM values for the SYSLOCKS parameter.

```
CA IDMS DB/DC Lock Monitor Version 18.0     Lock Manager Status   Tape: GJI00B
                          Storage Summary

          Startup            Overflow              Total
           242944                                242944

    Times in overflow      Overflow Allocations      Overflow Highwater
              0                    0                         0

                      Notify/Longterm Locks

                    Acquired          Freed          Pending
       Notify:             0              0              0
Longterm Share:                0              0              0
 Longterm Excl:                0              0              0

                          SYSLOCKS

                      Startup    HighWaterMark
                        454           160

CA IDMS DB/DC V73                                 Time:
```

Increased zIIP Utilization

Significant architectural changes have been made to the way in which CA IDMS utilizes zIIP processors. These changes increase the percentage of CPU time spent on zIIP processors while lowering the overhead associated with zIIP usage. As a result, more computing cycles can now be offloaded to zIIPs. This has the potential for increasing transaction throughput while lowering operational costs.

If you have enabled the use of zIIPs, no further action is necessary to take advantage of this feature.

Note: This feature is only available on z/OS.

For more information about enabling the use of zIIP processors for CA IDMS, see the *System Operations Guide.*

zIIP CPU Time Reporting

CA IDMS Version 18.0 includes the following enhancements to zIIP CPU time reporting:

- CPU times are recorded with a higher precision than in prior releases.
- Time spent on a zIIP processor is captured and reported separately from time spent on a CP.

As computer speeds increase, capturing CPU times with a precision of 10000th of a second no longer provides sufficient accuracy for many purposes. To address this issue, CPU times are now captured and recorded in the Time Of Day (TOD) format. The actual usable precision is model dependent, but is never less than microseconds. This precision provides the accuracy needed for meaningful chargeback and performance analysis.

System mode CPU time is now separated into the following categories:

- Time spent on CP
- Time spent on zIIP
- Time spent on CP because zIIP is unavailable

Segregating CPU time into these categories lets you use different chargeback rates for billing purposes. Additionally, you can use the statistics for time spent on CP because no zIIP is available for planning purposes; a high value indicates the need for additional zIIP processors.

A new DC extended statistics section has been added to the statistics control blocks described by the #TSTDS, #STRDS and #TSBDS DSECTs. For upward compatibility, all fields that existed in prior releases remain unchanged. Statistics records written to the log and SMF by CA IDMS, CA IDMS Task Analyzer and CA IDMS Performance Monitor contain the new extension.

Application programs can retrieve the DC extended statistics using the following enhanced DML statements:

- ACCEPT TRANSACTION STATISTICS
- END TRANSACTION STATISTICS
- #TRNSTAT

The following reporting facilities now show times in units of microseconds and, in many cases, also display zIIP time separately:

- DCMT DISPLAY STATISTICS SYSTEM
- DCMT DISPLAY SUBTASK
- Statistics reports 3, 5, 6, 7, 8, 9, 10, 11 and 21
- PRINT LOG
- CA IDMS Performance Monitor report 30
- CA IDMS Log Analyzer program reports

A new OPER WATCH CPU command reports CPU usage for active tasks in microseconds, with automatic scaling to display the most significant digits. For more information, see OPER Enhancements (see page 138).

DC Extended Statistics

A new DC extended statistics section has been introduced to record CPU times in the Time of Day (TOD) format. This section has been added to the following DSECTs:

- #STRDS—task statistics table
- #STRDS—statistics records written to the log
- #STBDS—statistics returned by a #TRNSTAT DML command

DC extended statistics consist of the following fields, as described by the #STBDS DSECT.

```
*                                    * DC Extended stats
*
      DS   2F
TSBBEG  DS   0D            BEGINNING OF DC EXTENDED STATS
TSBSYTI DS   D             System mode TCB CPU time in TOD
TSBCPTI DS   D             System mode zIIP time on CP in TOD
TSBZPTI DS   D             System mode zIIP time in TOD
TSBUSTI DS   D             User mode time in TOD
TSBTTTI DS   D             Total TCB CPU time in TOD
TSBENTI DS   D             Total SRB CPU time in TOD
      DS   14D           Reserved
*
*     Total System mode CPU time is defined as: TSBSYTI + TSBENTI
*
*     TSBTTTI = TSBSYTI + TSBUSTI
*
*     SMF Non-Billable system mode time = TSBCPTI + TSBZPTI
*
TSBXEND  DS   0D             END OF DC EXTENDED STATS
TSBDXLEN EQU  *-TSB          LENGTH OF DSECT.
```

Assembler application programs can retrieve these statistics for the current DC transaction by specifying a new LENGTH parameter on the #TRNSTAT macro.

COBOL and PL/I application programs can retrieve this information by specifying a new LENGTH parameter on an ACCEPT TRANSACTION STATISTICS or END TRANSACTION STATISTICS statement.

The default length for all languages is the length of the statistics returned in prior releases so that existing programs remain upwardly compatible. Existing programs will continue to use the shorter length, even if re-compiled. To retrieve all statistics including the new DC extended statistics, a program must specify a length of 560.

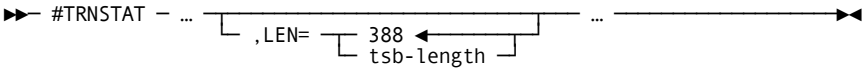
To facilitate use of these statistics in COBOL and PL/I applications, CA IDMS provides a new TRANSACTION-STATISTICS record definition describing the format of the information returned by ACCEPT and END TRANSACTION STATISTICS statements. You add this record description to a dictionary using the DLODPROT source member provided during installation.

#TRNSTAT Assembler DML Statement

The new LENGTH parameter of the #TRNSTAT DML statement enables a program to retrieve extended statistics about task-related activities.

#TRNSTAT Assembler Syntax

The following diagram shows the syntax placement and values for the new LEN parameter.



#TRNSTAT Assembler Parameters

This section describes the new parameter for the #TRNSTAT Assembler statement

LEN=

(#TRNSTAT TYPE=ACCEPT or END requests only)

Specifies the length of the TSB to be returned to the location identified by the RECORD parameter. If LEN= is not specified, the first 388 bytes of the TSB are returned.

tsb-length

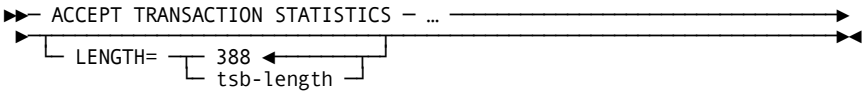
Specifies a register containing the length of the TSB to return; an absolute expression or a half word or full word field containing the length. The length must be an integer with a value of 388 or greater.

ACCEPT TRANSACTION STATISTICS DML Statement

The new LENGTH parameter of the ACCEPT TRANSACTION STATISTICS DML statement enables a program to retrieve extended statistics about task-related activities.

ACCEPT TRANSACTION STATISTICS Syntax

The following diagram shows the syntax placement and values for the new LENGTH parameter.



ACCEPT TRANSACTION STATISTICS Parameters

This section describes the new parameter for the ACCEPT TRANSACTION STATISTICS DML statement.

LENGTH=

Specifies the length of the TSB to be returned to the location identified by the INTO parameter.

tsb-length

Specifies either the symbolic name of a user-defined field that contains the length to be returned or the length expressed as a numeric constant. The length must be an integer with a value of 388 or greater.

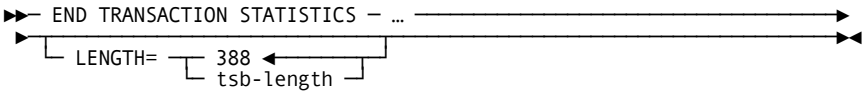
By default, if you do not specify a tsb-length, the first 388 bytes of the TSB are returned.

END TRANSACTION STATISTICS DML Statement

The new LENGTH parameter of the END TRANSACTION STATISTICS DML statement enables a program to retrieve extended statistics about task-related activities

END TRANSACTION STATISTICS Syntax

The following diagram shows the syntax placement and values for the new LENGTH parameter.



END TRANSACTION STATISTICS Parameters

This section describes the new parameter for the END TRANSACTION STATISTICS DML statement.

LENGTH=

Specifies the length of the TSB to be returned to the location identified by the INTO parameter.

tsb-length

Specifies either the symbolic name of a user-defined field that contains the length to be returned, or the length expressed as a numeric constant. The length must be an integer with a value of 388 or greater.

By default, if you do not specify a tsb-length, the first 388 bytes of the TSB are returned.

IDMSINTL User Exit

The IDMSINTL CICS interface now supports an OPTIXIT that allows rerouting of navigational DML request units to different back-end systems or databases. The exit contains information about an IDMS request and its default routing location. The exit may update the routing information to direct it to another location.

To take advantage of this feature, you must write an assembler program and link it with your CICSOPT options module.

When creating an OPTIXIT module for IDMSINTL, consider the following:

- The module must have an entry point of OPTIXIT.
- The module is passed a different parameter list than the OPTIXIT for the IDMSINTC interface. You cannot use the same routine for both.

When called, the exit is passed three parameters. The first and third parameters contain information about the request: the address of the subschema control block and the caller's registers at the time the request was issued, respectively.

The second parameter passed to the exit is the address of an OPTI structure generated either from parameters in the CICSOPT macro or from the SYSCTL file specified in the CICSOPT SYSCTL parameter. To direct the request unit to a different back-end CV, the exit must update the OPTI structure passed in the second parameter.

Registers at entry to exit:

- Register 1 points to a three-word parameter list described next
- Register 13 points to a save area for use by the exit
- Register 14 points to the return location
- Register 15 points to the exit’s entry address

Return codes: None

IDMSINTL User Exit Parameters

The following parameters are passed to the exit:

Fullword 1

Defines the address of the application program's subschema control block (SSC).

Fullword 2

Defines the address of an OPTI structure describing where the request will be routed unless overridden by the exit. (The OPTI structure is described by DSECT #OPIDS.)

Fullword 3

Defines the address of the savearea where the registers at entry to the interface were saved.

Example: OPTIXIT Program

The following example of an OPTIXIT for the IDMSINTL interface forces all requests to be routed to the target system accessed through SVC 226 and CV number 173.

```
        TITLE 'SAMPLE OPTIXIT FOR IDMSINTL'

*---------------------------------------------------------------------
*  SAMPLE OPTIXIT FOR USE WITH VERSION OF IDMSINTL THAT SUPPORTS
*  AN OPTIXIT
*
*  ON ENTRY:
*    R1  --> ADDRESS OF PARM LIST DESCRIBED BY DSECT OPTXPLST
*    R13 --> CALLER'S SAVE AREA
*    R14 --> RETURN ADDRESS
*    R15 --> THIS MODULE'S ENTRY POINT
*
*
* TO INSTALL THIS EXIT, LINK THE ASSEMBLED MODULE WITH IDMSINTL BY
* ADDING AN INCLUDE STATEMENT TO YOUR NORMAL IDMSINTL LINK JOB.
*
*---------------------------------------------------------------------
TESTEXIT CSECT
         ENTRY OPTIXIT
         USING  OPTIXIT,12
         USING  OPTIXIT,12
OPTIXIT  DS    0H
         STM   14,12,12(13)       SAVE CALLER'S REGISTERS
         LR    12,15              SET LOCAL BASE
         LA    15,SAVEAREA        POINT TO MY SAVE AREA
         ST    13,4(,15)          SA FORWARD CHAIN
         ST    15,8(,13)          SA BACKWARD CHAIN
         LR    13,15              SET SAVE AREA POINTER
         USING  OPTXPLST,1
         L     2,OPTXOPTA
         USING  OPI,2
         MVI   OPICVNUM,226       SAMPLE HARD-CODED CV NUMBER
         MVI   OPISVCNO,173       SAMPLE HARD-CODED SVC NUMBER
         L     13,4(,13)          CALLER'S SAVE AREA
         LM    14,12,12(13)       RESTORE REGISTERS
         XR    15,15              CLEAR RETURN CODE
         BR    14                 AND EXIT
         SPACE
SAVEAREA DC    18F'0'
         LTORG
OPTXPLST DSECT
OPTXSSCA DS    A(0)               A(16-CHARACTER SUBSCHEMA-CTRL
OPTXOPTA DS    A(0)               A(OPI DSECT STORAGE)
OPTXSAVA DS    A(0)               A(INTL CALLER'S SAVEAREA)
         COPY  #OPIDS
         END
```

Type Qualifier on Vary Program

The DCMT VARY PROGRAM command was enhanced to let you specify the type of program being varied.

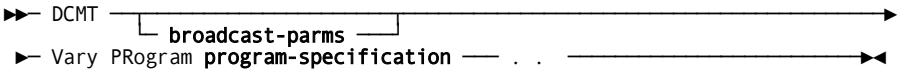
While more than one program can exist with the same name and version, each program has a different type. By specifying the program type on the DCMT VARY PROGRAM command, you can fully identify the program to be varied. This enhancement eliminates the need to issue a prompt for type information, which makes it easier for you to broadcast DCMT VARY PROGRAM commands and include them in UCF batch scripts.

DCMT VARY PROGRAM Statement

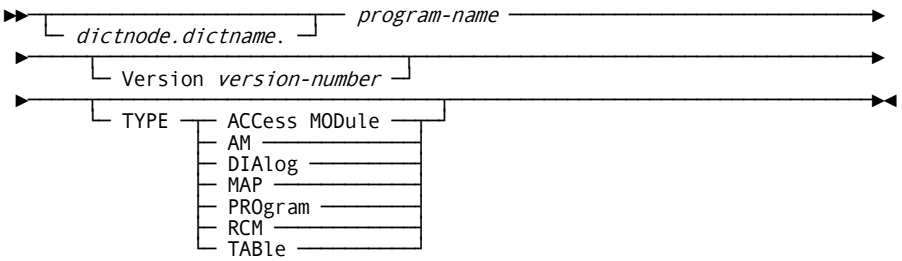
The DCMT VARY PROGRAM command now allows qualification by program type.

DCMT VARY PROGRAM Syntax

The following diagram shows the type qualifier enhancement to the DCMT VARY PROGRAM command:



Expansion of program-specification



DCMT VARY PROGRAM Parameters

This section describes the new parameters that support the type qualifier enhancement to the DCMT VARY MEMORY command.

broadcast-parms

Executes the DCMT command on all or a list of data sharing group members.

For more information on broadcasting and broadcast-parms syntax, refer to the section "How to Broadcast System Tasks" in the *CA IDMS System Tasks and Operator Commands Guide*.

program-specification

Specifies the program to vary.

dictnode

Specifies the DDS node that controls the data dictionary where the named program resides.

If a node name is not specified, the default DDS node established for the session is accessed. If a default DDS node has not been established, the local node is accessed.

dictname

Specifies the alternate data dictionary in which the named program resides.

Note: Although *dictnode* and *dictname* are both optional parameters, if *dictnode* is specified and dictname is not specified, a "." delimiter must be included to represent the missing *dictname* parameter as shown in the following example:

```
DCMT V PROGRAM dictnode..program-name V version-number
```

program-name

The name of a program that has been defined on a system generation PROGRAM statement or previously loaded by the DC/UCF system.

version-number

The version number of the program.

Default: 1

TYPe

The type of the program:

- ACCess Module
- AM
- DIAlog
- MAP
- PROgram
- RCM
- SUBschema
- TABle

Note: When the TYPE keyword appears as part of the program specification it is used to identify a particular program. The TYPE keyword following the DEFINE keyword is used to redefine the type of the program.

OPER Enhancements

Enhancements to the OPER system task include the following new items:

- Scrolling support—enables the display of multiple screens of output.
- WATCH CPU—displays CPU statistics for currently active tasks.
- WATCH DB DBNAME option—displays the DBNAME or segment name being accessed by database sessions.

Scrolling Support

Scrolling support is available for any WATCH command that can generate more than one screen of output. This capability lets you display all generated output.

Scroll through multiple screens of output using the PF7/PF8 function keys or by using the new scrolling subcommands.

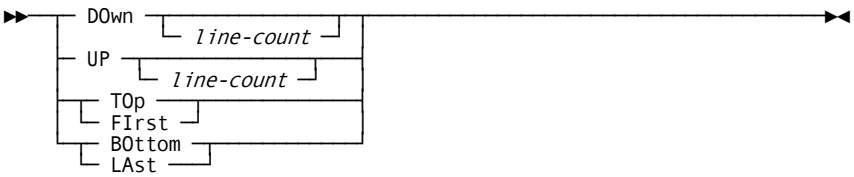
Scrolling Subcommands

Scrolling subcommands let you page through multiple screens of output and are supported for the following OPER commands:

- WATCH ACTIVE TASKS
- WATCH DB
- WATCH LTERM
- WATCH TIME
- WATCH USERS

Scrolling Subcommands Syntax

The following diagram shows the syntax scrolling subcommands for the OPER commands:



Scrolling Subcommands Parameters

This section describes the new scrolling subcommand parameters for the OPER commands:

DOwn

Displays the next set of output.

line-count

Specifies the number of lines of output that is to be skipped relative to the current display position. Replace line-count with a positive number to scroll forward; replace line-count with a negative number to scroll backwards.

Default: The number of detail lines that fit on the screen.

UP

Displays the previous set of output.

line-count

Specifies the number of lines of output that is to be skipped relative to the current display position. Replace line-count with a positive number to scroll backwards; replace line-count with a negative number to scroll forward.

Default: The number of detail lines that fit on the screen.

TOp|FIrst

Positions the display to the first screen of output.

BOttom|LAst

Positions the display to the last screen of output.

Scrolling Subcommands Usage

This section describes how to use the new scrolling subcommands for the OPER commands.

Using PFkeys

You can use PFKEYS in place of scrolling subcommands to page through a set of output:

- PF7 is equivalent to the UP subcommand with no line count.
- PF8 is equivalent to the DOWN subcommand with no line count.

Scrolling Position

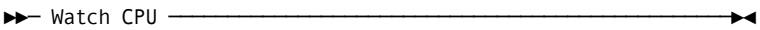
The current position within the set of output being displayed is shown at the top-right corner of the screen.

OPER WATCH CPU

The WATCH CPU command of the OPER system task displays CPU statistics for currently active tasks.

WATCH CPU Syntax

The following diagram shows the syntax for the new WATCH CPU command.



Example: OPER WATCH CPU Output

This example shows the output from the OPER WATCH CPU command:

```
  IDMS-DC Release 1800    Display Active Tasks CPU Time  Line 1    of 21
Task Id   Program  System time  zIIP(CP)time  zIIP(zIIP)t.  User time
0000000054 RHDCOPER  00:00.001052  00:00.000000  00:00.000000  00:00.000049
0000000000 *MASTER*  00:00.542986  00:00.000000  00:00.001658  00:00.000000
0000000001 *DBRC*    00:00.079350  00:00.000000  00:00.008751  00:00.000000
0000000014 UCFLINE   00:00.001007  00:00.000000  00:00.000000  00:00.000000
0000000015 CCILINE   00:00.001821  00:00.000000  00:00.000000  00:00.000000
0000000016 VTAM      00:00.003935  00:00.000000  00:00.000517  00:00.000000
0000000017 DDSVTAM   00:00.017025  00:00.000000  00:00.000032  00:00.000000
0000000018 D0FILINE  00:00.000018  00:00.000000  00:00.000000  00:00.000000
0000000019 TCPIP     00:00.000127  00:00.000000  00:00.000000  00:00.000000
0000000020 TSTTCP1   00:00.000104  00:00.000000  00:00.000000  00:00.000000
0000000021 TSTTCP2   00:00.000011  00:00.000000  00:00.000000  00:00.000000
0000000002 RHDCRUSD  00:00.000102  00:00.000000  00:00.000000  00:00.000000
0000000003 RHDCRUSD  00:00.000905  00:00.000000  00:00.000000  00:00.000000
0000000004 RHDCRUSD  00:00.000168  00:00.000000  00:00.000034  00:00.000000
0000000005 RHDCRUSD  00:00.001272  00:00.000000  00:00.000040  00:00.000000
0000000006 RHDCRUSD  00:00.000894  00:00.000000  00:00.000024  00:00.000000
0000000007 RHDCRUSD  00:00.000756  00:00.000000  00:00.000026  00:00.000000
0000000011 PMONCIOD  00:00.014094  00:00.000000  00:00.000280  00:00.000000
0000000013 RHDCDEAD  00:00.212727  00:00.000660  00:00.129131  00:00.000000
0000000012 PMONCROL  00:00.000582  00:00.000000  00:00.000217  00:00.000000
0000000022 RHDCPRNT  00:00.000064  00:00.000000  00:00.000000  00:00.000000
******************************** LAST PAGE ********************************



IDMS DB/DC V72      - Tasks active:21              Time: 09:02:59
```

Times are displayed in MM:SS:ffffff format unless they exceed 60 minutes. Times over 60 minutes are displayed in HH:MM:SS.ffff format.

The following items describe the columns from the example output:

Task ID

Task thread ID

Program

Name of the current program. If the program name is not available (for example, during task termination processing), then this field is blank.

For ERUS tasks, the program is determined from one of the following sources:

- CICS Task—PROGRAM-NAME in SUBSCHEMA-CTRL at the time of BIND RUN UNIT
- Batch program— Batch program name being executed
- IDMS/DC DDS task—Front-end task code

System Time

The total system mode CPU time consumed by the task.

zIIP(CP)time

The system mode CPU time consumed on a CP because zIIP is unavailable.

zIIP(zIIP)t.

The system mode CPU time consumed on a zIIP

User time

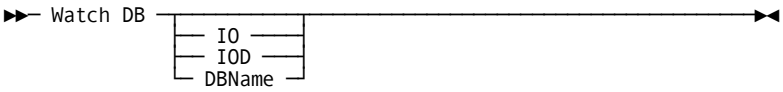
The user mode CPU time consumed by the task.

DBNAME for Database Sessions

The new option DBName on the OPER WATCH DB command shows the database name or segment name being accessed by active database sessions.

WATCH DB Syntax

The following diagram shows the syntax placement of the new database name option for the OPER WATCH DB command:



WATCH DB Parameter

This section describes the new database name parameter for the OPER WATCH DB command.

DBName

Displays the database name or segment name being accessed by active database sessions.

More information

- For more information about database drivers, see the *Database Administration Guide*.
- For more information about database I/O statistics, see the *System Operations Guide*.

  For more information about the scrolling subcommands, see the section Scrolling Subcommands Parameters (see page 139).

Examples: OPER WATCH DB DBNAME Output

This example shows the output from the OPER WATCH DB command:

```
    IDMS-DC Release 1700     Display DB activity          Line 1     of 15
 Task Id   Orig IDMSProg Subschem Pri Sta V# PageRead PageWrit CallIDMS LOCK-Rq
000000010 DBDC RHDCLGSD IDMSNWK9 253   A 56 00000000 00000000 00000003 00000001
0000000009 DBDC RHDCLGSD IDMSNWK9 253   A 56 00000000 00000000 00000003 00000001
0000000008 DBDC RHDCLGSD IDMSNWK9 253   A 56 00000000 00000000 00000003 00000001
0000000007 DBDC RHDCRUAL IDMSSECS 253   A 95 00000000 00000000 00000008 00000001
0000000007 DBDC RHDCRUAL IDMSSECS 253   A 95 00000002 00000000 00000013 00000001
0000000006 DBDC RHDCRUAL IDMSNWK8 253   A 54 00000000 00000000 00000003 00000001
0000000006 DBDC RHDCRUAL IDMSNWK8 253   A 54 00000000 00000000 00000003 00000001
0000000005 DBDC RHDCRUAL IDMSSECU 253   A 54 00000000 00000000 00000004 00000001
0000000005 DBDC RHDCRUAL IDMSSECU 253   A 95 00000002 00000000 00000027 00000001
0000000004 DBDC RHDCRUAL IDMSNWK6 253   A 95 00000000 00000000 00000009 00000001
0000000004 DBDC RHDCRUAL IDMSNWK6 253   A 95 00000039 00000000 00001749 00000001
0000000003 DBDC RHDCRUAL IDMSNWKL 253   A 54 00000000 00000000 00000003 00000001
0000000003 DBDC RHDCRUAL IDMSNWKL 253   A 54 00000000 00000000 00000003 00000001
0000000002 DBDC RHDCRUAL IDMSNWK7 253   A 54 00000000 00000000 00000003 00000001
0000000002 DBDC RHDCRUAL IDMSNWK7 253   H 95 00000015 00000000 00000139 00000040
 ********************************* LAST PAGE **********************************

IDMS DB/DC V72      - Tasks active:31                Time: 09:38:38
```

This example shows the output from the OPER WATCH DB IO command:

```
    IDMS-DC Release 1700    Display DB IO activity       Line 1    of 15
Task Id Org/Ltrm IDMSProg  PagReq  PgRead StIO PgIO  WrtRq  PgWrt JrWrt  R/W
00000010 DBDC    RHDCLGSD     0     0  0  0    0     0    0    0
00000009 DBDC    RHDCLGSD     0     0  0  0    0     0    0    0
00000008 DBDC    RHDCLGSD     0     0  0  0    0     0    0    0
00000007 DBDC    RHDCRUAL     3     0  0  0    0     0    0    0
00000007 DBDC    RHDCRUAL     6     2  0  0    0     0    0    0
00000006 DBDC    RHDCRUAL     0     0  0  0    0     0    0    0
00000006 DBDC    RHDCRUAL     0     0  0  0    0     0    0    0
00000005 DBDC    RHDCRUAL     0     0  0  0    0     0    0    0
00000005 DBDC    RHDCRUAL     5     2  0  0    0     0    0    0
00000004 DBDC    RHDCRUAL     3     0  0  0    0     0    0    0
00000004 DBDC    RHDCRUAL   874    39  0  0    0     0    0    0
00000003 DBDC    RHDCRUAL     0     0  0  0    0     0    0    0
00000003 DBDC    RHDCRUAL     0     0  0  0    0     0    0    0
00000002 DBDC    RHDCRUAL     0     0  0  0    0     0    0    0
00000002 DBDC    RHDCRUAL    51    15  0  0    0     0    0    0
 ********************************* LAST PAGE ***********************************


IDMS DB/DC V72      - Tasks active:31              Time: 09:38:38
```

This example shows the output from the OPER WATCH DB IOD command:

```
    IDMS-DC Release 1700    Display Active DB Drivers     Line 1    of 4
 Driver --------- Area -----------    Start-IO Pg-IOs  Pg/IO PgWrt JrWrt JWai
READ    DBCR.BRNCHTEL                      0     0
READ    DBCR.ACCTHIST                     0     0
READ    CATSYS.DDLCAT                      0     0
READ    CATSYS.DDLCATX                     0     0
 ********************************* LAST PAGE ***********************************


IDMS DB/DC V72      - Tasks active:32              Time: 09:36:04
```

External Sort Control

A new SYSIDMS parameter provides control over how CA IDMS interfaces with your external sort program. This feature lets CA IDMS invoke the external sort in an optimal way, and eliminates the need for tailoring sort control statements generated by CA IDMS utilities.

The new SYSIDMS SORT_FIELD_MAX_LEN parameter specifies the maximum field length that your external sort program accepts.

SORT_FIELD_MAX_LEN=*field-length*

Specifies the maximum sort field length supported by your external sort program.

Default: 256 (z/VSE) and 4092 (other operating systems)

If *field-length* is less than the value accepted by your sort program, CA IDMS may not be able to optimize its use of external sort. If *field-length* is greater than the value accepted by your sort program, an error may occur at runtime.

Note: CA IDMS currently utilizes a maximum sort field size of 2000, so there is no advantage in specifying a field length greater than 2000.

Utility Enhancements

Enhancements to the utilities give you the following capabilities:

- PRINT SPACE output reports page reserve information.
- Control over the number of sorts used by IDMSDBAN.
- Control over the warning threshold when a journal is nearly full during ARCHIVE JOURNAL processing.

PRINT SPACE Page Reserve Reporting

For Version 18.0, the PRINT SPACE utility was enhanced to report on the amount of available space reserved for the expansion of variable-length records. This information is useful when you are planning for database expansions and reorganizations.

The following report shows the output from the enhanced PRINT SPACE utility:

```
PRINT SPACE FOR AREA SYSSQL.DDLCATX FULL;
                          AVAILABLE Space Distribution Report
                                AVAIL       NUMBER
                                SPACE       OF PAGES
AREA      SYSSQL.DDLCATX
 PAGE SIZE        4,276
 PAGE RESERVE       212
   PAGES        10,001 THRU       13,000
                                91-100%           56
                                81-90 %          117
                                71-80 %          125
                                61-70 %          197
                                51-60 %          197
                                41-50 %          229
                                31-40 %          230
                                21-30 %          225
                                11-20 %          203
                                00-10 %        1,419
                                SMPS               2
                                TOTAL          3,000
   FILE   SYSSQL.DDLCATX
   BLOCKS           1 THRU        3,000
Total Space Allocated                     12,828,000
Total Space Available         (Percent)     3,240,368    (25.26%)
Total Reserved Space Available   (Percent)      511,128    ( 3.98%)
Total Unreserved Space Available (Percent)    2,729,240    (21.28%)
Total Space Used                          9,587,632
Logically Full Pages                              0
Total Space Unusable    (Percent)                 0    (  0.00%)
AREA      SYSSQL.DDLCATX          Distribution of USED Space Report
          Maximum                                 Percent of
Record Type  Length      Occurrences   Total Space Used      Total Used
Null Line       8             54              432            0.00
  SR5          48              1               48            0.00
  SR7          40             12              480            0.00
  SR8       1,404         28,567        9,482,184           99.93
Space Inv.  4,244              2            8,488            0.08
Overhead       32          3,000           96,000            0.98
      *** NO logically deleted records found ***
```

The PRINT SPACE utility output contains the following new fields:

PAGE RESERVE

The number of bytes initially reserved on each page for the expansion of variable length records.

Total Reserved Space Available

The total amount of space reserved for the expansion of variable length-records.

Total Unreserved Space Available

The total amount of available space that is not reserved for the expansion of variable length records.

IDMSDBAN Sort Control

With Version 18.0, there is a new SYSIDMS DBAN_SORT_PASSES parameter that specifies the number of sorts to perform. The SYSIDMS parameter gives you control over the number of sorts IDMSDBAN uses when auditing index orphan chains. Adjusting this number may enable IDMSDBAN to detect more errors.

DBAN_SORT_PASSES=*sort-passes*

Specifies the number of sorts that IDMSDBAN uses when auditing index orphan chains.

Default: 1

The higher the number of sorts, the greater the number of errors that can be detected, if any exist. However, increasing the number of sorts also increases the overhead for auditing orphan chains, which can be significant when processing large indexes. Because most errors are detected with a single sort, the default of one is sufficient in most cases. You should increase the number of sorts beyond 1 only when you need to identify every error in an index.

ARCHIVE JOURNAL Warning Threshold

A new SYSIDMS parameter lets you control the threshold that ARCHIVE JOURNAL uses to warn of a nearly full journal file. Increasing the percentage enables earlier detection of transactions that are in a loop or that do not commit often enough.

The new SYSIDMS ARCHIVE_JOURNAL_WARNING_PERCENT parameter specifies the percent to use as a trigger for issuing a warning:

ARCHIVE_JOURNAL_WARNING_PERCENT=*percent-number*

Specifies the threshold percent that ARCHIVE JOURNAL uses to issue a warning message that a journal file is nearly full. If the amount of available space after condensing a journal is less than percent-number, ARCHIVE JOURNAL issues a warning message indicating that the journal is nearly full of condensed segments.

Default: 10 percent

Display All Queues

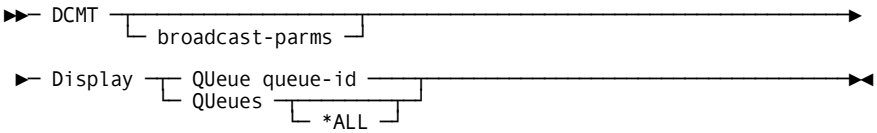
The enhanced DCMT DISPLAY QUEUE command displays all of the queues in the DC/UCF system, including dynamically defined queues. Providing complete information makes it easier to monitor queue activity.

DCMT DISPLAY QUEUE Command

The DCMT DISPLAY QUEUE command has been enhanced to display all queues.

DCMT DISPLAY QUEUE Syntax

The following diagram shows the syntax for the DCMT DISPLAY QUEUE command:



DCMT DISPLAY QUEUE Parameters

This section describes the parameters for the DCMT DISPLAY QUEUE command:

broadcast-parms

Executes the DCMT command on all or a list of data sharing group members.

For more information about broadcasting and broadcast-parms syntax, see *System Tasks and Operator Commands Guide*.

QUeue

Displays information about a specific queue.

queue-id

Identifies the queue to be displayed. queue-id must be the identifier of the queue to be displayed.

QUeues

Displays information about multiple queues.

***ALL**

Displays information about all queues known to the system. This includes both queues defined at system generation time and those defined dynamically.

Note: If *ALL is not specified, information is displayed only for queues defined at system generation time.

Example: Display all Queues

The following example displays all queues defined to the DC/UCF system.

```
 DCMT D QUEUE *ALL
   *** QUEUE DEFINITION TABLE ***
  QUEUE NAME   TASKCODE/INVOKED CURR THRT  MAX  RET GLOBAL
$ADCTEST     1              00004 00000 0000 255 NO
$ADSCIDX                    00001 00000 0000 255 NO
JPDD1                       00003 00000 0000 255 NO
JPDX1                       00003 00000 0000 255 NO
JPDX2                       00002 00000 0000 255 NO
KJMQUE1                     00001 00000 0000 255 NO
KJMQUE2                     00001 00000 0000 255 NO
OLQQNOTE       OLQTNOTE/00000  00000 00001 0000 001 NO
RTSVQ                       00002 00000 0000 255 NO
TASK_ANALYZER_12            00001 00000 0000 255 NO
```

System Definition Enhancements

The SYSGEN SYSTEM statement was enhanced to give you control over more system attributes. You can now perform the following actions:

- Specify default values for queue and message retention periods
- Override the line length for all printers in your system
- Control the number of message buffers used by the debugger

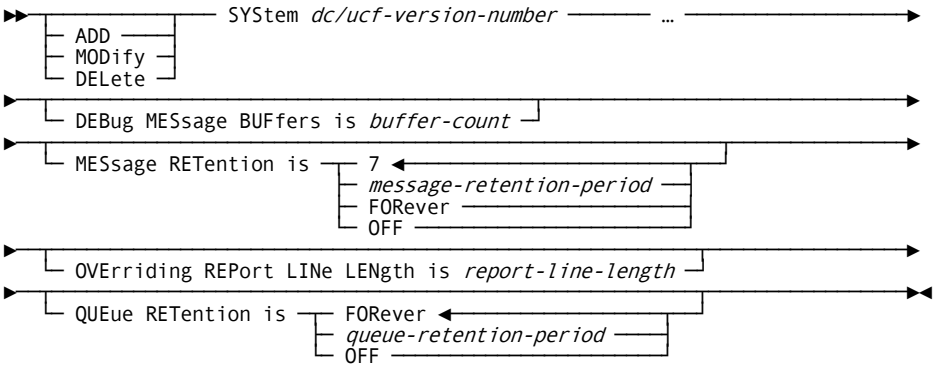
These changes facilitate tailoring a DC/UCF system to meet your operational needs.

SYSGEN SYSTEM Statement for Runtime Options

The SYSGEN SYSTEM statement was enhanced to give you control over more runtime options.

SYSGEN SYSTEM Syntax for Runtime Options

The following diagram shows the syntax enhancements for the system definition SYSTEM statement:



SYSGEN SYSTEM Parameters for Runtime Options

This section describes the new parameters for the system definition for the SYSTEM statement:

DEBug MESsage BUFfers is *buffer-count*

Specifies the number of message buffers used by the CA IDMS online debugger.

Increasing the number of message buffers enables the debugger to display more symbols and list more information. However, the larger the value you specify, the more storage is needed to execute the debugger.

Limits: 5 - 30

Default: 5 (buffers)

MESsage RETention is

Specifies the time period that the system retains messages generated by the SEND command. Messages whose retention period has expired are automatically deleted the next time you start your system.

message-retention-period

Specifies the number of days that messages are retained.

Limits: 0 - 255 (255 is synonymous with FOREVER; 0 is synonymous with 1)

Default: 7 (days)

FORever|OFF

Directs your system to not delete messages based on a retention period. FOREVER and OFF are synonyms that you can use interchangeably.

OVErriding REPort LINe LENgth is report-line-length

Specifies the line length to be used for all reports generated within a DC/UCF system.

Limits: 0 -255.

If this parameter is 0 (zero) or not specified, the line length used is based on the terminal's device type. Because specifying a non-zero value impacts all DC/UCF reports, exercise caution if you set this parameter.

QUEue RETention is

Specifies the default time period that the system retains queues that are created dynamically. This value is used only if no retention period is specified when a queue is created. Queues whose retention period has expired are deleted automatically when the system is next started.

queue-retention-period

Specifies the number of days that queues are to be retained.

Limits: 0 - 255 (255 is synonymous with FOREVER; 0 is synonymous with 1)

FORever|OFF

Directs the system not to delete queues based on a retention period. FOREVER and OFF are synonyms that you can use interchangeably.

Century Validation

You now have more control over how CA IDMS processes dates by specifying these directives:

- The century base year assumed by the DATEDIFF and DATEOFF built-in functions.
- Whether century values should be validated by built-in functions that accept four-digit years and the range of permissible values.

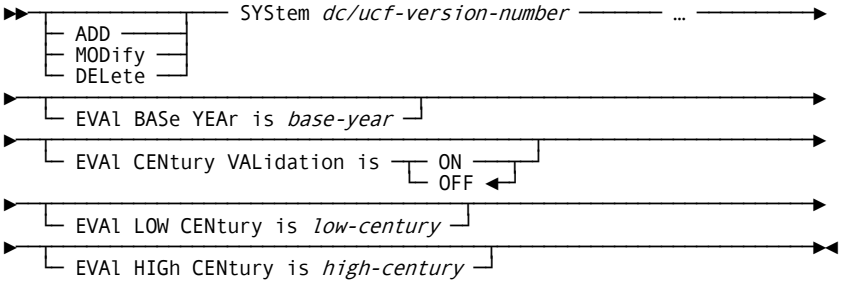
Note: You can specify these directives both as system generation and SYSIDMS parameters.

SYSGEN SYSTEM Statement for Century Validation

New SYSTEM statement parameters permit enhanced control over how dates are processed in a DC/UCF system.

SYSTEM Syntax for Century Validation

The following diagram shows the syntax enhancements to the SYSTEM statement:



SYSTEM Parameters for Century Validation

This section describes the parameters for the century validation enhancement to the SYSTEM statement:

EVAl BASe YEAr is *base-year*

Specifies the base year assumed by the DATEDIFF and DATEOFF built-in functions. The base year is used to determine whether a two-digit year is considered to be in the twentieth or twenty-first centuries. A year whose value is greater than the base year is considered to be in the twentieth century; values less than or equal to the base year are considered to be in the twenty-first century.

Limits: 1- 99

Default: 68 (so that years 00-68 are considered to be in the twenty-first century)

EVAl CENtury VALidation is

Specifies whether century values are validated by built-in functions that accept four-digit years such as GOODDATEX.

ON

Validates century values.

OFF

Does not to validate century values.

Default: OFF

EVAl LOW CENtury is *low-century*

Specifies the lowest century value to be considered valid. This value is used during century validation in built-in functions that accept four-digit years, such as GOODDATEX. Centuries are validated only if the EVAL CENTURY VALIDATION is ON.

Limits: 1 - 99

Default: 19

EVAl HIGh CENtury is *high-century*

Specifies the highest century value to be considered valid. The value specified is used during century validation in built-in functions that accept four-digit years such as GOODDATEX. Centuries are validated only if EVAL CENTURY VALIDATION is ON.

Limits: 1 - 99

Default: 20

SYSIDMS Parameters

The following new SYSIDMS parameters provide enhanced control over date processing in local mode:

EVAL_BASE_YEAR=base-year

Specifies the base year to be assumed by the DATEDIFF and DATEOFF built-in functions. The base year is used to determine whether a two-digit year is considered to be in the twentieth or twenty-first centuries. A year whose value is greater than the base year is considered to be in the twentieth century; values less than or equal to the base year are considered to be in the twenty-first century.

Limits: 1 - 99

Default: 68

EVAL_CENTURY_VALIDATION=ON|OFF

Specifies whether century values are to be validated by built-in functions that accept four-digit years such as GOODDATEX.

ON

Validate century values.

OFF

Does not validate century values.

Default: OFF

EVAL_LOW_CENTURY=low-century

Specifies the lowest century value to be considered valid. The value specified is used during century validation in built-in functions that accept four-digit years such as GOODDATEX. Centuries are validated only if the EVAL CENTURY VALIDATION is ON.

Limits: 1 - 99

Default: 19

EVAL_HIGH_CENTURY=high-century

Specifies the highest century value to be considered valid. The value specified is used during century validation in built-in functions that accept 4-digit years such as GOODDATEX. Centuries are validated only if the EVAL CENTURY VALIDATION is ON.

Limits: 1 - 99

Default: 20

SVC Screening Control

You can now allow additional SVCs to be issued from within a DC/UCF system by specifying the new DISABLE_SVC_SCREEN parameter for SYSIDMS.

SVC screening enables COBOL II, LE COBOL, PL/I and C programs to execute within the DC/UCF address space by substituting a DC service call in place of supported SVCs issued by the high-level language. For example, a GETMAIN is translated to a GET STORAGE request. If an assembler program or an unsupported high-level language function issues an unsupported SVC, it results in a Txxx abend, where xxx is the hexadecimal SVC number. (for example, SVC 44 results in a T02C abend.)

It is recommended not to issue unsupported SVCs from within the DC/UCF address space. Improper use of operating system SVCs can cause severe performance problems, task abends, storage corruption, DC system abends, and even operating system failures.

Wherever possible, unsupported SVCs should be replaced with an equivalent DC service request. If this is not possible, you must either invoke the program in such a way that SVC screening is not in effect, or disable SVC screening for the unsupported SVCs.

SVC screening is only in effect for assembler programs invoked using a high-level language call function. If an assembler program is invoked in some other way, such as through a DC TRANSFER CONTROL verb, SVC screening is not in effect.

To disable SVC screening of unsupported SVCs, use the new SYSIDMS DISABLE_SVC_SCREEN parameter. This parameter should not be used to disable screening of the following SVCs that have DC service equivalents: 4, 5, 6, 7, 8, 9, 10, 11, 13, 19, 20, 24, 35, 48, 56, 60, 109, 120, and 122.

The following SYSIDMS parameter lets you control the screening of SVCs.

DISABLE_SVC_SCREEN=SVC-number

Specifies the number of an SVC for which screening is to be disabled.

Limits: 0–255

Disabling screening of an SVC allows it to be issued by a program executing within the DC/UCF address space.

The DISABLE_SVC_SCREEN parameter can be specified multiple times.

Note: Use this parameter with caution, since issuing unscreened SVCs within the DC/UCF system can degrade performance and result in abends. For more information about SVC screening, see DML Reference Guide for Assembler.

CV Retry Message Routing

The SYSIDMS CVRETRY_MSG_CODES parameter gives you control over the destination where message DC208002 is routed. Message DC208002 is issued when a batch job attempts to communicate with an inactive DC/UCF system. An operator must respond to the message by either indicating that another attempt should be made to communicate with the CA IDMS system or that the job should be canceled.

Control over the routing of the DC208002 message facilitates the use of data center automation tools to respond to the message without operator intervention.

The following SYSIDMS parameter lets you control the routing of the DC208002 message.

CVRETRY_MSG_CODES=*descriptor-route-codes*

Specifies the descriptor and route codes to be used for batch message DC208002 (CV cv-number NOT ACTIVE. REPLY RETRY OR CANCEL). descriptor-route-codes must be an eight-digit hexadecimal value.

The first four digits of descriptor-route-codes represent the descriptor codes and the last four digits represent the route codes. Each bit within the descriptor or route codes represents a code value. The first bit (x'8000') represents code value 1 and the last bit (x'0001') represents code value 16. Multiple bits can be on in each set of codes so that x'8101' represents code values 1, 8 and 16.

Default: 00004000 – representing descriptor code zero (0) and route code two.

Note: The descriptor-route-codes value is automatically considered to be in hexadecimal format.

CA ADS Enhancements

This release of CA IDMS contains the following CA ADS enhancements:

- You can manage built-in functions more easily.
- You can specify the character to be used as a comment delimiter.

Improved BIF Management

CA ADS built-in functions are now managed in a way that provides better run time performance while simplifying product installation and maintenance.

In previous releases, you could improve run time performance by linking selected BIF modules with ADSOMAIN, which avoided the overhead of loading a module on each function invocation. The benefit was limited, however, because not all BIF modules could be linked with ADSOMAIN. Furthermore, you had to take extra steps each time the product was upgraded or maintenance was applied to ADSOMAIN or any of the BIF modules linked with it.

In Version 18.0, CA ADS manages BIFs to overcome these shortcomings. All CA-supplied BIF modules are linked with ADSOMAIN and all user-written BIF modules can optionally be linked together as a new ADSOVCON module. These changes have the following benefits:

- Reduced overhead for invoking CA-supplied BIFs because no additional modules are loaded at runtime.
- Reduced overhead for invoking user-written BIFs by creating an ADSOVCON module.
- Simplified product installation and maintenance because there is no need to relink a tailored ADSOMAIN when performing these activities.

Note: Creating an ADSOVCON module is not required. CA ADS dynamically loads user-written BIFs if they are not linked with ADSOVCON.

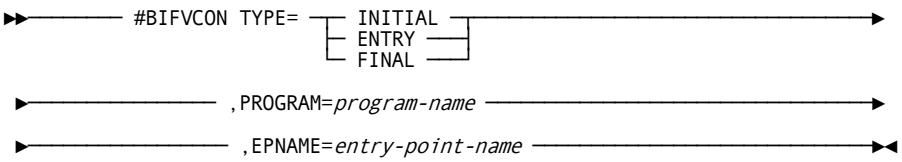
ADSOVCON Module Creation

Optionally create an ADSOVCON module using the #BIFVCON macro to identify your user-written BIF modules to be linked together. The following sample ADSOVCON module indicates that the UDATE and UCHECK BIF modules are to be linked with ADSOVCON.

```
#BIFVCON TYPE=INITIAL
#BIFVCON TYPE=ENTRY,PROGRAM=UDATE,EPNAME=UDATE
#BIFVCON TYPE=ENTRY,PROGRAM=UCHECK,EPNAME=UCHKEP1
#BIFVCON TYPE=FINAL
```

To create an ADSOVCON module, create a source member as described in the following section and save it in your custom source library. Then assemble and link it into your custom load library.

The following diagram shows the syntax for the #BIFVCON macro:



#BIFVCON Macro Parameters

This section describes the parameters for for the #BIFVCON macro.

TYPE

Indicates the type of BIFVCON statement being generated.

INITIAL

Identifies the first BIFVCON statement in the program.

ENTRY

Identifies a BIFVCON statement defining a user-written BIF module.

FINAL

Identifies the last BIFVCON statement in the program.

PROGAM=*program-name*

Identifies the name of a user-written BIF module to be linked with ADSOVCON. *program-name* must be the same as the program name associated with a built-in function declared in your RHDCEVBF module.

This parameter is valid only if TYPE=ENTRY is coded.

EPNAME=*entry-point-name*

Identifies the name of the entry point of the user-written BIF module. *entry-point-name* must be the name of the entry point in the program identified by program-name.

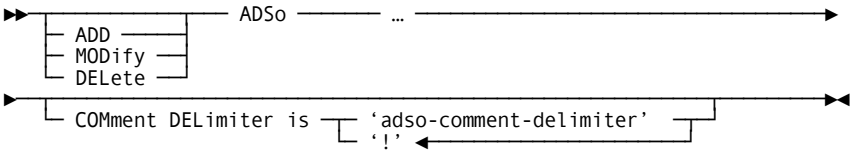
This parameter is valid only if TYPE=ENTRY is coded.

ADS Comment Delimiter

The ADSO SYSGEN statement has been enhanced to permit specifying the character to be used as a comment delimiter in the CA ADS process language.

ADSO SYSGEN Syntax

The following diagram shows the enhanced syntax for the ADSO SYSGEN statement:



ADSO SYSGEN Parameters

This section describes the new parameter for the ADSO SYSGEN statement:

COMment DELimiter is 'adso-comment-delimiter'

Specifies the character to be used as the comment delimiter. adso-comment-delimiter must be a single character enclosed in single quotes.

Default: An exclamation mark ('!')

CA Culprit for CA IDMS Enhancements

The enhanced CA Culprit for CA IDMS profile parameters and options let you do the following:

- Dump data using a DBCS-compatible character set
- Specify how social security numbers are to be formatted
- Suppress generating the record count message
- Specify the base year for determining centuries in CULLUS12
- Separate reports from other output, such as diagnostic messages

DBCS-Compatible Dumps

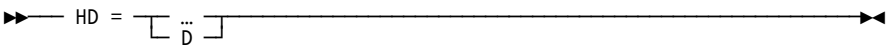
You can direct Culprit's error handling routine to produce DBCS- compatible dumps either as a site default using the Hexadecimal Dump profile option, or as an execution override using the HD profile parameter.

Hexadecimal Dump Option

The HD profile option specifies the default format to be used for hexadecimal dumps issued by the extended error handling facility. This option was enhanced to request DBCS-compatible dumps.

HD Profile Option Syntax

The following diagram shows the syntax for the enhanced HD profile option.



HD Profile Option Parameters

This section describes the enhanced hexadecimal dump profile option:

HD =

Specifies the default format to be used for hexadecimal dumps.

D

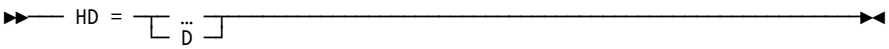
Generates dumps in a vertical format using a DBCS character set.

HD Profile Parameter

The HD profile parameter specifies the format to be used for hexadecimal dumps issued by the extended error handling facility.This option was enhanced to request DBCS-compatible dumps.

HD Profile Parameter Syntax

The following diagram shows the syntax for the enhanced HD profile parameter.



HD Profile Parameter Parameters

This section describes the enhanced hexadecimal dump profile option:

HD =

Specifies the format to be used for hexadecimal dumps.

D

Generates dumps in a vertical format using a DBCS character set.

Social Security Number Format

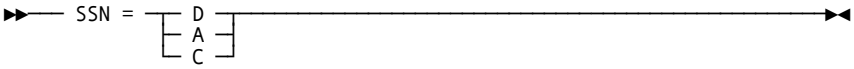
You can specify how Culprit is to format social security numbers in one of the following ways:

- As a site default using the Social Security Number profile option
- As an execution override using the SSN profile parameter

Social Security Number Option

The new SSN profile option specifies the default format for social security numbers when using an FS edit mask.

The following diagram shows the syntax for the new SSN option:



SSN Option Parameters

This section describes the parameters that support the SSN option parameters.

SSN =

Specifies the default format in which social security numbers will appear when using an FS edit mask.

D

Formats social security numbers using the date stamp (DS) option. This is provided for upward compatibility.

A

Formats social security numbers in American format: nnn-nn-nnnn.

C

Formats social security numbers in Canadian format: nnn-nnn-nnn.

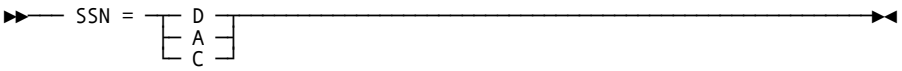
Default: D

SSN Profile Parameter

The new SSN profile parameter specifies the format for social security numbers when using an FS edit mask.

SSN Profile Parameter Syntax

The following diagram shows the syntax for the new SSN profile parameter:



SSN Profile Parameter Parameters

This section describes the parameters for the SSN profile option:

SSN =

Specifies the format for social security numbers when using an FS edit mask.

D

Formats social security numbers as specified by the date stamp (DS) option. This is provided for upward compatibility.

A

Formats social security numbers in American format: nnn-nn-nnn.

C

Formats social security numbers in Canadian format: nnn-nnn-nnn.

Record Count Suppression

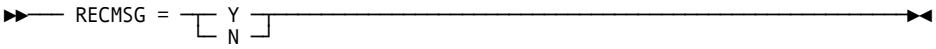
This release introduces a new profile option that allows you to set your site default for whether CA Culprit generates the Records Written message as part of the report output.

Records Written Message Option

The new RECMSG profile option specifies whether by default CA Culprit generates or suppresses message C750009 Records Written for Report.

RECMSG Profile Option Syntax

The following diagram shows the syntax for the new RECMSG profile option.



RECMSG Profile Option Parameters

This section describes the parameters for the RECMSG profile option.

RECMSG =

Sets the default for generating or suppressing message C750009, Records Written for Report.

Y

Issues message C750009.

N

Supresses message C750009.

Default: Y

Century Base Year

You can use the new CULLUS12 Default Year profile option to establish the base year that CULLUS12 uses to determine the century associated with a two-digit year.

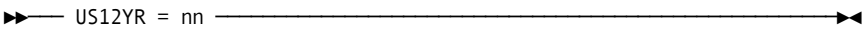
To specify the year that CULLUS12 uses to determine the century associated with a two digit year, use the US12YR profile option.

CULLUS12 Default Century Option

The new US12YR profile option specifies the base year for determining the century associated with a 2-digit year.

US12YR Syntax

The following diagram shows the syntax for the new US12YR profile option.



Accessibility Features

This section describes the US12YR profile option parameter.

US12YR = nn

Specifies the lowest year to be considered in the 21st century.

When using an output format code that requires a century (ccyy) and the input date does not include century, CULLUS12 determines the century value to be 20 if the year is greater than or equal to nn and 19 if the year is less than nn.

Default: 40

Report Separation

In z/OS, you can now separate reports from other output generated by Culprit, such as diagnostic messages, by using the new Output DDNAME Override profile option. This option changes the DDNAME used for the report output when using one-step JCL. You can use this option to segregate the report output from the listings and diagnostic messages.

Sample one-step JCL is shown next:

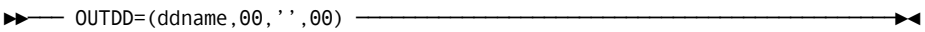
OUTDD=(ddname,00,",00)

Output DDNAME Override Option

The new OUTDD profile option specifies the DDNAME used for report output when using one-step JCL.

DDNAME Syntax

The following syntax shows the new OUTDD profile option.



DDNAME Parameter

This section describes the Output DDNAME Override option parameter.

OUTDD=(ddname,00,'',00)

Specifies the DDNAME to which CULE writes report output when using one-step JCL.

Default: SYS004

All reports in the same Culprit run are written to the file identified by ddname unless you specify a special file-type indicator (PS, IS, or NS) on the report's output card.

Note: If you specify this parameter, a new DD statement must be added to the execution JCL that describes the file. The DD statement must specify the file's block size; otherwise, the job will abend with an s013.

CA OPS/MVS Integration for z/OS

CA IDMS provides internal communications with the CA OPS/MVS Application Program Interface (API), employing START/UP/STOP/DOWN state notifications, heartbeats, and passes specific messages into the OPSLOG facility. This helps ease the automation of the production copies of CA IDMS.

State Reporting

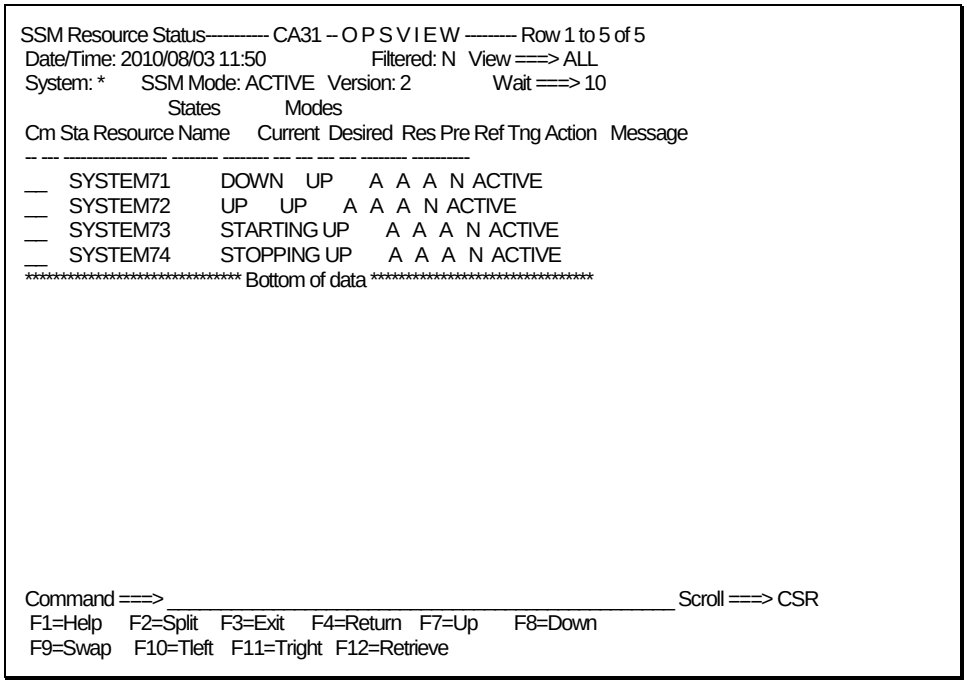
CA IDMS reports its current state information to the System State Manager (SSM) component of CA OPS/MVS via the OPS/MVS generic event API. This state can be forwarded to other Common Service components such as the MM Status Monitor. CA OPS/MVS generates these messages into the JES log whenever the status of the CA IDMS system changes, as in the following example:

```
OPSQNOTIFY CASTATE API received for <jobname> with STARTING , Version R18.0 and Level BAT  nnn
a=008E
OPO1370H <jobname> X'0000' X'0000' X'0200' NONE  300 OPSLOGSV   CASTATE <jobname>
applid:CAIDMS version:R18.0 level:BAT  nnn

OPSQNOTIFY CASTATE API received for <jobname> with UP , Version R18.0 and Level BAT  nnn a=008E
OPO1370H <jobname> X'0000' X'0000' X'0200' NONE  300 OPSLOGSV   CASTATE <jobname>
applid:CAIDMS version:R18.0 level:BAT  nnn
```

OPSQNOTIFY CASTATE API received for <jobname> with STOPPING , Version R18.0 and Level BAT nnn a=008E
OPO1370H <jobname> X'0000' X'0000' X'0200' NONE 300 OPSLOGSV CASTATE <jobname>
applid:CAIDMS version:R18.0 level:BAT nnn

OPSQNOTIFY CASTATE API received for <jobname> with DOWN , Version R18.0 and Level BAT nnn a=008E
OPO1370H <jobname> X'0000' X'0000' X'0200' NONE 300 OPSLOGSV CASTATE <jobname>
applid:CAIDMS version:R18.0 level:BAT nnn

The CA OPS/MVS OPSVIEW facility is able to manage and display System State Manager resources and its states:



Heartbeats

Heartbeats are used by applications to report that they are *alive,* their current processing status, and the reason for that status. This status can be forwarded to other Common Service components such as the MM Status Monitor and the Alert Monitor. CA IDMS currently reports a *NORMAL* status to indicate that the IDMS system is *alive* every five seconds. CA OPS/MVS generates messages whenever a change in status is detected, as follows:

OPO1370H <jobname> X'0000' X'0000' X'0200' NONE 300 OPSONOTIFY CAHEARTBT received for <jobname> applid:CAIDMS version:R18.0
OPSONOTIFY CAHEARTBT received for <jobname> applid:CAIDMS version:R18.0 level:BAT nnn status:NORMAL reason:ACTIVE

Heartbeat messages only appear in the log when there is a change. For example, when a product is no longer communicating a NORMAL state or no longer communicating at the expected interval. Likewise, when a problem does occur, the error is only reported in the log once. No other messages are received until the heart beats have been resumed.

Message Handling

CA IDMS employs OPS/MVS API calls to pass log, trace, and journal dump requests into OPS/MVS OPSLOG and to the OPS/MVS rules engine. CA IDMS currently pass these messages into OPSLOG:

- DC050001 – Log % full message?
- DC050004 – Log is FULL message?
- DC205003 – Journal message?
- DC050024 – TRC % FULL message?
- DC050027 – TRC is FULL message?

If any of these messages are encountered, an OPS/MVS API event is generated. You can then write your own API rule to take a specific action. See the Appendix Sample OPS/MVS API rule (see page 181) for a working example.

CA IDMS passes the event name to the OPS/MVS rules engine based on following rules:

CAIDMS*xxxn*

xxx can be

- LOG for DC050001,DC050004 messages
- JNL for DC205003 messages
- TRC for DC050024,DC050027 messages

n can be:

- A numeric value in range of 0 – 9 which means the LOG/TRC is 0 – 99% full in 10% increments, e.g. CAIDMSLOG8 means an event has been generated that the log is about 80% - 89% full.
- A “F” letter which means the log or TRC is full. For example CAIDMSTRCF.

Implementation CA OPS/MVS API rules

CA OPS/MVS Automated Operations Facility (AOF) can take an action in response to various types of system events. CA IDMS exploits API events to enable easy implementation of AOF rules which permit automation of the production copies of CA IDMS.

Notes:

- For information on how to implement CA OPS/MVS rules, refer to the *CA OPS/MVS® Event Management and Automation - AOF Rules User Guide.*
- For information on how to automate the offloading / archiving processes for DC log, DC traces and journals on CA IDMS, see the Appendix Sample OPS/MVS API rule (see page 181).

Chapter 8: CA IDMS Tools

This chapter describes enhancements to the CA IDMS Tools products.

This section contains the following topics:

Installation and Configuration Enhancements

Several changes to CA IDMS Tools make installation and maintenance easier. These changes separate user generated code from CA-distributed modules so you can apply maintenance without having to redo your customization.

The following changes make installation and maintenance easier:

- You now link the CA IDMS DML/Online (DMLO) user exit program as a separate load module. DMLO dynamically loads USDMLXIT during initial startup. For more information about implementing a DMLO user exit, see the DML Online User Guide.
- The CICS version of DMLO is now linked so that changes due to maintenance or a release upgrade are unlikely. For more information, see DMLO in a CICS Front-End (see page 168).
- You now link the CA IDMS/DC Sort parameter module (TPSPARM) as a separate load module since it is loaded dynamically at runtime. For more information about generating a TPSPARM parameter module, see the DC Sort User Guide.
- You now link the CA IDMS Dictionary Migrator Assistant parameter module (XDMTPARM) as a separate load module, since it is loaded dynamically at runtime. For more information about creating a CA IDMS Dictionary Migrator Assistant parameter module, see comments in the installed version of the XDMTPARM module source.

- You no longer link your IDMS CALC exit program (IDMSCLCX) with GSSCALC. The tools products now invoke the CALC exit through the named user exit facility that is new in CA IDMS Version 18.0.
- A set of CSD definitions are provided for the CICS versions of CA IDMS/DC Sort and DMLO. These definitions reside in the installed source library members TPSCICS and USDCICS.
- Use of CA IDMS Task Analyzer, CA IDMS Masterkey and CA ADS Alive no longer require linking CA-supplied exit routines with RHDCUXIT. For more information, see Simplified Tools Exit Management. For more information, see Simplified Tools Exit Management (see page 169).
- Use of CA IDMS Enforcer no longer requires linking special versions of the CA IDMS compilers. Instead, CA IDMS Enforcer is now invoked based on the presence of certain modules in your runtime library concatenation.For more information, see CA IDMS Enforcer Enablement (see page 170).

DMLO in a CICS Front-End

The CICS front-end version of DMLO now consists of two components:

- A module that is created by linking an IDMSCINT interface module with the CA-supplied DMLO stub module USDTPIF5.
- A new site-independent module USDTPIFC containing only CA-supplied code.

Since user-generated code is linked only with the stub module, CICS DMLO will typically not need to be regenerated as a result of applying maintenance.

Before using the Version 18.0 runtime libraries in your CICS system, you must relink any modules that include USDTPIF5. The following shows the link-edit instructions.

Note: For more information about executing DMLO under CICS, see Implementing CA IDMS DMLO in Multiple CVs Under CICS in the *Installation Guide*.

z/OS Link

To link DMLO for a CICS environment in z/OS, execute the z/OS Link-Edit JCL inserting the following binder statements.

```
ORDER   DFHEAI,USDTPIF5
INCLUDE CAGJLOAD(USDTPIF5)
INCLUDE CUSTLIB(idmscint)
ENTRY   DFHEAI
SETOPT PARM(AMODE=31,REUS=SERIAL,RMODE=ANY)
NAME usdtpifx(R)
```

idmscint

Specifies the name of your IDMSCINT program that interfaces with the target CA IDMS CV.

usdtpifx

Specifies the name of the program that your CICS DMLO transaction invokes.

Note: Do not use the names USDTPIF2, USDTPIF3, or USDTPIFC because these module names are reserved for CA-distributed programs.

Simplified Tools Exit Management

You no longer need to link CA-supplied exits with your RHDCUXIT module. All tools exit routines are loaded and managed independently from RHDCUXIT. This means you no longer need to relink your RHDCUXIT module if maintenance or release upgrades change one of the tools exit routines.

Another benefit of the simplified tools exit management in Version 18.0 is that you no longer need to tailor a USFUEXT module to identify exits to be invoked after the tools exits. Instead, you define your exits using the normal facilities for identifying numbered exits just as if no tools exits were involved. For information on how to define numbered exits, see the System Operations Guide.

As a result of this enhancement, you may need to change the startup JCL for one or more CVs:

- CA IDMS Task Analyzer users must specify a SYSIDMS parameter of TASK_ANALYZER_EXITS=ON in any CV in which CA IDMS Task Analyzer is to be used.
- CA IDMS Masterkey users must specify a SYSIDMS parameter of MASTERKEY_EXITS=ON in any CV in which CA IDMS Masterkey is to be used.

New SYSIDMS Parameters

New SYSIDMS parameters were introduced to indicate which CA IDMS tools exits you want to enable.

MASTERKEY_EXITS=ON|OFF

ON

Enables the CA IDMS Masterkey exits. ON must be specified if CA IDMS Masterkey is to be used in a CV.

OFF

Disables the CA IDMS Masterkey exits.

Default: OFF

TASK_ANALYZER_EXITS=ON|OFF

Controls whether CA IDMS Task Analyzer exits are enabled.

ON

Enables the CA IDMS Task Analyzer exits. ON is required if CA IDMS Task Analyzer is to be used in a CV.

OFF

Disables the CA IDMS Task Analyzer exits.

Default: OFF

CA IDMS Enforcer Enablement

The way you enable the use of CA IDMS Enforcer was enhanced to eliminate the need for linking special versions of the compilers. CA IDMS Enforcer is invoked based on the presence of the following modules in your STEPLIB or CDMSLIB library concatenation at runtime:

- ENFRXITO with an alias of ENFROXIT enables the use of CA IDMS Enforcer in an online environment
- ENFRXITB with an alias of ENFRBXIT enables the use of CA IDMS Enforcer in a batch environment

Note: If you indicate that Enforcer is to be used, these modules are placed in a special Enforcer load library during configuration.

CA ADS Alive Enhancements

The Version 18.0 enhancements to CA ADS Alive allow you to do the following tasks:

- Display more information when reviewing abends.
- Enable dialog animation to proceed despite the issuance of certain error messages.
- Use national characters in record names.

Enhanced Diagnostic Information

The QREVIEW task's diagnostic screen displays information about abends that occur while executing CA ADS applications. This screen now shows the name of the dictionary that was current at the time of execution. Having this information displayed may be useful for problem diagnosis in a multiple dictionary environment.

Error Tolerance

In Version 18.0, you can direct CA ADS Alive to allow dialog animation to proceed even if changes were made to a dialog after it was last generated. This ability avoids the need to regenerate the dialog after making non-structural changes, such as altering its user or class/attribute associations.

If CA ADS Alive detects that changes have been made to dialog process modules since the dialog was last generated, it issues one of the following error messages:

- USG0012E SOURCE OF DIALOG HAS CHANGED – REGENERATION REQUIRED
- USG0018E PROCESS <process name> VERSION n DATE/TIME CONFLICT

If these messages were issued in prior releases, dialog animation was suppressed. In Version 18.0, you can override this behavior using the new SKPUG18 parameter of the USGCPARM macro that is used to generate the USGTPARM options module. The default (N) for the SKPUG18 parameter suppresses animation if these messages are issued. You can change the value for SKPUG18 to Y to allow animation to proceed even if the messages are issued.

If the value of the SKPUG18 parameter is Y and CA ADS Alive detects that changes have been made since the dialog was last generated, the following warning message is issued to inform you of this condition:

- USG0072W SOURCE CHANGED SINCE DIALOG GENERATED – OVERRIDE SPECIFIED

You can then choose whether to continue with the animation or not.

Note: For more information about creating a tailored USGTPARM module, see comments in the installed version of the module.

Tailored Character Set

You can now define the set of characters that are allowed in record names. This ability avoids record names being flagged as invalid if they contain national characters.

The CHARTAB table in the USGTPARM options module specifies the set of allowable characters for record names. Each one-byte entry in the table corresponds to a hexadecimal value from x'00' to x'FF'. An entry value of x'00' indicates that the corresponding hexadecimal value represents a valid character. An entry value of x'01' indicates that the corresponding hexadecimal value does not represent a valid character.

Note: For more information about creating a tailored USGTPARM module, see comments in the installed version of the module.

CA IDMS DML Online Enhancements

CA IDMS DML Online (DMLO) has been enhanced to allow you to do the following tasks:

- Display sets associated with a given record.
- Return customized messages after invoking a user exit.
- Support larger subschemas.
- Provide a maintenance-independent means for linking the CICS front-end version of DMLO.

Set Information Display for a Record

DMLO now displays set information for a given record. If you type an X beside a record on the record display screen, DMLO lists all sets in which the record participates as a member. This enhancement makes it easier to navigate through the database.

Customizable Message Text

You can now customize the message text generated as a result of a non-zero return code that is set by the USDMLXIT user exit.

If your user exit sets a return code value between 1 and 9, you can tailor the text of the F880X message displayed to the user by DMLO to indicate the reason for the exception. The #MSG88001 table in the USDTPARM parameter module defines the message text to be displayed. To customize the messages, update the #MSG88001 table and regenerate your USDTPARM module.

For more information about creating a tailored USDTPARM module, see the *DML Online User Guide*.

Large Subschema Support

You can now use larger subschemas with DMLO by increasing the size of the runtime stack.

To increase the size of the stack, change the value of the STKSIZE parameter of the USDCPARM macro and regenerate your USDTPARM options module. The maximum size you can set for the STKSIZE parameter is16K (16384).

Default: 4K (4096)

For more information about creating a tailored USDTPARM module, see the *DML Online User Guide*.

CA IDMS Masterkey Customizable PFKEY Defaults

CA IDMS Masterkey was enhanced to give you customizable default PFKEY settings. The new #PFKTBL table in the SSKTPARM module defines the default PFKEY settings. Each entry in this table defines the command executed in response to a function or attention key.

Commands are specified as character strings.

Limits: 1 - 69 characters

After updating the table, regenerate your SSKTPARM module.

For these changes to take effect, you might need to delete existing user and terminal PFKEY settings.

To delete existing user and terminal PFKEY settings

1. Execute the DCMT DISPLAY QUEUE command.
2. Look for queues with identifiers of: KBOSSPRFuser-id/lterm-id.

   Note: You must delete each of these queues using a DCMT VARY QUEUE <id> DELETE command before the new defaults are applied to the users or terminals for which the queues exist.

For more information about creating a tailored SSKTPARM module, see the comments available in the installed version of this module.

CA IDMS Dictionary Migrator Enhancements

With Version 18.0, the CA IDMS Dictionary Migrator gives you additional controls over how entities are migrated. With these controls, you can eliminate duplicate edit and code table values, and force the use of the MODIFY verb when doing a NONCHANGEONLY export. The following new parameters provide the additional controls for the Dictionary Migrator:

- TABNULL—eliminates the duplication of values when migrating edit and code tables.
- EXPMOD—controls which verb is generated

Edit/Code Table Handling with the TABNULL Parameter

You can eliminate the duplication of values when migrating edit and code tables by using the new TABNULL parameter of the USMCPARM macro and regenerating your USMTPARM options module.

Setting the value of the TABNULL parameter to Y (Yes) adds the EXCLUDE VALUES NULL clause to edit and code table syntax. Using this clause removes existing table values before the new set of values are inserted. Not removing existing table values may duplicate the values if tables already exist in the target dictionary.

For more information about creating a tailored USMTPARM module, see the comments available in the installed version of your module.

Verb Control with the EXPMOD Parameter

You can now direct the CA IDMS Dictionary Migrator to generate MODIFY syntax instead of ADD syntax when doing an EXPORT CHANGEONLY execution. This ability is helpful if the records being migrated already exist in the target dictionary. Using MODIFY removes existing elements from the record structure before adding the new set of elements. Existing elements are not removed when using an ADD verb, so record elements may be replicated.

The new EXPMOD parameter of the USMCPARM macro controls which verb is generated. By setting the value of this parameter to Y (Yes) and regenerating your USMTPARM options module, the Dictionary Migrator generates MODIFY instead of ADD syntax when executing an EXPORT CHANGEONLY execution.

For more information about creating a tailored USMTPARM module, see the comments available in the installed version of your module.

CA IDMS Database Extractor Alternate DMCL Support

You can now direct the CA IDMS Database Extractor to use the DMCL identified in the SYSIDMS parameter file during load processing. This ability makes it easier to load data from one database to another database that is controlled by a different central version than the original.

Use the new SYSIDMS_DMCL parameter of the USVCPARM macro in your USVTPARM options module to control which DMCL to use. When this parameter value is set to Y (Yes), CA IDMS Database Extractor uses the DMCL identified in the SYSIDMS parameter file during load processing. If the value of this parameter is set to its default of N (No), the DMCL associated with the source subschema is used during load processing.

For more information about creating a tailored USVTPARM module, see the *Database Extractor User Guide*.

CA IDMS Journal Analyzer Run Unit Exclusion

With Version 18.0 you can exclude specific run units from being processed by the Journal Analyzer. This ability avoids skewing results with statistics for programs that are not part of your normal workload. For example, you can exclude statistics for CA IDMS DML/Online (DMLO) if that is not typically used within your DC/UCF system.

Use the new BYPASS parameter statement to exclude run units bound by a specific program. You can include up to ten BYPASS statements in one execution of the Journal Analyzer.

For more information about parameters for CA IDMS Journal Analyzer, see the *Journal Analyzer Users Guide*.

Example: Run Unit Exclusion

This example shows the parameter statements for excluding run units for the system generation compiler (RHDCSGEN) and DMLO (USDMAIN0) in a typical execution of CA IDMS Journal Analyzer.

```
PROCESS=ALL,CONT=N,FORMAT=SPARSE
REPORT=CHRONO,ALL=Y
DISPLAY=PROG,ALL=Y
BYPASS=USDMAIN0
BYPASS=RHDCSGEN
```

CA IDMS Log Analyzer Extended Statistics Support

The summary and detail program reports produced by CA IDMS Log Analyzer have been enhanced to show the new extended CPU statistics, as shown in the following example:

```
ID      RELEASE         CA IDMS LOG ANALYZER            DATE      TIME
        18.0            PROGRAM REPORT            3/02/10    :47:55
             SUMMARY FOR PROGRAM = IDMSBCF  BATCH
              2/16/10  13:35 -  2/16/10  13:35
RUN UNITS--TOTAL                    1
                           MEAN     ACCUMULATED   % OF SYSTEM
                           VALUE        VALUE    OCCURRENCES
COUNTS----PAGES READ                 59.00          59        70.24
        PAGES WRITTEN              51.00          51        77.27
        PAGES REQUESTED           775.00         775        58.98
        CALC RCDS ON HOME PAGE      5.00           5       100.00
        CALC RCDS OVERFLOW          2.00           2       100.00
        VIA RCDS ON OWNER PAGE     24.00          24       100.00
        VIA RCDS OVERFLOW           9.00           9       100.00
        RECORDS REQUESTED       1,151.00       1,151        67.07
        RECORDS BECOMING CURRENT  146.00         146        57.03
        CALLS TO IDMSDBMS         530.00         530        65.59
        FRAGMENTS STORED             .00           0         .00
        ROOTS OR RCDS RELOCATED      .00           0         .00
        TOTAL I/O                 110.00         110        73.33
        TOTAL CPU (100THS SEC)      5.00           5        71.43
        TASK SYSTEM MODE CPU     .060000      .068297        66.68
        TASK ZIIP ON CP CPU      .000000      .000000          .00
        TASK ZIIP ON ZIIP CPU    .000000      .001419        50.17
        TASK USER MODE CPU       .000000      .000000          .00
        TOTAL TASK TCB CPU       .050000      .053560        68.79
RATIOS----PAGES REQUESTED / PAGES READ                        13.14
        RECORDS REQUESTED / PAGES READ                        19.51
        RECORDS REQUESTED / RECORDS BECOMING CURRENT           7.88
        CALC RCDS OVERFLOW / CALC RCDS ON HOME PAGE             .40
```

Chapter 9: Sample JCL

This section contains the following topics:

Library References in JCL

References are made to the following libraries in z/OS JCL:

zOS.maclib

The data set name of the z/OS macro library

yourHLQ.CAGJMAC

The name of the deployed CA IDMS macro library

yourHLQ.CAGJLOAD

The name of the deployed CA IDMS mixed case load library.

yourHLQ.CAGJLMDU

The name of the deployed CA IDMS uppercase only load library.

This library exists only if you installed and deployed the uppercase only option.

your.custom.srclib

The data set name of the library containing the source modules customized during configuration.

your.custom.loadlib

The data set name of the library containing the load modules customized during configuration.

your.custom.lib

The data set name of the load or object library containing user-written programs, such as exits and built-in functions.

This may be the same library that contains the load modules customized during configuration.

Note: yourHLQ is the high level data set name qualifier that you established during deployment.

z/OS Assemble and Link-Edit JCL

Use the following JCL to assemble and link-edit a module in z/OS.

```
//ASMCL   EXEC HLASMCL
//C.SYSLIB   DD DISP=SHR,DSN=zOS.maclib
//          DD DISP=SHR,DSN=yourHLQ.CAGJMAC
//C.SYSIN    DD DISP=SHR,DSN=your.custom.srclib(source-member)
//L.SYSLMOD  DD DISP=SHR,DSN=your.custom.loadlib
//L.CAGJLOAD DD DISP=SHR,DSN=yourHLQ.CAGJLOAD
//L.CUSTLIB  DD DISP=SHR,DSN=your.custom.lib
//L.SYSIN DD *
 binder-statements
/*
```

Replace the variables in the JCL as explained next:

source-member

Replace *source-member* with the name of the member containing the source to be assembled.

binder-statements

Replace *binder-statements* with the binder input statements appropriate to the load module being created.

z/OS Link-Edit JCL

Use the following JCL to link-edit a module in z/OS:

```
//LNKUXIT EXEC PGM=HEWL,
//           PARM=(XREF,MAP,LET,LIST,NCAL)
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=VIO,SPACE=(1700,(600,100))
//SYSLMOD  DD  DISP=SHR,DSN=your.custom.loadlib
//CAGJLOAD DD  DISP=SHR,DSN=yourHLQ.CAGJLOAD
//CUSTLIB  DD  DISP=SHR,DSN=your.custom.lib
//SYSLIN   DD *
 binder-statements
/*
```

Replace the variables in the JCL as explained next:

binder-statements

Replace *binder-statements* with the binder input statements appropriate to the load module being created.

Chapter 10: Security Codes

The section describes the security codes associated with new and revised DCMT and utility commands.

Note: For more information about using command codes to secure DCMT and utility commands, see the *Security Administration Guide*.

This section contains the following topics:

DCMT Command Codes

The following command codes apply to new and revised DCMT commands:

Code	DCMT Command
N022	DISPLAY MEMORY
N022030	DMCL
N022031	BCR/BUFFER
N022032	JCB/JOURNAL
N022033	SEGMENT
N022034	DPR/AREA
N022035	FCB/FILE
N033	VARY MEMORY
N033003	PROGRAM
N035	HELP
N035051	AUTOTUNE
N035052	MODID
N041	DISPLAY/VARY/WRITE STATISTICS
N041008	DISPLAY STATISTICS ROLL
N041009	VARY STATISTICS ROLL TIME HH:MM
N041010	VARY STATISTICS ROLL FRE ddd
N041011	VARY STATISTICS ROLL TIME HH:MM FRE ddd
N041012	VARY STATISTICS NOROLL
N041013	WRITE STATISTICS ROLL

Code	DCMT Command
N053	VARY SNAP
N053008	TASK TRACE ON
N053009	TASK TRACE OFF
N053010	TASK TRACE TASK
N053011	TASK TRACE LIMIT *nnn*
N053012	TASK TRACE LIMIT OFF
N103	DISPLAY/VARY TRACE/SYSTRACE
N103007	DISPLAY TRACE
N103008	VARY TRACE
N110	DISPLAY AUTOTUNE
N110001	AUTOTUNE [*parameter-name*]
N110002	AUTOTUNE *
N111	VARY AUTOTUNE
N111001	AUTOTUNE … RESET
N111002	AUTOTUNE … OFF
N112	DISPLAY MODID
N112001	[FROM/TO]
N112002	*module-name*

The following DCMT command codes have been removed:

Code	DCMT Command
N103004	DISPLAY DBTRACE
N103005	VARY DBTRACE OFF
N103006	VARY DBTRACE ON

Utility Command Codes

The following command codes apply to new utility commands:

Code	Utility Command
ARCHIVETRACE	ARCHIVE TRACE
PRINTTRACE	PRINT TRACE

Appendix A: Sample OPS/MVS API rule

Use the following OPS/MVS API rule to automate the offloading/archiving processes for DC log, DC traces, and journals. This rule eliminates the need to include a WTOEXIT in the CA IDMS system, and also improves system performance when using the CA IDMS zIIP feature.

While there are many possible ways to write an API rule, this is one example of how you can implement a rule to complete the required job submissions for sample DC/UCF systems SYSTEM72 and SYSTEM73 on the same LPAR. This example processes all events starting with the string CAIDMS*. All other parsing of the messages for JOURNAL, LOG, and TRC is done by the rule itself.

```
)API CAIDMS*
)PROC
/*********************************************************************/
/*                                                                   */
/*          Proprietary and Confidential Information                 */
/*            and Intellectual Property of CA                        */
/*               Copyright (C) 2010 CA                               */
/*                All Rights Reserved.                               */
/*                                                                   */
/*  Name      - APIIDMS                                              */
/*  Purpose   - Respond to various CA IDMS generated API events.     */
/*  Related   - None                                                 */
/*  Globals   - None                                                 */
/*  Notes     -                                                      */
/*           This sample OPS/MVS request rule demonstrates           */
/*           using API rules for CA IDMS sample systems SYSTEM72     */
/*           and SYSTEM73.                                           */
/*                                                                   */
/*           The CA OPS/MVS generic event Application Program        */
/*           Interface (API) enables CA software products to         */
/*           directly generate CA OPS/MVS events. This example       */
/*           will process on API events initiated by the             */
/*           CA IDMS product.                                        */
/*                                                                   */
/*           This API example will process and perform               */
/*           the following actions:                                  */
/*                                                                   */
/*          Offload_DC_LOG - DC log is nn% full (DC050001)           */
/*                     or DC log is full (DC050004)                  */
/*                     messages were received.                       */
/*                     The action will be taken to submit            */
/*                     the DC log offload job for that               */
/*                     particular system.                            */
```

```
/*          Offload_DC_TRC - TRC is nn% full (DC050024)         */
/*                           or TRC is full (DC050027)          */
/*                           messages were received.            */
/*                           The action will be taken to submit  */
/*                           the traces offload job for that     */
/*                           particular system.                 */
/*          Archive_Journal - Disk Journal is full (DC205003)    */
/*                           message was received.              */
/*                           The action will be taken to submit  */
/*                           the archive journal job for that    */
/*                           particular system.                 */
/*                                                              */
/****************************************************************/

/*--------------------------------------------------------------*/
/* Obtain needed event data as well as any need system data that     */
/* required for processing.                                     */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/
msgtxt = overlay(' ',api.text,5)       /* remove * from IDMS message */
localmstcons= OPSINFO('LocMstConsNm')       /* Local master console */
msgid  = word(msgtxt,2)                        /* Message ID */
system = word(msgtxt,3)                    /* system making request */
system = right(system,length(system)-1)      /* Strip away leading V */

/*--------------------------------------------------------------*/
/* User threshold values for DC LOG and DC Trace offloading         */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/
LOG_threshold = log-offload-threshold
TRC_threshold = trc-offload-threshold

/*--------------------------------------------------------------*/
/* DC/UCF system version numbers at your IDMS site                  */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/
SYSTEM72 = 72
SYSTEM73 = 73
/* Add more systems if needed                                   */

/*--------------------------------------------------------------*/
/* Display debugging information                                */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/
MSG1 = "Text Received:" api.text
MSG2 = "MSGID:" msgid" Event:" api.id
MSG3 = "System:" system
```

```
/* Print Debugging information - uncomment the next line if needed    */
/* Call Debug_rule */

/*-------------------------------------------------------------------*/
/* Process this particular class type.                               */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/
select
/* Process "DC050001  Log % full" message */
 when msgid = 'DC050001' then do
   level = word(msgtxt,7)              /* Current % of LOG in use */
   level = left(level,length(level)-1)        /* Strip away % sign */
   If level >= LOG_threshold then   /* DC Log has exceeded threshold */
     Call Offload_DC_LOG
 end

/* Process "DC050004  Log is FULL" message */
 when msgid = 'DC050004' then Call Offload_DC_LOG

/* Process "DC205003  Disk Journal FULL" message */
 when msgid = 'DC205003' then Call Archive_Journal

/* Process "DC050024  TRC % full" message */
 when msgid = 'DC050024' then do
   level = word(msgtxt,7)              /* Current % of LOG in use */
   level = left(level,length(level)-1)        /* Strip away % sign */
   If level >= TRC_threshold then   /* DC Log has exceeded threshold */
     Call Offload_DC_TRC
 end

/* Process "DC050027  TRC is FULL'" message */
 when msgid = 'DC050027' then Call Offload_DC_TRC

 otherwise nop
end
return

/*-------------------------------------------------------------------*/
/* Offload DC LOG subroutine:                                        */
/*  This subroutine will submit offload IDMS DC log job for the      */
/*  Central Version which issued the event.                          */
/*                                                                   */
/*  Add statements for all desired DC/UCF systems to be processed    */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/
Offload_DC_LOG:
select
```

```
/* Submit the job for sample system SYSTEM72 */
  when system = SYSTEM72 then do
   MSGI = "Archive LOG job for the CV" system "has been submitted"
   Call Info_rule        /* Print informational message into JES log */
   address TSO
   "SUBMIT 'your.custom.jcllib(jcl-member)"'
   end

/* Submit the job for sample system SYSTEM73 */
  when system = SYSTEM73 then do
   MSGI = "Archive LOG job for the CV" system "has been submitted"
   Call Info_rule        /* Print informational message into JES log */
   address TSO
   "SUBMIT ' your.custom.jcllib(jcl-member)"'
   end

/* Otherwise do nothing */
  otherwise nop
end
return

/*-------------------------------------------------------------------*/
/* Archive Journal subroutine:                                       */
/*  This subroutine will submit archive journal job for the Central  */
/*  Version which issued the event.                                  */
/*                                                                   */
/*  Add statements for all desired DC/UCF systems to be processed    */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/
Archive_Journal:
select
/* Submit the job for sample system SYSTEM72 */
  when system = SYSTEM72 then do
   MSGI = "Archive journal job for the CV" system "has been submitted"
   Call Info_rule        /* Print informational message into JES log */
   address TSO
   "SUBMIT ' your.custom.jcllib(jcl-member)"'
   end

/* Submit the job for sample system SYSTEM73 */
  when system = SYSTEM73 then do
   MSGI = "Archive journal job for the CV" system "has been submitted"
   Call Info_rule        /* Print informational message into JES log */
   address TSO
   "SUBMIT 'your.custom.jcllib(jcl-member)"'
   end
```

```
/* Otherwise do nothing */
  otherwise nop
end
return

/*--------------------------------------------------------------------------*/
/* Offload DC TRC subroutine:                                               */
/*  This subroutine will submit offload IDMS DC TRC job for the             */
/*  Central Version which issued the event.                                 */
/*                                                                          */
/*  Add statements for all desired DC/UCF systems to be processed           */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/
Offload_DC_TRC:
select
/* Submit the job for sample system SYSTEM72 */
  when system = SYSTEM72 then do
   MSGI = "Archive TRC job for the CV" system "has been submitted"
   Call Info_rule        /* Print informational message into JES log */
   address TSO
   "SUBMIT 'your.custom.jcllib(jcl-member)'"
   end

/* Submit the job for sample system SYSTEM73 */
  when system = SYSTEM73 then do
   MSGI = "Archive TRC job for the CV" system "has been submitted"
   Call Info_rule        /* Print informational message into JES log */
   address TSO
   "SUBMIT 'your.custom.jcllib(jcl-member)'"
   end

/* Otherwise do nothing */
  otherwise nop
end
return

/*--------------------------------------------------------------------------*/
/* Info subroutine:                                                         */
/*  This subroutine is used to print various informational strings          */
/*  into the JES log.                                                       */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/
Info_rule:
 MLINXT.1 = 'OPS/MVS IDMS INFORMATIONAL MESSAGE:'
 MLINXT.2 = COPIES('*',70)
 MLINXT.3 = '* '||CENTER(MSGI,66)||' *'
 MLINXT.4 = COPIES('*',70)
 address WTO
```

```
 "Msgid(OPSNOTIFY) Textvar(MLINXT.) Cnname("localmstcons")"
 MSGI = ""
return

/*-------------------------------------------------------------------*/
/* Debug subroutine:                                                 */
/*  This subroutine is used to print various debugging strings into  */
/*  the JES log.                                                     */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/
Debug_rule:
 MLWTXT.1 = 'OPS/MVS IDMS DEBUG MESSAGE:'
 MLWTXT.2 = COPIES('*',70)
 MLWTXT.3 = '* '||CENTER(MSG1,66)||' *'
 MLWTXT.4 = '* '||CENTER(MSG2,66)||' *'
 MLWTXT.5 = '* '||CENTER(MSG3,66)||' *'
 MLWTXT.6 = COPIES('*',70)
 address WTO
 "Msgid(OPSNOTIFY) Textvar(MLWTXT.) Cnname("localmstcons")"
 MSG1 = ""
 MSG2 = ""
 MSG3 = ""
return
```

To complete the API rule, replace these variables:

log-offload-threshold

Replace log-offload-threshold with the DC log offload threshold value in percent.

trc-offload-threshold

Replace trc-offload-threshold with the DC traces offload threshold value in percent.

your.custom.jcllib

The data set name of the card image (FB/80/????) JCL library containing the DC log and DC traces offload jobs and archive journal jobs.

jcl-member

Replace jcl-member with the name of the member containing the JCL to offload or archive the DC log, DC traces or journals for the particular CA IDMS DC/UCF system.