CA IDMS - 19.0

Using CA IDMS Conversion Tools

Date: 15-Jan-2018

Table of Contents

Using CA IDMS Conversion Tools

This section describes how to process components of your Release 10.2 database environment to create a Release 12.0 physical database definition.

For more information, see the following topics:

Physical Database Definition

Convert DMCL before migrating dictionary

You must process your 10.2 DMCL definition(s) from a 10.2 DDLDML dictionary area before migrating the DDLDML area.

Conversion tools and environment

To generate a 12.0 physical database definition from a 10.2 DMCL definition you need:

This component	For these tasks
10.2 DMCL Compiler	To modify or validate 10.2 DMCLs, if necessary, before submitting them for processing by the DMCL Syntax Generator.
Release 10.2 data dictionary	The DMCL Syntax Generator extracts definitions from a 10.2 data dictionary.
Release 10.2 run-time environment Load libraries DMCL IDMSNWKA subschema describing the dictionary containing the DMCL to be converted For local mode processing, you need the IDMSDBTB database name table for processing secondary dictionaries	To run the DMCL Syntax Generator in a 10.2 environment.
DMCL Syntax Generator	To process 10.2 DMCL definitions and generate 12.0 physical database definition statements.

This component	For these tasks
The DMCL Syntax Generator (IDMSDMCC program) is a 10.2 module that must exist in a standalone load library and can only run in a 10.2 environment. It should have been installed in a separate load library during the installation process.	
Release 12.0 run-time environment Load libraries DMCL Data dictionary (database catalog segment which includes DDLCAT, DDLCATX, and DDLCATLOD areas)	To define, generate, punch, and link-edit physical database components.
CA IDMS Batch Command Facility	To process 12.0 physical database definition statements and populate the database catalog segment of a 12.0 data dictionary. To generate and punch 12.0 DMCL and database name table modules.

Convert 10.2 DMCL definitions

12.0 physical database components

In Release 12.0, the process of defining a physical database is completely separate from defining the logical description of a database. Defining a physical database includes defining:

- Segments -- Segment definitions include area and file definitions as well as a page group assignment and the maximum number of records or rows per page.
- DMCL -- A DMCL definition includes the run-time description of a physical database environment. It includes segment, journal and buffer definitions.

The DMCL Syntax Generator

The DMCL Syntax Generator (IDMSDMCC) generates 12.0 physical database definition statements from validated 10.2 DMCL and schema definitions. The DMCL Syntax Generator executes as a stand-alone batch compiler and will only process DMCL definitions that have been generated with the Release 10.2 DMCL compiler.

You identify the 10.2 DMCLs you want to process by submitting input parameters to IDMSDMCC. Since an area can be included in multiple DMCLs, consider choosing a global DMCL for syntax conversion.

Sequence of statements

The DMCL Syntax Generator creates

- One CREATE SEGMENT statement for each unique schema name and version associated with the areas included in the 10.2 DMCL.
- CREATE FILE statements for those files associated with the AREA statements included in the 10.2 DMCL.
- CREATE AREA statements for those areas included in the 10.2 DMCL.

Important! The DMCL Syntax Generator only converts 10.2 area and file definitions for those areas and files included in the 10.2 DMCL you are converting. You must explicitly add area and file definitions to your 12.0 physical database definition that are part of a schema but are **not** included in the 10.2 DMCL you are converting.

- DMCL statements following SEGMENT, FILE, and AREA statements.

Before you run the DMCL Syntax Generator

Before you run the DMCL Syntax Generator, review these items:

1. Ensure that 10.2 DMCL definitions are validated.
2. The DMCL Syntax Generator runs in a 10.2 environment in either local mode or under the central version.
3. While the DMCL Syntax Generator can process multiple DMCLs in a single job, you should consider processing one DMCL at a time.
4. Do not include any 12.0 load libraries in the IDMSDMCC execution JCL.

Execute the DMCL Syntax Generator

Purpose

The DMCL Syntax Generator processes 10.2 DMCL definitions and generates appropriate Release 12.0 physical database definition syntax.

The DMCL Syntax Generator is a data dictionary compiler and supports the IDD SIGNON statement and the CA IDMS DDL compiler SET OPTIONS statement.

Note: For more information about the IDD SIGNON statement, see the *CA IDMS IDD DDDL Reference Guide*. For more information about the CA IDMS DDL Compiler SET OPTIONS statement, see the *CA IDMS Database Administration Guide*.

Authorization

To display or punch DMCL definitions from a Release 10.2 dictionary, you must have authority to use the 10.2 DMCL compiler.

If you have dictionary signon security turned on or you are accessing a secondary dictionary and need to specify a DICTNAME, you must include a SIGNON DICTIONARY statement in the SYSIPT file before any DMCL Syntax Generator input statements.

Syntax

```
►►─┬─ DISplay ─┬─ DMCl name is dmcl-name ─────────────────────────────────►
   └─ PUNch ───┘

►─┬──────────────────────────────────────────────────────────────────────┬─►
  └─ of SCHema name is schema-name ─┬─────────────────────────────────────┤
                                    └─ Version is ─┬─ version-number ─┬─┘
                                                   ├─ LOWest ─────────┤
                                                   └─ HIGHest ────────┘

►─┬──────────────────────────┬─────────────────────────────────────────────►◄
  └─ AS ─┬─ SYNtax ──────────┤
         └─ COMments ◄ ───┘
```

Parameters

- **DISplay**
  Indicates that IDMSDMCC will display the physical database definition syntax it will generate.

- **PUNch**
  Indicates that IDMSDMCC will display and punch the physical database definition syntax it will generate.

- **DMCL name is *dmcl-name***
  Identifies the name of the Release 10.2 DMCL definition for which 12.0 syntax is generated.

- **of SCHema name is *schema-name***
  Identifies the name of the schema associated with the named DMCL.
  Schema-name is required only if the DMCL name is not unique in the dictionary.

- **Version is**
  Optionally qualifies the named schema with a version number.

  - ***Version-number***
    Specifies the explicit version number assigned to the named schema.
    Version-n is in the range 1 through 9999 and defaults to the current session option.

  - **LOWest**
    Specifies the lowest version number assigned to the named schema.

  - **HIGHest**
    Specifies the highest version number assigned to the named schema.

- **AS SYNtax**
  Indicates that IDMSDMCC will produce the physical database definition statements as syntax which you can then edit and resubmit to the CA IDMS Batch Command Facility.

- **COMments**
  Indicates that IDMSDMCC will produce the physical database definition statements with comment identifiers in columns 1 and 2.

Usage

Converting multiple DMCLs with the same name

If you have multiple 10.2 DMCLs with the same name, the DMCL Syntax Generator will generate multiple CREATE DMCL statements with the same name. You should resolve multiple DMCL names. Here are some suggestions:

- Change the name of the DMCL or
- Delete the DMCL if you don't need it or
- Assign the segments and other physical database components to another DMCL

IDMSDMCC compiler batch execution JCL

You submit the DMCL syntax generator statements to the IDMSDMCC compiler. IDMSDMCC will run in either local mode or under the central version.

z/OS execution JCL

Here is the z/OS execution JCL when running IDMSDMCC under the central version.

Important! Do not include any Release 12.0 load libraries in the execution JCL for IDMSDMCC.

IDMSDMCC JCL

```
//IDMSDMCC   EXEC  PGM=IDMSDMCC,REGION
//STEPLIB    DD    DSN=dmcc.loadlib,DISP=SHR
//           DD    DSN=cdms.loadlib,DISP=SHR
//sysctl     DD    DSN=dbdc.sysctl,DISP=SHR
//SYSLST     DD    SYSOUT=A
//SYSPCH     DD    SYSOUT=B
//SYSIPT     DD    *

Include dictionary SIGNON statement here

Include IDMSDMCC input statements here
```

dmcc.loadlib	Data set name of stand-alone load library that contains the IDMSDMCC program
cdms.loadlib	Data set name of 10.2 system software load library
sysctl	DDname for SYSCTL file
dbdc.sysctl	Data set name of SYSCTL file

To execute the DMCL Syntax Generator in local mode:

- Add the name of the load library that contains your 10.2 run-time DMCL and network subschema load modules. Also, include the IDMSDBTB database name table if you are using a non-default dictionary.
- Add the name of the data dictionary DDLDML area containing the DMCL you are processing.
- Add the appropriate journal specifications.

z/VSE execution JCL

Here is the z/VSE execution JCL when running IDMSDMCC under the central version.

Important! Do not include any Release 12.0 load libraries in the execution JCL for IDMSDMCC.

IDMSDMCC JCL

```
// LIBDEF *,SEARCH=10.2 libraries// DLBL   dmcc,'cdms.dmcc.lib'
// ASSGN  SYSxxx,DISK,VOL=nnnnnn,SHR
// DLBL   IDMSPCH,'syspch.dsn'
// EXTENT SYSPCH,nnnnnn,,,ssss,llll
// EXEC   IDMSDMCC,SIZE=1048K
Include dictionary SIGNON statement of SIGNON SECURITY is on
/*
Include IDMSDMCC input statements here
/*
```

10.2 libraries	10.2 library definitions
cdms.dmcc.lib	File identifier of the library containing the IDMSDMCC program
syspch.dsn	Filename of the SYSPCH file

To execute the DMCL Syntax Generator in local mode:

- Add the name of the load library that contains your 10.2 run-time DMCL and network subschema load modules. Also, include the IDMSDBTB database name table if you are using a non-default dictionary.
- Add the name of the data dictionary DDLDML area containing the DMCL you are processing.
- Add the appropriate journal specifications.

z/VM Commands

Here are the z/VM commands to run IDMSDMCC under the central version.

Important! Do not include any Release 12.0 load libraries in the execution JCL for IDMSDMCC.

IDMSDMCC JCL

```
FILEDEF SYSLST PRINTER
FILEDEF SYSPCH PRINTER
FILEDEF SYSIPT DISK idmsdmcc input a
GLOBAL LOADLIB dmcc idmslib
OSRUN IDMSDMCC
```

dmcc	Filename of the load library containing the IDMSDMCC program
idmslib	Filename of the CA IDMS 10.2 load library

To execute the DMCL Syntax Generator in local mode:

- Add the name of the load library that contains your 10.2 run-time DMCL and network subschema load modules. Also, include the IDMSDBTB database name table if you are using a non-default dictionary.
- Add the name of the data dictionary DDLDML area containing the DMCL you are processing.
- Add the appropriate journal specifications.

Important! Do not include any Release 12.0 load libraries in the execution JCL for IDMSDMCC.

IDMSDMCC JCL

```
/ FILE     LINK=sysctl,dbdc.sysctl,SHARUPD=YES
/ FILE     LINK=CDMSLIB,cdms.loadlib
/ SYSFILE  SYSOPT=cdms.sysopt
/ SYSFILE  SYSDTA=(SYSCMD)
/ EXEC     (IDMSDMCC,dmcc.loadlib)
Include dictionary SIGNON statements here

Include IDMSDMCC input statements here
```

dbdc.sysctl	Filename of SYSCTL file
cdms.loadlib	Filename of 10.2 system software load library
cdms.sysopt	Filename of the SYSOPT file
dmcc.loadlib	Filename name of stand-alone load library that contains the IDMSDMCC program

To execute the DMCL Syntax Generator in local mode:

- Add the name of the load library that contains your 10.2 run-time DMCL and network subschema load modules. Also, include the IDMSDBTB database name table if you are using a non-default dictionary.

- Add the name of the data dictionary DDLDML area containing the DMCL you are processing
- Add the appropriate journal specifications

What the DMCL Syntax Generator produces

CREATE SEGMENT statement

A segment statement is created for each unique schema identified in the OF SCHEMA NAME clause of the AREA statements of the named DMCL.

The PAGE GROUP and the MAXIMUM RECORDS PER PAGE (formerly the PAGE CONTAINS clause of the SCHEMA statement) are also part of the CREATE SEGMENT statement.

The DMCL Syntax Generator creates segment names as follows:

- The segment name is the same as the schema name for the first *schema-name version-n* encountered in the DMCL
- Subsequent occurrence(s) of the same schema name with different version numbers will be assigned segment names in the form:
  SEGnnnnn starting with SEG00001 through SEG99999.

Sample output

This is the name of a segment and the name and version of the schema to which the segment applies.

This segment name is created from another occurrence of the same schema name with a different version number.

```
IDMSDMCC  10.2        CA, INC.                                DATE     PAGE
 S10200               10.2 DMCL TO 12.0 SYNTAX GENERATOR     10/31/90  0001

0000001       DISPLAY DMCL TESTDMCL AS SYNTAX.
 *+   ***=============================================================**
 *+ SEGMENT TESTSCHM CREATED FROM SCHEMA TESTSCHM VERSION 1 ;
      CREATE SEGMENT TESTSCHM
          PAGE GROUP 0
          MAXIMUM RECORDS PER PAGE 255
          ;
 *+ SEGMENT SEG00001 CREATED FROM SCHEMA TESTSCHM VERSION 2 ;
      CREATE SEGMENT SEG00001
          PAGE GROUP 0
          MAXIMUM RECORDS PER PAGE 255
          ;
```

CREATE FILE statement

A CREATE FILE statement is created for each file included in the 10.2 DMCL.

The ASSIGN TO clause will reflect the external name associated with the DMCL file name either through the default schema file definition or through a DMCL file override.

Native VSAM

An access method clause is generated for native VSAM files only.

If the file is a native VSAM KSDS or PATH file, you must modify the CREATE FILE statement to include appropriate FOR CALC and/or FOR SET clauses on the access method clause.

Sample output

```
CREATE FILE TESTSCHM.MULTIPLE-FILE-1
    ASSIGN TO MULFILE1
    ;
CREATE FILE TESTSCHM.MULTIPLE-FILE-2
    ASSIGN TO MULTFIL2
    ;
CREATE FILE TESTSCHM.MULTIPLE-FILE-3
    ASSIGN TO MULFILE3
    ;
```

CREATE AREA statement

A CREATE AREA statement is created for each area included in the 10.2 DMCL.

The value specified in the PRIMARY SPACE clause is the value specified on the PAGE RANGE IS clause of the SCHEMA AREA statement.

PAGE SIZE and ORIGINAL PAGE SIZE information is extracted directly from the 10.2 DMCL AREA statement PAGE CONTAINS and SPACE MANAGEMENT INTERVAL clauses.

Symbolics are not generated.

Alias area names are not used by the IDMSDMCC compiler. IDMSDMCC uses the area name specified in the ALIAS OF *area-name* OF SCHEMA *schema-name* clause as the 12.0 physical area name.

Native VSAM

A WITHIN FILE clause will be generated for each non-path file mapping in the DMCL area. If the file map is a native VSAM ESDS, KSDS, or RRDS file, no FROM/THRU subclause is generated. If the file map is a native VSAM path file, a WITHIN PATH FILE clause is generated in place of the WITHIN FILE clause.

Sample output

PRIMARY SPACE is created from the SCHEMA AREA statement PAGE RANGE IS clause.

PAGE SIZE and ORIGINAL PAGE SIZE are created from the DMCL AREA statement PAGE CONTAINS and SMI BASED ON clauses.

```
      CREATE PHYSICAL AREA TESTSCHM.MULT-FILE-AREA
      PRIMARY SPACE 100 PAGES  FROM PAGE 400300
          PAGE SIZE 6144 CHARACTERS
          ORIGINAL PAGE SIZE 5120 CHARACTERS
          WITHIN FILE MULTIPLE-FILE-1  FROM 1 THRU 25
          WITHIN FILE MULTIPLE-FILE-2  FROM 1 THRU 30
          WITHIN FILE MULTIPLE-FILE-3  FROM 1 THRU 45
          ;
```

Generated DMCL Statements

Once the segment statements are created, the DMCL Syntax Generator builds these release 12.0 DMCL statements:

- CREATE DMCL statement
- CREATE BUFFER statements
- CREATE JOURNAL BUFFER statement
- CREATE JOURNAL statements

CREATE DMCL statement

A CREATE DMCL statement is created for the 10.2 DMCL you are processing. The name of the 10.2 DMCL becomes the name of the Release 12.0 DMCL.

Sample output

```
CREATE DMCL TESTDMCL ;
```

CREATE BUFFER statement

A buffer statement is created for each standard buffer defined in the 10.2 DMCL.

PAGE SIZE is the value specified on the PAGE CONTAINS clause in the 10.2 DMCL.

LOCAL MODE and CV MODE BUFFER INITIAL PAGES and MAXIMUM PAGES clauses are based on the 10.2 DMCL BUFFER CONTAINS clause. You may want to modify these values to create different size buffer pools for local mode and CV operations.

Note: MAXIMUM PAGES clause is not generated for native VSAM buffers.

Native VSAM

For native VSAM buffers, either of the following clauses is generated:

- NATIVE VSAM LSR KEYLEN *key-length* STRNO *string-number*
  or
- NATIVE VSAM NSR BUFNI *buffer-number* STRNO *string-number*

Sample output

PAGE SIZE is created from the PAGE CONTAINS clause on the BUFFER statement in the 10.2 DMCL.

LOCAL MODE and CV MODE BUFFER are created from the BUFFER CONTAINS clause of the BUFFER statement in the 10.2 DMCL.

```
      CREATE BUFFER TESTDMCL.MULTIPLE-FILESPAGE SIZE 6144 CHARACTERS
          LOCAL MODE BUFFER PAGES 9
          CV MODE BUFFER  INITIAL PAGES 9  MAXIMUM PAGES 9
          ;
      CREATE BUFFER TESTDMCL.MULTIPLE-TEST
          PAGE SIZE 6144 CHARACTERS
          LOCAL MODE BUFFER PAGES 9
          CV MODE BUFFER  INITIAL PAGES 9  MAXIMUM PAGES 9
          ;
      CREATE BUFFER TESTDMCL.BUF1-TEST
          PAGE SIZE 3156 CHARACTERS
LOCAL MODE BUFFER PAGES 4
CV MODE BUFFER INITIAL PAGES 4 MAXIMUM PAGES 4
          ;
```

CREATE JOURNAL BUFFER statement

A CREATE JOURNAL BUFFER statement is created for the journal buffer defined in the 10.2 DMCL. If no journal buffer is defined in the 10.2 DMCL, you must define a journal buffer in the 12.0 DMCL before you generate it.

Sample output

```
CREATE JOURNAL BUFFER TESTDMCL.JOURNAL-BUFFER
    PAGE SIZE 6144 CHARACTERS
    BUFFER PAGES 9
    ;
```

CREATE JOURNAL statements

The appropriate 12.0 DMCL journal statement is created for each journal file defined in the 10.2 DMCL. There are three types of JOURNAL statements:

- DISK JOURNAL
- ARCHIVE JOURNAL
- TAPE JOURNAL

CREATE DISK JOURNAL statement

A CREATE DISK JOURNAL statement is generated for each DISK JOURNAL FILE statement in the 10.2 DMCL.

Sample output

```
CREATE DISK JOURNAL TESTDMCL.DISK-JOURNAL
          FILE SIZE 5000 BLOCKS
          ASSIGN TO SYSJRNL
          ;
```

CREATE ARCHIVE JOURNAL statement

A CREATE ARCHIVE JOURNAL statement is created for each ARCHIVE JOURNAL statement in the 10.2 DMCL.

Sample output

```
CREATE ARCHIVE JOURNAL TESTDMCL.ARCHIVE-JOURNAL
    BLOCK SIZE 4096 CHARACTERS
    ASSIGN TO SYSARCH
    ;
```

CREATE TAPE JOURNAL

A CREATE TAPE JOURNAL statement is created for each TAPE JOURNAL statement if one is included in the 10.2 DMCL.

https://docops.ca.com/display/IDMS19/DMCL+Statements

Segment definitions within the DMCL

ALTER DMCL statement

Once the DMCL is defined, segment definitions and other components of the physical database definition are associated with it. To complete the 12.0 DMCL definition, the DMCL Syntax Generator creates:

- ALTER DMCL statement
- DEFAULT BUFFER clause specifying the first standard buffer defined in the 10.2 DMCL
- INCLUDE SEGMENT clauses to associate the previously defined segments with the DMCL
- INCLUDE FILE clauses for each file in the segment to explicitly assign the file to a buffer

> **Note:** INCLUDE FILE statements are not generated for files assigned to theDEFAULT BUFFER.

- INCLUDE AREA clauses as required for each area in the segment to accommodate the PAGE RESERVE clause if it was included in the 10.2 DMCL

Sample output

The INCLUDE SEGMENT clause associates the TESTSCHM segment with the TESTDMCL DMCL.

This INCLUDE FILE clause identifies the buffer the file uses.

This INCLUDE AREA clause identifies a page reserve for the MULT-FILE-AREA area.

```
      ALTER DMCL TESTDMCL
          DEFAULT BUFFER DEFAULT-BUFFER
 INCLUDE SEGMENT TESTSCHM
              INCLUDE FILE TESTSCHM.MULTIPLE-FILE-1
                  BUFFER MULTIPLE-FILES
              INCLUDE FILE TESTSCHM.MULTIPLE-FILE-2
                  BUFFER MULTIPLE-FILES
 INCLUDE FILE TESTSCHM.MULTIPLE-FILE-3
 BUFFER MULTIPLE-FILES              INCLUDE PHYSICAL AREA TESTSCHM.MULT-FILE-AREA
                  PAGE RESERVE 200
          INCLUDE SEGMENT SEG00001
              INCLUDE FILE SEG00001.MULTIPLE-TEST-1
                  BUFFER MULTIPLE-TEST
              INCLUDE FILE SEG00001.MULTIPLE-TEST-2
                  BUFFER MULTIPLE-TEST
              INCLUDE FILE SEG00001.MULTIPLE-TEST-3
                  BUFFER MULTIPLE-TEST
 INCLUDE PHYSICAL AREA SEG00001.MULT-FILE-AREA
 PAGE RESERVE 120
          ;
```

Transaction summary

The transaction summary summarizes the results of the processing by the DMCL Syntax Generator.

Error messages are displayed before the transaction summary.

Sample transaction summary

```
              ** TRANSACTION SUMMARY **
ENTITY                ADD MODIFY REPLACE DELETE DISPLAY
..................... ... ...... ....... ...... .......
DMCL                    0      0       0      0       1
NO ERRORS OR WARNINGS ISSUED FOR THIS COMPILE
```

Correct errors from IDMSDMCC

The possible errors you might receive from IDMSDMCC are those that the 10.2 DMCL compiler produces with one addition. If the DMCL you are converting isn't in a 10.2 format or isn't validated, IDMSDMCC generates the following message:

```
E DC643253 DMCL NOT 10.2 FORMAT OR CONTAINS ERRORS - 10.2
           VALIDATE REQUIRED
```

If you receive this message, check to see if the DMCL is validated using the 10.2 DMCL compiler. Validate the DMCL if it is not validated.

Create 12.0 DMCL module

Review the output from the IDMSDMCC compiler

You should review the physical database definition statements the IDMSDMCC compiler generates and modify them to reflect the requirements of your 12.0 database environment.

What to review

Here is a list of some of the items you should review before defining, generating, and punching a 12.0 DMCL. This is not meant to be a complete list and you should make all the necessary changes to meet the needs of your environment.

After you modify the physical database definition statements, submit them to the CA IDMS Batch Command Facility for processing.

Note: For more information about the Batch Command Facility, see the CA IDMS Command Facility.

Action	Statement
Add area and file definitions for those areas and files included in a schema and not included in the converted 10.2 DMCL. You must add these definitions to ensure there are no missing page and file block ranges.	CREATE FILE and AREA statements.
Review segment names and change them if they do not conform to your naming conventions. **Note:** It is recommended that you create a system dictionary that is separate from application dictionaries. Before you generate and punch your global DMCL module, see Dictionary Migration and Setup (see page 41) for a discussion of setting up a system dictionary.	CREATE SEGMENT, FILE, and AREA statements and INCLUDE SEGMENT, FILE, and AREA clauses under the ALTER DMCL statement.

Review area names and segment assignments.
 	
Change area names for dictionary DML and load areas to DDLDML and DDLDCLOD. If you have multiple, DDLDML and DDLDCLOD areas, assign them to different segments.
 	
Review the names of the areas in your subschemas and verify that they match area names in your segments. If they are different, change the area names in your segment definitions.
 	
Assign user database and dictionary areas to segments to support your processing requirements.
	CREATE AREA statement and possibly the INCLUDE AREA and FILEclauses.
	
	CREATE AREA and possibly the INCLUDE AREA clause.
	
	CREATE AREA and FILE statements and INCLUDE SEGMENT AREA and FILE clauses.

Assign the message area (DDLDCMSG) to the SYSMSG segment.	AREA statement and INCLUDE SEGMENT and AREA clauses.
Review the R120DMCL and determine if you need to define any of its dictionaries and sample databases in your global DMCL. **Note:** Additional segment definitions are required if you are using the CA IDMS central security facility. For more information about using the CA IDMS central security facility, see Security Migration (see page 30).	Add the appropriate statements to your global DMCL definition.
Add database name table name to DMCL definition. While CA IDMS does not require you to define a database name table, you will need to have a database name table to define a default dictionary, define database names and associate segments with them and to accommodate other run-time functions. You can add a database name table name to the DMCL definition before you actually define and punch the database name table. However, at run-time both the DMCL load module and the DBTABLE module it references must exist.	DBTABLE clause of the ALTER DMCL statement.
Review buffer specifications and the new BUFFER statement parameters. For example, local mode and central version buffer pool sizes can be different in the same DMCL.	Modify the BUFFER statement to alter parameters or include new parameters.

Action	Statement
Ensure that a journal buffer is defined.	CREATE JOURNAL BUFFER.
Native VSAM users must modify FILE statements. If the file is a native VSAM file or a native VSAM file representing a VSAM data set and index on which a set is defined, modify the FILE statement to include appropriate FOR CALC and/or FOR SET clauses on the access method clause.	Change the FILE statement.
Define the physical database definition and populate the database catalog segment of the data dictionary using the CA IDMS Batch Command Facility.	
Generate and punch the DMCL load module.	GENERATE DMCL. PUNCH DMCL LOAD MODULE utility statement.
Link-edit the resulting object module to a load (core-image) library.	Use the linkage-editor for your operating system.

Create a 12.0 database name table

CA IDMS/DB requires the presence of a database name table for most processing. Typically you only need one database name table for all DMCLs defined within a system dictionary. You do not need to create separate database name tables for each DMCL.

Why you need a database name table

You need a database name table:

1. To define a default dictionary for both online and local mode processing
2. To group multiple segments under one name for processing as a single database
3. To identify the database to be accessed by a run unit when no database name is provided by the application or run-time environment

What's next

The remainder of this section describes the differences between 10.0 and 12.0 database name tables and then provides guidelines for defining a database name table in Release 12.0 based on these differences.

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

Specifying the default dictionary

What is the default dictionary

The default dictionary is the dictionary accessed whenever one is not specified. In Release 10.2, it was sometimes referred to as the primary dictionary and its areas were those whose page ranges were in the IDMSNWKA subschema.

In Release 12.0, page ranges are no longer in subschemas; therefore another mechanism must be used to identify the default dictionary. The database name table is that mechanism.

Default dictionary identified as a DBNAME

The default dictionary is identified as the DBNAME specified in the DBTABLE mapping for the IDMSNWKL subschema. Since you typically want to map all subschemas whose names begin with "IDMSNWK" in the same way, the DBTABLE mapping statement usually looks like this:

```
SUBSCHEMA IDMSNWK? MAPS TO IDMSNWK? DBNAME SYSDICT
```

Every database name table must have a default dictionary specification.

Grouping segments

When binding a run unit in 10.2, the page ranges in the subschema identified the areas to be accessed. In 12.0, subschemas no longer contain page ranges so the areas to be accessed by a run unit must be identified in some other way.

Segment name as DBNAME

If the areas to be accessed are all within one segment, the application can simply specify the name of the segment as the DBNAME on the BIND RUNUNIT statement (or specify it externally through DCUF or SYSIDMS parameters).

For example, the EMPLOAD program executed during CA IDMS installation only requires access to areas of the EMPDEMO segment. The SYSIDMS parameter file in the execution job stream specifies DBNAME=EMPDEMO, identifying the segment to be accessed. No special DBNAME entry is required.

Multiple segments associated with a DBNAME

If, on the other hand, the application must access areas within multiple segments, those segments must be identified as part of the definition of a DBNAME in the database name table. When the bind takes place, CA IDMS/DB searches the segments associated with the DBNAME for a match on the area name in the subschema. Only one area will match because the name of all areas in segments associated with a DBNAME must be unique. If this condition is not satisfied, the DBNAME is flagged as "in error" and run units receive a 1494 error status when binding to the DBNAME. To avoid this situation, use the DMCL option of the IDMSLOOK utility to validate your database names prior to making the DMCL or DBTABLE available for processing.

Examples

The DBTABLE provided during installation again provides an example of using a DBNAME to group segments together as one database.

The system dictionary is the dictionary used to contain both physical database definitions (DMCLs, SEGMENTs, etc.) and the DC/UCF system definition. The logical name of this dictionary must be SYSTEM, since components of the run-time system access it under this name. However, it is composed of multiple segments:

- The CATSYS segment containing the DDLCAT, DDLCATX and DDLCATLOD areas
- The SYSTEM segment containing the DDLDML, DDLDCLOD and other system run-time areas
- The SYSMSG segment containing the messages issued by the run-time system

In order to treat all of these segments as a single database for processing by tools such as IDD and the CA IDMS Command Facility, the database name table contains a DBNAME called SYSTEM which includes all three segments as part of its definition.

Specifying a DBNAME at run time

In 10.2, it was possible to access a database without naming it. The only requirement was that the application identify a subschema whose page ranges matched those of the areas to be accessed. In 12.0, because subschemas no longer contain page ranges, the name of the database or segment to be accessed must be available at run time.

Options for specifying DBNAME at run time

A database name or segment name can be specified at run time any of the following ways:

- By the application, using the DBNAME parameter on the BIND RUNUNIT statement.
- From the session profile DBNAME attribute. Session attributes are established through user or system profiles, DCUF SET commands in DC/UCF or SYSIDMS parameters in batch.
- From the database name table through the use of subschema mapping parameters on the DBTABLE statement.

No DBNAME specified during bind processing

When binding a run unit in 12.0, if no DBNAME has been established by the application or the session profile, CA IDMS/DB searches the list of subschema mappings looking for one in which the *from-subschema* matches the name of the subschema specified on the BIND statement. If a match is found, the DBNAME specified in the subschema mapping identifies the segments (and areas) to be accessed by the run unit. If no match is found, the bind operation fails with an error status of 1491.

To ensure that run units will bind successfully, you must specify subschema mappings for all run units that bind without establishing a DBNAME through application program logic or session profile settings.

Guidelines for defining database name tables

It is impossible to provide a complete set of rules for defining database name tables, since each site will have different considerations. However, it is possible to provide some examples of simple situations and some points to be considered in more complex cases. Before defining your database name table, you should also read the appropriate sections in the *CA IDMS Database Administration Guide*.

You define 12.0 database name tables by submitting statements to the CA IDMS Batch Command Facility.

Basic DBTABLE definition

If your 10.2 system has no database names defined or has no duplicate area names (you did not have to use the ALIAS parameter of the AREA statement in defining your 10.2 DMCL), then define your 12.0 database name table as follows:

1. Define a DBNAME for each of your dictionaries. You will have a DBNAME for:
   - Your one and only application dictionary which includes the segment containing the DDLDML and DDLDCLOD areas of your migrated 10.2 dictionary and the SYSMSG segment containing your 10.2 DDLDCMSG area populated with 12.0 CA IDMS messages
   - The SYSTEM dictionary which should include the same segments as those defined during installation
   - The SYSDIRL dictionary as defined during installation, unless you run IDMSDIRL against your application dictionary
2. On the DBTABLE statement, identify the application dictionary as your default dictionary by including the following:
   ```
   SUBSCHEMA IDMSNWK? MAPS TO IDMSNWK? DBNAME DEFDICT
   ```
   Where DEFDICT is the DBNAME of your application dictionary
3. Define a default DBNAME that includes all non-dictionary segments
4. As the last subschema mapping parameter on the DBTABLE statement, include the following:
   ```
   SUBSCHEMA ???????? MAPS TO ???????? DBNAME DEFDB
   ```
   Where DEFDB is the name given to the default DBNAME.
   This will allow your applications to issue BIND RUNUNIT statements, even though they do not specify a DBNAME, since all subschemas (except those beginning with IDMSNWK) will use the default DBNAME.

The complete database name table might look like this:

```
CREATE DBTABLE ALLDBS
 SUBSCHEMA IDMSNWK? MAPS TO IDMSNWK? DBNAME DEFDICT
```

```
 SUBSCHEMA ???????? MAPS TO ???????? DBNAME DEFDB;

CREATE DBNAME ALLDBS.SYSTEM
 SEGMENT CATSYS
 SEGMENT SYSTEM
 SEGMENT SYSMSG;
CREATE DBNAME ALLDBS.DEFDICT
 SEGMENT DEFDICT
 SEGMENT SYSMSG;
CREATE DBNAME ALLDBS.DEFDB
 SEGMENT USER-SEGMENT1
 SEGMENT USER-SEGMENT2;
 .
 .
 .
```

Secondary dictionaries

Sharing dictionary areas

If your system has multiple dictionaries defined in 10.2, you may or may not need to define additional DBNAMEs depending on whether those dictionaries share areas:

- If every dictionary has its own DDLDML and DDLDCLOD area and shares the system message area (the SYSMSG segment in 12.0), then no additional DBNAME entries are required.
- If two or more dictionaries share the same DDLDCLOD area, then you must place the load area in its own segment and define a DBNAME for each dictionary which shares the load area. Include in the DBNAME definition both the segment that contains the DDLDML area and the segment that contains the shared load area.
- If two or more dictionaries share a DDLDCMSG area other than the system message area, then use the approach outlined above to share the message area.

Different page groups

If your dictionaries have different page groups (or db-key radixes), they cannot share areas. This also applies to the system message area (in segment SYSMSG), which can be included only in dictionaries having the same page group.

Conflicting area names

Used 10.2 DMCL ALIAS parameter

If you have user database areas with conflicting names requiring the use of the ALIAS parameter in your 10.2 DMCL definition, you must define separate DBNAMEs in your 12.0 database name table for each set of conflicting area names. You should not include segments containing conflicting area names in your default DBNAME definition.

Used DBNAMES in 10.2

If you used DBNAMEs in 10.2, then each 10.2 DBNAME will correspond to a 12.0 DBNAME. If, instead of using DBNAMEs in 10.2, your applications dynamically changed the name of the subschema to which they bound the run unit, then you must use subschema mapping rules on the DBTABLE statement to achieve the same result in 12.0.

For example, if an application program binds a run unit using either subschema EMPE01 or EMPW01, depending on whether it needs to access eastern region employee data or western region, then the 12.0 DBTABLE statement would contain the following mapping rules:

```
SUBSCHEMA EMPE01 MAPS TO EMPE01 USING DBNAME EMPEAST
SUBSCHEMA EMPW01 MAPS TO EMPW01 USING DBNAME EMPWEST
```

You must define DBNAMEs for EMPEAST and EMPWEST identifying the segment (or segments) containing the eastern or western region data.

Subschema naming standards

If you use naming standards for your subschemas, you can generalize the above mapping rules as:

```
SUBSCHEMA EMPE???? MAPS TO EMPE???? USING DBNAME EMPEAST
SUBSCHEMA EMPW???? MAPS TO EMPW???? USING DBNAME EMPWEST
```

You might also wish to eliminate separate subschemas for the western region. If the schema definitions for both east and west are identical (except for pages ranges in 10.2), then you can eliminate the use of separate subschemas by changing the second mapping rule to:

```
SUBSCHEMA EMPW???? MAPS TO EMPE???? USING DBNAME EMPWEST
```

Non-dictionary DBNAMEs

For each DBNAME representing a user database in your 10.2 system, you should define a similarly named DBNAME in your 12.0 database name table.

Review 10.2 subschema mapping rules

In most cases, you will no longer require the 10.2 subschema mapping rules within the DBNAME definition. In 10.2, these rules caused the name of the subschema known to the program to be changed to one with the appropriate page ranges. Since 12.0 subschemas don't contain page ranges, one subschema can be used to access many physical implementations of the same logical database. Changing the name of the subschema is useful only if the subschemas are derived from different schema definitions.

DBNAME specified externally in 10.2

In your DBNAME definition, you must identify the segments containing the data to be accessed by applications binding to the DBNAME. If all applications specify the DBNAME on the BIND statement, then only segments accessed by those applications need to be included in the DBNAME. If, on the other hand, the DBNAME is specified externally in 10.2 (by using DCUF commands or SYSCTL parameters, etc.) then you may need to include additional segments within your DBNAME definition or use subschema mapping rules to ensure that run units bind successfully.

Example

To illustrate this point, assume again that we have eastern and western region employee data, but in this case, the selection at run time is made by issuing a DCUF SET DBNAME command before executing the online application. If the application accesses only employee data, then the only segment that needs to be included in each DBNAME is the one containing the corresponding employee data. (In fact, if the segments are named the same as the 10.2 DBNAMEs, then you don't even need to define DBNAMEs.)

However, if the application also accesses corporate-wide insurance information (stored in its own segment) then:

1. If the insurance information is accessed in the same run unit as the employee information, include the segment containing the insurance information in both DBNAMEs
2. If the insurance information is accessed in a separate run unit, either:
   - Include the insurance segment in both DBNAMEs or
   - Use subschema mappings to redirect the insurance run unit to a different DBNAME

Redirecting run unit to different DBNAME

To redirect the run unit to a different DBNAME using subschema mappings, specify the following in the DBNAME definitions for EMPEAST and EMPWEST:

```
SUBSCHEMA EMP????? MAPS TO EMP?????
SUBSCHEMA ???????? USES DBTABLE MAPPING
```

These parameters have the effect of treating run units binding to subschemas other than those beginning with EMP, as if no DBNAME were specified, hence causing the DBNAME to be selected based on the subschema mapping rules associated with the DBTABLE statement. If you defined your DBTABLE as described under Basic DBTABLE definition (see page 26), then unless a more specific mapping rule applies, all run units will use the default DBNAME which should include the insurance segment.

Mixed page groups

If you have segments in different page groups (or having different db-key radixes), then there are some additional considerations.

In 12.0, just as in 10.2, you cannot access areas in different page groups within a single run unit. Any attempt to do this will result in a bind failure. The best way to avoid this type of error is to ensure that all segments associated with a DBNAME are in the same page group.

Note: CA IDMS 14.1 lets you specify MIXED PAGE GROUP BINDS ALLOWED for a DBNAME. When this option is used, a run unit can bind to a DBNAME that contains mixed page groups. For more information, see the *CA IDMS r14.1 Features Guide* and *CA IDMS Database Administration Guide*.

CA IDMS does not prevent mixing segments with different page groups in a DBNAME because there are conditions under which it is desirable to do so. In fact, it is possible that the basic DBTABLE defined above might have segments with different page groups within the default DBNAME. Provided that no one subschema references areas from segments in different page groups (unlikely given the initial assumptions), then this presents no problem.

You can detect potential problems by using the IDMSLOOK utility (or the DC/UCF LOOK task). The DMCL option will warn you if you have mixed page groups within any of your DBNAMEs. The BIND SUBSCHEMA option will notify you of any errors encountered in binding a specific subschema to a specific DBNAME.

Security Migration

The CA IDMS Central Security Facility

The CA IDMS central security facility enables you to control access to the resources in your CA IDMS environment. You can use this facility to secure all CA IDMS run-time components, when running under the central version or when running in local mode. You can also use the central security facility to provide definition-time protection for system generation, physical database, and SQL-defined entities.

You will continue to use dictionary security to protect the definitions of schemas, subschemas, records, and other dictionary entities.

The CA IDMS central security facility controls access to resources using CA IDMS facilities or an external security package such as CA ACF2 and CA Top Secret.

Note: You should be familiar with CA IDMS central security concepts beforemigrating security definitions. For more information about the central security facility, see the *CA IDMS Security Administration section*.

Security Definition Migration

If you choose to use the IDMS security facility to protect your CA IDMS resources, a conversion utility is provided that creates initial 12.0 security definitions from your 10.2 system definitions.

This section describes how to use the security definition migration utility to convert Release 10.2 security definitions to Release 12.0 security statements.

For more information, see the following topics:

- About security definition migration (see page 31)
- Convert security definitions (see page 33)
- Create execution JCL for the RHDCSMIG program (see page 33)
- Output from RHDCSMIG (see page 35)
- Review and modify output from RHDCSMIG (see page 39)

- Populate 12.0 system with security definitions (see page 40)

About security definition migration

What is the security definition migration utility

The security definition migration utility is a program that converts user, task, and program security definitions from a 10.2 dictionary to preliminary 12.0 security definitions and authorizations. After executing the security migration utility, review the security statements and modify them, as necessary, to reflect your security requirements.

What the security migration utility does

The security definition migration utility reads the records containing security definitions for users, tasks, and programs for a specified DC/UCF system from a 10.2 dictionary. It then creates the appropriate security statements to define resources and, as appropriate, grant the authority to use resources.

Specifically, the security migration utility reads security definitions from these records in a 10.2 DDLDML area:

- ACCESS-045
- SYS-041
- PROGLST-049
- TASKLST-023
- USER-047

The security definition migration utility is run in retrieval mode so that your 10.2 dictionary is not updated.

You should execute the utility against your 10.2 dictionary *before* migrating it to a 12.0 format.

What the security migration utility produces

For a specified DC/UCF system, the security definition migration utility produces the following statements:

- A CREATE RESOURCE SYSTEM SYST*nnnn* statement for the DC/UCF system processed.
- A CREATE USER statement for each user ID defined with SIGNON authority to the DC/UCF system.
- A CREATE SYSTEM PROFILE statement for each such user ID having INSTALLATION CODE or PRIORITY clauses assigned.
- A GRANT SIGNON ON SYSTEM SYST*nnnn* statement for each user having SIGNON authority to the DC/UCF system. A PROFILE clause is only included if the user ID has INSTALLATION CODE or PRIORITY clauses associated with it.

- A CREATE RESOURCE CATEGORY CAT_*nnn* statement for only those security classes assigned to a task or program.
- A GRANT EXECUTE ON CATEGORY CAT_*nnn* TO *user-id* statement for each user ID associated with the specified category (security class).
- A CREATE RESOURCE ACTIVITY DEFAULT.ACT_*nnn* NUMBER *nnn* statement for each CA ADS security class (1 through 255).
- A GRANT EXECUTE ON ACTIVITY ... TO *user-id* statement for each user ID associated with the specified activity (security class).

Sample output and a more detailed description of the output is provided later in this section.

How to execute the security definition migration utility

The security definition migration utility executes as program RHDCSMIG and uses the RHDCMIGA subschema.

Your 12.0 DMCL definition must contain the 10.2 file and area description for the 10.2 DDLDML area that will be processed by the security definition migration utility. The security migration utility only runs in a Release 12.0 environment.

You must execute the RHDCSMIG program against a 10.2 dictionary. You cannot run RHDCSMIG against a DDLDML area that has been migrated to a Release 12.0 format.

Conversion tools and environment

To use the security migration utility and generate your 12.0 central security definitions, you need:

This component	For these tasks
12.0 DMCL module	Run-time environment needed to execute the security migration utility (RHDCSMIG).
RHDCSMIG program and RHDCMIGA subschema	To convert security definitions.
10.2 DDLDML area containing security definitions to be converted	The RHDCSMIG program reads security definitions from a DDLDML area.
CA IDMS Command Facility	To populate the 12.0 user catalog and DDLDML area with modified 12.0 security statements.

Related CA documentation

You may need to refer to these documents during the process of migrating 10.2 security definitions:

- *CA IDMS Security Administration Guide*
- *CA IDMS Database Administration Guide*

Convert security definitions

To convert security definitions using the security definition migration utility, take these steps:

1. Identify the 10.2 DDLDML area that contains the security definitions to be converted.
2. Verify that this DDLDML area is defined in the 12.0 DMCL you'll use to execute the security definition migration utility.
3. Create execution JCL for the RHDCSMIG program.
4. Execute RHDCSMIG program.
5. Review security statements created by the RHDCSMIG program.
6. Modify security statements.
7. Submit security statements to the CA IDMS Command Facility to populate both the user catalog and the DDLDML area.

Some of these steps are discussed in more detail below.

Create execution JCL for the RHDCSMIG program

Runs in local mode or under the central version

The RHDCSMIG program can run either in local mode or under the central version. It is recommended that you run the program in local mode.

Input to RHDCSMIG

Input to RHDCSMIG is the *dc/ucf-version-number* of the DC/UCF system whose definition will be read by the RHDCSMIG program.

You specify the *dc/ucf-version-number* as a SYSIPT parameter as follows:

- DCSYSTEM=*nnnn*. There can be no intervening spaces between DCSYSTEM and *nnnn*.
- The DCSYSTEM keyword can begin in any column and must be the first record in the SYSIPT file.
- *Nnnn* must be an integer in the range 1 through 9999. Leading zeroes are not required.

The RHDCSMIG program produces a SYSPCH output file containing the 12.0 security statements.

An internal sort is performed during RHDCSMIG processing, so sort work files must be provided in the JCL.

The execution JCL required to run the RHDCSMIG is provided below.

z/VM command Details

```
FILEDEF SYSLST PRINTER
FILEDEF SORTMSG PRINTER
FI dictdb DISK vaddr
FI dcmsg DISK vaddr
FI SYSPCH DISK security syntax a
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK rhdcsmig input a
GLOBAL LOADLIB dbalib idmslib
OSRUN RHDCSMIG
```

dictdb	DDname of the 10.2 DDLDML area containing the security definitions to be converted. This is the ddname specified in your Release 12.0 DMCL.
dcmsg	DDname of the system message (DDLDCMSG) area
vaddr	Virtual address of the mini disk on which the file resides
security syntax a	File identifier of the file where the resultant 12.0 security syntax will be placed
sysidms input a	File identifier of the file containing the following: *dmcl-name* -- name of Release 12.0 DMCL *dbname or segment-name* -- database name or segment name identifying the DDLDML area to be migrated
rhdcsmig input a	File identifier of the file containing the RHDCSMIG input parameter: DCSYSTEM=*nnnn* -- An integer from 1 through 9999 that identifies the DC/UCF system whose definition will be read by the RHDCSMIG program. This is the *dc/ucf-version-number* assigned to the DC/UCF system when it was defined.
dbalib	Filename of the load library containing the DMCL and database name table load modules
idmslib	Filename of the load library containing executable Release 12.0 CA IDMS modules

Central version

To run the RHDCSMIG program under the central version, modify the local mode JCL as follows:

- Add a SYSCTL file
- Remove file definition for the DDLDML area

Output from RHDCSMIG

A description of the output from the security definition migration utility is provided below. You should review the output from your execution of the utility and modify it to reflect your security definition requirements. After the sample output is described, a checklist of items to review is provided.

CREATE RESOURCE SYSTEM statement

A CREATE RESOURCE SYSTEM SYST*nnnn* statement is created for the DC/UCF system identified on the DCSYSTEM SYSIPT control card.

Sample output

```
CREATE RESOURCE SYSTEM SYST0045;
```

CREATE USER statement

A CREATE USER statement is created for each user ID defined with SIGNON authority to the specified DC/UCF system. If the following information was part of the 10.2 USER definition, it will be included in the 12.0 user definition.

12.0 statement component	Created from 10.2 statement
DESCRIPTION '*description*'	DESCRIPTION IS clause of the IDD USER statement
NAME *user-name*	ALIAS IS clause of system generation USER statement or FULL NAME IS clause of IDD USER statement.
ENCRYPTED PASSWORD X'*nnnnnnnnnnnnnnnn*'	PASSWORD IS from either system generation or IDD USER statement. The contents of PASSWORD IS is migrated in its encrypted format (X'16-hex-digits'). Once the password is migrated, it can continue to be used as is or changed. The password is displayed in its encrypted format by RHDCSMIG.

CREATE SYSTEM PROFILE

A CREATE SYSTEM PROFILE *user-id_nnnn* statement is created for each user ID with SIGNON authority to the specified DC/UCF system. *Nnnn* is the DC/UCF system number.

The following information is included in the CREATE SYSTEM PROFILE statement if it was defined in the 10.2 USER definition.

12.0 statement component	Created from 10.2 statement
INSTCODE	INSTALLATION CODE from system generation or IDD USER statement.
PRIORITY	PRIORITY IS from system generation or IDD USER statement.

Sample CREATE USER and SYSTEM PROFILE output

```
CREATE USER "AAD260             "
   ENCRYPTED PASSWORD X'FA808D8EDB038200'
;
CREATE SYSTEM PROFILE "AAD260_0045       " ATTRIBUTES
   INSTCODE='                    AD260       ' OVERRIDE NO
;
CREATE USER "AAE120             "
   ENCRYPTED PASSWORD X'C0F2F9C551A73D12'
;
CREATE SYSTEM PROFILE "AAE120_0045       " ATTRIBUTES
   INSTCODE='                    AE120       ' OVERRIDE NO
;
CREATE USER "CULL DBA           "
   ENCRYPTED PASSWORD X'E57C572D47B0D600'
;
CREATE SYSTEM PROFILE "CULL DBA_0045     " ATTRIBUTES
   PRIORITY=050 OVERRIDE NO
;
```

GRANT SIGNON ON SYSTEM ... TO USER

A GRANT SIGNON ON SYSTEM SYST*nnnn* statement is created for each user ID with SIGNON authority to the specified DC/UCF system. This gives the user ID authority to sign on to the specified DC/UCF system.

The PROFILE *"user-id_nnnn"* clause is included if INSTALLATION CODE or PRIORITY clauses were part of the 10.2 user definition.

Sample output

```
GRANT SIGNON ON SYSTEM SYST0045
   PROFILE "AAD260_0045       "
   TO "AAD260             ";
GRANT SIGNON ON SYSTEM SYST0045
   PROFILE "AAE120_0045       "
   TO "AAE120             ";
GRANT SIGNON ON SYSTEM SYST0045
   PROFILE "CULL DBA_0045     "
   TO "CULL DBA           ";
```

CREATE RESOURCE CATEGORY

A CREATE RESOURCE CATEGORY CAT_*nnn* statement is created for each security class used by a task or program in the 10.2 dictionary. ADD TASK, ADD PROGRAM, and ADD LOAD MODULE statements are included for each CREATE RESOURCE CATEGORY statement.

If a security class is not used by a task or program, a CREATE RESOURCE CATEGORY statement is not created for the security class. However, if a security class is not used by any task or program and is associated with a user ID, a GRANT EXECUTE ON CATEGORY CAT_*nnn* TO *"user-id"* statement is still created by the security definition migration utility.

For more information, see "GRANT EXECUTE ON CATEGORY" below.

12.0 statement component	Created from 10.2 statement
CREATE RESOURCE CATEGORY CAT_*nnn*	SECURITY CLASS clause in TASK or PROGRAM statement.
ADD TASK	System generation and IDD ADD TASK statement and SECURITY CLASS clause.
ADD PROGRAM	

12.0 statement component	Created from 10.2 statement
	System generation PROGRAM statement for programs defined with either the DYNAMIC or LOADLIB parameters. RHDCSMIG creates a prefix to the program name as follows: Version 1 is prefixed with CDMSLIB. Other version numbers for the same program are prefixed with V*nnnn* where *nn nn* is the 10.2 version number for the program.
ADD LOAD MODULE	System generation PROGRAM statement for programs defined as either DYNAMIC with a type other than PROGRAM or with an associated DICTNAME. The RHDCSMIG program prefixes the load module name with V*nnnn* where *nnn n* is the 10.2 version number for the program.

GRANT EXECUTE ON CATEGORY

A GRANT EXECUTE ON CATEGORY CAT_*nnn* TO *"user-id"* statement is created for each user ID with assigned security classes. Note that this statement is created even if the security class is not assigned to a TASK or PROGRAM definition (and therefore will not have a corresponding CREATE RESOURCE CATEGORY statement).

Note: An error message is returned for those GRANT EXECUTE ON CATEGORYstatements (security classes) for which there is no preceding CREATE RESOURCE CATEGORY statement (not assigned to a task or program statement) and the dictionary is not populated with the GRANT EXECUTE statements. You can ignore the error messages or delete the GRANT EXECUTE statements.

Important! If signon definitions are maintained in a dictionary separate from the dictionary containing system generation definitions, you must run the RHDCSMIG program separately on both dictionaries and merge the output. In this case, do not delete any statements until after you merge and review the output from RHDCSMIG. For more information, see Review and modify output from RHDCSMIG (see page 39).

12.0 statement component	Created from 10.2 statement
GRANT EXECUTE ON CATEGORY CAT_*nnn* TO " *user-id"*	SECURITY CLASS clause of system generation USER statement or INCLUDE ACCESS of IDD ADD USER statement.
GRANT EXECUTE ON CATEGORY CAT_* TO *"use r-id"*	A wild-card character (*) is appended to the category name to assign all categories to a user ID if all security classes were previously assigned to the user.

Sample output

```
CREATE RESOURCE CATEGORY CAT_001
   ADD TASK ADS
   ADD TASK ASF
   ADD TASK ASFNT
   ADD TASK A303EIS
   ADD TASK CLIST
   ADD TASK DCUF
   ADD TASK ICMS
   ADD TASK ICMSCOMM
   ADD TASK IDB
   ADD TASK IDBCOMM
   ADD TASK IDD
   ADD TASK IDDM
   ADD TASK IDDMT
   ADD TASK IDDT
   ADD TASK OLQ
   ADD TASK OLQNT
   ADD TASK OLQT
   ADD TASK SEND
   ADD TASK SHOWMAP
   ADD TASK TCF
;
GRANT EXECUTE ON CATEGORY CAT_001 TO "AAD260            ";
GRANT EXECUTE ON CATEGORY CAT_001 TO "AAE120            ";
CREATE RESOURCE CATEGORY CAT_002
   ADD TASK DCMT
   ADD TASK DCPROFIL
;
GRANT EXECUTE ON CATEGORY CAT_002 TO "AAD260            ";
GRANT EXECUTE ON CATEGORY CAT_002 TO "AAE120            ";
 .
 .
 .

CREATE RESOURCE CATEGORY CAT_099
   ADD PROGRAM CDMSLIB.DBUGMAIN
   ADD PROGRAM   V0002.MYMAP
   ADD LOAD MODULE   V0002.MYMAP
   ADD TASK ADSA
   ADD TASK ADSAT
   ADD TASK ADSD
   ADD TASK ADSG
   ADD TASK ADSGT
   ADD TASK ADSOTATU
   ADD TASK DEBUG
   ADD TASK MRIDEFQ
   ADD TASK MRIMSGQ
   ADD TASK MRISTAQ
   ADD TASK OLM
   ADD TASK OLMT
;
```

CREATE RESOURCE ACTIVITY

A CREATE RESOURCE ACTIVITY DEFAULT.ACT_*nnn* NUMBER *nnn* is created for all security classes from 1 through 255. *Nnn* is the security class from the 10.2 dictionary.

GRANT EXECUTE ON ACTIVITY

The GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_*nnn* TO *"user-id"* statement gives the specified user ID EXECUTE authority on the specified CA ADS, DCMT, or Online debugger (DBUG) activity. *Nnn* is the security class from the 10.2 dictionary. A GRANT EXECUTE ON ACTIVITY... TO *"user-id"* statement is created for each user ID assigned the specified security class.

12.0 statement component	**Created from 10.2 statement**
GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_*nnn* to *"user-id"* You may wish to refine the access to the activity to a particular CA ADS application, the DCMT command facility, or the Online debugger. If so, define additional activities in which you replace DEFAULT with the name of the application: CA ADS DCMT DBUG	SECURITY CLASS clause of system generation ADD USER statement or INCLUDE ACCESS of IDD ADD USER statement

Sample output

```
CREATE RESOURCE ACTIVITY DEFAULT.ACT_001 NUMBER 001;
GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_001 TO "AAD260            ";
GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_001 TO "AAE120            ";
CREATE RESOURCE ACTIVITY DEFAULT.ACT_002 NUMBER 002;
GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_002 TO "AAD260            ";
GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_002 TO "AAE120            ";
CREATE RESOURCE ACTIVITY DEFAULT.ACT_003 NUMBER 003;
GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_003 TO "AAD260            ";
GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_003 TO "AAE120            ";
CREATE RESOURCE ACTIVITY DEFAULT.ACT_004 NUMBER 004;
GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_004 TO "AAD260            ";
GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_004 TO "AAE120            ";
CREATE RESOURCE ACTIVITY DEFAULT.ACT_005 NUMBER 005;
GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_005 TO "AAD260            ";
GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_005 TO "AAE120            ";
          ...
CREATE RESOURCE ACTIVITY DEFAULT.ACT_243 NUMBER 243;
CREATE RESOURCE ACTIVITY DEFAULT.ACT_244 NUMBER 244;
CREATE RESOURCE ACTIVITY DEFAULT.ACT_245 NUMBER 245;
CREATE RESOURCE ACTIVITY DEFAULT.ACT_246 NUMBER 246;
CREATE RESOURCE ACTIVITY DEFAULT.ACT_247 NUMBER 247;
CREATE RESOURCE ACTIVITY DEFAULT.ACT_248 NUMBER 248;
CREATE RESOURCE ACTIVITY DEFAULT.ACT_249 NUMBER 249;
CREATE RESOURCE ACTIVITY DEFAULT.ACT_250 NUMBER 250;
CREATE RESOURCE ACTIVITY DEFAULT.ACT_251 NUMBER 251;
CREATE RESOURCE ACTIVITY DEFAULT.ACT_252 NUMBER 252;
CREATE RESOURCE ACTIVITY DEFAULT.ACT_253 NUMBER 253;
CREATE RESOURCE ACTIVITY DEFAULT.ACT_254 NUMBER 254;
CREATE RESOURCE ACTIVITY DEFAULT.ACT_255 NUMBER 255;
GRANT EXECUTE ON ACTIVITY DEFAULT.ACT_*   TO "CULL DBA          ";
```

Review and modify output from RHDCSMIG

You should review the 12.0 security statements created by RHDCSMIG and modify them to reflect your security requirements. Here are some considerations:

1. Under 10.2 security, all versions of a program had the same security class. Under 12.0 security, each version of a program is controlled separately. You may need to add additional PROGRAM or LOAD MODULE clauses to your CATEGORY definitions in order to achieve identical results. You should also consider using the wild-card capability to reduce the number of definitions and to provide for load modules which are defined dynamically.

2. Under 10.2 security, all tasks and programs for which no security class was assigned were considered "unsecured," indicating that anyone could execute them. To achieve similar results under 12.0, you can issue the following statements:

   ```
   CREATE RESOURCE CATEGORY category-name  ADD PROGRAM *
     ADD TASK *
     ADD LOAD MODULE *;

   GRANT EXECUTE ON CATEGORY category-name to PUBLIC;
   ```

3. If you maintain user signon definitions in a dictionary separate from the dictionary containing system generation definitions, you need to run the RHDCSMIG program once against each dictionary and then merge the output files.
   Review the combined output and make the necessary changes to the combined file.
   For example, you will have duplicate CREATE SYSTEM RESOURCE statements and CREATE RESOURCE ACTIVITY statements.

4. Review the generated system name. It should be the same as the system name identified on the new system generation SYSTEM statement SYSTEM ID IS parameter. The dictionary migration utility will generate a value for the SYSTEM ID IS parameter as follows:

   - If a 10.2 system generation DDS statement LOCAL NODE IS clause is defined, the value specified on this clause will become the 12.0 system name or

   - If a 10.2 DDS statement is not specified, the system name defaults to SYST*nnnn* where *nnnn* is the value specified as *dc/ucf-version* on the system generation ADD SYSTEM statement

   The system name on the CREATE RESOURCE SYSTEM statement and the SYSTEM ID IS parameter of the system generation SYSTEM statement must be the same value.

5. Review system profile names and category names created by RHDCSMIG. If necessary, change the names generated to reflect your naming standards.
   If multiple users have the same priority and installation code values, consider having those users share a single profile.

6. To avoid errors, identify any GRANT EXECUTE CATEGORY statements for which there is no corresponding CREATE RESOURCE CATEGORY statement and delete them.

7. Review CREATE RESOURCE ACTIVITY and corresponding GRANT EXECUTE ON ACTIVITY statements. If you want to refine the access to the activity to a particular CA ADS application, DCMT command, or the online debugger, define additional sets of activities in which you replace the prefix DEFAULT with the name of the application (CA ADS, DCMT, or DBUG).

8. If multiple DC/UCF systems are defined in the same dictionary, they will all have the same security characteristics. A task or program is associated with the same category in all systems and users have the right to execute the same categories in all systems. If this is not desired, you must define the systems in separate dictionaries.

Populate 12.0 system with security definitions

Submit statements to the command facility

To populate the DDLDML area and the user catalog with your security definitions, submit the statements to the CA IDMS Command Facility.

All security statements except the CREATE USER statement are stored in the DDLDML area. CREATE USER statements are stored in the user catalog.

Connect to the system dictionary

For security definitions to be effective, you must access the SYSTEM dictionary when submitting statements. Be sure you are connected to the SYSTEM dictionary.

User definitions are automatically stored in the user catalog through a connect that is hard-coded in the CA IDMS software.

Note: For more information about using the CA IDMS Command Facility, see the *CA IDMS Command Facility Guide*.

Review listing from the command facility

Review the listing produced by the Command Facility and, if necessary, correct any errors.

Security Display Facility

To generate a report from the user catalog on the security definition for a DC/UCF system, you can use the Security Display Facility to generate an online or batch report.

Note: For more information about using the Security Display Facility, see the *CA IDMS Security Administration Guide*.

Dictionary Migration and Setup

This section describes the changes to the 12.0 dictionary environment and provides guidelines for migrating 10.0 dictionaries and setting up a 12.0 dictionary environment.

Migrate only the DDLDML Area

The DDLDML area has changed and you must migrate it to the new Release 12.0 format. CA IDMS provides a dictionary migration utility to migrate 10.0 DDLDML areas to their 12.0 format. For more information about the changes to the dictionary, see the *CA IDMS Dictionary Structure Reference section*.

Unchanged Dictionary Areas

These dictionary areas have not changed and you can use them in a 12.0 environment without any *structural* changes:

- DDLDCLOD
- DDLDCMSG

⚠

⚠

New Dictionary Areas

There are new dictionary areas and they are installed during the CA IDMS installation process. They are:

- DDLCAT -- Contains physical database entities. If you have the SQL Option, it will also contain SQL database entities.
- DDLCATX -- Contains indexes associated with DDLCAT entities
- DDLCATLOD -- Contains load modules for DMCLs, database name tables and access modules (SQL Option only)

For more information, see the following topics:

- About dictionary migration (see page 42)
- Planning to migrate dictionaries (see page 44)
- Migrate dictionaries (see page 45)
- Run Release 12.0 IDMSDIRL utility (see page 49)
- Update the Task Application Table (see page 50)
- Set up 12.0 dictionary environment (see page 50)
- Modify schemas and regenerate subschemas (see page 52)

About dictionary migration

The dictionary migration utility

CA IDMS provides a dictionary migration utility that migrates the DDLDML area of Release 10.0 data dictionaries to the new 12.0 format.

What the dictionary migration utility does

The dictionary migration utility makes these changes to DDLDML areas:

- Deletes records, elements, and set structures no longer needed in the 12.0 environment
- Restructures record occurrences from a 10.0 format to a 12.0 format
- Adds new record elements and set structures

The dictionary migration utility:

- Migrates only the DDLDML area of Release 10.0 data dictionaries
  You cannot migrate dictionaries prior to Release 10.0 with the 12.0 dictionary migration utility.
- Executes in a Release 12.0 environment only
- Executes in local mode only
- Executes as two programs:
  - Program RHDCMIG1 -- Deletes and modifies records, elements, and set pointers
  - Program RHDCMIG2 -- Deletes and modifies records
- **Does nothing to CA IDMS/DB databases. Do not migrate your CA IDMS/DB databases.**

When to migrate dictionaries

Once you have processed your 10.2 DMCL definitions using the DMCL Syntax Generator and produced your 12.0 security definitions using the RHDCSMIG utility program, you can then migrate the DDLDML area of your 10.0 data dictionaries. For details on using the DMCL Syntax Generator, see Physical Database Definition (see page 9).

Conversion tools and environment

You will need the following components to migrate 10.0 DDLDML areas:

This component	For these tasks
RHDCMIG1 and RHDCMIG2 programs and RHDCMIGA and IDMSNWKA (12.0 version) subschemas. These modules are installed during the installation process and should exist in the load (core-image) library that contains CA IDMS executable modules. RHDCMIGA is a subschema load module used by RHDCMIG1 that describes a 10.0 dictionary. IDMSNWKA is a 12.0 subschema that describes the 12.0 DDLDML area.	To migrate DDLDML areas.
Release 10.0 DDLDML areas.	

This component	For these tasks
	These are the dictionary areas the migration utility will migrate. You must define these DDLDML areas in the 12.0 DMCL you'll use to run the migration utility.
Release 12.0 DMCL load module, 12.0 database name table (if one is named by the 12.0 DMCL), and the DBA and LOADLIB load libraries.	To run RHDCMIG1 and RHDCMIG2 in a 12.0 environment.

Related CA documentation

You may need to refer to these documents during the process of migrating dictionaries:

- *CA IDMS Database Administration Guide*
- *CA IDMS Utilities Guide*
- *CA IDMS Installation and Maintenance Guide -- z/OS*

Planning to migrate dictionaries

Before you migrate any dictionaries, make sure you have the correct 10.2 and 12.0 environments set up to properly migrate DDLDML dictionary areas.

Here are some guidelines.

1. If the integrity of any of your DDLDML areas is questionable, you may want to run IDMSDBAN against them to verify that there are no integrity problems. The dictionary migration utility will not run against a DDLDML area with broken chains.
2. Evaluate the space available in each 10.0 DDLDML area you will migrate by reviewing the file utilization report produced by the IDMSDUMP utility.
3. If you need more space, expand the size of the area by either increasing the page size using IDMSXPAG or increasing the page size and/or page range within the area by using the IDMSUNLD and IDMSDBLU utilities.
   The dictionary migration process **does not** require that you increase the size of the DDLDML. However, IDMSDIRL will require approximately 5% additional space.
   If you increase the size of the DDLDML area, remember to modify your 12.0 DMCL to reflect this change.
4. Backup each DDLDML area you plan to migrate.
5. Make sure the Release 12.0 DMCL you'll be using to execute the dictionary migration utility includes the segments containing the DDLDML areas you will migrate. The name of each DDLDML area must be DDLDML. If you are migrating multiple DDLDML areas, they must be associated with different segments.

Migrate dictionaries

The dictionary migration process

Once you've prepared your environment to migrate your dictionaries, perform these steps:

1. Backup the DDLDML area to be migrated if you haven't done so already.
2. Code execution JCL for RHDCMIG1.
3. Run RHDCMIG1 against the 10.0 DDLDML area. RHDCMIG1 runs from the 12.0 load (core-image) library that contains CA IDMS executable load modules.
4. Verify results by reviewing the transaction summary.
5. If the DDLDML area you are migrating is large or in your estimation took a long time to migrate during phase I of the migration process, you may want to backup the area before running RHDCMIG2.
6. Code execution JCL for RHDCMIG2.
7. Run RHDCMIG2.
8. Verify results by reviewing messages provided by RHDCMIG2.
9. If the DDLDML area you've just migrated contains the IDMSNTWK schema, run the 12.0 IDMSDIRL utility against it once RHDCMIG1 and RHDCMIG2 have successfully executed.

What's next

The remainder of this section presents

- Execution JCL to run RHDCMIG1 and RHDCMIG2
- Guidelines for running Release 12.0 IDMSDIRL
- Guidelines for setting up the rest of your dictionary environment

Running RHDCMIG1

The execution JCL to run RHDCMIG1 in local mode is provided below.

JCL - z/OS execution

To run RHDCMIG1 in local mode, code the following execution JCL.

```
//RHDCMIG1  EXEC  PGM=RHDCMIG1,REGION
//STEPLIB   DD    DSN=idms.dba.loadlib,DISP=SHR
//          DD    DSN=idms.loadlib,DISP=SHR
//dictdb    DD    DSN=cdms.dictdb,DISP=SHR
//dcmsg     DD    DSN=idms.sysmsg.ddldcmsg,DISP=SHR
```

```
//SYSLST    DD    SYSOUT=A
//SYSUDUMP  DD    SYSOUT=A      OPTIONAL
//SYSIDMS   DD    *

Insert SYSIDMS parameters here. Code them on the same or different
lines.

DMCL=dmcl-name
DBNAME=dbname or segment-name
```

idms.dba.loadlib	Data set name of the newly installed CA IDMS/DB Release 12.0 load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the load library containing the executable Release 12.0 modules
dictdb	DDname of the 10.0 DDLDML area you will migrate. This is the ddname specified in your Release 12.0 DMCL.
cdms.dictdb	Data set name of the 10.0 DDLDML area you will migrate
dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	Data set name of the system message (DDLDCMSG) area
SYSIDMS	Name of the parameter file where you can specify run-time directives and operating system-dependent parameters. For more information about the SYSIDMS parameter file, see the *CA IDMS Database Administration Guide*.
dmcl-name	Name of the release 12.0 DMCL
dbname or segment name	Database or segment name identifying the DDLDML area to be migrated

z/VSE execution JCL Code

To run RHDCMIG1 in a z/VSE environment in local mode, code the following execution JCL.

```
// JOB    RHDCMIG1
// LIBDEF *,SEARCH=r120lib
// DLBL dictdb,'cdms.dictdb',da
// EXEC   PROC=IDMSLBLS
// EXEC   RHDCMIG1,SIZE=1024K
DMCL=dmcl-name
DBNAME=dbname or segment-name
/*
```

IDMSLBLS	Name of the procedure provided at installation that contains the file definitions for CA IDMS dictionaries, databases, disk journal files, and the SYSIDMS file. **Note:** For a complete listing of IDMSLBLS, see IDMSLBLS Procedure for z/VSE JCL.
dictdb	Filename of the 10.0 DDLDML area you will migrate
cdms.dictdb	File-ID of the 10.0 DDLDML area you will migrate area
dmcl-name	Name of the 12.0 DMCL

dbname or segment-name	Name of the database or segment containing the DDLDML area to be migrated

About z/VM Commands

To run RHDCMIG1 in a z/VM environment in local mode, code the following commands.

```
FILEDEF SYSLST PRINTER
FI dictdb DISK cdms dictdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FI dcmsg DISK cdms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FILEDEF SYSIDMS DISK sysidms input a
GLOBAL LOADLIB dbalib idmslib
OSRUN RHDCMIG1
```

dictdb	DDname of the 10.2 DDLDML area you will migrate. This is the ddname specified in your Release 12.0 DMCL.
cdms dictdb fm	File identifier of the 10.2 DDLDML area you will migrate
dcmsg	DDname of the system message (DDLDCMSG) area
cdms dmsgdb fm	File identifier of the system message (DDLDCMSG) area
ppp	Page size of the database file
nnn	Number of pages in the database file
SYSIDMS	Name of the CA-supplied parameter file where you can specify run-time directives and operating system-dependent parameters. For a complete description of the SYSIDMS parameter file, see the *CA IDMS Database Administration Guide*.
sysidms input a	File identifier of the file containing the following: *dmcl-name* -- name of Release 12.0 DMCL *segment-name* -- Database name or segment name identifying the DDLDML area to be migrated
dbalib	Filename of the load library containing the DMCL and database name table load modules for Release 12.0
idmslib	Filename of the load library containing executable Release 12.0 CA IDMS modules

Verify the results of RHDCMIG1

If RHDCMIG1 ran successfully, it will produce a transaction summary that:

- Lists the number of records found and modified for each record type affected by the restructure
- This message indicates it ran successfully:

```
RHDCMIG1    RELEASE 12.0
SUBS=RHDCMIGA
recname       FOUND: nnn          MODIFIED: nnn
    .
    .
    .
END OF 12.0 MIGRATION, PHASE I
```

Possible errors from RHDCMIG1

If RHDCMIG1 detects a database error, all CA IDMS/DB status fields are displayed and an abend dump taken. Note that if a database error occurs, the dictionary you are migrating will probably be corrupted (unless the error occurs during a BIND). You will have to restore the DDLDML area from a backup. Correct the problem before restarting RHDCMIG1.

Abend code

Possible abend code:

Abend	Description
2998	SYSLST cannot be opened. Check JCL.

Running RHDCMIG2

You may want to take a backup of the DDLDML area before running RHDCMIG2.

The execution JCL for running RHDCMIG2 in local mode is provided below.

z/VM commands 3

To run RHDCMIG2 in a z/VM environment in local mode, code the following commands.

```
FILEDEF SYSLST PRINTER
FI dictdb DISK cdms dictdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FI dcmsg DISK cdms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FILEDEF SYSIDMS DISK sysidms input a
GLOBAL LOADLIB dbalib idmslib
OSRUN RHDCMIG2
```

dictdb	DDname of the 10.2 DDLDML area you will migrate. This is the ddname specified in your Release 12.0 DMCL.
cdms dictdb fm	File identifier of the 10.2 DDLDML area you will migrate
dcmsg	DDname of the system message (DDLDCMSG) area
cdms dmsgdb fm	File identifier of the system message (DDLDCMSG) area
ppp	Page size of the database file
nnn	Number of pages in the database file
SYSIDMS	Name of the parameter file where you can specify run-time directives and operating system-dependent parameters.

	Note: For more information about the SYSIDMS parameter file, see the *CA IDMS Database Administration Guide*.
sysidms input a	File identifier of the file containing the following: *dmcl-name* -- name of Release 12.0 DMCL *segment-name* -- database name or segment name identifying the DDLDML area to be migrated
dbalib	Filename of the load library containing the DMCL and database name table load modules for Release 12.0
idmslib	Filename of the load library containing executable Release 12.0 CA IDMS modules

Verify the results of RHDCMIG2

If RHDCMIG2 ran successfully it will produce this message:

```
RHDCMIG2 RELEASE 12.0
SUBS=IDMSNWKA
END OF 12.0 MIGRATION, PHASE II
```

Possible errors from RHDCMIG2

If RHDCMIG2 detects a database error, it displays all CA IDMS/DB status fields and an abend dump is taken. Note that if a database error occurs, the dictionary you are migrating will probably be corrupted (unless the error occurs during a BIND). You will have to restore the DDLDML area from a backup. Correct the cause of the error before restarting RHDCMIG2.

Possible abend code:

Abend	Description
2998	SYSLST cannot be opened. Check JCL.

Run Release 12.0 IDMSDIRL utility

You must run the 12.0 IDMSDIRL utility against a migrated DDLDML area that contains the IDMSNTWK schema, if a 12.0 version of IDMSNTWK does not exist in your 12.0 environment.

The IDMSDIRL utility will delete the 10.0 IDMSNTWK schema, if present, as well as all elements and records associated with the 10.0 IDMSNTWK schema before adding the CA-supplied internal schema definitions for IDMSNTWK, IDMSSECS, and IDMSSECU, and all of the related element and record definitions and subschemas associated with these schemas.

Note: Only one dictionary per DC/UCF system needs to contain the IDMSNTWKschema. The SYSDIRL dictionary, created during the installation process, contains the IDMSNTWK schema.

Before you run the IDMSDIRL utility:

1. Punch the source for any user subschemas you want to save that are associated with the IDMSNTWK schema.
   IDMSDIRL will delete the 10.0 IDMSNTWK schema and all associated subschemas and add the 12.0 IDMSNTWK schema.

2. Backup the newly migrated 12.0 DDLDML area. IDMSDIRL may run longer than the dictionary migration utility. Taking a backup before you run IDMSDIRL will save considerable time if IDMSDIRL fails.

Note: For more information about the guidelines on running the IDMSDIRL utility, see the *CA IDMS Utilities Guide*.

Update the Task Application Table

For each application dictionary that you migrate from Release 10.2 to Release 12.0, you must update the Task Application Table (TAT) with a new 12.0 application, $TOOLTCF. $TOOLTCF contains the ADSA application structure for all of the Release 12.0 tools compilers (that is, ADSA, ADSC, and MAPC.) You can update the table either online using the ADSOTATU task code or in batch using the ADSOBTAT utility. $TOOLTCF should have been linked into your 12.0 load library as part of the installation process.

Note: For more information about the ADSOTATU task and the ADSOBTAT utility, see the *CA ADS Reference Guide*.

Before you add the $TOOLTCF application into your secondary dictionaries, you must delete the $ACF@GEN application. These two applications share common task codes such as ADSA; therefore, if you add $TOOLTCF before you remove $ACF@GEN, an error will occur.

Set up 12.0 dictionary environment

This section describes how to set up your Release 12.0 dictionary environment once you have migrated your dictionaries. This includes setting up a system dictionary and your application dictionaries.

Setting up a system dictionary

Create a separate system dictionary

It is recommended that you separate the system dictionary from your application dictionaries; reserving it explicitly for system startup information. Maintaining a separate system dictionary, ensures that it is updated only by authorized staff in a controlled manner. This is particularly critical since the system dictionary is also required for local mode processing if CA IDMS security is used.

Name system dictionary SYSTEM

You must name the dictionary DC/UCF uses to retrieve system definitions SYSTEM. This means that either the segment name of the DDLDML and DDLDCLOD areas has a name of SYSTEM or there is a database name entry of SYSTEM that includes the relevant segment, in the database name table.

How to set up a separate system dictionary

If your DC/UCF system definitions are currently in an application dictionary, take the following steps to move them to a separate system dictionary:

1. Define segments for the system dictionary. The SYSTEM segment should contain the DDLDML and DDLDCLOD areas. The CATSYS segment should contain the DDLCAT, DDLCATX, and DDLCATLOD areas in which your DMCL is defined.

2. Update your DMCL to include the new segments.

3. Generate, punch, and link edit the DMCL module.

4. Update your database name table to define the SYSTEM DBNAME and associate the segments defined in step 1 and the SYSMSG segment with it.

5. Format the new files that will contain the system dictionary.

6. Migrate the application dictionary that contains the system definition, if you have not done so already.

7. Punch the system definition as syntax from the migrated application dictionary.

8. Load the dictionary classes and attributes and DC device definitions into the new system dictionary using IDMSDDDL and source members DLODDEFS and DLODDCDV from the installed source library as input to IDMSDDDL.

9. Load CA IDMS 12.0 task and program modules into the new system dictionary using IDMSDDDL and the appropriate source members from the installed source library as input to IDMSDDDL. See DC/UCF System Generation and Startup (see page 57) for a list of the source library member names.

10. If you haven't already done so, load CA IDMS 12.0 messages into the message area using IDMSDDDL and source members DLODMSG1, DLODMSG2, DLODMSG3, DLODMSG4, DLODMSG5, and DLODMSG6 from the installed source library as input to the compiler.

11. Define your DC/UCF system in the new system dictionary using the system generation compiler and the punched system definition created in step 7. Refer to DC/UCF System Generation and Startup (see page 57) to review other changes you might make to your DC /UCF system definition before generating it in a 12.0 environment.

12. Delete the DC/UCF system definition from your application dictionary.

Setting up application dictionaries

To complete the set up of your application dictionary environment, you should load required dictionary modules, records, and other components into your application dictionary. You use the DDDL compiler and the source library members listed below as input to the compiler to load these required components.

Depending on the products you have installed, some of these source library members are:

- ADSRECDS -- For use with CA ADS
- ADSRECSQ -- For use with CA ADS and SQL Option
- DLODAGN2 -- For use with CA ADS/Generator
- DLODIRPT -- For use with CA IDMS ASF and CA ICMS
- DLODPROT -- Database protocols
- DLODDEFS -- Classes and attributes
- DLODAIDN -- A record used to define PF or PA key values

Updating the load (DDLDCLOD) area

Depending upon the products you have installed, you should update the DDLDCLOD area with new copies of the CA IDMS load modules provided at installation.

These modules are:

- RHDCEVBF -- For use with CA IDMS/DB, CA IDMS/DC, and CA ADS applications
- ASFIDB -- For use with CA IDMS ASF and CA ICMS

What's next

The remainder of this section presents what you need to do to 10.2 schemas and subschemas to use them in a 12.0 environment.

Modify schemas and regenerate subschemas

Modify schemas

Once you have migrated DDLDML areas, you may want to modify schema source statements to take advantage of new 12.0 features. For example, you may want to use symbolic parameters. You should modify schema source statements before generating subschemas in a 12.0 environment.

If your 10.0 schemas contain record definitions that specify page range restrictions using explicit page numbers, the page numbers are converted to page offsets by the 12.0 dictionary migration process. You should evaluate the use of symbolic parameters specifying these offsets in pages or as percentages, if these records participate in a multiple dictionary environment.

If any of the schemas in your migrated dictionary are used only to define a global DMCL, that is, they *are not associated with any subschemas,* you may want to delete them from the dictionary.

Regenerate subschemas

Whether you modify schema source statements or not, you should generate 10.0 subschemas in a 12.0 environment before using them in a 12.0 environment. You can regenerate subschemas in any of these ways:

- Use the REGENERATE ALL SUBSCHEMAS statement for each of your schemas. This is the recommended method if you want to regenerate all subschemas associated with a schema and if you want to use the new 12.0 features associated with the separation of logical and physical database definitions.

  Note: Area page range information will be removed from all subschemas whenyou generate them in a 12.0 environment. For more information about regenerating subschemas with the REGENERATE ALL SUBSCHEMAS statement, see the *CA IDMS Database Administration Guide*.

- Use the subschema compiler to generate individual subschemas in a 12.0 environment. This method is recommended if you need to selectively regenerate subschemas.

  Note: For more information about regenerating subschemas using the subschema compiler, see the *CA IDMS Database Administration Guide*.

- Use the IDMSSCON utility to convert 10.0 subschema load modules to 12.0 subschema load modules. This method is recommended for regenerating subschema load modules when the source statements don't exist.

  Note: IDMSSCON *does not* modify the contents of subschemasource. Therefore, 10.0 area page range information remains in subschemas generated using IDMSSCON.

  Guidelines for using IDMSSCON are presented below.

- In a 12.0 run-time environment, when a 10.0 subschema that you do not regenerate is accessed, CA IDMS will automatically convert the subschema to a 12.0 format. The converted subschema load module is not placed in a load area or load (core-image) library. It only exists for the duration of the run unit.

Important! This is a very costly method for temporarily regenerating a subschema load module and is only provided to assist you in your conversion to Release 12.0.

Using IDMSSCON to convert 10.0 subschema load modules

The IDMSSCON utility will convert 10.0 subschema load modules to 12.0 subschema load modules.

Here are some considerations when using IDMSSCON:

- Use IDMSSCON if you don't have subschema source statements. IDMSSCON only converts 10.0 load modules to 12.0 load modules and does not use any new 12.0 schema features.
- IDMSSCON resides in the 12.0 load (core-image) library containing CA IDMS executable modules.
- IDMSSCON runs in local mode.
- You can convert more than one subschema load module at a time.
- 10.0 subschema load modules must be in a load (core-image) library.
- IDMSSCON will convert 10.0 subschema load modules to a 12.0 format and place them in the DDLDCLOD area of a dictionary for subsequent punching and link-editing.

Running IDMSSCON

These are the requirements for running IDMSSCON:

- You specify the 10.0 subschema name (OLD SUBSCHEMA) and the 12.0 subschema name (NEW SUBSCHEMA) as SYSIPT parameters.
- The keyword OLD must begin in column 1 and the keyword NEW must begin in column 24. If you use the optional comma between old subschema name and new subschema name, it must be in column 23.
- Code each input statement on the same line
- You can convert more than one subschema at a time
- Old and new subschema names can be the same

z/OS execution - JCL

To run IDMSSCON in an z/OS environment in local mode, code the following execution JCL.

```
//IDMSSCON  EXEC PGM=IDMSSCON,REGION
//STEPLIB   DD   DSN=idms.dba.loadlib,DISP=SHR
//          DD   DSN=idms.loadlib,DISP=SHR
            DD   DSN=dbdc.r102.loadlib,DISP=SHR
//dloddb    DD   DSN=idms.appldict.ddldclod,DISP=SHR
//dcmsg     DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR        OPTIONAL
//SYSLST    DD   SYSOUT=A
```

```
//SYSIDMS   DD   *
Insert SYSIDMS parameters here

DMCL=dmcl-name
DICTNAME=dbname or segment-name containing dictionary load area

//SYSIPT    DD   *

Insert SYSIPT statements here

OLD SUBSCHEMA=10.0 subschema name,NEW SUBSCHEMA=12.0 subschema name
```

idms.loadlib	Data set name of the newly installed CA IDMS/DB Release 12.0 load library containing executable Release 12.0 modules
idms.dba.loadlib	Data set name of the newly installed CA IDMS/DB Release 12.0 load library containing the DMCL and database name table load modules
dbdc.r102.loadlib	Data set name of the load library containing 10.0 subschema load modules that you're converting with IDMSSCON
dloddb	DDname of the application dictionary load (DDLDCLOD) area where the converted subschema load module will be placed by IDMSSCON
idms.appldict.ddldclod	Data set name of the Release 12.0 application dictionary load (DDLDCLOD) area
dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	Data set name of the system message (DDLDCMSG) area
SYSIDMS	Name of the parameter file where you can specify run-time directives and operating system-dependent parameters. **Note:** For more information about the SYSIDMS parameter file, see the *CA IDMS Database Administration Guide*.
dmcl name	The name of the 12.0 DMCL load module to use to run IDMSSCON
dictname	Database name or segment name containing the DDLDCLOD area

z/VSE execution of JCL

To run IDMSSCON in a z/VSE environment in local mode, code the following execution JCL.

```
// JOB   IDMSSCON
// LIBDEF *,SEARCH=120 libraries and 10.2 load library
// EXEC PROC=IDMSDBLS
// EXEC   IDMSSCON,SIZE=1048K
DMCL=dmcl-name
DBNAME=dbname or segment-name containing dictionary load area
/*
OLD SUBSCHEMA=10.0 subschema-name,NEW SUBSCHEMA=12.0 subschema-name
/*
```

12.0 libraries and 10.2 load library	Release 12.0 library and the 10.2 load library containing the subschema load module to be converted
IDMSLBLS	Name of the procedure provided at installation that contains the file definitions for CA IDMS dictionaries, databases, disk journal files, and the SYSIDMS file.

	Note: For a complete listing of IDMSLBLS, see Appendix C, IDMSLBLS Procedure for z/VSE JCL.
dblod	Filename of the load library containing 10.0 subschema load modules to be converted with IDMSSCON
cdms.dictdb	File-ID of the 10.0 DDLDML area you will migrate area
dmcl name	The name of the 12.0 DMCL load module to use to run IDMSSCON
dictname	Database name or segment name containing the DDLDCLOD area

z/VM commands 4

To run IDMSSCON in a z/VM environment in local mode, code the following commands.

IDMSSCON (z/VM)

```
FILEDEF SYSLST PRINTER
FI dloddb DISK cdms appldlod fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FI dcmsg DISK cdms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK idmsscon input a
GLOBAL LOADLIB dbalib idmslib 102idms
OSRUN IDMSSCON
```

dloddb	DDname of the application dictionary load (DDLDCLOD) area where the converted subschema load module will be placed by IDMSSCON
cdms appldlod fm	File identifier of the 12.0 application dictionary load (DDLDCLOD) area
dcmsg	DDname of the system message (DDLDCMSG) area
cdms dmsgdb fm	File identifier of the system message (DDLDCMSG) area
ppp	Page size of the database file
nnn	Number of pages in the database file
sysidms input a	File identifier of the file containing the following: dmcl-name -- name of Release 12.0 DMCL segment-name -- database name (DBNAME) or segment name identifying the DDLDCLOD area
idmsscon input a	File identifier of the file containing the IDMSSCON input parameters OLD SUBSCHEMA=10.2 subschema name, NEW SUBSCHEMA=12.0 subschema name
dbalib	Filename of the load library containing the DMCL and database name table load modules
idmslib	Filename of the load library containing executable Release 12.0 CA IDMS modules
102idms	Filename of the load library containing 10.2 subschema load modules that you're converting with IDMSSCON

Verifying results of IDMSSCON

Using the IDMSLOOK utility

If IDMSSCON has executed successfully, you will get a zero return code. Additionally, you may want to review a report on the contents of any of the subschema load modules IDMSSCON converts. You can use the IDMSLOOK utility program to generate a report on the contents of any subschema load module.

Note: For more information about the IDMSLOOK utility, see the *CA IDMS Utilities Guide*.

Possible errors produced by IDMSSCON

Here are possible errors that you could get from running the IDMSSCON utility:

- OLD SUBSCHEMA=XXXXXXXX is a required parameter starting in column 1. Correct input parameters and resubmit job.
- NEW SUBSCHEMA=XXXXXXXX is a required parameter starting in card column 24. Correct input parameters and resubmit job.
- Unable to load the old subschema from the load library specified.
- Old subschema load module is invalid.

DC/UCF System Generation and Startup

This section describes:

- Guidelines for reviewing your migrated system definition before you generate it in a 12.0 environment
- The changes the dictionary migration utility made to your migrated system definition
- Guidelines for modifying DC/UCF startup JCL

For more information about CA IDMS Release 12.0 system generation and start up, see the *CA IDMS System Generation section* and *CA IDMS System Operations section*.

For more information, see the following topics:

- Modify migrated system definition (see page 58)

- Prepare your DC/UCF environment for startup (see page 63)
- Review and modify startup components (see page 64)

Modify migrated system definition

Here are some guidelines for reviewing and modifying your migrated system definition before you generate it in a 12.0 environment:

1. Review migrated system definition.
2. Review the statements and parameters the dictionary migration utility added or modified. These are listed below.
3. Change the values specified, if they don't meet your needs.
4. Verify that the information deleted from 10.2 system generation statements and parameters has been defined elsewhere. These are listed below.
5. Add new 12.0 statements or parameters that are not part of your migrated system definition. These are described below.
6. Change the SVC number to the SVC number of your new 12.0 SVC.
7. Add new CA IDMS task and program definitions to your migrated system definition.

The tables on the next few pages list the changes the dictionary migration utility made to system definitions and identifies where system definition information that has been replaced by a 12.0 function should be specified.

Changes made by the dictionary migration utility

The dictionary migration utility made the following changes to the SYSTEM statement.

Deleted parameters

Parameter	Change
BLDL/NOBLDL	Not needed
LOOKAROUND TIME IS	Not needed so not replaced
PREEMPTION THRESHOLD IS	Replaced by AREA ACQUISITION THRESHOLD
REGISTRATION	Replaced by 12.0 RUNUNIT security
RULOCKS	Not needed so not replaced
RUNUNITS FOR EXTENT	Extent areas replaced by standard areas

Modified parameters

RUNUNITS FOR SIGNON /DEST	Now specified as RUNUNITS FOR SIGNON. DEST is specified on RUNUNITS FOR SYSTEM/DEST clause.
UNDEFINED PROGRAM COUNT IS FOR	If you specified ALL, each valid 10.2 program type is listed. If you want to specify the new program type of ACCESS MODULE (SQL Option only), either Explicitly add ACCESS MODULE or Replace existing parameters with ALL

Parameters added to your system definition

These parameters have been added to your migrated system definition:

- SYSTEM ID IS
  - Uses the nodename specified on the 10.2 DDS statement LOCAL NODE IS clause as the system ID or
  - If DDS statement is not specified, defaults to SYST*nnnn* where *nnnn* is the value specified as *dc /ucf-version-n* on the ADD SYSTEM parameter
- AREA ACQUISITION THRESHOLD IS OFF RETRY FOREVER
- DEADLOCK DETECTION INTERVAL IS 5 1
- JOURNAL FRAGMENT INTERVAL IS 0
- JOURNAL TRANSACTION LEVEL IS 0
- RUNUNITS FOR SECURITY IS 1
- RUNUNITS FOR SYSTEM/DEST IS 1
- SCRATCH IN XA STORAGE IS NO
- XA STORAGE POOL IS 0

1 The DEADLOCK DETECTION INTERVAL defaults to the value assigned to the TICKER INTERVAL

System and teleprocessing monitor statement changes

The dictionary migration utility made additional changes to your system definitions. These are listed in the following table. In some cases where a statement or parameter is deleted, you may need to define the information as part of another CA IDMS component.

Statement Change	Where to redefine
ADSO	

Statement	Change	Where to redefine
	Added new parameter default of OPTIONAL to the COBOL MOVE IS YES /NO clause	
DBNAME	Deleted	Use the CREATE DBTABLE statement. See the *CA IDMS Database Administration section* for details on defining database name tables.
DDS	Deleted	Nodename specified on SYSTEM ID clause of the SYSTEM statement
IDMS AREA	Deleted	Specify area usage overrides using the area-override-specification clause of the ALTER DMCL statement. See the *CA IDMS Database Administration section* for a complete description of physical database definition syntax.
IDMS BUFFER	Deleted	Specify buffer sizes using the BUFFER statement. See the *CA IDMS Database Administration section* for a complete description of physical database definition syntax.
IDMS PROGRAM	Not used	PROGRAM REGISTRATION now specified using 12.0 run-unit security. For a complete discussion of the CA IDMS security facility, see the *CA IDMS Security Administration section*. Specify resource limits using the TASK statement.
OLM	Set HELP PFKEY IS to PF1	
OLQ	Added ACCESS IS IDMSSQL sql COMPLIANCE IS EXTENDED	
PROGRAM	Deleted SECURITY CLASS	Define 10.2 security classes as categories using the CA IDMS central security facility. See the *CA IDMS Security Administration section* for a complete discussion of the CA IDMS central security facility.
TASK	Deleted SECURITY CLASS and added AREA ACQUISITION THRESHOLD IS DEFAULT	Define 10.2 security classes as categories using the CA IDMS central security facility. See the *CA IDMS Security Administration section* for a complete discussion of the CA IDMS central security facility.
USER	Removed	Define users using the CA IDMS central security facility. See the *CA IDMS Security Administration section* for details on defining users with the CA IDMS central security facility.
DDS LINE	Deleted statement and all PTERM parameters	Specify using the system generation NODE statement and if using CA IDMS/DDS for cross-CPU communications, also define a CCI LINE statement.
LTERM	Deleted INTERACTIVE BREAK/NOBREAK and UPPER/LOWER parameters	Specify as profile attributes using CREATE PROFILE statement. See the *CA IDMS Security Administration section* for a complete description of the CREATE PROFILE syntax.

Other new statements and parameters

Listed below are new and modified system generation statements that are not part of your migrated system definition that you may need to define.

SYSTEM statement

The INTERNAL WAIT parameter is not used by DC/UCF systems using CA IDMS/DC and CICS TP monitors. It is replaced by the INACTIVE INTERVAL parameter of the SYSTEM statement. For more information, see "External request units," later in this section.

RESOURCE TABLE statement

You only need to use the RESOURCE TABLE statement to explicitly define remote resources that are accessed by the DC/UCF system.

NODE statement

You only need to use the NODE statement to define nodes on which remote resources reside.

Note: For more information about the NODE and RESOURCE TABLE statements, see the *CA IDMS System Generation Guide*.

LOADLIST statement

The version number for test load libraries is now specified as *Vnnnn* instead of *CDMSLnnn*. Remember to modify your startup JCL to reflect this change.

Add definitions of CA IDMS tasks and programs

You need to add CA IDMS 12.0 system generation task and program definitions to your migrated system definition to include new tasks and programs and modified task and program definitions.

Use the INCLUDE statement

There are several ways to accomplish this task. One of the easiest ways is to use an INCLUDE statement for each of the modules containing the task and program statements required to support the CA IDMS products installed in your environment.

Where to find module names

To identify the modules you need to include in your system definition, refer to the installation job that creates the SYSTEM and SYSDIRL dictionaries, or you can refer to Appendix B in the *CA IDMS System Generation Guide* for information on CA IDMS, CA IDMS Tools or CA Endevor/DB system definitions.

System generation modules

System name	Operating System	Source library member	Module name
SYSTEM90 definition	z/OS	DLODO90	OS-SGEN90
	z/VSE	DLODD90	DOS-SGEN90
	z/VM	DLODV90	CMS-SGEN90
SYSTEM99 definition	z/OS	DLODO99	OS-SGEN99
	z/VSE	DLODD99	DOS-SGEN99
	z/VM	DLODV99	CMS-SGEN99

Tasks and programs deleted

The table below identifies the tasks and programs deleted from CA IDMS system generation modules. These tasks and modules should be deleted from a migrated system before generation.

Source library member	Task	Program
DLODADSO	ADSG	ADSGMPDG
	ADSGT	ADSGMPDO
	ADSD	ADSGMPNC
		ADSGMPPM
		ADSGMPRS
		ADSOAMDS
		ADSOGEN1
		ADSOODSD
DLODIDDO	DMCL	IDMSDMDC
	DMCLT	IDMSDMM1
		IDMSDMTA
		IDMSDMTB
		IDMSDMTF
		IDMSDMTG
		IDMSDMTH
		IDMSDMTJ
DLODO99	REPLAY	RHDCRPLY

Source library member	Task	Program
	DCPASS	IDMSNWKC
		IDMSNWKM
		IDMSNWKN
		IDMSNWKO
		IDMSNWKP
		IDMSNWKQ
		IDMSNWKS
		IDMSNWKX
		IDMSNWK1
		IDMSNWK2
		IDMSNWK4
		IDMSSPF
		RHDCPASS

Regenerate your modified system definition

Once you have reviewed and modified your DC/UCF system definition, generate it in a 12.0 environment.

Prepare your DC/UCF environment for startup

Before you start up your 12.0 DC/UCF environment, perform these tasks:

1. Format DDLDCLOG and DDLDCRUN areas.
   The format of some of the records in the DDLDCLOG and DDLDCRUN areas are different.

2. Format DDLDCSCR area.

3. Format journal files.
   Journal record formats have changed.

4. Prepare ARCHIVE JOURNAL, ARCHIVE LOG, and PRINT LOG statements.
   IDMSAJNL utility program is replaced by the ARCHIVE JOURNAL utility statement and the RHDCPRLG utility program is replaced by the new utility statements ARCHIVE LOG and PRINT LOG.
   You should prepare these new utility statements and JCL streams to replace IDMSAJNL and

RHDCPRLG utility programs before bringing up your 12.0 system. This is especially important if you use a write-to-operator (WTOEXIT) exit to automatically initiate archive journal and archive and print log functions.

Note: For more information about all utility statements and programs, see the *CA IDMS Utilities Guide*.

Review and modify startup components

Review the list of system startup components below and make the necessary changes in your environment before bringing up your 12.0 DC/UCF system.

Note: For more information about DC/UCF system startup procedures, see the *CA IDMS System Operations Guide*.

Reassemble the #DCPARM macro

You must reassemble the #DCPARM macro whether or not you change any of the parameter values.

System startup components

Item	Action
#DCPARM macro	Review and modify the #DCPARM macro to include any changed parameters such as: Global DMCL name FREESTG DC/UCF system version number
WTOEXIT	Review the WTOEXIT source code to determine if any changes are needed to accommodate 12.0 control block changes.
Link edit the #DCPARM and WTOEXIT object modules	Link edit the #DCPARM and WTOEXIT object modules with the appropriate operating system-dependent module to create a new startup module
Startup routine JCL	Modify startup routine JCL to include changes to your environment for Release 12.0 such as: New startup module Region size

Item	Action
	New data set names of the required CA IDMS libraries New data set names of migrated dictionaries New data set names of additional dictionary areas and user catalog Replace CDMSL*nnn* names with V*nnnn* Any other required changes

Utilities

This section lists the new CA IDMS utility statements and identifies the 10.2 utility programs they replace. Enhancements to the remaining utility programs are presented.

Utility statements that replace programs

The following table shows the new utility statements that replace existing utility programs.

Program	New utility statement
IDMSAJNL	ARCHIVE JOURNAL
IDMSDBLU	FASTLOAD RELOAD
IDMSDUMP	BACKUP PRINT SPACE
IDMSINIT	FORMAT
IDMSJFIX	PRINT JOURNAL FIX ARCHIVE
IDMSLDEL	CLEANUP
IDMSPCON	RESTRUCTURE CONNECT
IDMSPFIX	FIX PAGE PRINT PAGE UNLOCK
IDMSPTRE	PRINT INDEX
IDMSRBCK	ROLLBACK
IDMSRFWD	ROLLFORWARD
IDMSRSTR	RESTORE
IDMSRSTU	RESTRUCTURE SEGMENT

Program	New utility statement
IDMSTBLU	MAINTAIN INDEX MAINTAIN ASF TABLE
IDMSUNLD	UNLOAD
IDMSXPAG	EXPAND PAGE
RHDCPRLG	ARCHIVE LOG PRINT LOG

Important! The format of the files created by IDMSDUMP is not compatible with the format expected by RESTORE. RESTORE accepts only files created by BACKUP. Therefore, once you have converted your environment, you should backup your database using BACKUP so you have a backup file that RESTORE can use, if you need to restore your database.

PUNCH statement

In addition to the other statements that replace programs, there is a new utility statement called PUNCH. PUNCH:

1. Retrieves DMCL load modules or database name table load modules from the catalog load area of the data dictionary
2. Writes the load modules in object form to the SYSPCH file
   Note: When you use the PUNCH statement with the CA IDMS Batch Command Facility, you can only punch one component at a time. You need to create separate JCL streams for each PUNCH statement.

Execution JCL

You need to create new execution JCL to run the new utility statements. You submit utility statements to the CA IDMS Batch Command Facility (IDMSBCF).

Note: For an overview of the basic execution JCL for the CA IDMS Batch Command Facility, see the *CA IDMS Command Facility Guide.* For more information about the execution JCL for each utility statement, see the *CA IDMS Utilities Guide*.

For more information, see the following topics:
- Utility programs (see page 66)
- Security for utilities (see page 68)

Utility programs

Status of utility programs

These utilities exist in their previous format:

- IDMSCALC

- IDMSDBAN (with enhancements)
- IDMSDIRL (with enhancements)
- IDMSLOOK (with enhancements)
- IDMSRADM
- IDMSRPTS (with enhancements)
- IDMSRSTC

Changes in existing programs

The following table describes enhancements to utilities that continue to exist in their previous format.

Utility	Enhancements
IDMSDBAN	New syntax Support for SQL-defined databases Audits indexes more efficiently
IDMSDIRL	Loads IDMSNTWK schema and two associated subschemas (IDMSNWKA and IDMSNWKG) Loads schemas IDMSSECS and IDMSSECU and subschemas IDMSSECS and IDMSSECU used for security processing Added option (SCHEMA-DELETE) to remove all CA-defined schema definitions (those listed above) from a dictionary without loading new definitions
IDMSLOOK	Revised parameters provide this output: Set information (SUBSCHEMA) Buffer information (DMCL) Information reflecting the separation of logical and physical definitions (DBTABLE) Information on segments (DBTABLE) Data set name of the library that the load module was loaded from (DATES and PROGRAM) New parameters provide: Information about IDMSLOOK parameters (HELP) Information about subschema load modules bound to a database (BIND SUBSCHEMA) Value of a date/time stamp (DATETIME STAMP) Additional parameters for SQL-related objects

Utility	Enhancements
IDMSR PTS	Removal of physical information from schema and subschema reports Enhancements to schema and subschema reports -- Reporting onnew compiler functions New reports for physical database definitions

Removal of IDMSRNWK

IDMSRNWK no longer exists. Because of the separation of logical and physical definitions, there is no longer a need for dictionary subschema reformatting.

Local mode execution JCL

You need to modify the execution JCL to run utility programs in local mode. You use the SYSIDMS parameter file to identify the DMCL, dictionary, or database CA IDMS/DB will use to execute the utility program in local mode.

Additionally, you may use the SYSIDMS parameter file to specify other parameters, depending upon the utility program.

Note: For more information about utility programs and associated execution JCL, see the *CA IDMS Utilities Guide*.

Security for utilities

The execution of utility statements is subject to security checking. The specific authority necessary to execute a utility statement depends on the function of the utility.

The *CA IDMS Utilities Guide* describes the authority required to use each utility statement and program.

User Exits and Database Procedures

This section identifies the changes you need to make to CA IDMS user exits and database procedures in order to use them in a 12.0 environment. In Release 12.0, new user exits have been added and one user exit has been deleted. Additionally, the CA IDMS/DB architecture has changed and the format of control blocks is different.

Here is a list of the new user exits:

- Exit 28 -- Security preprocessing
- Exit 29 -- Security postprocessing

- Exit 30 -- Deadlock victim selection
- Exit 31 -- Transaction statistics

User Exit

Release 10.2 user exit 3, the security check exit, is replaced by the new user exits 28 and 29. If you will continue to perform security checking in the 12.0 environment, use exits 28 and 29.

Control block changes

Because the CA IDMS/DB database architecture has changed significantly, the format of control blocks is different. Review the user exits you use in your CA IDMS environment carefully to determine the impact of control block changes.

You should make the necessary changes to the exit and then reassemble or recompile and relink each user exit in the 12.0 environment.

Database Procedures

The control blocks passed to database procedures have changed:

- The names of records, areas, error sets, error areas, etc. are now 18 bytes
- The order of the record control block is different

You must modify your CA IDMS database procedures to reflect control block changes.

Extent Areas

Extent areas are not supported in Release 12.0.

DDLDCQUE area

The DDLDCQUE area is not used in the CA IDMS 12.0 environment.

Use the DDLDCRUN area in place of the DDLDCQUE area. Remove any references you may have to the DDLDCQUE area in physical database definition components or application programs.

ASF extent areas

You must migrate ASF extent areas to standard areas in a 12.0 environment. This section describes how to migrate ASF extent areas to standard areas.

For more information, see the following topics:

- Migrate ASF extent areas (see page 70)

Migrate ASF extent areas

CA IDMS provides two programs, along with standard CA IDMS utility statements, to assist you in migrating ASF extent areas to standard areas.

Procedures

Migrating ASF extent areas involves these steps:

1. Prepare environment to migrate extent areas.
2. Run the IDMSRSSM program to:
   - Create syntax for the IDMSRSSD subschema that you will use to unload and reload ASF data areas defined as extent areas.
   - Update the catalog structure of the ASF dictionary with revised storage allocation information.
3. Review IDMSRSSD subschema syntax produced by the IDMSRSSM program and then generate it.
4. Unload and reload ASF data areas defined as extent areas.
5. Unload and reload the ASF definition area.
6. Run the IDMSRUPD program to update the RDEFREC records in the ASF definition area with the new db-keys of the table header records (RFUR-nnnnnn-OOAK).
7. Optionally, add a CALC record to the IDMSR schema to define an SR1 system record for 12.0 processing of ASF tables.

Each of these steps is described in the remainder of this section.

Prepare your environment to migrate extent areas

Release 10.2 environment

Before you migrate any ASF extent areas, take these steps in your 10.2 environment:

1. Identify the extent areas and the ASF-defined tables you want to migrate. Make sure the tables you will migrate are completely generated. The migration program will not migrate a table that is not completely generated or does not have a validated subschema.
   The IDMSRSSM program will identify all tables which cannot be migrated. If you want to migrate such a table, you must first generate (or regenerate) the table using 10.2 ASF.

2. Run the 10.2 IDMSDUMP utility program with the REPORTS=ONLY option against each ASF extent area (data areas and definition area) you will migrate. You can compare the records listed on the IDMSDUMP report with the output listing from the IDMSRSSM program to determine if all tables are processed by IDMSRSSM.

3. Backup your 10.2 format ASF dictionary and extent areas. You will need these if you must generate or regenerate an ASF table using 10.2 ASF.

Release 12.0 environment

Before you begin the migration process, be sure your Release 12.0 environment is prepared as follows:

1. A 12.0 DMCL contains all ASF areas you will migrate as well as ASF dictionary areas.

   Note: You could create a segment that contains all ASF areas being migrated or add ASF areas to an existing segment. If you make either of these changes, be sure your 12.0 DMCL contains the change. ASF areas are defined as standard areas in your 12.0 DMCL..note off

2. ASF dictionaries are in a 12.0 format (that is, they must have been migrated to a 12.0 format using the Release 12.0 Dictionary Migration Utility).

Run IDMSRSSM program

Purpose

Once your environment is prepared, run the IDMSRSSM program separately for each migrated ASF dictionary on a 12.0 system. The IDMSRSSM program:

- Creates the subschema syntax for the IDMSRSSD subschema and places it in an output file. The IDMSRSSD subschema is used to unload and reload ASF data areas.

- Updates the catalog structure in the ASF dictionary with the maximum and current table allocations for each user. Space allocations are now expressed as a number of tables rather than in K bytes.

  Important! In a 10.2 environment, if the user defaults total allocation limit is 1000 K bytes, IDMSRSSM converts that to a total allocation limit of 1000 tables for Release 12.0. You should review these allocations and modify or add them as appropriate.

  Note: For more information about defining and modifying allocation limits and defaults for user's tables, see the *CA IDMS ASF User Guide*.

- Indicates whether you need to add a CALC record to the IDMSR schema so that the SR1 system record is included in the schema for 12.0 ASF processing.
- Produces a listing of record IDs for the tables in the IDMSR schema version 1.

  Important! Tables that do not have an existing or validated subschema are not included in the IDMSRSSD subschema.

You can run the IDMSRSSM program against the same ASF dictionary area as often as is necessary. However, only the initial run of IDMSRSSM will update the catalog structure of the ASF dictionary. Subsequent executions of IDMSRSSM will not perform any updates but will report on the allocations.

Syntax

```
    ┌──────────────────────────────┐
►►─▼─ AREA=extent-area-name ───────┴──────────────────────────────►◄
```

Parameter

- **AREA=*extent-area-name***
  Specifies the name of the ASF extent area you will migrate. *Extent-area-name* must be an extent area defined in the dictionary.
  You can specify a maximum of 20 area statements for each run of the IDMSRSSM program.

Execution JCL

IDMSRSSM is a program in the load library provided at installation.

IDMSRSSM runs in a batch environment in either local mode or under the central version.

Execution JCL for the IDMSRSSM program is provided below.

z/OS JCL

IDMSRSSM

```
//BUILD    EXEC PGM=IDMSRSSM,REGION=
//STEPLIB  DD  DSN=idms.dba.loadlib,DISP=SHR
//         DD  DSN=idms.loadlib.,DISP=SHR
//asfdict  DD  DSN=idms.asfdict.ddldml,DISP=SHR
//dcmsg   DD  DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//sysjrnl  DD  DSN=idms.tapejrnl,DISP=(NEW,KEEP),UNIT=TAPE
//SYS004   DD  DSN=output.file,DISP=(,CATLG),UNIT=
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=nnnn)
//SYSLST   DD SYSOUT=A
//SYSIDMS  DD  *

   DMCL=dmcl-name
   DBNAME=dbname or segment-name

//SYSIPT   DD *

   AREA=extent-area-name    AREA=extent-area-name
```

idms.dba.loadlib	Data set name of the load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the load library containing executable Release 12.0 CA IDMS modules
asfdict	DDname of the ASF (DDLDML) dictionary area
idms.asfdict.ddldml	Data set name of the ASF (DDLDML) dictionary area
dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg. ddldcmsg	Data set name of the system message (DDLDCMSG) area
sysjrnl	DDname of a tape journal file, if journaling
idms.tapejrnl	Data set name of tape journal file
sys004	DDname of the output file for the subschema produced by the IDMSRSSM program
output.file	Data set name of the output file for the subschema produced by the IDMSRSSM program
dmcl-name	The name of a Release 12.0 DMCL
dbname or segment-name	Database name or segment name identifying the DDLDML area of the ASF dictionary
extent-area-name	The name of an ASF extent area

To run IDMSRSSM under the central version, add a SYSCTL file.

z/VSE JCL

IDMSRSSM

```
// JOB   IDMSRSSM
// LIBDEF *,SEARCH=120 libraries
// EXEC PROC=IDMSLBLS
// ASSGN sysnnn,DISK,VOL=nnnnnn,SHR
// DLBL SYS004,'output.file'
// EXEC   IDMSRSSM,SIZE=1048K
DMCL=dmcl-name
DBNAME=dbname or segment-name
/*
AREA=extent-area-name    AREA=extent-area-name
/*
```

IDMSLBLS	Name of the procedure provided at installation that contains the file definitions for CA IDMS dictionaries, databases, disk journal files, and the SYSIDMS file. **Note:** For more information about complete listing of IDMSLBLS, see Appendix C, IDMSLBLS Procedure for z/VSE JCL.
sys004	Filename of the output file for the subschema produced by IDMSRSSM program
dmcl-name	The name of a Release 12.0 DMCL
dbname or segment-name	Database name or segment name identifying the DDLDML area of the ASF dictionary
extent-area-name	The name of an ASF extent area

To run IDMSRSSM under the central version, add a SYSCTL file.

z/VM commands

IDMSRSSM

```
FILEDEF SYSLST PRINTER
FI asfdict DISK cdms asfdict fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FI dcmsg DISK cdms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FILEDEF sysjrnl TAP1 SL VOLID nnnnnn (RECFM VB LRECL lll BLKSIZE bbbb
FILEDEF SYS004 DISK sys004 output a
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK idmsrssm input a
GLOBAL LOADLIB dbalib idmslib
OSRUN IDMSRSSM
```

asfdict	DDname of the ASF (DDLDML) dictionary area
cdms asfdict fm	File identifier of the ASF (DDLDML) dictionary area
dcmsg	DDname of the system message (DDLDCMSG) area
cdms dmsgdb fm	File identifier of the system message (DDLDCMSG) area
ppp	Page size of the database file
nnn	Number of pages in the database file
sysjrnl	DDname of the tape journal file, if journaling
nnnnnn	Volume serial number of the tape journal file
lll	Record length of the tape journal file
bbbb	Block size of the tape journal file
sys004 output a	File identifier of file containing subschema produced by IDMSRSSM program
sysidms input a	File identifier of the file containing the following: dmcl-name -- Name of the DMCL module

	segment-name -- database name or segment name identifying the DDLDML area of the ASF dictionary
idmsrssm input a	File identifier of the file containing the names of the ASF extent areas you will migrate
dbalib	Filename of the load library containing the DMCL and database name table load modules
idmslib	Filename of the load library containing executable Release 12.0 CA IDMS modules

Output from IDMSRSSM

What IDMSRSSM produces

- An output listing that includes various information. This information is described in the table below.
- Subschema syntax for the IDMSRSSD subschema.

Sample IDMSRSSM output listing

Portions of a sample output listing from IDMSRSSM are provided below.

Allocation limits are represented as tables instead of K bytes and allocation defaults are removed from the catalog.

```
   RECORD ID:  31839
   RECORD ID:  31840
   RECORD ID:  31841
   RECORD ID:  31842
  VALIDATE NOT DONE FOR SUBSCHEMA:
  RECORD:  RFUR-000217-OOAK                 WILL NOT BE INCLUDED
  IN SYNTAX FOR AREA: IDMSR-AREA2

   RECORD ID:  31913
  VALIDATE NOT DONE FOR SUBSCHEMA:
  RECORD:  RFUR-000217-DATA                 WILL NOT BE INCLUDED
  IN SYNTAX FOR AREA: IDMSR-AREA2

   RECORD ID:  31914
  VALIDATE NOT DONE FOR SUBSCHEMA:
  RECORD:  RFUR-000218-OOAK                 WILL NOT BE INCLUDED
  IN SYNTAX FOR AREA: IDMSR-AREA2
   RECORD ID:  31934
  RECORD:  RFOR-000135-OOAK                 WILL NOT BE INCLUDED
  IN SYNTAX FOR AREA: IDMSR-AREA2

   RECORD ID:  31935
  RECORD:  RFOR-000135-DATA                 WILL NOT BE INCLUDED
  IN SYNTAX FOR AREA: IDMSR-AREA2

   RECORD ID:  31936
   RECORD ID:  31998
   RECORD ID:  31999
   AREAS ADDED: 0002  RECORDS ADDED: 0140   SETS ADDED: 0124

                  UPDATING CATALOG

   USER:  CORP                            EXG
VALUES FOUND: TOTAL USED:  00002291     COUNT OF BYTES:  00002291
REPLACED WITH TABLE COUNT:   00010
   COUNT OF TABLES UNALLOCATED:  00001

   USER:  CORP                            W FURR
VALUES FOUND: TOTAL USED:  00000000     COUNT OF BYTES:  00000000
```

```
   REPLACED WITH TABLE COUNT:   00000
   USER:  CORP                          DNT
VALUES FOUND: TOTAL USED:  00001014     COUNT OF BYTES:  00001014
   REPLACED WITH TABLE COUNT:   00004
COUNT OF TABLES UNALLOCATED:  00004

   USER:  CORP                          SJG
USER DEFAULTS TOTAL ALLOCATION:  00000100
BEING REMOVED - PRIMARY:  00000  SECONDARY: 00000 MAXIMUM: 00000
VALUES FOUND: TOTAL USED:  00030468     COUNT OF BYTES:  00030968
   REPLACED WITH TABLE COUNT:   00011

   USER:  CORP                          CULL DBA
VALUES FOUND: TOTAL USED:  00001512     COUNT OF BYTES:  00001513
   REPLACED WITH TABLE COUNT:   00002
   COUNT OF TABLES UNALLOCATED:  00001

COUNT OF USERS UPDATED:   00025
```

Review the IDMSRSSM output listing

The table below describes the information provided on the listing from IDMSRSSM that you should review. You should review this information as it might require you to take some action. Suggested actions are provided to assist you.

If you need to change the status of any ASF table after reviewing the output listing:

- Make the changes using 10.2 ASF on a 10.2 system using the backup copy of the 10.2 ASF dictionary
- Migrate the 10.2 dictionary to a 12.0 format

Note: For complete instructions on migrating a dictionary to a 12.0 format, see Security Migration (see page 30).

- Rerun IDMSRSSM to reflect the changes

Item in report	What it means	What to do
Optionally, this message: A DUMMY CALC RECORD MUST BE ADDED	This message appears after the area statements at the beginning of the listing only if an SR1 record needs to be added to the IDMSR schema.	If this message appears on your listing, add a record with a location mode of CALC to your IDMSR schema. See Optionally, add CALC record to IDMSR schema (see page 83) for instructions on how to include a CALC record in the IDMSR schema.
RECORD ID	The record ID of a table included in the IDMSRSSD subschema.	If you produced an IDMSDUMP report, compare it with the list of record IDs produced by IDMSRSSM to determine if all tables were processed.
	Identifies the record ID of a table that does not have a validated subschema.	

Item in report	What it means	What to do
VALIDATE NOT DONE FOR SUBSCHEMA		If you want to migrate this table, use ASF on a 10.2 system to determine why the subschema is not validated, fix the problem, and generate the table.
RECORD: RFOR-nnnnnn-DATA WILL NOT BE INCLUDED IN SYNTAX.....	Identifies tables that have existing definition record names preceded with 'RFOR-'. Existing record names preceded with 'RFOR-' indicate that a table is not completely regenerated. These records are not included in the IDMSRSSD subschema.	If you want to migrate the data from tables that correspond to an existing 'RFOR' record, access the table through ASF on a 10.2 system and complete the regeneration process.
TOTAL USED	The total number of K bytes a user's tables consume.	
COUNT OF BYTES	The number of K bytes allocated for a user as calculated by IDMSRSSM.	
REPLACED WITH TABLE COUNT	The number of tables generated by a user. This number is calculated by IDMSRSSM from information in the catalog and is now maintained as a number of tables instead of K bytes.	
COUNT OF TABLES UNALLOCATED	The number of tables defined but not generated by a user.	
USER DEFAULTS TOTAL ALLOCATION	The maximum number of tables a user can define. This was previously maintained in K bytes and is now represented as a number of tables. For example, if the user defaults total allocation for user PRK was 100 K bytes in 10.2, IDMSRSSM changes that to 100 tables in 12.0.	Review the updated allocation limits. If necessary, use ASF on a 12.0 system to change them to meet the requirements of your environment.
BEING REMOVED - PRIMARY: SECONDARY: MAXIMUM	The values associated with these allocation defaults are removed from the catalog and are no longer maintained.	

Generate the IDMSRSSD subschema

After reviewing the IDMSRSSM output, use the subschema compiler in a 12.0 environment to generate the IDMSRSSD subschema.

Note: IDMSRSSD is the default subschema name used by the IDMSRUPD program if a subschema name is not provided. You can change the subschema name before generating it. If you do, be sure to specify the new name when running the UNLOAD and RELOAD utility statements against ASF data areas and the IDMSRUPD program.

Unload and reload ASF data areas

After you generate the IDMSRSSD subschema, you need to:

1. Unload ASF data areas using the 12.0 UNLOAD utility statement.
2. Format each data area using the 12.0 FORMAT utility statement.
3. Reload the data unloaded by the previous unload step into the formatted 12.0 area using the 12.0 RELOAD utility statement.

Note: For more information about 12.0 utility statements, see the *CA IDMS Utilities Guide*.

Unload and reload ASF definition area

After you unload and reload ASF data areas, you can unload and reload the ASF definition area using the IDMSRSSA subschema. The IDMSRSSA subschema is only installed with ASF.

Note: All 10.2 subschemas must be in a 12.0 format for use in a 12.0 environment. CA IDMS automatically converts a 10.2 subschema load module to a 12.0 format (if it is not in a 12.0 format) when it accesses it at runtime. The converted 12.0 subschema load module is temporary and is not saved in a load area or load library. It only exists for the duration of the run unit. It is recommended that you permanently convert your 10.2 subschemas to a 12.0 format. For procedures on permanently converting a 10.2 subschema to a 12.0 subschema, see Security Migration (see page 30).

1. Unload ASF definition area using the 12.0 UNLOAD utility statement
2. Format the definition area using the 12.0 FORMAT utility statement
3. Reload the data unloaded by the previous unload step into the formatted 12.0 definition area using the 12.0 RELOAD utility statement

Note: For more information about 12.0 utility statements, see the *CA IDMS Utilities Guide*.

Run IDMSRUPD program

Once the ASF definition area and ASF data areas are migrated, you must run the IDMSRUPD program to update RDEFREC records in the ASF definition area with the new db-keys of related table header records (RFUR-nnnnnn-OOAK).

Purpose

The IDMSRUPD program:

- Accesses each RFUR-nnnnnn-OOAK record in the subschema produced by the IDMSRSSM program (default subschema name is IDMSRSSD) to get the table definition number of the table it represents
- Accesses the RDEFREC that the table definition number corresponds to in the definition area and updates the RDEFREC with the db-key of its corresponding RFUR-nnnnnn-OOAK record

Syntax

```
►►─┬──────────────────────┬──────────────────────────────────────────►◄
   ├─ SUB=subschema-name ─┤
   └─ SUB=IDMSRSSD ◄ ─────┘
```

Parameters

- ***SUB=subschema-name***
  Specifies the name of the subschema the IDMSRUPD program will use. *Subschema-name* must be a 1- through 8- character alphanumeric value.
  If you do not specify a *subschema-name*, IDMSRUPD will use IDMSRSSD as the default subschema name.

Execution of JCL

The IDMSRUPD program runs in a batch environment in local mode.

Note: Depending on the size of your data areas, IDMSRUPD may be a longrunning job.

Execution JCL for IDMSRUPD is provided below.

z/OS JCL

To run IDMSRUPD in local mode, code this JCL.

IDMSRUPD

```
//BUILD    EXEC PGM=IDMSRUPD,REGION=
//STEPLIB  DD  DSN=idms.dba.loadlib,DISP=SHR
//         DD  DSN=idms.loadlib,DISP=SHR
//asfdef   DD  DSN=idms.asfdef,DISP=SHR

If processing more than one data area,
add dd statements for them.
//asfdata  DD  DSN=idms.asfdata,DISP=SHR
//dcmsg    DD  DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//sysjrnl  DD  DSN=idms.tapejrnl,DISP=(NEW,KEEP),UNIT=TAPE
//SYSLST   DD SYSOUT=A
//SYSIDMS  DD  *

DMCL=dmcl-name
DBNAME=segment-name

//SYSIPT   DD *

SUB=subschema-name
```

idms.dba.loadlib	Data set name of the load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the load library containing executable Release 12.0 CA IDMS modules
asfdef	DDname of the ASF definition area
idms.asfdef	Data set name of the ASF definition area
asfdata	DDname of an ASF data area
idms.asfdata	Data set name of the ASF (DDLDML) dictionary area
dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg. ddldcmsg	Data set name of the system message (DDLDCMSG) area
sysjrnl	DDname of the tape journal file, if journaling
idms.tapejrnl	Data set name of the tape journal file, if journaling
dmcl-name	The name of a Release 12.0 DMCL
dbname	Database name or segment name identifying the DDLDML area of the ASF dictionary
subschema-name	Name of the subschema IDMSRUPD will use if not IDMSRSSD

z/VSE JCL

IDMSRUPD

```
// JOB   IDMSRUPD
// LIBDEF *,SEARCH=120 libraries
// EXEC PROC=IDMSLBLS
// ASSGN sysnnn,DISK,VOL=nnnnnn,SHR
// EXEC   IDMSRUPD,SIZE=1048K
DMCL=dmcl-name
DBNAME=dbname or segment-name
/*
SUB=subschema-name
/*
```

IDMSLBLS	Name of the procedure provided at installation that contains the file definitions for CA IDMS dictionaries, databases, disk journal files, and the SYSIDMS file.

	Note: For more information about IDMSLBLS, see Appendix C, IDMSLBLS Procedure for z/VSE JCL.
dmcl-name	The name of a Release 12.0 DMCL
dbname	Database name or segment name identifying the DDLDML area of the ASF dictionary
subschema-name	Name of the subschema IDMSRUPD will use if not IDMSRSSD

z/VM commands

IDMSRUPD

```
FILEDEF SYSLST PRINTER
FI asfdef DISK cdms asfdef fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn

If processing more than one data area, add additional database
 file assignments, as required.

FI asfdata DISK cdms asfdata fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FI dcmsg DISK idms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FILEDEF sysjrnl TAP1 SL VOLID nnnnnn (RECFM VB LRECL lll BLKSIZE bbbb
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK idmsrupd input a
GLOBAL LOADLIB dbalib idmslib
OSRUN IDMSRUPD
```

asfdef	DDname of the ASF definition area
cdms asfdef fm	File identifier of the ASF definition area
asfdata	DDname of the ASF data area
cdms asfdata fm	File identifier of the ASF data area
dcmsg	DDname of the system message (DDLDCMSG) area
idms dmsgdb fm	File identifier of the system message (DDLDCMSG) area
ppp	Page size of the database file
nnn	Number of pages in the database file
sysjrnl	DDname of the tape journal file, if journaling
nnnnnn	Volume serial number of the tape journal file
lll	Record length of the tape journal file
bbbb	Block size of the tape journal file
sysidms input a	File identifier of the file containing the following: dmcl-name -- name of Release 12.0 DMCL dbname or segment-name -- database name or segment name identifying the DDLDML area of the ASF dictionary

idmsrupd input a	File identifier of the file containing the name of the subschema IDMSRUPD will use if not IDMSRSSD
dbalib	Filename of the load library containing the DMCL and database name table load modules
idmslib	Filename of the load library containing executable Release 12.0 CA IDMS modules

Output from IDMSRUPD

Output from IDMSRUPD

The IDMSRUPD program produces an output listing identifying the records processed and whether or not the RDEFREC was updated with a db-key.

Sample IDMSRUPD output listing

A portion of the output listing produced by the IDMSRUPD program is provided below.

All errors that result from the execution of the IDMSRUPD program are due to a mismatch between the OOAK record and the table definition. When any error results, the RDEF record is *not* updated with a db-key.

Date/time in OOAK record is not the same as found in the RDEF record.

Status of table is not 'G' for generated.

```
               STARTING TO UPDATE IDMSR-AREA

PROCESSING RECORD ID:  31998   RFUR-000102-OOAK
DBKEY UPDATED IN RDEFREC FOR TDN:  000102

PROCESSING RECORD ID:  31996   RFUR-000101-OOAK
TABLE NOT GENERATED FOR TDN:  000101

PROCESSING RECORD ID:  31994   RFUR-000103-OOAK
DBKEY UPDATED IN RDEFREC FOR TDN:  000103

PROCESSING RECORD ID:  31992   RFUR-000104-OOAK
DBKEY UPDATED IN RDEFREC FOR TDN:  000104

PROCESSING RECORD ID:  31990   RFUR-000106-OOAK
DBKEY UPDATED IN RDEFREC FOR TDN:  000106

PROCESSING RECORD ID:  31988   RFUR-000105-OOAK
RECORD: RFUR-000105-OOAK
NOT FOUND IN AREA:  IDMSR-AREA2

PROCESSING RECORD ID: 31986 RFUR-000108-OOAK
DATE/TIME MISMATCH FOR TDN: 000108

PROCESSING RECORD ID:  31984   RFUR-000114-OOAK
DATE/TIME MISMATCH FOR TDN:  000114

PROCESSING RECORD ID:  31982   RFUR-000109-OOAK
DBKEY UPDATED IN RDEFREC FOR TDN:  000109

PROCESSING RECORD ID:  31980   RFUR-000111-OOAK
DBKEY UPDATED IN RDEFREC FOR TDN:  000111

PROCESSING RECORD ID:  31974   RFUR-000137-OOAK
```

```
DBKEY UPDATED IN RDEFREC FOR TDN:  000137

PROCESSING RECORD ID:  31972   RFUR-000115-OOAK
DBKEY UPDATED IN RDEFREC FOR TDN:  000115

PROCESSING RECORD ID:  31970   RFUR-000116-OOAK
DBKEY UPDATED IN RDEFREC FOR TDN:  000116

PROCESSING RECORD ID: 31968 RFUR-000117-OOAK
TABLE NOT GENERATED FOR TDN: 000117

PROCESSING RECORD ID:  31966   RFUR-000118-OOAK
DBKEY UPDATED IN RDEFREC FOR TDN:  000118

PROCESSING RECORD ID:  31964   RFUR-000205-OOAK
DBKEY UPDATED IN RDEFREC FOR TDN:  000205

PROCESSING RECORD ID:  31962   RFUR-000120-OOAK
DBKEY UPDATED IN RDEFREC FOR TDN:  000120

PROCESSING RECORD ID:  31960   RFUR-000121-OOAK
DBKEY UPDATED IN RDEFREC FOR TDN:  000121
```

Optionally, add CALC record to IDMSR schema

If the IDMSRSSM output listing indicates that you need to add a CALC record to the IDMSR schema, modify the IDMSR schema using the 12.0 schema compiler and the following syntax:

```
MODIFY SCHEMA NAME IS IDMSR VERSION IS 1.
ADD RECORD NAME IS ASF-DUMMY-REC
RECORD ID IS 9037
LOCATION MODE IS CALC USING OWNER-RECORD-DEF-NUMBER
DUPLICATES ARE NOT ALLOWED
WITHIN AREA IDMSR-AREA.
02 OWNER-RECORD-DEF-NUMBER
    PICTURE IS S9(8)
    USAGE IS COMP
    COMMENTS 'DUMMY CALC RECORD FOR IDMSR SCHEMA'.
VALIDATE.
```

Execution JCL 2

Sample execution JCL to execute the batch schema compiler is provided below.

z/OS JCL

IDMSCHEM

```
//UPDSTEP  EXEC PGM=IDMSCHEM,REGION=
//STEPLIB  DD  DSN=idms.loadlib,DISP=SHR
//asfdict  DD  DSN=idms.asfdict.ddldml,DISP=SHR
//dcmsg   DD  DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//sysjrnl  DD  DSN=idms.tapejrnl,DISP=(NEW,KEEP),UNIT=TAPE
//SYSLST   DD  SYSOUT=A
//SYSPCH   DD  SYSOUT=A
//SYSIDMS  DD  *

 DMCL=dmcl-name
 DICTNAME=dictionary-name

//SYSIPT  DD  *
```

```
Insert modified IDMSR schema statements here
/*
```

idms.loadlib	Data set name of the load library containing executable Release 12.0 CA IDMS modules
asfdict	DDname of the ASF (DDLDML) dictionary area
idms.asfdict.ddldml	Data set name of the ASF (DDLDML) dictionary area
dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg. ddldcmsg	Data set name of the system message (DDLDCMSG) area
sysjrnl	DDname of the tape journal file, if journaling
idms.tapejrnl	Data set name of the tape journal file, if journaling

z/VSE JCL

IDMSCHEM

```
// JOB   IDMSCHEM
// LIBDEF *,SEARCH=120 libraries
// EXEC PROC=idmslbls
// ASSGN sysnnn,DISK,VOL=nnnnnn,SHR
// ASSGN  SYSPCH,DISK,VOL=nnnnnn,SHR
// DLBL   IJSYSPH,'syspch.dsn'
// EXTENT SYSPCH,volume,x,x,x,x
// EXEC   IDMSCHEM,SIZE=1048K
DMCL=dmcl-name
DBNAME=dbname or segment-name
/*
  Insert modified IDMSR schema statements here
/*
```

IDMS LBLS	Name of the procedure provided at installation that contains the file definitions for CA IDMS dictionaries, databases, disk journal files, and the SYSIDMS file. **Note:** For more information about IDMSLBLS, see Appendix C, IDMSLBLS Procedure for z/VSE JCL.

z/VM commands

IDMSCHEM

```
FILEDEF SYSLST PRINTER
FILEDEF SYSPCH PRINTER
FILEDEF sysjrnl TAP1 SL VOLID nnnnnn (RECFM VB LRECL lll BLKSIZE bbbb
FI asfdict DISK cdms asfdict fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FI dcmsg DISK cdms dmsgdb fm (RECFM F LRECL ppp BLKSIZE ppp XTENT nnn
FILEDEF SYSIDMS DISK sysidms input a
FILEDEF SYSIPT DISK schema input a
GLOBAL LOADLIB idmslib
OSRUN IDMSCHEM
```

sysjrnl	DDname of the tape journal file, if journaling
nnnnnn	Volume serial number of the tape journal file

lll	Record length of the tape journal file
bbbb	Block size of the tape journal file
asfdict	DDname of the ASF (DDLDML) dictionary area
cdms asfdict fm	File identifier of the ASF (DDLDML) dictionary area
dcmsg	DDname of the system message (DDLDCMSG) area
cdms dmsgdb fm	File identifier of the system message (DDLDCMSG) area
ppp	Page size of the database file
nnn	Number of pages in the database file
sysidms input a	File identifier of the file containing the following: *dmcl-name* -- name of the DMCL module *dictionary-name* -- name of the dictionary containing the schema to be updated
schema input a	File identifier of the file containing the IDMSR schema statements
idmslib	Filename of the load library containing executable Release 12.0 CA IDMS modules

Application Environment

SYSIDMS Overview

Establishing Site Defaults

Site-specific defaults can be established for all SYSIDMS parameters by assembling a SYSIDMS defaults load module. If it exists, this load module is used at runtime to determine the default values for all SYSIDMS parameters. Defaults may then be overridden in an individual job step by including a SYSIDMS parameter file in the execution JCL.

For more information, see the following topics:

- Local Mode Processing (see page 86)
- Deleted CA IDMS modules (see page 106)
- New version of IDMSUTIO (see page 107)
- COBOL (IDMSDMLC) precompiler JCL (see page 107)
- PL/I (IDMSDMLP) precompiler JCL (see page 107)
- Considerations for relinking application programs (see page 107)
- SPF indexes (see page 108)
- Region size (see page 109)

Local Mode Processing

Creating a SYSIDMS Defaults Load Module

The following example illustrates how to code a SYSIDMS defaults load module. It is a table of 80-character constants, each of which may contain one or more SYSIDMS parameters, as described in Parameter Descriptions. A parameter and its value must be contained within a single 80-character constant, but more than one parameter may appear within a constant. The last constant must have a value of "END SYSIDMS DEFAULTS."

```
TITLE 'SYSIDMS - Build load module for SYSIDMS defaults'
SYSIDMS START 0
**************************************************************
* Code any SYSIDMS parms that you want to be part of this SYSIDMS
* defaults load module.  This SYSIDMS defaults load module will be
* processed first before trying to process any SYSIDMS parms defined
* in the JCL for any IDMS batch job.
***************************************************************
          SPACE
          DC  CL80'ECHO=ON DMCL=GLBLDMCL'
          DC  CL80'JOURNAL=OFF'
          SPACE
* The following statement is mandatory and must be the last statement
* in the SYSIDMS defaults load module.
          DC  CL80'END SYSIDMS DEFAULTS'
          END
```

Linking a SYSIDMS Defaults Load Module

The load module must have both a name and an entry point of SYSIDMS. For operating systems that support XA storage, the load module can be linked as AMODE 31, RMODE ANY.

Overriding SYSIDMS Parameter Defaults

SYSIDMS default values can be overridden for an individual job step by including a SYSIDMS parameter file in the execution JCL.

In the following example, the SYSIDMS parameters included in the job stream instruct CA IDMS/DB to use the DMCL LOCLDMCL to execute a job. DBNAME identifies EMPDB as the database to access at runtime, and the QSAM parameters instruct CA IDMS/DB to use the IDMSQSAM look-ahead read facility when accessing EMPSEG.EMPAREA:

```
//SYSIDMS  DD *
DMCL=LOCLDMCL DBNAME=EMPDB
IDMSQSAM=ON  QSAMAREA=EMPSEG.EMPAREA
```

In the following example, the SYSIDMS parameters used are typical for a batch job running under the central version:

```
//SYSIDMS  DD *
DBNAME=EMPDB  NODENAME=SYSTEM90
```

For more information, see the following topics:

- SYSIDMS Parameter File (see page 87)
- Security in local mode (see page 106)

SYSIDMS Parameter File

What is the SYSIDMS parameter file

Applications that run in a local mode environment will use the SYSIDMS parameter file in JCL streams to specify:

- Physical requirements of the environment such as the DMCL and database to use at run time
- Run-time directives that assist in application execution
- Operating system-dependent file information

For more information about SYSIDMS parameters, see Administrating CA IDMS Database (https://docops.ca.com/display/IDMS19/Administrating+CA+IDMS+Database).

Review JCL stream for local mode programs

You should review the JCL streams for all batch programs that run in local mode and:

- Identify the SYSIDMS parameters needed by the program
- Code a SYSIDMS parameter file
- Remove unnecessary parameters from the SYSIPT file
- Relink the program in a 12.0 environment

Note that the DMCL name defaults to IDMSDMCL in local mode. If you will use a different DMCL name, be sure to explicitly code it in the SYSIDMS parameter file.

SYSIDMS Parameter Coding Considerations

The following table describes the coding guidelines used for coding SYSIDMS parameters:

Item	Guideline
Input	The input must be in columns 1 to 72. Lowercase or mixed characters are translated to uppercase. A statement must be complete on one input line. You can submit multiple statements on the same input line.
Comments	Begin comments with a double hyphen (--). Comments continue to the end of an input line.

Other CA IDMS products using SYSIDMS

In addition to user-written batch applications, be sure to review and modify the JCL streams for batch jobs that execute:

- CA ADS/Batch
- CA CULPRIT
- CA OLQ

SYSIDMS Parameter Descriptions

- **ABEND_ON_DEADLOCK**
  Forces the abnormal termination of a task that encounters a database resource deadlock. In normal CV operations, a database resource deadlock results in control being returned to the application program with an indication that a deadlock occurred. This parameter causes the task to be abended.

  > **Note:** This parameter is meaningful only in the SYSIDMS file associated with a central version.

- **ABEND_SVC_DUMP=ON|OFF**
  Causes an SVC dump to be requested when CV abends prematurely (ON) or not at all (OFF).
  Default: ON

  > **Note:** Specifying OFF is not recommended. If the CV address space abends, a dump is required to determine the cause of the abend.

- **ABENDTRACE=ON|OFF**
  Activates the tracing of various pieces of CA IDMS data when using CA Optimizer/II or CA SymDump Batch. This parameter is meaningful only in the SYSIDMS file associated with a batch job.
  CA Optimizer/II or CA SymDump Batch is required to use this parameter.

  > **Note:** if CA Optimizer/II or CA SymDump Batch is not installed, and you do not specify ABENDTRACE=ON, you can still get an abbreviated trace of CA IDMS activity by setting optional bit 265 in RHDCOPTF.

- **ABENDTRACE_ENTRIES=*nnn***
  Overrides the default number of entries being traced by ABENDTRACE. This parameter is meaningful only in the SYSIDMS file associated with a batch job.
  CA Optimizer/II or CA SymDump Batch is required to use this parameter.

- **ABENDTRACE_SUBSCHEMA_DISPLAY=ON**
  Activates the display of information from the subschema in use at the time of abend when using ABENDTRACE. This parameter is meaningful only in the SYSIDMS file associated with a batch job.
  CA Optimizer/II or CA SymDump Batch is required to use this parameter.

- **ABENDTRACE_VIBSNAP=ON**
  Causes the dump of the VIB at the time of abend when using ABENDTRACE. This parameter is meaningful only in the SYSIDMS file associated with a batch job.
  CA Optimizer/II or CA SymDump Batch is required to use this parameter.

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

- **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

- **AREA_VALIDATION_MSGS=ON|OFF**
  ON causes the informational messages DB347042 and DB347043 to be displayed on the JES log during startup and shutdown for each area being shared in a SYSPLEX data sharing environment. If you are sharing many areas this can cause congestion on the JES log.
  Default: OFF
  This parameter is only applicable in a SYSPLEX data sharing environment.

- **BCF_INPUT_80=ON|OFF**
  IDMSBCF input is done over 80 characters by a #LINEIN INAREA=CARD,MAXIN=80,... statement. However, afterwards all information starting from column 73 until column 80 is cleared and overwritten by spaces. That is, the actual maximum input size is only 72 characters.
  Executing any SQL PUNCH command may produce a SYSPCH output exceeding column 72. When using this output as input for an IDMSBCF job afterwards, may result in possible syntax errors because the input is truncated after column 72.
  By specifying SYSIDMS parameter BCF_INPUT_80=ON, information between columns 73 and 80 is not cleared, and input for BCF will be over 80 characters. In all other cases it defaults to 72 characters, for example, no BCF_INPUT_80 parameter or not specifying BCF_INPUT_80=OFF. .
  Default: OFF

- **BUFFER_PURGE**
  Causes updated pages to be written to the database whenever the number of buffers containing such pages exceeds one-quarter of the number of pages in the buffer pool. This parameter may improve the performance of local mode update jobs that do not frequently commit, since it makes buffers available for pre-fetch. This parameter is meant for only local mode batch jobs.

- **BUFFERSTAT**
  Produces a report containing buffer pool I/O statistics that can be used for tuning purposes. The report is written to SYSLST at the end of the job. It is meant only for local mode batch jobs.

  Note: For a description of the fields in the report, see the BUFFERSTAT Report Field Descriptions (see page 103).

- **CICS_NAME=*cics-name***
  Specifies a 1 to 4 character value that identifies the CICS system name.

- **CVMACH=*cms-machine-name***
  (z/VM only) Specifies the virtual machine in which the DC/UCF system is executing.

- **CVNUM=*nnn***
  (z/VM only) Specifies the number of the central version that is accessible by CMS and is used to route database requests through the IDMSVMCF facility.
  Limits: 0-255

- **CVRETRY=ON|OFF**
  ON indicates that the following message is displayed on the operator console when the CA IDMS central version is not active:

  ```
  CV nnn NOT ACTIVE. REPLY RETRY OR CANCEL.
  ```

  Default: ON

- **CVRETRY_MSG_CODES=descriptor-route-codes**
  Specifies the descriptor and route codes, in an eight-hexadecimal format, to be used for batch message DC208002 (CV cv-number NOT ACTIVE. REPLY RETRY OR CANCEL).
  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).

- **CV_STARTUP_XA_REGION_MB=*nnn***
  Specifies the size of the initial XA storage pool acquired during early CV startup.

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

  Default: 32 MB

- **DATE_SIMULATOR_SVC=nnn**
  Causes CA IDMS to issue an SVC to call the DATE SIMULATOR instead of issuing STCK and STCKE commands.
  The purpose of this parameter is to run DATE SIMULATOR without affecting CA IDMS modules.

- **DB_DEADLOCK_DUMP**
  Specifies that a dump is produced for a task that abends due to a database resource deadlock. This parameter is used in conjunction with the ABEND_ON_DEADLOCK parameter. If not specified, no dump is produced when a task abends due to a database deadlock.

- **DBIO_HICCUP=nn**
  By default, the Central Version (CV) generates a "hiccup wait" if a task accesses 100 pages without generating a wait. This parameter allows 1-99 pages to be accessed before a wait is generated. Specifying a value greater than 100 results in a page count of 100.

Note: Specifying a value less than 99 pages could result in increased run times for some tasks, since they may wait too often. We recommend experimenting with different values to choose the appropriate value for your site.

- **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. 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.

- **DBNAME=*database-name***
  Specifies the name of the database to access at run-time, for non-SQL applications. DBNAME is either a segment name or a name defined in a database name table.

- **DC_DEADLOCK_NODUMP**
  Specifies that a dump is not produced for a task that abends due to a DC resource deadlock. This parameter overrides the DUMP/NODUMP sysgen parameter.

- **DC_DEADLOCK_0029**
  Specifies that tasks that encounter a DC resource deadlock abend with a code of 0029 rather than a code of DEAD.

- **DC_SCRATCH=ON|OFF**
  ON allows local jobs to use the central version's scratch area (DDLDCSCR) when a local scratch area (DDLOCSCR) is not defined in the DMCL.
  Default: ON

- **DCNAME=*member-name***
  (z/OS only) Specifies the member name of the system within a data sharing group. This name also becomes the system (node) name, overriding the value specified in the system definition. DCNAME must consist of characters A-Z, 0-9, $, #, or @.
  Limits: 1-8 characters

  Note: This parameter is only applicable in a data sharing environment.

- **DEADLOCK_ABEND_ERUS**
  Specifies that ERUS tasks that encounter a database resource deadlock be abnormally terminated. This parameter is meaningful only if the ABEND_ON_DEADLOCK parameter is also specified.

- **DEADLOCK_ABEND_0029**
  Specifies that tasks that abend due to database resource deadlocks use a code of 0029 rather xx29, where "xx" represents the major code of the DML request that was being issued at the time of the abend. This parameter has meaning only if the ABEND_ON_DEADLOCK parameter is also specified.

- **DEADLOCK_DETAILS=ON|OFF**
  ON specifies that more detail be provided in a deadlock situation.
  Default: OFF

- **DICTNAME=*dictionary-name***
  Specifies a dictionary to use when loading a subschema from a load area. For dictionary-related tools, for example, CA IDMS compilers and pre-compilers, and IDMSBCF, DICTNAME specifies the dictionary to access at run time. For SQL applications, DICTNAME specifies the name of the dictionary to connect to at run time.

- **DICTNODE=*dictionary-node-name***
  For SQL applications and dictionary-related tools under the central version, specifies the name of the DC/UCF system that controls the dictionary to access at run time. For applications running in local mode, this parameter is not applicable.

- **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 on SVC screening, see DML Reference Section for Assembler.

- **DLBLMOD=ON|OFF**
  (z/VSE only) ON specifies that the DLBL type in the disk label changes from 'DA' to 'SD' when sequential processing (IDMSQSAM) is activated. After the disk labels are processed as 'SD' during the QSAM file OPEN, the DLBLs are changed back to 'DA' to allow random BDAM processing.
  Default: OFF

- **DMCL=*dmcl-name***
  Specifies the name of the DMCL load module to use in local mode.
  Default: IDMSDMCL

- **DMLTRACE=ON|OFF**
  ON activates a trace facility that traces all navigational DML requests made by an application.
  Default: OFF

- **DSGROUP=*data-sharing-group-name***
  (z/OS only) Specifies the name of the data sharing group,which this system is a member. All CA IDMS systems that are members of the same group must specify the same group name. DSGROUP must consist of characters A-Z, 0-9, $, #, or @. Names that begin with SYS or UNDESIG are reserved and cannot be used. Names that begin with A to I may be in use by the operating system and should be avoided.
  Limits: 1-8 characters

Note: This parameter is only applicable in a data sharing environment.

- **DYNALLOC_WAIT=ON|OFF**
  ON specifies that dynamic allocation does an ENQ wait for the DSN until it becomes available, when a data set is in use.
  OFF specifies that dynamic allocation request fails, when DYNALLOC_WAIT_SECONDS option is zero or not specified.
  This option applies to local mode jobs and to CV startup.
  Default: OFF

- **DYNALLOC_WAIT_SECONDS=nnn**
  Specifies that when a data set is in use, dynamic allocation waits for a specified interval and retries the allocation. If the DSN is still unavailable, it repeats the process until the data set is available. If this option is specified with a non-zero value, it overrides the DYNALLOC_WAIT option. If zero is specified and the DYNALLOC_WAIT option is OFF or not specified; the dynamic allocation request fails.
  This option applies to local mode jobs and to CV startup.
  Limits: 255

- **ECHO=ON|OFF**
  Indicates whether SYSIDMS parameters are displayed on the JES log.
  Default: OFF

- **ENABLE_RRS=ON|OFF**
  (z/OS only) ON activates RRS support for batch applications.
  Default: OFF

- **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 EVAL CENTURY VALIDATION is ON.
  Limits: 1 - 99
  Default: 20

- **EXCLUDE_TCP/IP_STACK=stack=name**
  (z/OS and z/VM only) Creates an EXCLUDE list of up to eight TCP/IP stack names to override the list of stacks supplied by the operating system (z/OS) or through SYSGEN (and z/VM).
  All the stack-names that are specified on these SYSIDMS parameters are excluded from the run-time list of stacks. INCLUDE_TCP/IP_STACK and EXCLUDE_TCP/IP_STACK are mutually exclusive parameters and support wildcards.

- **EXIT14_BATCH_RU**
  Restricts calling EXIT14 to batch run-units.

- **FILE_BUF=*ddname=nnnnn***
  Allows users to increase the number of pages in a buffer used by a specific file for a local mode job without having to change the DMCL. In CV, a DCMT command can be used to alter the number of pages in a buffer. The FILE_BUF parameter provides a similar capability for local mode jobs. If specified, the number of pages in the buffer pool associated with the specified file is increased by *nnnnn*.
  This parameter can be used to tune PREFETCH processing by allowing the local mode user to increase the number of pages in a specific buffer for a job and thereby maximize the benefit of prefetch processing.

- **IDMSQSAM=ON|OFF**
  ON activates the IDMSQSAM facility (sequential access for look-ahead database reads).
  Default: OFF

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

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

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

  Default: OFF

- **INCLUDE_TCP/IP_STACK=stack-name**
  (z/OS and z/VM only) Creates an INCLUDE list of up to eight TCP/IP stack names to override the list of stacks supplied by the operating system (z/OS) or through SYSGEN (z/OS and z/VM). For z /VM only, it can be used to include a new stack.
  All the stack-names that are not specified on these SYSIDMS parameters are excluded from the runtime list of stacks. INCLUDE_TCP/IP STACK and EXCLUDE_TCP/IP_STACK are mutually exclusive parameters and support wildcards.

- **JOURNAL=ON|OFF**
  Specifies whether journaling is performed in local mode. OFF specifies that local mode journaling is not performed, even if there are tape journals defined in the DMCL.
  Default: ON

- **JRNLDTS=*yyyy-mm-dd-hh.mm.ss.nnnnnn***
  Provides a way to bypass a date timestamp mismatch problem between the DMCL and the journal files. The yyy-mm-dd-hh.mm.ss.nnnnnn is the date timestamp on the journal file. This should only be used if you know that the reason for the mismatch will not cause a problem. Inappropriate use of this parameter may cause database corruption.

- **LANG=*xxxxxxxxxxxxxxxxxxx***
  Sets an alternate environment for DBCS support. This parameter is useful for local mode batch jobs and is equivalent to issuing the DCUF SET LANG command for online users.
  Limits: 19 characters

- **LENGTH_PAGE=*nnn***
  Specifies the allowable number of lines on a page.
  Limits: 10 to 32,767 lines
  Default: 60

- **LIMIT_PREFETCH=*nnn***
  When a batch job is performing an area sweep, prefetch takes three-quarters of the number of buffers and divides by the number of records per track to determine the maximum number of start I/Os it can issue. If the number is greater than 100, a value of 100 is used. At the start of a job this can swamp the system with I/O requests.
  By coding LIMIT_PREFETCH=nnn prefetch will only issue nnn I/Os at any time. If this number is greater than the value calculated by CA IDMS, the lower value will be used.

- **LOADAREA=ON|OFF**
  Specifies whether the dictionary load (DDLDCLOD) area is accessed when loading a module. If OFF is specified, the dictionary load area is not accessed. Specify OFF only when all load modules are linked into an OPSYS load library.
  Default: ON

- **LOCAL=ON|OFF**
  Specifies whether a batch job executes in local mode. If ON is specified, all requests are processed locally even if an IDMSOPTI is link-edited with the program, or a SYSCTL file is specified in the JCL.
  Default: OFF

- **LOCAL_DYNAMIC_ALLOCATION= ON|OFF**
  OFF directs a local CA IDMS batch job to ignore any DSN information defined in the DMCL for database files, and requires that the DSN information be included in the JCL to access a database file.
  Default: ON

- **LOCAL_NOJOURNAL_RETRIEVAL**
  Specifies that local batch jobs not journal RETRIEVAL ONLY transactions.

- **LOCALPUR=ON|OFF**
  ON forces purging of the local mode buffer pool whenever a transaction terminates.
  This parameter addresses a change in the way local mode buffers are handled, between 10.21 and later releases, as follows:

  - In release 10.21, a local mode job with multiple transactions (run units) separates buffer pools for each transaction, and each transaction has no knowledge of the others. When a transaction terminates its buffer pool purges.

  - Starting in release 12.0, a local job with multiple transactions has just one buffer pool shared by all transactions. When a transaction terminates the buffer remains unchanged until the last transaction terminates, and the shared buffer pool is purged. To make the system purge the common buffer pool when each transaction terminates (mimicking what happens in release 10.21), use the parm LOCALPUR=ON.
    Default: OFF

    Note: Only use this parameter if there is a compatibility problem, since there can be performance implications in specifying LOCALPUR=ON.

- **MSGDICT=ON|OFF**
  Specifies whether the dictionary message (SYSMSG) area is accessed to retrieve the text of messages. If OFF is specified, the dictionary message area is not accessed. OFF should be specified only when using a DMCL that does not contain the SYSMSG segment, such as during installation.
  Default: ON

- **MULTIDSN=ON|OFF**
  (z/VSE only) ON specifies that tape files may span multiple volumes. At the end of a tape reel, EOF (end of file) or EOV (end of volume) prompts the user to specify an END OF JOB or an END OF VOLUME condition. OFF specifies that END OF JOB is automatically the condition at the end of a tape reel.
  Default: OFF

- **NODENAME=*nodename***
  For non-SQL applications running under the central version, identifies the DC/UCF system to bind to at run time.

- **PARM=*'parameter-string'***
  Allows you to specify parameters typically specified in a JCL EXEC PARM statement. The format is the same as the IBM PARM parameter on the EXEC JCL statement. PARM can contain 1 to 256 characters, and can be specified on multiple lines.

- **PDAT_LOCATION=ANY|BELOW**
  Specifies whether the UCF line driver allocates PDAT storage above or below the 16 MB line. The default, ANY, allocates PDAT storage above the line. BELOW allocates PDAT storage below the line. BELOW must be specified if a back-end PDAT application runs with AMODE 24.
  Default: ANY

- **PREFETCH=ON|OFF**
  OFF overrides the default ON and prevents CA IDMS from prefetching database pages, the normal processing when an area or index sweep is detected. Specify OFF, as follows:

  - For a local batch job to prevent prefetching database pages for the job step.

  - In the SYSIDMS file associated with a central version to prevent prefetching pages for all transactions running with the central version.

  Default: ON

- **PREFETCH_BUF=*nnnnn***
  Specifies the minimum number of pages in a buffer pool that must be present before CA IDMS uses prefetch processing for non-area sweep requests. This parameter applies to both local and central version environments.

- **PRMPT_MSG='*prompt_message*'**
  Customizes the WTOR (Write-To-Operator-With-Reply) message. Customizing this message lets you distinguish the different CVs that are running.
  Enclose *prompt_message* in single (') or double (") quotes. You can write the message in free-form, but, do not enter internal quotes.
  Limits: The message can be 1 to 58 characters (without quotes).
  Default: If the parameter is not specified, the following message is displayed*,* where *nnn* is the system number that is present, regardless if the default or a customized message is displayed.

  ```
  REPLY WITH REQUEST TO IDMS Vnnn
  ```

- **PROCTRACE=ON|OFF**
  ON activates a trace of key user blocks that participate in an SQL PROCEDURE call.
  Default: OFF

- **QSAMAREA=*qsam-area-name***
  Specifies the physical area in the DMCL for which the IDMSQSAM facility does look-ahead reads. If this parameter is omitted and the IDMSQSAM=ON parameter is specified, the look-ahead reads are performed on the first area that is accessed by the transaction.

  Note: This parameter may also be used against journal files when executing the ARCHIVE JOURNAL utility, by specifying QSAMAREA=ARCHIVE.JOURNAL in the utility's SYSIDMS file.

  .

- **QSAMBUF#=*nnn***
  (z/OS only) Specifies the number of buffers to use when the IDMSQSAM facility is active. QSAMBUF# enables you to set the number of QSAM buffers used without coding JCL for the file

being processed by IDMSQSAM.
If QSAMBUF# is not specified, the number of buffers is determined by the DCB parameter BUFNO=nnn, or defaults to 5 buffers.
Limits: 1-255
Default: 5 buffers

- **QSAMTRACE=ON|OFF**
  ON activates a trace of all the IDMSQSAM look-ahead I/O reads. This trace shows the name of the file(s) accessed by IDMSQSAM, each RBN that is read using QSAM or BDAM (DAM/EXCP), and a summary of the number of RBN's read through QSAM and BDAM. It also shows the area being accessed and the number of OPSYS QSAM buffers used as determined by the JCL.
  Default: OFF

- **REREAD_SYSCTL=ON|OFF**
  ON directs local mode operations to reread the SYSCTL file for each new transaction. This allows you to do the following:

  1. Include a SYSCTL in a batch job step's JCL.
  2. Start a transaction that executes under central version, based on the contents of the SYSCTL file.
  3. Deallocate the SYSCTL file defined in the JCL.
  4. Start another transaction to execute in local mode.

  Default: OFF

- **RETRIEVAL_CV**
  Specifies that IDMS converts READY UPDATE requests to READY RETRIEVAL when accessing an area set to RETRIEVAL mode on the CV.

- **ROLLBACK3490**
  Enables the ROLLBACK utility to process archive files residing on devices that do not support backward read, such as disk and 3490E devices.

- **SAVE_SQL_SYNTAX=ON|OFF**
  If ON is specified during CV startup, all dynamic SQL syntax is saved before it gets optimized. The SQL server module IDMSQSRV allocates a piece of long term storage each time an sLTE (that is a secondary LTE representing the SQL session) is acquired. That storage is anchored off the sLTE and is used to save the dynamic SQL syntax before it is processed by the syntax parser and optimized.
  The storage is obtained only once per session and is re-used if there is more SQL syntax to save for the same session. The storage is released during the end of session processing just prior to freeing the sLTE.
  You can obtain the saved SQL syntax by Exit 39 - SQL Syntax Collecting Exit.
  Default: OFF

- **SCRATCH_IN_STORAGE=ON|OFF|ANY|XA|64-bit**
  Enables storage allocation from the operating system for scratch processing.

  - **ON**
    Specifies the same as SCRATCH_IN_STORAGE=ANY.

  - **OFF**
    Specifies to allocate the scratch area as defined in the DMCL.
  - **ANY**
    Acquires 64-bit storage if possible. If the request to allocate 64-bit storage fails, XA storage is acquired.
  - **XA**
    Acquires 31-bit storage.
  - **64-bit**
    Acquires 64-bit storage. If the request to allocate 64-bit storage fails, no attempt to acquire XA storage is done.

  Default: OFF

  Note: Usage of 64-bit storage is controlled by the MEMLIMIT parameter of the JOB or EXEC JCL statement on z/OS, and by the MEMLIMIT option on the SYSDEF MEMOBJ command on z/VSE.

- **SCRATCH_LIMIT=*limit-with-unit***
  Specifies the maximum amount of scratch storage. The system continues to allocate more storage for scratch processing until the sum of all allocations reaches the value specified by *limit-with-unit* . Enter a number in the range 1 to 32767 followed by a unit of KB (Kilobyte: 2**10), MB (Megabyte: 2**20), GB (Gigabyte: 2**30), TB (Terabyte: 2**40), or PB (Petabyte: 2**50).
  The default value is determined as follows:
  - If the DMCL contains a scratch area definition, the default is the number of pages in the area multiplied by the page size.
  - If no scratch area is defined in the DMCL, the default is the size of the primary allocation plus 99 times the size of the secondary allocation.
- **SCRATCH_PRIMARY_EXTENT=*prim-size-with-unit***
  Specifies the primary scratch allocation size. Enter a number in the range 1 to 32767 followed by a unit of KB (Kilobyte: 2**10), MB (Megabyte: 2**20), GB (Gigabyte: 2**30), TB (Terabyte: 2**40), or PB (Petabyte: 2**50).
  The default value is determined as follows:
  - If the DMCL contains a scratch area definition, the default value is the number of pages in the area multiplied by the page size.
  - If no scratch area is defined in the DMCL, the system default value is 1 MB.
- **SCRATCH_SECONDARY_EXTENT=*sec-size-with-unit***
  Specifies the amount of storage to allocate when all existing storage is in use. Enter a number in the range 1 to 32767 followed by a unit of KB (Kilobyte: 2**10), MB (Megabyte: 2**20), GB (Gigabyte: 2**30), TB (Terabyte: 2**40), or PB (Petabyte: 2**50).
  The default size of the secondary allocation is equal to the size of the primary allocation.

- **SHUTDOWN_STALL_TIME=*nnn***
  Causes the stall interval for active user tasks to be set to *nnn* seconds after a SHUTDOWN command is issued. If the parameter is not coded or an interval of 0 (zero) is specified, then there is no effect on SHUTDOWN processing. This parameter allows normal SHUTDOWN processing even if a user task is in a long wait for an unavailable resource. The specified interval is used to force user tasks to be aborted if a given wait exceeds the interval even if the task normally has a longer STALL interval. This parameter affects online application tasks and ERUS tasks in either an INTERNAL or EXTERNAL wait. It does not have any effect on system tasks such as service drivers, line drivers, factotums, and helots.

- **SORT FIELD MAX LEN=n**
  Specifies the maximum sort key field length allowed by a sort.
  Default (z/OS): 2000
  Default (z/VSE): 256

- **SORTSIZE=ON|OFF**
  Directs whether CA IDMS utilities generate the SIZE= sort parameter card. Some sort packages cannot handle the SIZE= parameter.
  Default: OFF. The SIZE= sort parameter is not generated.

- **SQL_CACHE_ENTRIES=n**
  Specifies the maximum number of entries that used in the dynamic SQL cache. One entry holds one cached SQL statement. Specify zero (0) to disable caching. The maximum theoretical value is 2,147,483,647. The actual maximum depends on available memory.
  Default: 200

- **SQL_INTLSORT=ON|OFF**
  Allows you to force the internal CA IDMS sort to be used in local mode. If ON is specified, an internal SORT rather than an operating system SORT will be performed on SQL commands issued in a local batch job that contain an ORDER BY clause. In many cases, an internal SORT is faster than an operating system SORT when you are not dealing with a large amount of data. OFF is the default, indicating an operating system SORT will be used.
  Default: OFF

- **SQLTRACE=ON|OFF**
  ON activates a trace facility of all the SQL database requests made by an application.
  Default: OFF

- **SUPPRESS_RECORD_ON_STATUS=<minor code>**
  Identifies the custom error status minor code that an AFTER GET DB procedure sets when suppressing a record occurrence. This activates Record Suppression Assistance. You can duplicate this line up to three times to support multiple codes. All alphanumeric minor codes, plus, the numeric code 99. Numeric codes 00-98 are reserved for IDMS use.

- **SYS_MSG=UPLOW|UPPER**
  UPPER directs CA IDMS to translate the text of internal #WTL messages to uppercase before being displayed at the output destination. The default is UPLOW. This allows the text of an internal #WTL message issued by CA software to display in mixed case letters.
  Specify UPPER in the following conditions:

  - In local batch jobs to translate any internal #WTL messages issued by CA software to uppercase for that job step.

- In the SYSIDMS file associated with a central version to translate any internal #WTL messages issued by CA software to uppercase for that CV region.

- **SYSCTL=*ddname***
  Specifies an alternate ddname for the SYSCTL file (other than the default ddname of SYSCTL).

- **SYSTRACE=ON|OFF**
  Controls whether basic system tracing is enabled.

  - **ON**
    Enables basic system tracing.
    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.

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

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

  Default: SYSTRK

- **TCP/IP_STACK_*<1 to 8>*=tcp/ip-stack-name**
  In CA IDMS Release 16, the default TCP/IP stack name was extracted from the SYSTCPD file, and the following SYSIDMS parameters can be used to include eight additional stacks to the CA IDMS TCP/IP environment. In Release 17, these parameters are supported for compatibility reasons, but, the method to define the TCP/IP stacks using the INCLUDE STACK clause in SYSGEN, or the two new SYSIDMS parameters INCLUDE_TCP/IP_STACK and EXCLUDE_TCP/IP_STACK take precedence on Release 16 definitions:

  - **TCP/IP_STACK_1=tcp/ip-stack-name**

  - **TCP/IP_STACK_2=tcp/ip-stack-name**

  - **TCP/IP_STACK_3=tcp/ip-stack-name**

  - **TCP/IP_STACK_4=tcp/ip-stack-name**

  - **TCP/IP_STACK_5=tcp/ip-stack-name**

  - **TCP/IP_STACK_6=tcp/ip-stack-name**

  - **TCP/IP_STACK_7=tcp/ip-stack-name**

  - **TCP/IP_STACK_8=tcp/ip-stack-name**

- **TCP/IP_STATUS=ON|OFF**
  Specifies the default status of TCP/IP support at startup. It overwrites the DEFAULT STATUS value defined to SYSGEN. OFF disables TCP/IP support for CA IDMS at startup. ON enables TCP/IP support.

- **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.

- **TRANSACTION_SHARING=ON|OFF**
  ON activates the 'Transaction Sharing' feature for all database activity used in a batch application.
  Default: OFF

- **UPPER=INPUT|OUPUT|BOTH|OFF**
  Specifies whether or not input and/or output files are converted to uppercase:

  - **INPUT**
    Converts SYSIPT input files to uppercase.

  - **OUTPUT**
    Converts SYSLST output files to uppercase.

  - **BOTH**
    Converts both SYSIPT input files and SYSLST output files to uppercase.

  - **OFF**
    Does not convert SYSIPT input files or SYSLST output files to uppercase.

  Default: OFF

- **USERCAT=ON|OFF**
  Specifies whether or not the user catalog is to be accessed. Specify OFF only when formatting the user catalog or when the DMCL does not have access to a user catalog.
  Default: ON

- **WIDTH_PAGE=*nnn***
  Specifies a maximum number of characters to be printed on a SYSLST output line.
  Limits: 71-132
  Default: 132

- **WORK=n**
  CA IDMS utilities that generate SORT input cards for use within a VSE environment typically hard code the number of SORTWK files to be allocated. This value can be insufficient for the amount of data to be sorted. This parameter allows sites to specify the number of SORTWK files to be allocated by their SORT.
  Limits: 1-9
  Any other values are ignored and the default value of the utility is used instead.

- **XA_SCRATCH=ON|OFF**
  Specifies whether or not scratch space is allocated out of XA storage or not.
  Default: OFF. Indicates that a scratch file will be used.

Note: XA_SCRATCH=ON/OFF is maintained only for compatibility reasons. Instead, use SCRATCH_IN_STORAGE instead.

BUFFERSTAT Report Field Descriptions

The following table gives the descriptions for the fields displayed on the report produced by the BUFFERSTAT SYSIDMS parameter.

- ***** Buffer Name *****
  The name of the buffer from the DMCL which has been opened during the processing of this job. Only those buffers which are open or have had some I/O activity against them will appear in this report.
- ***** Pages *****
  The number of pages allocated to the buffer. This is the actual number of pages 'in use' by this buffer for this job. This number is the total from the a) DMCL Local Mode Buffer Pages nnn, or b) JCL DCB=BUFNO=nnn, or c) SYSIDMS Parm FILE_BUF=ddname=nnn.
- ***** Prefetch Minimum *****
  The minimum number of buffers 'in use' needed to allow Prefetch to operate from Random access verbs (that is, non-area sweep processing).
- **DB Page Requests**
  The number of times IDMSDBMS requests a database page from the buffer pool by calling IDMSDBIO.
- **Sequential Area Request**
  Of the DB Page requests in the 'DB Page Requests' count, how many were GET/NEXT/PRIOR in AREA verbs.
- **Random Request**
  Of the DB Page requests in the 'DB Page Requests' count, how many were not GET/NEXT/PRIOR in AREA verbs.
- **Found in Buffer**
  Of the DB Page requests in the 'DB Page Requests' count, how many DB pages were already in the buffer pool and did not require an I/O.
- **Not Found in Buffer**
  Of the DB Page requests in the 'DB Page Requests' count, how many DB pages were not in the buffer pool and therefore required an I/O.
- **Buffer 'hit' Ratio**
  Calculated as 'Found in Buffer' times 100, divided by the DB Page Requests value.
- **Found in Pref Buffer**
  Found in Prefetch Buffer. Of the 'DB Page Requests' count, how many DB pages were found in the buffer pool that had been read by a Prefetch Read?

- **Found in Cache**
  Of the 'DB Page Requests' count, how many DB pages were found in the shared cache?

- **Total DB Pages Read**
  The total number of DB pages read by both Prefetch and standard I/O. This is not the number of I /O's or EXCP's, but the number of DB pages read into a buffer as a standalone I/O, 'Start I/O - Reads', or as part of a Prefetch I/O 'Pages Read - Prefetch'.

- **DB Pg Req:Tot Pages Read**
  The number of 'DB Page Requests' count divided by the 'Total DB Pages Read' count.

- **DB Pg Req:Strt I/O Read**
  The number of 'DB Page Requests' count divided by the 'Start I/O Reads' count.

- **NonPrefetch I/O Rqst**
  The number of standard of non-Prefetch I/O requests. This is the number of the 'DB Pages Request' count which are 'Not found in Buffer' and were not eligible for Prefetch. Prefetch was either not allowed, turned off, or the request was started by a 'Random Request' for which the '*** Prefetch Minimum ***' is higher than the number of '*** Pages ***'.

- **Start I/O - Reads**
  The number of standard or non-Prefetch I/O reads. This number is the result of 'DB Page Requests' which were 'Not found in Buffer' and Prefetch is turned off, or the request was started by a 'Random Request' for which the '*** Prefetch Minimum ***' is higher than the number of '*** Pages ***' count, or the request is not eligible for Prefetch processing. Each of these will show up as 1 EXCP.

- **Start I/O - Writes**
  The number of writes started against the database. Each of these will show up as 1 EXCP.

- **Read forces Write**
  In order to read a DB page into the buffer, a DB page had to be written out to the database first (based on the least recently used algorithm). When this occurs, Prefetch is effectively turned off.

- **Prefetch Requests**
  With Prefetch operating, the number of DB page requests 'Not found in Buffer', which were eligible for Prefetch processing.

- **Sequential Area Request**
  Of the 'Prefetch Requests' count, how many were GET NEXT/PRIOR in AREA type verbs.

- **Random Request**
  Of the 'Prefetch Requests' count, how many were not GET NEXT/PRIOR in AREA type verbs.

- **Pref Req Denied:Buffers**
  Of the 'Prefetch Requests' count, how many did not use Prefetch due to too many buffer pages with the 'must write switch' on (over 1/2 the number of pages in the buffer pool).

- **Start I/O - Prefetch**
  The number of the 'Prefetch Requests' count that actually "Start an I/O" or "Start Subchannel". Each of these will show up as 1 EXCP.

- **Pages Read - Prefetch**
  The number of DB pages that were "carried" with every 'Start I/O - Prefetch'. This number plus the 'Start I/O - Reads' will equal the 'Total DB Pages Read' count.

- **Pref Strt I/O:Pref Req**
  The 'Start I/O - Prefetch' count divided by the 'Prefetch Requests' count.

- **Pref Pages:Pref Strt I/O**
  The 'Pages Read - Prefetch' count divided by the 'Prefetch Requests' count.

- **Pref Pages:Pref Strt I/O**
  The 'Pages Read - Prefetch' count divided by the 'Start I/O - Prefetch' count. This value shows how effective the Prefetch operation is. Compare this number to the "pages per track" to see how effective each Prefetch I/O is. If this number is around 3/1 or less, you probably will not see enough improvement in performance to warrant using Prefetch.

For each file in use, there will be a set of counts:

- Filename -- The DDname of the file using this buffer. Only the files that are open will show up in this report.

- Pgs read -- The number of DB pages read from this file. This is not the number of I/Os or EXCPs.

- Written -- The number of DB pages written to this file. This is the number of I/Os or EXCPs.

- In buffer -- The number of 'DB Page Requests' that were found in the buffer pool which this file maps to.

- In prefetch -- The number of 'DB Page Requests' that were found in the buffer pool which this file maps to due to Prefetch.

z/VSE File Parameters

The following parameters can be used to override default values for z/VSE work files and for the system logical units SYSRDR, SYSIPT, SYSPCH, and SYSLST in batch jobs only.

- **FILENAME=*file-name***
  Specifies the name of the file whose attributes are to be overridden by the following SYSIDMS parameters.

- **BLKSIZE=*block-size***
  Specifies the block size for a file. BLKSIZE and BLOCKS are mutually exclusive parameters.

- **BLOCKS=*block-count***
  Specifies a blocking factor for a file. BLKSIZE and BLOCKS are mutually exclusive parameters.

- **DEVADDR=SYSxxx**
  Specifies a device address for a tape file (SYSIPT, SYSLST, SYSRDR, SYSPCH, or SYSlogical-unit-number).

- **FILABL=STD/NO**
  Specifies a no-label option for a tape file. FILABL=STD is the default.
  Default: FILABL=STD

- **FILETYPE=T/D/I**
  Specifies a file type of tape, disk, or file independent.
- **LRECL=*logical-record-size***
  Specifies the logical record size for a file.
- **RECFM=F/V**
  Specifies if the current file contains fixed-length or variable-length records. V is the default.
  Default: V
- **REWIND=YES/NO/UNLOAD**
  Specifies the position of a tape file when it is opened or closed. REWIND=UNLOAD is the default.
  Default: REWIND=UNLOAD

Security in local mode

Based on how 12.0 central security is implemented in your environment, you may need to add additional JCL statements to the execution JCL for programs running in local mode to define the system dictionary and user catalog.

Note: For more information about security in local mode, see the *CA IDMS Security Administration Guide*.

Deleted CA IDMS modules

Remove references to deleted modules

If any of your 10.2 applications are linked with the CA modules listed below, remove the reference and link edit the program in a 12.0 environment.

- IDMSINTB
- IDMSQSAM
- IDMSTRAC

SYSIDMS replaces the need for IDMSQSAM and IDMSTRAC

SYSIDMS parameters replace the need for IDMSQSAM and IDMSTRAC.

The only way to use IDMSQSAM is to specify it in a SYSIDMS parameter file.

Create a SYSIDMS parameter file in the execution JCL of 10.2 programs that link to IDMSQSAM and IDMSTRAC and link edit the program in a 12.0 environment.

New version of IDMSUTIO

New 12.0 version of IDMSUTIO

If any of your batch programs are linked with IDMSUTIO to make use of file I/O routines, you must relink the program with the 12.0 version of IDMSUTIO.

If you use IDMSUTIO for DBNAME and DBNODE processing, relink the program without IDMSUTIO and remove the parameters from the SYSIPT file. Specify DBNAME and DBNODE parameters using the SYSIDMS parameter file.

COBOL (IDMSDMLC) precompiler JCL

There is new CA IDMS COBOL precompiler (IDMSDMLC) JCL. Review the execution JCL for CA IDMS COBOL programs that use IDMSDMLC and make the necessary modifications.

Note: For more information about IDMSDMLC JCL, see the *CA IDMS DML Reference Guide for COBOL*.

PL/I (IDMSDMLP) precompiler JCL

There is new CA IDMS PL/I precompiler (IDMSDMLP) JCL. Review the execution JCL for CA IDMS PL/I programs that use IDMSDMLP and make the necessary modifications.

Note: For more information about IDMSDMLP JCL, see the *CA IDMS DML Reference Guide for PL/I*.

Considerations for relinking application programs

Changes only necessary on the next recompile or relink

Changes are necessary in the link-edit parameters of some CA IDMS application programs. In all cases, except DC-BATCH applications, these changes are necessary *only when you next recompile or relink a program*; until then, existing load modules will execute successfully.

IDMS is the only entry module

These changes are necessary, because in Release 12.0, IDMS is the one and only entry module to CA IDMS services in batch and DC/UCF execution environments.

IDMS contains several entry points to distinguish different types of services (such as navigational or SQL database requests) or different host languages (such as COBOL or PL/I) in DC/UCF.

Upward compatibility is provided for application programs linked with 10.0 interface modules (IDMSIDVS, IDMSCOBI, IDMSPLI, etc.), but these modules are not distributed as part of the 12.0 software and must be replaced with IDMS the next time the programs are linked.

Considerations by application protocol

Here are considerations by application protocol:

- IDMS-DC COBOL programs
  The next time it is necessary to relink a COBOL program that has a protocol mode of IDMS-DC, you must include module IDMS instead of IDMSCOBI with your application program.

- IDMS-DC PL/I programs
  The next time it is necessary to recompile a PL/I program that has a protocol mode of IDMS-DC, you must do the following:

  - Add the following declaration to your program source:

    ```
    DECLARE IDMSPLI ENTRY OPTIONS (INTER, ASSEMBLER)
    ```

  - Link it with the module IDMS instead of IDMSPLI

- DC-BATCH programs
  All programs, regardless of language, which have a protocol mode of DC-BATCH, must be relinked before being executed in a 12.0 environment. In the new link-edit, replace both IDMSDCBI and IDMSINTB with IDMS.
  PL/I programs must also be recompiled before being executed. You must add the following declaration to your program source:

  ```
  DECLARE IDMSDCP ENTRY OPTIONS (INTER, ASSEMBLER)
  ```

- Batch programs in a z/VSE environment
  The next time it is necessary to relink a COBOL or PL/I program that has a protocol mode of BATCH, you must include the IDMS module and remove IDMSIDVS or IDMSLDPT.

In Release 12.0, the CA IDMS batch environment always uses GETVIS for storage acquisition. When executing programs in a 12.0 environment which were linked with 10.0 IDMS and IDMSLDPT (using COMREG for storage management), it may be necessary to increase the size of the partition or relink the application using the 12.0 IDMS.

SPF indexes

You must convert SPF indexes to integrated indexes before migrating applications that use SPF indexes to a 12.0 environment.

Modify 10.2 schema definitions containing SPF indexes and create appropriate 10.2 subschemas that replace SPF indexes with integrated indexes. Then execute the 10.2 IDMSTBLU utility program to convert SPF indexes to integrated indexes.

After converting SPF indexes in a 10.0 environment, you may want to use the 12.0 UNLINKED index option. If so, you will need to restructure the database in the 12.0 environment.

Note: For more information about restructuring a database, see the *CA IDMS Utilities Guide* .

If an application contains BIND statements for SPF system records, remove the BIND statements and recompile the program.

Region size

You may need to increase the region size for local mode applications to accommodate the increase in the size of the enhanced local mode environment.

Dictionary Segments

This section includes a list of the segments, files, and areas that comprise the dictionaries and databases provided at installation.

System and Application Dictionary Segments

Here is a list of the segments that comprise the installed system and application dictionaries.

Segment	File	Assign	Area
SYSTEM	SYSTEM.DCDML	DCDML	SYSTEM.DDLDML
	SYSTEM.DCRUN	DCRUN	SYSTEM.DDLDCRUN
	SYSTEM.DCLOG	DCLOG	SYSTEM.DDLDCLOG
	SYSTEM.DCSCR	DCSCR	SYSTEM.DDLDCSCR
	SYSTEM.DCLOD	DCLOD	SYSTEM.DDLDCLOD
CATSYS	CATSYS.DCCAT	DCCAT	CATSYS.DDLCAT
	CATSYS.DCCATX	DCCATX	CATSYS.DDLCATX
	CATSYS.DCCATL	DCCAT	CATSYS.DDLCATLOD
APPLDICT	APPLDICT.DICTDB	DICTDB	APPLDICT.DDLDML
	APPLDICT.DLODDB	DLODDB	APPLDICT.DDLDCLOD
SYSDIRL	SYSDIRL.DIRLDB	DIRLDB	SYSDIRL.DDLDML
	SYSDIRL.DIRLLOD	DIRLLOD	SYSDIRL.DDLDCLOD
SYSLOC	SYSLOC.DCLOC	DCLOC	SYSLOC.DDLOCSCR

Segment	File	Assign	Area
SYSMSG	SYSMSG.DCMSG	DCMSG	SYSMSG.DDLDCMSG
SYSUSER	SYSUSER.SECDD	SECDD	SYSUSER.DDLSEC
SYSSQL	SYSSQL.SQLDD	SQLDD	SYSSQL.DDLCAT
	SYSSQL.SQLXDD	SQLXDD	SYSSQL.DDLCATX
	SYSSQL.SQLLOD	SQLLOD	SYSSQL.DDLCATLOD
ASFDICT	ASFDICT.ASFDML	ASFDML	ASFDICT.DDLDML
	ASFDICT.ASFLOD	ASFLOD	ASFDICT.DDLDCLOD
	ASFDICT.DEFN	DEFN	ASFDICT.IDMSR-AREA
	AFSDICT.DATA	DATA	ASFDICT.IDMSR-AREA2
EMPDEMO	EMPDEMO.EMPDEMO	EMPDEMO	EMPDEMO.EMPDEMO
	EMPDEMO.INSDEMO	INSDEMO	EMPDEMO.INSDEMO
	EMPDEMO.ORGDEMO	ORGDEMO	EMPDEMO.ORGDEMO
SQLDEMO	SQLDEMO.EMPLDEMO	EMPLDEMO	SQLDEMO.EMPLAREA
	SQLDEMO.INFODEMO	INFODEMO	SQLDEMO.INFOAREA
	SQLDEMO.INDXDEMO	INDXDEMO	SQLDEMO.INDXAREA
PROJSEG	PROJSEG.PROJDEMO	PROJDEMO	PROJSEG.PROJAREA

- SYSDIRL contains IDMSDIRL output plus CULPRIT report source.
- ASFDICT is optional. You only need it if you are installing ASF.
- SYSSQL is optional. You only need it if you are installing the CA IDMS/SQL Option.
- SQLDEMO is an optional SQL-defined sample database for use with the SQL Option. It is defined in SYSSQL.
- EMPDEMO is an optional non SQL-defined sample database. It is defined in APPLDICT.
- SYSLOC is a scratch area used in local mode. It is optional. If it is not present, DDLDCSCR is used instead.

For more information, see the following topics:

R120DMCL and R120DBTB Listing

R120DMCL Listing

This is an IDMSLOOK report of the Release 12.0 DMCL that defines the 12.0 environment provided at installation.

```
This DMCL uses dbtable R120DBTB         The Operating System is OS
                            Page        Low          High       Page
```

```
                         Low          High
     Area Name              Group       Page          Page      Size    DDNAME      File
 Name                    Page          Page
**********************      *****   **********   **********  ******  ********  *******
****************      **********   **********
APPLDICT.
DDLDML                 0       60,001       62,000   4,276  DICTDB    APPLDICT.
DICTDB                  ***  SAME  ***
APPLDICT.
DDLDCLOD               0       70,001       70,500   4,276  DLODDB    APPLDICT.
DLODDB                  ***  SAME  ***
ASFDICT.
DDLDML                  0       80,001       82,000   4,276  ASFDML    ASFDICT.
ASFDML                   ***  SAME  ***
ASFDICT.IDMSR-
AREA              0       83,001       84,000   4,276  ASFDEFN   ASFDICT.
ASFDEFN                  ***  SAME  ***
ASFDICT.IDMSR-
AREA2             0       85,001       87,000   4,276  ASFDATA   ASFDICT.
ASFDATA                  ***  SAME  ***
ASFDICT.
DDLDCLOD                0       88,001       90,000   4,276  ASFLOD    ASFDICT.
ASFLOD                   ***  SAME  ***
CATSYS.DDLCAT                   0          1          300    4,276  DCCAT     CATSYS.
DCCAT                      ***  SAME  ***
CATSYS.DDLCATX                  0        801          900    4,276  DCCATX    CATSYS.
DCCATX                     ***  SAME  ***
CATSYS.DDLCATLOD                0        901          950    4,276  DCCATL    CATSYS.
DCCATL                     ***  SAME  ***
EMPDEMO.EMP-DEMO-
REGION         0       75,001       75,050   4,276  EMPDEMO   EMPDEMO.
EMPDEMO                  ***  SAME  ***
EMPDEMO.INS-DEMO-
REGION         0       75,101       75,125   4,276  INSDEMO   EMPDEMO.
INSDEMO                  ***  SAME  ***
EMPDEMO.ORG-DEMO-
REGION         0       75,151       75,175   4,276  ORGDEMO   EMPDEMO.
ORGDEMO                  ***  SAME  ***
PROJSEG.
PROJAREA                0       77,401       77,450   4,276  PROJDEMO  PROJSEG.
PROJDEMO                 ***  SAME  ***
SQLDEMO.
EMPLAREA                0       77,001       77,100   4,276  EMPLDEMO  SQLDEMO.
EMPLDEMO                 ***  SAME  ***
SQLDEMO.
INFOAREA                0       77,201       77,250   4,276  INFODEMO  SQLDEMO.
INFODEMO                 ***  SAME  ***
SQLDEMO.
INDXAREA                0       77,301       77,350   4,276  INDXDEMO  SQLDEMO.
INDXDEMO                 ***  SAME  ***
SYSDIRL.
DDLDCLOD                0        4,001        4,010   4,276  DIRLLOD   SYSDIRL.
DIRLLOD                  ***  SAME  ***
SYSDIRL.
DDLDML                  0        5,001        7,000   4,276  DIRLDB    SYSDIRL.
DIRLDB                   ***  SAME  ***
SYSLOC.DDLOCSCR                 0      55,001       57,000    4,276  DCLSCR    SYSLOC.
DCLSCR                     ***  SAME  ***
SYSMSG.DDLDCMSG                 0      10,001       14,000    4,276  DCMSG     SYSMSG.
DCMSG                      ***  SAME  ***
SYSSQL.DDLCAT                   0      20,001       22,000    4,276  SQLDD     SYSSQL.
SQLDD                      ***  SAME  ***
SYSSQL.DDLCATLOD                0      25,001       25,500    4,276  SQLLOD    SYSSQL.
SQLLOD                     ***  SAME  ***
SYSSQL.DDLCATX                  0      28,001       28,500    4,276  SQLXDD    SYSSQL.
SQLXDD                     ***  SAME  ***
SYSTEM.DDLDML                   0       1,001        2,000    4,276  DCDML     SYSTEM.
DCDML                      ***  SAME  ***
SYSTEM.DDLDCLOD                 0       3,001        3,100    4,276  DCLOD     SYSTEM.
DCLOD                      ***  SAME  ***
SYSTEM.DDLDCLOG                 0      30,001       34,000    4,276  DCLOG     SYSTEM.
```

```
DCLOG                      ***  SAME  ***
SYSTEM.DDLDCRUN              0       40,001       41,000   2,676  DCRUN     SYSTEM.
DCRUN                      ***  SAME  ***
SYSTEM.DDLDCSCR              0       50,001       52,000   2,676  DCSCR     SYSTEM.
DCSCR                      ***  SAME  ***
SYSUSER.
DDLSEC                   0       48,001       48,500   4,276  SECDD     SYSUSER.
SECDD                     ***  SAME  ***

   File Name                  DDNAME    Type    Buffer Name         Data Set Name (DS
N)
************************      ********   ****   ******************   *****************
****************************************
APPLDICT.DICTDB              DICTDB     BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.APPLDICT.DDLDML
APPLDICT.DLODDB              DLODDB     BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.APPLDICT.DDLDCLOD
ASFDICT.ASFDATA              ASFDATA    BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.ASFDICT.ASFDATA
ASFDICT.ASFDEFN              ASFDEFN    BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.ASFDICT.ASFDEFN
ASFDICT.ASFDML               ASFDML     BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.ASFDICT.DDLDML
ASFDICT.ASFLOD               ASFLOD     BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.ASFDICT.ASFLOD
CATSYS.DCCAT                 DCCAT      BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.CATSYS.DCCAT
CATSYS.DCCATL                DCCATL     BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.CATSYS.DCCATLOD
CATSYS.DCCATX                DCCATX     BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.CATSYS.DCCATX
EMPDEMO.EMPDEMO              EMPDEMO    BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.EMPDEMO.EMPDEMO
EMPDEMO.INSDEMO              INSDEMO    BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.EMPDEMO.INSDEMO
EMPDEMO.ORGDEMO              ORGDEMO    BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.EMPDEMO.ORGDEMO
PROJSEG.PROJDEMO             PROJDEMO   BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.PROJSEG.PROJDEMO
SQLDEMO.EMPLDEMO             EMPLDEMO   BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SQLDEMO.EMPLDEMO
SQLDEMO.INDXDEMO             INDXDEMO   BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SQLDEMO.INDXDEMO
SQLDEMO.INFODEMO             INFODEMO   BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SQLDEMO.INFODEMO
SYSDIRL.DIRLDB               DIRLDB     BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SYSDIRL.DDLDML
SYSDIRL.DIRLLOD              DIRLLOD    BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SYSDIRL.DDLDCLOD
SYSLOC.DCLSCR                DCLSCR     BDAM   DEFAULT_BUFFER
SYSMSG.DCMSG                 DCMSG      BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SYSMSG.DDLDCMSG
SYSSQL.SQLDD                 SQLDD      BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SYSSQL.DDLCAT
SYSSQL.SQLLOD                SQLLOD     BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SYSSQL.DDLCATL
SYSSQL.SQLXDD                SQLXDD     BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SYSSQL.DDLCATX
SYSTEM.DCDML                 DCDML      BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SYSTEM.DDLDML
SYSTEM.DCLOD                 DCLOD      BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SYSTEM.DDLDCLOD
SYSTEM.DCLOG                 DCLOG      BDAM   LOG_BUFFER           DISP=SHR,
DSN=DBDC.SYSTEM82.SYSTEM.DDLDCLOG
SYSTEM.DCRUN                 DCRUN      BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SYSTEM.DDLDCRUN
SYSTEM.DCSCR                 DCSCR      BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SYSTEM.DDLDCSCR
SYSUSER.SECDD                SECDD      BDAM   DEFAULT_BUFFER       DISP=SHR,
DSN=DBDC.SYSTEM82.SYSUSER.DDLSEC
```

```
            DMCL Journals         Page Size      # of Pages
            *************         **********     **********

            J1JRNL                     2,004          5,000
            J2JRNL                     2,004          5,000
            J3JRNL                     2,004          5,000
            J4JRNL                     2,004          5,000
            SYSJRNL                    8,000        Archive

            Journal Buffers      Buffer Size   # of Buffers
            ***************      ***********   ************

            JNL_BUFFER                 2,004              5
                         Buffer    CV      CV   Total CV     Local  Local  Total Local
DMCL Buffers              Size   Buffers  Type    Size      Buffers  Type    Size
************             ******   ******   **   **********   ******   **   **********

LOG_BUFFER                4,276        5   OS       21,380        5   DC       21,380
DEFAULT_BUFFER            4,276       30   OS      128,280       20   OS       85,520

                  0 Bytes used for CV buffers in DC storage
            149,660 Bytes used for CV buffers in OS storage
            149,660 Bytes used for CV DMCL Buffers

             21,380 Bytes used for LOCAL buffers in DC storage
             85,520 Bytes used for LOCAL buffers in OS storage
            106,900 Bytes used for LOCAL DMCL Buffers
```

R120DBTB Database Name Table

```
    Dbtable=R120DBTB             Compiled Date=91-12-10  02.30.12
                The DEFAULT Dictionary is SYSDICT

DBNAME is *DEFAULT   match on subschema is OPTIONAL
    Subschema IDMSNWK? maps to IDMSNWK? using DBNAME  SYSDICT
    Subschema IDMSCAT? maps to IDMSCAT? using DBNAME  SYSDICT

DBNAME is ASFDICT    match on subschema is OPTIONAL
             Include SEGMENT  ASFDICT
             Include SEGMENT  SYSMSG

DBNAME is SYSDICT    match on subschema is OPTIONAL
             Include SEGMENT  APPLDICT
             Include SEGMENT  SYSMSG
             Include SEGMENT  SYSSQL

DBNAME is SYSDIRL    match on subschema is OPTIONAL
             Include SEGMENT  SYSDIRL
             Include SEGMENT  SYSMSG

DBNAME is SYSTEM     match on subschema is OPTIONAL
             Include SEGMENT  CATSYS
             Include SEGMENT  SYSMSG
             Include SEGMENT  SYSTEM

IDMSLOOK  -   1 Selection Card  Processed
```

External Security Managers

This section explains how the central security system in CA IDMS 12.0 interfaces with CA ACF2 and CA Top Secret. It explains how the upgrade to CA IDMS 12.0. affects the relationship between the external security manager and CA IDMS.

For more information, see the following topics:

- Global considerations (see page 114)
- CA ACF2 conversion (see page 125)
- CA Top Secret conversion (see page 130)
- Sample user exit 14 processing (see page 134)

Global considerations

About this section

The information in this section applies to how CA IDMS central security works with any external security manager. Information specific to a particular external security manager is noted as appropriate.

In release 10.2, CA ACF2 and CA Top Secret security for CA IDMS is provided using the CA ACF2 IDMS and CA Top Secret security subsystems. You must install these subsystems to use either CA ACF2 or CA Top Secret security. CA ACF2 IDMS and CA Top Secret IDMS allow ACF2 and Top Secret to be used as an external security manager for CA IDMS resources.

CA IDMS 12.0 provides an integrated and centralized security manager so that separate security interfaces such as CA ACF2 IDMS and CA Top Secret IDMS do not have to be installed to use CA ACF2 or CA Top Secret.

Signon Processing

Password control

Under CA IDMS 12.0, one-line signons can be controlled by an optional PTF to RHDCSNON.

CA ACF2: One-line SIGNONs to CA IDMS Release 10.2 are controlled by QLOGON parameter of @MOPTS in CA ACF2.

When a user exceeds the password violation threshold defined to the external security manager, the action is not reported to CA IDMS/DC.

Password reverification

In releases of CA IDMS 12.0 prior to the 9212 maintenance release, password reverification is not performed automatically and is not supported by CA IDMS components. A return code 16 (X'10') from a #SECHECK indicates the need to reverify the user's password. Beginning with the 9212 maintenance release, password reverification is performed by CA IDMS central security.

Automatic terminal signon

Automatic terminal signon is optional for CA IDMS 12.0. It can be specified with the DFLTSGN parameter of #SECRTT.

Note: DFLTSGN is valid on the 9212 and later maintenance releases of CA IDMS 12.0.

In processing automatic terminal signon for online tasks, CA IDMS 12.0 uses:

- The VTAM node name
- The PTERM name, for non-VTAM terminals
- The LTERM name, if there is no physical terminal (for example, with queue-initiated tasks)

For batch jobs under the central version, the logonid of the batch address space (that is, the submitter of the batch job) is used for access validation.

CA ACF2: Under CA IDMS 10.2, the DFLTLID parameter on the @MOPT macro in CA ACF2 specified a default logonid used for access validation if no user is signed on.

CA Top Secret: Under CA IDMS 10.2, the ATS Acidname specifies a default logonid used for access validation if no user is signed on.

User session attributes

In CA IDMS 12.0 attributes for a user session are specified in a system profile and/or a user profile. Certain attribute keywords have meaning to CA IDMS, such as PRIORITY (user priority) and CLIST (signon CLIST).

Note: For more information about attributes and profiles, see the *CA IDMS Security Administration Guide* and the *CA IDMS System Operations Guide*.

If the user is defined to CA IDMS, the system and user profile may be explicitly associated with the user definition.

If the user is not defined to CA IDMS (user validation is performed externally), you can define a default user profile in the CA IDMS user catalog, and you can define a default system profile in the system dictionary. The default profiles will be processed at signon if no profiles are associated explicitly with the user definition and:

- The user profile name matches the logonid or the default user profile name specified in the SRTT through the USERPROF parameter of the #SECRTT macro
- The system profile name is 'DEFAULT' or the default system profile name specified in the SRTT through the SYSPROF parameter of the #SECRTT macro

Note: The USERPROF and SYSPROF parameters of the #SECRTT macro are valid on the 9212 and subsequent maintenance releases of CA IDMS. For more information about the SRTT and the #SECRTT macro, see the *CA IDMS Security Administration Guide*.

Multiple signons

Multiple signons to CA IDMS are controlled by the MULTIPLE SIGNON clause of the SYSTEM statement in CA IDMS/DC system generation. This control applies only to signons originating from interactive terminals.

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

CA ACF2: Prior to CA IDMS 12.0, multiple signon was controlled by SIGNQNM.

CA Top Secret: Prior to CA IDMS 12.0, multiple signon was only controlled by the Top Secret facility option SIGN(m). In CA IDMS 12.0, you must still specify SIGN(m) along with the CA IDMS MULTIPLE SIGNON clause to allow multiple signons.

Other considerations

- **Signon authorization (access to system)**
  At signon, a resource check is issued using specifications from the #SECRTT entry for RESTYPE SGON.
  CA ACF2: AUTHFLD in the logonid record is not used in CA IDMS 12.0.
  The base resource name matches the specification for SYSTEM ID in the CA IDMS/DC system generation SYSTEM statement.

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

  Resource class must be specified in the EXTCLS parameter of the #SECRTT entry. The resource name passed by CA-IDMS central security will be built according to the specifications in the EXTNAME parameter of the #SECRTT macro.
  The user must have access to the resource for successful signon to the DC system.

- **Signon message**
  CA IDMS/DC includes a facility to display a "good morning" message at signon. No definition in the external system is required.

- **Security class codes**
  Exit 29 can be used to move a bit map of security class codes for the DEFAULT application to the SON after external SIGNON and before internal SIGNON. These security classes will be used if the ACTI resource type has been secured. The DEFAULT application is a mechanism that allows execution of existing applications that use Release 10.2 security classes without having to explicitly define activities for each application.

  Note: For more information about the ACTI resource type and the DEFAULT application, see the *CA IDMS Security Administration Guide*.

Signon Security Options

Installation Default

At CA IDMS installation, the security option for signon (the SGON resource in the SRTT) is 'OFF'.

This means that when the online user requests signon, or the first security check request is issued on behalf of the executing user in a local mode batch application, signon is unsecured and unvalidated. In an unvalidated signon, the user is successfully signed on whether or not the user ID and password have been defined.

Internal Signon Security

If you specify the internal security option for the SGON resource in an #SECRTT macro, signon is secured and password checking will be performed by the internal security system.

External Signon Security

If you specify the external security option for the SGON resource in a #SECRTT macro, signon is secured and password or PassTicket checking will be performed by the external security system.

If the external security system issues a failure or even a warning on user identification and validation, signon fails. This means that you cannot use internal security as a backup security system when you specify external security for signon.

Using Pass Tickets

The value of the password passed to the external security system and used for external signon authorization can be either a:

- password associated with the user in the external security system
  or

- a PassTicket.

A PassTicket is a temporary substitute for a password that is used for authentication for a specific application. A PassTicket is generated from values associated with the userid and application to which the signon is targeted and is valid for only a short period of time. PassTickets are often used as an alternative to sending a clear text password over a network, thereby improving network security.

Note: For externally secured signons, PassTickets are treated in the same way as passwords in the remainder of the signon processing description.

To enhance performance, CA IDMS uses password caching in multi-signon CV environments. Caching is also applied to PassTickets, as PassTickets are implemented by an external security system.

Under certain circumstances, this means that PassTickets can be used more than once. For this reason, CA recommends the implementation of the following points to maximize security when you implement PassTickets:

- Do not allow multiple signons
- Do not allow signon retention for external run-units
- Start Central Versions as started tasks

When signing in to CA IDMS through IDMS Server, there are some cases where a PassTicket behaves like a password, that means the PassTicket can be used multiple times.

Signon When Security Options are Mixed

The security option for some other resources can be different from the security option for signon. For example, signon security might be external while security for other resources is internal.

In the case of mixed security options, the user must be identified to both the external and internal security systems, and a request for signon invokes signon to both security systems. Password checking is performed by either the external system or the internal system, depending on security option for the SGON resource in the SRTT.

If a DDS connection between two CVs exists and one system has internal security while the other has external security, the CV running with internal security must have the userid of its job defined to allow two-phase commit resynchronization processing to complete successfully across the connected CVs.

What is Signon Processing?

Contents

- Additional Signon Processing (see page 121)

Signon Processing Functions

The major function of signon processing is to identify and validate the user requesting CA IDMS services. In addition, signon processing will also cache user-related information such as the list of groups to which a user belongs and profile information.

Explicit Signon

From within a DC/UCF system, signon processing can be initiated explicitly by executing the SIGNON task code or by linking to RHDCSNON from within a user-written application. If CA IDMS/DC is directly controlling terminal access, then an explicit signon must be issued in order to identify the user accessing DC/UCF from an interactive terminal.

Automatic Signon

Signon processing occurs automatically under the following conditions:

- In local mode batch, signon processing occurs within the batch address space when the first security check is issued.
- Within the central version, system signon processing occurs when the first database request is issued from the externally executing application. This applies to applications executing in batch, CMS, TSO, or a front-end teleprocessing monitor such as DC or CICS.
- In UCF applications, signon processing occurs in the UCF back-end when the UCF connection is made from the front-end application.

General Processing Flow

The processing at each step of signon, and whether or not a particular step is actually executed, is based on a number of factors, such as the environment in which signon is occurring and how signon processing is controlled. These factors and their influence on signon processing are discussed later in this section.

Signon processing consists of the following steps:

1. Identify the user requesting CA IDMS services.
2. In DC/UCF:
   - If a user is already signed to the terminal, sign the user off.
   - If the user is signing on to an interactive terminal and is already signed on to another interactive terminal, deny the signon request unless multiple signon is allowed.
3. Validate the user and password. An asymmetric uni-directional non-unique hash routine is used to "encrypt" the password associated with a user id. When a user signs on, the password entered is processed using the hash routine and compared to the "encrypted" password value associated with the user id.
4. In DC/UCF, update the user's password if requested (explicit signon requests only).

5. Build the group list for the user.
6. Build the session profile from system and user profile information, subject to specifications on the initial #SECRTT.
7. If signon is the result of linking to RHDCSNON, invoke the CLIST identified by the CLIST attribute, if one exists in the session profile.

Identifying the User

Explicit Signon

When an explicit signon request is issued by executing the SIGNON task or linking to RHDCSNON, the user is identified by the user ID specified on the signon request. The password to be used for verification is also specified as part of the signon request.

Automatic Signon

During automatic signon, the user is identified by the authorization ID under which the application is being executed.

If the signon does not occur within the CA IDMS system, the executing user is extracted from the operating system by issuing the appropriate call to the integration services layer of the CA Common Services architecture.

When signon processing is initiated automatically, no password verification is performed by CA IDMS software. CA IDMS assumes that the user has already been validated by the environment within which the application is executing.

Default Signon

You can allow CA IDMS to perform a signon using a specific name when a security check request is issued and the user is not signed on. This is done by specifying DFLTSGN=YES and the DFLTUID parameter on the initial #SECRTT macro.

Note: For more information, see #SECRTT (https://docops.ca.com/display/IDMS19/SECRTT).

User Validation Processing

Dependencies

User validation processing is dependent on the following:

- Whether signon processing is controlled externally, internally or not at all (OFF).
- Whether IDMS resources are controlled externally or internally.
- Whether an explicit or automatic signon is being done.
- Whether the signon is occurring within the DC/UCF system or within a separate address space.

Internally Secured Signon

If signon processing is occurring within a DC/UCF system (whether explicit or automatic), the user must have been granted the SIGNON privilege on the DC/UCF system. If this condition is not satisfied, the user is not signed on and all subsequent security checks will fail.

The user being signed on must also have been defined in the user catalog with a CREATE USER statement, and, in the case of an explicit signon, the password specified must match the password associated with the user definition. If either of these conditions is not satisfied, the user is not signed on and all subsequent security checks will fail.

The user will also be signed on to the external security system if one or more IDMS resources are controlled externally. No password verification takes place for the external signon. If the external signon request fails, the user will not be signed on.

Externally Secured Signon

If signon is controlled externally, the user being signed on must be defined to the external security system. In the case of an explicit signon, the password must match the password associated with the user definition in the external security system or the PassTicket must be validated by the external security system. If the external signon request fails, the user is not signed on and all subsequent security checks will fail.

The user will also be signed on to the internal security system if one or more IDMS resources are controlled internally. No password verification takes place for the internal signon. If the internal signon request fails because the user is not defined to CA IDMS, processing will continue but subsequent security checks for internally controlled resources may fail since the user has no associated groups other than PUBLIC.

No Signon Security

If security for the SGON resource is 'OFF', the user will be signed on to the internal security system without password verification. Regardless of whether the user is defined in the user catalog, processing will continue.

The user will also be signed on to the external security system if one or more IDMS resources are controlled externally. The processing of the external signon request is the same as if signon processing were being controlled internally.

Additional Signon Processing

Updating the Password

In an explicit signon request to CA IDMS/DC, the user can change the password if the user is not already signed on to another terminal. The user can request a change in password during signon processing whether internal or external security is used to control signon processing.

If signon processing is controlled internally, the user's request can be honored if the user catalog (SYSUSER.DDLCSEC area) is available in update mode to the system for which the signon request is issued. Thus, to prevent users from updating their passwords, you can make the user catalog available to users in retrieval mode only.

If signon processing is controlled externally, the user's ability to update the password is subject to any restrictions imposed by the external security system.

Building the User's Group List

As part of internal signon processing, an in-core list of group IDs is built and anchored in the SON control block. The list includes the authorization IDs of all groups of which the user is a member as well as the group PUBLIC.

If signon is secured externally, you can still take advantage of CA IDMS groups to administer security. However, users must be defined in the user catalog in order to be included in a group.

Building the Session Profile

As part of signon processing, the security system will attempt to locate a system profile and a user profile for the user unless directed not to by a USRPROF=OFF or SYSPROF=OFF specification in the initial #SECRTT.

- If no user profile was specified in the user definition, or if there is no user definition in the user catalog (and signon is validated externally), the security system will search the user catalog for, if specified, the user profile designated in the USRPROF= parameter of the initial #SECRTT macro.

- If USRPROF= is not specified, a default user profile definition whose name matches the ID of the signed-on user.

If a user profile is found, the system builds a session profile with the attributes defined in the user profile.

If no **system profile** was specified in the grant of signon privilege to the user, or if there is no grant of signon privilege (and signon is validated externally), the security system will search the system dictionary for the following:

- If specified, the system profile designated in the SYSPROF= parameter of the initial #SECRTT macro.

- If SYSPROF= is not specified, a system profile named 'DEFAULT'.

If a system profile is found, the attributes specified in the system profile are merged into the session profile. If user and system profile attributes match, the attribute value in the system profile takes precedence.

Note: For more information on how to tailor user and system profiles when signon is secured externally, see Securing User Profiles (https://docops.ca.com/display/IDMS19/Securing+User+Profiles).

Signon Control Block

Pointers to Security Data

When the user is successfully signed on, a signon control block (SON) is constructed. User authority data is brought into memory and linked to the SON, as shown in the following illustration:

```
 USER SON     USER-GROUP LIST



           caTEGORY BIT MAP
                   ACTIVITY BIT MAP
```

When Data is Linked to the SON

The following table shows when the various security data are brought into memory and linked to the SON.

Security data	Brought into memory
User-group list	During signon
category bit map	At the first security check which requires categories
Activity bit map	When the application issues its first activity security check

Retaining Signon Information

You can specify that CA IDMS should retain signon information originating from external request units (ERUs). In some situations this provides a performance benefit.

To retain ERU signon information, specify SGNRETN=*time-interval* on the initial #SECRTT macro.

Note: For more information, see #SECRTT (https://docops.ca.com/display/IDMS19/SECRTT).

Resource access validation

Protection by default

For all resource types in the SRTT supplied with CA IDMS 12.0, security is turned *off*.

If the entry for a resource type is modified to specify SECBY=INT or SECBY=EXT on the #SECRTT macro, security for all occurrences of that resource type is turned on.

If the resource is not defined in the SRTT, access to the resource is denied.

Safe lists

Occurrence overrides of the SRTT security option for a resource type provide a capability equivalent to safe lists.

CA ACF2: Safe list capability for CA IDMS 10.2 was provided through @GRCE SAFE.

CA Top Secret: Safe list capability for CA IDMS 10.2 tasks was provided through optional fixes. Safe list capability in 12.0 is provided by occurrence overrides.

CA IDMS 12.0 supports occurrence overrides for:

- Programs (resource SPGM)
- Tasks (resource type TASK)
- Databases (resource type DB)
- User-defined resource types

If SECBY=OFF is coded on an occurrence override, access is always allowed regardless of SIGNON status; that is, such resources are unsecured for public access and can be accessed by a user who is not signed on.

In this example, the #SECRTT macro specifies an occurrence override that allows public access to program RHDCBYE:

```
#SECRTT RESTYPE=SPGM,
      TYPE=OCCURRENCE,
      RESNAME=RHDCBYE,
      SECBY=OFF
```

Suspending user IDs for violation threshold

As distinct from exceeding the password violation threshold, if a user exceeds the violation threshold defined in the external security manager for access to secured resources, this condition is reported by a return code 16 on #SECHECK to CA IDMS. Given this condition, CA IDMS ends the session by signing the user off.

CA Top Secret: Violation threshold is defined with the VTHRESH option.

Suspension of the user ID is controlled by the external security manager.

Controlling where validation will occur

#SECRTT SECBY=EXTERNAL directs that access to the resource is controlled by the external security manager. #SECRTT SECBY=INT or OFF means that access control, if any, will not be provided by the external security manager.

CA ACF2: #SECRTT SECBY= is equivalent to @GRCE VALIDTE=. #SECRTT SECBY=EXT is equivalent to VALIDTE=YES, and #SECRTT SECBY=INT or OFF is equivalent to VALIDTE=NO.

Subschema and area security

Under CA IDMS 12.0, there are two possible approaches to securing subschemas and areas using existing rules in the external security manager:

- Activate database security and supply an SRTT entry for run units (subschemas) and areas. This approach allows safe-listing only at the database level but does not require a user exit.

- Define your own resource types for subschema and area in the SRTT and associate them with the corresponding predefined resources in the external security manager.
  This approach allows safe-listing using occurrence overrides but requires user exit 14.

Detailed information about subschema and area security is presented in the sections later in this section that are specific to each external security manager.

User Exits

Various security exits driven in earlier releases of CA IDMS have been replaced by exits 28 and 29 in CA IDMS 12.0. Exits 28 and 29 are driven for SIGNON, SIGNOFF, resource checks, and initialization. For more information about these exits, see the *CA IDMS Security Administration section*

CA ACF2 conversion

About this section

This section contains information for administrators at sites where CA ACF2 is used to protect CA IDMS resources on a 10.2 system that is to be upgraded to 12.0.

Administration

Specifying security options

For CA IDMS 12.0, MUSOPT (in the address logonid record) is no longer used to specify CA IDMS security options to CA ACF2. Security options are specified in the SRTT (RHDCSRTT), which is loaded at start-up as part of the nucleus. Therefore, RHDCSRTT:

- Should be in the CDMSLIB concatenation

- May be LPA-resident

Resource classes

@GRCE INTTYPE corresponds to #SECRTT parameter RESTYPE=. This table presents the closest CA IDMS correspondences to pre-defined resources in CA ACF2:

CA ACF2 @GRCE INTTYPE	CA IDMS #SECRTT RESTYPE=
PGM, PGN	SPGM
DAT	AREA
TSK	TASK
SSC	NRU

@GRCE INSTYPE= corresponds to #SECRTT EXTCLS=. Additional mapping in CA ACF2 may be required to relate EXTCLS to a specific rule type.

Note: For more information about mapping external classes to ACF2 rule type codes, see the *CA ACF2 Administrator Guide*.

Resource Rule Directories

The RULES parameter of the GRCE macro is not supported in CA IDMS 12.0. If you want resource rule directories to be globally resident, you must explicitly make a change to the CA ACF2 z/OS Global System options to designate them as globally resident. If you designate resource rule directories to be locally resident, there is currently no way in CA IDMS 12.0 to reload or refresh them.

External resource names

For CA ACF2 5.2, be sure resource names will not exceed 40 characters. Define rules to match resource names specified in the SRTT.

For example, assume the #SECRTT entry for resource type TASK was coded:

```
#SECRTT RETYPE=TASK,
      EXTCLS='TSK',
      EXTNAME=(SYSTEM,RESNAME),
      SECBY=EXTERNAL
```

The format for TASK resource names presented to CA ACF2 for validation will be *system-identifier.* *task-code*. If the system identifier is PROD, and PAYU is the task code to be protected, code the rule in CA ACF2/MVS 5.2:

```
ACF
SET RESOURCE(TSK)
COMPILE *
$KEY(PROD.PAYU) TYPE(TSK)
UID(-) ALLOW
```

For CA ACF2 6.0, the rule could be coded:

```
ACF
SET RESOURCE(TSK)
COMPILE *
$KEY(PROD) TYPE(TSK)
PAYU UID(-) ALLOW
```

In the above example, the key is combined with the resource name on the rule line to produce a resource name of PROD.PAYU. This is one approach to creating a definition that allows resource names longer than 40 characters.

Using existing rules

To use existing CA ACF2 rules, code #SECRTT entries in this form:

```
#SECRTT TYPE=ENTRY,
      RESTYPE=resource-type,
      EXTCLS='external-class',
      EXTNAME=RESNAME,
      SECBY=EXTERNAL
```

For example, assume you have defined this rule:

```
ACF
T RESOURCE(TSK)
COMPILE *
$KEY(PAYU) TYPE(TSK)
UID(-) ALLOW
```

The SRTT entry to use this rule would be:

```
#SECRTT TYPE=ENTRY,
     RESTYPE=TASK,
     EXTCLS='TSK',
     EXTNAME=RESNAME,
     SECBY=EXTERNAL
```

Environment integrity

Since security is now an integral part of the CA IDMS/DC nucleus, no RFTIME definition is needed in CA ACF2.

CA-IDMS control blocks can be protected via CA IDMS/DC storage protection.

Defining security schemes by region

If you have multiple central versions and need to implement a different security scheme for each one, you can define a separate #SECRTT macro for each central version. This is a logical equivalent to the functionality provided by the IDMS MODE setting parameter. You can also choose to do any of the following in your CA ACF2 environment to implement a different security scheme for multiple central versions:

- Write general allow-all rule
- Write extensive safe lists
- Use special CA ACF2 z/OS exit to turn off validation

How to define minilid

To define minilid, be sure to:

- Use ACFFDR instead
- Include @MUSASS for each IDMS region logonid
- Insure that each @MUSASS definition refers to an appropriate minilid definition

Note: For more information on defining minilid, see the *CA ACF2 z/OS Systems Programmer Guide*.

Subschema security

Using database security

To secure subschemas using database security, you create SRTT entries similar to these examples:

1. Secure databases externally:

   ```
   #SECRTT TYPE=ENTRY,
        RESTYPE=DB,
        SECBY=EXTERNAL,
        .
        .
        .
   ```

2. Define run unit information for use by external security:

   ```
   #SECRTT TYPE=ENTRY,
        RESTYPE=NRU,
        SECBY=EXTERNAL,
        EXTCLS='SSC',
        EXTNAME=(SSNAME)
   ```

   By securing databases externally, you cause a security check to be routed to CA ACF2 before a run unit is started. By specifying EXTNAME=(SSNAME) on the SRTT entry for NRU, you direct CA IDMS to send the subschema name to CA ACF2 to check the user's authorization to access the resource.

Using user exit 14

To secure subschemas with a user-defined resource, you create SRTT entries similar to these examples:

1. Define the resource and secure it externally:

   ```
   #SECRTT TYPE=ENTRY,
        RESTYPE=SSC,
        SECBY=EXTERNAL,
        EXTCLS='SSC',
        EXTNAME=(RESNAME)
   ```

2. Create a safe list by defining one or more occurrence overrides:

   ```
   #SECRTT TYPE=ENTRY,
        RESTYPE=SSC,
        SECBY=OFF,
        RESNAME=subschema-name
   ```

   Because the SSC resource type is user-defined, you are able to deactivate security for specific subschemas by specifying SECBY=OFF on the occurrence overrides. Security checking is performed by user exit 14, which is called on a BIND RUN UNIT.

   Note: For more information about user exit 14, see the Sample user exit 14 processing (see page 134) and the *CA IDMS System Operations Guide*.

Area security

What CA IDMS provides

CA IDMS protects areas when DB security is activated. This includes protection against access through CA IDMS utilities, such as FORMAT, FIX PAGE, and UNLOCK. When areas are secured externally, READ or UPDATE authority is required for access. Create an SRTT entry for the AREA resource type to secure specify the external resource class for areas:

```
#SECRTT TYPE=ENTRY,
     RESTYPE=AREA,
     SECBY=EXTERNAL,
     EXTCLS='DAT',
     EXTNAME=(RESNAME)
```

All areas in all databases are secured except for areas in a database for which an occurrence override deactivates security. In this case, no areas of the database are secured. Thus, safe-listing is possible only at the database level.

Securing individual areas

You can use existing CA ACF2 rules and secure individual areas by defining a resource type, securing it, associating it with the pre-defined 'DAT' resource in CA ACF2, and then specifying occurrence overrides, as in these examples:

1. Define the resource and secure it externally:

   ```
   #SECRTT TYPE=ENTRY,
        RESTYPE=DAT,
        SECBY=EXTERNAL,
        EXTCLS='DAT',
        EXTNAME=(RESNAME)
   ```

   Note: RESTYPE=DAT names DAT as a user-defined resource to CA IDMS; however, any 4-character name not already defined in the SRTT would be valid. The association with the pre-defined resource 'DAT' in CA ACF2 is made by EXTCLS='DAT'.

2. Create a safe-list by specifying one or more occurrence overrides:

   ```
   #SECRTT TYPE=OCCURRENCE,
        RESTYPE=DAT,
        SECBY=OFF,
        RESNAME=area-name
   ```

   By defining the DAT resource type to CA IDMS and securing it externally, you are able to deactivate security for specific area occurrences. Security checking is performed by user exit 14, which is called on a READY AREA.

   Note: For more information about user exit 14, see the Sample user exit 14 processing (see page 134) and the *CA IDMS System Operations Guide*.

CA ACF2 CA IDMS interface

The status of the security functions supported by the CA ACF2 CA IDMS 10.2 interface in the CA IDMS 12.0 environment is summarized below:

CA ACF2 CA IDMS function	Status in CA IDMS 12.0
ACMSRSON	Supported by the #SECSGON macro
ACMSRSOF	Supported by the #SECSGOF macro
ACMSRRSV	Supported by the #SECCHECK macro
ACMSRRLD	No comparable support
ACMSRRDD	No comparable support
ACMSRDLU	No comparable support
ACMSRPRV	Beginning with the 9212 maintenance release for CA IDMS 12.0, supported through centralized security, but not by a macro. However, client can explicitly code a call to CA ACF2 using ACFSVC with ACVALD or by using HLI.
ACMSRCVT	No comparable support
ACMSRCTL	No comparable support
Note: The functions listed above are described in the CA ACF2 DSECT, ACMSERV.	

Note: For more information about #SECSGON, #SECSGOF, and #SECCHECK see the *CA IDMS Security Administration Guide*. For more information about using ACFSVC or HLI, see the *CA ACF2 Systems Programmer Guide* for the appropriate operating system. For more information about the CA ACF2 CA IDMS interface, see the *CA ACF2 CA IDMS Support Guide* for the appropriate operating system.

CA Top Secret conversion

About this section

This section contains information for administrators at sites where CA Top Secret is used to protect CA IDMS resources on a 10.2 system that is to be upgraded to 12.0.

Administration Information

Resource classes

This table presents #SECRTT parameter RESTYPE= values that most closely correspond to pre-defined CA Top Secret resource classes:

CA Top Secret resource class	CA IDMS #SECRTT RESTYPE=
PROGRAM	SPGM
AREA	AREA
LCF	TASK
SUBSCHEM	NRU

External resource names

CA Top Secret resource names cannot exceed 44 characters; therefore, in constructing external resource names using the #SECRTT EXTNAME parameter, be sure an occurrence of the resource name could not exceed this limit. Keep in mind that:

- In CA IDMS, SQL resource names may be as many as 18 characters
- Ownership of CA Top Secret resources is defined using 1- to 8-character prefixes

Using existing rules

To use existing CA Top Secret rules, code #SECRTT entries in this form:

```
#SECRTT TYPE=ENTRY,
     RESTYPE=resource-type,
     EXTCLS='external-class',
     EXTNAME=RESNAME,
     SECBY=EXT
```

For example, assume you have defined this rule:

```
TSS ADD(USER01) PROGRAM (TSTAA455)
TSS PERMIT(USER22) PROGRAM (TSTAA455)
```

The SRTT entry to use this rule would be:

```
#SECRTT TYPE=ENTRY,
     RESTYPE=SPGM,
     EXTCLS='PROGRAM',
     EXTNAME=RESNAME,
     SECBY=EXTERNAL
```

Task security

Using external security

To secure tasks using external security, you create SRTT entries similar to these examples:

1. Secure tasks externally:

```
#SECRTT TYPE=ENTRY,
     RESTYPE=TASK,
     SECBY=EXTERNAL,
     EXTCLS='LCF',
     EXTNAME=(RESNAME)
```

2. Create occurrence entries for any "safe" tasks. For example, public tasks or custom signon tasks:

```
#SECRTT TYPE=OCCURRENCE,
     RESTYPE=TASK,
     SECBY=OFF,
     RESNAME=ANY1
```

Coding an EXTCLS of LCF provides both an OTRAN and LCF type check when running TSS4.3. In contrast, when running TSS4.2 only an LCF type check is provided. If you want an OTRAN type check performed, then code EXTCLS=OTRAN.

CA Top Secret definitions

OTRAN protected tasks are owned and permitted like normal resources. For example:

```
TSS ADD(acid) OTRAN(PAY1)
TSS PERMIT(acid) OTRAN (PAY1)
```

LCF protection is provided by defining either an inclusive list (TRANS) or an exclusive list (XTRANS), but not both.

The example below shows the definition of an inclusive list, allowing a user access to only the transactions listed.

```
TSS ADD(acid) TRANS(IDMSPROD,(PAY1,PAY2))
```

The example below shows the definition of an exclusive list which denies a user access to the transactions listed.

```
TSS ADD(acid) XTRANS(IDMSPROD,(PAY5,PAY6))
```

Secure Subschema

Using database security

To secure subschemas using database security, you create SRTT entries similar to these examples:

1. Secure databases externally:

```
#SECRTT TYPE=ENTRY,
     RESTYPE=DB,
     SECBY=EXTERNAL,
     .
     .
     .
```

2. Define run unit information for use by external security:

```
#SECRTT TYPE=ENTRY,
     RESTYPE=NRU,
     SECBY=EXTERNAL,
     EXTCLS='SUBSCHEM',
     EXTNAME=(SSNAME)
```

By securing databases externally, you cause a security check to be routed to CA Top Secret before a run unit is started. By specifying EXTNAME=(SSNAME) on the SRTT entry for NRU, you direct CA IDMS to send the subschema name to CA Top Secret to check the user's authorization to access the resource.

Using user exit 14

To secure subschemas with a user-defined resource, you create SRTT entries similar to these examples:

1. Define the resource and secure it externally:

   ```
   #SECRTT TYPE=ENTRY,
        RESTYPE=SSCH,
        SECBY=EXTERNAL,
        EXTCLS='SUBSCHEM',
        EXTNAME=(RESNAME)
   ```

   This entry defines SSCH as a resource type in CA IDMS and associates it (through the EXTCLS= parameter) with the pre-defined SUBSCHEM resource in CA Top Secret.

2. Create a safe list by specifying one or more occurrence overrides:

   ```
   #SECRTT TYPE=ENTRY,
        RESTYPE=SSCH,
        SECBY=OFF,
        RESNAME=subschema-name
   ```

   Since the SSCH resource type is user-defined, you are able to specify occurrence overrides for it. Overrrides that specify SECBY=OFF effectively deactivate security for the specified subschema occurrences. Security checking is performed by user exit 14, which is called on a BIND RUN UNIT.

   Note: For more information about user exit 14, see the Sample user exit 14 processing (see page 134) and the *CA IDMS System Operations Guide*.

CA Top Secret definitions

In CA Top Secret you establish ownership of a subschema and grant access to the subschema, as in these examples:

```
TSS ADD(acid) SUBSCHEM(subschema-name)
TSS PERMIT(acid) SUBSCHEM(subschema-name)
```

What is Area Security?

What CA IDMS provides

CA IDMS protects areas when DB security is activated. This includes protection against access through CA IDMS utilities, such as FORMAT, FIX PAGE, and UNLOCK. When areas are secured externally, READ or UPDATE authority is required for access. Create an SRTT entry for the AREA resource type to secure specify the external resource class for areas:

```
#SECRTT TYPE=ENTRY,
     RESTYPE=AREA,
     SECBY=EXTERNAL,
     EXTCLS='AREA',
     EXTNAME=(RESNAME)
```

All areas in all databases are secured except for areas in a database for which an occurrence override deactivates security. In this case, no areas of the database are secured. Thus, safe-listing is possible only at the database level.

Securing individual areas

You can use existing CA Top Secret rules and secure individual areas by defining a resource type, securing it, associating it with the pre-defined 'AREA' resource in CA Top Secret, and then specifying occurrence overrides, as in these examples:

1. Define the resource and secure it externally:

   ```
   #SECRTT TYPE=ENTRY,
        RESTYPE=IDAR,
        SECBY=EXTERNAL,
        EXTCLS='AREA',
        EXTNAME=(RESNAME)
   ```

2. Create a safe list by specifying one or more occurrence overrides:

   ```
   #SECRTT TYPE=ENTRY,
        RESTYPE=IDAR,
        SECBY=OFF,
        RESNAME=area-name
   ```

   Since the IDAR resource type is user-defined, you are able to specify occurrence overrides for it. Overrrides that specify SECBY=OFF effectively deactivate security for the specified area occurrences. Security checking is performed by user exit 14, which is called on a READY AREA.

Sample user exit 14 processing

The examples below are intended to show you how to use exit 14 and are not intended to be used as actual executable routines.

Subschema example

This code presents an example of validating access to a subschema resource (an occurrence of user-defined resource type SSC) on a BIND RUN UNIT using user exit 14:

```
SECWORK  DSECT
WKAUTHB  DS    CL6
WKSRB    DS    CL(SRBSCLEN)
SECWKLNF EQU   ((*-SECWORK+3)/4)*4
          :
         LM    R6,R7,0(R1)
        USING  SSC,R7
```

```
         C     R6,=F'59'
         BE    E14BND
         C     R6,=F'97'
         BNE   E14RDY
         XR    R15,R15
E14RTN   DS    0H
         #RTN
*---------------------------------------------------------------------
*  Request is BIND RUN UNIT
*---------------------------------------------------------------------
E14BND   DS    0H
         #GETSTK =SECWKLNF,REG=(R6)
        USING  SECWORK,R6
*---------------------------------------------------------------------
*  Set status to 1469 if security violation.  Site may choose to
*  return a status other than 1469.
*---------------------------------------------------------------------
         L     R2,SSCIDBCM+16
         #SECHECK SRB=WKSRB,                                          X
               RESTYPE='SSC',                                         X
               RESNAME=(R2),                                          X
               AUTHRTY=EXECUTE,                                       X
               RGSV=(R2-R8)
         LTR   R15,R15
         BZ    E14RTN
         MVC   SSCSTAT,=C'1469'
         B     E14RT
```

Area example

This code presents an example of validating access to an area resource (an occurrence of user-defined resource type DAT) on a READY AREA using user exit 14:

```
*---------------------------------------------------------------------
*  Request is READY AREA of some sort
*---------------------------------------------------------------------
E14RDY   DS    0H
         #GETSTK =SECWKLNF,REG=(R6)
        USING  SECWORK,R6
         MVC   WKAUTHB,CSANULLS
*---------------------------------------------------------------------
*  If retrieval usage mode
*---------------------------------------------------------------------
         OI    WKAUTHB,SAARSELM
         B     E14SCHKA
*---------------------------------------------------------------------
*  If update usage mode
*---------------------------------------------------------------------
         OI    WKAUTB,SAARUPDM
*---------------------------------------------------------------------
*  Set status 0966 if security violation.  Site may choose to return
*  a status other tham 0966.
*---------------------------------------------------------------------
E14SCHKA DS    0H
         L     R2,SSCIDBCM+12
         #SECHECK SRB=WKSRB,                                          X
               RESTYPE='DAT',                                         X
               RESNAME=(R2),                                          X
               AUTHRTY=WKAUTHB,                                       X
               RGSV=(R2-R8)
         LTR   R15,R15
         BZ    E14RTN
         MVC   SSCSTAT,=C'0966'
         B     E14RTN
```

DSECT Replacement

With CA IDMS 12.0, you must generate several database-related DSECTs using macro statements rather than COPY books. This section lists 10.2 COPY books replaced by new macros in 12.0.

New 12.0 Macros

Macros replacing 10.2 COPY books

The following table identifies the 12.0 macros which replace 10.2 COPY books. New 12.0 macros which you must generate by a macro statement are also listed.

If you have programs that reference the 10.2 COPY book, replace the COPY book with the appropriate 12.0 macro.

10.2 COPY book	New macro	Comments
COPY #DMCEQ		Obsolete
COPY #DMCDS	#DMCLDS	All DMCL blocks
None	#SUBCB VARS=YES	All SUBSCHEMA blocks
COPY #SUBEQ	#SUBCB VARS=YES, TYPE=EQUATES	All SUBSCHEMA blocks using EQUs
COPY #AFMDS	#DMCLDS TYPE=AFMDS	FM61 block
COPY #BCRDS	#DMCLDS TYPE=BCRDS	BC53 block
COPY #DPRDS	#DMCLDS TYPE=DPRDS	PR60 block
COPY #FCBDS	#DMCLDS TYPE=FCBDS	FC59 block
COPY #JBCDS	#DMCLDS TYPE=JBCDS	JB63 block
COPY #JCBDS	#DMCLDS TYPE=JCBDS	JD62 block
None	#DMCLDS TYPE=SEGDS	SG49 block
None	#DMCLDS TYPE=SYMDS	SY40 block
COPY #VACDS	#FACDS VARS=YES	AC56 and VAC blocks
COPY #VCRDS	#FCRDS VARS=YES	CR55 and VCR blocks
None	#FFKDS VARS=YES	FK76 and VFK blocks
COPY #VMRDS	#FMRDS VARS=YES	MR53 and VMR blocks
COPY #VORDS	#FORDS VARS=YES	OR52 and VOR blocks
COPY #VPCDS	#FPCDS VARS=YES	PC70 and VPC blocks
None	#FSADS VARS=YES	SA73 and VSA blocks
COPY #VSRDS	#FSRDS VARS=YES	SR51 and VSR blocks
COPY #VLRDS	#FLRDS VARS=YES	LR80 and VLR blocks

10.2 COPY book	New macro	Comments
COPY #VLQDS	#FLQDS VARS=YES	LQ81 and VLQ blocks
COPY #VPSDS	#FPSDS VARS=YES	PS83 and VPS blocks
COPY #FACDS	#FACDS	AC56 block
COPY #FAMDS	#FAMDS	AM57 block
COPY #FAPDS	#FAPDS	AP72 block
COPY #FCRDS	#FCRDS	CR55 block
COPY #FFDDS	#FFDDS	FD54 block
COPY #FIBDS	#FIBDS	IB50 block
None	#FFKDS	FK76 block
None	#FLBDS	LB74 block
COPY #FMRDS	#FMRDS	MR53 block
COPY #FORDS	#FORDS	OR52 block
COPY #FPCDS	#FPCDS	PC70 block
COPY #FRPDS	#FRPDS	RP71 block
None	#FSADS	SA73 block
COPY #FSRDS	#FSRDS	SR51 block
COPY #FLRDS	#FLRDS	LR80 block
COPY #FLQDS	#FLQDS	LQ81 block
COPY #FPHDS	#FPHDS	PH85 block
COPY #FPSDS	#FPSDS	PS83 block
COPY #FSDDS	#FSDDS	SD82 block