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

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

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

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

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

The manufacturer of this Documentation is CA.

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

Copyright © 2017 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.
# Table of Contents

## Create and Generate Physical Database ........................................ 16
- CA IDMS conversion activities .................................................. 17
- Release 12.0 test environment .................................................. 22
- New Installation Features ....................................................... 23
- Combined Installation Manual .................................................. 23
- Separate Installs ................................................................. 25
- CA IDMS Installation Types ...................................................... 25
  - Complete Base Installation ............................................... 26
  - Upgrade Installation ......................................................... 26
  - Add-On Installation .......................................................... 27
- Product Distribution and Installation Materials ......................... 27
- Preinstallation Considerations ................................................. 28
  - Installation Configuration Updates for 18.5.00 ....................... 29
- Installation Methodology .......................................................... 30
  - MSHP .............................................................................. 30
  - Sample JCL for CA IDMS .................................................. 31
  - Customizing JCL .............................................................. 31
  - Loading the Product Code .................................................. 31
    - Mounting the Install Media as a Virtual Tape ....................... 31
- System Generation Basics ....................................................... 32
- Using This Guide ..................................................................... 32

## DC/UCF System Generation ...................................................... 33
- Online Components and System Tasks ........................................ 34
- Data Dictionary Entities ........................................................... 36
- Source and Object Records ..................................................... 38
- DC/UCF System Reports .......................................................... 38
  - Automatic Tuning ............................................................ 40
    - How Automatic Tuning Works ........................................ 41

## DC/UCF Concepts .................................................................. 41
- Using the DDLDCLOG Area ..................................................... 41
System Generation Compiler .............................................................................................................. 76

Using System Generation.................................................................................................................. 77

Compiler Activities.......................................................................................................................... 77

System Definition Backup ................................................................................................................. 79

Currency ......................................................................................................................................... 80

Coding Considerations..................................................................................................................... 81

Statement Format.............................................................................................................................. 81
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using IS and '=' Interchangeably</td>
<td>83</td>
</tr>
<tr>
<td>Delimiters</td>
<td>83</td>
</tr>
<tr>
<td>Quotation Marks</td>
<td>84</td>
</tr>
<tr>
<td>Input Lines</td>
<td>84</td>
</tr>
<tr>
<td>Comments</td>
<td>85</td>
</tr>
<tr>
<td>Carriage Control Statements</td>
<td>86</td>
</tr>
<tr>
<td>SKIP Statement Syntax</td>
<td>86</td>
</tr>
<tr>
<td>SKIP Statement Parameters</td>
<td>86</td>
</tr>
<tr>
<td>EJECT Statement Syntax</td>
<td>87</td>
</tr>
<tr>
<td>EJECT Statement Parameters</td>
<td>87</td>
</tr>
<tr>
<td>Adding, Modifying, and Deleting Entities</td>
<td>87</td>
</tr>
<tr>
<td>ADD Verb</td>
<td>87</td>
</tr>
<tr>
<td>MODIFY Verb</td>
<td>89</td>
</tr>
<tr>
<td>DELETE Verb</td>
<td>90</td>
</tr>
<tr>
<td>Displaying and Punching Entities</td>
<td>92</td>
</tr>
<tr>
<td>DISPLAY Syntax</td>
<td>93</td>
</tr>
<tr>
<td>DISPLAY Parameters</td>
<td>93</td>
</tr>
<tr>
<td>MODULE Syntax</td>
<td>95</td>
</tr>
<tr>
<td>MODULE Parameters</td>
<td>95</td>
</tr>
<tr>
<td>Example MODULE</td>
<td>96</td>
</tr>
<tr>
<td>Compiler-Directive Statements</td>
<td>97</td>
</tr>
<tr>
<td>SET OPTIONS Statement</td>
<td>98</td>
</tr>
<tr>
<td>SET OPTIONS Statement Syntax</td>
<td>99</td>
</tr>
<tr>
<td>SET OPTIONS Statement Parameters</td>
<td>99</td>
</tr>
<tr>
<td>Example SET OPTIONS Statement</td>
<td>103</td>
</tr>
<tr>
<td>DISPLAY/PUNCH OPTIONS Statement</td>
<td>104</td>
</tr>
<tr>
<td>DISPLAY/PUNCH OPTIONS Statement Syntax</td>
<td>104</td>
</tr>
<tr>
<td>DISPLAY/PUNCH OPTIONS Statement Parameters</td>
<td>104</td>
</tr>
<tr>
<td>Example DISPLAY/PUNCH OPTIONS Statement</td>
<td>105</td>
</tr>
<tr>
<td>SIGNOFF Statement</td>
<td>106</td>
</tr>
<tr>
<td>SIGNOFF Statement Syntax</td>
<td>106</td>
</tr>
<tr>
<td>Example SIGNOFF Statement</td>
<td>106</td>
</tr>
<tr>
<td>SIGNON Statement -- Initiates an Online or a Batch Session</td>
<td>107</td>
</tr>
<tr>
<td>SIGNON Statement Syntax</td>
<td>108</td>
</tr>
<tr>
<td>SIGNON Statement Parameters</td>
<td>108</td>
</tr>
<tr>
<td>Example SIGNON Statement</td>
<td>110</td>
</tr>
<tr>
<td>GENERATE Statement</td>
<td>111</td>
</tr>
<tr>
<td>GENERATE Statement Syntax</td>
<td>111</td>
</tr>
<tr>
<td>COPY Statement</td>
<td>111</td>
</tr>
<tr>
<td>COPY Statement Authorization</td>
<td>111</td>
</tr>
<tr>
<td>COPY Statement Syntax</td>
<td>112</td>
</tr>
<tr>
<td>COPY Statement Parameters</td>
<td>112</td>
</tr>
</tbody>
</table>
Teleprocessing Network Statements .................................................. 280

LINE Statement ........................................................................... 281
  LINE Statement Syntax ................................................................. 281
  LINE Statement Parameters .......................................................... 282
  Example LINE Statement ............................................................... 284

PTERM Statement .......................................................................... 284
  PTERM Statement Syntax ............................................................... 285
  PTERM Statement Parameters .......................................................... 286
  PTERM Statement Usage ................................................................. 288
  Example PTERM Statement ............................................................ 289
  More Information ............................................................................ 289

LTERM Statement ........................................................................... 290
  LTERM Statement Syntax ............................................................... 290
  LTERM Statement Parameters .......................................................... 291
  LTERM Statement Usage ................................................................. 295
  Example LTERM Statement ............................................................ 296

Device Definitions .......................................................................... 297

ASYNC ............................................................................................. 299
  ASYNC Syntax .............................................................................. 299
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASYNC Parameters</td>
<td>299</td>
</tr>
<tr>
<td>ASYNC Usage</td>
<td>300</td>
</tr>
<tr>
<td>Example ASYNC</td>
<td>301</td>
</tr>
<tr>
<td>BSC2</td>
<td>301</td>
</tr>
<tr>
<td>BSC2 Syntax</td>
<td>301</td>
</tr>
<tr>
<td>BSC2 Parameters</td>
<td>303</td>
</tr>
<tr>
<td>BSC3</td>
<td>307</td>
</tr>
<tr>
<td>BSC3 Syntax</td>
<td>307</td>
</tr>
<tr>
<td>BSC3 Parameters</td>
<td>309</td>
</tr>
<tr>
<td>Example BSC3</td>
<td>313</td>
</tr>
<tr>
<td>CCI</td>
<td>314</td>
</tr>
<tr>
<td>CCI Syntax</td>
<td>314</td>
</tr>
<tr>
<td>CCI Parameters</td>
<td>314</td>
</tr>
<tr>
<td>CONSOLE</td>
<td>315</td>
</tr>
<tr>
<td>CONSOLE Syntax</td>
<td>315</td>
</tr>
<tr>
<td>CONSOLE Parameter</td>
<td>315</td>
</tr>
<tr>
<td>Example 1</td>
<td>315</td>
</tr>
<tr>
<td>DDS</td>
<td>316</td>
</tr>
<tr>
<td>DDS Syntax</td>
<td>316</td>
</tr>
<tr>
<td>DDS Parameters</td>
<td>316</td>
</tr>
<tr>
<td>DDS Usage</td>
<td>317</td>
</tr>
<tr>
<td>Example DDS</td>
<td>318</td>
</tr>
<tr>
<td>INOUTL</td>
<td>318</td>
</tr>
<tr>
<td>INOUTL Syntax</td>
<td>319</td>
</tr>
<tr>
<td>INOUTL Parameters</td>
<td>319</td>
</tr>
<tr>
<td>INOUTL Usage</td>
<td>320</td>
</tr>
<tr>
<td>INOUTL Example</td>
<td>320</td>
</tr>
<tr>
<td>LAPPCEMU</td>
<td>321</td>
</tr>
<tr>
<td>LAPPCEMU Syntax</td>
<td>321</td>
</tr>
<tr>
<td>LAPPCEMU Parameters</td>
<td>321</td>
</tr>
<tr>
<td>Example LAPPCEMU</td>
<td>321</td>
</tr>
<tr>
<td>L3270B</td>
<td>322</td>
</tr>
<tr>
<td>L3270B Syntax</td>
<td>322</td>
</tr>
<tr>
<td>L3270B Parameters</td>
<td>322</td>
</tr>
<tr>
<td>Example 3</td>
<td>322</td>
</tr>
<tr>
<td>L3280B</td>
<td>323</td>
</tr>
<tr>
<td>L3280B Syntax</td>
<td>323</td>
</tr>
<tr>
<td>L3280B Parameters</td>
<td>324</td>
</tr>
<tr>
<td>Example L3280B</td>
<td>325</td>
</tr>
<tr>
<td>SOCKET</td>
<td>325</td>
</tr>
<tr>
<td>SOCKET Syntax</td>
<td>326</td>
</tr>
</tbody>
</table>
## VTAM Considerations

- VTAM Entries .......................................................... 359
- System Generation Entries 1 ........................................... 360
- DC/UCF System Generation Statements ............................... 361
- VTAMLST Entries ........................................................ 364
- APPL Type Major Node Definition ....................................... 364
- Sample LOCAL Type Major Node ......................................... 364
- Mode Table .............................................................. 364
- Runtime Considerations ...................................................... 367
  - Runtime Flow of Control ................................................... 367
  - Specifying File Characteristics .......................................... 371
    - Where to Specify File Characteristics ................................ 372
    - Specifying a Record Length ........................................... 373
    - Specifying a Block Size ............................................... 373
    - Default File Characteristics .......................................... 374
  - Specifying Characteristics of a Concatenated Data Set .......... 374
  - Specifying Characteristics of a Suspense File .................... 375
  - Tape Labels ............................................................ 375
- Application Components and Physical Data Sets .................. 376
- Application Restartability .................................................. 377
  - Checkpoint Record .................................................... 377
  - Taking a Checkpoint at Runtime ..................................... 378
  - Restarting the Application After an Abend ....................... 379
- CA ADS Batch Trace Facility .............................................. 380
- Multiple External Run Units Under CA ADS Batch .................. 383
- Operator Shutdown ....................................................... 384
- Runtime Control Parameters .............................................. 385
  - PARM Field Parameters ................................................ 385
  - Control Statements .................................................... 386
- Loading Runtime Components ............................................ 393
- JCL ............................................................................ 394
- Real APPC Support Considerations ..................................... 398

## SNA and LU Considerations

- SNA Terminology .......................................................... 399
- Bind Parameters for LU 6.2 Sessions .................................... 399
- LU 6.2 Restrictions ........................................................ 400
- Multiple Session Support .................................................. 400
- Sample Definitions for SNA Support ..................................... 401
Using System Generation

This section describes the activities involved in converting a CA IDMS Release 10.2 environment to a Release 12.0 environment. In addition, the Release 12.0 test environment created during the installation of CA IDMS is described. The conversion process assumes that CA IDMS Release 12.0 is installed in your environment.

Before you begin converting any part of your CA IDMS Release 10.2 environment, you should:
- Read the documents identified in the preface to this document
- Read the table of CA IDMS conversion activities presented at the beginning of this section
- Review the information in the sections on the installed 12.0 test environment. You can use this environment to perform most of the conversion tasks.
- Choose a small, test 10.2 environment to convert before you convert any production environments. Once you have experience with the conversion process, conversion tools, and new CA IDMS facilities you will be better prepared to convert your larger, production environment.
- Create a conversion plan. Identify the steps you will take to convert your initial environment. Creating a conversion plan, before you begin, will help to organize your conversion effort and keep track of conversion activities once you get started.
  - Create and Generate Physical Database (see page 16)
  - DC/UCF System Generation (see page 33)
  - DC/UCF Concepts (see page 41)
  - System Generation Compiler (see page 76)
  - System Definition (see page 127)
  - SYSTEM Statement (see page 131)
  - System Generation Statements (see page 180)
  - Teleprocessing Network Statements (see page 280)
  - System Generation Data Dictionary Structure (see page 353)
  - System Programs and Tasks (see page 355)
  - VTAM Considerations (see page 359)
  - SNA and LU Considerations (see page 399)
  - Sample System Definition (see page 404)
  - Tailoring the Banner Page (see page 413)
  - TCAM Considerations (see page 414)
  - IDMSLBLS Procedure for z/VSE JCL (see page 415)
  - SYSGEN User Exit Program (see page 423)

Create and Generate Physical Database

To create and generate Release physical database definition components, follow these steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| Create 12.0 physical database definition statements from 10.2 DMCL definitions | Identify the 10.2 DMCLs to convert to 12.0
| | Ensure the 10.2 DMCLs are validated
| | Code input statements to the DMCL Syntax Generator to identify the DMCL you want to convert
| | Process 10.2 DMLCs by executing IDMSDMCC against the 10.2 data dictionary that contains the DMCL
**CA IDMS conversion activities**

The following table lists the activities involved in converting to CA IDMS Release 12.0. A detailed description of most conversion activities is provided in subsequent sections of this document. There is one section for most major activities identified in the table. The sections are presented in the order that the conversion activity appears in the table.

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Conversion activity</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Database Definition (<a href="https://docs.ca.com/display/IDMS19/Physical+Database+Definition">https://docs.ca.com/display/IDMS19/Physical+Database+Definition</a>)</td>
<td>Physical database definition</td>
<td>Use the DMCL Syntax Generator to generate 12.0 physical database definition statements from 10.2 DMCL definition.</td>
</tr>
<tr>
<td>Create DMCL</td>
<td>Create database name table</td>
<td></td>
</tr>
<tr>
<td>Chapter</td>
<td>Conversion activity</td>
<td>Action</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modify statements as needed and submit them to the CA IDMS Batch Command Facility to populate the dictionary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generate and punch DMCL load module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Determine requirements for 12.0 database name table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Define database name table by submitting appropriate statements to the CA IDMS Batch Command Facility.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generate and punch database name table load module.</td>
</tr>
<tr>
<td></td>
<td>Migrate 10.0 security definitions to 12.0 central security definition.</td>
<td>Populate security system using CA IDMS Command Facility.</td>
</tr>
<tr>
<td></td>
<td>Review 12.0 security statements.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Populate security system with 12.0 security definitions.</td>
<td></td>
</tr>
<tr>
<td>Dictionary Migration and Setup (<a href="https://docops.ca.com/display/IDMS19/Dictionary+Migration+and+Setup">https://docops.ca.com/display/IDMS19/Dictionary+Migration+and+Setup</a>)</td>
<td>Dictionary migration and setup</td>
<td>If the space utilization of any of your DDLDML areas is at or close to 70%, consider increasing the size of it.</td>
</tr>
<tr>
<td>Area sizing</td>
<td>Determine the space available in each DDLDML area you will convert.</td>
<td>If necessary, increase the size of the DDLDML area that will contain the 12.0 CA-supplied internal schemas.</td>
</tr>
<tr>
<td></td>
<td>12.0 DDLDML area records and sets do not require an increase in the size of 12.0 DDLDML areas.</td>
<td>Use the dictionary migration utility to migrate 10.0 DDLDML areas.</td>
</tr>
<tr>
<td></td>
<td>CA-supplied internal schema and subschema definitions may require 5% additional space in the DDLDML area where they reside.</td>
<td>Load the CA IDMS message module from the source library provided during CA IDMS installation into your message area.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Load CA IDMS system task and program modules from the source library provided during CA IDMS installation into your system dictionary.</td>
</tr>
</tbody>
</table>
### Chapter 17: Conversion activity

<table>
<thead>
<tr>
<th>Conversion activity</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate 10.0 DDLDML areas to 12.0 format.</td>
<td>Load the database protocol module from the source library provided during CA IDMS installation into each of your migrated application dictionaries.</td>
</tr>
<tr>
<td>New messages.</td>
<td></td>
</tr>
<tr>
<td>CA IDMS system task and program modules.</td>
<td></td>
</tr>
<tr>
<td>CA IDMS database protocol module.</td>
<td></td>
</tr>
</tbody>
</table>

#### Dictionary Migration and Setup

- **Dictionary Migration and Setup** ([link](https://docops.ca.com/display/IDMS19/Dictionary+Migration+and+Setup))
  - **Subschema regeneration**
    - Regenerate 10.0 subschemas in a 12.0 environment.
    - Choose from these options to generate 10.0 subschemas in 12.0 environment:
      - REGENERATE ALL SUBSCHEMAS of schema syntax
      - Generate individual subschemas
  - **DC/UCF system generation and startup**
    - Review migrated system definition and modify as necessary. Some changes are:
      - Add new SYSTEM and system generation statement parameters as appropriate.
      - Generate migrated system definition.
      - Prepare DC/UCF environment for startup.
      - The format of records in these areas has changed:
        - DDLDCLOG
        - DDLDCRUN
        - DDLDCSCR area is different.
        - RHDCPLOG replaced by ARCHIVE LOG and PRINT LOG utility statements.
        - IDMSAJNL replaced by ARCHIVE JOURNAL utility statement.
    - Include new CA IDMS 12.0 system task and program definitions.
    - Generate migrated system definition in a 12.0 environment.
    - Format DDLDCLOG and DDLDCRUN areas before using them in a 12.0 environment.
    - Format DDLDCSCR area.
    - Prepare ARCHIVE LOG and PRINT LOG utility statements and new JCL before starting up your 12.0 system.
    - Prepare ARCHIVE JOURNAL utility statement and new JCL before starting up your 12.0 system.
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Conversion activity</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Journal record formats have changed and a new journal record has been added.</td>
<td>Initialize journal files before using them in a 12.0 environment.</td>
<td>Review DCPARM macro parameters and make any necessary changes and then reassemble it.</td>
</tr>
<tr>
<td>#DCPARM macro.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Utilities (<a href="https://docops.ca.com/display/IDMS19/Utilities">https://docops.ca.com/display/IDMS19/Utilities</a>)</td>
<td>Utilities</td>
<td>Prepare utility statements for each function replaced by a utility statement that you will be using.</td>
</tr>
<tr>
<td>New utility statements replace many utility programs.</td>
<td>Backup database using BACKUP utility.</td>
<td></td>
</tr>
<tr>
<td>New execution JCL for new utility statements.</td>
<td>Create execution JCL for utility statements.</td>
<td></td>
</tr>
<tr>
<td>Modify 10.2 execution JCL for utility programs.</td>
<td>Modify 10.2 execution JCL for utility programs to identify DMCL, dictionary, and/or database name in SYSDMS parameter file for utility programs that will run in local mode. Make other JCL changes as required.</td>
<td></td>
</tr>
<tr>
<td>User Exits and Database Procedures (<a href="https://docops.ca.com/display/IDMS19/User+Exits+and+Database+Procedures">https://docops.ca.com/display/IDMS19/User+Exits+and+Database+Procedures</a>)</td>
<td>User exits and database procedures</td>
<td>Review user exits for potential changes. Modify and reassemble the code, as necessary, in a 12.0 environment.</td>
</tr>
<tr>
<td>User exits</td>
<td>Replace deleted exit with new exits.</td>
<td></td>
</tr>
<tr>
<td>The format of control blocks has changed.</td>
<td>All database procedures must be modified to use a 12.0 format subschema control block and recompiled in a 12.0 environment.</td>
<td></td>
</tr>
<tr>
<td>A user exit has been removed.</td>
<td>Four new user exits have been added.</td>
<td></td>
</tr>
<tr>
<td>Database procedures.</td>
<td>The format of the subschema control block that is passed as a parameter to a database procedure has changed.</td>
<td></td>
</tr>
<tr>
<td>Extent Areas (<a href="https://docops.ca.com/display/IDMS19/Extent+Areas">https://docops.ca.com/display/IDMS19/Extent+Areas</a>)</td>
<td>Extent areas and ASF</td>
<td>Replace queue extent areas (DDLDCQUE) with standard queue areas (DDLDCRUN).</td>
</tr>
<tr>
<td>Extent areas are no longer supported.</td>
<td>Convert ASF extent areas to standard areas.</td>
<td></td>
</tr>
<tr>
<td>Reports</td>
<td></td>
<td>Review user written reports and any standard DDR reports you have modified to determine if changes in dictionary definition require changes to reports. Make necessary changes.</td>
</tr>
<tr>
<td>None</td>
<td>User written DDR reports.</td>
<td></td>
</tr>
</tbody>
</table>
## Chapter 21

### Conversion activity

User-modified standard DDR reports.

Standard DDR reports supplied on the installation media have been updated to include changes to 12.0 dictionary definition.

### Action

**Application Environment**

- **Application environment**
  - Review all execution JCL for changes. Modify execution JCL of programs that run in local mode to use the SYSIDMS parameter file.
  - Re-link programs that access these programs.
  - Re-link programs that access IDMSUTIO for file I/O routines.
  - Review new IDMSDMLC and IDMSDMLP JCL and modify 10.2 JCL as necessary.
  - Modify 10.2 schema and create 10.2 subschema replacing SPF index with an integrated index.
  - Execute the 10.2 IDMSTBLU utility to convert the SPF index to an integrated index.
  - If necessary, remove SPF system records from application and recompile the application.
  - Review the region size of batch jobs and increase as necessary.

**Application Environment**

- CA ADS
  - CA ADS 10.21 dialogs will run, without any modifications, in a 12.0 environment.
  - Review and modify the execution JCL for CA OLQ batch jobs. For more information, see the CA OLQ User Guide.

**None**

- CA OLQ
  - Batch execution JCL

**None**

- CA CULPRIT
  - Execution JCL

**None**

- CA IDMS/DDS
  - No conversion activities are required.
  - For more information about CA IDMS/DDS Release 12.0, see the CA IDMS r12 Features Summary and the CA IDMS System Generation Guide.
Release 12.0 test environment

During the installation process, a Release 12.0 environment is created. This environment is provided to assist you in getting started using CA IDMS 12.0.

What the test environment includes

The CA IDMS Release 12.0 test environment includes:

- Dictionary environment -- Both a system and application dictionary environment are installed. A list of the installed segments and areas that comprise the system and application dictionary environments is in Dictionary Segments.

- DMCL module -- A 12.0 DMCL called R120DMCL is installed and describes the installed 12.0 runtime environment. The R120DMCL load module is in the DBALIB load library provided during the installation process. You can use this DMCL to run the dictionary migration utility and to create your 12.0 DMCL and database name table modules. A description of R120DMCL is in R120DMCL and R120DBTB Listing.

- Database name table -- A database name table that defines the databases and data dictionaries accessible by the R120DMCL is provided. It is called R120DBTB and a description of it appears in R120DMCL and R120DBTB Listing.

- DC/UCF system -- A 12.0 DC/UCF system (SYSTEM90) is installed for you to use to convert components of your 10.2 environment.

- Load libraries -- Two 12.0 load libraries exist:
  - LOADLIB -- includes CA IDMS system software load modules
  - DBALIB -- includes R120DMCL and R120DBTB load modules

Review dictionary and DMCL descriptions

You should review the list of segments and areas that comprise the installed dictionary environment and the installed DMCL to become familiar with the characteristics of your 12.0 test environment. See Dictionary Segments and R120DMCL and R120DBTB Listing.
New Installation Features

Several new features are available in this release of CA IDMS the installation procedures, and the installation section designed to simplify the installation process and assist you while maintaining the CA IDMS software.

The installation section contains the installation and maintenance procedures for the entire CA IDMS product family. Detailed information can be found in Combined Installation Manual (see page 23).

Also, there is a detailed discussion of the three types of installs available. See CA IDMS Installation Types (see page 25), for more information.

Combined Installation Manual

In the CA IDMS product family, the installation procedures for CA IDMS, CA IDTOOL, and CA IDMS. Option for CMS are merged into a single installation manual. These product lines and their respective products are listed below:

- CA IDMS
  - CA IDBFE
  - CA IDBFEASF
  - CA IDBFF
  - CA IDBFM
  - CA IDBFK
  - CA IDBGJ
  - CA IDBEJ
  - CA IDBGQ
  - CA IDBGU
  - CA IDBGX
  - CA IDBGX
- CA IDBGT
- CA IDBXS
- CA IDBGP
- CA IDTOOL
  - CA IDTO5
  - CA IDTO7
  - CA IDTDI
  - CA IDTH4
  - CA IDTH7
  - CA IDTIR
  - CA IDTO2
  - CA IDTDI
  - CA IDTDIA
  - CA IDTM3
  - CA IDTDQF
  - CA IDTDMLO
  - CA IDTO3
  - CA IDTFP
  - CA IDTF7
  - CA IDTF8
  - CA IDTF9
  - CA IDTO4
  - CA IDTGI
  - CA IDTO6
- CA IDMS. Option for CMS
Separate Installs

While the installation manual is consolidated, the installation procedures are separate. The CA IDMS. core products installation must be completed before installing the CA IDTOOL. products.

Example

Suppose you are installing CA IDBGJ., CA IDBGQ., CA IDBGY. and CA IDTDMLO..

1. Install the core products (CA IDBGJ., CA IDBGQ., CA IDBGY.), using:
   a. System Requirements (https://docops.ca.com/display/IDMS19/System+Requirements)
   b. Installing CA IDMS. (https://docops.ca.com/display/IDMS19/Installing+CA+IDMS)

2. Install the Tools products (CA IDTDMLO.), using:
   a. System Requirements (https://docops.ca.com/display/IDMS19/System+Requirements)
   b. Installing CA IDTOOL. (https://docops.ca.com/display/IDMS19/Installing+CA+IDMS+Tools)

CA IDMS Installation Types

From a base-level install tape, you can perform one of the following mutually exclusive installations:

- Complete Base Installation
- Upgrade Installation
- Add-On Installation

<table>
<thead>
<tr>
<th> </th>
<th>Complete Base Installation</th>
<th>Upgrade Installation</th>
<th>Add-On Installation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates New Install Libraries</td>
<td>Yes</td>
<td>Yes</td>
<td>Depends on product</td>
</tr>
<tr>
<td>Allocates/Formats Database Areas</td>
<td>Yes</td>
<td>No</td>
<td>Depends on product</td>
</tr>
<tr>
<td>Updates Existing Database Areas</td>
<td>No</td>
<td>Yes</td>
<td>Depends on product</td>
</tr>
<tr>
<td>Requires Previous Base Install</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Prior Install Must Be Same Genlevel</td>
<td>N/A</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Base Tape Required</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Available on Maintenance Tape</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Base tapes are not mass mailed; they are available only on individual order. Since prior-genlevel base tapes are expired, only the latest genlevel base tape can be ordered.

**Complete Base Installation**

Any site may execute a Complete Base Installation, although it is best suited for sites that have never installed CA IDMS. Release 12.0 or later. A Complete Base Installation:

- Creates new CA IDMS. software libraries
- Allocates, formats, and initializes new dictionaries
- Creates new DMCL and DB name table load modules
- Creates a new DC system

A complete installation can be performed for CA IDMS or CA IDTOOL.

**Upgrade Installation**

If you have already installed CA IDMS. Release 12.0 or later, you may choose to perform an Upgrade Installation. An Upgrade installation updates the environment with the latest software and operates with your existing dictionaries and database files. It upgrades the products currently installed.

An Upgrade installation does not allocate, format, or initialize database or dictionary files. It does update dictionaries with new entities, such as:

- Messages and Codes
- System Records
- CA IDMS Protocols
- System Classes and Attributes
- Built-in Functions
- CA IDMS Reports
- CA IDMS Load Modules

A new set of libraries is allocated and populated with the new software. New software is created in the loadlibs.

Load modules, such as the DB name table and the DMCL, that are created by punching and linking source from your dictionaries do not need to be recreated.

An Upgrade install ignores allocation and page range parameters for database files. When you specify the customized values in the Select Miscellaneous Parameters screen you do not have to modify the defaults for these items.
Add-On Installation

An Add-On installation allows you to install additional CA IDMS or CA IDTOOL products into an existing CA IDMS environment. Add-On installations must be performed on same-genlevel systems; bring your system up to the current service pack level by installing all necessary maintenance before adding the new product.

In the Product Selection screen, specify $ for the product(s) you are adding. Depending on the product, new minidisks may be required. For example, if you are installing CA IDBGT, a minidisk is required for the SQL database files.

Product Distribution and Installation Materials

This section contains information for installing and maintaining CA IDMS™ and related products in a VSE environment.

Background and preparatory information is provided first, followed by step-by-step procedures to install, customize, and maintain the CA IDMS product line. Additionally, Installing Add-On Products describes the process for installing products (add-on) to an existing CA IDMS installation. This section provides an overview to the CA IDMS installation process. Topics are as follows:

- Product distribution and installation materials
- Preinstallation considerations
- Installation methodology
- MSHP considerations

Important! Carefully review any cover letters or Product Information Bulletins (PIBs) provided in your installation package before beginning any installation activity. Cover letters and PIBs contain important information about the latest genlevel of the product line and may provide information you need to consider before beginning the installation process.

CA provides installation media containing all of the materials needed to install the product line. The JCL used to install and execute CA IDMS is included on the installation media and appears, as appropriate, throughout this document.
The machine readable program materials required for the installation are distributed on the installation media.

The installation media contains these files:

- File 1 contains CAIJMP prototype JCL for CA IDMS
- File 2 contains the input to the CA IDMS dictionary load program, IDMSDIRL. IDMSDIRL loads the SYSDIR segment during the installation process.
- File 3 contains the CA Culprit test data.
- File 4 contains SPG Text for CA IDMS SASO
- Files 9 - 21 are reserved for future expansion and are currently unused.
- File 22 and beyond contain LIBR backups for SAMPJCL, and MSHP backups for IDMSDB and IDMSDBU.

In addition to delivering CA IDMS on an ESD File for Electronic delivery, CA IDMS is also available on tape. The tape format is a 3480 cartridge. The format of the tape requires a combination of standard VSE utilities and CA's utilities to perform the installation.

### Preinstallation Considerations

**Contents**

- Installation Configuration Updates for 18.5.00 (see page 29)

CA IDMS products use several CA-CIS Services. You must install CA-CIS Services before you install CA IDMS.

⚠️ **Note**: For more information on installing CA-CIS Services, see *CA-CIS Getting Started -- for z/VSE*.

The CA-CIS Services installation tape, installation materials, and documentation are included in the CA IDMS installation package.

CA IDMS in the VSE environment requires the following CA-CIS Services:

- **System Adapter** provides operating system-dependent services such as, installing operating system interfaces, for a variety of CA products.

- **Catalog Management** is a high-performance access method developed by CA for all CA products. It supports many different record formats and can be accessed at high rates due to its sophisticated catalog index structure.
CA IDMS - 19.0

Note: If you already have System Adapter and Catalog Management installed, be sure they are at Release level 6.0 or later before installing CA IDMS.

- **CAICCI (Common Communication Interface)** allows CA IDMS products to work together across platforms and is required by CA IDMS DDS.

**Installation Configuration Updates for 18.5.00**

Several changes have been made to streamline the installation process of CA IDMS R18.5 for z/VSE. These changes are detailed below:

1. All software components are delivered to the customer. The customer chooses which products to configure and install during configuration of the installation.

2. Product keys are no longer required for product installation; product usage is managed by the new RHDCPINT feature introduced with R18.5.

3. The software libraries are restored using MSHP. The utilities CAPRDSEL and CAPTFSEL are no longer supported.

4. Sample JCL to mount the VSE installation media as a virtual tape has been included in the SAMPJCL library. The installation has been updated to take advantage of virtual tapes.

5. The #DEFSVC macro has changed. It no longer requires being submitted twice, and the parameters were simplified. It is backwards-compatible so the same R18.5 SVC can be used for older releases of CA IDMS for z/VSE.

The functions of the various install jobs have been changed and simplified. There are now more jobs but their functions are more focused. They have also been modified so that they can be re-run if there are problems during execution. A brief overview of the new job steps follows.

Note: that depending on the options chosen during configuration, not all of these steps will be generated for your installation.

Job 1 Creates the labels procedures (The functions of the old Job1 have been pulled out into the RESTLBS members included in the SAMPJCL library.)

Job 2 Assembles the SVC, and if required, an SVC exit

Job 3 Creates source and run assemblies for all modules which have site-specific requirements

Job 4 Creates or modifies the the DMCL and DBTABLE. If the DMCL or DBTABLE are modified during an add-on or upgrade, backup PHASE members will be created for each load module.

Job 5 Creates the SYSTEM dictionary as well as the SYSTEM 99 and SYSTEM 90 CV definitions, optionally runs backups
Installation Methodology

CA IDMS and CA IDMS Tools are both delivered on the product installation media. The modules for all CA IDMS and CA IDMS Tools products are installed at every customer site. Control of product usage is performed by the values implemented in the Product Intent Table module, RHDCPINT, which is assembled during product installation. It can also be assembled after the initial installation if your product mix changes or you desire to trial a new product.

Two sub-libraries are installed:

1. IDMSDB, which contains all of the product code and the mixed case version of modules which also have an uppercase version.

2. IDMSDBU, which contains the uppercase version of modules with a corresponding mixed case version.

This allows you to use mixed case modules on one CV while using the uppercase versions on a different CV simply by placing the uppercase sub-library in the SEARCH sequence ahead of the mixed case library.

The installation process is managed by the IBM Maintain System History Program (MSHP) product. Additionally, the installation methodology requires the use of the CAIIJMP program which generates customized JCL for the installation process. These topics are discussed separately as follows. MSHP concepts are discussed in more detail in MSHP Overview (https://docops.ca.com/display/IDMS19/MSHP+Overview).

MSHP

In addition to the installation materials and utilities provided by CA, the CA IDMS installation process requires the use of the IBM Maintain System History Program (MSHP) product. MSHP controls the installation and maintenance and provides a history of all installation and maintenance activities.

MSHP provides a standard installation methodology for all CA’s VSE products and ensures the stability and integrity of the execution libraries.

General MSHP installation considerations are listed as follows. MSHP is presented in more detail in MSHP Overview (https://docops.ca.com/display/IDMS19/MSHP+Overview).

- MSHP Release 2.1 or above is required for the installation and maintenance of CA IDMS.

- MSHP maintains a record of the entire system on the System History File (IJSYSHF). The System History File contains information about the components and products installed, and the maintenance activities applied to the system.

- The System History File is required for all MSHP processing and CA recommends that a new System History File, dedicated to CA IDMS, be used for this installation. CAIIJMP generates JCL to create and initialize a dedicated System History File.
Sample JCL for CA IDMS

The installation tape includes a file containing sample installation JCL. This file, SAMPJCL, contains all the JCL required to initiate and complete the installation process. All sample JCL is contained in the base CA IDMS sub-library. Each sample member has a type of SAMPJ, making it easier to locate these members.

Customizing JCL

A JCL generation program, CAIIJMP, is used to generate customized JCL for the installation process. All JCL required by the installation process, including JCL to setup and configure the MSHP environment for CA IDMS, is generated by CAIIJMP. The JCL is customized by user-defined parameter values which indicate the products you are installing and define your site environment. Instructions for using the CAIIJMP utility are provided in CA IDMS Installation Steps (https://docops.ca.com/display/IDMS19/CA+IDMS+Installation+Steps).

For a complete listing of CAIIJMP parameters, see CA IDMS CAIIJMP Parameter List.

Loading the Product Code

Contents
- Mounting the Install Media as a Virtual Tape (see page 31)

The product code is delivered in 2 sub-libraries, IDMSDB and IDMSDBU.

All object modules are delivered in addition to the pre-linked phases. All maintenance is delivered as object module replacements.

Mounting the Install Media as a Virtual Tape

If you have received the installation media as an AWS file, you can mount it as a virtual tape on your VSE machine. This process requires the IBM VSE Virtual Tape Server product. This software runs on your local computer and allows your VSE machine to remotely mount the AWS file from the local machine to use as a virtual tape.

The install JCL can be customized with commands to mount the AWS file as a virtual tape, or you can use the sample JCL in the SAMPJCL library. The member name is VTAPMNT.SAMPJ. The JCL is also displayed below:

```
* MOUNTING IDMS .AWS FILE AS VIRTUAL TAPE
* MAKE SURE THE BELOW LOC AND FILE POINT TO YOUR .AWS FILE AND COMPUTER
* CHANGE 590 TO THE VIRTUAL TAPE DEVICE # ON YOUR VSE MACHINE
/*
// VTAPE START,UNIT=590,LOC=XXX.XXX.XXX.XXX:2386,
FILE='C:\VTape\R185.AWS',READ
DVCUP 590
/*
// PAUSE IDMS 18.5 INSTALL TAPE MOUNTED, RESPOND TO RELEASE
DVCDN 590
// VTAPE STOP,UNIT=590
/*
```
To use this JCL, update the LOC to point to the IP address of your local machine. Update the FILE to point to the file location of the R185 Install AWS file. Update the appearances of ‘590’ to the virtual tape device defined to your VSE machine. This JCL will call the Tape Server on your local machine to mount the AWS file, and then PAUSE. The partition running the tape will then wait with the tape mounted until you respond to it, at which point it will bring the tape server down.

You can then access the device number that you specified as a tape device. Whenever the install has a // PAUSE for you to mount a tape, it will already be mounted.

System Generation Basics

This section is intended for the person responsible for creating and maintaining DC/UCF system definitions. Typically, this person is the system administrator.

This section:

- Provides conceptual information needed to define a CA IDMS/DC and DC/UCF system
- Explains how to define a DC/UCF system
- Serves as a reference for system generation statements and JCL used to define a DC/UCF system

This section uses the term CA IDMS to refer to any one of the following components:

- CA IDMS/DB -- The database management system
- CA IDMS/DC -- The data communications system and proprietary teleprocessing monitor
- DC/UCF -- The universal communications facility for accessing IDMS database and data communications services through another teleprocessing monitor, such as CICS
- CA IDMS DDS -- The distributed database system

The terms DB, DC, UCF, and DDS are used to identify the specific CA IDMS component only when it is important to your understanding of the product.

Using This Guide

If you are not familiar with system generation, the following information will assist you:

- Read DC/UCF System Generation (see page 33), DC/UCF Concepts (see page 41), and System Generation Compiler (see page 76) when first learning how to use the system generation compiler.

- Use the tables in System Definition (see page 127) to determine the system generation statements needed to define a DC/UCF system that meets your site-specific requirements.
Refer to the syntax, syntax rules, and examples in SYSTEM Statement (see page 131), System Generation Statements (see page 180), and Teleprocessing Network Statements (see page 280), as needed when coding system generation statements.

Refer to supplemental information provided in the sections as needed.

**DC/UCF System Generation**

System generation is the process of defining and maintaining the system that controls central version and teleprocessing operations in the CA IDMS environment.

**Central Version Operations**

Central version operations enable multiple batch and online applications to access and update a CA IDMS database at the same time. The system controls the flow of information between the database management system (DBMS) and programs requesting access to the database. The system also monitors database activity and initiates automatic recovery procedures for applications that terminate abnormally.

**Teleprocessing Operations**

Teleprocessing operations enable users to execute online, transaction-oriented applications from multiple terminals at the same time. The teleprocessing network is managed either by DC or UCF:

- **DC** is the CA teleprocessing monitor. DC manages terminal input and output operations and all devices in the teleprocessing network.

- **UCF** is an interface that enables online CA IDMS applications and software components to be executed from terminals controlled by a teleprocessing (TP) monitor other than DC. UCF supports DC, CICS, CMS, and TSO.

A system that includes DC is called a **DC system**. A system that includes UCF and not DC is called a **UCF system**. Throughout this document, references to the **DC/UCF system** apply equally to DC systems and UCF systems.

**CA IDMS DDS Network**

You can define a DC/UCF system to be part of a DDS network. DDS allows DC systems located on different CPUs to communicate with one another.

**Note:** For more information about DDS, see the *CA IDMS DDS Design and Operations section*.

**The System Generation Compiler**
The system generation compiler is used to create and maintain a DC/UCF system definition. The compiler processes source system generation statements and populates the data dictionary with the entities that make up the system definition. The compiler uses the system definition to generate the executable form of the system.

**UCF Systems Require UCF Macros**

For UCF systems, you must also assemble and link edit macros that define TP-monitor dependent procedures and the characteristics of terminals controlled by the TP monitor. The modules generated from UCF macros are stored in libraries rather than in the data dictionary.

⚠️ **Note:** For more information about defining UCF macros, see the CA IDMS System Operations section.

**The Dictionary Contains System Definitions**

A data dictionary can contain definitions of multiple DC/UCF systems. Each system is identified by a unique version number that you supply during system generation.

### Online Components and System Tasks

A DC/UCF system can include CA IDMS online components and system tasks. Some online components are supplied with DC and UCF; others must be purchased as separate products. System tasks are supplied with DC and UCF.

**Online Components**

CA supplies the following online components for a DC/UCF system:

- **CA ADS** -- A fourth-generation application development environment used to create and execute online applications that access and update CA IDMS databases
- **Automatic System Facility (ASF)** -- A facility to create and update non-sql defined data tables
- **IDD menu facility** -- A menu-driven facility to use the IDD Data Dictionary Definition Language (DDDL) compiler
- **CA IDMS Performance Monitor** -- CA IDMS Performance Monitor A product to report on DC/UCF task activity
- **CA ICMS** -- CA ICMS A product to link personal computers to the data management and storage capabilities of the mainframe computer
- **Online debugger** -- A facility to detect, trace, and eliminate errors in programs running under the control of DC or UCF
- **CA IDMS Online Command Facility (OCF)** -- CA IDMS Online Command Facility A facility used to create and maintain CA IDMS physical database definitions and SQL-defined databases

- **Online IDD** -- An online version of the IDD DDDL compiler

- **CA IDMS Mapping Facility** -- An online version of the facility used to define maps for communication between applications and terminal operators

- **CA OLQ** -- A product to retrieve and format information from CA IDMS databases through an English-like query language or SQL

- **Online schema compiler** -- An online version of the compiler used to create and update non-SQL defined schemas

- **Online subschema compiler** -- An online version of the compiler used to create and update subschemas

- **Online system generation compiler** -- An online version of the compiler used to define DC/UCF systems

- **Transfer control facility (TCF)** -- A facility to control multiple sessions of one or more CA IDMS online development tools for a single user

**System Tasks**

CA-supplied system tasks are:

- **BYE** or **B** signs the user off from the DC/UCF system and terminates the connection with the system.

- **CLIST** executes a series of task statements that are in a module and stored in the data dictionary.

- **CLOD** erases logically deleted modules from the load area (DDLDCLOD) of the data dictionary.

- **DCMT** allows DC/UCF users to monitor system runtime activities and to update system definitions while a DC/UCF system is executing.

- **DCPROFIL** allows DC/UCF users to display information about the configuration of a DC/UCF system.

- **DCUF** provides user functions that control various aspects of a DC/UCF terminal session.

- **LOCKMON** allows DC/UCF users to display information about the lock status in their DC/UCF system.

- **LOOK** allows DC/UCF users to view the contents of selected load modules.

- **OLP (online PLOG)** allows DC/UCF users to display the current contents of the DDLDLOG area online.

- **OPER** enables DC/UCF users to monitor system activity and to cancel active task threads.
QUED erases expired queues from the queue area (DDLDCRUN) of the data dictionary.

SDEL erases logically deleted users from the database security area (SYSUSER.DDLSEC).

SEND transmits a user-supplied message to other DC/UCF terminals.

SHOWMAP displays maps defined using the CA IDMS Mapping Facility.

SIGNOFF signs the user off from the DC/UCF system but maintains the connection with the system.

SIGNON or S signs the user on to the DC/UCF system.

SUSPEND terminates UCF dedicated mode but maintains the connection with the UCF back end.

The SYSGEN members that must be included in the DC/UCF system definition for the CA IDMS online components and system tasks listed previously are presented in System Programs and Tasks (see page 355).

Data Dictionary Entities

A DC/UCF system is represented in the data dictionary by entity occurrences associated with one another through set relationships. When you define a DC/UCF system, the system generation compiler stores the following entities in the system dictionary:

- System
- Program
- Task
- Teleprocessing network entities
- Destination
- Queue

System Entity

The system entity identifies the system and its operating environment and defines system resource allocations and runtime characteristics. One system entity occurrence exists for each DC/UCF system.

In the data dictionary, extensions of the system entity contain:

- Definitions of the runtime environment for CA ADS, CA IDMS Mapping Facility, and CA OLQ
- 3270-type terminal control-key assignments for online components and for line-mode I/O
- Definitions of secondary 24-bit and 31-bit storage pools
- The default usage mode for IDD DDDL compiler access to the data dictionary
- Autotask definitions

- Alternative map tables for runtime mapping operations

- Load lists

- Information on predefined run units for system services such as security enforcement program loading and message access

- Information on resources and nodes located on remote DC/UCF systems

- Definition of an SQL Cache for caching of dynamic SQL statements.

**Program Entity**

The program entity identifies each program available to the system and defines runtime characteristics for the program (for example, whether the program can be used by multiple tasks at the same time). A program can be a CA-supplied program, a user program, a CA ADS dialog, a subschema, a database procedure, a map, an edit or code table, or an access module. One program entity occurrence exists for each program associated with a system.

**Task Entity**

The task entity defines a logical unit of online processing. A task is associated with one or more programs. One task entity occurrence exists for each task associated with a system.

**Teleprocessing Network Entities**

The teleprocessing network entities define the components of a teleprocessing network:

- The **line entity** identifies and defines runtime characteristics for a group of hardware devices (physical terminals) that use the same access method (for example, VTAM). One line entity occurrence exists for each line associated with a system.

- The **physical terminal entity** defines a teleprocessing device, such as a CRT terminal, a printer, or a teletypewriter. One physical terminal entity occurrence exists for each physical terminal associated with a line.

- The **logical terminal entity** defines processing characteristics for a physical terminal. One logical terminal entity occurrence exists for each physical terminal entity associated with a line (except DDS lines).

**Destination Entity**

The destination entity represents a group of users, logical terminals, or printer terminals as a single logical destination for messages or reports. One destination entity occurrence exists for each destination associated with a system.

**Queue Entity**

The queue entity defines a disk work area used to store records for subsequent processing. One queue entity occurrence exists for each queue associated with a system.
Entities can exist independently in the data dictionary (that is, not associated with a system). You can use the system generation compiler to copy unassociated entities into a system definition.

Source and Object Records

The system generation compiler stores two types of records in the data dictionary for each entity; source and object records.

Source Records

Source records represent independent entities. Source records are associated with DC/UCF systems through object records.

Object Records

Object records represent entities that participate in DC/UCF systems. Each entity in a system is represented by an object record that is connected to both the source record for that entity and the source record for the system. The object record for a system is connected only to the system source record.

Each object record contains a copy of the information in the source record with which it is associated. Only object records are used when the DC/UCF system is executed.

Because the system generation compiler uses separate records to represent an entity and the association between the entity and a system, one entity can participate in multiple systems. Only one source record exists for the entity; however, multiple object records exist, each connected to a different system.

More Information:

For more information about the data dictionary structures created by the system generation compiler, see System Generation Data Dictionary Structure (see page 353).

DC/UCF System Reports

Contents

- Automatic Tuning (see page 40)
  - How Automatic Tuning Works (see page 41)

You can report on DC/UCF systems by using CA IDMS reports.

Categories of DC/UCF System Reports

- Entities that make up DC/UCF system definitions (represented by source records in the data dictionary).
- Entities that participate in executable DC/UCF systems (represented by object records in the data dictionary).

- Messages. Messages are added, modified, and deleted in the data dictionary through the IDD DDDL compiler.

- Device definitions. Device definitions are added to the data dictionary during CA IDMS installation.

- Map definitions. Maps are added, modified, and deleted in the data dictionary through the mapping compiler.

- The contents of the data dictionary load area (DDLDCLOD).

The following table lists the available DC/UCF system reports.

### DC/UCF System Reports

<table>
<thead>
<tr>
<th>Report Description</th>
<th>Report Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC/UCF system options (source)</td>
<td>25</td>
</tr>
<tr>
<td>DC/UCF system options (object)</td>
<td>11</td>
</tr>
<tr>
<td>DDS listing of nodes</td>
<td>43</td>
</tr>
<tr>
<td>DDS listing of defined resources</td>
<td>44</td>
</tr>
<tr>
<td>CA ADS description (source)</td>
<td>45</td>
</tr>
<tr>
<td>CA ADS description (object)</td>
<td>40</td>
</tr>
<tr>
<td>CA OLQ system description (source)</td>
<td>46</td>
</tr>
<tr>
<td>CA OLQ system description (object)</td>
<td>41</td>
</tr>
<tr>
<td>Program description (source)</td>
<td>19</td>
</tr>
<tr>
<td>Program description (object)</td>
<td>04</td>
</tr>
<tr>
<td>Task description (source)</td>
<td>20</td>
</tr>
<tr>
<td>Task description (object)</td>
<td>05</td>
</tr>
<tr>
<td>Task description within program (source)</td>
<td>21</td>
</tr>
<tr>
<td>Network description by line (source)</td>
<td>14</td>
</tr>
<tr>
<td>Network description by line (object)</td>
<td>01</td>
</tr>
<tr>
<td>Network description by physical terminal (source)</td>
<td>15</td>
</tr>
<tr>
<td>Network description by physical terminal (object)</td>
<td>02</td>
</tr>
<tr>
<td>Network description by logical terminal (source)</td>
<td>17</td>
</tr>
<tr>
<td>Network description by logical terminal (object)</td>
<td>03</td>
</tr>
<tr>
<td>Physical terminal within line (source)</td>
<td>16</td>
</tr>
</tbody>
</table>
## CA IDMS - 19.0

### Report Description

<table>
<thead>
<tr>
<th>Report Description</th>
<th>Report Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical terminal by physical terminal (source)</td>
<td>18</td>
</tr>
<tr>
<td>Destination report (source)</td>
<td>24</td>
</tr>
<tr>
<td>Destination report (object)</td>
<td>07</td>
</tr>
<tr>
<td>Queue description (source)</td>
<td>22</td>
</tr>
<tr>
<td>Queue description (object)</td>
<td>06</td>
</tr>
<tr>
<td>Queue description within task (source)</td>
<td>23</td>
</tr>
<tr>
<td>Messages in the data dictionary</td>
<td>28</td>
</tr>
<tr>
<td>Device definitions</td>
<td>29</td>
</tr>
<tr>
<td>Map record indexes</td>
<td>30</td>
</tr>
<tr>
<td>Map field indexes</td>
<td>31</td>
</tr>
<tr>
<td>List of maps by panel</td>
<td>32</td>
</tr>
<tr>
<td>List of maps</td>
<td>33</td>
</tr>
<tr>
<td>List of maps by record name</td>
<td>34</td>
</tr>
<tr>
<td>List of maps by element name</td>
<td>35</td>
</tr>
<tr>
<td>Data dictionary load area description</td>
<td>50</td>
</tr>
</tbody>
</table>

---

**Note:** For more information about samples of the DC/UCF system reports and instructions on generating the reports, see the *CA IDMS Reports section*.

---

### Automatic Tuning

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

The following system definition parameters are eligible for automatic tuning:

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

How Automatic Tuning Works

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

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

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

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

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

DC/UCF Concepts

Using the DDLDCLOG Area

LOG DATABASE Parameter

To direct the system log to the DDLDCLOG area of the data dictionary, specify LOG DATABASE in the system generation SYSTEM statement.

Benefits of Using the DDLDCLOG Area

Logging to the DDLDCLOG area provides the following benefits:

- The online version of the print log utility (online PLOG) can be used to display the current contents of the DDLDCLOG area at a terminal. Online PLOG is described in the CA IDMS System Operations section.
All or selected portions of the log can be printed while the DC/UCF system is executing.

The log can be archived while the DC/UCF system is executing.

A partially full log does not need to be archived when the system is shut down. The DC/UCF system begins writing to the unused portion of the DDLCLOG area at each system startup.

**Archiving the Log Area**

Use the ARCHIVE LOG utility statement to archive the log from the DDLCLOG area and the PRINT LOG utility statement to print all or selected portions of the log. Instructions on using the ARCHIVE LOG and PRINT LOG utility statements are provided in the *CA IDMS Utilities section*.

**Monitoring Available Space in the Log Area**

When logging to the DDLCLOG area, the DC/UCF system monitors the available space in the area. When the amount of unused space in the area halves, the system sends a message to the operator's console indicating the percentage of used space in the area. The message is issued, for example, when the log is 50% full, 75% full, 88% full, and so forth.

Users can check the percentage of used space in the log at any time by issuing a DCMT DISPLAY LOG command. DCMT commands are described in the *CA IDMS System Tasks and Operator Commands section*.

When the DDLCLOG area is 100% full, the system halts execution and waits for the log to be archived. Typically, the WTOEXIT user exit is used to automate the archiving of the system log. User exits are described in the *CA IDMS System Operations section*.

For more information, see the following topics:

- Logging Options (see page 42)
- Statistics Collection (see page 45)
- Abend Detection and Timed Functions (see page 47)
- Database Locks (see page 53)
- Resource Management (see page 54)
- External user sessions (see page 63)
- System Run Units (see page 67)
- Program Pools (see page 68)
- Storage Pools (see page 70)
- Storage Protection (see page 75)

---

**Logging Options**

Before defining a DC/UCF system, you need to be familiar with various aspects of DC/UCF operations.

A DC/UCF system records information on system activity in the system log.

**What the System Log Contains**
The log documents:

- System startups and shutdowns
- Terminal operator requests
- Abends and cancellations

Information written to the log includes DC/UCF system messages, statistics, trace information, and snap dumps.

**Note:** For more information about these topics and on starting up a DC/UCF system, monitoring system performance, and tuning the system, see the *CA IDMS System Operations section*.

**Where You Define a Log File**

Use the LOG parameter of the system generation SYSTEM statement to establish the log file assignment. You can assign the log either to the DDLDCLOG area of the data dictionary or to one or two sequential files.

**More Information:**

For more information about the LOG parameter, see SYSTEM Statement (see page 131).

**Using Sequential Files**

**LOG FILE1/FILE2 Parameter**

To direct the system log to sequential files, code the LOG FILE1/FILE2 parameter in the system generation SYSTEM statement.

**Benefits of Logging to Sequential Files**

Logging to sequential files can be useful when:

- A large log file is needed
- Access to the current contents of the log is not required while the system is executing

Sequential log files can be directed to disk or print devices. Logging to sequential disk files and logging to a print device are discussed separately.

**z/VM systems:** Logging to sequential disk files is not advisable under z/VM because disk I/O for the log is synchronous. The DC/UCF system suspends execution while awaiting completion of a synchronous I/O operation.
Logging to Sequential Disk Files

When logging to sequential disk files, you can assign the log either to a single disk file or to two alternate disk files.

Logging to a Single File

When logging to a single file, the system writes all log records to one sequential file. When the file becomes full, the system performs the following tasks:

1. Sends a message to the operator's console describing the condition
2. Continues to write to the file, overwriting records previously written to the file

The DC/UCF system writes log records to the beginning of the file at each system startup. If log records are to be saved, the appropriate operating system utility must be used to offload the file when the system is shut down.

Logging to Alternate Files

When logging to alternate files, the system writes log records to one file at a time. When the active file becomes full, the system performs the following tasks:

1. Closes the active file.
2. Sends a message to the operator's console describing the condition.
3. Opens the other file and begins logging to it. The full file can be offloaded while the system writes log records to the other file.

A full log file should be offloaded immediately. If the alternate file becomes full before the first (full) file is offloaded, the system begins logging to the first file again, overwriting records previously written to the file.

The DC/UCF system writes log records to the beginning of the first file at each system startup. If log records are to be saved, the appropriate operating system utility must be used to offload both files when the system is shut down.

Logging to a Print Device

Under z/OS

Specify a SYSOUT class for printed output (for example, SYSOUT=A) in the log-file DD statement of the DC/UCF startup JCL.

A log directed to a print device is not printed until the system is shut down. An alternate log file assignment is usually not necessary when logging to a print device under z/OS.

⚠️ Note: This release of CA IDMS supports z/OS V2R10 as well as z/OS 1.1 and higher. However, we will always refer to z/OS in this document.
Under z/VSE

Specify either SYSLST or a specific print device in the LOG parameter of the SYSTEM statement. The following considerations apply:

- **SYSLST** -- A log directed to SYSLST normally is not printed until the system is shut down. An alternate log file assignment is not necessary when logging to SYSLST.

  Note: Because SYSLST can be assigned to output media other than a print device, the DC/UCF startup JCL must specify the output medium required. Typically, SYSLST is assigned to a print device for DC/UCF logging. However, you can use SYSLST to direct the log to a tape device.

- **Specific print device** -- The log can be directed to a specific print device under either of the following circumstances:

  - The system has available one or more printers other than the printer to which SYSLST is routed.
  - The system is running in a partition controlled by a spooler such as POWER/VS, where multiple dummy printers can be assigned within the partition.

A log directed to a specific print device can be printed while the system is active. If you assign an alternate log file, the system begins logging to the alternate file when the record count for the active file is exceeded. If you do not assign an alternate log file, the system closes and then immediately reopens the log file when the record count for the file is exceeded.

Under z/VM

Specify either PRINTER or the virtual address of a print device in the log-file FILEDEF command. A log directed to a print device can be printed while the system is active by closing and reopening the file.

Statistics Collection

A DC/UCF system collects and writes five types of statistics:

- System
- Task
- Transaction
- CA ADS dialog
- DC/UCF histograms

System Statistics
System statistics record data on a system-wide basis. System statistics are always collected. They are written to the system log:

- At normal system shutdown.
- At an interval specified by the STATISTICS parameter of the system generation SYSTEM statement.
- Upon explicit request with a DCMT WRITE STATISTICS command.

**Task Statistics**

Task statistics record data on a by-task basis. Task statistics are collected only if requested by means of the STATISTICS TASK WRITE parameter of the system generation SYSTEM statement. You can specify whether separate task CPU-time statistics are to be maintained for system-mode time and user-mode time. Statistics for a task are written to the system log when the task terminates.

⚠️ **Note:** Collecting and writing tasks statistics generates a large amount of data and increases overhead.

**Transaction Statistics**

Transaction statistics record data on a logical unit of work which can span multiple tasks. Transaction statistics are collected only if task statistics collection is enabled.

The collection of transaction statistics is enabled either during system generation by means of the STATISTICS TASK TRANSACTION parameter of the SYSTEM statement or at runtime by means of the TRANSACTION parameter of the DCMT VARY STATISTICS command.

At runtime, the system collects transaction statistics only when requested to do so by a user program.

**CA ADS Dialog Statistics**

CA ADS dialog statistics record data on a by-dialog basis. Statistics can be collected for all dialogs or for selected dialogs. Dialog statistics can be collected only if transaction statistics collection is enabled.

The collection of dialog statistics is enabled either during system generation by means of the DIALOG STATISTICS parameter of the ADSO statement or at runtime by means of the DCMT VARY ADSO STATISTICS command. Individual dialogs are selected for statistics collection either during system generation by means of the ADSO DIALOG STATISTICS parameter of the PROGRAM statement or at runtime by means of the ADSO STATISTICS parameter of the DCMT VARY PROGRAM statement.

**DC/UCF Systems Maintain Histograms**

The DC/UCF system maintains additional statistics in the form of histograms. Histograms show statistical data for events in terms of frequency of occurrence within predefined value ranges. For example, one histogram may show the number of loads into the program pool of programs of up to 250 bytes, of 251- to 500-byte programs, of 501- to 750-byte programs, and so forth.
Types of Histograms Maintained by DC/UCF Systems

The system maintains the following histograms:

- **System-wide histograms** are always maintained.

- **By-task histograms** are maintained only if requested by means of the STATISTICS TASK COLLECT parameter of the system generation SYSTEM statement.

- **By-line histograms** are maintained only if requested by means of the STATISTICS LINE parameter of the system generation SYSTEM statement.

All histograms maintained by the system are written to the log whenever system statistics are written to the log.

You can run reports to retrieve statistics from the system log.

More Information:

- For more information about the SYSTEM statement, see System Statement (see page 131).

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

- For more information about the ADSO and PROGRAM statements, see ADSO Statement (see page 180) and PROGRAM Statement (see page 232).

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

- For more information about running statistics reports, see the CA IDMS Reports section.

- For more information about the statistics collected by the DC/UCF system, see the CA IDMS System Operations section.

Abend Detection and Timed Functions

The DC/UCF system uses the following mechanisms to detect abends, loops, and other abnormal processing conditions:

- Check-user tasks

- External wait time

- Inactive interval

- Resource timeout interval

- Runaway interval

- Ticker interval
Check-User Tasks

Check-user tasks are operating system subtasks attached by the DC/UCF system at startup.

How the DC/UCF System Uses Check-User Tasks

The system uses check-user tasks to detect abnormally terminated batch external request units running under the central version as follows:

1. The DC/UCF system associates a check-user task with a batch program when the first BIND request is issued by that program. An external request unit (ERUS) task is started to process the normal requests for the external request unit. A check-user task is also started when a UCFBATCH program is started. Check-user tasks are not associated with external request units from a CICS front-end or from another CA IDMS-DC/UCF front-end.

2. The check-user task attempts to enqueue a unique resource held by the batch interface module for the program. While the request unit exists, the resource is not available to the check-user task.

3. If the batch program terminates the run-unit using a FINISH or ROLLBACK or if the batch program terminates abnormally, the check-user task is then able to enqueue the resource.

4. If the batch program has explicitly ended the request unit, then the ERUS task will have already been terminated. If the batch job has ended abnormally, the ERUS task will still be active. The check-user task informs the DC/UCF system that the batch request unit has terminated.

5. The DC/UCF system aborts the ERUS task.

What Happens if No Check-User Task is Available when a Batch Request Unit is Started: The system starts the ERUS task normally. The system notes that there is no check-user task associated with the ERUS task. If a check-user task becomes available while the ERUS task is still active, then that check-user task will be associated with the ERUS task and the regular enqueue mechanism will go into effect. If the ERUS task terminates normally before any check-user task is available, then the system simply discards the notation of a potential need for a check-user task.

How You Define Check-User Tasks

The check-user mechanism is controlled by the CHKUSER TASKS parameter of the system generation SYSTEM statement. Use this parameter to specify the number of subtasks to be attached by the system. Specifying zero disables the check-user mechanism.

⚠️ Note: For more information about the SYSTEM statement, see SYSTEM Statement (see page 131).

How Many Check-User Tasks Should Be Defined
Check-user tasks are associated only with batch jobs (DML access or UCFBATCH). There is never a need for more check-user tasks than the maximum number of batch request units which will be active simultaneously. In most cases, only a few check-user tasks will be needed. A batch job may start when no check-user task is available. Suppose that job terminates abnormally before a check-user task becomes available. A check-user task will become associated with the corresponding ERUS task as soon as some other batch job terminates and its check-user task becomes available. The system will then abort the ERUS task which had been started for the program that terminated abnormally.

**External Wait Time**

External wait time is the amount of time the DC/UCF system waits for an external user session to issue a database request before assuming that it has terminated. When the external wait time elapses, the system abnormally terminates the external request unit with an error status of nn69 and initiates recovery procedures.

**How You Define External Wait Time**

Use the EXTERNAL WAIT parameter of the system generation SYSTEM statement to specify the external wait time or to disable the external-wait mechanism.

The TASK statement may be used to override the EXTERNAL WAIT time specified on the SYSTEM statement.

⚠️ **Note:** For more information about the SYSTEM statement, see SYSTEM Statement (see page 131).

The following considerations apply to specifying an external wait time:

- If the value specified is high, abends can go undetected for long periods of time, making valuable resources unavailable to other programs.
- If the value specified is low, the system may determine that a program has abended when, in fact, it has not.

**Inactive Interval**

The inactive interval is the amount of time the DC/UCF system permits a task to wait for a resource. When the inactive interval expires, the system abnormally terminates the task.

**How You Define an Inactive Interval**

Use the INACTIVE INTERVAL parameter of the system generation SYSTEM statement to specify the inactive interval or to disable the inactive-interval mechanism. You can use the STALL parameter of the DCMT VARY TIME command at runtime to override the system generation specification.
How to Override the Inactive Interval

You can override the inactive interval for individual tasks either during system generation by means of the INACTIVE INTERVAL parameter of the TASK statement or at runtime by means of the STALL parameter of the DCMT VARY TASK command.

More Information:

- For more information about the SYSTEM statement, see SYSTEM Statement (see page 131).
- For more information about the TASK statement, see TASK Statement (see page 259).
- For more information about the DCMT commands, see the CA IDMS System Tasks and Operator Commands section.

Internal Wait Time

Internal wait time is the amount of time the DC/UCF system permits an external request unit to wait for a resource (for example, a locked area or a storage allocation). When the internal wait time elapses, the system abnormally terminates the request unit with an error status of nn00 or nn69 and initiates recovery procedures.

How to Implement an Internal Wait Time

The INTERNAL WAIT parameter is accepted by the SYSGEN compiler, but it is not functional. Default wait time limits for external run-units on internal resources are controlled by the INACTIVE INTERVAL clause of the TASK parameter for the RHDCNP3S task.

You can specify wait times for individual batch or CICS run units by defining a TASK statement to the system definition of the CV. The name of the task must be the name of the batch program that first establishes the connection to the CV or the CICS task code.

Note:

- For more information about the SYSTEM statement, see SYSTEM Statement (see page 131).
- For more information about the TASK statement, see TASK Statement (see page 259).

Resource Timeout Interval

The resource timeout interval is the amount of time the DC/UCF system permits a terminal to be inactive. Terminal activity occurs when the user presses a control key (such as ENTER or PF1) that passes data to the system.
When the resource timeout interval expires, the system invokes the resource timeout program. Typically, the resource timeout program releases the resources held by the terminal and returns control to the DC/UCF system.

**How You Define the Resource Timeout Interval**

Use the RESOURCE TIMEOUT parameter of the system generation SYSTEM statement to specify the resource timeout interval and the resource timeout program or to disable the resource-timeout mechanism.

**How to Override the Resource Timeout Specification**

Use the RESOURCE parameter of the DCMT VARY TIME command at runtime to override the system generation specifications.

You can override the resource timeout specifications for individual terminals:

- During system generation by means of the RESOURCE TIMEOUT parameter of the TASK statement.
- At runtime by means of the RESOURCE INTERVAL and RESOURCE PROGRAM parameters of the DCMT VARY TASK command.
- At program execution time by means of the resource timeout parameters of the applicable DML statement. In Assembler, resource timeout specifications are overridden by means of the RESINT and RESPGM parameters of the #RETURN statement. In COBOL and PL/I, resource timeout specifications are overridden by means of the TIMEOUT parameter of the DC RETURN statement.

Task level overrides apply when the task is the last task to be executed from the terminal.

**More Information:**

- For more information about the SYSTEM statement, see *SYSTEM Statement (see page 131).*
- For more information about the TASK statement, see *TASK Statement (see page 259).*
- For more information about the DCMT commands, see the *CA IDMS System Tasks and Operator Commands section.*
- For more information about the DML statements, see the *CA IDMS DML Reference section* for the applicable language.

**Runaway Interval**

The runaway interval is the amount of time the DC/UCF system permits a task to execute without returning control to the system. A task returns control to the system for each system service call and each database operation. When the runaway interval expires, the system abnormally terminates the task.

**How You Define a Runaway Interval**
Use the RUNAWAY INTERVAL parameter of the system generation SYSTEM statement to specify the runaway interval or to disable the runaway-interval mechanism. You can use the RUNAWAY parameter of the DCMT VARY TIME command at runtime to override the system generation specification.

The runaway interval is useful for detecting and terminating looping programs. However, this mechanism does not catch all tasks that contain errors in program logic. For example, the runaway-interval mechanism will not detect a loop in which the same database operation is performed repeatedly.

Program loops that include system service calls or database operations can be detected by means of limits on task resource usage.

More Information:

- For more information about the SYSTEM statement, see SYSTEM Statement (see page 131).
- For more information about DCMT commands, see the CA IDMS System Tasks and Operator Commands section.
- For more information about limits on task resource usage, see Resource Management (see page 54)

Ticker Interval

The ticker interval determines the frequency with which the DC/UCF system checks for time-related events (such as runaway tasks).

How You Define a Ticker Interval

Use the TICKER INTERVAL parameter of the system generation SYSTEM statement to specify the ticker interval. You can use the TIMER parameter of the DCMT VARY TIME command at runtime to override the system generation specification.

To be effective, the ticker interval must be less than or equal to the lowest nonzero value specified for the following timed functions:

- Deadlock detection interval
- External wait time
- Inactive interval
- Resource timeout interval
- Runaway interval

Typically, the ticker interval for a DC/UCF system used exclusively for batch processing is larger than the ticker interval for a system used for online (or both online and batch) processing.

More Information:
Database Locks

Database locks control concurrent access to database resources: areas or individual records and rows. Locks are used to ensure that applications only see committed changes. CPU is consumed to acquire locks and storage is consumed in recording them. The greater the number of locks held concurrently, the greater the amount of storage needed.

Note: For more information about database locking, see the CA IDMS Database Administration section.

Locking Related Parameters

The following SYSTEM statement parameters are related to database locking:

- RETRIEVAL LOCK/NOLOCK controls whether record locks are acquired for areas readied in shared retrieval.
- UPDATE LOCK/NOLOCK controls whether record locks are acquired for areas readied in protected update.
- SYSLOCKS estimates of the maximum number of database locks that will be held at any time.
- LOCK LIMIT specifies the maximum number of database locks that can be concurrently held by a transaction.

Controlling Record Locking for the System

The RETRIEVAL and UPDATE parameters control whether record locks are acquired for areas readied in certain modes. By specifying NOLOCK, you reduce the number of locks acquired, but expose retrieval applications to the possibility of accessing modified records whose changes haven’t been committed.

Note: For more information about setting these options, see the CA IDMS Database Administration section.

Estimating Maximum Locks for the System
The SYSLOCKS parameter provides an estimate of the maximum number of locks held concurrently. It is used in conjunction with MAX TASKS and the number of LTERMS defined to the system to calculate the amount of operating system storage acquired at startup to hold lock-related information. If more storage is needed because of a spike in the number of concurrently held locks, additional storage will be acquired from the storage pool and freed when no longer needed. Although this process allows the system to continue to service lock requests, it costs additional CPU resources to acquire and free the storage and it could lead to fragmentation of the storage pool or short-on-storage conditions. Consequently, it is important to provide a realistic value for the SYSLOCKS parameter.

The best way to determine a value for this parameter is through trial and error. Start with a value from another system that executes a similar mix of transactions or accept the system generation default value. After executing a typical work load, use either LOCKMON or DCMT DISPLAY LOCK STATISTICS to determine whether secondary lock storage was acquired. If secondary storage was acquired frequently, increase the SYSLOCKS value. Ideally, you want to avoid any overflow storage acquisitions; however, infrequent acquisitions may be acceptable.

**Limiting Locks Acquired by Task**

You can limit the number of locks acquired by tasks on a task-by-task basis or for all tasks within a system. You use the LOCK LIMIT clause of the SYSTEM statement to limit locks set by all tasks within a system. You use the LOCK LIMIT clause of the TASK statement to override the system limit for a specific task.

**Resource Management**

The performance of a DC/UCF system depends upon several factors in your environment. One major factor is the availability of resources at runtime. There are system generation parameters that you can establish when you define a DC/UCF system that can impact resource availability and utilization at runtime. The following aspects of resource management and the system generation parameters that affect them are discussed in this section:

- How the system keeps track of the resources used by a task
- How the system detects tasks that are deadlocked
- How you can limit the resources used by a task

**Task Resource Usage**

**Control Blocks that Manage Resources**

The DC/UCF system uses control blocks to manage task usage of resources:

- The **task control element (TCE)** represents an active task. The TCE includes timer information for the task and serves as an anchor for all resources used by the task while it is running.

- The **resource control element (RCE)** represents a resource in use by one or more tasks. The RCE contains the id of the task for which the RCE was created, indicates the number of tasks currently using the resource, and identifies the resource.
- The **resource link element (RLE)** links a task to a resource. One RLE exists for each resource in use by a given task. The RLE contains a pointer to the next RLE used by the task.

The DC/UCF system allocates all TCEs, RCEs, and RLEs at startup using specifications stored by the system generation compiler. The compiler calculates the number of each type of element to be allocated, as follows:

**Calculating the Number of TCEs**

At startup, the system calculates the maximum number of tasks of all types that will be allowed to be active at any one given time. The system allocates that number of TCEs.

The number of TCEs is calculated to be equal to:

- The MAXIMUM TASKS value specified on the system generation SYSTEM statement.

- Plus the value specified by the MAXIMUM ERUS parameter of the system generation SYSTEM statement. If a value greater than 255 is entered, only 255 corresponding TCEs are included in the calculation. This prevents over-allocating memory below the 16 Mb line.

- Plus a value calculated for various system tasks. This value includes the following:
  - Plus 1 for each line included in the system definition.
  - Plus 2 for the system control tasks, MASTER and DBRC.
  - Minus 1 for each line defined as an operator’s console.
  - Plus 1 for each pre-defined run unit included in the RUN UNIT clause of the SYSTEM statement.
  - Plus 1 for the Deadlock Detection task.
  - Plus 1 for the Printer task.
  - Plus 1 (3 if running multi-tasking) if the LOG DATABASE clause of the SYSTEM statement as specified.
  - Plus 3 if running as a member of a data sharing group.
  - Plus 2 if Performance Monitor is active.
  - Plus an indeterminate number which may be reserved for other miscellaneous system functions.

The maximum number of active TCEs can be modified at run time by using the DCMT VARY ACTIVE TASKS MAX TASK command.

---

**Note:** For more information see the section "DCMT VARY ACTIVE TASK" in the *System Tasks and Operator Commands section.*
Specifying RCEs and RLEs

The numbers of **RCEs and RLEs** are specified by the RCE COUNT and RLE COUNT parameters of the system generation SYSTEM statement. You can specify explicit values in these parameters, or you can allow the compiler to calculate the values for you:

- The compiler calculates the number of **RCEs** to equal 25 times the number of TCEs.
- The compiler calculates the number of **RLEs** to equal 40 times the number of TCEs.

**At Runtime**

At startup, the system allocates the specified number of RLEs and RCEs above the 16 Mb line. When an executing CA IDMS system exhausts its primary allocation of RCEs and/or RLEs, it creates a secondary allocation in XA storage. The size of the secondary allocation is 25% of the value defined in the SYSTEM statement.

The following considerations apply to specifying the RCE COUNT and the RLE COUNT values:

- The number of RCEs and RLEs required to satisfy processing needs varies with the applications in use.
- For performance, secondary allocations of RCEs and RLEs should be avoided.
- Specifying too high a value for the RCE COUNT or RLE COUNT parameter results in wasted space.

⚠️ **Note:** For more information about the SYSTEM statement, see [SYSTEM Statement (see page 131)](#).

The following figure illustrates the control blocks used to manage task resource usage.

**Resource Management Control Blocks**
Deadlock Detection

A deadlock is an unresolved contention for the use of a resource. Resources are either DC/UCF system resources or database resources. Different control block structures are used to track contention for DC/UCF system resources and database resources. Although deadlocks are tracked using different control block structures, the same deadlock detection mechanism is used to resolve deadlocks.

You can control the allocation of some system resources used by the deadlock detection mechanism with parameters on the system generation SYSTEM statement. The discussion that follows briefly describes how the system detects and manages both DC/UCF system and database deadlocks to help you establish appropriate values for these parameters.

DC/UCF System Deadlocks

Control blocks

The DC/UCF system uses three control blocks to detect deadlocked tasks:

- Internal event control block (ECB) -- Each resource in use by a task is associated with an internal event control block (ECB). The ECB is used when another task must wait for the resource. For example, one task may be using a nonconcurrent program. When another task requests the program, the requesting task must wait for the ECB associated with the program to be posted.
 Dispatch control element (DCE) -- Each active task is associated with a dispatch control element (DCE). The DCE is used by the dispatcher to determine the status of the task (for example, waiting or ready to run). If the task is waiting for a resource, the DCE points to the ECB associated with the resource.

 Deadlock prevention element (DPE) -- When a task must wait for a resource that is in use by one or more other tasks, the system creates one deadlock prevention element (DPE) for each task already using the resource. The DPEs link the DCEs for the tasks to the ECB for the resource.

These structures are used by the deadlock detector to verify that two or more tasks are in a deadlock.

**Detecting Deadlocks for DC/UCF System Resources**

The following figure illustrates how the system uses control blocks to detect deadlocks.

Task A is using resource 1.
Task B is using resource 2.

Task A requests resource 2.
The system builds an ECB for resource 2 and a DPE linking resource 2 to task B.
Task A waits for resource 2.

Task B requests resource 1.
The system builds an ECB for resource 1 and a DPE linking resource 1 to task A.
The system checks the ECB for resource 2 and finds the DPE linking resource 2 to task B.
The deadlock detector uses these structures to determine that a deadlock has occurred.

**Specifying a DPE Count**

The DC/UCF system allocates all DPEs at startup. The number of DPEs is specified by the DPE COUNT parameter of the system generation SYSTEM statement. You can specify an explicit value in this parameter, or you can allow the system generation compiler to calculate the value for you. The compiler calculates the number of DPEs to equal:

- The number of TCEs to be allocated by the system
- Multiplied by 20
• Plus 1 for each physical terminal included in the system definition

⚠️ **Note:** For more information about the DPE COUNT parameter, see [SYSTEM Statement](#) (see page 131).

**At Runtime**

At startup, the system allocates the specified number of DPEs above the 16 Mb line. When an executing CA IDMS system exhausts its primary allocation of DPEs, it creates a secondary allocation in XA storage. The size of the secondary allocation is 25% of the value defined in the SYSTEM statement.

The following considerations apply to specifying the DPE COUNT value:

- For performance, secondary allocations of DPEs should be avoided.
- Specifying too high a value for the DPE COUNT parameter results in wasted space.

**Database Resource Deadlocks**

**Lock Entities**

The system uses three distinct entities to track contention for database resources:

- **Lock** -- Each resource in use by a task is represented by a lock. The lock identifies the resource and other information about the resource such as its usage mode. All locks for the same resource are chained together so that all tasks sharing the resource can be identified.

- **Lock wait** -- When a task requires access to a database resource that is unavailable, a lock wait is created to indicate that the task is waiting on a database resource. The lock wait contains the ECB that the task is waiting on. All lock waits for the same resource are chained together so that all tasks waiting on the resource can be identified.

- **Lock resource** -- The lock resource defines the db-key that has been locked. Each lock is chained to the resource it has been placed on. Similarly, all locks placed on the same db-key are chained off the resource that defines the db-key.

**How the System Detects a Deadlock**

The mechanism used to detect a deadlocked task is the same for both DC/UCF system resources and database resources.

Deadlock detection is carried out in these major phases:

1. **Identifying stalled tasks** -- To identify tasks that are stalled, all DCEs in the system are examined. Any DCE found stalled while waiting on a resource is entered into the deadlock detection matrix (DDM). All subsequent processing begins with the DCE address stored in the DDM table. This eliminates scanning all DCEs in the system.
2. **Identifying task dependencies** -- Next, the dependencies between the stalled tasks are identified. The deadlock detection matrix is updated to reflect these relationships. For each task, a bit in the deadlock detection matrix is enabled for the tasks it is waiting on.

3. **Identifying deadlocks** -- To determine the tasks involved in a deadlock, each pair of deadlocked tasks is examined. From this examination, one of the tasks is identified as the victim.

4. **Selecting a victim** -- The task running for the shortest period of time is chosen as the victim of the two tasks as long as:
   - The priority of the victim task is less than that of the other task
   - The victim task’s wait was not entered with COND=NONE and the other task’s wait was entered with COND=DEAD.

   The task running for the shortest period of time is chosen as the victim because most likely it has consumed fewer resources than a longer running task. As a result, less duplication of work should be required when the victim is restarted, with these exceptions:
   - If the other task is of a higher priority, implying that it is of more importance
   - If the victim task entered the deadlock with COND=NONE and the other task specified COND=DEAD. In this case, the task specifying COND=DEAD is chosen as the victim since COND=DEAD indicates that the task is designed to handle and recover from deadlock situations. This prevents an abend.

**Victim Selection User Exit**

The algorithm used to select a victim in a deadlock situation may not be optimal for your installation or applications. A user exit is provided to allow victims to be selected based upon your requirements. The exit is passed the TCE addresses of each pair of deadlocked tasks and may take one of two actions:

- Choose one of the tasks as the victim task
- Return control to the deadlock detector by requesting that the default deadlock detection logic be applied

**Note:** For more information about the Victim Selection user exit, see the *CA IDMS System Operations section*.

**Detecting Deadlocks**

The following figure illustrates how the deadlock detection mechanism detects deadlocks.

Task A is using resource 1.  
Task B is using resource 2.  
A lock is created for

![Diagram of deadlock detection](image)

Task A

Task B
resource 1 and resource 2.

Every lock is connected to the task that owns the resource.

Task A requests resource 2.

A lock wait is created to indicate that task A is waiting on resource 2.

Task A points to the lock wait for resource 2. The lock wait points to resource 2 to indicate the resource it is waiting on.

Task A waits on resource 2.

Task B requests resource 1.

A lock wait is created to indicate that task B is waiting on resource 1.

Task B waits.

When the deadlock detector is activated, it finds both task A and task B waiting on resource 2 and may then check for deadlocks.

The lock wait-to-resource chain shows the resource that the task is waiting on. The resource-to-lock-to task chain shows which tasks own a given resource. Using this information, the deadlock detector can determine tasks that are deadlocked.

Deadlock Detection Interval

You can control the frequency with which the deadlock detection mechanism searches for deadlocked tasks using the DEADLOCK DETECTION parameter of the SYSTEM statement.

The DEADLOCK DETECTION parameter allows you to specify the amount of time that elapses before the deadlock detection mechanism searches for deadlocked tasks.

You can use the DCMT VARY DEADLOCK command at runtime to override the system generation specification.

More Information:

- For more information about the SYSTEM statement, see SYSTEM Statement (see page 131).
For more information about the DCMT commands, see the CA IDMS System Tasks and Operator Commands section.

Limits on Task Resource Usage

Using system generation parameters, you can limit task usage of the following resources:

- System service calls
- Database I/O operations
- Record locks
- Storage

You can establish limits for online tasks. Since external request units are processed using online task, limits may also be specified for them.

You use the following parameters to define online task limits and control their enforcement.

Establishing Resource Limits

The CALL/DBIO/LOCK/STORAGE LIMIT FOR ONLINE TASKS parameter of the system generation SYSTEM statement defines the limit for each resource.

Overriding Limits

You can override these limits:

- During system generation by means of the LIMIT ONLINE parameter of the TASK statement
- Or at runtime by means of the DCMT VARY TASK command

Enforcing Defined Limits

The LIMITS FOR ONLINE ARE ENABLED/DISABLED/OFF parameter of the system generation SYSTEM statement controls the enforcement of defined limits. You can override the system generation specification at runtime by means of the DCMT VARY LIMITS command.

Note: Limits on task resource usage can be enforced only if TASK is specified on the Statistics parameter of the SYSTEM statement. Task statistics are described briefly under Statistics Collection (see page 45).

Limiting the Number of Application Threads
You can achieve additional control over task resource usage by limiting the number of threads that can be active concurrently for a given task. For tasks that are large consumers of system resources, you should include the MAXIMUM CONCURRENT THREADS parameter in the system generation TASK statement. Alternatively, the MAXIMUM CONCURRENT parameter of the DCMT VARY TASK command can be used at runtime to limit the number of concurrently active task threads.

More Information:

- For more information about the SYSTEM statement, see SYSTEM Statement (see page 131).
- For more information about the TASK statement, see TASK Statement (see page 259).
- For more information about DCMT commands, see the CA IDMS System Tasks and Operator Commands section.

External user sessions

An external user session is a logical connection between a DC/UCF system and an application executing outside that DC/UCF system. An external user session is initiated when the application initiates the first request for services within the DC/UCF system and is terminated when the last service initiated by the application is terminated. When an external user session is initiated the user is effectively signed on to the DC/UCF system and is signed off when the user session terminates.

Types of External User Sessions

There are different types of external user sessions:

- External request units (ERUs)
- UCF sessions
- DDS sessions
- LU 6.2 sessions
- TCP/IP sessions

More Information:

For more information about system generation considerations for LU 6.2 sessions, see Teleprocessing Network Statements (see page 280) and SNA and LU 6.2 Considerations (see page 399).
External Request Units

An external request unit is an online task initiated within a DC/UCF system to service database requests from an external user session.

An external request unit is initiated when a request for database services originates from:

- A batch application
- Another DC/UCF system (using either DC-to-DC communications through an SVC or DDS)
- Another TP monitor such as CICS.
- An application executing on the PC that uses CAICCI to communicate, such as CA Visual Express

All concurrent database transactions initiated by one user session and processed on the same back-end DC/UCF system are managed as a single external request unit. All database requests issued as part of the external request unit are serviced by the same back-end task.

Specifying Runtime Characteristics of ERUs

Specify resource limits, dispatching priority and other runtime characteristics of external request units using the system generation TASK statement.

By default all external request units are serviced by a task named RHDCNP3S. The attributes defined for the RHDCNP3S task are used for all external request units processed under that task code. Alternate task codes can be defined for external request units so that different runtime characteristics are used.

Note: For more information about specifying runtime characteristics of external request units, see the CA IDMS System Operations section.

System Generation Requirements

The following system generation parameters are related to external request unit processing:

- Task and program definitions for RHDCNP3S
- The MAXIMUM ERUS parameter of the SYSTEM statement determines the maximum number of concurrent non-DDS external request units that can be processed by a DC/UCF system
- CHKUSER TASKS parameter of the SYSTEM statement controls a mechanism for detecting abnormally terminated batch applications in z/OS and z/VSE
- EXTERNAL WAIT parameter on the SYSTEM and TASK statements determines the amount of time the system waits between external requests before assuming that the external user session has terminated
Additional parameters to define a DDS environment

**DDS Sessions**

DDS allows DC/UCF systems that are located on different CPUs to communicate with each other. Different access methods can be used for the communication: TCP/IP, VTAM, or CAICCI (Common Communications Interface).

To define the remote DC/UCF systems that will be accessed by another DC/UCF system using DDS, take the following steps:

1. Define the DC/UCF systems and the databases to be accessed using the system generation RESOURCE TABLE and NODE statements.
2. Depending on the access method to use, define the following entities:
   - CCI: define a CCI line
   - TCP/IP: define a DDSTCPIP type PTERM in a SOCKET line
   - VTAM: define a VTAM type PTERM in a DDS line
3. Define a task and program for RHDCNP3S.

You can specify execution characteristics for DDS sessions using the RHDCNP3S task, and you can define additional tasks for specific applications or front-end systems.

**Note:**

- For more information about defining and maintaining DDS, see the *CA IDMS DDS Design and Operations section*.
- For more information about specifying runtime characteristics of external request units, see the *CA IDMS System Operations section*.

**UCF Sessions**

UCF processing allows DC/UCF online tasks to be executed on behalf of terminals controlled by another terminal monitor front-end system. The front-end system forwards terminal input to the UCF back-end for processing and writes the output screen images received from the back-end to the terminal.

**System Generation Requirements**

The following SYSGEN parameters are related to UCF processing:
External Request Elements

External request elements (EREs) are control blocks used with SVC communications to maintain the link between an external user session and the DC/UCF system servicing its requests.

Each external request unit takes up one ERE. The ERE is dedicated to that request unit from the time the first service request is issued until all services for the user session are terminated.

EREs in a UCF Session

A UCF session requires the use of an ERE only while screen images are being transferred between the front-end and back-end systems. At other times, such as while the back-end task is executing or while the front-end system is waiting on a terminal interrupt, the UCF session does not hold an ERE. Therefore several UCF sessions can share a relatively small number of EREs.

Number of Available EREs

The number of EREs available is specified using the MAXIMUM ERUS parameter of the SYSTEM statement. When determining the value for this parameter considers the following:

- The value should be at least as high as the number of concurrent external sessions you wish to support.
- Allow for UCF requirements by increasing the number by an amount that is between 10% and 20% of the number of UCF terminals defined in the system generation.
- If all EREs are in use, a program or UCF front-end attempting to initiate a session receives a 1473 error status.

TCP/IP Sessions

TCP/IP allows communication between a DC/UCF system and any other system that supports TCP/IP. TCP/IP is supported in the following environments:
To allow TCP/IP access within DC/UCF, define the TCP/IP SYSGEN entity using the TCP/IP statement. If generic listening or DDS using TCP/IP are used, the LISTENER and DDSTCPIP type PTERMS must be defined in a SOCKET line. Multiple lines of type SOCKET may be defined in a DC/UCF system, and starting with r17, all may be active at the same time.

More Information:

- For more information about the TCP/IP statement, see TCP/IP Statement (see page 272).
- For more information about the SOCKET LINE statement, see SOCKET (see page 325).
- For more information about the implementation of TCP/IP support within DC/UCF, see the CA IDMS Callable Services section.

System Run Units

System run units are initiated by CA IDMS software components to satisfy requests for services. For example, loading a module from a load area or extracting message text from the message area.

Predefined System Run Units

A predefined system run unit is one which is defined in the DC/UCF system generation and is started during the DC/UCF system initialization.

System run units can be predefined for the following:

- **Load area (DDLDCLOD) processing.** Use the RUNUNITS FOR LOADER parameter of the system generation SYSTEM statement to predefine load-area run units for the default dictionary. Use the system generation RUNUNITS statement to predefine load-area run units for an alternate dictionary.

  Note: For more information about the RUNUNITS statement, see RUNUNITS Statement (see page 251).

- **Message area (DDLDCMSG) processing** by means of the RUNUNITS FOR MSGDICT parameter of the system generation SYSTEM statement.

- **Queue area (DDLDCRUN) processing** by means of the RUNUNITS FOR QUEUE parameter of the system generation SYSTEM statement.

- **Signon processing in the user catalog** by means of the RUNUNITS FOR SIGNON parameter of the system generation SYSTEM statement.
• Destination processing by means of the RUNUNITS FOR DEST parameter of the system generation SYSTEM statement.

• Security enforcement for system resources by means of the RUNUNITS FOR SECURITY parameter of the system generation SYSTEM statement.

• Security enforcement for SQL by means of the SQL SECURITY parameter of the system generation RUNUNITS statement.

• Load area (DDLCATLOD) processing for SQL by means of the SQL LOADER parameter of the system generation RUNUNITS statement.

How Predefined Run Units are Used

Predefined run units are initiated at startup and exist for the duration of system execution. At startup, the DC/UCF system allocates storage for each predefined run unit. The system maintains this storage until system shutdown.

Overflow Run Units

At runtime, if the number of predefined run units is insufficient to satisfy program processing requirements, the system initiates additional run units as needed. These additional run units are referred to as overflow run units. The system terminates overflow run units when they are no longer needed, freeing their storage.

An insufficient number of predefined run units can result in an excessive amount of journal activity. Only one BIND/FINISH sequence is issued for each predefined run unit, regardless of the number of requests the run unit is used to satisfy. Overflow run units, initiated on an as-needed basis, exist only to process individual requests. One BIND/FINISH sequence is issued per request. For example, if a single predefined run unit initiated at startup processes 100 signon requests, only one BIND/FINISH sequence is issued for those requests. If overflow run units process 100 signon requests, 100 BIND/FINISH sequences are issued.

Program Pools

A program pool is an area of storage in the DC/UCF region/partition that is used for loading both resident and nonresident programs. Typically, program pools hold:

• CA ADS applications
• CA ADS dialogs
• Database procedures
• Edit and code tables
• Maps
• Programs invoked by online tasks (both CA-supplied and user-written)
• Subschemas
Access modules

Relational command modules

A DC/UCF system can include up to four program pools. Each pool must be a different type. The following table describes each type of program pool.

**Note:** The page size of the reentrant pools (512 bytes) can be smaller than the page size of the standard program pools (4K bytes) because the storage protect key of reentrant pages never needs to change. User programs are not permitted to modify reentrant pages. The smaller page size reduces the amount of unusable space in the reentrant pools.

<table>
<thead>
<tr>
<th>Type of Pool</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>24-bit program pool</td>
<td>Mandatory for each DC/UCF system</td>
</tr>
<tr>
<td></td>
<td>Page size 4K bytes (allocated in 4K increments)</td>
</tr>
<tr>
<td></td>
<td>Defined by the <a href="#">PROGRAM POOL</a> parameter of the system generation SYSTEM statement</td>
</tr>
<tr>
<td>24-bit reentrant pool</td>
<td>Optional; for reentrant programs (for example, CA ADS dialogs, subschemas, and programs defined as reentrant in a system generation PROGRAM statement)</td>
</tr>
<tr>
<td></td>
<td>Page size 512 bytes (allocated in 512-byte increments)</td>
</tr>
<tr>
<td></td>
<td>Defined by the <a href="#">REENTRANT POOL</a> parameter of the system generation SYSTEM statement</td>
</tr>
<tr>
<td>31-bit program pool</td>
<td>Optional (only for operating systems that support 31-bit addressing); for programs with a residency mode (RMODE) of ANY</td>
</tr>
<tr>
<td></td>
<td>Page size 4K bytes (allocated in 4K increments)</td>
</tr>
<tr>
<td></td>
<td>Defined by the <a href="#">XA PROGRAM POOL</a> parameter of the system generation SYSTEM statement</td>
</tr>
<tr>
<td>31-bit reentrant pool</td>
<td>Optional (only for operating systems that support 31-bit addressing); for reentrant programs with a residency mode (RMODE) of ANY</td>
</tr>
<tr>
<td></td>
<td>Page size 512 bytes (allocated in 512-byte increments)</td>
</tr>
<tr>
<td></td>
<td>Defined by the <a href="#">XA REENTRANT POOL</a> parameter of the system generation SYSTEM statement</td>
</tr>
</tbody>
</table>

**Specify at Least One 24-bit Program Pool**

Every DC/UCF system must include the 24-bit program pool. The minimum size for this pool is 4K. The size that you specify in the system definition depends on the size and number of nonreentrant and quasireentrant programs that may be executed concurrently.
Note: Program pool space is allocated in 4K increments. For example, a 3K module is allocated 4K; a 5K module is allocated 8K.

Typically, a DC/UCF system includes the 24-bit reentrant pool. When you are first defining the system, the recommended value for the size of this pool is 1200K if ASF is used under the system and 500K if ASF is not used.

Resident programs, including nucleus and driver modules, are loaded into the appropriate program pools at DC/UCF startup. The system automatically increases the size of each pool by the amount of space occupied by the resident programs in the pool.

Under those operating systems that support 31-bit addressing, the system attempts to load all modules created by CA IDMS compilers and all programs with an RMODE of ANY into the applicable 31-bit pool, if defined. If the 31-bit pool does not contain adequate space, the system loads the program into the corresponding 24-bit pool.

Note: For more information about program loading under systems supporting the XA feature, see the CA IDMS System Operations section.

Storage Pools

A storage pool is an area of storage in the DC/UCF region/partition from which the system and programs executing under the system acquire space for work areas and control blocks. Typically, storage pools hold:

- Program variable storage
- COBOL working storage
- Variable portions of CA ADS dialogs
- Variable portions of subschema tables and access modules
- Currency blocks
- Database lock tables
- Buffer pools (if IDMS storage is specified on the DMCL BUFFER statement or through the DCMT VARY BUFFER command)
- Secondary allocations of null PDEs
- User trace buffers

256 Storage Pools is the Maximum
A DC/UCF system can include up to 256 storage pools. Each pool is identified by a unique number. The following table describes the storage pools that you can define.

<table>
<thead>
<tr>
<th>Pool Id</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage pool 0</td>
<td>Primary storage pool&lt;br&gt;Mandatory for each DC/UCF system&lt;br&gt;Located in 24-bit address space&lt;br&gt;Defined by the STORAGE POOL parameter of the system generation SYSTEM statement</td>
</tr>
<tr>
<td>Storage pools 1 through 127</td>
<td>Secondary storage pools&lt;br&gt;Optional&lt;br&gt;Located in 24-bit address space&lt;br&gt;Defined by the system generation STORAGE POOL statement</td>
</tr>
<tr>
<td>Storage pools 128 through 254</td>
<td>Secondary storage pools&lt;br&gt;Optional (only for operating systems that support 31-bit addressing)&lt;br&gt;Located in 31-bit address space&lt;br&gt;Defined by the system generation XA STORAGE POOL statement</td>
</tr>
<tr>
<td>Storage pool 255</td>
<td>Secondary storage pool&lt;br&gt;Located in 31-bit address space&lt;br&gt;Defined by the XA STORAGE POOL parameter of the SYSTEM statement</td>
</tr>
</tbody>
</table>

### Storage Pools 0 and 255

Every DC/UCF system must include storage pool 0. It should be large enough to contain the amount of variable non-XA storage required by both CA-supplied and user-written programs (for example, CA ADS dialogs, system tasks, and CA OLQ).

Storage pool 0 also serves as an "overflow" pool for those storage types for which no non-XA secondary storage pool exists. Overflow occurs when there is no secondary storage pool that has enough space available for a storage request.

Every DC/UCF system must include storage pool 255. It should be large enough to contain the amount of variable XA storage needed by the DC/UCF system for internal house-keeping storage (for example, communication buffers, report printing and scratch management). When starting a DC/UCF system, the generated pool 255 is dynamically extended with a secondary pool 255 that contains internal house-keeping storage needed during startup.

### Displaying the Actual Size of Storage Pools
You can use the DCMT DISPLAY STORAGE command to determine the actual size of the pools.

**More Information:**

- For more information about DCMT commands, see the *CA IDMS System Tasks and Operator Commands section*.
- For more information about runtime allocation of XA storage pools, see the *CA IDMS System Operations section*.

## Increasing the Efficiency of Storage Pool Usage

Here are some suggestions to make storage pool usage more efficient.

### Define a Storage Cushion

Define a storage cushion for one or more storage pools. A storage cushion is space reserved in a storage pool for use by tasks that are already executing. Use of a storage cushion helps prevent space in the storage pool from being exhausted.

When the amount of unused space in storage pool 0 is smaller than the storage cushion, the DC/UCF system is said to be short on storage. The system starts no new tasks and uses the cushion to satisfy storage requests from tasks that are already executing. Storage cushions in secondary storage pools are also used only when the system is short on storage.

When space in storage pool 0 is freed and the amount of unused space exceeds the storage cushion, the system accepts requests to start new tasks. Additionally, the system no longer uses the cushion to satisfy storage requests from tasks that are already executing.

Use the CUSHION parameter of the system generation SYSTEM statement to specify the size of the cushion for storage pool 0. For secondary storage pools, use the CUSHION parameter of the corresponding system generation STORAGE POOL or XA STORAGE POOL statement.

**More Information:**

- For more information about the SYSTEM statement, see [SYSTEM Statement](#).
- For more information about the STORAGE POOL and XA STORAGE POOL statements, see [STORAGE POOL Statement](#) and [XA STORAGE POOL Statement](#).

### Size of the Cushion for Storage Pool 0

Typically, the size of the cushion for storage pool 0 is calculated to equal:

- The value specified by the STORAGE POOL parameter of the system generation SYSTEM statement
- Plus 3
- Divided by three times the value specified by the MAXIMUM TASKS parameter of the system generation SYSTEM statement
CA ADS Runtime System

Direct the CA ADS runtime system to calculate the amount of storage required for record buffer blocks (RBBs) instead of using the size specified in the system generation ADSO statement. Calculated storage reduces the amount of wasted space in the storage pool but increases CPU usage.

To take full advantage of calculated storage, include as many records as possible either in the application global records or in the first dialog in the application thread. Either of these strategies minimizes the allocation and release of RBBs during links to lower-level dialogs and programs.

Use the STORAGE MODE IS CALCULATED parameter of the system generation ADSO statement to request calculated storage.

⚠️ **Note:** For more information about the ADSO statement, see ADSO Statement (see page 180).

Define a Fast Mode Threshold

The fast mode threshold is the point at which the DC/UCF system writes CA ADS record buffer blocks (RBBs) and statistics control blocks to the scratch area (DDLDCSCR) across a pseudo-converse. If the total size of the RBBs and statistics control blocks in all storage pools exceeds the fast mode threshold, the system writes the RBBs and statistics control blocks to scratch.

The fast mode threshold applies only when the system generation ADSO statement specifies RESOURCES ARE FIXED. If the ADSO statement specifies RESOURCES ARE RELOCATABLE, RBBs and statistics control blocks are always written to scratch across a pseudo-converse.

Use the FAST MODE THRESHOLD parameter of the system generation ADSO statement to specify a fast mode threshold.

⚠️ **Note:** For more information about the ADSO statement, see ADSO Statement (see page 180).

Relocatable Threshold

Define a relocatable threshold for one or more storage pools. The relocatable threshold is the point at which the DC/UCF system writes relocatable storage to the scratch area (DDLDCSCR) across a pseudo-converse.

Relocating storage makes more efficient use of the storage pool but increases I/O to the scratch area. You should define a threshold such that the system relocates storage only when the storage pool is heavily used.
Use the RELOCATABLE THRESHOLD parameter of the system generation SYSTEM statement to specify the relocatable threshold for storage pool 0. For secondary storage pools, use the RELOCATABLE THRESHOLD parameter of the corresponding system generation STORAGE POOL or XA STORAGE POOL statement.

More Information:

- For more information about the SYSTEM statement, see SYSTEM Statement (see page 131).
- For more information about the STORAGE POOL and XA STORAGE POOL statements, see STORAGE POOL Statement (see page 255) and XA STORAGE POOL Statement (see page 277).

Define Secondary Storage Pools

Define secondary storage pools from which user programs can obtain storage space. Defining secondary storage pools allows the isolation of user-program storage from system storage and helps prevent the system from stalling when user programs require large amounts of storage space.

Operating systems that support 31-bit addressing: If secondary storage pools are defined, at least one of the pools should be in 24-bit memory (that is, non-XA).

You can assign one or more types of storage to each secondary storage pool that you define. To minimize the number of control tables that the system must search for a particular type of storage, however, you should assign each type of storage to no more than one storage pool.

Use the STORAGE POOL and XA STORAGE POOL statements to define secondary storage pools.

⚠️ **Note:** For more information about these statements, see STORAGE POOL Statement (see page 255) and XA STORAGE POOL Statement (see page 277).

Segregating User and System Storage

As mentioned in the previous section, segregating user and system storage helps to prevent the system from stalling when a user program uses large amounts of storage. This storage segregation is also a prerequisite for enabling the High Performance Storage Protect feature.

⚠️ **Note:** For more information about the High Performance Storage Protect feature, see Storage Protection (see page 75).

System generation permits defining four types of user-oriented storage: user, user-kept, shared, and shared-kept, as well as two types of system storage: database and terminal.
The storage pools must be defined in such a manner that all forms of user-oriented storage are segregated from the system storage. In other words, define both an XA and a non-XA storage pool for user storage types. Storage types: user, user-kept, shared, and shared-kept, can be together, but they must be defined to secondary storage pools and must be isolated from any storage pools that contain database or terminal type storage.

To segregate user and system storage, perform the following steps:

1. In SYSGEN, define at least one storage pool in the range 128 to 254 to support types user, user-kept, shared, and shared-kept. In addition, no pool can be defined to support type ALL (the default) or a mixture of these types and system (database or terminal) storage.

2. To remove user storage types from pool 0, you must SYSGEN at least one pool in the range of 1 to 127 that supports types user, user-kept, shared, and shared-kept. In addition, no pool can be defined to support type ALL (the default) or a mixture of these types and system (database or terminal) storage.

---

**Storage Protection**

DC/UCF storage protection protects pages in the system region/partition from being overwritten.

Generally, storage protection is used for developing, debugging, and testing new programs to ensure that the programs do not overwrite the storage allocated to production programs. To preserve operating system integrity and protect CA IDMS from user written code, storage protect should always be used. A special form of storage protect is available for the production system which provides negligible processing overhead yet protects CA IDMS and the operating system from user written code.

**How to Implement Storage Protection**

Storage protection is implemented through storage protect keys provided in the hardware. The DC/UCF system uses a primary protect key and an alternate protect key:

- When the system nucleus has control of the DC/UCF region/partition, all storage pages in the region/partition are set to the primary protect key. The nucleus executes with the program status word (PSW) set to the primary protect key which allows the system nucleus to modify any page in the region/partition.

- When a program is executing in user mode, all non-fully reentrant pages used by the program are set to the alternate protect key. All other pages remain set to the primary protect key. The program executes with the PSW set to the alternate protect key which allows the program to modify only those pages set to the alternate key. If the program attempts to modify a page set to the primary protect key, the program is terminated abnormally.

- When key 9, the default, is used as the alternate protect key, all user storage and non-reentrant programs are swapped into key 9 at startup, and only the PSW key is changed during program execution. Thereby, providing negligible CPU overhead compared to running without storage
protection, while still protecting both the CA IDMS system and the operating system from user written code. The non-reentrant programs and the user storage are not protected from each other. This is explicitly intended for the high performance requirements of the production CV. Program development should not be done with key 9 as the alternate protect key.

System Generation Parameters

Use the following system generation parameters to control the use of storage protection:

- STORAGE KEY parameter of the SYSTEM statement -- defines the value of the alternate protect key.
- PROTECT/NOPROTECT parameter of the SYSTEM statement -- enables or disables the use of storage protection.
- PROTECT/NOPROTECT parameter of the PROGRAM statement -- specifies whether the program being defined runs with the alternate protect key when the SYSTEM statement specifies PROTECT.

More Information:

- For more information about the SYSTEM statement, see SYSTEM Statement (see page 131).
- For more information about the PROGRAM statement, see PROGRAM Statement (see page 232).

Enabling High Performance Storage Protect

This feature is explicitly intended for servicing the high performance requirements of the production CV. Assuming that a storage protected system has already been successfully created, the following steps can be used to enable this feature:

1. On the existing system, display all the storage pools (DCMT DISPLAY ALL STORAGE POOLS), taking note of what pools support any type of user storage, that is, user, user-kept, shared, shared-kept, or ALL.

2. Follow the instructions under Segregating User and System Storage (see page 74).

3. In SYSGEN, on the system statement, specify STORAGE KEY IS 9.

4. Generate and start the system. If the storage pool definitions have not been properly set, message DC004001 HPSPO HAS BEEN DISABLED DUE TO INCORRECT STORAGE POOL DEFINITIONS is issued at startup.

System Generation Compiler

The system generation compiler maintains DC/UCF systems according to statements that you submit either online or through a batch job. You can direct the compiler to perform three types of activities:

- Store and update a system definition
- Validate a system definition
• Generate the executable form of a system

**Store and Update a System Definition**

For each type of entity in a system definition, corresponding statements allow you to add, modify, delete, display, and punch occurrences of the entity. For example, to add a task to a system definition, you submit an ADD TASK statement. Additional statements allow you to add, modify, delete, display, and punch the portions of a system definition that are stored as an extension of the system entity.

When you submit a statement to add a system to the data dictionary, the system generation compiler creates a system source record. When you submit a statement to add an entity other than system (a component entity), the compiler creates a source record for the entity, if one does not already exist, and an object record connecting the entity to the system you are defining.

**Validate a System Definition**

You submit a VALIDATE statement to request that the system generation compiler cross-check the relationships between entities in the system definition. For example, in response to the VALIDATE statement, the compiler verifies that each physical terminal is associated with a single logical terminal.

Explicitly requesting validation of the system definition is an optional activity. The system generation compiler automatically cross-checks relationships before generating the executable form of a system.

**Note:** For more information and syntax, see VALIDATE Statement (see page 114).

For more information, see the following topics:

- Compiler Activities (see page 77)
- System Definition Backup (see page 79)
- Currency (see page 80)
- Coding Considerations (see page 81)
- Adding, Modifying, and Deleting Entities (see page 87)
- Displaying and Punching Entities (see page 92)
- Compiler-Directive Statements (see page 97)
- Compiler Messages (see page 116)
- Execution Modes (see page 117)
- Sublibrary ID Syntax (see page 122)
- Sublibrary ID Parameters (see page 122)
- Lib-name Syntax (see page 124)
- Lib-name Parameters (see page 124)

**Compiler Activities**

**Generate the Executable Form of a System**

You submit a GENERATE statement to request that the system generation compiler perform the following tasks:
- Validate the system definition
- Create the object record for the system entity
- Copy information from new and/or updated source records into the corresponding object records
- Calculate defaults and modify values where applicable
- Complete the connections between:
  - Programs and tasks
  - Tasks and queues
  - Lines and physical terminals
  - Physical terminals and logical terminals
  - Destinations and logical terminals
- Flag all entities in the system as executable

Additional steps are required to create the routine used to start up a DC/UCF system.

**System Generation Compiler Activities**

The following figure illustrates the activities of the system generation compiler.
System Definition Backup

Before modifying an executable system, you should duplicate the system definition. The duplicate definition provides a backup system that can be generated and executed if the modified system fails to start up.

To duplicate a system definition:

1. Display the system definition online as syntax
2. Modify the system version number
3. Resubmit the system definition to the compiler

⚠️ **Note:** For more information about using the online system generation compiler to modify entities, see Online System Generation (see page 117).
Currency

The system generation compiler uses currency to determine:

- The system with which a component entity is associated
- The line with which a physical terminal is associated
- The physical terminal with which a logical terminal is associated

Accordingly, the compiler maintains currency for systems, lines, and physical terminals.

System Currency

System currency is established by an ADD, MODIFY, DISPLAY, or PUNCH SYSTEM statement. Until you submit one of these statements to the compiler, no system is current.

System currency is the only way to associate an entity with a system. You must establish system currency before you can add, modify, delete, display, or punch component entities.

System currency changes each time the compiler processes an ADD, MODIFY, DISPLAY, or PUNCH SYSTEM statement for a different system. No system is current after the compiler processes a DELETE SYSTEM statement.

All statements that you submit to add or delete component entities apply to the current system. For example, to add program MYPROG to system 12, you must first make system 12 current:

MODIFY SYSTEM 12.
ADD PROGRAM MYPROG.

Line Currency

Line currency is established by an ADD, MODIFY, DISPLAY, or PUNCH LINE statement. Until you submit one of these statements to the compiler, no line is current.

Line currency changes each time the compiler processes an ADD, MODIFY, DISPLAY, or PUNCH LINE statement for a different line. No line is current after the compiler processes a DELETE LINE statement.

Statements that you submit to add or delete a physical terminal apply to the current line. For example, to add physical terminal PT001 to line L001, you must first make line L001 current:

MODIFY LINE L001.
ADD PTERM PT001.

You can override line currency for a specific physical terminal by including the IN LINE parameter in the PTERM statement.

Physical Terminal Currency

Physical terminal currency is established by an ADD, MODIFY, DISPLAY, or PUNCH PTERM statement. Until you submit one of these statements to the compiler, no physical terminal is current.
Physical terminal currency changes each time the compiler processes an ADD, MODIFY, DISPLAY, or PUNCH PTERM statement for a different physical terminal. No physical terminal is current after the compiler processes a DELETE PTERM statement.

Statements that you submit to add or delete a logical terminal apply to the current physical terminal. For example, to associate logical terminal LT001 with physical terminal PT001, you must first make physical terminal PT001 current:

```
MODIFY PTERM PT001.
ADD LTERM LT001.
```

You can override physical terminal currency for a specific logical terminal by including the PTERM parameter in the LTERM statement.

### Coding Considerations

When coding input for the system generation compiler, you should consider the following:

- Statement format
- Delimiters
- Quotation marks
- Input lines
- Comments
- Carriage control statements

### Statement Format

System generation statements that reference entities have five parts.

1. **Verb**

The verb specifies the action to be taken by the system generation compiler:

- The **action verbs** are:
  - ADD establishes a new entity occurrence in the data dictionary.
  - MODIFY updates an existing entity occurrence.
  - DELETE removes an existing entity occurrence from the data dictionary.

- The **display verbs** are:
  - DISPLAY displays one or more existing entity occurrence definitions.
PUNCH displays one or more existing entity occurrence definitions and writes the definitions either to the SYSPCH file or to a data dictionary module.

If you omit the verb from a system generation statement, the compiler supplies a verb.

ADD is the default verb when either of the following conditions applies:
- The specified entity occurrence does not exist in the data dictionary.
- The specified entity occurrence exists in the data dictionary but does not participate in the current system.

MODIFY is the default verb when the specified entity occurrence exists in the data dictionary and already participates in the current system.

The action verb DELETE and the display verbs DISPLAY and PUNCH must always be specified explicitly.

2. Entity Type Name

The entity type name identifies the type of entity to which the statement applies. The entity type name can also identify extended characteristics of the system entity.

Valid Entity Type Names

The following are the valid entity type names.

<table>
<thead>
<tr>
<th>ADSO</th>
<th>LTERM</th>
<th>QUEUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOTASK</td>
<td>MAPTYPE</td>
<td>RESOURCE TABLE</td>
</tr>
<tr>
<td>DESTINATION</td>
<td>NODE</td>
<td>RUNUNITS</td>
</tr>
<tr>
<td>IDD</td>
<td>OLM</td>
<td>SQL CACHE</td>
</tr>
<tr>
<td>KEYS</td>
<td>OLQ</td>
<td>STORAGE POOL</td>
</tr>
<tr>
<td>LINE</td>
<td>PROGRAM</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>LOADLIST</td>
<td>PTERM</td>
<td>TASK</td>
</tr>
<tr>
<td>TCP/IP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XA STORAGE POOL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Entity Occurrence Name

The entity occurrence name identifies a specific occurrence of the named entity type. For example, in the following statement, UPDCUST is the entity occurrence name:

ADD PROGRAM UPDCUST.

Occurrences of the following entities can be further identified by a version number: destinations, lines, logical terminals, physical terminals, programs, queues, and tasks.

The following entity type names are not accompanied by an entity occurrence name: ADSO, IDD, OLM, OLQ, RESOURCE TABLE, RUNUNITS, TCP/IP, and SQL CACHE.

4. Parameters
The **parameters** either define characteristics of the entity occurrence or specify display options for the entity occurrence definition. For example, in the following statement, the LANGUAGE IS COBOL parameter defines the program's source language:

```plaintext
ADD PROGRAM ADDCUST LANGUAGE IS COBOL.
```

Parameters can be either optional or required, as indicated in the syntax, and can be coded in any order. If a statement contains multiple occurrences of the same parameter, the system generation compiler uses the last occurrence coded.

### 5. End of Statement

The period terminates a system generation statement and is required in all statements. The period can directly follow the last word in the statement, can be separated from the last word by blanks, or can be coded on a separate line.

⚠️ **Note:** You can establish recognition of the semicolon as an alternate end of statement character if you specify the SEMICOLON ALTERNATE END OF SENTENCE IS ON clause on the SET OPTIONS statement or if this option has been activated by the IDD DDDL compiler as a default for the dictionary. For more information about the SET OPTIONS statement, see [SET OPTIONS Statement](#).

The following example illustrates the parts of a typical system generation statement:

```plaintext
ADD TASK TXT007 INVOKES PROGRAM PXT007 INTERNAL.
```

**IDMSDB Statement Format**

### Using IS and '=' Interchangeably

You can use the equal sign (=) in place of the word 'IS' in any system generation statement.

The equal sign (=) is not shown in the system generation syntax diagrams in this document.

### Delimiters

You must use one or more blanks as delimiters between words in a system generation statement. Commas, semicolons, and colons, which are treated as blanks by the compiler, can be used as additional delimiters to enhance readability.
Note: You cannot use the semicolon as a delimiter if it has been established as an alternate statement terminator.

Quotation Marks

Use Quotation Marks with User-Supplied Names

You must use quotation marks to enclose user-supplied names containing one or more embedded blanks or punctuation marks (that is, apostrophes, colons, commas, parentheses, periods, quotation marks, and semicolons). If the closing quotation mark is omitted, the compiler interprets the name to include everything on the line up to the end of the input column range.

Single Quotation Mark is Installation Default

The single quotation mark (’) is the default established during installation. This default can be changed to the double quotation mark (") with the IDD DDDL compiler. You can use the SET OPTIONS statement to establish either quotation mark as the standard for the current session.

Note: For more information about the SET OPTIONS statement, see SET OPTIONS Statement (see page 98).

If the current standard quotation mark is embedded in a user-supplied name, the quotation mark must be coded as two consecutive quotation marks. For example, the name MARY’S PROGRAM must be coded as 'MARY''S PROGRAM' if the single quotation mark (’) is the current standard. If the double quotation mark (") is the current standard, MARY’S PROGRAM must be coded as "MARY'S PROGRAM".

Non-quoted Letters Converted to Upper Case

All input submitted to the system generation compiler that is not in quotes is converted to upper case. Input submitted not in quotes in lower case is automatically converted to upper case.

Input Lines

80-Character Card-Image Format

You submit system generation statements to the compiler in an 80-character card-image format. The range of columns within which you can code the statements is determined by the SET OPTIONS statement. The maximum range is 1 through 80. The default range is 1 through 79 for online input and 1 through 72 for batch input.

Note: For more information about the SET OPTIONS statement, see SET OPTIONS Statement (see page 98).
Code Statements on Single or Multiple Lines

Each system generation statement can be coded on a single line or on multiple lines within the input column range. Additionally, multiple statements can be coded on a single line.

The following considerations apply to continuing a statement on another line:

- No continuation character is required.
- Words cannot be split across lines.

Note: The compiler views a user-supplied name within quotation marks as a single word.

Using Blank Lines

To enhance readability, you can include blank lines at any point among the input statements. Blank lines are ignored by the compiler.

The following examples show two acceptable input formats for the same system generation statement:

Example 1
MODIFY SYSTEM 30 PROGRAM POOL IS 300.

Example 2
MODIFY SYSTEM 30
  PROGRAM POOL IS 300
  .

Comments

How to Indicate a Comment

You can include comments among the statements submitted to the system generation compiler. Use any of the following symbols on an input line preceding the comment text to indicate that the remainder of the line is a comment:

- Asterisk (*) (Must be coded in position 1 to be accepted as a comment)
- Asterisk and plus sign (*+)
- Two dashes (--)
When the system generation compiler encounters the initial symbol, it disregards the entire line of input.

Comment Lines Redisplayed with ECHO or LIST Option

Although comment lines are not interpreted as system generation statements, they are redisplayed along with other input lines if the ECHO or LIST option of the SET OPTIONS statement is in effect. Note, however, that comment lines beginning in positions 1 and 2 of an input line with an asterisk and a plus sign (\*+) or two dashes (\--) are not echoed.

![Note: For more information about the SET OPTIONS statement, see SET OPTIONS Statement (see page 98).]

Carriage Control Statements

SKIP and EJECT Statements

The SKIP and EJECT carriage control statements are used to format the CA IDMS/DC SYSGEN Compiler Activity List. SKIP directs the compiler to insert one, two, or three blank lines in the listed output. EJECT directs the compiler to continue printing the listed output on a new page. Typically, EJECT is used to format the CA IDMS/DC SYSGEN Compiler Activity List by entity type. These statements are not printed, nor do they affect the operation of the system generation compiler.

When used online, both SKIP and EJECT cause the compiler to insert a single blank line in the displayed output.

Code SKIP and EJECT Between Statements

The SKIP and EJECT statements can be coded between any two system generation statements, but each must be the only statement in the input line. These statements do not include a terminating period.

Syntax for the SKIP and EJECT statements is shown below. The keyword SKIP and the specified integer cannot be separated. For example, SKIP1 is valid; SKIP 1 is invalid.

SKIP Statement Syntax

```
SKIP statement syntax
  
  SKIP E 1 2 3
```

SKIP Statement Parameters

1

Directs the compiler to insert one blank line in the CA IDMS/DC SYSGEN Compiler Activity List.
2
Directs the compiler to insert two blank lines in the CA IDMS/DC SYSGEN Compiler Activity List.

3
Directs the compiler to insert three blank lines in the CA IDMS/DC SYSGEN Compiler Activity List.

**EJECT Statement Syntax**

```
EJECT statement syntax
```

**EJECT Statement Parameters**

**EJECT**

Directs the compiler to continue printing the listed output on a new page.

**Adding, Modifying, and Deleting Entities**

The action verbs ADD, MODIFY, and DELETE direct the system generation compiler to store, update, or remove system definitions in the data dictionary.

⚠️ **Note:** For more information about specific ADD/MODIFY/DELETE syntax for each entity type, see [SYSTEM Statement](#) (see page 131), [System Generation Statements](#) (see page 180), and [Teleprocessing Network Statements](#) (see page 280).

**ADD Verb**

The **ADD** verb adds system and component entity definitions to the data dictionary and associates component entities with the current system. ADD also establishes currency for systems, lines, and physical terminals.

When adding an entity, you can explicitly specify characteristics of the entity and/or you can accept one or more default characteristics. Component entities added to an existing system are not reflected in the executable form of the system until the next successful GENERATE statement for that system. You must generate the system definition before bringing it up.

**How the Compiler Handles ADD Requests**

The system generation compiler responds to ADD requests, as follows:

- **In response to an ADD SYSTEM request**, the compiler creates a system source record. The system object record is not created until the compiler successfully processes a GENERATE statement.
In response to an ADD request for a component entity that does not exist in the data dictionary, the compiler creates both a source record for the entity and an object record that associates the entity with the current system.

In response to an ADD request for a component entity that already exists in the data dictionary but that is not associated with the current system, the compiler creates an object record that associates the entity with the current system. If you explicitly specify entity characteristics in the ADD request, the system generation compiler also modifies the source record for the entity. Characteristics not explicitly overridden remain unchanged.

Object Records are Flagged for Update

Object records created in response to ADD requests are flagged for update. These records do not yet contain a copy of the information in the corresponding source records. In response to a GENERATE statement, the system generation compiler copies the information about the entity from the source record to the object record and removes the update flag.

Using DEFAULT IS ON Option

The system generation compiler accepts the ADD verb in reference to an existing system or to a component entity already associated with the current system only if you have specified DEFAULT IS ON in the SET OPTIONS statement. When DEFAULT IS ON has been specified, the system generation compiler changes the ADD verb to MODIFY for existing entities.

Note: For more information and syntax for the SET OPTIONS statement, see SET OPTIONS Statement (see page 98).

The following table illustrates system generation compiler actions in response to ADD requests.

<table>
<thead>
<tr>
<th>Statements Compiler Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADD SYSTEM 20.</strong></td>
</tr>
<tr>
<td><strong>ADD TASK X.</strong></td>
</tr>
<tr>
<td>(system 20 current)</td>
</tr>
<tr>
<td><strong>ADD SYSTEM 30.</strong></td>
</tr>
<tr>
<td><strong>ADD TASK X.</strong></td>
</tr>
<tr>
<td>(system 30 current)</td>
</tr>
<tr>
<td><strong>GENERATE.</strong></td>
</tr>
</tbody>
</table>
MODIFY Verb

The **MODIFY** verb updates existing system and component entity definitions in the data dictionary. MODIFY also establishes currency for systems, lines, and physical terminals.

When modifying an entity, you must explicitly specify all characteristics that are to be changed. Characteristics not explicitly modified remain unchanged. Changes made to a system and its component entities are not reflected in the executable form of the system until you successfully generate that system with its changed characteristics. When any of the SYSGEN entities are modified, a GENERATE statement must be executed before recycling CV. If the SYSGEN is not successfully generated, the modified entities may not be recognized the next time CA IDMS CV is started.

The system generation compiler responds to MODIFY requests, as follows:

- **In response to a MODIFY SYSTEM request**, the compiler updates the system source record with any explicitly specified characteristics. The next successful GENERATE statement for the system causes the compiler to copy information from the system source record into the system object record.

- **In response to a MODIFY statement for a component entity**, the compiler updates the source record for the entity with any explicitly specified characteristics. The compiler also places update flags on all object records associated with the entity. A successful GENERATE statement for a system that includes the entity causes the compiler to copy information from the modified source record into the object record that connects the entity to the system.

What the Compiler does on a MODIFY

The following table illustrates system generation compiler actions in response to MODIFY requests.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Compiler Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODIFY SYSTEM 20</td>
<td>Sets system 20 current and modifies source record for system 20.</td>
</tr>
<tr>
<td>MAX TASKS IS 15</td>
<td>Modifies source record for task X and flags all task X object records for update (U).</td>
</tr>
<tr>
<td>DISABLED.</td>
<td></td>
</tr>
</tbody>
</table>
DELETE Verb

The **DELETE** verb disassociates component entities from systems and deletes system and component entities from the data dictionary. Deletions of component entities from a system are not reflected in the executable form of the system until the next successful GENERATE statement for that system.

**How the Compiler Handles a DELETE Action**

The system generation compiler responds to DELETE requests, as follows:

- **In response to a DELETE SYSTEM request**, the compiler deletes both the source and object records for the system and all object records that connect component entities to the system. To prevent the inadvertent deletion of entity definitions, the compiler does not delete source records for component entities when deleting a system. A GENERATE statement is not required after a DELETE SYSTEM statement.

  To delete source records for component entities that no longer participate in any systems (as a result of DELETE SYSTEM requests), you can use the IDD DDDL compiler.

  ![Note: For more information about using the DDDL compiler to delete entities, see the CA IDMS IDD DDDL Reference section.](image)

- **In response to a DELETE request for a component entity**, the compiler places a delete flag on the object record that connects the entity to the current system. A successful GENERATE statement for the system causes the compiler to delete the flagged object record. If no other object records exist for the entity (that is, if the entity is not associated with any other system), the compiler also deletes the source record for the entity.
**Note**: The system generation compiler deletes source records only for entities created through the system generation compiler. You must use the IDD DDDL compiler to delete entities created through the DDDL compiler.

The following table illustrates system generation compiler actions in response to DELETE requests.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Compiler Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status before deletions.</td>
<td></td>
</tr>
<tr>
<td><strong>DELETE SYSTEM 20.</strong></td>
<td>Deletes source and object records for system 20 and object records connecting tasks X, Y, and Z to system 20.</td>
</tr>
<tr>
<td><strong>MODIFY SYSTEM 30.</strong></td>
<td>Sets system 30 current. Places delete flag (D) on object record connecting task X to system 30.</td>
</tr>
<tr>
<td><strong>DELETE TASK X.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>MODIFY SYSTEM 40.</strong></td>
<td>Sets system 40 current. Places delete flag (D) on object record connecting task X to system 40.</td>
</tr>
<tr>
<td><strong>DELETE TASK X.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DELETE TASK Y.</strong></td>
<td>Places delete flag (D) on object record connecting task Y to system 40.</td>
</tr>
<tr>
<td>(system 40 current)</td>
<td></td>
</tr>
<tr>
<td><strong>GENERATE.</strong></td>
<td>Deletes object records connecting tasks X and Y to system 40. Copies information from source record for system 40 to object record for system 40.</td>
</tr>
<tr>
<td>(system 40 current)</td>
<td></td>
</tr>
<tr>
<td><strong>MODIFY SYSTEM 30.</strong></td>
<td>Sets system 30 current. Deletes object record connecting task X to system 30 and source record for task X. Copies information from source record for system 30 to object record for system 30.</td>
</tr>
<tr>
<td><strong>GENERATE.</strong></td>
<td></td>
</tr>
</tbody>
</table>
Displaying and Punching Entities

The display verbs DISPLAY and PUNCH list system and component entity definitions from the dictionary. DISPLAY and PUNCH also establish currency for system, lines, and physical terminals. For component entity definitions other than system, a successful DISPLAY or PUNCH request reflects only the presence of the definition in the dictionary, NOT an assurance that the definition is associated with the current system. To confirm this association, use a DISPLAY ALL entity-type request, which lists the component entities associated with the current system.

How the Compiler Handles the DISPLAY Verb

DISPLAY causes the compiler to list the requested entity definition either online at the terminal or in batch mode in the CA IDMS/DC SYSGEN Compiler Activity List (described under Batch System Generation (see page 119)).

When DISPLAY is used online, you can edit the displayed definition and resubmit it as input to the compiler.

⚠️ Note: For more information about resubmitting displayed output, see Online System Generation (see page 117).

How the Compiler Handles the PUNCH Verb

PUNCH causes the compiler to display the requested entity definition and to write the definition either to the SYSPCH file or to a data dictionary module.

A definition written to the SYSPCH file can be edited and resubmitted as input to the compiler in batch mode. A module containing punched definitions can be included in other system definitions.

⚠️ Note: For more information about including modules in system definitions, see INCLUDE Statement (see page 114).

Default Actions with the DISPLAY and PUNCH

DISPLAY and PUNCH statement defaults are determined by the SET OPTIONS statement. You can override the SET OPTIONS options in the individual DISPLAY and PUNCH statements that you submit to the compiler.

⚠️ Note: For more information about the SET OPTIONS statement, see SET OPTIONS Statement (see page 98).
DISPLAY Syntax

```
DISplay  
PUNch  
  entity-type  entity-occurrence-name  
  ALL  entity-type  
    Version is  
    version-number  
    WITH  
    ALSo WITH  
    WITHOut  
      DETails  
      HISTORY  
      ALL  
      NONE  
    AS  
    COMments  SYNtax  
    TO  
    module-specification  

DISPLAY Parameters

entity-type

Identifies the type of entity to be displayed or punched.

Entity-type must be a valid system generation entity type.

⚠️ Note: For more information about valid entity types, see Statement Format (see page 81).

entity-occurrence-name

Specifies a single entity occurrence to be displayed or punched.

Entity-occurrence-name must be the name of an existing occurrence of the specified entity type. The named occurrence does not have to be associated with the current system, line, or physical terminal.

- **Version is version-number**
  Qualifies the named entity occurrence with a version number.
  Version-number must be an integer in the range 1 through 9,999. The default is 1.

ALL entity-type

Directs the system generation compiler to display or punch either all entities associated with a specific system or all occurrences of the named entity type associated with the current system. Entity-type-name must be a valid system generation entity type.
Note: For more information about valid entity types, see Statement Format (see page 81).

WITH

Directs the system generation compiler to override the options specified by the DISPLAY WITH and DISPLAY ALSO WITH parameters of the SET OPTIONS statement and to include only the indicated information in the displayed or punched definition. The override applies to the current DISPLAY/PUNCH request only.

ALSo WITh

Directs the system generation compiler to include the indicated information in the displayed or punched definition in addition to the information included because of the options specified by the DISPLAY WITH and DISPLAY ALSO WITH parameters of the SET OPTIONS statement. The additional information is included for the current DISPLAY/PUNCH request only.

WITHOut

Directs the system generation compiler to exclude the indicated information from the information included in the displayed or punched definition because of the options specified by the DISPLAY WITH and DISPLAY ALSO WITH parameters of the SET OPTIONS statement. The information is excluded for the current DISPLAY/PUNCH request only.

DETa ils

Includes or excludes the entity-specific characteristics that make up the entity definition.

HIStory

Includes or excludes the dates when the entity definition was created and last updated and the identification of the users who created and last updated the definition (given by the PREPARED BY and REVISED BY parameters). History is maintained only for the following entity types: DEFAULT PROGRAM, DESTINATION, LINE, LOGICAL TERMINAL, PHYSICAL TERMINAL, PROGRAM, QUEUE, SYSTEM, and TASK.

ALL

Includes or excludes both the entity history and the details of the entity definition.

NONe

Includes only the name and version number (if applicable) of the entity occurrence in the displayed or punched definition.

NONE is meaningful only with WITH.

AS

Specifies whether the displayed or punched definition appears as comments or syntax.
• **COMments**
  Specifies that each line of the entity definition is displayed or punched as a comment. Comment lines generated by the compiler are indicated by an asterisk and a plus sign (*+) in the first two columns and are ignored when resubmitted to the compiler.

• **SYNTAX**
  Specifies that the displayed or punched definition appears in syntax format. Definitions that are displayed (online only) or punched (online and batch) as syntax can be edited and resubmitted to the compiler as input.

**TO**
For PUNCH statements only, specifies the destination of the punched output.

• **module-specification**
  Causes the compiler to write punched output to the specified module. Expanded syntax for module-specification is presented next.

• **SYSpch**
  Causes the compiler to write punched output to the SYSPCH file.

**MODULE Syntax**

```
 MODULE module-name

 Version is version-number
   HIGhest
   LOWest

 LANguage is language
```

**MODULE Parameters**

**MODule module-name**

Identifies a module previously defined in the data dictionary.

**Note:** For information on defining modules, refer to the *CA IDMS IDD DDDL Reference section*.

**Version is**

Qualifies the named module with version number.
- **version-number**
  Specifies an explicit version number.
  *Version-number* must be an integer in the range 1 through 9,999. The default is 1.

- **HIGhest**
  Specifies the highest existing version of the named module.

- **LOWest**
  Specifies the lowest existing version of the named module.

**LANguage is language**

Qualifies the named module with a language.

**Example MODULE**

The following screens show system generation compiler responses to three sample DISPLAY/PUNCH requests. The default options specified by the DISPLAY parameter of the SET OPTIONS statement are WITH ALL and AS COMMENTS.

```
SYSGEN 17.0    NO ERRORS    DICT=CORPDICT    1/17    cv-name
DISPLAY ALL SYSTEM 9 WITH NONE.
  *+  ADD SYSTEM 9
  *+  ADD ADSO
  *+  ADD OLM
  *+  ADD PROGRAM RHDCBYE
  *+  ADD PROGRAM PXT001
  *+  ADD PROGRAM PXT002
  *+  ADD TASK TXT001
  *+  ADD TASK TXT002

SYSGEN 17.0    NO ERRORS    DICT=CORPDICT    1/15    cv-name
PUNCH ALL PROGRAMS WITHOUT DETAILS.
  *+  ADD PROGRAM RHDCBYE
  *+  DATE CREATED IS 09/29/07
  *+  DATE LAST UPDATED IS 09/30/07
  *+  PREPARED BY ABC
  *+  REVISED BY ABC
  *+  ADD PROGRAM PXT001
  *+  DATE CREATED IS 09/30/07
  *+  PREPARED BY ABC
  *+  ADD PROGRAM PXT002
  *+  DATE CREATED IS 09/30/07
  *+  PREPARED BY ABC

SYSGEN 17.0    NO ERRORS    DICT=CORPDICT    1/26    cv-name
DISPLAY TASK TXT001 AS SYNTAX.
  *+  ADD TASK TXT001
  *+  DATE CREATED IS 09/30/07
```
Compiler-Directive Statements

Compiler-directive statements direct the execution of the system generation compiler:

- **SIGNON** initiates an online session or batch run of the system generation compiler.
- **SIGNOFF** terminates an online session or batch run of the system generation compiler.
- **SET OPTIONS** establishes default processing options for the current online session or batch run of the system generation compiler.
- **DISPLAY/PUNCH OPTIONS** lists either the default processing options established in the data dictionary or the processing options in effect for the current online session or batch run of the system generation compiler.
- **VALIDATE** directs the system generation compiler to cross-check the relationships among all components of the current system.
- **GENERATE** instructs the system generation compiler to validate (see above) the current system and, if validation is successful, to produce the executable form of the system.
- **COPY** copies all or portions of an existing system definition into a new or other existing system definition.
- **INCLUDE** submits as input to the system generation compiler statements stored as a module in the data dictionary.

Each compiler-directive statement is described below.
SET OPTIONS Statement

The SET OPTIONS statement establishes processing options for the current system generation compiler session. The processing options determine:

- Disposition of ADD statements issued for existing entities
- Format of and information included in displayed or punched entity definitions
- Format of compiler input and output
- End-of-file indicator
- Quotation mark

Processing options remain in effect until explicitly changed by a SET OPTIONS statement. You can override the processing options that affect displayed and punched definitions in the individual entity statements. You cannot override any other compiler processing options on an individual basis.

Processing Options You Can Change

The following table lists the installation defaults for processing options that can be changed by the system generation compiler SET OPTIONS statement.

<table>
<thead>
<tr>
<th>Option</th>
<th>Batch Default</th>
<th>Online Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT IS</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>DISPLAY WITH</td>
<td>ALL</td>
<td>ALL</td>
</tr>
<tr>
<td>DISPLAY AS</td>
<td>COMMENTS</td>
<td>COMMENTS</td>
</tr>
<tr>
<td>ECHO/NO ECHO</td>
<td>ECHO</td>
<td>ECHO</td>
</tr>
<tr>
<td>EOF IS</td>
<td>/*</td>
<td>/*</td>
</tr>
<tr>
<td>HEADER/NO HEADER</td>
<td>HEADER</td>
<td>NO HEADER</td>
</tr>
<tr>
<td>INPUT COLUMNS ARE</td>
<td>1 THRU 72</td>
<td>1 THRU 79</td>
</tr>
<tr>
<td>LIST/NO LIST</td>
<td>LIST</td>
<td>LIST</td>
</tr>
<tr>
<td>OUTPUT LINE SIZE IS</td>
<td>132</td>
<td>80</td>
</tr>
<tr>
<td>PROMPT/NO PROMPT</td>
<td>NO PROMPT</td>
<td>Line mode device:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PROMPT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3270: NO PROMPT</td>
</tr>
<tr>
<td>PUNCH TO</td>
<td>SYSPCH</td>
<td>SYSPCH</td>
</tr>
<tr>
<td>SEMICOLON ALTERNATE END OF SENTENCE</td>
<td>Default established through DDDL compiler</td>
<td>Default established through DDDL compiler</td>
</tr>
<tr>
<td>QUOTE IS</td>
<td>'</td>
<td>'</td>
</tr>
</tbody>
</table>
SET OPTIONS Statement Syntax

DEFault is

ON
OFF

DISplay

WITH
ALSo WITH
WITHOUT
DETails
HISTORY
ALL
NONE

EOF is

/*
eof-indicator'
OFF

HEAder

NO HEAder

INPut columns are

start-column-number
THRu
end-column-number

LISt

NO LISt

OUTput line size is

80
132

PROmpt

NO PROmpt

PUNch TO

module-specification

SYSpch

QUOte is

'
" 

SEMicolon alternate end of sentence is

ON
OFF

SET OPTIONS Statement Parameters

DEFault is

Determines whether the system generation compiler accepts or rejects ADD statements that reference existing entity occurrences.

• ON

Directs the compiler to accept ADD statements that reference existing entity occurrences. The compiler interprets the statements as MODIFY statements and issues the message ADD CHANGED TO MODIFY.
OFF
Directs the compiler to reject ADD statements that reference existing entity occurrences. The compiler issues an error message and terminates processing of the statement.

DISPLAY
Determines the format of and information included in displayed or punched entity definitions.

WITH
Directs the compiler to override all previously established display options and to include only the indicated information in displayed or punched definitions.

ALSO WITH
Directs the compiler to include the indicated information in displayed or punched definitions in addition to any information included because of previously established display options.

WITHOUT
Directs the compiler to exclude the indicated information from information included in displayed or punched definitions because of previously established display options.

DETAILS
Includes or excludes the entity-specific characteristics that make up the entity definition.

HISTORY
Includes or excludes the dates when the entity definition was created and last updated and the identification of the users who created and last updated the definition (given by the PREPARED BY and REVISED BY parameters). History is maintained only for the following entity types: DEFAULT PROGRAM, DESTINATION, LINE, LOGICAL TERMINAL, PHYSICAL TERMINAL, PROGRAM, QUEUE, SYSTEM, TASK, and USER.

ALL
Includes or excludes both the entity history and the details of the entity definition.

NONE
Includes only the name and version number (if applicable) of the entity occurrence in the displayed or punched definition. NONE is meaningful only with WITH.

AS
Specifies whether displayed or punched definitions appear as comments or syntax.

COMMENTS
Specifies that each line of the entity definition is displayed or punched as a comment. Comment lines generated by the compiler are indicated by an asterisk and a plus sign (*+) in the first two columns and are ignored when resubmitted to the compiler.

SYNTAX
Specifies that the displayed or punched definition appears in syntax format. Definitions that are displayed (online only) or punched (online and batch) as syntax can be edited and resubmitted to the compiler as input.

ECHO
Specifies that the system generation compiler is to redisplay each input line (online) or to print each input line in the CA IDMS/DC SYSGEN Compiler Activity List (in batch mode). Comment lines that begin with an asterisk and a space (*) in positions 1 and 2 are echoed. Comment lines that begin with an asterisk and a plus sign (**) or two dashes (--) in positions 1 and 2 of an input line are not echoed.

**ECHO** has the same effect as the LIST option. The ECHO option is ignored when the NO LIST option is in effect.

**NO ECHO**

Directs the system generation compiler not to redisplay input lines, whether errors are present.

NO ECHO is useful when statements are submitted one line at a time (for example, from a hard-copy terminal or from a TSO or CMS terminal during a batch run of the compiler).

When NO ECHO is in effect, the LIST/NO LIST option is ignored.

⚠️ **Note:** The NO ECHO option suppresses execution of the SKIP and EJECT carriage control statements. For more information about carriage control statements, see Carriage Control Statements (see page 86).

**EOF**

Specifies the logical end-of-file indicator.

- **eof-indicator**
  Specifies the logical end-of-file indicator as a two-character value enclosed in quotation marks. The installation default is /* for both online and batch processing.
  When the end-of-file indicator is coded in the first two input columns of the input range, the compiler processes only the statements that precede the indicator. The compiler does not recognize statements that follow the end-of-file indicator.

- **OFF**
  Specifies that no end-of-file indicator is active.

**HEAder**

For batch mode only, specifies that header lines that identify the system generation compiler are to be printed on each page of the CA IDMS/DC SYSGEN Compiler Activity List.

For sample pages of the CA IDMS/DC SYSGEN Compiler Activity List with header lines, see Batch System Generation (see page 119).

**NO HEAder**

For batch mode only, specifies that no header lines are to be printed on the pages of the CA IDMS/DC SYSGEN Compiler Activity List.

**INPut columns are start-column-number THRu end-column-number**
Specifies the column range for statements input to the system generation compiler:

- **Start-column-number** must be an integer in the range 1 through 70
- **End-column** must be an integer in the range 10 through 80

**Start-column-number** and **end-column-number** must be at least 10 columns apart.

**LST**

Specifies that the system generation compiler is to redisplay each input line (online) or to print each input line in the CA IDMS/DC SYSGEN Compiler Activity List (in batch mode). Comment lines that begin with an asterisk and a plus sign (\*+) are not echoed.

LIST has the same effect as the ECHO option. The LIST option is ignored when the NO ECHO option is in effect.

**NO LIST**

Specifies that the system generation compiler is to redisplay only input lines that contain errors.

The NO LIST option is ignored when the NO ECHO option is in effect. When NO LIST is in effect, the ECHO option is ignored.

⚠️ **Note:** The NO LIST option suppresses execution of the SKIP and EJECT carriage control statements. For more information about carriage control statements, see *Carriage Control Statements* (see page 86).

**OUTput line size is**

Specifies the width of header and message lines in the terminal display or batch activity listing.

- **80**
  Header and message lines contain 80 characters.
  When the output line size is 80, line numbers are not displayed.

- **132**
  Header and message lines contain 132 characters.

**PROMpt**

Directs the system generation compiler to issue the prompt ENTER when it is ready to accept input.

PROMPT is useful when statements are submitted from a dial-up device or from a TSO or CMS terminal during a batch run of the compiler.

**NO PROMpt**

Directs the system generation compiler not to issue prompts for input.
PUNch to

For PUNCH statements only, specifies the destination of the punched output.

- **module-specification**
  Causes the compiler to write punched output to the specified module.
  Expanded syntax for *module-specification* is presented as part of the DISPLAY and PUNCH syntax described in *Displaying and Punching Entities* (see page 92).

- **SYSpch**
  Causes the compiler to write punched output to the SYSPCH file.

QUOte is

Specifies the quotation mark for the current compiler session.

- `'`
  The quotation mark is the single quotation mark (').

- `"`
  The quotation mark is the double quotation mark (").

SEMicolon alternate end of sentence is

Indicates whether the semicolon is to be recognized as an alternative end-of-statement character. If an option is not selected, the default is established by the IDD DDDL SEMICOLON ALTERNATE END OF SENTENCE IS statement.

- **ON**
  Specifies that both semicolons and periods are to be recognized as end-of-statement characters.

- **OFF**
  Specifies that the semicolon is *not* recognized as an alternative end-of-statement character.

Example SET OPTIONS Statement

The following sample SET OPTIONS statement instructs the compiler to list DISPLAY/PUNCH output in syntax format and to redisplay each line of input:

```
SET OPTIONS FOR SESSION
  DISPLAY AS SYNTAX
  ECHO.
```

The sample SET OPTIONS statement shown next, establishes // as the end-of-file indicator for the current compiler session and specifies that subsequent input must be coded within columns 5 and 70:

```
SET OPTIONS FOR SESSION
  EOF IS '//'
  INPUT COLUMNS ARE 5 THRU 70.
```
DISPLAY/PUNCH OPTIONS Statement

The DISPLAY/PUNCH OPTIONS statement allows you to view either signon information and default processing options for the current session or the default processing options established in the data dictionary.

DISPLAY/PUNCH OPTIONS Statement Syntax

```
DISPLAY OPTIONS for SESSION

WITH DETAILS
ALSO WITH DETAILS

AS SYNTAX

TO module-specification
```

DISPLAY/PUNCH OPTIONS Statement Parameters

for

Specifies whether the system generation compiler is to display the processing options for the current session or for the dictionary.

- **SESSION**
  
  Directs the compiler to list the processing options in effect for the current session. SESSION is the default when you omit the FOR parameter from the DISPLAY/PUNCH OPTIONS statement.

- **DICTIONARY**

  Directs the compiler to list the processing options established for the data dictionary. The date the dictionary was created and the date of the most recent update to the dictionary are included in the listing.

WITH DETAILS

Directs the compiler to display all processing options.

ALSO WITH DETAILS

Directs the compiler to display all processing options.

You must specify either WITH DETAILS or ALSO WITH DETAILS to display the processing options if the current display option for the session does not include DETAILS.

WITHOUT DETAILS
Directs the compiler to display only the keywords SET OPTIONS FOR SESSION or SET OPTIONS FOR DICTIONARY.

**AS**

Specifies whether the processing options are displayed or punched as comments or syntax.

- **COMments**
  Specifies that each line of the processing options is displayed or punched as a comment. Comment lines generated by the compiler are indicated by an asterisk and a plus sign (*+) in the first two columns and are ignored when resubmitted to the compiler.

- **SYntax**
  Specifies that the displayed or punched processing options appear in syntax format. Statements that are displayed (online only) or punched (online and batch) as syntax can be edited and resubmitted to the compiler as input.

**TO**

For PUNCH statements only, specifies the destination of the punched output.

- **module-specification**
  Causes the compiler to write punched output to the specified module. Expanded syntax for module-specification is presented as part of the DISPLAY and PUNCH syntax described in Displaying and Punching Entities (see page 92).

- **SYSpch**
  Causes the compiler to write punched output to the SYSPCH file.

**Example DISPLAY/PUNCH OPTIONS Statement**

The following sample screen shows the system generation compiler response to a DISPLAY OPTIONS request.

```
SYSGEN 17.0  NO ERRORS  DICT=CORPDICT  1/16  cv-name
DISPLAY OPTIONS FOR SESSION AS COMMENTS.
*+  SET OPTIONS FOR SESSION
  *+  NODE NAME IS '  
  *+  DICTIONARY NAME IS CORPDICT
  *+  USAGE MODE IS UPDATE
  *+  QUOTE IS '
  *+  EOF IS '/*'
  *+  DEFAULT IS OFF
  *+  NO PROMPT
  *+  ECHO
  *+  LIST
  *+  NO HEADER
  *+  INPUT COLUMNS ARE 1 THRU 79
  *+  OUTPUT LINE SIZE IS 80
  *+  PUNCH TO SYSPCH
  *+  .
```
SIGNOFF Statement

The SIGNOFF statement signals the end of an online session or batch run of the system generation compiler. SIGNOFF causes the compiler to:

- Display a transaction summary
- Free all resources held by the compiler
- Remove the session from the list of active sessions maintained by the transfer control facility (if executing under TCF)

Online, SIGNOFF does not transfer control to DC/UCF or TCF. The CLEAR key, or the top-line command CLEAR, must follow SIGNOFF to exit the system generation compiler.

⚠️ **Note:** For more information about top-line commands, see the CA IDMS Common Facilities section.

The SIGNOFF statement is recommended as the best way to terminate a compiler session; however, SIGNOFF is not required to terminate a session:

- **Online,** you can terminate a compiler session by entering the top-line command END.
- **In batch mode,** the system generation compiler assumes a SIGNOFF statement if none is present at the end of the input file.

SIGNOFF Statement Syntax

```
SIGNOFF
BYE
LOGOFF
```

SIGNOFF, BYE, and LOGOFF are synonyms and can be used interchangeably.

Example SIGNOFF Statement

The following sample screen shows the system generation compiler response to a SIGNOFF statement:

```
SIGNOFF.  SYSGEN 17.0  NO ERRORS END OF COMPILER cv-name  
** I DC601073 SIGNOFF ACCEPTED **  WORD 1
** TRANSACTION SUMMARY **
ENTITY             ADD MODIFY REPLACE DELETE DISPLAY
......................... ................ ................ ................
SYSTEM              0   1   0   0   0
PROGRAM             3   0   0   1   0
TASK                0   2   0   1   0
LINE                0   0   0   0   2
PTERM               0   0   0   0   2
```
NO ERRORS OR WARNINGS ISSUED FOR THIS COMPIL

The system generation compiler ignores any statements following a SIGNOFF statement. In the following example, the SIGNON and MODIFY SYSTEM statements are ignored:

SIGNOFF.
SIGNON DICTIONARY=OTHERDD.
MODIFY SYSTEM 77 PROGRAM POOL IS 560.

In the example, in which the user tried to end one session and begin another, elimination of the SIGNOFF statement would have produced the expected results.

SIGNON Statement -- Initiates an Online or a Batch Session

The SIGNON statement initiates an online or batch session of the system generation compiler and allows you to specify:

- The id and password of the user to be signed on to the system if not already signed on
- The data dictionary to be accessed by the compiler in a multiple dictionary environment
- The node that controls the dictionary to be accessed by the compiler
- The usage mode in which each dictionary area is to be accessed

Note: The teleprocessing network for a DC/UCF system can be defined only in a data dictionary that contains device definitions. Device definitions are stored in the installation default dictionary during installation. For more information about storing device definitions in a data dictionary, see the CA IDMS Installation and Maintenance section -- z/OS.

How the Compiler Performs a Signon Operation

When invoked, the system generation compiler automatically performs a signon operation using:

- The id of the user that is currently signed on to the DC/UCF system or the user running the batch job
- The current session default dictionary
- The current session default dictionary node

Therefore, you typically do not need to issue an explicit SIGNON statement after you invoke the compiler.

When You Need to Code a SIGNON Statement

The SIGNON statement is required if any of the following conditions apply:
If you have not signed on to the system and the DC/UCF system definition to be accessed is secured in the central security system, then you must issue a SIGNON statement. In this case, the SIGNON statement must include a user ID and a password (if the user ID is associated with a password). The signon also results in your being signed on to the DC/UCF system.

If you are already signed on to the DC/UCF system, you cannot enter a user ID or password on the IDD SIGNON statement.

You want to establish new compiler session parameters; that is, you want to access a different dictionary, or access a dictionary area in a different usage mode. SIGNON forces the termination of the current compiler session and initiates a new session with the specified parameters.

You want to access one or more areas of the data dictionary in a usage mode other than shared update.

SIGNON Statement Syntax

```
SIGNON

USEr name is user-id

PASsword is password

DICtionary name is dictionary-name

DICTIONARY NAME is ''

DBName

NODe name is node-name

NODEName is ''

USAge mode is UPDATE

PROtected UPDate for DDLDML

DDLDCLOD

DDLDCMSG

ALL
```

SIGNON Statement Parameters

**USER name is user-id**

Identifies the user who is signing on to the DC/UCF system.

*User-id* must be the identifier of a user defined to the CA IDMS central security facility.

This parameter is allowed only when running the system generation compiler online and no previous DC/UCF signon has been issued.

**PASsword is password**

Specifies the password assigned to the user.

You must provide a password if the specified user id is associated with a password. You should omit the PASSWORD parameter if the specified user id is not associated with a password.

**DICtionary name/DICTName/DBName is**

Specifies the data dictionary to be accessed by the system generation compiler.
If you omit the DICTIONARY NAME parameter from the SIGNON statement the compiler accesses the current session default dictionary. If no session default dictionary has been established, the compiler accesses the default dictionary established by the database name table in effect for the runtime environment.

DICTIONARY NAME, DICTNAME, and DBNAME are synonyms and can be used interchangeably.

- **dictionary-name**
  Explicitly specifies the name of the data dictionary to be accessed.
  The dictionary name specified must be defined in the database name table in effect for the local mode session or for the DC/UCF system controlling access to the dictionary.

- "
  Indicates the default data dictionary:

  - **Online or in batch mode under the central version**, the compiler accesses the system default dictionary as defined in the database name table for the system from which you initiated the compiler session or by the system identified by the NODE NAME parameter below.

  - **In local mode**, the compiler accesses the default dictionary defined by the database name table in effect for the local mode session.

**NODe name/NODEName is**

Identifies the node that controls the data dictionary to be accessed by the system generation compiler.

If you omit the NODE NAME parameter from the SIGNON statement:

- **Online or batch under the central version**, the compiler uses the current session default dictionary node if one has been established otherwise the request is forwarded to the node associated with the dictionary in the resource table of the DC/UCF system.

- **In local mode**, the NODE NAME parameter is not used.

- **node-name**
  Explicitly specifies the name of the node (DC/UCF system) controlling access to the dictionary accessed by the system generation compiler.
  *Node-name* must be the name of a DC/UCF system defined in the CA IDMS communications network.

- "
  Directs the compiler to route the request based solely on the dictionary name ignoring any session default dictionary node.

**USAge mode is**

Specifies the usage mode in which the compiler is to access the data dictionary.
### UPDATE
Directs the compiler to ready the dictionary in shared update usage mode. You can perform both update and retrieval operations against the dictionary. If the compiler is executing online or in batch mode under the central version, other users can also perform both update and retrieval operations against the dictionary. 

UPDATE is the default when you omit the USAGE MODE parameter from the SIGNON statement.

### PROtected UPDate
Directs the compiler to ready the dictionary in protected update usage mode. You can perform both retrieval and update operations against the dictionary. If the compiler is executing online or in batch mode under the central version, other users can access the dictionary only in shared retrieval usage mode while the compiler is active. While the compiler is waiting for input, other users can access the dictionary in an update usage mode.

### RETrieval
Directs the compiler to ready the dictionary in shared retrieval usage mode. You can perform only retrieval operations against the dictionary. If the compiler is executing online or in batch mode under the central version, other users can access the data dictionary in a retrieval or update usage mode.

#### for
Indicates the area of the data dictionary to which the specified usage mode applies.

Areas not explicitly specified are readied in the default usage mode, shared update.

- **DDLDML**
  Directs the compiler to ready the DDLDML area in the specified usage mode.

- **DDLDCLOD**
  Directs the compiler to ready the load area in the specified usage mode.

- **DDLDCMSG**
  Directs the compiler to ready the message area in the specified usage mode.

- **ALL**
  Directs the compiler to ready the DDLDML, DDLDCLOD, and DDLDCMSG areas in the specified usage mode.

### Example SIGNON Statement
The following example is a sample SIGNON statement:

```
SIGNON
  DICTIONARY NAME IS TSTDICT
  USAGE MODE IS RETRIEVAL FOR DDLDCMSG.
```

This statement has the following effects:

- The compiler accesses dictionary TSTDICT.
- The DDLDCMSG area of the dictionary is readied in shared retrieval usage mode. The DDLDML and DDLDCLOD areas are readied in shared update usage mode.
GENERATE Statement

The GENERATE statement directs the system generation compiler to validate a system definition and, if validation is successful, to create the executable form of the system. A DC/UCF system is not executable until the compiler successfully processes a GENERATE statement for the system.

When processing a GENERATE statement, the compiler updates object records in the data dictionary. If validation is not successful, however, no updating occurs and the system is not executable.

You must establish system currency before you submit a GENERATE statement.

GENERATE Statement Syntax

```
GENerate
```

Note: For more information about the compiler response to a GENERATE statement, see Compiler Activities (see page 77).

COPY Statement

The COPY statement copies entity definitions from one system to another. An entire system definition or portions of a system definition can be copied. When the copy operation is completed, system currency is set to the system receiving the copied definitions.

You can use the COPY statement to copy definitions from systems defined with the system generation compiler or the IDD DDDL compiler. However, definitions copied from IDD-defined entities are not complete for system generation purposes.

The SYSTEM and PARAMETERS options create new systems. All other options modify existing systems.

COPY Statement Authorization

<table>
<thead>
<tr>
<th>To perform this operation</th>
<th>You need this authority or privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy the system entity or system parameters from one DC/UCF system (source) to another DC/UCF system (target)</td>
<td>DCADMIN authority or CREATE on the system ID of the source system</td>
</tr>
<tr>
<td>Copy non-system entities (PROGRAMS, TASKS, LINES, PTERMS, LTERMS, DESTINATIONS, and QUEUES) from one DC/UCF system (source) to another DC/UCF system (target)</td>
<td>DCADMIN authority or DISPLAY on the system ID of the source system and ALTER on the system ID of the target system</td>
</tr>
<tr>
<td>Copy system entity from one IDD system (source) to another IDD system (target)</td>
<td>DCADMIN authority or CREATE on the system ID of the source IDD system. The IDD system ID is created internally as SYSTnnnn where nnnn is the version number of the ID system.</td>
</tr>
</tbody>
</table>
To perform this operation | You need this authority or privilege
--- | ---
Copy non-system entities from one IDD system (source) to another IDD system (target) | DCADMIN authority or ALTER on the system ID of the target system

COPY Statement Syntax

```
COPY DESTinations
    LINes
    LTERms
    PARAMETERS/PARMs
    PROgrams
    PTERms
    QUEues
    SYStem
    TASks

from system dc/ucf-version-number system-name 1 version-number

TO dc/ucf-version-number
```

COPY Statement Parameters

**DESTinations**

Copies all destination definitions.

**LINes**

Copies all line definitions.

**LTERms**

Copies all logical terminal definitions.

**PARAMeters/PARMs**

Copies the portions of the system definition that are defined by the following statements: ADSO, AUTOTASK, DEFAULT PROGRAM, IDD, KEYS, LOADLIST, MAPTYPE, OLM, OLQ, RUNUNITS, SQL CACHE, STORAGE POOL, SYSTEM, and XA STORAGE POOL.

PARAMETERS and PARMS are synonyms and can be used interchangeably.

**PROgrams**

Copies all program definitions.

**PTERms**

Copies all physical terminal definitions.

**QUEues**

Copies all queue definitions.
SYStem

Copies the entire system definition.

TASks

Copies all task definitions.

from system

Identifies the system whose definition is being copied.

- dc/ucf-version-number
  Specifies the version number of a DC/UCF system defined in the data dictionary with the system generation compiler.
  dc/ucf-version-number must be an integer in the range 1 through 9,999.

- system-name
  Specifies the name of a system defined in the data dictionary with the IDD DDDL compiler.

- version-number
  Qualifies the named system with a version number.
  Version-number must be an integer in the range 1 through 9,999. The default is 1.

TO dc/ucf-version-number

Identifies the system to which the specified definition is being copied. dc/ucf-version-number must be an integer in the range 1 through 9,999, with the following restrictions:

- If you specify SYSTEM or PARAMETERS, dc/ucf-version-number cannot be the version number of an existing system. The compiler creates a new system with the specified version number and copies the appropriate portions of the definition to the new system.

- If you specify an option other than SYSTEM or PARAMETERS, dc/ucf-version-number must be the version number of an existing system. The compiler copies the appropriate portions of the definition to the existing system.

Example COPY Statement

When copying component entity definitions, the compiler works more efficiently if the system to which the definitions are being copied does not already include any occurrences of the entity. For example, if system 9 is to include all the same programs as system 8 plus the programs PXT078 and PXT079, the first set of statements shown next would add the programs to system 9 more efficiently than would the second set of statements:

Recommended: COPY PROGRAMS FROM SYSTEM 8 TO 9.
ADD PROGRAM PXT078.
ADD PROGRAM PXT079.
Not recommended: ADD PROGRAM PXT078.
ADD PROGRAM PXT079.
COPY PROGRAMS FROM SYSTEM 8 TO 9.
VALIDATE Statement

The VALIDATE statement directs the system generation compiler to cross-check the relationships between entities in a system definition. You must establish system currency before you submit a VALIDATE statement.

You can issue a VALIDATE statement to ensure that a correct system definition exists before processing continues. A system is not executable, however, nor do component entities participate in an executable system, until the compiler successfully processes a GENERATE statement for the system.

VALIDATE Statement Syntax

```
►►─── VALidate ────────────────────────────────────────────────────────────────►◄
```

Example VALIDATE Statement

The following sample screen shows the system generation compiler response to a VALIDATE statement:

```
SYSGEN 17.0 2 ERRORS DICT=CORPDICT 1/9 cv-name
VALIDATE.
*+ E DC301030 UPDATE OF TASK 'TXT002' BUT PROGRAM TO BE INVOKED DOES WORD 2
*+ - DC301030 NOT EXIST
*+ E DC301030 UPDATE OF TASK 'TXT009' BUT PROGRAM TO BE INVOKED DOES WORD 2
*+ - DC301030 NOT EXIST
*+ W DC301019 WARNING - ERRORS FOUND DOING SYSTEM VALIDATION WORD 2
```

Note: For more information about the validation process, see Compiler Activities (see page 77).

INCLUDE Statement

The INCLUDE statement submits statements stored as a module in the data dictionary to the system generation compiler. Modules containing system generation statements can be added to the data dictionary by using the IDD DDDL compiler. Each statement must be complete and must conform to the processing options established for input statements for the current session.

You can code any number of INCLUDE statements in the system generation source input. Upon encountering an INCLUDE statement, the compiler retrieves and processes each statement from the named module. When the compiler reaches the end of an included module, it continues processing with the next statement (if any) after the INCLUDE statement.
Note: Included modules cannot be nested; that is, a module that is the object of an INCLUDE statement cannot itself contain an INCLUDE statement. Only one INCLUDE statement can be coded on a line.

INCLUDE Statement Authorization

To use the INCLUDE statement you must have IDD authority in the data dictionary.

INCLUDE Statement Syntax


code
```
INCLUDE module module-name
```


code
```
Version 1 version-number
```


code
```
LANguage is language
```

INCLUDE Statement Parameters

module module-name

Specifies the module to be included.

Module-name must be the name of a data dictionary module containing system generation statements.

Version version-number

Qualifies the module name with a version number.

Version-number must be an integer in the range 1 through 9,999. The default is 1.

LANguage is language

Qualifies the module name with a language.

If the data dictionary contains two or more modules with the specified name and version number, the INCLUDE statement must specify a language.

Example INCLUDE Statement

The next example illustrates the use of the INCLUDE statement. Assume that the data dictionary includes the modules SYSDPT01 version 2 and TSKDPT01 version 1, which contain the following system generation statements:

SYSDPT01 version 2:

```
DEFAULT TEMPORARY PROGRAM
LANGUAGE IS ASSEMBLER
NOSAVEAREA.
PROGRAM TEST6.
```

TSKDPT01 version 1:

```
TASK TTASK6 INVOKES PROGRAM TEST6 INPUT.
TASK TTASK7 INVOKES PROGRAM TEST7 NOINPUT.
```
When the following statements are submitted to the system generation compiler, system 5 is modified to include the programs defined in module SYSDPT01 version 2, the tasks defined in module TSKDPT01 version 1, and the queue TESTQ as defined in the QUEUE statement:

```plaintext
MODIFY SYSTEM 5.
INCLUDE MODULE SYSDPT01 VERSION 2.
INCLUDE TSKDPT01.
QUEUE TESTQ INVOKES TASK TTASK7 THRESHOLD 5.
```

**Compiler Messages**

The system generation compiler lists error, warning, and informational messages as comments immediately following the statements to which they apply. At the end of each message line, the compiler indicates the position in the statement of the word that caused the message to be issued.

**Message Lines Begin with '*+'**

Each message line begins with an asterisk followed by a plus sign (+) in the first two columns. You do not need to erase compiler-generated messages when resubmitting displayed or punched output. The compiler ignores lines that begin with *+.

**Transaction Summary**

At the end of an online compiler session or batch run of the compiler, the transaction summary generated by the SIGNOFF statement indicates the total number of E-level and W-level messages issued during the compiler session. Additionally, when the compiler is executed online, the number of E-level messages produced for each group of statements processed is displayed in the top-line message area.

Messages issued by the compiler are documented in *CA IDMS Messages and Codes section*. You can use the HELP DC command to display a description of any message issued by the system generation compiler.

**Compiler Messages Syntax**

```
HELP DC message-number
```

**Compiler Messages Parameter**

`message-number`

Specifies the 6-digit number associated with the error, warning, or informational message. The characters DC and the message number cannot be separated. For example, to display a description of the message with identifier 601014, you would enter:

```
HELP DC601014
```
Execution Modes

You can execute the system generation compiler either online or in batch mode. All system generation statements are valid in both modes and produce the same response except as noted in the descriptions of the individual statements.

Online System Generation

The Compiler Uses a Text Editor

When executing online, the system generation compiler uses a text editor that operates independently of the compiler. The text editor writes input to and output from the compiler to a work file allocated for each online session. The contents of the work file are displayed at the terminal. You can manipulate and modify the contents of the work file by using the text editor commands. Instructions for using the text editor are presented in the CA IDMS Common Facilities section.

Invoking the Online Compiler

You can invoke the online system generation compiler under the transfer control facility (TCF). TCF allows you to switch between the system generation compiler and other CA IDMS online development tools without terminating the current compiler session. Instructions for using TCF are presented in the CA IDMS Common Facilities section.

The following aspects of online system generation are discussed below:

- An online session
- Online system modification
- Copying definitions from another dictionary
- Session recovery

Online System Modification

How to Modify an Entity Definition Online

You can modify an entity definition online:

1. **Use the DISPLAY or PUNCH verb to display the entity definition as syntax.** The DISPLAY and PUNCH verbs are described under 4.6, “Displaying and Punching Entities”.

2. **Modify the definition by typing over the displayed syntax.** You can make changes to the entire work file by scrolling through the file and by using the top-line command APPLY (or a control key associated with the APPLY command) to apply the changes to the files without submitting the syntax to the compiler.

3. **Submit the modified syntax to the compiler by using the top-line command UPDATE** (or a control key associated with the UPDATE command).
Considerations when Modifying a System Online

When modifying a system online, the following considerations apply:

- Be sure that currency is established on the correct system before submitting the modified syntax to the compiler. If necessary, you can insert an ADD or MODIFY SYSTEM statement into the work file to establish the appropriate currency.

  Note: For more information about system currency, see Currency (see page 80).

- Be sure that all syntax to be resubmitted to the compiler is in syntax, not comment, format. You can use the space bar to remove the *+ from syntax lines that are displayed as comments.

- Entity definitions are displayed as ADD statements. Be sure to change ADD to the appropriate verb before submitting the modified syntax to the compiler. If the processing options for the session specify DEFAULT IS ON, the compiler changes ADD statements for existing entities to MODIFY statements. You use the SET OPTIONS statement to establish processing options for a session.

  Note: For more information about the SET OPTIONS statement, see SET OPTIONS Statement (see page 98).

- All system generation statements in the work file (not just the displayed statements) are processed when the UPDATE command is executed. Be sure to erase any statements that you do not want to resubmit to the compiler.

- Be sure to submit VALIDATE and GENERATE statements, as appropriate, for the systems that you modify. VALIDATE (VALIDATE Statement (see page 114)) and GENERATE (GENERATE Statement (see page 111)) are described under Compiler-Directive Statements (see page 97).

Copying Definitions from Another Dictionary

You can use the online system generation compiler to copy entity definitions from one data dictionary to another:

1. Sign on to the dictionary containing the definition to be copied.

2. Display the definition to be copied as syntax.

3. Insert a SIGNON statement that names the dictionary to which the definition is to be copied at the beginning of the work file.

  Note: For more information about the SIGNON statement, see SIGNON Statement (see page 107).
4. Use the top-line command UPDATE (or a control key associated with the UPDATE command) to submit the contents of the work file to the system generation compiler. Top-line commands are documented in the CA IDMS Common Facilities section.

**Session Recovery**

The system generation compiler updates the data dictionary as it processes the syntax that you submit. In the event that either the compiler or the DC/UCF system terminates abnormally during an online compiler session, updates already made to the data dictionary are unaffected.

**After a Compiler Abend**

After a compiler abend, you can resume the abended session after recovery has occurred. When you enter the task code to invoke the compiler, the online session resumes with the processing options that were in effect before the abend and the current work file intact.

**After a DC/UCF System Abend**

After a DC/UCF system abend, the work file and all session options are lost. When you enter the task code to invoke the compiler, the compiler initiates a new online session.

**Note:** If the system terminates while an online session is suspended, the work file and all session options are lost.

**Batch System Generation**

**Running the Compiler in Batch Mode**

When executing in batch mode, the system generation compiler reads source statements from an input file and writes all output except punched definitions to the CA IDMS/DC SYSGEN Compiler Activity List. The compiler writes punched definitions either to the SYSPCH file or to a data dictionary module.

The batch compiler can run either under the central version or in local mode. To ensure data integrity when using local mode operations, you should either back up the data dictionary before executing the system generation compiler or perform journaling during compiler execution.

**Running in Local Mode**

When running in local mode, the compiler cannot ready in an update usage mode any areas already held by a DC/UCF system in an update usage mode. An update usage mode is required for:

- The **DDLDML area** when the source input includes an ADD, MODIFY, DELETE, VALIDATE, or GENERATE statement.
• The **DDLDCLOD area** when the source input includes a GENERATE statement for a system that includes an added or modified resource table (as a result of NODE and RESOURCE TABLE statements).

JCL used to execute the system generation compiler in batch mode is presented next for z/OS, z/VSE, and z/VM systems, followed by a sample CA IDMS/DC SYSGEN Compiler Activity List.

### z/OS Systems

#### Central Version JCL

The following JCL is used to execute the system generation compiler under the central version.

**RHDCCSGEN**

```plaintext
// EXEC  PGM=RHDCSGEN,REGION=1024K
//STPLIB DD DSN=idsm.dba.loadlib,DISP=SHR
//                  DD DSN=idsm.custom.loadlib,DISP=SHR
//                  DD DSN=idsm.cagjload,DISP=SHR
//                  DD DSN=idsm.sysctl,DISP=SHR
//                  DD DSN=sysmsg.ddlcmdmsg,DISP=SHR
//SYSLST DD SYSOUT=A
//SYSPCH DD SYSOUT=A
//SYSSIDMS DD *
DMCL=dmcl-namePut other SYSSIDMS parameters here, as appropriate/*
//SYSIPT DD *
Put source system generation statements here/*
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>idsm.dba.loadlib</code></td>
<td>Name of the CA IDMS load library containing the DMCL and database name table load modules</td>
</tr>
<tr>
<td><code>idsm.custom.loadlib</code></td>
<td>Data set name of the load library containing the customized CA IDMS load modules</td>
</tr>
<tr>
<td><code>idsm.cagjload</code></td>
<td>Data set name of the load library containing the vanilla CA IDMS software</td>
</tr>
<tr>
<td><code>idsm.sysctl</code></td>
<td>Data set name of the SYSCTL file</td>
</tr>
<tr>
<td><code>sysmsg.ddlcmdmsg</code></td>
<td>DDname of the system message area (DDLDCMMSG)</td>
</tr>
<tr>
<td><code>idsm.sysmsg.ddlcmdmsg</code></td>
<td>Data set name of the system message area (DDLDCMMSG)</td>
</tr>
<tr>
<td><code>SYSSIDMS</code></td>
<td>DDname of the SYSSIDMS parameter file used to specify physical requirements of the environment (i.e. DMCL, dictionary name), runtime directives, and operating system-dependent file information</td>
</tr>
</tbody>
</table>

**Note:** For more information about SYSSIDMS parameters, see the CA IDMS Common
Local Mode JCL

To execute the system generation compiler in local mode:

- Remove the SYSCTL DD statement
- Insert the following statements after the STEPLIB DD statements:

  ```
  //dcdml DD DSN=ids.system.ddldml,DISP=SHR
  //dclod DD DSN=ids.system.ddldclod,DISP=SHR
  //sysjrnl DD DSN=ids.tapejrnl,DISP=
  ```

<table>
<thead>
<tr>
<th>idms.system.ddldml</th>
<th>Data set name of the system dictionary definition area (DDLDML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.system.ddldclod</td>
<td>Data set name of the system dictionary definition load area (DDLDCLOD)</td>
</tr>
<tr>
<td>idms.tapejrnl</td>
<td>Data set name of the tape journal file</td>
</tr>
<tr>
<td>dcdml</td>
<td>DDname of the system dictionary definition area (DDLDML)</td>
</tr>
<tr>
<td>dclod</td>
<td>DDname of the system dictionary definition load area (DDLDCLOD)</td>
</tr>
<tr>
<td>dcmgs</td>
<td>DDname of the system message area (DDLDCLMSG)</td>
</tr>
<tr>
<td>sysjrnl</td>
<td>DDname of the tape journal file (if the DMCL uses a tape journal)</td>
</tr>
</tbody>
</table>

z/VSE Systems

Central Version JCL

The following JCL is used to execute the system generation compiler under the central version.

**RHDCSGEN**

````
// EXEC PROC=IDMSLBLS
// UPSI b if specified in the IDMSOPTI module
// EXEC RHD CSGEN
/*
Input source system generation statements here/*
```

You can define a SYSCTL file in the JCL to override IDMSOPTI specifications for central version operations. The SYSCTL file definition is included in the IDMSLBLS procedure (shown next).

<table>
<thead>
<tr>
<th>IDMSLBL</th>
<th>Name of the procedure provided at installation that contains file definitions for CA IDMS LS dictionaries and databases.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Appropriate one- through eight-character UPSI bit switch, as specified in the IDMSOPTI module</td>
</tr>
</tbody>
</table>

Local Mode JCL

To execute the system generation compiler in local mode:
• Remove the UPSI specification.

• Insert the following statement before the EXEC statement:

```plaintext
// TLBL   sys009,'idms.tapejrnl',,nnnnnn,,f
// ASSGN   sys009,TAPE,vol=nnnnnn
```

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.tapejrnl</td>
<td>File-id of the tape journal file</td>
</tr>
<tr>
<td>sys009</td>
<td>Logical unit assignment of the tape journal file</td>
</tr>
</tbody>
</table>

### Store Source Statements in a Source Library

You can store some or all of the source statements to be submitted to the system generation compiler as a member in a source statement library. To copy the library member into the job stream, you use the =COPY facility, which is implemented through the =COPY IDMS statement.

The =COPY IDMS statement identifies the source statement library member. You code the statement in the JCL along with other source statements (if any) to be submitted to the compiler. You can code multiple =COPY IDMS statements to identify multiple source statement library members that contain different system generation statements.

=COPY IDMS statements and source system generation statements can be intermixed in the JCL. The source statements are submitted to the compiler in the order in which they occur, whether they are coded directly in the JCL or copied in through the =COPY facility.

### Sublibrary ID Syntax

```
=COPY IDMS
```

```
A. sublibrary-id.
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sublibrary-id</td>
<td>Identifies the source statement sublibrary that includes the member identified by member-name. The default is A.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>member-name</td>
<td>Identifies the source statement library member that contains the source statements to be submitted to the system generation compiler.</td>
</tr>
</tbody>
</table>

⚠️ **Note:** If the source statements are stored as a member in a private source statement library, the DLBL file type for the library must be specified as DA.
The following z/VM commands are used to execute the system generation compiler under the central version.

**RHDCSGEN**

- **FILEDEF SYSLST PRINTER**
- **FILEDEF SYSIPT DISK sgen input a**
- **EXED IDMSFD**
- **OSRUN RHDCSGEN**

<table>
<thead>
<tr>
<th>sysipt</th>
<th>DDname for user-defined system generation statements modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>sgen input a</td>
<td>File identifier of the file containing the source system generation statements</td>
</tr>
<tr>
<td>ppp</td>
<td>Record length of the sysipt data file</td>
</tr>
<tr>
<td>nnn</td>
<td>Blocksize of the sysipt data file</td>
</tr>
<tr>
<td>IDMSFD</td>
<td>z/VM EXEC that defines all system-required FILEDEFs, TXTLIBs, and LOADLIBs</td>
</tr>
<tr>
<td>RHDCSGEN</td>
<td>Program name to be executed from the z/VM LOADLIB</td>
</tr>
</tbody>
</table>

**Local Mode**

To execute the system generation compiler in local mode, perform one of the following:

- Link RHDCSGEN with an IDMSOPTI program that specifies local as the execution mode
- Specify *LOCAL* as the first input parameter of the filename, type, and mode identified by SGEN INPUT A in the IDMSFD exec
- Modify the OSRUN statement:
  ```
  OSRUN RHDCSGEN PARM=’*LOCAL*’
  ```

**Note:** This option is valid only if the OSRUN command is issued from a System Product interpreter or an EXEC2 file.

To create the SYSIPT file, enter these z/VM commands:

```zsh
XEDIT sysipt data a (NOPROF INPUT
Put system generation input statements here - Hit enter of end input
FILE
```

Instructions for creating an IDMSOPTI module are presented in *CA IDMS System Operations section*. 

⚠️ **Note:** This option is valid only if the OSRUN command is issued from a System Product interpreter or an EXEC2 file.
Note: For more information about central version and local mode operations in the z/VM environment, see the CA IDMS Installation and Maintenance section – z/VM.

Lib-name Syntax

\[ \texttt{COPY IDMS \ lib-name member-name} \]

Lib-name Parameters

/lib-name

Identifies the link name of a library containing the member identified by \textit{member-name}.

/member-name

Identifies the library member that contains the source statements to be submitted to the system generation compiler.

CA IDMS/DC SYSGEN Compiler Activity List

What the Activity List Contains

The CA IDMS/DC SYSGEN Compiler Activity List contains all text output by the system generation compiler, including echoed input lines, responses to DISPLAY requests, and compiler-generated messages. When the output line size is 132, as specified in the SET OPTIONS statement, each echoed line begins with a line number, and messages issued by the compiler bear the line number of the line to which they refer. When the output line size is 80, the CA IDMS/DC SYSGEN Compiler Activity List does not show line numbers. The SET OPTIONS statement is described under \textit{SET OPTIONS Statement}\ (see page 98).

A sample CA IDMS/DC SYSGEN Compiler Activity List is shown next.

```
RHDCSGEN 17.0    CA, INC.    CA IDMS/DC SYSGEN COMPILER ACTIVITY LIST    DATE    TIME    PAGE
volser   mm/dd/yy  hhmmssss  pppp

000001   SIGNON USER ABC PASSWORD ?  DICTIONARY CORPDICT.
000003   ADD SYSTEM 9
000004   MAXIMUM ERUS IS 20      
000005   MAXIMUM TASKS IS 50
000006   PROGRAM POOL IS 100
000007   REENTRANT POOL IS 200
000008   STORAGE POOL IS 1000.
000009   ADD ADSO.
000010   ADD OLM.
000012   ADD PROGRAM RHDCBYE.
000013   ADD PROGRAM PXT001
000014   LANGUAGE IS ASSEMBLER.
000016   ADD TASK TXT001
000017   INVOKES PROGRAM PXT001.
000019   GENERATE.
```
**System 09 Generation Summary**

- **Entity Update Delete Totals**

- **Programs**: 2
- **Tasks**: 1
- **Queues**: 0
- **Lines**: 0
- **PTERMS**: 0
- **LTERMS**: 0
- **Destinations**: 0

**System 09 Generation Summary**

- **Add System 9**
  - ABEND STORAGE IS 200
  - ABRU NOSNAP
  - CUSHDEN TASKS IS 0
  - CUSHION IS 0
  - CVMNUMBER IS 0
  - CWA SIZE IS 0
  - JRE COUNT IS DEFAULT
  - NODUMP
  - ECB LIST IS DEFAULT
  - EXTERNAL WAIT IS 600
  - INACTIVE INTERVAL IS OFF
  - INTERNAL WAIT IS 1800
  - JOURNAL RETRIEVAL
  - LOG TYPE IS FILE IS COMSLOGA
  - MAXIMUM ERUS IS 20
  - MAXIMUM TASKS IS 50
  - NEW COPY IS MANUAL
  - OPERATING SYSTEM IS MVS
  - PAGE RELEASE IS NO
  - PRINT KEY IS OFF
  - PRINTER CHECKPOINT IS OFF
  - PROGRAM POOL IS 100
  - NOPROTECT
  - RCE COUNT IS DEFAULT
  - REENTRANT POOL IS 200
  - REPORT RETENTION IS 7
  - RESOURCE TIMEOUT INTERVAL IS FOREVER PROGRAM IS RHDCBYE
  - RUPRTY IS 100
  - QUEUE JOURNAL ALL
  - STACKSIZE IS 1200
  - STATISTICS INTERVAL OFF NOLINE NOTASK
  - STORAGE KEY IS 9
  - STORAGE POOL IS 1000

**Additional System 9 Details**

- ADD ADS0
  - ADSTASK IS ADS
  - ADS2TASK IS ADS2
  - ADS2TF TASK IS ADS2U
+++ ACTIVITY LOG IS YES
+++ AUTOSTATUS IS YES OPTIONAL
+++ COBOL MOVE IS NO
+++ DIAGNOSTIC SCREEN IS YES
+++ FAST MODE THRESHOLD IS 6000
+++ MAXIMUM LINKS IS 10
+++ MENU IS USER
+++ NEWPAGE MAPOUT IS NO
+++ PRIMARY POOL IS 4000
+++ SECONDARY POOL IS 2000
+++ RESOURCES ARE FIXED
+++ DIALOG STATISTICS OFF
+++ STATUS DEFINITION RECORD IS ADSO-STAT-DEF-REC OPTIONAL
+++ .
+++ ADD DUM
+++ DATA FIELD CHARACTER IS X'6D'
+++ DELIMIT CHARACTER IS X'1C'
+++ FIELD SELECT CHARACTER IS X'6C'
+++ FIELD START CHARACTER IS X'1E'
+++ TRANSLATE CHARACTER IS C'@'
+++ NEW COPY IS NO
+++ FIELD EDIT IS NO
+++ NUMERIC FIELD ORDER IS STANDARD
+++ NUMERIC FIELD DECIMAL-POINT IS PERIOD
+++ PAGE FORWARD PFKEY IS PF8
+++ PAGE BACKWARD PFKEY IS PF7
+++ PAGING STORAGE IS 10
+++ QUEUE RETENTION IS 255
+++ .
+++ ADD PROGRAM RHDCBEYE
+++ RHDCGEN 17.0 CA, INC. DATE TIME PAGE
+++ volser CA IDMS/DC SYSGEN COMPILER ACTIVITY LIST mm/dd/yy hhmmssss pppp
+++ CONCURRENT
+++ DUMP THRESHOLD IS 0
+++ ENABLED
+++ ERROR THRESHOLD IS 5
+++ ISA SIZE IS 0
+++ LANGUAGE IS COBOL
+++ NOMAINLINE
+++ NEW COPY IS ENABLED
+++ OVERLAYABLE
+++ PROGRAM
+++ PROTECT
+++ QUASIREENTRANT
+++ NONRESIDENT
+++ REUSABLE
+++ SAVEAREA
+++ .
+++ ADD PROGRAM PX7001
+++ CONCURRENT
+++ DUMP THRESHOLD IS 0
+++ ENABLED
+++ ERROR THRESHOLD IS 5
+++ ISA SIZE IS 0
+++ LANGUAGE IS ASSEMBLER
+++ NOMAINLINE
+++ NEW COPY IS ENABLED
+++ OVERLAYABLE
+++ PROGRAM
+++ PROTECT
+++ QUASIREENTRANT
+++ NONRESIDENT
+++ REUSABLE
+++ SAVEAREA
+++ .
+++ ADD TASK TXT001
+++ ENABLED
+++ EXTERNAL
+++ INACTIVE INTERVAL IS SYSTEM
+++ INVOKES PROGRAM PX7001
+++ INPUT
+++ NOJOURNAL
+++ NOMAP
+++ PRINT KEY IS SYSTEM
+++ PRIORITY IS 100
+++ RESOURCE TIMEOUT INTERVAL IS SYSTEM PROGRAM IS SYSTEM
+++ NOSAVE
+++ LOCATION IS BELOW
+++ TRANSACTION START
+++ RHDCGEN 17.0 CA, INC. DATE TIME PAGE
System Definition

This section outlines the system generation statements that you use to define:

- Basic System (see page 127)
- DC Teleprocessing Network (see page 128)
- UCF Teleprocessing Environment (see page 129)
- Online Products (see page 130)
- DDS Network (see page 131)

Basic System

The following table presents the system generation statements that you use to define the basic DC/UCF system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM</td>
<td>One for each DC/UCF system to be defined in the data dictionary</td>
</tr>
<tr>
<td>Statements</td>
<td>Comments</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DEFAULT PROGRAM</td>
<td>One each time you want to change default characteristics for subsequent program definitions</td>
</tr>
<tr>
<td>IDD</td>
<td>One to define default usage modes for IDD DDDL-compiler access to the data dictionary</td>
</tr>
<tr>
<td>LOADLIST</td>
<td>One for each load list to be available to the DC/UCF system at runtime</td>
</tr>
<tr>
<td>NODE</td>
<td>One to identify the communications method that is used by the DC/UCF system to access other nodes in the network</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>One for each database procedure not link edited with a subschema</td>
</tr>
<tr>
<td></td>
<td>One for each subschema not eligible for automatic program definition at runtime</td>
</tr>
<tr>
<td>RESOURCE TABLE</td>
<td>One to define the location (node) of remote resources that the DC/UCF system accesses</td>
</tr>
<tr>
<td>RUNUNITS</td>
<td>One for each alternate dictionary for which you want to predefined load-area run units</td>
</tr>
<tr>
<td></td>
<td>One for each database catalog containing SQL-defined and SQL security objects for which you want to predefined load-area run units</td>
</tr>
<tr>
<td></td>
<td>One for signon run units you wish to predefined</td>
</tr>
<tr>
<td>STORAGE POOL</td>
<td>One for each secondary 24-bit storage pool</td>
</tr>
<tr>
<td>XA STORAGE POOL</td>
<td>One for each 31-bit storage pool (for systems supporting 31-bit addressing only)</td>
</tr>
</tbody>
</table>

**Note:** To facilitate the monitoring and maintenance of the system, the data dictionary, and the database, a system configured primarily to perform database services for batch programs should include the following online components:

- The DCMT and OPER system tasks
- The online DDDL compiler
- The online schema compiler
- The online command facility
- The online subschema compiler

**DC Teleprocessing Network**

The following table presents the system generation statements that you use to define the DC teleprocessing network. These statements are coded in addition to the statements used to define the basic system to create a system that includes the teleprocessing monitor DC.
<table>
<thead>
<tr>
<th>Statements</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE</td>
<td>One to define the UCF line</td>
</tr>
<tr>
<td>PTERM</td>
<td>One for each physical terminal associated with the UCF line</td>
</tr>
<tr>
<td>LTERM</td>
<td>One for each physical terminal associated with the UCF line</td>
</tr>
</tbody>
</table>

More Information:

- For more information about the UCF macros, see the CA IDMS System Operations section.

- For more information about the LINE (LINE Statement (see page 281)), PTERM (PTERM Statement (see page 284)), and LTERM (LTERM Statement (see page 290)) statements, see Teleprocessing Network Statements (see page 280).
Online Products

The following table presents the system generation statements used to define CA IDMS products that function online under the DC/UCF system. These statements are coded in addition to the statements listed in the preceding tables in this section to create a DC/UCF system that includes CA ADS, CA ICMS, Performance Monitor, Online mapping facility, and CA OLQ.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CA ADS and CA ADS/Batch:</strong></td>
<td></td>
</tr>
<tr>
<td>ADSO</td>
<td>One to define dialog-generation and runtime characteristics of the CA ADS environment</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>One for each program and map supplied with CA ADS</td>
</tr>
<tr>
<td>TASK</td>
<td>One for each task used to invoke the application and dialog generators and the CA ADS runtime system</td>
</tr>
<tr>
<td><strong>Performance Monitor:</strong></td>
<td></td>
</tr>
<tr>
<td>PROGRAM</td>
<td>One for each program and map supplied with Performance Monitor</td>
</tr>
<tr>
<td>TASK</td>
<td>One for each task used to invoke the Performance Monitor</td>
</tr>
<tr>
<td><strong>CA ICMS (Information Center Management System):</strong></td>
<td></td>
</tr>
<tr>
<td>PROGRAM</td>
<td>One for each program and map used by IDB™ Manager and IDB Communications</td>
</tr>
<tr>
<td>TASK</td>
<td>One for each task used to invoke IDB Manager or IDB Communications</td>
</tr>
<tr>
<td><strong>CA OLQ:</strong></td>
<td></td>
</tr>
<tr>
<td>OLQ</td>
<td>One to define runtime characteristics of CA OLQ</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>One for each program and map supplied with CA OLQ</td>
</tr>
<tr>
<td>TASK</td>
<td>One for each task used to invoke CA OLQ</td>
</tr>
<tr>
<td><strong>Online mapping facility (OLM):</strong></td>
<td></td>
</tr>
<tr>
<td>OLM</td>
<td>One to define runtime characteristics of OLM</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>One for each program and map supplied with OLM</td>
</tr>
<tr>
<td>TASK</td>
<td>One for each task used to invoke OLM</td>
</tr>
</tbody>
</table>

**More Information:**

- For more information about the LINE (LINE Statement (see page 281)), PTERM (PTERM Statement (see page 284)), and LTERM (LTERM Statement (see page 290)) statements, see Teleprocessing Network Statements (see page 280).

- For more information about system tasks and programs, see System Programs and Tasks (see page 355).
DDS Network

The following table presents the system generation statements that you use to define the DDS network (CA IDMS DDS users only). These statements are coded in addition to the statements listed in the preceding tables in this section to define a DC/UCF system in a DDS network.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE</td>
<td>One CCI LINE statement, or one DDS LINE statement with a specific VTAM type PTERM, or one SOCKET LINE statement with a specific DDSTCP/IP type PTERM.</td>
</tr>
<tr>
<td>PTERM</td>
<td>One for each physical terminal that routes data to and from other DDS nodes concurrently</td>
</tr>
<tr>
<td>LTERM</td>
<td>One for each PTERM defined</td>
</tr>
</tbody>
</table>

⚠️ Note: For more information about the LINE (LINE Statement (see page 281)) and PTERM (PTERM Statement (see page 284)) statements, see Teleprocessing Network Statements (see page 280).

SYSTEM Statement

The SYSTEM statement is used to establish various characteristics of the DC/UCF runtime system. Each DC/UCF system definition includes one SYSTEM statement.

This section presents the syntax for the SYSTEM statement and describes each SYSTEM statement parameter. Additionally, information on using various parameters and examples are provided.

Parameters are presented in alphabetical order. ADD/MODIFY/DELETE and DISPLAY/PUNCH syntax are presented together and the syntax rules follow the presentation of both types of syntax. Syntax rules presented in are not repeated unless special considerations apply.

SYSTEM Statement Parameters

SYSTEM statement parameters determine the behavior of the DC/UCF runtime system with respect to:

- Abend handling and timed functions
- Backup and recovery
- The operating environment
- Print control
- Program management
- Record locking
- Resource management
- Run-unit management
- Storage management
- Storage protection
- System monitoring
- Task management

**Function of SYSTEM Statement Parameters**

The following table briefly describes the function of each SYSTEM statement parameter. The parameters are grouped by the categories listed above.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abend Handling and Time Parameters</strong></td>
<td>Activate mechanisms to detect and process abending programs and tasks</td>
</tr>
<tr>
<td>ABEND STORAGE</td>
<td>Allocates storage for abend processing in the event of a storage stack overflow in a task control element</td>
</tr>
<tr>
<td>CHKUSER TASKS</td>
<td>Allocates check-user subtasks to detect abnormally terminated external request units</td>
</tr>
<tr>
<td>DEADLOCK DETECTION INTERVAL</td>
<td>Specifies the frequency with which the system checks for deadlocks</td>
</tr>
<tr>
<td>EXTERNAL WAIT</td>
<td>Specifies the amount of time the system waits for an external request unit to issue a database request</td>
</tr>
<tr>
<td>INACTIVE INTERVAL</td>
<td>Specifies the amount of time the system permits a task to wait for a resource</td>
</tr>
<tr>
<td>INTERNAL WAIT</td>
<td>Specifies the amount of time the system permits an external request unit to wait for a resource</td>
</tr>
<tr>
<td>QUIESCE WAIT</td>
<td>Specifies the amount of time the system permits a task to wait for a quiesce operation to terminate before abnormally terminating the task</td>
</tr>
<tr>
<td>RECOVERY WAIT</td>
<td>Specifies the amount of time the system permits a task to wait for a resource to be recovered by a failed data sharing group member before abnormally terminating the task</td>
</tr>
<tr>
<td>RESOURCE TIMEOUT</td>
<td>Specifies the amount of time the system permits a terminal or external request unit to be inactive and identifies the program the system invokes to handle resources associated with a terminal or external request unit that remains inactive beyond the specified time limit</td>
</tr>
<tr>
<td>Parameters</td>
<td>Descriptions</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
</tr>
<tr>
<td><strong>RUNAWAY INTERVAL</strong></td>
<td>Specifies the amount of time the system permits a task to execute without returning control to the system</td>
</tr>
<tr>
<td><strong>SNAP SYSTEM</strong></td>
<td>Specifies whether to write a system snap dump to the DC/UCF log.</td>
</tr>
<tr>
<td><strong>SNAP SYSTEM PHOTO</strong></td>
<td>Specifies whether to write a system photo snap to the DC/UCF log.</td>
</tr>
<tr>
<td><strong>SNAP TASK</strong></td>
<td>Specifies whether to write a task snap dump to the DC/UCF log.</td>
</tr>
<tr>
<td><strong>SNAP TASK TRACE</strong></td>
<td>Specifies whether to write task traces to the DC/UCF log.</td>
</tr>
<tr>
<td><strong>SNAP TASK PHOTO</strong></td>
<td>Specifies whether to write a task photo snap to the DC/UCF log.</td>
</tr>
<tr>
<td><strong>TICKER INTERVAL</strong></td>
<td>Specifies the frequency with which the system checks for time-related events (such as runaway tasks)</td>
</tr>
</tbody>
</table>

**Backup and Recovery Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Control journaling options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JOURNAL</strong></td>
<td>Specifies whether the system writes BGIN and ENDJ checkpoints to the journal file for transactions that perform no updates</td>
</tr>
<tr>
<td><strong>RETRIEVAL</strong></td>
<td>Specifies the number of journal blocks to be written to the journal file before the system writes a dummy segment record to the journal file</td>
</tr>
<tr>
<td><strong>JOURNAL FRAGMENT INTERVAL</strong></td>
<td>Directs the system to defer journal I/O based on the number of active transactions running in the system</td>
</tr>
<tr>
<td><strong>QUEUE JOURNAL</strong></td>
<td>Specifies whether the system writes after images of queue records to the journal file</td>
</tr>
</tbody>
</table>

**Operating Environment Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Identify the environment in which the system executes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CVNUMBER</strong></td>
<td>Identifies the system to the CA IDMS SVC (z/OS, z/VSE systems only)</td>
</tr>
<tr>
<td><strong>DESCRIPTION CODES</strong></td>
<td>Specifies operator-message description codes (z/OS systems only)</td>
</tr>
<tr>
<td><strong>EVAL</strong></td>
<td>Controls date processing</td>
</tr>
<tr>
<td><strong>EVAL BASE YEAR</strong></td>
<td>Specifies the base year to be used by EVAL when doing built-in functions DATEDIFF and DATEOFF</td>
</tr>
<tr>
<td><strong>EVAL CENTURY VALIDATION</strong></td>
<td>Indicates whether century values are to be validated by EVAL when processing built-in functions that accept 4-digit years, such as GOODDATEX</td>
</tr>
<tr>
<td><strong>EVAL HIGH CENTURY</strong></td>
<td>Specifies the highest century value that EVAL is to consider valid when processing built-in functions that accept 4-digit years, such as GOODDATEX</td>
</tr>
<tr>
<td><strong>EVAL LOW CENTURY</strong></td>
<td>Specifies the lowest century value that EVAL is to consider valid when processing built-in functions that accept 4-digit years, such as GOODDATEX</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GENERATION IDENTIFICATION</td>
<td>Provides an identifier for the system options table</td>
</tr>
<tr>
<td>MESSAGE RETENTION</td>
<td>Specifies the time period that system retains messages generated by the SEND command</td>
</tr>
<tr>
<td>MULTIPLE SIGNONS</td>
<td>Specifies whether the same user-ID can be signed on to multiple interactive terminals simultaneously.</td>
</tr>
<tr>
<td>OPERATING SYSTEM</td>
<td>Identifies the host operating system</td>
</tr>
<tr>
<td>PAGE RELEASE</td>
<td>Invokes virtual storage operating system services to free real page frames (z/VSE systems only)</td>
</tr>
<tr>
<td>ROUTE CODES</td>
<td>Specifies operator-message routing codes (z/OS systems only)</td>
</tr>
<tr>
<td>SVC</td>
<td>Identifies the CA IDMS SVC used for communication between the system and programs executing outside the system region/partition (z/OS and z/VSE systems only)</td>
</tr>
<tr>
<td>SYSCTL</td>
<td>Defines the system control file used by programs executing outside the system region/partition</td>
</tr>
<tr>
<td>SYSTEM ID</td>
<td>Specifies the nodename of the system</td>
</tr>
<tr>
<td>Print Control Parameters</td>
<td>Control printing options</td>
</tr>
<tr>
<td>OVERRIDING REPORT LINE LENGTH</td>
<td>Specifies the line length to be used for all reports generated within a DC/UCF system</td>
</tr>
<tr>
<td>PRINT KEY</td>
<td>Identifies the default control key used to print the contents of a terminal screen</td>
</tr>
<tr>
<td>PRINTER CHECKPOINT</td>
<td>Specifies the frequency with which the system writes checkpoints for each report as it is printed</td>
</tr>
<tr>
<td>PRINTER CONTROL</td>
<td>Specifies the printer form feed options</td>
</tr>
<tr>
<td>Program Management Parameters</td>
<td>Control program loading and execution</td>
</tr>
<tr>
<td>LOADLIST</td>
<td>Identifies the default load list used by the system</td>
</tr>
<tr>
<td>MULTIPLE ENCLAVE</td>
<td>Specify enclave sharing for the system</td>
</tr>
<tr>
<td>NEW COPY</td>
<td>Determines how the system handles attempts to load deleted programs</td>
</tr>
<tr>
<td>PROGRAM POOL</td>
<td>Specifies the size of the standard program pool for programs that use 24-bit addressing</td>
</tr>
<tr>
<td>REENTRANT POOL</td>
<td>Specifies the size of the program pool for reentrant programs that use 24-bit addressing</td>
</tr>
<tr>
<td>UNDEFINED PROGRAM COUNT</td>
<td>Enables automatic definition of programs at runtime</td>
</tr>
<tr>
<td></td>
<td>Specifies the size of the standard program pool for programs that use 31-bit addressing</td>
</tr>
<tr>
<td>Parameters</td>
<td>Descriptions</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>XA PROGRAM POOL</td>
<td>Specifies the size of the program pool for reentrant programs that use 31-bit addressing</td>
</tr>
<tr>
<td></td>
<td>Database Locking Parameters Allocate initial lock storage and determine when locks are maintained</td>
</tr>
<tr>
<td>RETRIEVAL LOCK /NOLOCK</td>
<td>Specifies whether the system maintains locks for records in areas accessed in shared retrieval usage mode</td>
</tr>
<tr>
<td>SYSLOCKS</td>
<td>Specifies an estimate for the maximum number of locks that will be held concurrently within the system</td>
</tr>
<tr>
<td></td>
<td>Resource Management Parameters Control the allocation of system resources</td>
</tr>
<tr>
<td>AREA ACQUISITION THRESHOLD</td>
<td>Specifies whether the system accumulates area locks when attempting to ready multiple database areas for a single database transaction</td>
</tr>
<tr>
<td>DPE COUNT</td>
<td>Specifies the number of elements available to prevent tasks from deadlocking when acquiring resources</td>
</tr>
<tr>
<td>ECB LIST</td>
<td>Specifies the size of the event control block list used to synchronize events between the DC/UCF system and the host operating system</td>
</tr>
<tr>
<td>CALL/DBIO /LOCK /STORAGE LIMIT</td>
<td>Establishes limits for resources used by individual tasks</td>
</tr>
<tr>
<td>LIMITS FOR ONLINE</td>
<td>Specifies whether the system enforces limits on task resource usage</td>
</tr>
<tr>
<td>QUEUE RETENTION</td>
<td>Specifies the time period that the system retains queues that are created dynamically</td>
</tr>
<tr>
<td>RCE COUNT</td>
<td>Specifies the number of resource control elements available to all tasks</td>
</tr>
<tr>
<td>REPORT RETENTION</td>
<td>Specifies the amount of time the system retains reports in the queue area</td>
</tr>
<tr>
<td>RLE COUNT</td>
<td>Specifies the number of resource link elements available to all tasks</td>
</tr>
<tr>
<td>System Run Unit Management Parameters</td>
<td>Control the execution of run units that access the database and data dictionary</td>
</tr>
</tbody>
</table>

26-Dec-2017
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXIMUM ERUS</td>
<td>Specifies the maximum number of external request units the system can service concurrently</td>
</tr>
<tr>
<td>RUNUNITS FOR LOADER/</td>
<td>Predefines system run units</td>
</tr>
<tr>
<td>MSGDICT /QUEUE/</td>
<td></td>
</tr>
<tr>
<td>SECURITY /SIGNON /SYSTEM /DEST</td>
<td></td>
</tr>
</tbody>
</table>

**Storage Management Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSHION</td>
<td>Specifies the amount of storage in the primary storage pool the system reserves for use by tasks that are already executing</td>
</tr>
<tr>
<td>CWA SIZE</td>
<td>Specifies the size of the common work area available to all tasks</td>
</tr>
<tr>
<td>RELOCATEABLE THRESHOLD</td>
<td>Specifies the point at which the system writes relocatable storage to the scratch area</td>
</tr>
<tr>
<td>SCRATCH IN STORAGE</td>
<td>Enables storage allocation from the operating system for scratch processing and optionally its dynamic extension.</td>
</tr>
<tr>
<td>STORAGE POOL</td>
<td>Specifies the size of the primary storage pool</td>
</tr>
<tr>
<td>XA STORAGE POOL</td>
<td>Specifies the size of the 31-bit storage pool</td>
</tr>
</tbody>
</table>

**Storage Protection Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROTECT /NOPROTECT</td>
<td>Enables programs to use storage protection</td>
</tr>
<tr>
<td>STORAGE KEY</td>
<td>Identifies the alternate storage protect key</td>
</tr>
</tbody>
</table>

**System Monitoring Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABRU SNAP /NOSNAP</td>
<td>Specifies whether the system writes snap dumps to the system log for abended external request units</td>
</tr>
<tr>
<td>DEBUG MESSAGE BUFFERS</td>
<td>Specifies the number of buffers used by the CA IDMS online debugger</td>
</tr>
<tr>
<td>DUMP /NODUMP</td>
<td>Determines the conditions under which the system writes a memory dump</td>
</tr>
</tbody>
</table>
Parameters | Descriptions
--- | ---
LOG | Specifies the file assignment for the DC/UCF system log
STATISTICS | Determines the types of statistics collected and the frequency with which statistics are written to the system log
SYSTRACE | Enables the system trace facility
USERTRACE | Enables the user trace facility

Task Management Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>System Generation Override</th>
<th>Startup Override</th>
<th>Runtime Override</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXIMUM TASKS</td>
<td>Specifies the maximum number of online tasks the system can service concurrently</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ON COMMIT</td>
<td>Specifies options that control commit behavior</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ON ROLLBACK CONTINUE</td>
<td>Specifies options that control rollback behavior</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STACKSIZE</td>
<td>Specifies the size of the storage stack in each task control element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRANSACTION SHARING</td>
<td>Specifies the default transaction sharing option for all tasks within the system</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Overriding SYSTEM Statement Parameters

You can override certain SYSTEM statement parameters in one or more of the following ways:

- **During system generation** with LTERM, and TASK statement parameters. The TASK statement is described in TASK Statement (see page 259). The LTERM statement is described in LTERM Statement (see page 290).

- **At system startup** in response to prompts issued at the operator's console. The prompts are governed by #DCPARM macro specifications. The #DCPARM macro is described in the CA IDMS System Operations section.

- **At runtime** with DCMT and DCUF commands. DCMT and DCUF commands are described in the CA IDMS System Tasks and Operator Commands section.

The following table lists the SYSTEM statement parameters that can be overridden along with the applicable overrides.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>System Generation Override</th>
<th>Startup Override</th>
<th>Runtime Override</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSHION</td>
<td>CUSH=</td>
<td></td>
<td>DCMT VARY STORAGE POOL 0</td>
</tr>
<tr>
<td>CVNUMBER</td>
<td>CVNUM=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPE COUNT</td>
<td>DPECOUNT=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEADLOCK DETECTION INTERVAL</td>
<td></td>
<td></td>
<td>DCMT VARY DEADLOCK</td>
</tr>
<tr>
<td>DUMP/NODUMP</td>
<td>DUMP/NODUMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECB LIST</td>
<td>ECBLIST=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTERNAL WAIT</td>
<td>TASK statement EXTERNAL WAIT</td>
<td>Extwait=</td>
<td>DCMT VARY TASK EXTERNAL WAIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DCMT VARY DYNAMIC TASK EXTERNAL WAIT</td>
</tr>
<tr>
<td>INACTIVE INTERVAL</td>
<td>TASK statement INACTIVE INTERVAL</td>
<td>Inactint=</td>
<td>DCMT VARY TIME STALL</td>
</tr>
<tr>
<td>INTERNAL WAIT</td>
<td>INWAIT=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JOURNAL/NOJOURNAL RETRIEVAL</td>
<td></td>
<td>JOURRET/ NOJOURRET</td>
<td></td>
</tr>
<tr>
<td>JOURNAL FRAGMENT INTERVAL</td>
<td></td>
<td></td>
<td>DCMT VARY JOURNAL</td>
</tr>
<tr>
<td>JOURNAL TRANSACTION LEVEL</td>
<td></td>
<td></td>
<td>DCMT VARY JOURNAL</td>
</tr>
<tr>
<td>LIMIT FOR ONLINE</td>
<td>TASK statement LIMITs</td>
<td></td>
<td>DCMT VARY TASK LIMITs</td>
</tr>
<tr>
<td>LIMITS FOR ONLINE</td>
<td></td>
<td></td>
<td>DCMT VARY LIMITS</td>
</tr>
<tr>
<td>LOADLIST</td>
<td></td>
<td></td>
<td>DCUF SET LOADLIST</td>
</tr>
<tr>
<td>MAXIMUM ERUS</td>
<td>MAXERUS=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM TASKS</td>
<td>MAXTASK=</td>
<td></td>
<td>DCMT VARY ACTIVE TASK MAX TASK</td>
</tr>
<tr>
<td>MULTIPLE ENCLAVE</td>
<td>PROGRAM statement MULTIPLE ENCLAVE</td>
<td></td>
<td>DCMT VARY PROGRAM MULTIPLE ENCLAVE ON/OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DCMT VARY DYNAMIC PROGRAM MULTIPLE ENCLAVE ON/OFF</td>
</tr>
<tr>
<td>PRINT KEY</td>
<td>TASK statement PRINT KEY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>System Generation Override</td>
<td>Startup Override</td>
<td>Runtime Override</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------</td>
<td>------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>PRINTER CHECKPOINT</td>
<td>LTERM statement PRINTER CHECKPOINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRINTER CONTROL</td>
<td>LTERM statement PRINTER CONTROL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGRAM POOL</td>
<td>PROGPOOL=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROTECT/ NOPROTECT</td>
<td>PROTECT/ NOPROTECT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUIESCE WAIT</td>
<td>TASK statement QUIESCE WAIT</td>
<td>DCMT VARY TASK</td>
<td>DCMT VARY DYNAMIC TASK QUIESCE WAIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE WAIT</td>
<td>QUIESCE WAIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DCMT VARY TIME QUIESCE WAIT</td>
</tr>
<tr>
<td>RECOVERY WAIT</td>
<td></td>
<td>DCMT VARY TIME</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECOVERY WAIT</td>
<td></td>
</tr>
<tr>
<td>RCE COUNT</td>
<td>RCECOUNT=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESOURCE TIMEOUT</td>
<td>TASK statement RESOURCE TIMEOUT</td>
<td>RESOURCENET=</td>
<td>DCMT VARY TIME RESOURCE INTERVAL</td>
</tr>
<tr>
<td>INTERVAL</td>
<td>INTERVAL</td>
<td></td>
<td>DCMT VARY TASK RESOURCE INTERVAL</td>
</tr>
<tr>
<td>RESOURCE TIMEOUT</td>
<td>TASK statement RESOURCE TIMEOUT</td>
<td>RESOURCENET=</td>
<td>DCMT VARY TIME RESOURCE INTERVAL</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>PROGRAM</td>
<td>RESOURCENET=</td>
<td>DCMT VARY TASK RESOURCE PROGRAM</td>
</tr>
<tr>
<td>RUNAWAY INTERVAL</td>
<td>RUNAWAY=</td>
<td></td>
<td>DCMT VARY TIME RUNAWAY</td>
</tr>
<tr>
<td>STACKSIZE</td>
<td>STACKSIZ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATISTICS INTERVAL</td>
<td></td>
<td></td>
<td>DCMT VARY STATISTICS INTERVAL</td>
</tr>
<tr>
<td>STATISTICS TASK</td>
<td></td>
<td></td>
<td>DCMT VARY STATISTICS TRANSACTION</td>
</tr>
<tr>
<td>TRANSACTION /NOTRANSACTION</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STORAGE POOL</td>
<td>STGPOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSLOCKS</td>
<td>SYSLOCKS=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSTRACE</td>
<td>SYSTRACE/ NOSYSTRACE/ SYSTRACENUM=</td>
<td></td>
<td>DCMT VARY SYSTRACE</td>
</tr>
<tr>
<td>TICKER INTERVAL</td>
<td>TICKINT=</td>
<td></td>
<td>DCMT VARY TIME TIMER</td>
</tr>
</tbody>
</table>
SYSTEM Statement Syntax Rules

ADD/MODIFY/DELETE and DISPLAY/PUNCH syntax for the system entity type are presented in this section. Syntax rules follow the presentation of both types of syntax. Options that apply to only one verb are noted in the rules. Syntax rules presented in are not repeated unless special considerations apply.

Authorization

<table>
<thead>
<tr>
<th>To perform this operation</th>
<th>You need this authority or privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a system definition</td>
<td>DCADMIN authority or</td>
</tr>
<tr>
<td>Note: If you change an ADD operation to a MODIFY, the authority required is the same as for a MODIFY operation.</td>
<td>CREATE on the resource type SYSTEM and the system name of the system you are creating</td>
</tr>
<tr>
<td>Modify a system definition</td>
<td>DCADMIN authority or</td>
</tr>
<tr>
<td></td>
<td>ALTER on the system name of the system definition you are modifying.</td>
</tr>
<tr>
<td></td>
<td>If you are modifying the name of the system (SYSTEM ID parameter of the SYSTEM statement), you must also have CREATE authority on the new system name</td>
</tr>
<tr>
<td>Delete a system definition</td>
<td>DCADMIN authority or</td>
</tr>
<tr>
<td></td>
<td>DROP on the system name of the system definition you are deleting</td>
</tr>
<tr>
<td>Display or punch a system definition</td>
<td>DCADMIN authority or</td>
</tr>
<tr>
<td></td>
<td>DISPLAY on the system name of the system you want to display</td>
</tr>
</tbody>
</table>

SYSTEM Statement Syntax

```
SYStem dc/ucf-version-number
```

```
ADD
MODify
DELeete
```

```
SYSTEM ID is system-name
```
RETRieval LOCK
NOLock

RLE count is rle-count
DEFault θ

RLE AUTotune is OFF ON

ROUTE codes = ( os-routing-code )

RUNaway interval is 10 runaway-task-time
OFF

RUNUnits for LOAder
MSGdict
QUEue
SECurity
SIGnon
SYStem/dest

SCRatch in STOrage is NO
YES

SNAp SYStem is ON OFF

SNAp SYStem PHOto is ON OFF

SNAp TASk is ON OFF

SNAp TASk TRAce is ON OFF LIMit is 1000
'<snap-trace-limit>'
OFF

SNAp TASk PHOto is ON OFF

STAcksize is 1200
'tce-stack-size'

STATistics

INTerval statistics-interval
OFF θ

ROLL interval-roll-parms

NOROLL
LINE
NOLine
TASk
NOTask

STOrage KEY is 9
'storage-protect-key'

STOrage POOl is 50
'storage-pool-size'
Expansion of print-ctrl-options

SCS FORMfeed is  FF-NL

ON REPORT START (FORMfeed for NON-native print is  INSert
SUPpress NO-Insert)

FORMfeed for NATive print is  INSert
SUPress NO-Insert

SCS CR-LF is  ON
OFF

ON REPORT END (FORMfeed for NON-native print is  INSert
NO-Insert)

FORMfeed for NATive print is  INSert
NO-Insert

Expansion of interval-roll-parms

TIME is  24:00
interval-roll-time

FREquency is  1
day-frequency

Expansion of scratch-storage-parms

LOCATION XA
ANY
64-bit

PRIMARY extent is  prim-size-with-unit
DEFAULT

SECONDary extent is  sec-size-with-unit
DEFAULT

LIMIT is  limit-with-unit
DEFAULT

Expansion of interval-roll-parms

TIME is  24:00
interval-roll-time

FREquency is  1
day-frequency

DISPLAY/PUNCH SYSTEM Statement Syntax

DISPLAY  ALL
SYStem dc/ucf-version-number

PUNCH  ALL
CURrent SYStem

WITH
ALSo WITH
WITHOUT

DETAils
HISTORY
ALL
NONE

AS  COMments
SYNtax

TO  module-specification
SYSpch
SYSTEM Statement Parameters Information

**SY Stem dc/ucf-version-number**
Identifies the DC/UCF system being added, modified, deleted, displayed, or punched. *Dc/ucf-version-number* must be an integer in the range 1 through 9,999. All systems defined in the data dictionary with the system generation compiler are automatically assigned the name DCSYSTEM. **Note:** The DBA may elect to reserve consecutive SYSTEM (and CV) numbers for use in cloning CV’s. For more information about cloning CV’s, see *Administrating CA IDMS System Operations* (https://docops.ca.com/display/IDMS19/Administrating+CA+IDMS+System+Operations).

- **SYS Tem ID is system-name**
  Specifies the name (nodename) by which the DC/UCF system is known to other nodes in the DC/UCF communications network. *System-name* must be a one- through eight-character name that is unique throughout the DC/UCF communications network and must be the first parameter you specify after the ADD SYSTEM statement.
  The default is SYSTnnnn where nnnn is the value specified as *dc-ucf-version* in the ADD SYSTEM statement.

  **Note:** *System-name* is the name of the DC/UCF system you name in the CA IDMS Security DDL CREATE RESOURCE SYSTEM statement.

- **ABEnd storage is abend-storage-size**
  Specifies the amount of storage, in fullwords, available to the system for processing abends in the event of a task control element (TCE) stack overflow (for example, as a result of a recursive abend).
  The STACKSIZE value will be used instead of this value when it is larger.
  *Abend-storage-size* must be an integer in the range 0 through 32,767. The default is 200.

  **Note:** Abend storage is a single-threaded resource.

- **ABRu SNAP/NOSnap**
  Specifies whether the system writes a snap dump to the log when an external request unit terminates abnormally.

- **AREa acquisition THReshold is count**
  Specifies the point at which, during ready processing, the system begins to accumulate area locks for a database transaction. This value only applies if the system is readying multiple areas at one time.
  *Count* must be an integer in the range 1 through 32,767.

  - **OFF**
    Directs the system not to accumulate area locks until it can acquire all areas needed by a database transaction.
    OFF is the default.
- **RETry count**
  Defines a limit on the number of times the system continues trying to gain access to all areas without accumulating area locks.
  `Count` must be an integer in the range 1 through 32,767.

- **RETry FOREver**
  Directs the system to continue to try to gain access to all needed areas until it successfully acquires all areas or until operating system resource and time limits are exceeded. You should specify the default of FOREVER unless experience shows that a transaction is not gaining access to areas as needed.
  FOREVER is the default.
  OFF RETRY FOREVER is the default when you omit the AREA ACQUISITION THRESHOLD parameter from the SYSTEM statement.

- **CHKuser tasks IS check-user-task-count**
  For z/OS and z/VSE operating systems only, specifies the number of subtasks to be attached by the system at runtime to detect abnormally terminated batch external request units.
  `Check-user-task-count` must be an integer in the range 0 through 255.
  Specify a value equal to the number of CA IDMS batch jobs that execute concurrently in the DC/UCF system.
  0, the default, directs the system not to use the check-user mechanism to detect abends.

- **CUSHion is storage-cushion-size**
  Specifies the amount of storage, in 1K bytes, to be reserved in the primary storage pool for use by tasks that are already executing.
  `Storage-cushion-size` must be an integer in the range 0 through 16,383. The default is 0.
  Note: For more information about relocatable storage, see Storage Pools (see page 70).

- **CVNumber is cvnumber**
  Identifies the DC/UCF system to the CA IDMS SVC. Additionally, if a SYSCTL file is defined, the value specified by the CVNUMBER parameter is written to the SYSCTL file at system startup.
  `Cvnumber` must be an integer in the range 0 through 255. The default is 0. Note: For more information about the SYSCTL file, see the CA IDMS System Operations section.

  Up to 256 DC/UCF systems can execute concurrently and independently in a single machine. Systems executing concurrently must be defined with unique CVNUMBER values, even if the systems use different SVCs.
  z/VM systems: A DC/UCF system running in a CMS virtual machine must be defined with CVNUMBER IS 0.

- **CWA size is cwa-size**
  Specifies the size, in bytes, of the common work area (CWA). The CWA is a block of shared user storage that is available to all user tasks. A program accesses the CWA by issuing a GET STORAGE request with a storage id of CWA. Use of the CWA facilitates communication among tasks.
  `Cwa-size` must be a positive integer in the range 0 through 2,147,483,647; the default is 0.

- **DEAdlock DETection INTerval is deadlock-check-interval**
  Specifies the amount of time, in wall clock seconds, that elapses before the system searches for deadlocked tasks.
  `Deadlock-check-interval` must be an integer in the range 0 through 1,200 and must be:
  - Less than the value specified in the INACTIVE INTERVAL parameter of the SYSTEM statement
- Greater than or equal to the value specified in the TICKER INTERVAL parameter of the SYSTEM statement

The default is 5.

- **DEBug MEssage BUffers is**
  Specifies the number of debug message buffers to be used by the DEBUG task.

  - **number-of-buffers**
    Specifies the number of debug message buffers to be used by the DEBUG task. If this parameter is omitted then the default number of debug message buffers is 5. Increasing the debug message buffers above the default of 5 will increase the amount of storage needed to process the DEBUG task messages. The value specified can be in a range of 5 through 30.

- **DEBug MEssage BUffers is buffer-count**
  Specifies the number of message buffers used by the CA IDMS online debugger. Increasing the number of message buffers enables the debugger to display more symbols and list more information. However, the larger the value you specify, the more storage is needed to execute the debugger.

  **Limits:** 5 - 30  
  **Default:** 5 (buffers)

- **DEscription codes =**
  For z/OS systems only, specifies operator-message description codes to be passed to the DESC parameter for write-to-operator (WTO) macros issued by the system.

  - **(Os-description-code)**
    Specifies one or more operator-message description codes, as described in the applicable operating system supervisor services and macro instructions. **Os-description-code** must be an integer in the range 1 through 16. Multiple description codes must be separated by blanks or commas.

  - **0**
    Clears any previously established operator-message description codes.

- **DPE AUTotune is**
  Indicates if CA IDMS should automatically tune the DPE count value using execution statistics.

  - **OFF**
    Turns off automatic tuning of DPE count.

  - **ON**
    Turns on automatic tuning of DPE count.

  **Default:** OFF

- **DPE count is**
  Specifies the number of deadlock prevention elements (DPEs) to allocate at system startup.

  - **dpe-count**
    Directs the system to allocate the specified number of DPEs at startup. **Dpe-count** must be an integer in the range 1 through 32,767.
- **DEFault/0**
  Directs the system generation compiler to calculate the number of DPEs to allocate. DEFAULT and 0 are synonyms and can be used interchangeably. DEFAULT is the default when you omit the DPE COUNT parameter from the SYSTEM statement.

- **DUMP**
  Directs the system to write a memory dump for all system abend codes (that is, 39nn). In most cases, a dump is not necessary because the abend code itself provides sufficient information to determine the cause of the abend.

- **NODump**
  Directs the system to write a memory dump only for system abend codes that do not provide sufficient information to determine the cause of the abend. NODUMP is the default when you specify neither DUMP nor NODUMP in the SYSTEM statement. The DUMP/NODUMP specification does not affect task or external request abend requests issued from DML programs.

- **ECB list is**
  Specifies the amount of storage to allocate for the external event control block (ECB) list.

  - **ecb-list-size**
    Specifies the size, in words, of the amount of storage to allocate for the ECB list. Ecb-list-size must be an integer in the range 0 through 32,767.

  - **DEFault/0**
    Directs the system generation compiler to calculate the size of the ECB list. The compiler calculates the size to equal two times the number of TCEs. DEFAULT and 0 are synonyms and can be used interchangeably. DEFAULT is the default when you omit the DPE COUNT parameter from the SYSTEM statement.

- **EVAl BAse YEAr is**
  Specifies the base year to be used by EVAL when doing built-in functions DATEDIFF and DATEOFF.

  - **base-year**
    Specifies the EVAL base year when doing built-in functions DATEDIFF and DATEOFF to determine the century associated with the requested date. If this parameter is omitted then the default base year is 68 meaning that years 69 and greater are in the 20th century, and years 68 and lower are in the 21th century. The value specified can be in a range of 1 through 99.

- **EVAl CENtury VAli dation is**
  Indicates whether century values are to be validated by EVAL when processing built-in functions that accept 4-digit years, such as GOODDATEX.

  - **ON**
    Indicates that EVAL is to validate century values.

  - **OFF**
    OFF indicates that EVAL is not to validation century values. The default for new systems is OFF.
- **EVAL HIGh CENtury is**
  Specifies the highest century value that EVAL is to consider valid when processing built-in functions that accept 4-digit years, such as GOODDATEX.

  - **high-century**
    Specifies the high century to be used by EVAL when doing built-in functions GOODDATEX century validation.
    If this parameter is omitted then the default high century is 20.
    The value specified can be in a range of 1 through 99. This parameter has no effect if the setting for EVAL CENTURY VALIDATION is OFF.

- **EVAL LOW CENtury is**
  Specifies the lowest century value that EVAL is to consider valid when processing built-in functions that accept 4-digit years, such as GOODDATEX.

  - **low-century**
    Specifies the low century to be used by EVAL when doing built-in functions GOODDATEX century validation.
    If this parameter is omitted then the default low century is 19.
    The value specified can be in a range of 1 through 99. This parameter has no effect if the setting for EVAL CENTURY VALIDATION is OFF.

- **E XTernal wait is**
  Specifies the amount of time the system is to wait for the next request to be issued within an external request unit before abnormally terminating the task. For more information about external user sessions, see External user sessions (see page 63).

- **GENeration identification is generation-id**
  Specifies a unique identifier for the system. The identifier is included in the system options table, which is built in the DC/UCF region/partition during system startup. In a memory dump, the identifier appears at the beginning of the system options table.
  Generation-id must be a one- through eight-character alphanumeric value.

- **INActive interval is**
  Specifies the amount of time the system permits an online task or an external user session to wait for a resource before abnormally terminating the task or session.

  - **inactive-wait-time**
    Specifies the inactive interval in wall-clock seconds.
    Inactive-wait-time must be an integer in the range 1 through 32,767.

  - **OFF**
    Directs the system not to terminate tasks or external user sessions based on an inactive interval.
    OFF is the default when you omit the INACTIVE INTERVAL parameter from the SYSTEM statement.

- **IN Ternal wait is**
  Specifies the amount of time the system permits an external request unit to wait for a resource before abnormally terminating the request unit.
**Important:** The INTERNAL WAIT parameter is not functional. Its function is performed by the INACTIVE INTERVAL parameter of the TASK statement (see page 261) for task RHDCNP3S. You can specify wait times for individual batch or CICS run-units by defining TASK statements where the task name is the first batch program that establishes communication with the CV for the run-unit or the CICS task code.

- **JOURNAL retrieval**
  Directs the system to write BGIN and ENDJ checkpoints to the journal file for run units that perform no updates (that is, for retrieval run units). JOURNAL RETRIEVAL is the default.

  **Note:** For the purposes of this parameter, a run unit that readies an area in an update usage mode but that does not update the database is considered a retrieval run unit.

- **NOJournal retrieval**
  Directs the system not to write BGIN and ENDJ checkpoints for retrieval run units.

- **JOURNAL FRAGMENT INTERVAL** is `fragment-interval-number`
  Specifies the maximum number of journal blocks to write to the journal file before the system writes a dummy segment (DSEG) record to the journal file.
  `Fragment-interval` must be an integer in the range 100 through 32,767.
  0, the default, turns off the journal fragment interval.

- **JOURNAL TRANSACTION LEVEL** is `transaction-level-number`
  Specifies the number of active transactions that must be running in a DC/UCF system to defer the writing of a journal block.
  `Transaction-level-number` must be an integer in the range 1 through 100.
  If the number of active transactions is less than the value specified, the system performs journal I/O each time a transaction checkpoint is written to the journal buffer page.
  0, the default, directs the system to write a journal buffer to the journal block each time a transaction checkpoint is taken or when the journal buffer is full.

- **LIMIT for tasks is `limit`**
  Defines limits for task resource usage. `Limit` specifies the limit for each resource.
  0, the default, directs the system not to limit task usage of the named resource.
  You can code two LIMIT parameters for each resource: one for online tasks and one for ERUS tasks. Specify only one resource in each LIMIT parameter.

  - **CALL**
    Limits the number of system service calls (for example, #GETSTG, #LOAD, or OBTAIN CALC) a task can issue. When you specify CALL, `limit` must be an integer in the range 0 through 2,147,483,647.

  **Note:** For performance purposes, external limits are only checked after every 100 calls. Therefore, if the SYSTEM statement specifies CALL LIMIT FOR EXTERNAL TASK IS 150, a batch program could issue 200 DML calls before the system determines that the limit is exceeded.
- **DBIo**
  Limits the number of database I/O operations (that is, reads and writes) that are performed for a task. When you specify DBIO, \textit{limit} must be an integer in the range 0 through 2,147,483,647.

- **LOCk**
  Limits the number of record locks that a transaction can concurrently hold. When LOCk is specified, \textit{limit-n} must be an integer in the range 0 through 2,147,483,647.
  \textbf{Note:} For more information about record locks, see Database Locks (see page 53).

- **STOrage**
  Limits the amount of storage that a task holds at one time. The limit is expressed in \textit{1K bytes}. When you specify STOrage, \textit{limit} must be an integer in the range 0 through 16,383.

- **ONLine**
  Indicates the specified limit applies to all tasks defined to the system either during system generation or at runtime.

- **EXTernal**
  Indicates the specified limit applies to all tasks associated with external request units (that is, ERUS tasks).

- **LIMits for ... are**
  Controls the enforcement of limits on task resource usage. You can code one LIMIT parameter for each system.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{note.png}
\caption{Note: Limits on task resource usage can be enforced only if task statistics are being collected.}
\end{figure}

- **LOAdlist**
  Identifies the default load list to be used by the system when searching for programs.

  - **SYSLOAD**
    Directs the system to use the predefined SYSLOAD load list. SYSLOAD is the default when you omit the LOADLIST parameter from the SYSTEM statement.

- **\textit{load-list-name}**
  Directs the system to use a user-defined load list. \textit{Load-list-name} must be the name of a load list defined by means of the system generation LOADLIST statement.

- **LOG**
  Defines the system log file.

  - **DATabase**
    Directs the system log to the log area (DDLDCLOG) of the data dictionary.

  - **FILE1**
    Directs the system log to one or two sequential files and identifies the first (or only) file.

  - **\textit{ddname-1}**
    Specifies the z/OS or z/VM ddname of the first log file. The default is CDMSLOGA.
filename-1
Specifies the z/VSE filename of the first log file. The default is CDMSLGA.

COUNT1 record-count-1
Specifies the maximum number of records to write to the first log file. When the specified number of records is reached, the system closes the file and opens the second log file, if one is defined. If a second log file is not defined, the system reopens the first log file.
Record-count-1 can be any positive integer.
The COUNT1 parameter is required when the log is being directed to sequential disk files.

FILE2
Identifies the second of two sequential log files.

dname-2
Specifies the z/OS or z/VM ddname of the second log file.

filename-2
Specifies the z/VSE filename of the second log file.

COUNT2 record-count-2
Specifies the maximum number of records to write to the alternate log file. When the specified number of records is reached, the system closes the alternate log file and opens the first log file.
Record-count-2 can be any positive integer.
The COUNT2 parameter is required when the log is directed to sequential disk files.

TYPe device-type
For z/VSE systems only, specifies the device type of the sequential file to which the log is directed. For more information about how the system logs errors, see the CA IDMS System Operations section.

MAXimum ERUs is external-request-unit-count
Specifies the number of EREs to allocate at system startup. External-request-unit-count must be an integer in the range 0 through 1000.

Default: 0
Specify a value equal to the total number of external request units you allow to execute concurrently in the DC/UCF system plus 10% - 20% of the total number of 3270-type terminals defined to the DC/UCF system.

Note: When up to 255 EREs are requested, a corresponding TCE/DCE pair will be allocated for each one during startup. If more than 255 ERES are requested, a maximum of 255 TCE/DCE pairs are allocated for these external request units.

MAXimum TASks is task-count
Specifies the maximum number of application and transient system tasks that can be active at a given time. The system generation compiler uses this value when calculating the number of TCEs to allocate at system startup.
Task-count must be an integer in the range 0 through 255. The default is 0.

Note: For more information about Task-Count, see Task Resource Usage (see page 54), and also the section “System Performance” in the System Operations section.

- MESSAGE RETENTION is
  Specifies the amount of time the system is to retain a message that was created by the SEND command in the queue area. Messages that exceed the specified retention period are deleted automatically at system startup.

  - message-retention-period
    Specifies the message retention period in days.
    Limit: 0 - 255 (0 is synonymous with 1; 255 is synonymous with FOREVER)
    Default 7 (days)

- FOREVER | OFF
  Directs the system not to delete reports based on a retention period. FOREVER and OFF are synonyms and can be used interchangeably.

- MULTIPLE ENCLAVE is
  Specifies whether the system allows the same LE process/enclave to be used for multiple COBOL programs in the same task.

  - ON
    Specifies that the same process/enclave can be used for multiple programs.

  - OFF
    Specifies that the same process/enclave cannot be used for multiple programs. This is the default.

- MULTIPLE SIGNON is
  Specifies if the same user-ID can be signed on to multiple interactive terminals simultaneously.

  - YES
    Specifies the same user-ID can be signed on to multiple interactive terminals simultaneously.

  - NO
    Specifies the same user-ID cannot be signed on to multiple interactive terminals simultaneously.
    NO is the default.

- NEW COPY is
  Specifies the action the system is to take when a load request for a program in the load area (DDLDCLOD) of the data dictionary names a program that is deleted from its program pool.

  - MANUAL
    Directs the system not to attempt to load the program until the user issues a DCMT VARY PROGRAM NEW COPY command. The system disables the program and terminates the task or run unit that issued the load request.
    MANUAL is the default when you omit the NEW COPY parameter from the SYSTEM statement.

  - AUTOMATIC
    Directs the system to mark the program for new copy automatically and to reattempt to load the program.
ON COMMIT
Specifies options that control commit behavior. These options apply only to commit operations in which the database session remains active.

Write COMT
Specifies that a COMT journal record should be written.

Write ENDJ
Specifies that an ENDJ journal record should be written.

NEW ID
Specifies that a new local transaction ID should be assigned to the next transaction associated with the database session.

RETain ID
Specifies that the existing local transaction ID should be assigned to the next transaction associated with the database session.

ON ROLLBACK continue
Specifies options that control rollback behavior. These options apply only to rollback operations in which the database session remains active.

RETain ID
Specifies that following a rollback, the current local transaction ID should be assigned to the next transaction associated with the database session.

NEW ID
Specifies that following a rollback, a new local transaction ID should be assigned to the next transaction associated with the database session.

OPERating system is operating-system
Identifies the host operating system under which the DC/UCF system is to run. Operating-system must be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMS</td>
<td>z/VM</td>
</tr>
<tr>
<td>VSE</td>
<td>z/VSE</td>
</tr>
<tr>
<td>MVS</td>
<td>z/OS</td>
</tr>
</tbody>
</table>

If not specified, the operating system defaults to:

- MVS for z/OS and z/VM systems
- VSE for z/VSE systems

PAGe release is
For z/OS and z/VSE systems only, specifies whether the system is to invoke operating system services (that is, PGRLE for z/OS systems, RELPAG for z/VSE systems) to release real page frames associated with virtual pages that are no longer required and whose contents do not need to be saved.
YES
Directs the system to use the page release feature. Real page frames associated with virtual pages that are no longer needed are available to other operating system tasks.

z/VSE systems: When you specify YES, the operating system SUPVR macro must specify PAGEIN=n.

NO
Directs the system not to use the page release feature. z/OS users should specify NO. NO is the default when you omit the PAGE RELEASE parameter from the SYSTEM statement.

PRInt KEY is
Specifies the control key to use to print the contents of a terminal screen.

PAn
Identifies a program attention key. N must be the integer 1, 2, or 3.

PFN
Identifies a program function key. N must be an integer in the range 1 through 24.

OFF
Indicates that no system-wide print-screen key is defined. OFF is the default when you omit the PRINT KEY parameter from the SYSTEM statement.

PRInter CHEckpoint is
Specifies the frequency with which the system is to write checkpoints for each report as the report is printed.

page-count
Directs the system to write printer checkpoints for all active reports. The system writes checkpoints for a given report every page-count pages. If printing is interrupted (for example, by a task or system abend), the system resumes printing the report at the last checkpoint. Page-count must be an integer in the range 0 through 32,767. A value of 0 is synonymous with OFF.

OFF
Directs the system not to write printer checkpoints. The system resumes printing interrupted reports at the beginning of the file. OFF is the default when you omit the PRINTER CHECKPOINT parameter from the SYSTEM statement.

PRInter CONtrol is
Specifies the printer form feed options.

SCS FORMfeed is
Specifies the form feed sequence sent to SCS devices.

FF-NL
Specifies the form feed sequence is "FF" (form feed) followed by "NL" (new line). FF-NL is the default.

FF
Specifies the form feed sequence contains "FF" only.
ON REPORT START
Specifies what happens when a report starts printing.
FORMFEED FOR NATIVE PRINT IS
Specifies the form feed processing for native mode print reports.
FORMFEED FOR NON-NATIVE PRINT IS
Specifies the form feed processing for non-native mode print reports.
Parameters for the two FORMFEED options are:
  INSERT
  A form feed is inserted, if needed.
  SUPPRESS
  The report never starts with a form feed, i.e., if the report starts with a form feed, it is removed.
  NO-INSERT
  No change to the report contents is made.
SCS CR-LF IS
Suppresses the Carriage Return/Line Feed sequence at the beginning of a report transmitted to an SCS printer.

ON REPORT END
Specifies what happens when a report finishes printing.
FORMFEED FOR NATIVE PRINT IS
Specifies the form feed processing for native mode print reports.
FORMFEED FOR NON-NATIVE PRINT IS
Specifies the form feed processing for non-native mode print reports.
Parameters for the two FORMFEED options are:
  INSERT
  A form feed is inserted.
  NO-INSERT
  No form feed is inserted.

OVERRIDING REPORT LINE LENGTH IS REPORT-LINE-LENGTH
Specifies the line length to be used for all reports generated within a DC/UCF system.
Limits: 0 - 255
If this parameter is 0 (zero) or not specified, the line length used is based on the printer terminal's device type. Because specifying a non-zero value impacts all DC/UCF reports, exercise caution if you set this parameter.

PROGRAM POOL IS PROGRAM-POOL-SIZE
Specifies the size, in 1K bytes, of the 24-bit program pool.
Program-pool-size must be an integer in the range 4 through 16,383. The default is 50.

PROTECT
Enables the use of storage protection. Programs for which the system generation PROGRAM statement specifies PROTECT will run with the alternate protect key defined by the STORAGE KEY parameter.

NOPROTECT
Disables the use of storage protection. Programs will run with the primary protect key regardless of the PROGRAM statement PROTECT/NOPROTECT specification. NOPROTECT is the default when you specify neither PROTECT nor NOPROTECT in the SYSTEM statement.
- **QUEUE JOURNAL**
  Specifies whether the system is to write after images of queue records to the journal file.

  **Important!** If you code the QUEUE JOURNAL parameter immediately following the RUNUNITS parameter, the system generation compiler interprets the keyword QUEUE as a continuation of the RUNUNITS parameter.

- **QUEUE RETention**
  Specifies the default time period that the system retains queues that are created dynamically. This value is used only if no retention period is specified when a queue is created. Queues whose retention period has expired are deleted automatically when the system is next started.

  - **queue-retention-period**
    Specifies the number of days that queues are to be retained.
    **Limits:** 0 - 255 (255 is synonymous with FOREVER; 0 is synonymous with 1)

  - **FOREver/OFF**
    Directs the system not to delete queues based on a retention period. FOREVER and OFF are synonyms that you can use interchangeably.

- **QUiesce wait**
  Specifies if the system permits a task to wait for a quiesce operation to terminate; and if waiting is permitted the amount of time the task waits before it is abnormally terminated.

  - **STAll interval**
    Specifies the quiesce wait time for a task is the same as the task's stall interval. This is the default.

  - **quiesce-wait-time**
    Specifies the quiesce wait interval in wall-clock seconds. *quiesce-wait-time* must be an integer in the range 1 through 32,767.

  - **FOREver**
    Directs the system not to terminate tasks based on quiesce wait time.

  - **NOWait**
    Specifies tasks do not wait for quiesce operations to terminate and instead, receive an error indicating an area is unavailable. For navigational DML requests, this results in an error-status value of 'xx66'.

- **RCE count**
  Specifies the number of resource control elements (RCEs) to allocate at system startup.

  - **rce-count**
    Directs the system to allocate the specified number of RCEs at startup. *rce-count* must be an integer in the range 0 through 32,767.
- **DEFault/0**
  Directs the system generation compiler to calculate the number of RCEs to allocate. DEFAULT and 0 are synonyms and are used interchangeably. DEFAULT is the default when you omit the RCE COUNT parameter from the SYSTEM statement.
  **Note:** For more information about RCEs, see Resource Management (see page 54).

- **RCE AUTotune is**
  Indicates if CA IDMS should automatically tune the RCE count value using execution statistics.
  - **OFF**
    Turns off automatic tuning of RCE count.
  - **ON**
    Turns on automatic tuning of RCE count.
    **Default:** OFF

- **REcovery wait is**
  Specifies the amount of time the system permits a task to wait for a resource to be recovered by a failed data sharing group member before the task is abnormally terminated.
  - **NOT ALLOWed**
    Directs the system to immediately cancel the task. NOT ALLOWED is the default.

  - **recovery-wait-time**
    Specifies the recovery wait time in seconds. Recovery-wait-time must be an integer in the range 1 through 32,767. A value of zero is treated as if NOT ALLOWED was specified.

  - **FORever**
    Directs the system to permit a task to wait indefinitely.

- **REEntrant pool is reentrant-pool-size**
  Specifies the size, in 1K bytes, of the 24-bit reentrant pool. Reentrant-pool-size must be an integer in the range 0 through 16,383 and must be large enough to hold the largest reentrant program that will execute under the DC/UCF system. The default is 0.

- **RELocatable threshold is**
  Specifies the point at which the system is to write relocatable storage in storage pool 0 to the scratch area (DDLDCSCR) of the data dictionary.
  - **threshold-percentage**
    Directs the system to write relocatable storage to the scratch area across a pseudo-converse when the amount of used space in the storage pool exceeds the specified threshold. Threshold-percentage must be an integer in the range 0 through 100. A value of 0 is synonymous with YES. A value of 100 is synonymous with NO.

  - **YES**
    Directs the system always to write relocatable storage to the scratch area across a pseudo-converse. YES is the default.
• **NO**
  Directs the system never to write relocatable storage to the scratch area across a pseudo-converse.

• **REPort retention is**
  Specifies the amount of time the system is to retain a report in the queue area. Reports that exceed the specified retention period are deleted automatically at system startup.

  - **report-retention-period**
    Specifies the report retention period in days.
    *Report-retention-period* must be an integer in the range 0 through 255. The default is 7. A value of 255 is synonymous with FOREVER.

• **FOREver/OFF**
  Directs the system not to delete reports based on a retention period. FOREVER and OFF are synonyms and can be used interchangeably.

• **RESourse timeout**
  Controls the resource-timeout mechanism.

  - **INTerval is**
    Specifies the amount of time the system is to wait for input from the terminal operator before invoking the resource timeout program.

  - **resource-timeout-interval**
    Specifies the resource timeout interval in wall-clock seconds.
    *Resource-timeout-interval* must be an integer in the range 0 through 32,767.

  - **FOREver/OFF**
    Disables the resource-timeout mechanism. FOREVER and OFF are synonyms and can be used interchangeably. FOREVER is the default when you omit the RESOURCE TIMEOUT INTERVAL parameter from the SYSTEM statement.

• **PROgram is resource-timeout-prog-name**
  Specifies the program the system invokes to handle the resources owned by an inactive terminal when the resource timeout interval expires.
  *Resource-timeout-prog-name* must be the name of a program included in the system definition. The default is RHDCBYE. RHDCBYE deletes all resources owned by the terminal and returns control to the system.

• **Version version-num**
  Qualifies the named program with a version number. *Version-num* must be an integer in the range 1 through 9,999. The default is 1.
  **Note:** For more information about the resource-timeout mechanism, see *Abend Detection and Timed Functions* (see page 47).

• **RETrieval**
  Specifies whether the system is to maintain locks automatically for records in areas accessed in shared retrieval usage mode.
- **LOCk**
  Directs the system to maintain locks for records in areas accessed in shared retrieval usage mode. Records accessed by retrieval run units are prevented from being updated. However, the maintenance of the additional locks adds overhead.

- **NOLock**
  Directs the system not to maintain locks for records in areas accessed in shared retrieval usage mode.
  NOLOCK is the default when you omit the RETRIEVAL parameter from the SYSTEM statement.

- **RLE count is**
  Specifies the number of resource link elements (RLEs) to allocate at system startup.
  
  - **re-count**
    Directs the system to allocate the specified number of RLEs at startup.
    
    *re-count* must be an integer in the range 0 through 32,767.

  - **DEFault/0**
    Directs the system generation compiler to calculate the number of RLEs to allocate. DEFAULT and 0 are synonyms and are used interchangeably.
    
    DEFAULT is the default when you omit the RLE COUNT parameter from the SYSTEM statement.

- **RLE AUTotune is**
  Indicates if CA IDMS should automatically tune the RLE count value using execution statistics.
  
  - **OFF**
    Turns off automatic tuning of RLE count.

  - **ON**
    Turns on automatic tuning of RLE count.

  Default: OFF

- **ROUte codes =**
  For z/OS systems only, specifies operator-message routing codes to pass to the ROUTCDE parameter for write-to-operator (WTO) macros issued by the system.
  
  - **(Os-routing-code)**
    Specifies one or more operator-message routing codes, as described in the applicable operating system supervisor services and macro instructions.
    
    *Os-routing-code* must be an integer in the range 1 through 16. You must separate multiple routing codes by blanks or commas.

  - **0**
    Clears any previously established operator-message routing codes.

- **RUNaway interval is**
  Specifies the maximum amount of time the system is to permit a task to execute without returning control to the system.
- **runaway-task-time**
  Specifies the runaway interval in wall-clock seconds.  
  *Runaway-task-time* must be an integer in the range 1 through 32,767. The default is 10.

- **OFF**
  Directs the system not to check for runaway tasks.

- **RUNUnits for ... = run-unit-count**
  Specifies the number of predefined run units the system is to initiate at startup to service requests for:
  - Load area processing in the system default dictionary
  - Message area processing
  - Queue area processing
  - Security checking for system-level resources such as tasks, queues, and load modules
  - Signon processing in the user catalog
  - Destination processing in the system default dictionary
  
  *Run-unit-count* must be an integer in the range 0 through 32,767. The default is 1.

- **LOAder**
  Predefines the specified number of run units for load area processing in the default dictionary.

- **MSGdict**
  Predefines the specified number of run units for message area processing.

- **QUEue**
  Predefines the specified number of run units for queue area processing.

- **SECurity**
  Predefines the specified number of run units for security processing on system-level resources.

- **SIGnon**
  Predefines the specified number of run units for signon processing in the user catalog.

- **SYStem/dest**
  Predefines the specified number of run units for destination processing.

- **SCRatch in STOrage is**
  Specifies whether or not scratch information resides in memory.

  - **NO**
    Specifies that the scratch information is not memory-resident.
• YES
  Specifies that the scratch information is memory-resident.

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

  • ANY|XA|64-bit
    Determines the storage location. The storage needed for scratch processing is allocated directly from the operating system and not from the CA IDMS storage pools.
    ANY
    Acquires 64-bit storage, if possible. If the request to allocate 64-bit storage fails, XA storage is acquired.
    XA
    Acquires 31-bit storage.
    64-bit
    Acquires 64-bit storage. If the request to allocate 64-bit storage fails, no attempt to acquire XA storage is done.

    • SCRATCH IN XA STORAGE IS YES is synonymous to SCRATCH IN STORAGE IS YES LOCATION XA.

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

• PRIMARY extent is
  Specifies the primary scratch allocation size.

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

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

• SECONDARY extent is
  Specifies the secondary scratch allocation size.

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

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

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

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

- **SNAP SYStem is**
  Specifies whether to write a system snap dump to the DC/UCF log file. A system snap dump writes a formatted display of the resources allocated to all active tasks.

  - **ON**
    Enables the writing of a system snap dump. This is the default for the ADD SYSTEM statement.

  - **OFF**
    Disables the writing of a system snap dump.

- **SNAP SYStem PHOto is**
  Specifies whether to write a system photo snap to the DC/UCF log file. A system photo snap provides a summary of resources for all active tasks.

  - **ON**
    Enables the writing of a system photo snap. This is the default for the ADD SYSTEM statement.

  - **OFF**
    Disables the writing of a system photo snap.

- **SNAP TASk is**
  Specifies whether to write a task snap dump to the DC/UCF log file. A task snap dump writes a formatted display of the resources allocated to the task being snapped.

  - **ON**
    Enables the writing of a task snap dump. This is the default for the ADD SYSTEM statement.

  - **OFF**
    Disables the writing of a task snap dump.

- **TRAce is ON|OFF|TASk**
  Controls the inclusion of trace information in task snaps.

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

  - **OFF**
    Includes no trace information in a task snap.
- **TASk**
  Includes only trace information for the task for which the snap is being issued.
  
  **Default**: ON

- **LIMit is snap-trace-limit|OFF**
  Limits the number of trace entries reported in a task snap.

- **snap-trace-limit**
  Specifies the maximum number of trace entries that are reported in a task snap.
  
  **Limit**: 0 - 32767
  
  **Default**: 1000

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

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

- **SNAp TAsk PHOtO is**
  Specifies whether to write a task photo snap to the DC/UCF log file. A task photo snap provides a summary of the resources for the task being snapped.

  - **ON**
    Enables the writing of a task photo snap. This is the default for the ADD SYSTEM statement.

  - **OFF**
    Disables the writing of a task photo snap.

- **STAcksize is tce-stack-size**
  Specifies the size, in words, of the work storage stack in each task control element (TCE).
  
  **Tce-stack-size** must be an integer in the range 0 through 32,767. The default is 1200.

  **Note**: Usage of the stack increases when the system is extended with applications that run in system mode. Examples of such applications are numbered exits, database procedures, TCP/IP generic listeners (defined with MODE IS SYSTEM), and SQL-invoked routines (defined with SYSTEM MODE).

- **STATistics**
  Controls the collection and writing of statistics.

  - **INTerval**
    Specifies the frequency with which system statistics and histograms are written to the log file. Regardless of the STATISTICS INTERVAL specification, system statistics and histograms are written to the log at normal system shutdown and upon explicit request by means of a DCMT WRITE STATISTICS command.
Note: If the system is to write statistics based on a time interval, the internal statistics task, RHDCSTTS, and the program it invokes, RHDCSTTS, must both be accessible to the DC/UCF system. If these programs are secured through CA IDMS central security, you should assign them to a category and grant public access to that category. If these programs are secured through an external security package, you should grant public access to them. For more information about the CA IDMS central security facility, see the CA IDMS Security Administration section.

- **statistics-interval**
  Specifies the statistics interval in wall-clock seconds. 
  *Statistics-interval* must be an integer in the range 0 through 32,767. A value of 0 is synonymous with OFF.

- **OFF**
  Directs the system not to write statistics based on a time interval. 
  OFF is the default when you omit the STATISTICS INTERVAL parameter from the SYSTEM statement.

- **ROLL**
  Directs the system to write system statistics and histograms to the log file and roll them out at specified time and day interval. 
  If you change the local time while Statistics Interval Roll is active, the interval will continue to be every 24 hours from the local time originally used as input. This means that after a time change, such as to daylight savings time, the statistics will run an hour later or earlier in local time. If you want to reset the statistics so that they run at the original local time, you must either restart your system, where the interval will be reset based on the values in the SYSGEN, or use the DCMT VARY STATISTICS ROLL TIME HH:MM command to reset the interval to the local time.

- **TIME is interval-roll-time**
  Specifies the time of day in twenty-four hour format (HH:MM) at which statistics are to be written and reset. The time is interpreted as local time. 
  **Default:** 24:00

- **FREquency is day-frequency**
  Specifies the day frequency at which system statistics and histograms are to be written and reset. 
  **Range:** 1 - 999 
  **Default:** 1

- **NOROLL**
  Directs the system not to perform statistics interval roll.

- **LINE**
  Directs the system to collect by-line histograms.

- **NOLINE**
  Directs the system not to collect by-line histograms. 
  NOLINE is the default when you specify neither LINE nor NOLINE in the STATISTICS parameter of the SYSTEM statement.
Note: Regardless of the LINE/NOLINE specification, system statistics for lines are always collected.

- **TASK**
  Directs the system to collect by-task histograms or task statistics.  
  TASK is the default when you specify neither TASK nor NOTASK in the STATISTICS parameter of the SYSTEM statement.

- **COLlect**
  Directs the system to collect by-task histograms.  
  COLLECT is the default when you specify neither COLLECT nor WRITE in the STATISTICS TASK parameter of the SYSTEM statement.

- **WRite**
  Directs the system to collect and write task statistics for each task.

  Note: Task statistics must be collected to implement limits on task resource usage.

*z/VSE systems*: Task statistics are collected only if the IDMS$SVC macro is assembled with DC=YES.

- **USER**
  Directs the system to maintain separate task CPU-time statistics for system-mode time and user-mode time.

- **NOUser**
  Directs the system to maintain one statistic representing total task execution time. NOUSER is the default when you specify neither USER nor NOUSER in the STATISTICS TASK parameter of the SYSTEM statement.

- **TRANsaction**
  Enables the collection of transaction statistics.

  Note: The collection of transaction statistics must be enabled if CA ADS dialog statistics are collected.

- **NOTransaction**
  Disables the collection of transaction statistics.  
  NOTRANSACTION is the default when you specify neither TRANSACTION nor NOTRANSACTION in the STATISTICS TASK parameter of the SYSTEM statement.

- **NOTask**
  Directs the system not to collect by-task histograms or task statistics.

  Note: Regardless of the TASK/NOTASK specification, system statistics for tasks are always collected.
**STOrage KEY is storage-protect-key**
Specifies the value of the alternate protect key to use for programs running with storage protection enabled:

- **For z/OS, z/VSE, and z/VM systems**, storage-protect-key must be an integer in the range 1 through 15

The default value for storage-protect-key is 9.
When key 9, the default, is used as the alternate protect key, the CA IDMS system and the operating system are protected from user written code. However, non-reentrant programs and user storage are not protected from each other. Alternate protect key 9 is explicitly intended for the high performance requirements of the production CV. When key 9 is used as the alternate protect key, all user storage and non-reentrant programs are swapped into key 9 at startup, and only the PSW key is changed during program execution, thereby, providing negligible CPU overhead compared to running without storage protection.
The storage pools must be defined in such a manner that all forms of user-oriented storage are segregated from the system storage. In other words, define both an XA and a non-XA storage pool for user storage types. Storage types: user, user-kept, shared, and shared-kept, can be together, but they must be defined to secondary storage pools and must be isolated from any secondary pools that contain database or terminal type storage. Program development should not be done with key 9 as the alternate protect key.

**STOrage POOl is storage-pool-size**
Specifies the size, in 1K bytes, of storage pool 0 (the primary storage pool).
Storage-pool-size must be an integer in the range 1 through 16,383. The default is 50.

- **SVC is**
  Specifies whether a CA IDMS SVC is used for communication between the DC/UCF system and programs executing in other regions/partitions.
  **svc-number**
  Identifies the CA IDMS SVC.
  Svc-number must be an integer in the range 1 through 255.
  If a SYSCTL file is defined, the value specified by the SVC parameter is written to the SYSCTL file at system startup.
  **NO**
  Specifies the system does not use an SVC for communication.
  NO is the default when you omit the SVC parameter from the SYSTEM statement.
  **z/VM systems**: The IDMSVSVC module, and not the CA IDMS SVC, is used for communication under z/VM. The SYSTEM statement, therefore, must specify SVC IS NO.

- **SYSctl is**
  Defines the system control (SYSCTL) file batch programs requesting database services from the DC/UCF system will use.
  **ddname**
  Specifies the z/OS or z/VM ddname of the SYSCTL file.
  filename
  Specifies the z/VSE filename of the SYSCTL file.
  **Note**: To change the filename for z/VSE, the filename must also be included in an idmsopti module.
  **SYSctl**
  Specifies SYSCTL as the default value for the z/OS ddname, or the z/VSE filename of the
SYSCTL file.

**DBName is**
Identifies the database or data dictionary to which the system will route requests from programs using the SYSCTL file.

**database-name**
Specifies a database name included in the SYSCTL file.

**Database-name** must be the name of a database accessible by the system being defined or by the system identified by the NODENAME parameter.

**DEFFault**
Indicates that a program using the SYSCTL file will access the named database only if neither the IDMSOPTI module nor the program name a database.

**DEFAULT** is the default when you specify neither DEFAULT nor ALWAYS in the SYSCTL DBNAME parameter of the SYSTEM statement.

**ALWays**
Indicates that programs using the SYSCTL file will always access the named database.

**NULI**
Specifies the SYSCTL file will not contain a database name.

**NODename is**
Identifies the nodename (system-name of SYSTEM ID parameter) to which the system will route requests from programs using the SYSCTL file.

**node-name**
Specifies a node name to be included in the SYSCTL file.

**Node-name** must be the name of a DC/UCF system defined in the CA IDMS communications network.

**DEFFault**
Indicates that database requests from programs using the SYSCTL file are directed to the named node only if neither the IDMSOPTI module nor the program name a node.

**DEFAULT** is the default when you specify neither DEFAULT nor ALWAYS in the SYSCTL NODENAME parameter of the SYSTEM statement.

**ALWays**
Indicates that database requests from programs using the SYSCTL file are always directed to the named node.

**NULI**
Specifies the SYSCTL file will not contain a node name.

**NO**
Specifies that programs requesting services from the DC/UCF system will not use a SYSCTL file.

**NO** is the default when you omit the SYSCTL parameter from the SYSTEM statement.

- **SYSLocks is system-lock-count**
  Estimates the maximum number of database locks that will be held at one time within the system.

  **System-lock-count** must be an integer in the range 1 through 2,147,483,647. The default is 40,000.

- **SYSLocks AUTotune is**
  Indicates if CA IDMS should automatically tune the SYSLOCKS parameter value using execution statistics.

  - **OFF**
    Turns off automatic tuning of SYSLOCKS.

  - **ON**
    Turns on automatic tuning of SYSLOCKS.
SYTrace
Enables or disables the system trace facility.
ON
Enables the system trace facility.
OFF
Disables the system trace facility. Typically, OFF is specified in production environments to eliminate storage and processing overhead.
OFF is the default when you omit the SYSTRACE parameter from the SYSTEM statement.

TRAce
Specifies system trace options.

TABLE SIZE table-size KB|MB
Specifies the size of the system trace table in kilobytes (KB) or megabytes (MB).
Limit: 0 - 9999
Default: 4 MB
Note: Two copies of the system trace table are allocated on all z/OS systems.

ADJunct table SIZE adjunct-size KB|MB
Specifies the size of the adjunct trace table in kilobytes (KB) or megabytes (MB).
Limit: 0 - 9999
Default: 0

SAVE ON|OFF
Controls whether trace information is saved for future reporting.

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

OFF
Disables saving of trace information.

Default:
OFF

TICKer interval is time-check-interval
Specifies the frequency, in wall-clock seconds, with which the system is to check timed functions (such as the runaway interval).
Time-check-interval must be an integer in the range 1 through 32,767. The default is 1. z/VSE systems with BTAM terminals should accept the default value of 1. The recommended value for all other systems is 5.

UNDefined program count is
Specifies the number of null program definition elements (PDEs) to make available to the system.
- **primary count**
  Specifies the number of null PDEs to include in the program definition table at system startup. Primary-count must be an integer in the range 0 through 32,767. The default is 0.

- **secondary-count**
  Specifies the number of null PDEs to include in secondary allocations of null PDEs. When all existing null PDEs, if any, are occupied, the system allocates additional null PDEs from Storage pool 0. Secondary-count must be an integer in the range 0 through 32,767. The default is 0. If the secondary count equals 0, the system will terminate tasks requesting null PDEs when all the null PDEs allocated at startup are occupied. The comma separating the primary and secondary count values is optional.

- **FOR**
  Identifies the types of programs that are eligible for automatic definition. Enclose the program type specification in parentheses. Multiple types must be separated by blanks or commas. Program types not included in the FOR parameter are ineligible for automatic definition. For example, FOR (DIALOGS MAPS TABLES) disallows automatic definition of subschemas.

  **Note:** At runtime, the DC/UCF system enforces restrictions on automatic definition only for programs that reside in the load area of a data dictionary and subschemas that reside in a load library. Dialogs, maps, and tables that reside in a load library are always eligible for automatic definition.

- **ACCess MODule**
  Allows automatic definition of access modules. An access module is an executable form of the SQL statements that programs issue when using the CA IDMS SQL.

- **DIAlogs**
  Allows automatic definition of dialogs created by the CA ADS dialog compiler (ADSC or ADSOBCOM).

- **MAPs**
  Allows automatic definition of maps created by the CA IDMS Mapping Facility.

- **SUBschemas**
  Allows automatic definition of subschemas created by the subschema compiler.

- **TABles**
  Allows automatic definition of edit and code tables created by the IDD DD compiler.

- **ALL**
  Allows automatic definition of CA ADS dialogs, maps, edit, and code tables, and subschemas. ALL is the default when you omit the FOR parameter from the UNDEFINED PROGRAM COUNT parameter in the SYSTEM statement.

- **TRAnsaction SHAring is**
  Specifies the default transaction sharing option for all tasks within the system.

  - **OFF**
    Specifies transaction sharing is disabled for all tasks in the system.
- **ON**
  Specifies transaction sharing is enabled for all tasks in the system.

- **UPDATE**
  Specifies whether the system is to maintain locks automatically for records in areas accessed in protected update usage mode.

- **LOCK**
  Directs the system to maintain locks for records in areas accessed in protected update usage mode. Records updated are not accessed by retrieval run units. However, the maintenance of the additional locks adds overhead. Typically, LOCK is specified in conjunction with the RETRIEVAL LOCK parameter.

- **NOLock**
  Directs the system not to maintain locks for records in areas accessed in protected update usage mode. NOLOCK is the default when you omit the UPDATE parameter from the SYSTEM statement.

- **USERtrace**
  Determines the size of the buffer the user trace facility uses at runtime.

  - **ON entries = user-trace-buffer-entry-count**
    Specifies the number of words allocated to the user trace buffer when the user trace facility is enabled for a terminal. Each user trace entry requires approximately 25 words of storage. *User-trace-buffer-entry-count* must be an integer in the range 1 through 9,999.

  - **OFF**
    Specifies the system will allocate a minimum user trace buffer of 253 words (or approximately 10 entries) when the user trace facility is enabled for a terminal. OFF is the default when you omit the USERTRACE parameter from the SYSTEM statement.

- **XA PROgram pool is xa-program-pool-size**
  For systems supporting 31-bit addressing only, specifies the size, in 1K bytes, of the 31-bit program pool. *Xa-program-pool-size* must be an integer in the range 0 through 2,097,151. The default is 0.

  **Note:** An XA PROGRAM pool may be added, or its size changed, at system generation time. At runtime, the DCMT VARY SYSGEN REFRESH command with the PROGRAM POOL option makes the change effective.

- **XA REEntrant pool is xa-reentrant-pool-size**
  For systems supporting 31-bit addressing only, specifies the size, in 1K bytes, of the 31-bit reentrant pool. *Xa-reentrant-pool-size* must be an integer in the range 0 through 2,097,151. The default is 0.

  **Note:** An XA REENTRANT pool may be added, or its size changed, at system generation time. At runtime, the DCMT VARY SYSGEN REFRESH command with the PROGRAM POOL option makes the change effective.
**XA STOrage pool is xa-storage-pool-size**

For systems supporting 31-bit addressing only, specifies the size, in 1K bytes, of storage pool number 255 (31-bit storage pool). Storage pool 255 is used to allocate XA storage for database processing.

*xa-storage-pool-size* must be an integer in the range 0 through 2,097,151. The default is 0.

Use the XA STORAGE POOL statement to define storage pools 128 through 254. These storage pools are used to allocate XA storage for user applications.

**Note:** An XA STORAGE pool may be added, or its size changed, at system generation time. At runtime, the DCMT VARY SYSGEN REFRESH command with the STORAGE POOL option makes the change effective.

**ALL**

In DISPLAY/PUNCH statements only, directs the system generation compiler to display or punch the entire system definition (that is, the system entity and all entities associated with the specified system).

**CURRENT SYSTEM**

In DISPLAY/PUNCH statements only, directs the system generation compiler to display or punch the definition of the current system.

---

**SYSTEM Statement Usage**

**The SYSTEM ID Parameter**

The name you specify as the *system-name* on the SYSTEM ID parameter of the SYSTEM statement is the identifier of the DC/UCF system you are defining. *System-name* is the nodename of the DC/UCF system and should be specified as the object of a nodename variable as appropriate. Additionally, *system-name* is the name you should specify when creating the resource SYSTEM as part of CA IDMS central security set-up.

**Using AUTOTUNING**

To use auto-tuning, change tracking must be active for the DC/UCF system, since statistical information needed for tuning is captured in the SYSTRK files.

New DCMT commands display and reset values used in tuning.

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

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

**Choosing an Area Acquisition Threshold**
Before the area acquisition threshold count is reached, the system will free locks on areas previously locked by the database transaction if the transaction must wait to place a lock on another area. Once the threshold is reached, the system will not release existing area locks before waiting for a new area lock.

You should specify the default of OFF RETRY FOREVER unless experience shows that a database transaction is not gaining access to areas as needed.

**Choosing a Deadlock Detection Interval**

Always set the DEADLOCK DETECTION INTERVAL parameter to 1.

**System-Internal Parameters**

DPE COUNT is a system-internal parameter. You should accept the default value unless experience indicates that this value is insufficient to meet processing needs.

ECB LIST is a system-internal parameter; you should accept the default value unless experience indicates that this value is insufficient to meet processing needs.

RCE COUNT is a system-internal parameter; you should accept the default value unless experience indicates that this value is insufficient to meet processing needs.

RLE COUNT is a system-internal parameter; you should accept the default value unless experience indicates that this value is insufficient to meet processing needs.

**The ECB List**

The ECB list is used to synchronize events between the DC/UCF system and the host operating system. The list contains pointers to ECBs for each task waiting for an operating system event (for example, a disk read). Upon completing an event for which a task is waiting, the operating system posts the appropriate ECB. The DC/UCF system can then dispatch the waiting task.

**Journaling Retrieval Run Units**

Specifying NOJOURNAL reduces the size of the journal file; however, potentially valuable audit-trail and statistical information is not recorded.

Regardless of the JOURNAL/NOJOURNAL specification, the system always writes the following records to the journal file:

- BFOR and AFTR entries
- TIME, AREA, and ABRT checkpoints
- Records written by a user program

**Specifying a Journal Fragment Interval**
If your journal files are large, a journal fragment interval can significantly enhance warmstart processing. The warmstart logic begins its recovery processing at the most recently accessed journal fragment. Additionally, used with the journal driver task, a journal fragment interval can result in journal records being written to the journal more quickly thus reducing journal file bottlenecks.

To activate the journal fragment interval, you must specify a journal fragment interval of at least 100.

**Specifying a Journal Transaction Level**

You can reduce journal I/O activity by activating the journal transaction level and specifying the point at which CA IDMS will defer the writing of journal buffers to the journal file. Instead, CA IDMS will wait until the journal buffer is full or until the number of active transactions drops below the journal transaction level, before writing the journal buffer to the journal file.

Specify at least a journal transaction level of 4.

**Specifying a Print-Screen Key**

The print-key specification overrides the control-key assignments in the keys table used for online products in the CA IDMS environment. For example, if the keys table for online IDD assigns PF12 the function of printing the entire work file and the SYSTEM statement names PF12 as the print-screen key, PF12 will print only the current screen during an online IDD session. For more information about keys tables, see [KEYS Statement (see page 201)](##).

![Note: The transfer control facility (TCF) uses PF3 and PF9 for the TERMINATE and SUSPEND functions. If you define the system-wide print-screen key as PF3 or PF9, be sure to specify an override for tasks that execute under TCF. You use the system generation TASK statement to override the print-screen key for individual tasks. For more information about the TASK statement, see [TASK Statement (see page 259)](##).](##)

**Adjusting the TCE Stack Size**

The recommended value for the STACKSIZE parameter is 1200. Use this value unless experience indicates it is insufficient to meet processing needs. Runtime task abend code D009 and system abend code 3973 indicate that the storage stack size requires adjustment.

**No CA IDMS SVC under z/OS or z/VSE**

If the SYSTEM statement specifies SVC IS NO under z/OS or z/VSE:

- Batch programs cannot communicate with the system.

- Programs executing under a TP monitor can communicate with the system only if the TP monitor is running in the same region/partition as the DC/UCF system (that is, in attach mode). In this case, the SYSCTL parameter of the SYSTEM statement must identify a system control file.

- Storage protection is not functional.

- The runaway-interval mechanism is not available.
The system cannot include XA program and storage pools.

The IDMSUSVC module must be link edited with the DC/UCF startup routine.

The System Trace Facility

The system trace facility is used to trace system events during the development and debugging of user programs and can be managed at run time with the DCMT DISPLAY SYSTRACE and DCMT VARY SYSTRACE commands. When the system trace facility is enabled, each database or teleprocessing request is traced through the system service modules used in processing the request.

System trace information is recorded in a trace table buffer allocated at system startup. Each entry in the trace table contains the task id, the request type, and the contents of registers 11 through 16 and 1 through 8 for a service request. When the trace table becomes full, the system begins recording new entries at the beginning of the table, overwriting previously written entries. When a program issues a SNAP request, the system writes the trace table to the log.

Automatic Program Definition

Null PDEs are used for the automatic definition of programs not defined by PROGRAM statements during system generation. When you execute a program that is eligible for automatic definition, the system uses a null PDE for the program definition.

Programs that are not eligible for automatic definition must be explicitly defined either during system generation by means of the PROGRAM statement or at runtime by means of the DCMT VARY DYNAMIC PROGRAM command.

More Information:

- For more information about the PROGRAM statement, see PROGRAM Statement (see page 232).
- For more information about the DCMT commands, see the CA IDMS System Operations section.

The User Trace Facility

The user trace facility is used for debugging purposes.

When the user trace facility is enabled for a terminal, information on requests issued from the terminal is recorded in the user trace buffer. When the buffer becomes full, new entries are written to the beginning of the buffer; previously recorded entries are overwritten.

The user trace facility is enabled at runtime by means of either the DCUF USERTRACE command or the DCMT VARY LTERM command.

Specifying commit and rollback options:

You can specify options that control the following commit and rollback behavior.

- The type of journal record written on a commit
- Whether a new local transaction ID is assigned on a rollback continue or commit
You can control whether a COMT or ENDJ journal record is written on a commit operation in which the database session remains active. Writing an ENDJ can reduce recovery time because less data has to be examined to locate the start of a recovery unit. This benefit applies to online recovery, warmstart, and ROLLBACK and ROLLFORWARD recovery operations. ENDJ is most beneficial in cases where long-running transactions perform relatively infrequent updates between commit operations. In cases where update transactions are committed frequently, writing ENDJ journal records can negatively impact the amount of information journaled because another BEGN journal record will have to be written before the first update following a commit operation.

⚠️ **Note:** ENDJ journal records are always written when system rununits are committed, regardless of the ON COMMIT option specified.

You can control whether a new local transaction ID is assigned following a commit or rollback operation in which the database session remains active. Assigning a new transaction ID reduces the chance of duplicate IDs should this value wrap within a single cycle of a central version. It also has the effect of recording journal statistics separately by commit recovery unit rather than across all recovery units within a database transaction. You can assign a new ID on a commit operation only if you also specify that an ENDJ checkpoint record be written.

⚠️ **Note:** A new transaction ID is always assigned when system run units are committed or rolled out.

### Task Snap Dump and System Snap Dump Options

A task snap can provide useful information when developing and debugging user programs. These options can be overridden dynamically by the DCMT command at the Task and Program level.

### Example SYSTEM Statements

#### Contents
- Non-Stop Processing (see page 179)
  - Expanded Statistics Fields (see page 179)

#### Adding a System

The following SYSTEM statement shown adds DC/UCF system 80 to the data dictionary. Parameters not explicitly coded default as indicated in the syntax.

```
ADD SYSTEM 80
  CUSHION IS 6
  DEADLOCK DETECTION INTERVAL IS 300
  DUMP
  GENERATION IDENTIFICATION IS SYS80
  INACTIVE INTERVAL IS 300
  NOJOURNAL RETRIEVAL
  LOG DATABASE
```
MAXIMUM ERUS IS 10
MAXIMUM TASKS IS 13
OPERATING SYSTEM IS MVS
PRINT KEY IS PF12
PRINTER CHECKPOINT IS 50
PROGRAM POOL IS 100
REENTRANT POOL IS 300
RESOURCE TIMEOUT INTERVAL IS 1800
RUNAWAY INTERVAL IS 60
RUNUNITS
  FOR MSGDICT = 2
  FOR SIGNON = 2
  FOR SECURITY = 1
  FOR SYSTEM/DEST = 2
SNAP SYSTEM OFF
SNAP SYSTEM PHOTO OFF
STACKSIZE IS 1200
STATISTICS INTERVAL 3600
STORAGE POOL IS 345
SVC IS 173
SYSCTL IS SYSCTL
SYSLOCKS IS 1200
SYSTRACE ON ENTRIES = 250
TICKER INTERVAL IS 5
USERTRACE ON ENTRIES = 500.

Modifying a System

The following SYSTEM statement modifies the definition of DC/UCF system 80. Parameters not explicitly coded remain unchanged.

MODIFY SYSTEM 80
  NODUMP.

Displaying a System

The SYSTEM statement shown next, if issued after the MODIFY SYSTEM 80 statement shown above, displays the definition of DC/UCF system 80. The display options default as specified in the SET OPTIONS statement.

DISPLAY CURRENT SYSTEM.

The SET OPTIONS statement is described in SET OPTIONS Statement (see page 98).

Deleting a System

The following SYSTEM statement deletes the definition of DC/UCF system 80 from the data dictionary.

DELETE SYSTEM 80.

Non-Stop Processing

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

Expanded Statistics Fields

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

The field size has been increased from single to double words for statistics related to storage management and CPU utilization. These changes affect the following DSECTs:
The following system tasks and operator commands now display larger values:

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

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

System Generation Statements

The system generation statements presented in this section are used to define the component entities of a DC/UCF system, with the exception of the teleprocessing network entities. Statements used to define the teleprocessing network are described in Teleprocessing Network Statements (see page 280).

Statements in this section are presented in alphabetical order. The description of each statement is accompanied by syntax, syntax rules, and examples.

ADD/MODIFY/DELETE and DISPLAY/PUNCH syntax for each entity type are presented together. Syntax rules follow the presentation of both types of syntax. Options that apply to only one verb are noted in the rules. Syntax rules presented are not repeated unless special considerations apply.

ADSO Statement -- Define CA ADS Generation

The ADSO statement is used to define the CA ADS generation and runtime environments. This statement supplies information used by the DC/UCF system to build the CA ADS control block (OCB). The ADSO statement is required if CA ADS is to be used with a DC/UCF system.

CA ADS/Batch users: If ADSO statement parameters are modified before using CA ADS/Batch or the ADSOBCOM and ADSORPTS utilities, a new ADSOOPTI load module must be generated by running ADSOBSYS. For instructions on running the ADSOBSYS utility, see the CA ADS Reference section.
ADSO Statement Syntax

- ADD
- MODIFY
- DELETE

- ACTivity log is YES NO

- ADStask is ADS task-code

- ADS2task is ADS2 task-code

- ADS2TCf task is ADS2U task-code

- AUTODialog is dialog-name

- AUTOStatus is YES NO

- COBol MOVE is YES NO

- COMment DELimiter is adso-comment-delimiter 

- DIAGnostic screen is YES NO

- DIAlog STAtistics

- ON SELECTed
- ALL

- CHEckpoint interval is NULL 200

- FAST mode threshold is OFF

- MAXimum LINKs is 10

- MENu is USER ALL KEEP

- NEWpage mapout is YES NO

- OPERating system is operating-system

- PRImary pool is 4000

- RE Cord COMpression is ON OFF

- RESources are FIXed
ADSO Statement Parameters

**ACTivity log is**

Requests the CA ADS dialog compiler to perform activity logging for database commands in a dialog. Database activity records document potential dialog database activity and are stored in the dictionary when the dialog is generated.

- **YES**
  Enables activity logging.
  YES is the default when you omit the ACTIVITY LOG parameter from the ADSO statement.

- **NO**
  Specifies no activity logging is performed.

**ADStask is task-code-a**

Specifies the task code used to invoke the CA ADS runtime system. Task-code must be a task code that invokes the program ADSORUN1, as defined in the data dictionary with a TASK statement. The default is ADS.

**ADS2task is task-code**
Specifies the secondary task code used to invoke the CA ADS runtime system when not running under the CA IDMS Transfer Control Facility (TCF). (The ADS2TCF parameter, described below, defines the secondary task code under TCF.) This task code is used internally by CA ADS and will never be entered by the terminal operator.

Task-code must be a task code that invokes the program ADSOMAIN, as defined in the data dictionary with a TASK statement. The default is ADS2.

**ADS2TCF task is task-code**

 Specifies the secondary task code used to invoke the CA ADS runtime system while running under TCF. This task code is used internally by CA ADS and will never be entered by the terminal operator.

Task-code must be a task code that invokes the program ADSOMAIN, as defined in the data dictionary with a TASK statement. The default is ADS2U.

**AUTODialog is dialog-name**

Specifies a dialog to be initiated automatically when the terminal operator invokes the CA ADS runtime system but does not enter a dialog name or ADSA application task code.

Dialog-name must be the name of a mainline dialog accessible to the terminal operator at runtime.

**AUTOSTatus is**

Specifies the default AUTOSTATUS setting for dialog generation sessions.

- **YES**
  Enables AUTOSTATUS for all dialogs at the beginning of a dialog generation session.
  YES is the default when you omit the AUTOSTATUS parameter from the ADSO statement.

- **NO**
  Disables AUTOSTATUS at the beginning of a dialog generation session.

- **OPTIONal**
  Allows the application developer to override the default AUTOSTATUS setting during dialog generation.
  OPTIONAL is the default when you specify neither OPTIONAL nor MANDATORY in the AUTOSTATUS parameter of the ADSO statement.

- **MANdatory**
  Prevents the application developer from changing the default AUTOSTATUS setting during dialog generation.

**COBol MOVe is**

Defines the treatment of numeric values placed into alphanumeric fields by arithmetic and assignment commands. The COBOL MOVE parameter assigns the default option displayed on the CA ADS dialog compiler Dialog Options screen.
- **YES**
  Indicates that CA ADS automatically moves the result of an arithmetic or assignment command into the target field according to standard COBOL rules.

- **NO**
  Indicates that CA ADS automatically moves the result of an arithmetic or assignment command into the target field according to standard CA ADS rules. NO is the default when you omit the COBOL MOVE parameter from the ADSO statement.

- **OPTIONal**
  Indicates the application developer can change the COBOL MOVE setting on a dialog-by-dialog basis.

- **MANdatory**
  Indicates the application developer cannot change the COBOL MOVE setting on a dialog-by-dialog basis.

**Note:** For more information about the COBOL MOVE parameter, see the CA ADS Reference section.

**COMment DELimiter is ‘adso-comment-delimiter’**

Specifies the character to be used as the comment delimiter. adso-comment-delimiter must be a single character enclosed in single quotes.

**Default:** An exclamation mark (‘!’)

**DIAgnostic screen is**

Controls the display of the Dialog Abort Information screen when the CA ADS runtime system detects an abend condition in an executing dialog.

- **YES**
  Indicates that CA ADS will display the Dialog Abort Information screen and will write error messages to the system log. YES is the default when you omit the DIAGNOSTIC SCREEN parameter from the ADSO statement.

- **NO**
  Indicates that CA ADS will not display the Dialog Abort Information screen. When the runtime system terminates an abending dialog, CA ADS writes error messages to the system log and displays the following to the terminal operator:

```
ERROR OCCURRED DURING PROCESSING. CA ADS DIALOG ABORTED.
```

**DIAlog STAtistics**

 Specifies whether dialog statistics will be collected for CA ADS dialogs.
Note: Users can override DIALOG STATISTICS parameter options at runtime by means of
the DCMT VARY ADSO STATISTICS command. DCMT commands are described in CA IDMS
System Tasks and Operator Commands section. Note: CA ADS/Batch dialog statistics
collection is enabled by means of the ADSOBSYS utility.

ON

Indicates that dialog statistics and application overhead activity statistics are collected.

CA ADS dialog statistics are collected only if transaction statistics collection is enabled.

SELECTED

Enables statistics collection for dialogs on a dialog-by-dialog basis in conjunction with the
DIALOG STATISTICS parameter of the PROGRAM statement (described in PROGRAM
Statement.)

ALL

Enables statistics collection for all dialogs.

CHECKPOINT INTERVAL IS

Specifies the frequency with which dialog statistics are written to the system log file. The
checkpoint interval is based on the number of times statistics are accumulated for any
dialog.

checkpoint-interval

Specifies that dialog statistics are written to the log file after being accumulated the
indicated number of times.

Checkpoint-interval must be an integer in the range 0 through 32,767. The default is 200. A
value of 0 is synonymous with NULL.

NULL

Specifies that no checkpoint interval applies and that statistics records are written to the
log file after an application terminates.

OFF

Indicates that no dialog statistics will be collected.

OFF is the default when you omit the DIALOG STATISTICS parameter from the ADSO
statement. Note: For more information about dialog statistics, see the System Operations
section.

FAST mode threshold is
Specifies the point at which the CA ADS runtime system is to write record buffer blocks (RBBs) and statistics control blocks to the scratch area (DDLDCSCR) of the data dictionary across a pseudo-converse. If the total size, in bytes, of the RBBs and statistics control blocks in all storage pools exceeds the fast mode threshold (specified in bytes), and if the ADSO statement specifies RESOURCES ARE FIXED, the system writes the RBBs and statistics control blocks to scratch.

Note: If the ADSO statement specifies RESOURCES ARE RELOCATABLE, the system writes RBBs and statistics control blocks to scratch across a pseudo-converse regardless of the FAST MODE THRESHOLD specification.

MAXimum LINks is max-link-count

Specifies the maximum number of dialog levels that can be established by each CA ADS application thread.

Max-link-count must be an integer in the range 0 through 9,999. The default is 10.

The value specified for max-link-count should be equal to at least 3. A value of 5 is sufficient for most systems.

Important! Some CA products are CA ADS applications. Specifying too low a value for the MAXIMUM LINKS parameter will affect the performance of these products.

MENu is

Controls the display of mainline dialogs on the CA ADS runtime system Dialog Selection screen.

- USER
  Specifies that CA ADS will display a mainline dialog on the menu screen only if the user holds the authority to execute the dialog.
  For more information about security, see the CA IDMS Security Administration section.
  USER is the default when you omit the MENU parameter from the ADSO statement.

- ALL
  Specifies that CA ADS will display all mainline dialogs on the menu screen.

- KEEP
  Directs CA ADS to save the menu screen across pseudo-converses. If you do not specify KEEP, CA ADS rebuilds the menu screen each time it is displayed.

NEWpage mapout is

Specifies how CA ADS is to perform a mapout when a dialog's map is already displayed as the result of a previous mapout.
**YES**
Indicates that CA ADS will always perform a new page mapout, mapping out literal fields as well as data, message, and page fields.

**NO**
Indicates that CA ADS will transmit only the map's data fields, message field, and page field (pageable maps only). A new page mapout that includes literal fields will occur, however, if problems have occurred or if the terminal operator has pressed the CLEAR key prior to the last mapin.
NO is the default when you omit the NEWPAGE MAPOUT parameter from the ADSO statement.

⚠️ **Note:** If the BACKSCAN option is enabled for any of the map's data fields, data from the previous mapout may remain in the fields.

**P**r**i**mary pool is **primary-buffer-size**

Specifies the size, in bytes, of the primary buffer. The primary buffer is acquired at runtime for any CA ADS dialog that uses a database, map, subschema, logical, or work record and is used to hold those records. The primary buffer can be allocated from storage pool 0 or from an XA storage pool if one is defined. The LOCATION parameter on the ADS task statement determines where storage is allocated. Storage can be obtained from pool 0 or an XA storage pool when LOCATION=ANY is specified. LOCATION=BELOW gets storage from pool 0. If the primary buffer becomes full, CA ADS allocates storage for a secondary buffer, also determined by the LOCATION parameter (see the SECONDARY POOL parameter).

*Primary-buffer-size* must be an integer in the range 0 through 2,147,483,647; the default is 4,000.

⚠️ **Note:** The size of the primary buffer for CA ADS/Batch dialogs is specified by the ADSOBSYS utility.

**RE**c**o**rd **C**om**p**r**e**ssion is

Specifies whether record buffer blocks (RBBs) are compressed across a pseudo-converse when they are retained in the storage pool.

- **ON**
  Specifies that RBBs are compressed across a pseudo-converse and restored at the end of the pseudo-converse.

- **OFF**
  Specifies that RBBs are not compressed across a pseudo-converse. OFF is the default when you omit the RECORD COMPRESSION parameter from the ADSO statement.

**RE**s**o**ur**ces are
Specifies whether storage used by the CA ADS runtime system is eligible for writing to the scratch area (DDLDCSCR) of the data dictionary across a pseudo-converse.

- **FIXed**
  Indicates that:
  - **Storage used for record buffer blocks (RBBs) and statistics control blocks** is written to scratch across a pseudo-converse only when the fast mode threshold is exceeded. You use the ADSO statement FAST MODE THRESHOLD parameter (described above) to establish a fast mode threshold.
  - **Storage used for currency blocks, CA ADS terminal blocks (OTBs), OTB extensions, and variable dialog blocks (VDBs)** is not eligible for writing to scratch.

- **RELocatable**
  Indicates that:
  - **Storage used for RBBs and statistics control blocks** is always written to scratch across a pseudo-converse, regardless of the relocatable threshold.
  - **Storage used for currency blocks, OTBs, OTB extensions, and VDBs** is written to scratch across a pseudo-converse only when the relocatable threshold is exceeded.

You use the RELOCATABLE THRESHOLD parameter of the SYSTEM statement to define a relocatable threshold for the primary storage pool.
You use the RELOCATABLE THRESHOLD parameter of the system generation STORAGE POOL and XA STORAGE POOL statements to define relocatable thresholds for secondary storage pools.
The SYSTEM statement is described in **SYSTEM Statement** (see page 131).
The STORAGE POOL and XA STORAGE POOL statements are described later in this section.

**SECondy pool is secondary-buffer-size**

Specifies the size, in bytes, of the secondary buffer. When the primary buffer becomes full, the secondary buffer is allocated either from the DC/UCF system storage pool 0 or from the XA storage pool (see the PRIMARY POOL parameter above). If the secondary buffer becomes full, the CA ADS system allocates additional secondary buffers as necessary.

**Secondary-buffer-size** must be an integer in the range 0 through 2,147,483,647. The default is 2,000.

⚠️ **Note:** The size of the secondary buffer for CA ADS/Batch dialogs is specified by the ADSOBSYS utility.

**STAtus definition record is status-definition-record-name**

Specifies the default status definition record used by both CA ADS and CA ADS/Batch dialogs to reference error-status values returned by executing dialogs.

**Status-definition-record-name** must be the name of a status definition record previously defined in the data dictionary. The default is ADSO-STAT-DEF-REC.
- **Version is version-number**
  Qualifies the named status definition record with a version number. 
  *Version-number* must be an integer in the range 1 through 9,999. The default is 1.

- **OPTIONal**
  Allows the application developer to override the default status definition record specification during dialog generation. The application developer can supply the name of an alternate status definition record for each generated dialog. 
  *OPTIONAL* is the default when you specify neither *OPTIONAL* nor MANDATORY in the STATUS DEFINITION RECORD parameter of the ADSO statement.

- **MANDatory**
  Prevents the application developer from overriding the default status definition record specification during dialog generation.

**STOrage MODe is**

Determines the amount of storage to be allocated for record buffer blocks (RBBs).

- **SGEnsize**
  Directs the CA ADS runtime system to use the buffer sizes specified by the PRIMARY POOL and SECONDARY POOL parameters (described above) when acquiring storage space for RBBs. 
  *SGENSIZE* is the default when you omit the STORAGE MODE parameter from the ADSO statement.

- **CALculated**
  Directs the CA ADS runtime system to calculate the size of the RBBs for an application or dialog and to use the calculated size when acquiring storage space for the RBBs.

**Note:** For more information about calculated storage, see Storage Pools (see page 70).

### ADSO Statement Usage

**CA ADS Buffer Sizes**

The following considerations apply to specifying the primary and secondary buffer sizes in the ADSO statement:

- Each buffer contains one record-buffer-block (RBB) header. The RBB header requires 32 bytes of storage.

- For each record, CA ADS maintains a header:
  - Database record headers require 52 bytes of storage
  - Logical record headers require 68 bytes of storage

- Each buffer must be large enough to accommodate the largest subschema, map, work, database, or logical record used by a dialog, plus the RBB header, plus the record header.
• All records and headers are aligned on doubleword boundaries.

• The primary and secondary buffer sizes are also used by the CA ADS and CA ADS/Batch dialog compilers, by the CA ADS application compiler, and by the ASF interface into CA ADS to allocate work areas for internal control blocks.

Example ADSO Statements

Defining the CA ADS Environments

The following statement defines the generation and runtime environments for CA ADS:

ADD ADSO
  ADSTASK IS ADSO
  ADS2TASK IS ADS02
  AUTOSTATUS IS YES OPTIONAL
  DIAGNOSTIC SCREEN IS YES
  DIALOGSTATISTICS ON ALL
  FAST MODE THRESHOLD IS 10000
  MAXIMUM LINKS 4
  MENU IS USER KEEP
  NEWPAGE MAPOUT IS NO
  PRIMARY POOL IS 6000
  SECONDARY POOL IS 10000
  STATUS DEFINITION RECORD IS ADSO-STAT-DEF-REC OPTIONAL.

Modifying the CA ADS Runtime System

The following statement modifies the CA ADS runtime system definition by changing the maximum links value to 5:

MODIFY ADSO
  MAXIMUM LINKS 5.

Deleting the CA ADS Definition

The following statement deletes the CA ADS definition from the data dictionary:

DELETE ADSO.

AUTOTASK Statement -- Define Tasks

The AUTOTASK statement is used to define a task called an autotask. Autotasks are automatically initiated by the system at startup and/or shutdown. Autotasks are noninteractive tasks; that is, they are not associated with a logical terminal. Autotasks are often used for statistics collection, message handling, or special terminal handling.

AUTOTASK Statement Syntax

ADD/MODIFY/DELETE AUTOTASK Statement

ADD AUTotask task-code

MODify
AUTOTASK Statement Parameters

**AUTotask task-code**

Specifies the system will execute the task identified by *task-code* automatically.

*Task-code* must be a task code previously defined in the data dictionary with a TASK statement. The TASK statement should specify the NOINPUT option, which causes the task to execute as soon as it receives control.

**INVoked at**

Specifies when the system will execute the autotask.

- **STArtup**
  Specifies the system is to execute the task at startup, after initialization is complete, before any user tasks are executed. STARTUP is the default when you omit the INVOKED AT parameter from the AUTOTASK statement.

- **BOTH**
  Specifies the system is to execute the task both at startup and at shutdown.

- **PREempt**
  Indicates the startup autotask will execute to completion before any user tasks are executed.

- **ABEnd**
  Directs the system to terminate abnormally if the startup autotask terminates abnormally.
AUTOTASK Statement Usage

Defining Autotasks

You define each autotask with a separate AUTOTASK statement, specifying a task code and the time at which the autotask is invoked. When the AUTOTASK statement is successfully submitted to the system generation compiler, the compiler cross-references the autotask with its corresponding task in the data dictionary. At runtime, the DC/UCF system executes multiple autotasks in the order in which they are defined.

Note: The AUTOTASK statement should not be confused with the LTERM statement AUTOTASK parameter, which defines a task to be initiated automatically when a logical terminal is enabled. For more information about the LTERM statement, see LTERM Statement (see page 290).

Startup Autotasks

The system attaches startup autotasks after initialization is complete. A startup autotask can preempt execution of other user tasks or can execute concurrently with user tasks. For more information about special considerations for startup autotasks, see the AUTOTASK syntax description.

Shutdown Autotasks

The system attaches shutdown autotasks during a planned shutdown (that is, in response to a SHUTDOWN command) after all other user tasks are terminated.

The ABEND/NOABEND Parameter

Before defining a startup autotask as PREEMPT ABEND, you should thoroughly test the task to ensure that it runs normally. Otherwise, if the startup autotask abends, the system will abend. The suggested specification is PREEMPT NOABEND.
Example AUTOTASK Statements

Adding an Autotask

The following statement defines task code ABC as an autotask that will execute in response to a SHUTDOWN command:

```plaintext
ADD AUTOTASK ABC
INVOKED AT SHUTDOWN.
```

Modifying an Autotask

The following statement modifies autotask ABC to execute before user signon as well as at shutdown. The specification PREEMPT indicates that, during startup, autotask ABC will execute to completion before any other user task is executed:

```plaintext
MODIFY AUTOTASK ABC
INVOKED AT BOTH PREEMPT.
```

Deleting an Autotask

The following statement deletes autotask ABC:

```plaintext
DELETE AUTOTASK ABC.
```

DEFAULT PROGRAM Statement -- Assign Default Values

The DEFAULT PROGRAM statement is used to establish global defaults for system generation PROGRAM statement parameters, overriding the defaults shown in the PROGRAM statement syntax.

You can assign all PROGRAM statement parameters as defaults with the DEFAULT PROGRAM statement. If the program is a map or subschema, however, defaults for the following parameters are set automatically and cannot be changed:

- LANGUAGE IS ASSEMBLER
- CONCURRENT
- NOSAVEAREA
- REENTRANT
- REUSABLE

DEFAULT PROGRAM Statement Syntax

```
DEFAULT PROGRAM Statement
```
DEFAULT PROGRAM Statement Parameters

TEMPorary

Indicates the specified defaults are temporary.

TEMPORARY is the default when you specify neither TEMPORARY nor PERMANENT in the DEFAULT PROGRAM statement.

PERmanent

Indicates the specified defaults are permanent.

program-statement-parameter

Specifies a default value to apply to the applicable PROGRAM statement parameter.

Program-statement-parameter is any PROGRAM statement parameter, including VERSION, described under PROGRAM Statement (see page 232). Specify as many PROGRAM parameters as necessary to override PROGRAM statement defaults.

DEFAULT PROGRAM Statement Usage

Permanent and Temporary Defaults

You can designate DEFAULT PROGRAM statement defaults as permanent or temporary:

- Permanent -- The defaults apply to all executions of the system generation compiler until overridden by another DEFAULT statement. Permanent defaults are stored in the data dictionary as a prototype program source record with the name 0DEFAULT, where 0 is hexadecimal 00.
• **Temporary** -- The defaults apply only to the current execution of the system generation compiler and are not stored in the data dictionary.

Defaults established with the DEFAULT PROGRAM statement apply to all PROGRAM statements following the DEFAULT PROGRAM statement. The system generation compiler maintains a list of all defaults in effect. The compiler updates the list each time the defaults are changed with a DEFAULT PROGRAM statement. The DISPLAY/PUNCH PERMANENT/TEMPORARY DEFAULT PROGRAM statement can be used to view the current list.

### Example DEFAULT PROGRAM Statement

**Specifying Temporary Defaults**

The following DEFAULT PROGRAM statement assigns temporary default values to four PROGRAM statement parameters that apply to CA ADS dialogs:

```
DEFAULT TEMPORARY PROGRAM
  LANGUAGE IS ADSO
  DIALOG
  NOMAINLINE
```

### DESTINATION Statement -- Group Users or Terminals

The DESTINATION statement is used to group users or logical terminals into a single logical destination for the purpose of message or report routing. The destination definition comprises a destination identifier and the list of users or logical terminals that constitute the destination. Each destination consists of one or more occurrences of a single type of entity (logical terminal, user, or printer).

### DESTINATION Statement Syntax

**ADD/MODIFY/DELETE DESTINATION Statement**

```
ADD/MODIFY/DELETE DESTINATION

  Destination destination-id

  Version is version-number

  ENAbled
  DISabled

  LTERm (logical-terminal-id)
  DELete (logical-terminal-id)
```
DESTINATION Statement Parameters

**DESTination destination-id**

Specifies the destination identifier.

*Destination-id* must be a one- through eight-character alphanumeric value.

**Version is version-number**

Qualifies the destination with a version number.

*Version-number* must be an integer in the range 1 through 9,999. The default is 1.

Although multiple versions of a destination can exist in the data dictionary, a DC/UCF system uses only one version of a destination at runtime. If multiple ADD DESTINATION statements specify the same destination identifier, the system uses the first statement that appears in the system definition.

**ENABLEd**

Specifies the destination is enabled when the DC/UCF system starts up.

ENABLED is the default when you specify neither ENABLEd nor DISABLEd in the DESTINATION statement.

**DISABLEd**

Specifies the destination is not enabled when the DC/UCF system starts up.
Note: Users can override the ENABLED/DISABLED parameter at runtime for the duration of system execution with the DCMT VARY DESTINATION command.

**LTErm (logical-terminal-id)**

Specifies one or more logical terminals that constitute the destination.

*Logical-terminal-id* must be the identifier of a logical terminal previously defined in the data dictionary with an LTERM statement.

The logical terminal identifiers must be enclosed in parentheses. Multiple identifiers must be separated by one or more commas or blanks.

**DELete (logical-terminal-id)**

Directs the DC/UCF system to delete the specified logical terminal identifiers from the destination definition.

*Logical-terminal-id* must be the identifier of a logical terminal previously included in the destination.

The logical terminal identifiers must be enclosed in parentheses. Multiple identifiers must be separated by one or more commas or blanks.

This parameter is valid only for MODIFY operations.

**PRInter (printer-logical-terminal-id)**

Defines the destination as a printer destination and specifies one or more logical terminals that constitute the destination.

*Printer-logical-terminal-id* must be the identifier of a logical terminal previously defined in the data dictionary as a printer (with an LTERM statement).

**DELete (printer-logical-terminal-id)**

Directs the DC/UCF system to delete the specified printer identifiers from the destination definition.

*Printer-logical-terminal-id* must be the identifier of a printer previously included in the destination.

The printer identifiers must be enclosed in parentheses. Multiple identifiers must be separated by one or more commas or blanks.

This parameter is valid only for MODIFY operations.

**USEr (user-id)**

Specifies one or more users that constitute the destination. If the user identifier contains blanks or special characters, the identifier must be enclosed in site-standard quotation marks.
The user identifiers must be enclosed in parentheses. Multiple identifiers must be separated by one or more commas or blanks.

**DELETE (user-id)**

Directs the DC/UCF system to delete the specified user identifiers from the destination definition.

*User-id* must be the identifier of a user previously included in the destination. If the user identifier contains blanks or special characters, the identifier must be enclosed in site-standard quotation marks.

The user identifiers must be enclosed in parentheses. Multiple identifiers must be separated by one or more commas or blanks.

This parameter is valid only for MODIFY operations.

**DESTINATION Statement Usage**

**At Least One Entity in a Destination**

One LTERM, USER, or PRINTER option must be specified for each DESTINATION statement to generate an executable system.

**Defining a Destination with Logical and Printer Terminals**

When defining a destination composed of logical terminals or printer terminals, you should ensure that an LTERM statement and its associated PTERM statement exist in the system definition for each destination entry. The optional PTERM statement PRINTER DESTINATION parameter assigns a destination for WRITE TO PRINTER requests. You should ensure that, if a DESTINATION entity occurrence is deleted; all PTERM PRINTER DESTINATION parameters are updated to reference existing destinations.

**Destinations not Required for Message Routing Facilities**

You do not have to define destinations to use DC/UCF message routing facilities. Messages can be routed to individual users or logical terminals.

**Example DESTINATION Statements**

**Defining a Destination with Logical Terminals**

The following statement creates destination ABC, which consists of four logical terminals:

```
ADD DESTINATION ABC
LTERM (LTM001 LTM012 LTM041 LTM153).
```

**Defining a Destination with Users**

The following statement creates destination XYZ, which consists of four users:
ADD DESTINATION XYZ
    USER (WHH JPK HMS HAL).

**Defining a Printer Destination**

The following statement creates the printer destination PRT, which consists of the logical terminal PRTLTI:

ADD DESTINATION PRT
    PRINTER (PRTLTI).

**Modifying a Destination**

The following statement modifies destination XYZ by adding users ALK and RIL and deleting users HMS and HAL:

MODIFY DESTINATION XYZ
    USER (ALK, RIL)
    DELETE (HMS, HAL).

**Deleting a Destination**

To delete destination ABC from earlier example:

MOD DESTINATION ABC
    DEL (LTM001 LTM012 LTM041 LTM153).
    GEN.
    DELETE DESTINATION ABC.
    GEN.

**IDD Statement -- Define Default Usage Mode**

The IDD statement is used to define the default usage mode in which the IDD DDDL compiler will access the data dictionary at runtime. Different default usage modes can be specified for separate areas of the data dictionary. The IDD statement applies to all dictionaries defined to the DC/UCF system regardless of the segment to which it belongs.

⚠️ **Note:** The usage mode can be overridden at runtime when the user signs on to IDD using the DDDL compiler SIGNON statement. For more information, see the *CA IDMS IDD DDDL Reference section.*

**IDD Statement Syntax**

ADD/MODIFY/DELETE IDD Statement

```
ADD/MODIFY/DELETE IDD Statement

ADD
MODify
DELETE

USAGE mode is UPDATE PROtected UPDATE for DDLDMl
```
IDD Statement Parameters

**Usage mode is**

Specifies the area usage mode in which the DDDL compiler can access the data dictionary.

- **Update**
  Allows all IDD users to update the data dictionary concurrently. The DDDL compiler has extensive code to prevent deadlock conditions or situations in which users must wait for commands issued by other users to be processed.
  
  UPDATE is the default when you omit the USAGE MODE parameter from the IDD statement.

  **Note:** Shared update is the suggested usage mode for IDD.

- **Protected Update**
  Allows only one IDD user to update the data dictionary at a time. Other users can perform only retrieval operations against the data dictionary. During an online IDD session, one user has exclusive control for update only when the DDDL compiler is invoked. Between terminal interactions, the areas can be updated by other users.

- **Retrieval**
  Allows all IDD users to perform only retrieval operations against the data dictionary.

**for**

Specifies the areas to which the selected usage mode applies.

- **DDLDMl**
  Indicates the specified usage mode applies only to the DDLDMl area.

- **DDLCLOD**
  Indicates the specified usage mode applies only to the DDLCLOD area.
• **DDLDCMSG**
  Indicates the specified usage mode applies only to the DDLDCMSG area.

• **ALL**
  Indicates the specified usage mode applies to all areas.
  ALL is the default when you omit the FOR parameter from the USAGE MODE parameter of the IDD statement.

**Example IDD Statement**

**Defining the Default Usage Mode for IDD**

The following example illustrates the use of the IDD statement. The default usage mode for all areas of the data dictionary is set to protected update for IDD.

`ADD IDD USAGE IS PROTECTED UPDATE FOR ALL.`

**KEYS Statement -- Define a Keys Table**

The KEYS statement is used to define a table that associates 3270-type terminal control keys with the following online components of a DC/UCF system:

- CA ADS Runtime System
- Line-mode I/O operations
- Online debugger menu mode
- Online IDD
- CA OLQ menu facility
- Online schema compiler
- Online subschema compiler
- Online system generation compiler

⚠️ **Note:** The control-key assignment made by the PRINT KEY parameter of the system generation SYSTEM statement overrides all KEYS statement assignments for the specified key. For more information about the SYSTEM statement, see SYSTEM Statement (see page 131).
KEYS Statement Syntax

ADD/MODIFY/DELETE KEYS Statement

```
<table>
<thead>
<tr>
<th>ADD</th>
<th>KEYS for TABle keys-table-name</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODIFY</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>are</th>
<th>CLEar is function function-number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ENTer</td>
</tr>
<tr>
<td></td>
<td>PA</td>
</tr>
<tr>
<td></td>
<td>PFn</td>
</tr>
</tbody>
</table>

```

Expansion of application-specification

```
<table>
<thead>
<tr>
<th>for</th>
<th>APPLICATION application-name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>NULL</td>
</tr>
<tr>
<td>CLEar</td>
<td>is function function-number</td>
</tr>
<tr>
<td>ENTer</td>
<td></td>
</tr>
<tr>
<td>PA</td>
<td></td>
</tr>
<tr>
<td>PFn</td>
<td></td>
</tr>
</tbody>
</table>

```

DISPLAY/PUNCH KEYS Statement

```
<table>
<thead>
<tr>
<th>DISPLAY</th>
<th>KEYS for TABle keys-table-name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUNCH</td>
<td>ALL KEYS</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>WITH</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALSO WITH</td>
<td>HISTORY</td>
</tr>
<tr>
<td>WITHOUT</td>
<td>ALL NONE</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>AS</td>
<td>COMments SYNTAX</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TO</td>
<td>module-specification</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```

KEYS Statement Parameters

**TABLE keys-table-name**

Specifies the name of the keys table.

*Keys-table-name* must be a one- through eight-character alphanumeric value.

**are**
Specifies the control-key assignments in the keys table. You can specify any number of control-key assignments.

- **CLEar**
  Assigns the specified function to the CLEAR key.

- **ENTER**
  Assigns the specified function to the ENTER key.

- **PA**
  Assigns the specified function to the named program attention key.
  \( N \) must be 1, 2, or 3.

- **PFN**
  Assigns the specified function to the named program function key.
  \( N \) must be an integer in the range 1 through 24.

- **is function function-number**
  Specifies the application function to which the named control key is to be assigned. \( Function-number \) must be an integer in the range 0 through 255 that corresponds to an application function. Leading zeros are not required.

**application-specification**

Associates one or more applications with the keys table and optionally establishes unique control-key assignments for each application.

Expanded syntax for application-specification is shown above immediately following the KEYS statement syntax.

**for APPlication application-name**

Associates the named application with the keys table.

**Application-name** must be the one- through eight-character name of an application.

The applications listed in the SYSTEM table are referenced by the following values: IDD, OLM, OLQ, ADSO, ADSA, ADSORUN, or LINEMODE. IDD refers to online IDD, the online schema compiler, the online subschema compiler, and the online system generation compiler, and the online subschema compiler.

**DElete**

Disassociates the named application from the keys table.

**NULI**

Clears the control-key assignments for the named application. Control-key assignments for the application revert to those specified in the ARE parameter.

**CLEar**

Assigns the specified function to the CLEAR key for the named application only.
ENTer
Assigns the specified function to the ENTER key for the named application only.

PAn
Assigns the specified function to the named program attention key for the named application only.

N must be 1, 2, or 3.

PFn
Assigns the specified function to the named program function key for the named application only.

N must be an integer in the range 1 through 24.

is function function-number
Specifies the application function to which the named control key is to be assigned.

Function-number must be an integer in the range 0 through 255 that corresponds to an application function. Leading zeros are not required.

KEYS Statement Usage

Defining Keys Tables
You can include any number of keys tables in a system definition. Each table can contain separate control-key assignments for the components included in the table and/or common control-key assignments to be shared by one or more components. Within each table, multiple keys can be defined to invoke the same function.

Establishing Default Control-Key Assignments
Default control-key assignments are supplied for each of the components listed above in a predefined keys table named SYSTEM. The definition of the SYSTEM keys table is stored in a data dictionary module named DC-PFKEY-DEF during installation. You can use the INCLUDE statement to include the definition of the SYSTEM keys table in the current system definition. For more information about the INCLUDE statement, see INCLUDE Statement (see page 114).

You can use the KEYS statement to modify the SYSTEM keys table or to add a different table named SYSTEM to the system definition; however, a table named SYSTEM should be included in all system definitions. At runtime, the DC/UCF system uses the table named SYSTEM as a default. If you do not include a keys table named SYSTEM in the system definition, the DC/UCF system will not have default control keys.

Displaying Keys Tables
At runtime, users can issue the DCUF SET TABLE command to establish a different keys table as the session default. Additionally, users can issue a DCUF SHOW TABLES command to display the available keys tables and a DCUF SHOW KEYS command to view control-key assignments. DCUF commands are described in the CA IDMS System Tasks and Operator Commands section.
The SYSTEMS Keys Table

The following table lists the default control-key assignments in the SYSTEM keys table. Note that online IDD, the online schema compiler, the online subschema compiler, and the online system generation compiler share default control-key assignments.

<table>
<thead>
<tr>
<th>Application Name</th>
<th>Function</th>
<th>Description</th>
<th>Default Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADSORUN</td>
<td>001</td>
<td>Unassigned</td>
<td>PA1</td>
</tr>
<tr>
<td>CA ADS runtime system</td>
<td>002</td>
<td>Leave application</td>
<td>CLEAR</td>
</tr>
<tr>
<td></td>
<td>003</td>
<td>Leave CA ADS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>004</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td></td>
<td>005</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td></td>
<td>006</td>
<td>RETURN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>007</td>
<td>RETURN CONTINUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>008</td>
<td>RETURN CLEAR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>009</td>
<td>RETURN CLEAR CONTINUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>010</td>
<td>RETURN TO TOP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>RETURN TO TOP CONTINUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>012</td>
<td>RETURN TO TOP CLEAR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>013</td>
<td>RETURN TO TOP CLEAR CONTINUE</td>
<td></td>
</tr>
<tr>
<td>DEBUG</td>
<td>001</td>
<td>Return to prompt mode</td>
<td>PF9</td>
</tr>
<tr>
<td>Online debugger menu mode</td>
<td>002</td>
<td>Display Usage screen</td>
<td>PF1</td>
</tr>
<tr>
<td></td>
<td>003</td>
<td>Display activity screen for current command</td>
<td>PF3</td>
</tr>
<tr>
<td></td>
<td>004</td>
<td>current command</td>
<td>PF4</td>
</tr>
<tr>
<td></td>
<td>005</td>
<td>Display help screen for current command</td>
<td>PF2</td>
</tr>
<tr>
<td></td>
<td>006</td>
<td>command</td>
<td>PF6</td>
</tr>
<tr>
<td></td>
<td>007</td>
<td>Unassigned</td>
<td>PF7</td>
</tr>
<tr>
<td></td>
<td>008</td>
<td>Display Keys screen</td>
<td>PF8</td>
</tr>
<tr>
<td></td>
<td>009</td>
<td>Display prior page</td>
<td>PF5</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td>Display next page</td>
<td>PA2</td>
</tr>
<tr>
<td>Application Name</td>
<td>Function</td>
<td>Description</td>
<td>Default Key</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>--------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>Display Symbols screen</td>
<td>PF11</td>
</tr>
<tr>
<td></td>
<td>012</td>
<td>Exit debugger</td>
<td>PF12</td>
</tr>
<tr>
<td></td>
<td>013</td>
<td>Unassigned</td>
<td>ENTER</td>
</tr>
<tr>
<td></td>
<td>014</td>
<td>Print</td>
<td>PA1</td>
</tr>
<tr>
<td></td>
<td>015</td>
<td>Process current screen</td>
<td>PF10</td>
</tr>
<tr>
<td></td>
<td>016</td>
<td>Refresh current screen</td>
<td>CLEAR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return up one level</td>
<td></td>
</tr>
<tr>
<td>IDD</td>
<td>001</td>
<td>Display next page</td>
<td>PF1/PF8/ PF13/PF20</td>
</tr>
<tr>
<td>Online IDD</td>
<td>002</td>
<td>Display prior page</td>
<td>PF2/PF7/ PF14/PF19</td>
</tr>
<tr>
<td>Online schema compiler</td>
<td>003</td>
<td>Display next line</td>
<td>PF3/PF15</td>
</tr>
<tr>
<td>Online subschema compiler</td>
<td>004</td>
<td>Insert a screen of lines in work</td>
<td>PF4/PF16</td>
</tr>
<tr>
<td>Online system generation compiler</td>
<td>005</td>
<td>file</td>
<td>PF5/PF17</td>
</tr>
<tr>
<td></td>
<td>006</td>
<td>Update screen contents and work</td>
<td>PF6/PF18</td>
</tr>
<tr>
<td></td>
<td>007</td>
<td>file</td>
<td>PF8</td>
</tr>
<tr>
<td></td>
<td>008</td>
<td>Update the work file and execute</td>
<td>PF9/PF21</td>
</tr>
<tr>
<td></td>
<td>009</td>
<td>the compiler</td>
<td>PF10</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td>Unassigned</td>
<td>PF11</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>Unassigned</td>
<td>PF12/PF24</td>
</tr>
<tr>
<td></td>
<td>012</td>
<td>Swap</td>
<td>PA1</td>
</tr>
<tr>
<td></td>
<td>013</td>
<td>Unassigned</td>
<td>PA2</td>
</tr>
<tr>
<td></td>
<td>014</td>
<td>Unassigned</td>
<td>CLEAR</td>
</tr>
<tr>
<td></td>
<td>015</td>
<td>Print</td>
<td>ENTER</td>
</tr>
<tr>
<td></td>
<td>016</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cancel changes to current screen/</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reline prior screen</td>
<td></td>
</tr>
<tr>
<td>Application Name</td>
<td>Function</td>
<td>Description</td>
<td>Default Key</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clear work file</td>
<td>PA1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update screen contents and work file OR update work file and execute the compiler</td>
<td></td>
</tr>
</tbody>
</table>

| LINEMODE         | 001      | Page forward                                                      | PA1         |
| Line-mode I/O operations | 002      | Page backward                                                     | PA2         |
|                  | 003      | Page forward or to selected page                                  | ENTER       |
|                  | 004      | Exit line mode                                                    | CLEAR       |

<p>| OLM              | 001      | Display Initial (Map) Definition screen                           | PF1         |
| Batch mapping compiler | 002      | screen                                                            | PF2         |
|                  | 003      | Display Format screen                                             | PF3         |
|                  | 004      | Display Field Selection screen                                    | PF4         |
|                  | 005      | Display Field Edit screen                                         | PF5         |
|                  | 006      | Display Map Image screen                                           | PF6         |
|                  | 007      | Set cursor on Format screen                                       | PF7         |
|                  | 008      | Change session execution mode (STEP/FAST)                         | PF8         |
|                  | 009      | (STEP/FAST)                                                       | PF9         |
|                  | 010      | Propagate fields on Format screen                                 | PF10        |
|                  | 011      | Display Correct/Incorrect                                         | ENTER       |
|                  | 012      | Attributes screen                                                 | CLEAR       |
|                  | 013      | Display Extended Field Edit                                        | PA1         |
|                  | 014      | screen                                                            | PA2         |
|                  | 015      | Update screen contents and/or proceed to next screen               | PF11        |
|                  | 016      | Display previous screen/terminal session                           | PF13        |
|                  |          | Clear current processing/                                          |             |</p>
<table>
<thead>
<tr>
<th>Application Name</th>
<th>Function</th>
<th>Description</th>
<th>Default Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLQ</td>
<td>001</td>
<td>Display Report Processing screen</td>
<td>PF2/PF14</td>
</tr>
<tr>
<td>CA OLQ menu facility</td>
<td>002</td>
<td>Display Signon Database View</td>
<td>PF6/PF18</td>
</tr>
<tr>
<td></td>
<td>003</td>
<td>screen</td>
<td>PF5/PF17</td>
</tr>
<tr>
<td></td>
<td>004</td>
<td>Display Menu screen</td>
<td>PF4/PF16</td>
</tr>
<tr>
<td></td>
<td>005</td>
<td>Display Record Select screen</td>
<td>PF1/PF13</td>
</tr>
<tr>
<td></td>
<td>006</td>
<td>Display Field Select screen</td>
<td>PF3/PF15</td>
</tr>
<tr>
<td></td>
<td>007</td>
<td>Display Display Report screen</td>
<td>PF9/PF21</td>
</tr>
<tr>
<td></td>
<td>008</td>
<td>Display Error screen</td>
<td></td>
</tr>
<tr>
<td></td>
<td>009</td>
<td>Display Help screen</td>
<td></td>
</tr>
<tr>
<td></td>
<td>010</td>
<td>Quit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>Swap to native mode OLQ</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Execute path (Retrieval Completed screen)</td>
<td></td>
</tr>
<tr>
<td>OLQ (cont.)</td>
<td>012</td>
<td>Unassigned</td>
<td>PF12/PF24</td>
</tr>
<tr>
<td></td>
<td>013</td>
<td>Display Path Select screen</td>
<td>ENTER</td>
</tr>
<tr>
<td></td>
<td>014</td>
<td>Display Report Fields - Break/Sort screen</td>
<td>CLEAR</td>
</tr>
<tr>
<td></td>
<td>015</td>
<td>Sort screen</td>
<td>PA1</td>
</tr>
<tr>
<td></td>
<td>016</td>
<td>Display Report Fields - Header</td>
<td>PF2/PF14</td>
</tr>
<tr>
<td></td>
<td>017</td>
<td>screen</td>
<td>PA2</td>
</tr>
<tr>
<td>Application Name</td>
<td>Function</td>
<td>Description</td>
<td>Default Key</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>---------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>018</td>
<td></td>
<td>Display Report Fields - Edit</td>
<td>PF8/PF20</td>
</tr>
<tr>
<td>019</td>
<td></td>
<td>screen</td>
<td>PF7/PF19</td>
</tr>
<tr>
<td>020</td>
<td></td>
<td>Unassigned</td>
<td>PF11/PF23</td>
</tr>
<tr>
<td>021</td>
<td></td>
<td>Unassigned</td>
<td>PF10/PF22</td>
</tr>
<tr>
<td>022</td>
<td></td>
<td>Display Relational Record Menu</td>
<td></td>
</tr>
<tr>
<td>023</td>
<td></td>
<td>screen</td>
<td></td>
</tr>
<tr>
<td>024</td>
<td></td>
<td>Display Print Processing screen</td>
<td></td>
</tr>
<tr>
<td>025</td>
<td></td>
<td>Display Qfile Processing screen</td>
<td></td>
</tr>
<tr>
<td>026</td>
<td></td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>027</td>
<td></td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>028</td>
<td></td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>029</td>
<td></td>
<td>Process screen - step mode</td>
<td></td>
</tr>
<tr>
<td>030</td>
<td></td>
<td>Process screen - fast mode</td>
<td></td>
</tr>
<tr>
<td>031</td>
<td></td>
<td>Display previous screen</td>
<td></td>
</tr>
<tr>
<td>032</td>
<td></td>
<td>Cancel all screen changes</td>
<td></td>
</tr>
<tr>
<td>033</td>
<td></td>
<td>(RESTART)</td>
<td></td>
</tr>
<tr>
<td>034</td>
<td></td>
<td>Display Record List</td>
<td></td>
</tr>
<tr>
<td>035</td>
<td></td>
<td>Display Field List</td>
<td></td>
</tr>
<tr>
<td>036</td>
<td></td>
<td>Display commands (global help)</td>
<td></td>
</tr>
<tr>
<td>037</td>
<td></td>
<td>Display first</td>
<td></td>
</tr>
<tr>
<td>038</td>
<td></td>
<td>Display last</td>
<td></td>
</tr>
<tr>
<td>039</td>
<td></td>
<td>Cancel current screen changes</td>
<td></td>
</tr>
<tr>
<td>040</td>
<td></td>
<td>(RESHOW)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Route control to transfer control facility (SWITCH)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Print current report at default</td>
<td></td>
</tr>
</tbody>
</table>
Application Name | Function Description | Default Key
--- | --- | ---
 | destination |  
 | Display next |  
 | Display prior |  
 | Display right |  
 | Display left |  

### Example KEYS Statement

**Defining a Keys Table with an Application Override**

The following statement creates the keys table COMMTAB, which establishes shared control-key assignments for the IDD and LINEMODE applications and, for IDD, redefines the CLEAR key to invoke function 15:

```
ADD KEYS FOR TABLE COMMTAB
PF7 = FUNCTION 2
PF8 = FUNCTION 1
CLEAR = FUNCTION 4
FOR APPLICATION LINEMODE
FOR APPLICATION IDD
CLEAR = FUNCTION 15.
```

**Clearing an Application-Specific Control-Key Assignment**

The following statement modifies the keys table COMMTAB, clearing the control-key assignment for IDD:

```
MODIFY KEYS FOR TABLE COMMTAB
FOR APPLICATION IDD
NULL.
```

**Disassociating an Application from a Keys Table**

The following statement modifies the keys table COMMTAB, removing the LINEMODE application from the table:

```
MODIFY KEYS FOR TABLE COMMTAB
FOR APPLICATION LINEMODE
DELETE.
```

**Deleting a Keys Table**

The following statement deletes the keys table COMMTAB from the current system definition:

```
DELETE KEYS FOR TABLE COMMTAB.
```
LOADLIST Statement -- Define Load Lists

The LOADLIST statement is used to define load lists. A load list identifies the load libraries and data dictionaries the DC/UCF system is to search for programs to be loaded. The system searches the libraries and dictionaries in the order in which they are named in the load list.

Using load lists, you can:

- Control the location from which programs are loaded at runtime
- Tune system performance by defining load lists with a minimum number of entries
- Concatenate multiple test load libraries (z/OS systems only) and alternate dictionaries in a load list for greater flexibility in a test environment

LOADLIST Statement Syntax

ADD/MODIFY/DELETE LOADLIST Statement

►►─┬──────────┬─  LOAdlist  ─┬─ loadlist-name
[125x424]loadlist-name
[125x416]├─  ADD  ──┤
[125x408]├─  MODify  ──┤
[125x400]└─  DELete  ──┘
[125x384]►─┬─────────────────────────────────────────────────────────┬────────────────►◄
[125x376]│  ┌─────────────────────────────────────────────────────┐  │
[125x368]└─▼─┬───────────┬─┬─  dictionary-specification  ─┬─  V
[125x360]dictionary-specification
[125x352]└─  EXClude  ──┘  ┌─  LOAdlib  is  └─  V
[125x356]LOAdlib
[125x352]nnnn
[125x344]├─  USER-Default  ──┤
[125x344]└─  SYSTEM-Default  ──┘
[125x323]►─┬──────────────────────────────────────┬───────────────────────────────────►
[125x314]└─  DICTNAme  is  ─┬─dictionary-name  ─┬─
[125x304]dictionary-name
[125x296]├─  USER-Default  ──┤
[125x296]└─  SYSTEM-Default  ──┘
[125x323]►─┬───────────────────────────────────────────────┬──────────────────────────►◄
[125x312]└─  DICTNOde  is  ─┬─node-name  ─┬─
[125x304]node-name
[125x296]├─  USER-Default  ──┤
[125x296]└─  SYSTEM-Default  ──┘
[125x314]►─┬─────────────────────────────────────────────────────┬───────────────────►
[125x304]└─  load  module  Version  is  ─┬─1  ─┬─
[125x296]version-number
[125x288]├─  USER-Default  ──┤
[125x288]└─  SYSTEM-Default  ──┘

DISPLAY/PUNCH LOADLIST Statement

►►─┬─  DISPlay  ──┬─load-list-name  ──►
[264x168]load-list-name
[125x160]└─  PUNch  ──┘
[125x144]►─┬───────────────────────────────────────┬──────────────────────────────────►
[125x136]│  ┌───────────────────────────────────┐  │
[125x128]│  │          ┌───────────────┐  │          │
[125x120]└─▼─┬─  WITh  ───────┬─▼─┬─  DETails  ─┬─▼─┴─▼─┘
[125x112]├─  ALSo  ──┤       │       │       │
[125x104]└─  WITHOut  ──┘       │       │       │
[125x96]└─  WITH  ──┘       │       │       │
[125x80]└─  NONe  ──┘       │       │       │
[125x72]►─┬─────────────────────┬────────────────────────────────────────────────────►
[125x72]└─  AS  ──┬─  COMments  ──┘
LOADLIST Statement Parameters

LOAdlist load-list-name

Identifies the load list being added, modified, deleted, displayed, or punched.

Load-list-name must be the one- through eight-character name of the load list.

EXCLUDE

Excludes the following load library or data dictionary from the load list.

dictionary-specification

Specifies a data dictionary to include in the load list.

Expanded syntax for dictionary-specification is shown above immediately following the LOADLIST statement syntax.

LOAdlib is

Identifies a load library to include in the load list.

⚠️ Note: Multiple load libraries are supported under z/OS only.

DICTNAme is

Identifies a data dictionary to include in the load list.

- dictionary-name
  Explicitly specifies a data dictionary.
  Dictionary-name must be the name of a data dictionary included in a segment of the DMCL defined either for the current system or for the system identified by the DICTNODE parameter (described below).

- USER-Default
  Specifies the current session default dictionary established by the DCUF SET DICTNAME command.

- SYSTEM-Default
  Specifies the system default dictionary.

DICTNOde is
Specifies the name of the DC/UCF system that controls the dictionary specified by the DICTNAME parameter.

- **node-name**
  Specifies a node name. 
  *Node-name* must be the name of a DC/UCF system defined in the CA IDMS communications network.

- **USER-Default**
  Specifies the current session default node established by the DCUF SET DICTNODE command.

- **SYSTEM-Default**
  Specifies the current DC/UCF system. 
  SYSTEM-DEFAULT is the default when you omit the DICTNODE parameter from the DICTNAME parameter of the LOADLIST statement.

**load module Version is**

Specifies the version number the DC/UCF system will use when searching for load modules in the data dictionary identified by the DICTNAME parameter.

- **version-number**
  Explicitly specifies a version number. 
  *Version-number* must be an integer in the range 1 through 9,999. The default is 1.

- **USER-Default**
  Specifies the current session default version number established by the DCUF TEST command. 
  If USER-Default is specified, but no DCUF SET TEST is in effect, the entry in the load list is bypassed.

**LOADLIST Statement Usage**

**Defining Loadlists**

You can include any number of load lists in a DC/UCF system definition. You use the SYSTEM statement to designate one load list as the system default. The SYSTEM statement is described in SYSTEM Statement (see page 131).

**System Default Loadlist (SYSLOAD)**

One predefined load list, SYSLOAD, is available to all DC/UCF systems at runtime. SYSLOAD is used as the system default load list if you do not designate a system default during system generation. The SYSLOAD load list has the following definition:

```
ADD LOADLIST SYSLOAD
  DICTNAME IS USER-DEFAULT VERSION IS USER-DEFAULT
  DICTNAME IS SYSTEM-DEFAULT VERSION IS USER-DEFAULT
  LOADLIB IS USER-DEFAULT
  DICTNAME IS USER-DEFAULT VERSION IS 1
  DICTNAME IS SYSTEM-DEFAULT VERSION IS 1
  LOADLIB IS SYSTEM-DEFAULT.
```

**Changing the Loadlist in Effect for a DC/UCF Session**
At runtime, users can vary the load list in effect for the current session by means of the DCUF SET LOADLIST command. DCUF commands are described in CA IDMS System Tasks and Operator Commands section.

**Changing the User-Default Dictionary and Load Library**

The user-default dictionary can be set at runtime

- In a user profile with the dictname parameter
- If not set in the user profile or if the dictname parameter of the user profile can be overridden, you can use the DCUF SET DICTNAME command

The user-default load library and version can be set at runtime using the DCUF TEST command.

**Adding Elements to a Loadlist**

Load libraries and data dictionaries added to a previously existing load list are placed at the end of the list. To insert a library or dictionary before the last entry in a load list, you must first delete all entries beyond the position in which the new entry is to be inserted. Then you can add the new entry and the deleted entries in the appropriate order.

**Program Loading Criteria**

In addition to being defined in a loadlist, programs must also meet one of the criteria below to be loaded (whether they reside in a load library or a load area designated in the load list) into a program pool.

- The specific version of the program must be explicitly defined in a system generation PROGRAM statement.
- The specific version of the program must be explicitly defined at runtime with a DCMT VARY DYNAMIC PROGRAM command.
- The program must be eligible to be loaded with a null PDE. For more information about null PDEs and undefined programs, see SYSTEM Statement (see page 131).

**Example LOADLIST Statements**

**Creating a Loadlist**

The following statement creates load list SYS09TST:

```
ADD LOADLIST SYS09TST
    DICTNAME IS TSTDICT VERSION IS 5
    DICTNAME IS TSTDICT VERSION IS 1
    DICTNAME IS MISDICT
    LOADLIB IS SYSTEM-DEFAULT
    DICTNAME IS SYSTEM-DEFAULT.
```

**Deleting an Element from a Loadlist**

The following statement deletes the MISDICT data dictionary from the SYS09TST load list:
MODIFY LOADLIST SYS09TST
   EXCLUDE DICTNAME MISDICT.

Deleting a Loadlist

The following statement deletes the SYS09TST load list from the system definition:
DELETE LOADLIST SYS09TST.

MAPTYPE Statement -- Creates Alternative Map Table

The MAPTYPE statement creates an alternative map table and builds entries in the table. Alternative map tables are used by the alternative map support feature during runtime mapping operations. Alternative map support allows you to implement similar maps, for example, in different languages.

For more information about alternative map support, see the CA IDMS Mapping Facility section.

MAPTYPE Statement Syntax

MAPTYPE Statement Parameters

MAPtype maptype-name
Identifies the name of the alternative map table.

*Maptype-name* must be a one- through eight-character alphanumeric value.

**MAP map-name-1 maps TO map-name-2**

Adds an entry to the alternative map table. The map identified by *map-name-1* is replaced by the map identified by *map-name-2*, when appropriate, at application runtime. At least one entry must be specified for ADD MAPTYPE statements.

*Map-name-1* is the one- through eight-character name of a map load module invoked by an application program. *Map-name-2* is the one- through eight-character name of the corresponding map that will override *map-name-1* in order to provide compatibility between the application program and the user.

In supplying map names, you can specify a mask for generic translation. The question mark (?) serves as the substitution character. At runtime, each question mark in *map-name-1* will match any character in the corresponding position in *map-name-2*. Conversely, any character in *map-name-1* will match a question mark in the corresponding position in *map-name-2*. If *map-name-1* and *map-name-2* each contain a question mark in the same position, the character in that position in the map name passed by the application program will remain unchanged when the DC/UCF system relates *map-name-1* to *map-name-2*.

If an entry for *map-name-1* already exists in the alternative map table, the system generation compiler replaces the second map in the entry with the new specification for *map-name-2*. If no entry is found, a new entry is added.

Include as many MAP MAPS TO parameters as necessary to replace map load modules invoked by application programs with maps that are compatible with runtime users.

**EXClude MAP map-name-1**

Deletes an entry from the alternative map table.

*Map-name-1* must be the name of a map previously defined in the alternative map table with a MAP MAPS TO parameter.

To delete multiple alternative map table entries, use an EXCLUDE MAP parameter for each map.

This parameter is valid in MODIFY statements only.

**MAPTYPE Statement Usage**

**Defining an Alternative Map Table**

Each entry in an alternative map table consists of a pair of map names. At application runtime, DC/UCF substitutes map load modules invoked by the program with the associated maps named in the table.

**Associating Alternative Map Tables with Users**
An alternative map table can be associated with a user with the MAPTYPE parameter of a user profile. When the user signs on to DC/UCF, the system automatically accesses the associated alternative map table.

For more information about user profiles, see the CA IDMS Security Administration section.

Example MAPTYPE Statement

Creating an Alternative Map Table

The following statement defines an alternative map table named FRENCH and specifies an entry that substitutes any map name beginning with ENGMAP with a map name beginning with FCHMAP. The last two characters of the first map name remain unchanged.

```
ADD MAPTYPE FRENCH
    ENGMAP?? MAPS TO FCHMAP??.
```

Adding an Entry to an Alternative Map Table

The following statement adds an entry to alternative map table FRENCH:

```
MODIFY MAPTYPE FRENCH
    MAP GRKMAP01 MAPS TO FRCMAP01.
```

Deleting an Entry from an Alternative Map Table

The following statement deletes an entry from alternative map table FRENCH:

```
MODIFY MAPTYPE FRENCH
    EXCLUDE GRKMAP01.
```

Deleting an Alternative Map Table

The following statement deletes alternative map table FRENCH:

```
DELETE MAPTYPE FRENCH.
```

NODE Statement -- Defines a Node

The NODE statement identifies the communication method used to access a specified node name in the DC/UCF communications network.

NODE Statement Syntax

```
ADD/MODIFY/DELETE NODE Statement

ADD                   NODe nodename

MOVED                   NODe nodename

DELete                   NODe nodename

cci                       cv-number

svc-number
```

26-Dec-2017
DISPLAY/PUNCH NODE Statement

NODE Statement Parameters

**NODE nodename**

Specifies the node name (SYSTEM ID) of the CA IDMS CV that is accessed from the DC/UCF system being defined.

*Node name* is:

- A unique one- to eight-character alphanumeric value beginning with an alphabetic character.
- Matches the name of a system defined in the DC/UCF communications network.

When using a type of GROUP, the group name replaces the SYSTEM ID in the *nodename* parameter.

**CCI**

For DDS users only, specifies CAICCI, the Common Communications Interface component of the CA Integration Services layer, is used to access the named node.

**CVNumber** is *cv-number*

Specifies the communication method for the named node is a CA IDMS central version and identifies its central version number. The CVNUMBER IS parameter is used with SVC-type connections only.

*CVNumber* is an integer ranging from 0 to 255. The combination of *cv-number* and *svc-number* (described next) must uniquely identify the named node.

**SVC svc-number**

Specifies the SVC number of the CA IDMS CV receiving packets sent by the CA IDMS system currently being defined.

*SVC* is an integer ranging from 0 to 255.

**BLOcksize blk-size** Specifies the size of the packet used to pass data between nodes.

*Blocksize* is the blocksize of the data packet sent between CVs.

The default *blk-size* is 8192.
LOCal

Specifies the named node is the DC/UCF system being defined.

LOCAL is the default.

GROup

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

DEFnault NODe

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

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

- LOCal
  Specifies the node to use is local, which is the current system. The database session is processed by the system being defined.

- NULI
  Specifies there is no default node. If a database session is routed to the named DBGroup, and there are no CVs available to service the request, the database session fails.

VTAm

Requires DDS. Specifies VTAM is the communication method used to access the named node.

VTAM can be specified as the access method for resources residing on different CPUs.

TCP/ip or TCPip

Specifies that the TCP/IP protocol is used to access the named node. This parameter is for DDS users only.

**NODE Statement Usage**

**Defining a Node Table**

At runtime, the node table determines the communication method used to access resources identified in the system generation RESOURCE TABLE statement.

If all database requests in your DC/UCF communications network occur within a single region, you do not have to explicitly code the NODE statement. The system automatically generates a node name entry in the node table of the system being defined with these characteristics:

- The node name is the `system-name` specified on the SYSTEM ID parameter of the SYSTEM statement
The communication method is LOCAL

Defining Remote Nodes

For each DC/UCF system accessing resources located on a remote node, you must:

- Identify the resources and their location (node) using the RESOURCE TABLE statement
- Identify the communication method used by the system to access the node using the NODE statement

Example NODE Statement

**Defining a Node for a Remote DC/UCF System**

ADD NODE SYSTEM84
   CVNUMBER IS 20
   SVC 174
   BLOCKSIZE 8192.

ADD NODE SYSTEM EDCQAM01
   CVNUMBER IS 103
   SVC 173
   BLOCKSIZE 8192.

**Deleting a Node**

DELETE NODE SYSTEM84.

**Defining a DBGroup to the Front-End Definition for CV IDMS070**

MODIFY SYSTEM IDMS070.

ADD NODE EMPGROUP
   GROUP DEFAULT NODE IDMS071.

ADD NODE IDMS071
   VTAM.

GENERATE.

**OLM Statement -- Define OLM Characteristics**

The OLM statement is used to define definition-time and runtime characteristics of the CA IDMS Mapping Facility.

The OLM definition includes the online mapping:

- Special characters used on the MAPC layout screen
- Runtime paging session and field editing specifications

The OLM statement is required if online mapping is to be used with a DC/UCF system.
OLM Statement Syntax

ADD/MODIFY/DELETE OLM Statement

- **ADD**
- **MODify**
- **DELETE**

- **Data field character is** `X'6D'` (C'`data-field-character`' or `X'hex-data-field-character'`)
- **Delimiter character is** `X'1C'` (C'`delimit-character`' or `X'hex-delimit-character'`)
- **Field Select character is** `X'6C'` (C'`field-select-character`' or `X'hex-field-select-character'`)
- **Field Start character is** `X'1E'` (C'`field-start-character`' or `X'hex-field-start-character'`)
- **Help PFKey is** PF1
- **Numeric field Decimal-point is** COMma or PERiod
- **Numeric field Order is** STAndard or REVersed
- **New Copy is** YES or NO
- **Page Backward PFKey is** PF7
- **Page Forward PFKey is** PF8
- **PAGING Storage is** 10 paging-storage-size
- **QueuE Retention is** 255 queue-retention-period
- **Translate character is** C '@' or C'`translate-character'`

DISPLAY/PUNCH OLM Statement

- **Display all OLM**
- **Punch**

- **With Also With Without**
- **Details History**
- **All None**
- **As Comments**
OLM Statement Parameters

**DATa field character is**

Identifies the keyboard character used to show the location and length of each variable field on the OLM Format and Map Image screens.

- `C'data-field-character'`
  Specifies the data field character as a character literal.

- `X'hex-data-field-character'`
  Specifies the data field character as a hexadecimal literal. The default is X'6D', the underscore (_) character.

**DELimit character is**

Identifies the keyboard character used to mark the end of a delimited variable field on the OLM Format screen.

- `C'delimit-character'`
  Specifies the delimit character as a character literal.

- `X'hex-delimit-character'`
  Specifies the delimit character as a hexadecimal literal. The default is X'1C' (the DUP character).

**FIELD SElect character is**

Identifies the keyboard character used to select a map field on the OLM Format screen for editing or deletion.

- `C'field-select-character'`
  Specifies the field select character as a character literal.

- `X'hex-field-select-character'`
  Specifies the field select character as a hexadecimal literal. The default is X'6C', the percent (%) sign.

**FIELD STArt character is**

Identifies the keyboard character used to mark the beginning of a map field on the OLM Format screen.

- `C'field-start-character'`
  Specifies the field start character as a character literal.
- **X’hex-field-start-character’**
  Specifies the field start character as a hexadecimal literal. The default is X’1E’ (the FIELD MARK key).
  **UCF/TSO environment:** In TSO, use of the default field start character of X’1E’ (FIELD MARK) will cause display problems when using the online mapping compiler (MAPC task code). Choose another value for the field start character in this environment.

**HELP PFKEY is PF1/PFnn**

Specifies the default program function key to be associated with the help function during runtime mapping sessions.

*Nn* is an integer in the range 1 through 24.

PF1 is the default.

**NUMeric field DECimal-point is**

Establishes the character to be interpreted as the decimal point for numeric data fields at runtime.

⚠️ **Note:** Map developers must enter external pictures with the appropriate decimal character when defining numeric map data fields.

**NUMeric field ORDer is**

Establishes international editing options for numeric map data fields at runtime.

- **STandard**
  Specifies that runtime mapping will not reverse the data contained in a numeric data field. **STANDARD** is the default when you omit the NUMERIC FIELD ORDER parameter from the OLM statement.

- **REversed**
  Specifies that runtime mapping will reverse all data contained in a numeric data field at both mapout and mapin. This option might be used, for example, in Hebrew-language applications, where textual information is read from right to left.

⚠️ **Note:** If you specify ORDER IS REVERSED, IBM®-supplied 3276 and 3278 device modifications must be installed. With these modifications, the cursor will move from left to right in numeric fields and from right to left in alphanumeric fields.

**NEW COPY is**

Specifies whether a map is to be marked as new copy automatically when the map load module is generated.
- **YES**
  Indicates the map is marked as new copy automatically.

- **NO**
  Indicates the map is not marked as new copy automatically. The user must issue a DCMT VARY PROGRAM NEW COPY statement for the map to make the load module eligible for loading. NO is the default when you omit the NEW COPY parameter from the OLM statement.

**PAGe BACKward pfkey is PFn**

Specifies the default program function key (PF key) associated with the paging backward function during runtime paging sessions.

\[N\] must be a valid PF-key number in the range 1 through 24. The default is 7.

**PAGe FORward pfkey is PFn**

Specifies the default program function key (PF key) to be associated with the paging forward function during runtime paging sessions. \[N\] must be a valid PF-key number in the range 1 through 24. The default is 8.

**PAGing STOrage is paging-storage-size**

Specifies the maximum amount of storage, in 1K bytes, to be made available in the scratch area (DDLDCSCR) of the data dictionary to maintain a paging session at map runtime. At runtime, the system loads paging storage with the header, footer, and detail occurrences to be displayed on the map.

\[Paging-storage-size\] must be an integer in the range 0 through 32,767. The default is 10.

**QUEue RETention is queue-retention-period**

Specifies the amount of time, in days, the system is to retain the records from a suspended OLM map-definition session in the queue area. When the queue retention period is exceeded, the queue record is deleted automatically at system startup.

\[Queue-retention-period\] must be an integer in the range 0 through 255. The default, 255, directs the system not to delete the queue based on a retention period.

**TRAnslate character is translate-character**

Assigns a translation character to mask bad data into a displayable character.

\[Translate-character\] must be a character literal. The default is C'@'.

The translation character must be displayable. If you specify either C' ' (blank) or C'' (null), the translation is bypassed at program execution.

---

**OLM Statement Usage**

**Efficient Use of the Queue Area**
To make most efficient use of the queue area, specify a queue retention period that is as short as possible.

Example OLM Statement

Defining OLM Characteristics

The following statement defines OLM definition-time characteristics and the mapping facility runtime environment:

ADD OLM
FIELD START CHARACTER IS X'1E'
FIELD SELECT CHARACTER IS C,'%'
DELIMIT CHARACTER IS X'1C'
DATA CHARACTER IS C'
NUMERIC FIELD DECIMAL-POINT IS PERIOD
PAGE FORWARD PFKEY IS PF8
PAGE BACKWARD PFKEY IS PF7
PAGING STORAGE IS 12
QUEUE RETENTION IS 100.

Changing the OLM Execution Mode

The following statement changes the

MODIFY OLM
PAGING STORAGE IS 20.

Deleting the OLM Definition

The following statement deletes the mapping facility definition from the data dictionary:

DELETE OLM.

For more information about calculating the amount of storage required by pageable maps, see the CA IDMS Mapping Facility section.

OLQ Statement -- Define OLQ Runtime Environment

The OLQ statement is used to define the CA OLQ runtime environment. The OLQ statement is required if CA OLQ is used with the system being generated.

For more information about CA OLQ, see the CA OLQ Reference section and the CA OLQ Using section.

OLQ Statement Syntax

ADD/MODIFY/DELETE OLQ statement

| ADD | OLQ | MOD |
OLQ Statement Parameters

**ACCess is**

Specifies whether CA OLQ or CA IDMS will process SQL statements used to access a CA IDMS database.

- **IDMssql**
  Indicates CA IDMS will process the SQL statements.
  IDMssql, IDMs sql, and IDMS-SQL are synonyms and used interchangeably.

- **OLQsql**
  Indicates CA OLQ will process the SQL statements.
  OLQ SQL is the default.
  OLQsql, OLQ sql, and OLQ-SQL are synonyms and used interchangeably.

**BATch CLAss is** *batch-class*

For z/OS systems only, specifies the print class used by CA OLQ when submitting batch jobs.

*Batch-class* must be an integer from 0 - 64.

The default value of 0 indicates that online submission of OLQ batch jobs is not allowed. Assign a print class from 1 - 64 if you want to allow online submission. Verify that the print class is assigned to a printer associated with a SYSOUTL line.

For more information on submitting JCL via the internal reader, see the section "Writing JCL to a JES2 Internal Reader" in the *Navigational DML Programming section*.

**COMment character is** '*comment-character'*

Specifies the character that flags a remark. All text following the comment character is ignored during execution.

*Comment-character* must be a one-character alphanumeric value enclosed in site-standard quotation marks. The default is the semicolon (;).
CONtinuation character is 'continuation-character'

Specifies the continuation character to use in q-files or batch input when the length of a CA OLQ command is greater than one line. The continuation character is specified at the end of each line that is to be continued.

Continuation-character must be a one-character alphanumeric value enclosed in site-standard quotation marks. The default is the hyphen (-).

INPut line size is input-line-size

For 3270 terminals only, specifies the number of lines on the screen that are available for input.

Input-line-size must be an integer in the range 1 through 256. The default is 2. The maximum number of lines defined by input-line-size is limited to the total number of lines on the screen minus the number of lines reserved for output.

INTernal storage page size is internal-storage-page-size

Specifies the size, in bytes, of the CA OLQ internal storage pages used to store control data across a pseudo-converse. These pages are written to the data dictionary scratch area (DDLDCSCR) and are returned to main memory when the CA OLQ task thread resumes execution.

Internal-storage-page-size must be an integer in the range 1 through 32,767. The default is 1,920.

INTERRupt count is interrupt-count

Specifies the number of records CA OLQ will read before issuing the following message at runtime:

CONtinue YES/NO?

  nnn RECORDS READ  nnn RECORDS SELECTED
  nnn DATA ERRORS

Interrupt-count must be an integer in the range 1 through 9,999. The default is 100.

MAXimum INTERRupt count is max-interrupt-count

Indicates the maximum interrupt count that a user can specify at runtime.

Max-interrupt-count must be an integer in the range 0 through 32,767. The default is 32,767. A value of 0 indicates the user-specified interrupt count is unlimited.

Typically, the maximum interrupt count is greater than or equal to the value specified by the INTERRUPT COUNT parameter (described above).

MAXimum report COUnt is max-report-count

Specifies the maximum number of reports that are saved in the queue area for each user.

Max-report-count must be an integer in the range 0 through 32,767. The default is 5.

MAXimum report PAGes is max-report-size

Specifies the maximum length, in pages, of a CA OLQ report written to the queue area.
Max-report-size must be an integer in the range 0 through 32,767. The default is 5.

When a CA OLQ report exceeds the MAXIMUM REPORT PAGES specification, the system ends the retrieval process and stops writing report pages. The report exists in the queue area and is available for viewing, but is incomplete. If a user attempts to sort the data or to display the last page of the report, the system returns a message indicating that an unexpected end of report was encountered.

MAXimum report RETention is

Defines the maximum length of time a user can specify when saving a CA OLQ report in the queue area. When the user issues a SAVE command and specifies a report retention period, the system automatically uses the maximum report retention period. If the specified retention period is greater than the maximum retention period, it is overridden and replaced by the maximum retention period. Reports that exceed their retention period in the queue area are deleted automatically at system startup.

- max-report-retention-period
  Specifies the maximum report retention period in days. Maximum-report-retention-period must be an integer in the range 0 through 255. The default is 5. A value of 255 is synonymous with FOREVER.

- FOREVER
  Directs the system not to delete reports based on a retention period.

MAXimum SORt storage is max-sort-storage-size

Specifies the maximum amount of storage, in K bytes, that CA OLQ can use for sort operations.

Max-sort-storage-size must be an integer in the range 0 through 32,767. The default is 100.

Important! Specifying too low a value will prevent CA OLQ from performing sort operations.

MENu mode

Specifies whether the menu facility is available to users at runtime.

- ONLY
  Specifies that CA OLQ users must use the menu facility.

- ALLOWed
  Specifies that CA OLQ users can use the menu facility. ALLOWED is the default when you omit the MENU MODE parameter from the CA OLQ statement.

- DISSallowed
  Specifies that CA OLQ users cannot use the menu facility.

olq transaction identifier is task-code

Specifies the task code that invokes the CA OLQ runtime system.
Task-code must be a task code previously defined in the data dictionary with a TASK statement. The default is CA OLQ.

This parameter is required to create an executable system.

PFKey MODule is qfile-name

Identifies the q-file that is executed each time a user invokes CA OLQ.

Qfile-name must be the name of a q-file defined in the data dictionary.

PRInt LINe size is print-line-size

Specifies the line length, in characters, for CA OLQ reports output on TTY-type terminals.

Print-line-size must be an integer in the range 20 through 255. The default is 80.

PRInt PAGe size is print-page-size

Specifies the page length, in lines, for CA OLQ reports output on TTY-type terminals.

Print-page-size must be an integer in the range 5 through 127. The default is 60.

REPort DICtname is dictionary-name

Identifies the data dictionary in which catalog information about CA OLQ saved reports is stored.

Dictionary-name must be the name of a data dictionary in the database name table defined for the system.

REPort FILe page size is report-file-page-size

Specifies the size, in bytes, of CA OLQ report file pages written to the queue area.

Report-file-page-size must be an integer in the range 256 through 32,767. The default is 4,000 bytes.

The specified page size must be:

- Large enough to accommodate the largest database or logical record placed in the report file
- At least 100 bytes smaller than the page size of the queue area

REPort RETention is

Specifies the length of time the system is to retain a CA OLQ report in the queue area. The system automatically uses the REPORT RETENTION value when a user issues a SAVE command without specifying a report retention period. Reports that exceed their retention period in the queue area are deleted automatically at system startup.
- **report-retention-period**
  Specifies the report retention period in days.
  *Report-retention-period* must be an integer in the range 0 through 255. The default is 1. A value of 255 is synonymous with FOREVER.

- **FORever**
  Directs the system not to delete reports based on a retention period.

SEParator character is *separator-character*

Specifies the command separation character used to concatenate CA OLQ commands.

*Separator-character* must be a one-character alphanumeric value enclosed in site-standard quotation marks. The default is the exclamation point (!).

### OLQ Statement Usage

#### Overriding the ACCESS IS Parameter

You can override the ACCESS IS parameter using the following:

- The **IDD USER statement** allows you to override which product (CA OLQ or CA IDMS) will process SQL statements on a user-by-user basis.

- The **CA OLQ SET command** allows you to override which product (CA OLQ or CA IDMS) will process SQL statements for a specific CA OLQ session.

#### Efficient use of the Queue Area

To make most efficient use of the queue area, specify a *report retention* period that is as short as possible.

### Example OLQ Statement

#### Defining the CA OLQ Environment

The following statement creates the OLQ system OLQ10:

```
ADD OLQ
  OLQ ID IS OLQ10
  INTERNAL_STORAGE_PAGE SIZE IS 3552
  INTERRUPT COUNT IS 50
  PRINT LINE SIZE IS 80
  PRINT PAGE SIZE IS 60
  REPORT FILE PAGE SIZE IS 2208.
  REPORT RETENTION IS 1.
  MAXIMUM REPORT RETENTION IS 6.
  CONTINUATION CHARACTER IS '+'.
```

#### Modifying the CA OLQ Environment

The following statement modifies OLQ10 by changing the interrupt count:
MODIFY OLQ
   INTERRUPT COUNT IS 100.

Deleting the CA OLQ Environment

The following statement deletes OLQ10:
DELETE OLQ.

PROGRAM Statement -- Defines and Associates a Program

The PROGRAM statement is used to define a program and associate it with a system. DC/UCF programs are subschemas, database procedures, and system-supplied programs (for example, IDMSOCF). DC/UCF programs are also maps, edit and code tables, SQL access modules, CA ADS, and user programs written in COBOL, PL/I, and Assembler.

Each program the DC/UCF system uses must exist as a load module, and must be known to the runtime system. Programs are identified to the runtime system in any of the following ways:

- Manually by defining them in the data dictionary using the system generation PROGRAM statement or the DC OPTION of the DDDL compiler PROGRAM statement.
- Providing for their automatic definition in the data dictionary at runtime. You enable automatic program definition with the system generation SYSTEM statement UNDEFINED PROGRAM COUNT parameter.
- Defining programs dynamically at runtime using the DCMT VARY DYNAMIC PROGRAM command.

You must code one PROGRAM statement to define each of the following programs to the system:

- All database procedures, including CA-supplied database procedures (for example, IDMSCOMP for data compression and IDMSDCOM for data decompression) and user-written procedures.
- All sub-schemas that are not compiled into the data dictionary with the sub-schema compiler.
- All CA-supplied programs that are associated with the online components and system tasks included in the system definition. For more information, see System Programs and Tasks (see page 355).
- All user-written programs that are invoked through SQL as table procedures, procedures, or functions.

PROGRAM Statement Syntax

ADD/MODIFY/DELETE PROGRAM Statement
The defaults shown can be overridden with the DEFAULT PROGRAM statement. Additionally, users can override certain PROGRAM statement parameters at runtime with the DCMT VARY PROGRAM command. The allowable overrides are noted in the syntax rules for the applicable parameters. For a complete description of overriding PROGRAM statement defaults, see DEFAULT PROGRAM Statement (see page 193).

```
PROgram program-name

ADD
MODify
DELeTe

Version is version-number

ADSo dialog STAtistics ON OFF

CONcurrent NONconcurrent

ACCESS MODule
DIALog
MAP
map HELp
PROGRAM
SUBschema
TABLE

DICTName is SYSTEM-default
LOADlib

DUMP threshold is dump-count

DYNamic NODynamic

ENAbled DISabled

ERROR threshold is error-count

ISA size is initial-storage-size

LANguage is ADSo ASSemblery COBol PLI PL1

MAInline NOMAinline

MPMode is SYStem ANY

MULTiple ENClave is ON OFF

NEW COPy is ENAbled DISabled

OVErlayable
```
PROGRAM Statement Parameters

**PROgram program-name**

Specifies the name of the program being added, modified, or deleted.

*Program-name* must be the one- through eight-character name of a program load module. *Program-name* must begin with an alphabetic or national (#, $, @) character.

**Version is version-number**

Qualifies the named program with a version number.

*Version-number* must be an integer in the range 1 through 9,999. The default is 1.

**ADSo dialog STAtistics**

For CA ADS dialogs only, specifies whether statistics are collected for this CA ADS dialog.
CONcurrent

Specifies the program can be used by multiple run units and/or tasks at the same time. If the program is reentrant or quasireentrant, one copy of the program is used to process all requests. If the program is nonreentrant, as many copies of the program are used as necessary to process requests concurrently.

CONCURRENT is the default when you specify neither CONCURRENT nor NONCURRENT in the PROGRAM statement.

Note: The system automatically assigns the default value CONCURRENT to maps and subschemas.

NONconcurrent

Specifies the program can be used by only one run unit or task at a time.

ACCESS MODULE

Identifies the program as an access module.

An access module is the executable form of the SQL statements that a program issues when using the CA IDMS SQL Option.

DIALOG

Identifies the program as a CA ADS dialog.

MAP

Identifies the program as a map.

map HELP

Identifies the program as a help module for use with the CA IDMS Mapping Facility.

PROGRAM

Identifies the program as either an executable program or a user-defined table.

SUBSCHEMA

Identifies the program as a subschema.

TABLE

Identifies the program as an edit or code table.
Identifies the program as an edit or code table.

**DICTName is SYSTEM-default**

Indicates the program resides in the load area (DDLDCL0D) of the default dictionary. (Programs that reside in the load area of an alternate dictionary cannot be defined during system generation.)

**LOADlib (default)**

Indicates the program resides in a load library. Under z/OS, the load library is identified by the program version number. For example, a program with a version number of 2 resides in the V0002 load library.

LOADLIB is the default when you specify neither the DICTNAME nor the LOADLIB parameter in the PROGRAM statement.

**Dump threshold is dump-count**

Specifies the number of times that a dump is to be taken for program check errors that occur in the program. A memory dump is taken each time a program check occurs up to and including the value specified by `dump-count`. When this limit is reached and additional program check errors occur, the task that uses this program is terminated abnormally with no memory dump.

`dump-count` must be an integer in the range 0 through 255. The default is 0. The default of 0 indicates that a dump will be produced every time a program check occurs.

⚠️ **Note:** Users can override this parameter at runtime with the DCMT VARY PROGRAM DUMP THRESHOLD command. For more information about DCMT commands, see the CA IDMS System Tasks and Operator Commands section.

**DYNamic**

Allows users to define additional versions of the program at runtime either by means of the DCMT VARY DYNAMIC PROGRAM command or, if the program is eligible for automatic definition, through the definition of null PDEs on the SYSTEM statement.

DYNAMIC is the default when you specify neither DYNAMIC nor NODYNAMIC in the PROGRAM statement.

**NODynamic**

Prevents users from defining additional versions of the program at runtime. Additionally, the system generation compiler ensures that only one version of the program is included in the system definition.

**ENAbled**

Indicates the program is enabled at system startup.
ENABLED is the default when you specify neither ENABLED nor DISABLED in the PROGRAM statement.

DISABLED

Indicates the program is not enabled at system startup. A DC/UCF system cannot access a disabled program.

⚠️ **Note:** Users can override the ENABLED/DISABLED parameter at runtime with the DCMT VARY PROGRAM ENABLE/DISABLE command.

**ERROR threshold is error-count**

Specifies the number of program check errors that can occur before the program is disabled by the system. When the value specified by error-count is exceeded, the program is not executed and tasks attempting to use the program are terminated abnormally.

Error-count must be an integer in the range 1 through 255. The default is 5.

⚠️ **Note:** Users can override this parameter at runtime with the DCMT VARY PROGRAM CHECK THRESHOLD command. **Note:** During startup, the system ignores the values for programs ADSOMAIN and ADSORUN1. These programs are assigned an unlimited error count.

**ISA size is initial-storage-size**

For Assembler and PL/I programs only, specifies the amount of storage, in bytes, allocated for the program's initial storage area (ISA). If an ISA is specified, GET STORAGE statements are not required in the program because the DC/UCF system will automatically allocate the requested storage when the program begins executing. The storage address is passed in register 11.

Initial-storage-size must be an integer in the range 0 through 2,147,483,647. The default is 0.

**LANguage**

Identifies the language in which the program is written.

⚠️ **Note:** The system automatically assigns the default value ASSEMBLER to maps and subschemas.

**MAInline**
For CA ADS dialogs only, indicates the dialog is a mainline dialog. Dialogs defined as MAINLINE are entry points into applications. The names of mainline dialogs are eligible for display on the CA ADS menu screen if so allowed by ADSO statement specifications (described under ADSO Statement.).

If you specify MAINLINE, the dialog must be generated with the MAINLINE attribute but does not have to be assigned a task code during system generation.

- **MENU**
  Indicates the dialog you are defining will appear on the CA ADS menu screen.

- **NOMENU**
  Indicates the dialog you are defining will not appear on the CA ADS menu screen.

**NOMainline**

For CA ADS dialogs only, indicates the dialog is not a mainline dialog.

NOMAINLINE is the default when you specify neither MAINLINE nor NOMAINLINE in the PROGRAM statement.

**MPMode is**

Specifies the multiprocessing mode (MPMODE) for the program.

- **SYStem**
  Directs the DC/UCF system to assign an MPMODE to the program at execution time. SYSTEM is the default when you omit the MPMODE parameter from the PROGRAM statement.

- **ANY**
  Assigns an MPMODE of ANY to the program. ANY is appropriate for reentrant and quasireentrant programs that are defined without storage protection.

  **Note:** If you allow programs to run with an MPMODE of ANY, you should install mechanisms and standards to ensure the programs do not simultaneously update shared storage.

**MULtiple ENClave is**

Specifies whether the program can share the same high level Language Environment process/enclave with other programs in the same task.

- **ON**
  Specifies the program can share the same process/enclave with other programs. This option is ignored unless enclave sharing is enabled for the system. ON is the default.

- **OFF**
  Specifies that the program must use its own unshared process/enclave.

**NEW COPy is**

Specifies whether the new copy facility is enabled for the program or subschema.
- **ENABLEd**
  Enables the new copy facility for the program or subschema. If you specify NEW COPY IS ENABLED, users can issue DCMT VARY PROGRAM NEW COPY commands for the program. If the program is a subschema, users can also issue DC/UCF operator VARY SUBSCHEMA NEW COPY commands.
  ENABLED is the default when you omit the NEW COPY parameter from the PROGRAM statement.

- **DISABLEd**
  Disables the new copy facility for the program or subschema.

**OVERlayable**

Specifies the program can be overlaid in the program pool. You should specify OVERLAYABLE only for executable programs invoked through normal DC mechanisms (for example, LINK or XCTL).

OVERLAYABLE is the default when you specify PROGRAM or LANGUAGE IS COBOL, PLI, or ASSEMBLER in the PROGRAM statement.

**NONOOverlayable**

Specifies the program cannot be overlaid in the program pool. You should specify NONOVERLAYABLE for nonexecutable programs (for example, tables) to prevent such programs from being overwritten in the program pool while they are in use.

NONOVERLAYABLE is the default when you specify MAP, SUBSCHEMA, or TABLE in the PROGRAM statement.

**Important!** You must specify the NONOVERLAYABLE option for program RHDCMT00.

**PROTect**

Enables the storage protection feature for the specified program. Storage protection cannot be activated, however, unless the SYSTEM statement also specifies the PROTECT option.

PROTECT is the default when you specify neither PROTECT nor NOPROTECT in the PROGRAM statement.

**Note:** For more information about storage protection, see Storage Protection (see page 75).

**NOProtect**

Indicates the storage protection feature is not enabled for the program.
Note: Users can override the PROTECT/NOPROTECT specification at runtime with the DCMT VARY PROGRAM STORAGE PROTECT command.

QUAsireentrant

For COBOL programs only, specifies the program is quasireentrant. To be declared quasireentrant, a program must not modify its own code unless the program ensures the modified code is returned to its original state when the program is not in control. Quasireentrant programs are permitted to use working storage because each time the program is executed the system creates a separate copy of its working storage in the storage pool. This technique makes the program, in effect, reentrant.

QUASIREENTRANT is the default for COBOL programs.

REEEntrant

Specifies the program is reentrant. To be declared reentrant, the program must acquire all variable storage dynamically and must not modify its own code.

REENTRANT is the default for all non-COBOL programs.

Note: The system automatically assigns the default value REENTRANT to maps and subschemas.

NONReentrant

Specifies the program is nonreentrant. Programs that modify their own code and do not ensure the modified code is returned to its original state when the program is not in control must be declared NONREENTRANT.

RESident

Specifies the program is made resident at system startup. Resident programs remain in the DC/UCF region/partition for the duration of system execution.

NONRESident

Specifies the program is not made resident. Nonresident programs are loaded into a program pool as needed and are deleted from the program pool when they are not in use, as dictated by demands for space.

NONRESIDENT is the default when you specify neither RESIDENT nor NONRESIDENT in the PROGRAM statement.

REUSable

Specifies the program can be executed repeatedly. When a request to load the program is issued, the system will load a copy of the program from external storage only if no copy exists in the program pool.
To be declared REUSABLE, a program must return all modified code to its original state for each execution. Generally, code is returned to its original state either at the start of a new execution of the program or at the finish of the previous execution. By definition, reentrant and quasireentrant programs are always reusable; however, reusable programs are not necessarily reentrant or quasireentrant.

REUSABLE is the default when you specify neither REUSABLE nor NONREUSABLE in the PROGRAM statement.

**Note:** The system automatically assigns the default value REUSABLE to maps and subschemas.

**NONREUsable**

Specifies that the program cannot be executed repeatedly. When a request to load the program is issued, the system will load a copy of the program from external storage. Programs that modify their own code without returning the code to its original state must be declared NONREUSABLE.

**SAVEarea**

For Assembler programs only, specifies the system will acquire a save area automatically before each execution of the program. The save area address is passed to the program in register 13. You should specify SAVEAREA or accept it by default if the program uses normal IBM calling conventions and, at the start of execution, saves registers in the save area.

SAVEAREA is the default when you specify neither SAVEAREA nor NOSAVEAREA in the PROGRAM statement.

**NOSavearea**

For Assembler programs only, specifies the system will not acquire a save area for the program automatically.

**Note:** The system automatically assigns the default value NOSAVEAREA to maps and subschemas.

---

**PROGRAM Statement Usage**

**Manual Program Definition**

The system generation compiler populates the data dictionary with a program source record for each PROGRAM statement input to the compiler. At generate time, these source records are copied to object records. At system startup, the system uses the information in the object record to build a program definition element (PDE) for each program. The PDE contains information used by the system at runtime.
Automatic Program Definition

For each program load module output by a CA IDMS compiler (that is, map, edit or code table, access module, subschema, or dialog), a program source record is added to the dictionary. The first time a task or program issues a request to load a map, edit/code table, access module, subschema, or dialog for which no PDE exists, the system allocates a PDE for that program from the null PDE pool. The system stores information contained in the program’s source record in the allocated null PDE.

The system then attempts to load the program:

- If the program is found, the null PDE is used to load the program. The null PDE then remains allocated to service subsequent load requests that specify the same program.

- If the program cannot be found, the null PDE is returned to the null PDE pool.

You enable automatic program definition during system generation with the SYSTEM statement UNDEFINED PROGRAM COUNT parameter, which directs the system to allocate a null PDE pool and specifies the size of the pool. For more information about automatic program definition, see the description of the UNDEFINED PROGRAM COUNT parameter in SYSTEM Statement (see page 131).

Dynamic Program Definition

The system uses information supplied in the DCMT VARY DYNAMIC PROGRAM command to build a PDE for the program. Programs defined dynamically exist only for the duration of system execution and have no effect on the system; no definitions are stored in the data dictionary. Typically, dynamic program definition is used in a test environment to accommodate new programs without shutting down the entire system. All programs are eligible for dynamic definition, provided they are not already defined to the system.

For more information about dynamic program definition, see the CA IDMS System Operations section.

Describing the Location of Program Load Modules

Program load modules are stored either in the DDLDCLOD area of a data dictionary or in system load libraries:

- **DDLDCLOD** contains load modules created by CA IDMS compilers (for example, access modules, subschemas, maps, edit and code tables, and dialogs).

- **Load (core-image) libraries** contain load modules for database procedures, system programs, and user programs. Additionally, these libraries can contain any DDLDCLOD-eligible modules that have been punched from DDLDCLOD by using the IDD DDDL compiler and link edited into the appropriate library.

Selecting Dialogs to Appear on the CA ADS Menu Screen

The MENU/NOMENU option of the MAINLINE/NOMAINLINE parameter allows you to specify which mainline dialogs will appear on the CA ADS menu screen. CA ADS APPC dialogs, which are defined as mainline dialogs, do not have to automatically appear on the CA ADS menu screen. CA ADS APPC users can specify, on a dialog-by-dialog basis, which mainline dialogs are to appear on the CA ADS menu screen.

Multithreaded Programs
Reentrant and quasireentrant programs can be multithreaded; that is, they can execute multiple requests concurrently through a single copy of code. Nonreentrant programs are always single-threaded; that is, they can execute only one request for each copy of the code.

**Multiple Enclave**

The performance of certain online applications which use COBOL programs under the IBM Language Environment can be improved significantly by specifying MULTIPLE ENCLAVE ON. These are the restrictions on what COBOL programs can participate in a multiple-program enclave:

1. No enabled program can do a DC RETURN DML call and then be reentered via a subsequent TRANSFER. Note that this restriction does not apply to a program which contains a DC RETURN with no subparameters. That is because the DML compiler generates a GOBACK for such a statement. The restriction does apply if the DC RETURN statement does have subparameters. For example, it is not valid to execute a "DC RETURN NEXT TASKCODE ..." statement and then reenter the same program in the same task.

2. No enabled program can issue a TRANSFER CONTROL NORETURN or a TRANSFER CONTROL XCTL.

3. Optional bit 196 is ignored for programs which are participating in a MULTIPLE USE enclave. Therefore, any program which depends on the use of optional bit 196 must have MULTIPLE ENCLAVE OFF.

**Example PROGRAM Statement**

**Accepting Default Values**

The following statement defines program XYZ. All PROGRAM statement defaults are accepted.

ADD PROGRAM XYZ.

**Overriding Default Values**

The following statement adds program ABC:

ADD PROGRAM ABC LANGUAGE IS ASSEMBLER REENTRANT NOSAVEAREA

**Modifying a Program**

The following statement modifies program ABC by disallowing the new copy facility:

MODIFY PROGRAM ABC NEW COPY IS DISABLED.

**Deleting a Program**

The following statement deletes program ABC:

DELETE PROGRAM ABC.
QUEUE Statement -- Defines DC/UCF System Queues

The QUEUE statement is used to define queues to a DC/UCF system. A queue is a disk work area that is shared by all tasks on all DC/UCF terminals and by batch programs. Queues might be used to collect records for subsequent printing at a centrally located printer. Entries, or queue records, are directed to queues by user programs. You do not have to define queues during system generation in order for user programs to allocate and access queue records. However, queues defined with the QUEUE statement can be assigned the following processing enhancements:

- When the number of entries in a queue reaches a specified threshold, the DC/UCF system will invoke a task defined to process the queue entries.
- When the number of entries in a queue reaches a specified upper limit, the DC/UCF will not permit further entries to be directed to the queue.

Note: Data sharing groups provide the ability to share DC queues between members of the group. This enables online tasks executing in different group members to communicate with one another, as if they were executing within the same CA IDMS system. For more information, see the CA IDMS System Operations section.

QUEUE Statement Syntax

**ADD/MODIFY/DELETE QUEUE Statement**

```
ADD  MODIFY  DELETE

queue-id

Version is 1

ENAbled

INVokes task task-code

RETention is queue-retention-period

THReshold is 1

UPper limit is 0
```

**DISPLAY/PUNCH QUEUE Statement**

```
DISPLAY  PUNch

queue-id

Version is 1
```
QUEUE Statement Parameters

**QUEUE queue-id**

Specifies the queue identifier.

*Queue-id* must be a one- through 16-character alphanumeric value.

**Version is version-number**

Qualifies the queue with a version number.

*Version-number* must be an integer in the range 1 through 9,999. The default is 1.

Although multiple versions of a queue can exist in the data dictionary, the DC/UCF system uses only one version of a queue at runtime. If multiple ADD QUEUE statements specify the same queue identifier, the system uses the first statement that appears in the system definition.

**ENAbled**

Specifies the queue is be enabled when the DC/UCF system starts up.

ENABLED is the default when you specify neither ENABLED nor DISABLED in the QUEUE statement.

**DISabled**

Specifies the queue is not enabled when the DC/UCF system starts up.

- **Note:** Users can override the ENABLED/DISABLED parameter at runtime for the duration of system execution with the DCMT VARY QUEUE command.

**INVokes task task-code**

Specifies the task the DC/UCF system invokes when the number of entries in the queue reaches the limit specified in the THRESHOLD parameter (see below).
Task-code must identify a task previously defined in the data dictionary with a TASK statement.

The INVOKES TASK parameter is required for ADD operations.

⚠️ **Note:** Users can override this parameter at runtime for the duration of system execution with the DCMT VARY QUEUE command.

**RETention is**

Specifies the amount of time the DC/UCF system is to retain the queue. The retention period begins when the first record is stored in the queue.

- **queue-retention-period**
  
  Specifies the queue retention period in days.
  
  Queue-retention-period must be an integer in the range 1 through 255. A retention period of 255 is synonymous with FOREVER. A retention period of 0 defaults to 1.

- **FOREver**
  
  Directs the DC/UCF system not to delete the queue based on a retention period.
  
  FOREVER is the default when you omit the RETENTION parameter from the QUEUE statement.

**THReshold is queue-entry-count**

Specifies the number of entries that must exist in the queue before the DC/UCF system invokes the task specified in the INVOKES TASK parameter to process the entries.

Queue-entry-count is an integer in the range 1 through 32,767. The default is 1.

⚠️ **Note:** Users can override this parameter at runtime for the duration of system execution with the DCMT VARY QUEUE command.

**UPper limit is maximum-entry-count**

Specifies the maximum number of entries permitted in the queue.

Maximum-entry-count is an integer in the range 1 through 32,767. The default, 0, directs DC/UCF not to monitor the number of entries in the queue.

⚠️ **Note:** Users can override this parameter at runtime for the duration of system execution with the DCMT VARY QUEUE command.
QUEUE Statement Usage

Efficient use of the Queue Area

To make most efficient use of the queue area, specify a retention period that is as short as possible.

Upper Limits in Test and Production Environments

The UPPER LIMIT specification is useful in a test environment to prevent a looping program from filling the queue area. The recommended value for this parameter in a production environment is 0.

Example QUEUE Statement

Adding a Queue with the Default Version Number

The following statement adds queue ABCQ, which is version 1 by default:

```
ADD QUEUE ABCQ
    INVOKES TASK QTSK1.
```

Adding a Queue with a Specified Version Number

The following statement adds queue ABCQ version 2:

```
ADD QUEUE ABCQ VERSION IS 2
    DISABLED
    INVOKES TASK QTSK2
    THRESHOLD IS 5
    UPPER LIMIT IS 2000.
```

Modifying a Queue

The following statement modifies queue ABCQ version 2 by changing the upper limit value to 1,000:

```
MODIFY QUEUE ABCQ VERSION 2
    UPPER LIMIT IS 1000.
```

Deleting a Queue

The following statement deletes version 1 of queue ABCQ:

```
DELETE QUEUE ABCQ VERSION 1.
```

RESOURCE TABLE Statement -- Defines a Resource Table

The RESOURCE TABLE statement defines the nodes on which resources in your DC/UCF communications network are located. CA IDMS creates a resource name table from the RESOURCE TABLE statement. The resource name table is used at runtime to identify the location of resources required to satisfy database requests.
If all database requests in your DC/UCF communications network will take place within a single region, then you do not have to explicitly code a RESOURCE TABLE statement. CA IDMS will automatically generate a resource name and node entry in a resource name table for the resources located on the system you are defining.

You must use the RESOURCE TABLE statement to explicitly define resources located on another node within your DC/UCF communications network, if they can be accessed by the DC/UCF system you are defining.

### RESOURCE TABLE Statement Syntax

#### ADD/MODIFY/DELETE RESOURCE TABLE Statement

```
ADD/MODIFY/DELETE RESOURCE TABLE Statement
```

```
ADD/MODIFY/DELETE RESOURCE TABLE

DBName is database-name

DESTination is destination-node

EXclude
```

#### DISPLAY/PUNCH RESOURCE TABLE Statement

```
DISPLAY/PUNCH RESOURCE TABLE Statement
```

```
DISPLAY/PUNCH RESOURCE TABLE

AS COMMENTS
```

### RESOURCE TABLE Statement Parameters

**DBName is *db-name***

Specifies the name of the database or dictionary included in the resource name table. *Db-name* is a one- through eight-character alphanumeric value. The last character of *db-name* can be an asterisk (*) indicating a wildcard character. For more information about the use of the wildcard character, see Usage (see page 249).

- **VIA *nodename***
  
  Identifies the name of the DC/UCF system where the named resource is located or the name of an intermediate node through which the request for data will be routed. *Nodename* is the system name identified on the SYSTEM ID parameter of the SYSTEM statement or overridden by the DCNAME parameter in the SYSIDMS file at startup and must match a nodename defined with the NODE statement.
- **VIA groupname**
  Identifies the name of the DBGROUP to which the request for data will be dynamically routed. *Groupname* is the name of a DBGROUP defined in the database name table of one or more DC/UCF systems any of which are capable of servicing the request for data. Groupname must match a nodename defined by a NODE statement specifying a GROUP parameter.

- **LOCAL**
  Specifies the resource is located on the DC/UCF system you are defining. LOCAL is the default.

**DESTination is destination-node**

Specifies a node name to which requests for data are sent.

Specify a *destination-node* rather than a *database-name* when user applications will specify a nodename on a native DML BIND RUNUNIT statement or a DC batch application BIND TASK statement.

The last character of destination-node can be an asterisk (*) indicating a wildcard character. For more information about the use of the wildcard character, see Usage (see page 249).

*Destination-node* is a unique one- through 8- character alphanumeric value.

**EXclude**

- **DBName database-name**
  Identifies the name of a previously defined database to exclude from the resource name table.

- **DEStination destination-node**
  Identifies the name of a previously defined *destination-node* to exclude from the resource name table.

**RESOURCE TABLE Statement Usage**

**What are Resources**

A resource is either a database or a node (system) from which requests for a database are forwarded. You specify a node name as a DESTINATION when:

- You don’t know the ultimate target node or
- No physical communication link exists except for the DC/UCF system you are defining and the final destination node

**Using Wildcards for Resource Names**

You can use a wildcard character with the *db-name* and *destination-name* variables. For example, you could define:

```
ADD RESOURCE TABLE
DBNAME IS MIS* VIA SYSTEMB4
```
MIS* would be a single entry in the resource name table. At runtime, any request for a database name with MIS, would be matched with MIS* and routed to SYSTEM84.

Additionally, CA IDMS automatically generates the following entries as the last two entries in the resource name table:

```
DBNAME * LOCAL
DESTINATION *
```

This ensures that all database requests are routed to the DC/UCF system you are defining.

**Sequence of Resource Names is Important**

CA IDMS generates entries in the resource name table in the order they are defined on the RESOURCE TABLE statement. This is important because the resource name table is searched from the beginning of the table to the end until a match is found. The first match CA IDMS finds in the resource name table is the node name it will use to locate the requested resource.

You should define specific resource names first and less specific resource names next.

If the resource is not located in the resource name table or the node name does not match the node name specified on the request, an error is returned to the application program.

**Specifying Resources**

The type of resource you specify is either a database or a destination (nodename). In most cases, you will specify a database name.

The only time you need to explicitly define a destination resource is when the target node is not defined using a NODE statement. This can be the case because:

- There is no direct communications path from the system being defined to the target system, thus requiring database requests to be routed through an intermediate node. If this is the case, you must define the target node as a destination resource and specify the intermediate node in its VIA parameter.

- Dynamic routing through DBGROUPs is used to direct database connections to any of several systems capable of processing the request. The default access method is used to access the dynamically selected target node. If this is the case, you must define the members of the DBGROUP as destination resources and specify the name of the DBGROUP in their VIA parameter. If the node names of all DBGROUP members follow a unique naming convention, then wildcarding can be used in the destination nodename to reduce the number of required definitions and to allow new members to be added without necessitating changes to the resource table.

**Modifying the Resource Table**

You can modify and delete individual entries in the resource table without deleting the entire resource table. Use the EXCLUDE clause of the RESOURCE TABLE statement to delete a specific DBNAME or DESTINATION entry from the resource table.
Use the MODIFY RESOURCE TABLE statement to add new entries to an existing resource table. Note that new entries added to the resource table on a MODIFY operation are added to the end of the resource table. To control the sequence of entries in the resource table when adding new entries to an existing resource table, you have to delete and then add the resource table; submitting the entries in the sequence you want them to appear in the resource table.

You can also modify the LOCAL/VIA option to change the location of a database name resource.

**Example RESOURCE TABLE Statement**

**Defining a Resource Table**

```
ADD RESOURCE TABLE
  DBNAME IS SYS104 VIA EDCQAM01
  DESTINATION IS PRODC* VIA CUSTGRP
  DBNAME IS MIS* VIA SYSTEM84.
```

**Modifying a Resource Table to Add a Database Name**

```
MODIFY RESOURCE TABLE
  DBNAME IS SYS103 VIA EDCQA01.
```

**Modifying the Resource Table to Delete an Entry**

```
MODIFY RESOURCE TABLE
  EXCLUDE DESTINATION SYSQA02.
```

**RUNUNITS Statement -- Creates Predefined Run Units**

The RUNUNITS statement is used to predefine:

- System run units for load area processing for an alternate dictionary or catalog
- Run units to perform security checking for SQL objects in an alternate catalog

Run units predefined can be controlled at runtime by means of the DCMT DISPLAY and VARY RUN UNIT LOADER commands. For more information about DCMT commands, see the *CA IDMS System Tasks and Operator Commands* section.

**RUNUNITS Statement Syntax**

```
ADD/MODIFY/DELETE RUNUNITS Statement

►►─┬──────────┬─ RUNunits for ─┬── LOAder ──►
├─ ADD ─────┤            └── SQL LOAder ─┘
├─ MODIFY ─┤            └── SQL SECurity ┘
└─ DELETE ─┘

►─── DICtname is ─ dictionary-name ───►
└─ COUnt is ─ 1 ┘
```
RUNUNITS Statement Parameters

**LOAder**
Allows you to predefine rununits for loading from dictionary load areas.

**SQL LOAder**
Allows you to predefine rununits for loading access modules when using the SQL option.

**SQL SECurity**
Allows you to predefine rununits that will perform security checking against SQL-related objects such as tables, access modules and SQL-defined schemas.

**DICtname is dictionary-name**
Identifies the data dictionary for which the run units are being predefined.

*Dictionary-name* must be the one- through eight-character name of a data dictionary included in a segment of the DMCL defined for the runtime environment.

You must explicitly identify an alternate dictionary. *Dictionary-name* cannot be the system default dictionary. You specify a default dictionary for predefined run units on the RUNUNIT FOR clause of the SYSTEM statement.

**COUnt is run-unit-count**
Specifies the number of run units being predefined.

*Run-unit-count* must be an integer in the range 1 through 999. The default is 1.
**Idle interval is**

Specifies how long the DC/UCF system is to permit a predefined run unit to be inactive before issuing a FINISH command for the run unit. A terminated run unit will be rebound when a program requires access to the load area and no predefined run units are available.

- **idle-interval**
  Specifies the idle interval in minutes.
  
  *Idle-interval* must be an integer in the range 1 through 9,999.

- **OFF**
  Directs the system not to terminate inactive run units based on an idle interval.
  
  OFF is the default when you omit the IDLE INTERVAL parameter from the RUNUNITS statement.

### Example RUNUNITS Statements

#### Predefining Run Units

The following statement creates five predefined run units for load area processing in the TSTDICT dictionary. The run units will be terminated if they remain inactive for more than eight minutes.

```
ADD RUNUNITS FOR LOADER
   DICTNAME IS TSTDICT
   COUNT IS 5
   IDLE INTERVAL IS 8.
```

#### Changing the Idle Interval

The following statement changes the idle interval for the predefined run units for the TSTDICT dictionary:

```
MODIFY RUNUNITS FOR LOADER
   DICTNAME IS TSTDICT
   IDLE INTERVAL IS 5.
```

#### Deleting Predefined Run Units

The following statement deletes the predefined run units for the TSTDICT dictionary:

```
DELETE RUNUNITS FOR LOADER
   DICTNAME IS TSTDICT.
```

For more information about predefined run units, see System Run Units (see page 67).

### SQL CACHE Statement -- Controls SQL Caching

The SQL CACHE statement controls SQL caching in a central version.

#### SQL CACHE Statement Syntax

**ADD/MODIFY/DELETE SQL CACHE Statement**
SQL CACHE Statement Parameters

statement-count

Specifies the maximum number of SQL statements that can be placed in the SQL statement cache. The default value is 100. The maximum theoretical value is 2,147,483,647. The actual maximum depends on available memory.

DEFault caching

- ON - Specifies that caching of dynamic SQL statements is enabled. ON is the default.
- OFF - Specifies that caching of dynamic SQL statements is disabled.

connect-name

Specifies the name of a dictionary or catalog to which a user of the CV can connect. You can specify multiple connect-names to form an exception list to the default caching specification.

SQL CACHE Statement Usage

**Dynamically changing caching attributes:** All of the options that can be specified in an SQL CACHE statement can be changed dynamically by issuing SQL DML statements against CA-supplied tables. For more information, see the CA IDMS SQL Reference section.
Default caching status: If an SQL CACHE statement is not specified for a system, the dynamic SQL caching is disabled at system startup. SQL caching can be dynamically enabled by inserting a row into the SYSCA.SQLCACHEOPT table. For more information, see the CA IDMS SQL Reference section.

Specifying an exception list: You can specify an exception list of connect-names for which caching is enabled or disabled depending on what was implicitly or explicitly specified in the DEFAULT CACHING clause. If default caching is enabled, caching is disabled for any session connected to a dictionary or database whose name appears in the exception list. Conversely, if default caching is disabled, caching is enabled for any such session.

Example SQL CACHE Statement

Enabling an SQL Cache

The following statement enables an SQL cache to hold up to 500 SQL statements. Caching will be active for all connect names except for the SYSTEM dictionary.

```
ADD SQL CACHE
  FOR 500 STATEMENTS
  DEFAULT CACHING IS ON
  EXCEPT CONNECT TO(SYSTEM).
```

Modifying the Connect Exception List

The following statement will set the dictionaries SYSTEM and APPLDICT in the exception list.

```
MOD SQL CACHE
  EXCEPT CONNECT TO(SYSTEM APPLDICT).
```

Removing SQL caching

The following statement removes the SQL cache from the current system definition. Please note that it will still be possible to dynamically add SQL caching through SQL DML statements. Refer to the examples in the appropriate section of the CA IDMS SQL Reference section.

```
DELETE SQL CACHE.
```

STORAGE POOL Statement -- Defines Secondary 24-Bit Storage Pools

The STORAGE POOL statement is used to define secondary 24-bit storage pools and their characteristics, including the storage pool number, size, and type of storage the pool contains. Using the STORAGE POOL statement, you can define up to 127 alternate 24-bit storage pools, each containing up to six types of storage.

You define each storage pool with a separate STORAGE POOL statement, specifying a number that identifies the storage pool. You can specify optional parameters or allow parameter defaults noted in the syntax description.
At runtime, users can display summary statistics for all storage pools in the system with the DCMT DISPLAY ALL STORAGE POOLS statement. Users can display statistics for a single storage pool using the DCMT DISPLAY ACTIVE STORAGE statement.

When using storage protect in a production environment, the storage pools must be defined in such a manner that all forms of user-oriented storage are segregated from the system-oriented storage. In other words, explicitly define both an XA and a non-XA storage pool for user storage types. Storage types: User, User kept, Shared, and Shared kept, can be together, but they must be defined to secondary storage pools and must be isolated from any secondary pools which contain either Database or Terminal type storage. Do not use the ALL keyword in the sub-clause "Contains types (ALL)" and do not permit this sub-clause to default on any storage pool being defined for a storage protected production environment, instead the types must be explicitly stated for each pool being defined.
STORAGE POOL Statement Parameters

STOrage POOl storage-pool-number

Specifies a number that uniquely identifies a 24-bit storage pool.

Storage-pool-number must be an integer in the range 1 through 127.

- **CONTains types**
  Specifies the types of storage the named storage pool can accommodate. Storage types must be enclosed by parentheses. If you specify more than one storage type, each type must be separated from the next by a blank or a comma.

  - **DATabase**
    Specifies the storage pool can contain storage for DMCL buffers. The DMCL buffers hold copies of database pages.

  - **SHAred**
    Specifies the storage pool can contain storage that is designated shared.

  - **SHARED-Kept**
    Specifies the storage pool can contain storage that is designated shared kept.

  - **TERminal**
    Specifies the storage pool can contain storage for terminal buffers.

  - **USER**
    Specifies the storage pool can contain storage that is designated user.

  - **USER-Kept**
    Specifies the storage pool can contain storage that is designated user kept.

  - **ALL**
    Specifies the storage pool can contain all of the storage types listed above. ALL is the default when you omit the CONTAINS TYPES parameter from the STORAGE POOL statement.

- **CUShion is storage-cushion-size**
  Specifies the amount of free storage, in 1K bytes, to be reserved in the specified storage pool to help prevent exhausting space in the storage pool. Storage-cushion-size must be an integer in the range 1 through 16,383. The default is 1.

  **Note:** Users can override this parameter at runtime with the DCMT VARY STORAGE POOL CUSHION command.
- **PGFix**  
  For virtual storage systems only, specifies that pages in the storage pool are fixed as they are allocated.

- **NOPgfix**  
  For virtual storage systems only, suppresses page fixing.  
  NOPGFIX is the default when you specify neither PGFIX nor NOPGFIX in the STORAGE POOL statement.

- **RELocatable threshold is**  
  Specifies the point at which the DC/UCF system is to write relocatable storage in the named storage pool to the scratch area (DDLDCSCR) of the data dictionary.

  - **threshold-percentage**  
    Directs the system to write relocatable storage to the scratch area across a pseudo-converse when the amount of used space in the storage pool exceeds the specified threshold.  
    *Threshold-percentage* must be an integer in the range 0 through 100. A value of 0 is synonymous with YES. A value of 100 is synonymous with NO.

  - **YES**  
    Directs the system always to write relocatable storage to the scratch area across a pseudo-converse.  
    YES is the default when you omit the RELOCATABLE THRESHOLD parameter from the STORAGE POOL statement.

  - **NO**  
    Directs the system never to write relocatable storage to the scratch area across a pseudo-converse.

- **SIZE is storage-pool-size**  
  Specifies the amount of storage, in 1K bytes, to be made available for system and program variable storage.  
  *Storage-pool-size* must be an integer in the range 1 through 16,383. The default is 1.

  **Note:** For more information about runtime storage allocation, see the *CA IDMS System Operations section.*

---

**Example STORAGE POOL Statement**

**Adding a Storage Pool with Default Characteristics**

The following statement defines storage pool 3. All STORAGE POOL statement defaults are accepted:

```
ADD STORAGE P00L 3.
```

**Changing the Types of Storage Allowed in a Storage Pool**

The following statement modifies storage pool 3 by changing the storage types:
MODIFY STORAGE POOL 3
   CONTAINS TYPES (TERMINAL DATABASE).

Adding a Storage Pool with Specified Characteristics

The following statement adds storage pool 5:

ADD STORAGE POOL 5
   SIZE IS 250
   CUSHION IS 50
   PGFIX
   CONTAINS TYPES (SHARED SHARED-KEPT).

Changing the Size of a Storage Pool

The following statement modifies storage pool 5 by changing its size to 300:

MODIFY STORAGE POOL 5
   SIZE IS 300.

Deleting a Storage Pool

The following statement deletes storage pool 5:

DELETE STORAGE POOL 5.

TASK Statement

The TASK statement is used to define a task and its characteristics, including the code used to invoke the task, and to associate the task with a DC/UCF system. The task definition names the first program to be invoked by the task. Subsequent programs are invoked according to program instructions (for example, LINK) and are not specified in the task definition. The TASK statement optionally specifies an upper limit for system resources owned by a task.

Users can override certain TASK parameters at runtime by using the DCMT VARY TASK command. The allowable overrides are noted in the syntax rules for the applicable parameters.

TASK Statement Syntax

ADD/MODIFY/DELETE TASK

ADD/MODIFY/DELETE TASK
   TASK task-code
   ADD
   MODIFY
   DELETE

   Version is 1
   version-number

   AREA acquisition THRESHOLD is count
   DEFAULT OFF
   RETRY count
   FOREVER

   ENABLED
   DISABLED
TASK Statement Parameters

**TASK task-code**

Specifies a code that identifies the task and is used at runtime by a terminal operator or program to invoke the task.

*Task-code* must be a one- through eight-character value.

⚠️ **Note:** CA ADS mainline dialogs do not require task codes in order to be invoked.

**Version is version-number**

Differentiates tasks with identical task codes (for example, production and test versions).

*Version-number* must be an integer in the range 1 through 9,999. The default is 1.
Although multiple versions of a task can exist in the data dictionary, only one version of a task is used by a system at runtime. If multiple ADD TASK statements specify the same task code, at runtime the system uses the one that appears first in the system definition.

**AREA acquisition THRESHOLD is**

Controls whether the system will accumulate area locks during ready processing of multiple areas for a single database transaction.

The values you specify on the TASK statement will override those specified on the SYSTEM statement.

- **count**
  Specifies the number of times, during ready processing, the system will wait on an area lock before it starts to accumulate area locks for a transaction. This value only applies if multiple areas are being readied at one time.
  
  *Count* must be an integer in the range 1 through 32,767.
  
  Before the threshold count is reached, the system will free locks on areas previously locked by the database transaction if the transaction must wait to place a lock on another area.
  
  Once the threshold is reached, the system will not release existing area locks while waiting for a new area lock.

  - **DEFAULT**
    Directs the system to use the values specified in the AREA ACQUISITION THRESHOLD of the SYSTEM statement.
    
    DEFAULT is the default.

- **OFF**
  Directs the system not to accumulate area locks until it can acquire all areas needed by a database transaction.

  - **RETry count**
    Defines a limit on the number of times the system will continue trying to gain access to all areas without accumulating area locks. *Count* must be an integer in the range 1 through 32,767.

  - **RETry FOREver**
    FOREver is the default and directs the system to continue to try to gain access to all needed areas until it successfully acquires all areas or until operating system resource and time limits are exceeded. You should specify the default of FOREver unless experience shows that a transaction is not gaining access to areas as needed.

**ENABLED**

Indicates the task is enabled at system startup and can access the system.

ENABLED is the default when you specify neither ENABLED nor DISABLED in the TASK statement.

**DISABLED**

Indicates the task is not enabled at system startup and is prohibited from accessing the system until the task is enabled.

**Note:** Users can override the ENABLED/DISABLED parameter at runtime with the DCMT VARY TASK command.
**EXTERNAL WAIT is**

Specifies the length of time the system waits for an external request for the named task before abnormally terminating the task.

⚠️ **Note:** Users can override the EXTERNAL WAIT parameter at runtime with the DCMT VARY TASK command.

**INActive interval is**

Specifies the amount of time the system permits a DC task or external user session to wait for a resource before abnormally terminating the task.

⚠️ **Note:** Users can override this parameter at runtime with the DCMT VARY TASK STALL command.

**Note:** For more information about the inactive interval, see Abend Detection and Timed Functions (see page 47).

**INPut**

Specifies the terminal input buffer can contain data along with the task code. You should specify INPUT if the program invoked by the task requests a transfer of data from the terminal buffer to the task via a READ TERMINAL, READ LINE TERMINAL, or MAPIN request, as its first I/O operation, or if an external task permits the terminal operator to specify parameters or commands following the task code (for example, DCUF SHOW USERS).

INPUT is the default when you specify neither INPUT nor NOINPUT in the TASK statement.

**NOInput**

Specifies the terminal buffer can contain only the task code. You should specify NOINPUT if the program invoked by the task does not request a transfer of data from the terminal buffer to the task as its first I/O operation or if an external task does not permit the terminal operator to specify parameters or commands following the task code (for example, BYE).

**INTernal**

Specifies the task can be invoked internally only. A task is invoked internally when an executing program specifies the task code in an ATTACH or DC RETURN NEXT TASK CODE request.

**EXTernal**

Specifies the task can be invoked either externally or internally. A task is invoked externally when the user enters a task code in response to the system prompt.
EXTERNAL is the default when you specify neither INTERNAL nor EXTERNAL in the TASK statement.

**INVokes program program-name**

Specifies the name of the initial program to be invoked by the system for the task.

*Program-name* must be the name of a program previously defined in the data dictionary.

The INVOKES PROGRAM parameter is required in ADD TASK statements.

⚠️ **Note:** Users can override this parameter at runtime with the DCMT VARY TASK PROGRAM command.

**LIMit is**

Specifies limits on system resources held by tasks. You can code one LIMIT parameter for each resource type.

⚠️ **Note:** To enforce limits on a task's resources, task statistics must be collected. You use the STATISTICS parameter of the system generation SYSTEM statement to enable the collection of task statistics. For more information about runtime considerations for task resource limits, see the CA IDMS System Operations section. **Note:** Users can override this parameter at runtime with the DCMT VARY TASK command.

**CALL**

Limits the number of system service calls (for example, OBTAIN CALC) issued by an online task. When you specify CALL, *limit* must be an integer in the range 1 through 2,147,483,647.

**DBIo**

Limits the number of database I/O operations (reads and writes) performed by online task. When you specify DBIO, *limit* must be an integer in the range 1 through 2,147,483,647.

**LOCK**

Limits the number of record locks allocated to an online task during the life of the task. When LOCK is specified, *limit-n* is an integer in the range 1 through 2,147,483,647.

**STOrage**

Limits the amount of storage, in 1K bytes, that an online task can hold at one time. When you specify STORAGE, *limit* must be an integer in the range 1 through 16,383.
Directs the system to use the specified limit for the indicated resource.

**SYStem**

Directs the system to use the limit for the resource defined in the SYSTEM statement LIMIT FOR ONLINE TASKS parameter.

SYSTEM is the default when you omit the LIMIT parameter from the TASK statement.

**OFF**

Directs the system not to limit the task’s use of the named resource.

**LOCation is**

Specifies the memory location from which the DC/UCF system loads programs and allocates storage for the named task.

⚠️ **Note:** Users can override this parameter at runtime with the DCMT VARY TASK LOCATION command.

**MAP**

Specifies the task performs a mapout automatically upon initiation. The DC/UCF user program will not perform this I/O operation.

**NOMap**

Specifies the task will not perform a mapout automatically upon initiation. If you specify NOMAP, the task is not dedicated to a mapout operation and the task’s initial program, as specified in the INVOKES PROGRAM parameter, does not have to be a map.

NOMAP is the default when you specify neither MAP nor NOMAP in the TASK statement.

**MAXimum CONcurrent threads is**

Limits the number of threads that can be active concurrently for the task. When the limit is exceeded, a message is routed to the terminal.

⚠️ **Note:** Users can override this parameter at runtime with the DCMT VARY TASK MAXIMUM CONCURRENT command.

**ON COMmit**
Specifies options that control commit behavior. These options apply only to commit operations in which the database session remains active.

- **SYSTEM**
  Specifies that the commit behavior for the task should default to that specified for the system.

- **WRite COMT**
  Specifies that a COMT journal record should be written.

- **WRite ENDJ**
  Specifies that an ENDJ journal record should be written.

**NEW ID**
Specifies that a new local transaction ID should be assigned to the next transaction associated with the database session.

**RETain ID**
Specifies that the existing local transaction ID should be assigned to the next transaction associated with the database session.

ON ROLlback continue

Specifies options that control rollback behavior. These options apply only to rollback operations in which the database session remains active.

- **SYSTEM**
  Specifies that rollback behavior for the task should default to that specified for the system.

- **RETain ID**
  Specifies that following a rollback, the current local transaction ID should be assigned to the next transaction associated with the database session.

- **NEW ID**
  Specifies that following a rollback, a new local transaction ID should be assigned to the next transaction associated with the database session.

**PRINt key is**

Overrides the system-wide print-screen key assignment (specified by the PRINT KEY parameter of the system generation SYSTEM statement) for the named task.

- **PA\(n\)**
  Identifies a program attention key. \(N\) must be the integer 1, 2, or 3.

- **PF\(n\)**
  Identifies a program function key. \(N\) must be an integer in the range 1 through 24.

**Note:** The transfer control facility (TCF) uses PF3 and PF9 for the TERMINATE and SUSPEND functions. For tasks that execute under TCF, do not specify PF3 or PF9 as the print-screen key.
• **SYSTEM (default)**
  Indicates the print-key assignment specified in the PRINT KEY parameter for the current SYSTEM statement is used.
  SYSTEM is the default when you omit the PRINT KEY parameter from the TASK statement.

• **OFF**
  Disables the print-screen facility for the task.

**Note:** For more information about assigning a print-screen key, see the description of the SYSTEM statement PRINT KEY parameter in SYSTEM Statement (see page 131).

**PRiority is task-priority**

Specifies the dispatching priority for the online task.

*Task-priority* must be an integer in the range 0 (lowest priority) through 240 (highest priority). The default is 100.

For more information about the dispatching priority of certain tasks, see Specifying a TASK PRIORITY

**Note:** Users can override this parameter at runtime with the DCMT VARY TASK PRIORITY command. **Note:** The PRIORITY clause has no impact on tasks initiated as a result of an ATTACH statement in a COBOL or PL/I program or a #ATTACH macro in an Assembler program.

**PROduct code is product-code**

Identifies a product and related task codes by a generic name.

*Product-code* must be a one- through eight-character generic name of the product associated with the task.

When using this parameter in conjunction with TCF, *product-code* must be one of the following values: IDD, OLM, SCHEMA, SSC, SYSGEN, OLQ, ADSA, ADS2, or ADSC.

**PROtocol is**

Specifies the response protocol to be used by the task when communicating with terminals associated with a VTAMLIN type line.

• **DEFresp**
  Specifies the task is to use definite response protocol. The terminal from which the task is executed must respond with a completion status when a terminal write is requested from the DC/UCF system.
  DEFRESP is the default when you omit the PROTOCOL parameter from the TASK statement.
Printers run with definite response protocol regardless of the protocol specification for the task. **Note:** If you specify DEFRESP the VTAM line definition statement, then you must specify EXPRESP on the task definition statement for each individual task.

- **EXPresp**
  Specifies the task is to use exception response protocol. Exception response protocol reduces transmission time because terminals are not required to respond with a completion status when a terminal write is requested from the DC/UCF system.

  **Note:** If you specify EXPRESP on the VTAM line definition statement, then all tasks will run using the exception response protocol, and the specification you make here has no effect.

- **QUIesce wait is**
  Specifies if the system permits the task to wait for a quiesce operation to terminate; and if waiting is permitted the amount of time the task waits before it is abnormally terminated.

  **Note:** Users can override the QUIESCE WAIT parameter at runtime with the DCMT VARY TASK command.

- **RESOURCE timeout**
  Defines the task's resource timeout interval and resource timeout program used for pseudo-conversational transactions.

  **Note:** Users can override this parameter at runtime with the DCMT VARY TASK command.

- **SAVe**
  Indicates that current terminal output associated with the task when the terminal on which the task is executing receives a write-direct-to-terminal data stream.

  **Note:** With line-mode I/O, terminal output is always saved unless a program specifies the NOBACKPAGE parameter.

- **NOSave**
  Indicates that the system will not save the contents of the terminal screen before writing a write-direct-to-terminal data stream.
NOSAVE is the default when you specify neither SAVE nor NOSAVE in the TASK statement.

**Note:** Users can override the SAVE/NOSAVE specification at runtime with the DCMT VARY TASK SAVE/NOSAVE command.

TCF task is `tcf-task-code`

Specifies the named task is eligible to run under the transfer control facility (TCF).

`tcf-task-code` must be a previously defined task code that invokes the TCF control program (RHDCUMBR).

- **Version is version-number**
  Differentiates identical TCF task codes (for example, production and test versions).
  Version-number must be an integer in the range 1 through 9,999. The default is 1.

TRAnsaction SHAring is

Specifies options that control transaction sharing.

- **OFF**
  Specifies that transaction sharing is initially disabled when a task of this type is initiated.

- **ON**
  Specifies that transaction sharing is initially enabled when a task of this type is initiated.

- **SYStem**
  Specifies that the initial transaction sharing setting for a task is the value specified in the TRANSACTION SHARING clause of the SYSTEM statement. SYSTEM is the default if you omit the TRANSACTION SHARING parameter from the TASK statement.

**Note:** For more information about transaction sharing, see the *CA IDMS Database Administration section*.

**TASK Statement Usage**

**Effect on DC RETURN NEXT TASK CODE**

If a DC RETURN NEXT TASK CODE request is issued and the next task code is assigned the INPUT attribute, the task does not execute until the terminal operator enters data. If a DC RETURN NEXT TASK CODE request is issued and the next task code is assigned the NOINPUT attribute, the task executes immediately.

**Associating a Task with a PF Key**
You can associate a task with a program function (PF) key by specifying task-code in the form PFn. N must be an integer in the range 1 through 24. At runtime, to invoke a task associated with a PF key, the terminal operator presses the appropriate key.

**Location Mode under XA Systems**

If the DC/UCF system is running under an operating system supporting the XA feature, tasks should be defined with LOCATION IS BELOW unless all programs invoked under the task have been tested to ensure that they can execute in 31-bit addressing mode.

**Specifying MAP**

Typically, you specify MAP for the first task to be executed in a pseudo-conversational transaction. In this case, the task’s initial program, as defined in the INVOKES PROGRAM parameter (see above), must be a map, and the first field of the map must specify the next task code to be initiated. The field can be a variable data field into which the terminal operator will enter the appropriate task code, or it can be a literal field that contains the task code. If a literal field is used, it must be defined with a MODIFY DATA TAG specification of YES and, typically, with the darkened and protected attributes.

**Specifying a TASK PRIORITY**

For the following task types, the task-priority specified on the PRIORITY clause is the dispatching priority for the task at runtime:

- A task initiated as a result of exceeding a queue threshold
- A startup or shutdown autotask
- A task initiated as a result of a SET TIMER START TASK statement in a COBOL or PL/I program or a #SETTIME macro in an Assembler program, if no priority is specified in the SET TIMER statement or the #SETTIME macro.

For tasks associated with a logical terminal, the specified priority is added to the values specified in the PRIORITY parameters in the LTERM statement to determine the priority assigned to the task at runtime. If the sum of the above values is greater than 240, a dispatching priority of 240 is assigned. A task that is initiated in one of the following ways is associated with a logical terminal:

- The task code is entered by the terminal operator in response to the "Enter Next Task Code" prompt
- The task is initiated as a result of a previous task specifying a next task code parameter on a DC RETURN statement or #RETURN macro
- The task is an LTERM autotask

**Use of Product Codes**

When a task code is different from the product name (for example, TASK IDDTCF INVOKES PROGRAM IDMSDDDC, where IDDTCF differs from IDD, the installation default), the PRODUCT CODE parameter relates the task code to the product. The system adds an entry to the task definition element (TDE), listing the task code, referenced by its product code.
The transfer control facility uses the table to switch to a different session. At runtime, the system first scans the list of task codes, then scans the product codes to identify and transfer control to the product requested by the terminal operator.

**Specifying commit and rollback options**

You can specify options that control the following commit and rollback behavior:

- The type of journal record written on a commit
- Whether a new local transaction ID is assigned on a rollback continue or commit.

You can control whether a COMT or ENDJ journal record is written on a commit operation in which the database session remains active. Writing an ENDJ can reduce recovery time because less data has to be examined to locate the start of a recovery unit. This benefit applies to online recovery, warmstart, and ROLLBACK and ROLLFORWARD recovery operations. ENDJ is most beneficial in cases where long-running transactions perform relatively infrequent updates between commit operations. In cases where update transactions are committed frequently, writing ENDJ journal records might negatively impact the amount of information journaled because another BEGN journal record has to be written before the first update following a commit operation.

**Note:** ENDJ journal records are always written when system rununits are committed, regardless of the ON COMMIT option specified.

You can control whether a new local transaction ID is assigned following a commit or rollback operation in which the database session remains active. Assigning a new transaction ID reduces the chance of duplicate IDs should this value wrap within a single cycle of a central version. It also has the effect of recording journal statistics separately by commit recovery unit rather than across all recovery units within a database transaction. You can assign a new ID on a commit operation only if you also specify that an ENDJ checkpoint record be written.

**Note:** A new transaction ID is always assigned when system run units are committed or rolled out.

**Example TASK Statements**

**Adding a TCF Task**

The following statement directs the system generation compiler to add task IDDTCF, to run under TCF, to the data dictionary:

```
ADD TASK IDDTCF
EXTERNAL
INVOKES PROGRAM IDMSDDDC
PRIORITY IS 65
INPUT
```
Adding a User Task

The following statement directs the system generation compiler to add task ABC to the data dictionary:

```
ADD TASK ABC
EXTERNAL
INVOKES PROGRAM ABCPROG
PRIORITY IS 60
INPUT
LOCATION IS ANY
RESOURCE TIMEOUT
INTERVAL IS SYSTEM
PROGRAM IS SYSTEM
SAVE
```

Modifying a Task

The following statement modifies task ABC by changing its priority class to 32:

```
MODIFY TASK ABC
PRIORITY IS 32.
```

Deleting a Task

The following statement deletes task ABC:

```
DELETE TASK ABC.
```

TCP/IP Statement -- Defines TCP/IP Runtime Environment

The TCP/IP statement is used to define the TCP/IP runtime environment of a DC/UCF system.

TCP/IP Statement Syntax

```
ADD/MODIFY/DELETE TCP/IP Statement
```

```
ADD Modify Delete TCP/ip
```

```
DEFault STATus is ON OFF
```

```
DEFault TCP_NODelay option is ON OFF
```

```
DEFault STACK is DEFault
```

TCP/IP Statement Parameters

TCP/ip and TCPip

These are equivalent keywords that can be used the same way everywhere. They can be specified in SYSGEN definitions or DCMT commands.

DEFAULT STATUS is

Specifies the status of TCP/IP support in CA IDMS. OFF disables TCP/IP support for CA IDMS at startup. ON enables support. The default is ON if an ADD TCP/IP statement is coded. If the TCP/IP entity is not defined to the system, the default is OFF.

 mereka: Note:

DEFAULT TCP_NODelay option is
Specifies the default value for the NO_DELAY SOCKET option. Unless overridden for a specific socket connection, this is the value that will be used for all communication. The default value is OFF, meaning that a delay may be experienced between consecutive sends in order to optimize overall data movement.

**Note:** For more information about the TCP_NODELAY socket option, see the GETSOCKOPT and SETSOCKOPT SOCKET functions in the *CA IDMS Callable Services section.*

**DEFault STACk is**

Specifies the default stack to be used by the DC/UCF system.

- **stack-name**
  Identifies the name of the stack to be used as a default.

- **DEFault**
  Specifies that the CA IDMS assigned default stack is to be used. DEFAULT is the default. The default stack for a DC/UCF system varies by operating system.

  - On z/OS, the operating system assigns a specific stack as the default stack. Unless that stack is explicitly excluded from use by SYSGEN or SYSIDMS parameters, CA IDMS uses the operating-system assigned default stack. If the default stack has been excluded, CA IDMS chooses the first active stack from the list of stacks as the default.

  - On z/VM, the default stack is the first stack in the list of stacks.

**INClude/EXClude STack stack-spec**

Controls or limits the stacks that can be used by the socket applications running in the CA IDMS system. This option is useful only in a multiple stack environment where multiple TCP/IP stacks run concurrently; it is ignored on systems where only one TCP/IP stack is active. It is used differently depending on the operating system:

- On z/OS, the INCLUDE or EXCLUDE list is used to customize the default stacks list returned by the operating system when CINET is active. Both lists are mutually exclusive. If no INCLUDE or EXCLUDE list is specified, all stacks in the list returned by the operating system are included. If specified, the resulting list of stacks depends on the type of list being defined:

  - INCLUDE List -- This list is built by excluding all the stack names that are not present in the SYSGEN INCLUDE list.

  - EXCLUDE List -- This list is built by excluding all the stack names that are present in the SYSGEN EXCLUDE list.
Wildcards can be used as special names for \texttt{stack-spec} to define groups of stack names starting with the same pattern. When wildcards are used in the INCLUDE or EXCLUDE list, the EXCEPT list can be used to refine the set of included or excluded stacks by excluding specific stacks. Wildcards can also be used for \texttt{stack-spec} in the EXCEPT list if they represent a sub-group of names from a larger group declared in the INCLUDE or EXCLUDE list. See examples at the end of this section.

- On z/VM, only an INCLUDE list is available. Use the INCLUDE list to define the full list of stacks to use in the CA IDMS system. Wildcards are not accepted.
  This list can be used to replace the r16 definitions using the SYSTCPD file and the existing SYSIDMS parameters; these definitions are ignored when the stacks are defined through SYSGEN.

An empty list can be specified for the INCLUDE, EXCLUDE or EXCEPT list in order to remove all entries from the corresponding list. Duplicate names are ignored when specified within the same list of stacks.

\textbf{MAXimum NUMber of SOckets is} \texttt{max-socket}

Specifies the maximum number of sockets that can be created globally in the DC/UCF system. \texttt{max-socket} is a positive number between 1 and 65535. If DEFAULT is specified, a default value is assigned at startup. This default value depends on the operating system: 65535 on z/OS, 8000 on z/VSE, and 512 on z/VM.

The maximum number of sockets that can be created in one address space can also be limited by the operating system, for example, through USS definitions under z/OS.

\textbf{MAXimum NUMber of SOckets per TASk is} \texttt{max-socket-per-task}

Specifies the maximum number of sockets that can be created by a single task in the DC/UCF system. The maximum value and the default value for this parameter are both equal to the value assigned at runtime to \texttt{max-socket}. If the \texttt{max-socket-per-task} value is greater than \texttt{max-socket}, it is truncated.

\textbf{PLUgin MODule is} \texttt{plugin module}

Specifies the name of the plug-in module that implements support for specific TCP/IP stack implementations. The only plug-in module name that is accepted is RHDCD1IP; this is also the default value.

\textbf{Note:} In prior releases, the name of the plug-in module was specified on the MODULE is plug-in clause of the SOCKET LINE SYSGEN statement. While this clause is still supported for upward compatibility, it is no longer required and the name of the plug-in module should now be specified using the PLUGIN MODULE clause of the new system generation TCP/IP statement.

\textbf{SERvices FIle is}

Defines the file to be used for translating service names to port numbers and vice versa.
file-name
Specifies the ddname (z/OS and z/VM) or the file name (z/VSE) of the services file. If the data set or file corresponding to file-name cannot be found at runtime (DD card not specified in the startup JCL or data set not cataloged), an error message is written to the log file. Subsequent calls to the GETSERVBYNAME or GETSERVBYPORT socket function returns a specific ERRNO code.

OFF
Indicates that no services file is available and port number/service name resolution is not supported. OFF is the default.

CASE
Indicates whether the service name specified on input to a GETSERVBYNAME socket function is case-sensitive or case-insensitive. The default value is case-sensitive.

Note: For more information about the Services File and the Services Resolver, see the CA IDMS System Operations section.

Example TCP/IP Statement

Including or Excluding TCP/IP Stacks

This example illustrates a list of INCLUDE and EXCLUDE TCP/IP stack definitions and the TCP/IP stacks generated from them.

Assume the special system call on z/OS returns the following list of TCP/IP stacks as defined to CINET:
TCPSY100 - TCPSY110 - TCPSY200 - RUNTCP10 - RUNTCP11 - TESTTCP

The following SYSGEN definitions illustrate how to specify the TCP/IP stacks to include or exclude in the CA IDMS system:

- This statement

  MOD TCPIP
  INCLUDE STACK (TCPSY*,RUNTCP*) EXCEPT (TCPSY2*,RUNTCP11).

  produces the following list of stacks:
  TCPSY100 - TCPSY110 - RUNTCP10

- This statement

  MOD TCPIP
  INCLUDE STACK (*) EXCEPT (TCPSY*,TESTTCP).

  produces the following list of stacks:
  RUNTCP10 - RUNTCP11

- This statement
MOD TCPIP
   EXCLUDE STACK (TCP*) EXCEPT (TCPSY200).

produces the following list of stacks:
   TCPSY200 - RUNTCP10 - RUNTCP11 - TESTTCP

**XA STORAGE POOL Statement -- Defines the 31-Bit Storage Pools**

The XA STORAGE POOL statement, for operating systems supporting the XA feature, is used to define 31-bit storage pools and their characteristics, including the storage pool number, size, and types of storage the pool contains.

The XA STORAGE POOL statement is used to define 31-bit storage pools for user applications. To define the 31-bit storage pool (storage pool number 255) for the processing of system modules, use the XA STORAGE POOL parameter of the system generation SYSTEM statement.

You can define up to 127 31-bit storage pools, each of which can contain up to four types of storage.

You define each XA storage pool with a separate XA STORAGE POOL statement, specifying a number that identifies the storage pool. You can specify optional parameters or allow parameter defaults, noted in the syntax description.

Users can display summary statistics for all storage pools in the system with the DCMT DISPLAY ALL STORAGE POOLS statement. Users can display statistics for a single XA storage pool using the DCMT DISPLAY ACTIVE STORAGE statement.

**XA STORAGE POOL Statement Syntax**

ADD/MODIFY/DELETE XA STORAGE POOL Statement

```
ADD/MODIFY/DELETE XA STORAGE POOL storage-pool-number
   CONTAINS types ( DATABASE, SHARED, SHARED-Kept, TERMINAL, USER, USER-Kept, ALL )
   CUSHion is 1 storage-cushion-size
   PGFix
   NOpgfix
   RELocatable threshold is YES threshold-percentage
   NO
```

```
**XA STORAGE POOL Statement Parameters**

**XA STOrage POOl storage-pool-number**

Specifies a number that uniquely identifies a 31-bit storage pool.

*Storage-pool-number* must be an integer in the range 128 through 254.

⚠️ **Note:** For more information about storage pools, see *Storage Pools (see page 70).*

**CONtains types**

Specifies the types of storage the named storage pool can accommodate.

Storage types must be enclosed by parentheses. If you specify more than one storage type, each type must be separated from the next by a blank or a comma.

- **DATabase**
  Specifies the storage pool can contain storage for DMCL buffers. The DMCL buffers hold copies of database pages.

- **SHAred**
  Specifies the storage pool can contain storage that is designated shared.

- **SHARED-Kept**
  Specifies the storage pool can contain storage that is designated shared kept.

- **TERminal**
  Specifies the storage pool can contain storage for terminal buffers.
- USER
  Specifies the storage pool can contain storage that is designated user.

- USER-Kept
  Specifies the storage pool can contain storage that is designated user kept.

- ALL
  Specifies the storage pool can contain all of the storage types listed above.
  ALL is the default when you omit the CONTAINS TYPES parameter from the XA STORAGE POOL statement.

CUSHion is storage-cushion-size

Specifies amount of free storage, in 1K bytes, to be reserved in the specified storage pool to help prevent exhausting space in the storage pool.

Storage-cushion-size must be an integer in the range 1 through 16,383. The default is 1.

⚠️ Note: Users can override this parameter at runtime with the DCMT VARY STORAGE POOL CUSHION command.

PGFix

For virtual storage systems only, specifies pages are fixed in the storage pool as they are allocated.

NOPgfix

For virtual storage systems only, suppresses page fixing.

NOPGFIX is the default when you specify neither PGFIX nor NOPGFIX in the XA STORAGE POOL statement.

RELocatable threshold is

Specifies the point at which the system is to write relocatable storage in the named storage pool to the scratch area (DDLDSCR) of the data dictionary.

- threshold-percentange
  Directs the system to write relocatable storage to the scratch area across a pseudo-converse when the amount of used space in the storage pool exceeds the specified threshold.
  Threshold-percentange must be an integer in the range 0 through 100. A value of 0 is synonymous with YES. A value of 100 is synonymous with NO.

- YES
  Directs the system always to write relocatable storage to the scratch area across a pseudo-converse.
  YES is the default when you omit the RELOCATABLE THRESHOLD parameter from the XA STORAGE POOL statement.
NO
Directs the system never to write relocatable storage to the scratch area across a pseudo-converse.

Note: For more information about relocatable storage, see Storage Pools (see page 70).

SIZE is storage-pool-size
Specifies the amount of storage, in 1K bytes, to be made available for system and program variable storage.

Storage-pool-size must be an integer in the range 1 through 2,097,151. The default is 1.

Note: For more information about runtime storage allocation, see the CA IDMS System Operations section.

Example XA STORAGE POOL Statement

Adding a 31-bit Storage Pool
The following statement adds XA storage pool 200:

ADD XA STORAGE POOL 200
SIZE IS 400
CUSHION IS 75
PGFIX
CONTAINS TYPES (USER USER-KEPT).

Modifying a 31-bit Storage Pool
The following statement modifies XA storage pool 200 by suppressing page fixing:

MODIFY XA STORAGE POOL 200
NOPGFIX.

Deleting a 31-bit Storage Pool
The following statement deletes XA storage pool 200:

DELETE XA STORAGE POOL 200.

Teleprocessing Network Statements

Three Types of Teleprocessing Network Statements
You use three system generation statements to define a teleprocessing network:
- **LINE** defines a group of hardware devices (physical terminals) that use the same access method.
- **PTERM** associates a physical terminal with a line and defines device characteristics. A physical terminal is any teleprocessing device or group of devices including printers, teletypes, and CRT terminals. A physical terminal can be associated with only one line.
- **LTERM** associates a logical terminal with a physical terminal and defines the physical terminal as an interactive terminal, a printer, or a batch device.

**Associating Logical and Physical Terminals and Lines**

Logical terminal/physical terminal associations and physical terminal/line associations are established during system generation in one of two ways:

- **Explicitly** by naming the associated physical terminal in the LTERM statement and by naming the line associated with the physical terminal in the PTERM statement
- **Implicitly** by establishing currency with a LINE statement and coding PTERM statements for each physical terminal in the line, and by establishing currency with a PTERM statement and coding an LTERM statement for the associated logical terminal

This section presents:

- Syntax and syntax rules for the LINE, PTERM, and LTERM statement parameters that apply to all device types. ADD/MODIFY/DELETE and DISPLAY/PUNCH syntax for each entity type are presented together. Syntax rules follow the presentation of both types of syntax. Rules presented in System Generation Compiler are not repeated.
- Device-specific LINE and PTERM statement parameters.
- A sample teleprocessing network definition.

**LINE Statement**

The LINE statement represents a method of communication and defines line-dependent characteristics to the data dictionary. The ADD, MODIFY, and DISPLAY LINE statements are also used to establish line currency.

You code one LINE statement for each line to be associated with the DC/UCF system. All system definitions should include a LINE statement for the operator’s console.

**LINE Statement Syntax**

```
LINE  line-id
  ADD
  MODIFY
  DELETE
  Version is 1
  COMPact
```
LINE Statement Parameters

**LINE line-id**

Specifies the line identifier.

*Line-id* must be a one- through eight-character value.

- **Version is version-number**
  Qualifies the line definition with a version number.
  *Version-n* must be an integer in the range 1 through 9999. The default is 1.
  Although multiple versions of a line can exist in the data dictionary, only one version can be used by a system at runtime. If multiple ADD LINE statements specify the same line identifier, the system uses the version that appears first in the system definition.

- **COMPact**
  Indicates that redundant data in data streams transmitted to 3270-type terminals is condensed.

- **NOCompact**
  Indicates that redundant data in data streams transmitted to 3270-type terminals is not condensed.
  NOCOMPACT is the default when you specify neither COMPACT nor NOCOMPACT in the LINE statement.
- **ENABLEd**
  Indicates the line is enabled at system startup. 
  ENABLED is the default when you specify neither ENABLED nor DISABLED in the LINE statement.

- **DISABLEd**
  Indicates the line is not enabled at system startup.

**Note:** Users can override the ENABLED/DISABLED specification at runtime with the DCMT VARY LINE command.

- **PROTOCOL**
  For the line type VTAMLIN only, specifies the response protocol to be assigned to the line.

  - **DEFresp**
    Specifies definite response protocol is in effect for the line. Terminals associated with the line must respond with a completion status when a terminal write is requested from the DC/UCF system.
    DEFRESP is the default when you omit the PROTOCOL parameter from the LINE statement. Printers associated with the line run with definite response protocol regardless of the PROTOCOL parameter specification.

  - **EXPresp**
    Specifies that exception response protocol is in effect for the line. All tasks performed on the line use exception response protocol.
    Exception response protocol reduces transmission time because terminals are not required to respond with a completion status when a terminal write is requested from the DC/UCF system.

- **TYPE is line-type**
  Specifies the line type. Valid values for line-type are:

<table>
<thead>
<tr>
<th>Line type</th>
<th>Devices supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASYNC</td>
<td>Asynchronous start/stop devices</td>
</tr>
<tr>
<td>BSC2</td>
<td>Binary synchronous switched or nonswitched point-to-point devices</td>
</tr>
<tr>
<td>BSC3</td>
<td>Binary synchronous leased multipoint devices</td>
</tr>
<tr>
<td>CCI</td>
<td>Common Communications Interface (CCI) line driver</td>
</tr>
<tr>
<td>CONSOLE</td>
<td>Operator's console</td>
</tr>
<tr>
<td>DDS</td>
<td>DDS line driver</td>
</tr>
<tr>
<td>INOUTL</td>
<td>SYSIN/SYSOUT devices</td>
</tr>
<tr>
<td>LAPPCE</td>
<td>Emulated APPC support</td>
</tr>
<tr>
<td>MU</td>
<td></td>
</tr>
<tr>
<td>L3270B</td>
<td>Local 3270 devices using BTAM</td>
</tr>
<tr>
<td>L3280B</td>
<td>Local 3280 devices using BTAM</td>
</tr>
</tbody>
</table>
**Line type Devices supported**

<table>
<thead>
<tr>
<th>Line type</th>
<th>Devices supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCKET</td>
<td>Support Generic Listening and DDS using TCP/IP</td>
</tr>
<tr>
<td>SYSOUTL</td>
<td>SYSOUT devices using QSAM (z/OS systems only)</td>
</tr>
<tr>
<td></td>
<td>Printers that will receive spooled output (z/VM systems only)</td>
</tr>
<tr>
<td></td>
<td>When line type SYSOUTL is specified, all LTERMs defined for the line must be explicitly set to PRINTER.</td>
</tr>
<tr>
<td>S3270Q</td>
<td>Simulated 3270 devices using QSAM</td>
</tr>
<tr>
<td>TCAMLIN</td>
<td>3270 or 3280 devices using TCAM</td>
</tr>
<tr>
<td>UCFLINE</td>
<td>Universal Communications Facility (UCF) line driver</td>
</tr>
<tr>
<td>VTAMLIN</td>
<td>Asynchronous VTAM devices</td>
</tr>
<tr>
<td>VTAMLU</td>
<td>SNA/VTAM logical units</td>
</tr>
</tbody>
</table>

The TYPE parameter is required to create an executable system.

- **additional parameters applicable to line-type**
  - Specify optional characteristics that complete the line definition.
  - Depending on the value specified for line-type, one or more additional parameters may be required.
  - Syntax for the additional parameters applicable to each line type is presented under the heading for the line type later in this section.

### Example LINE Statement

Examples of the LINE statement for various line types can be found under Device Definitions (see page 297).

### PTERM Statement

The PTERM statement is used to define a physical terminal to the data dictionary and to associate the physical terminal with a line. The ADD, MODIFY, and DISPLAY PTERM statements are also used to establish physical terminal currency.

One PTERM statement is coded for each physical terminal to be associated with a line. Multiple physical terminals can be associated with a line; however, only one physical terminal can be associated with a line defined as an operator's console.

A single PTERM statement can be used to define multiple physical and logical terminals with identical characteristics by using the REPEAT COUNT clause. This eliminates the need for using individual LTERM and PTERM statements for each terminal.
PTERM Statement Syntax

ADD/MODIFY/DELETE PTERM Statement

- **ADD**
- **MODify**
- **DELETE**

- Version is
- **Enabled**
- **Disabled**

- in LINE

- maximum ERRors is
- **PRIInter BUffer size is**

- **PRIInter CLAss is**

- REAdbuffer

- REPeat COUnt is

- SCReen type is

- TYPE is

- additional parameters applicable to physical-terminal-device-type

Cloning LTERM and PTERM Definitions

- **ADD**
- **MODify**
- **DELETE**

DISPLAY/PUNCH PTERM Statement

- Display

- PUNch

- PTERM

- **ALL PTERms**

- WITH

- ALSO WITH

- WITHOUT

- DETails

- HIStory

- ALL

- NONE

- AS

- COMments

- SYNTAX
PTERM Statement Parameters

PTerm physical-terminal-id

Specifies the physical terminal identifier. Physical-terminal-id must be a one- through eight-character value.

Version is version-number

Qualifies physical terminal definition with a version number.

Version-number must be an integer in the range 1 through 9999. The default is 1.

Although multiple versions of a physical terminal can exist in the data dictionary, a system can use only one version of the physical terminal at runtime. If multiple ADD PTERM statements specify the same physical terminal identifier, the system uses the version that appears first in the system definition.

ENAbled

Indicates the physical terminal is enabled at system startup.

ENABLED is the default when you specify neither ENABLED nor DISABLED in the PTERM statement.

DISabled

Indicates the physical terminal is not enabled at system startup.

⚠️ Note: See the CA IDMS System Tasks and Operator Commands section for PTERM runtime override options.

in LiNe line-id

Specifies the line with which the physical terminal is associated.

Line-id must be the identifier of a line previously defined in the data dictionary with a LINE statement.

- Version version-number
  
  Qualifies the line identifier with a version number.
  
  Version-number must be an integer in the range 1 through 9999. The default is 1.
  
  This parameter can be omitted if the physical terminal is to be associated with the current line.

maximum ERRors is line-error-count

Specifies the maximum number of retries (after a terminal I/O error) the system is to perform before disabling the physical terminal.
Line-error-count must be an integer in the range 0 through 255. The default is 3.

**PRInter BUffer size is printer-buffer-size**

Defines the size, in bytes, of the printer's physical buffer. The buffer size determines the largest data stream that can be written to the printer at one time.

Printer-buffer-size must be an integer in the range 0 through 32,767. The default is 0. If 0 is specified, the largest data stream that can be written to the printer is 480 bytes.

⚠️ **Note:** If the value specified by this parameter exceeds the actual size of the printer buffer, data may be lost. The user is not notified when data is lost.

**PRInter**

Establishes the default class or destination for WRITE TO PRINTER requests issued from the physical terminal.

- **CLAss is printer-class**
  Establishes the default printer class for WRITE TO PRINTER requests issued from the physical terminal. Printer classes are associated with a printer device through the PRINTER CLASS parameter of the printer's LTERM statement (described under LTERM Statement (see page 290)).
  Printer-class must be an integer in the range 0 through 64. The default is 1. If 0 is specified, print output is not required for the physical terminal. For example, the terminal is a printer.

- **DEStination is**
  Establishes the default printer destination for WRITE TO PRINTER requests issued from the physical terminal.

- **printer-destination-id**
  Specifies the printer destination.
  Printer-destination-id is the name of a printer destination previously defined with a DESTINATION statement.
  You must be sure the destination entity occurrence exists. When a destination is deleted from a system definition, all affected PTERM statements must be updated to ensure the system remains executable.

- **NULl**
  Indicates the physical terminal has no default printer destination.

**READbuffer**

For 3270-type devices only, allows the specified PTERM to execute a READ BUFFER command.

READBUFFER is the default when you specify neither READBUFFER nor NOREADBUFFER in the PTERM statement.

**NOReadbuffer**
For 3270-type devices only, prevents the specified PTERM from executing a READ BUFFER command.

REPeat COUnt is repeat-count

Specifies the number of times the physical and eventually associated logical terminal should be cloned when a central version is started. repeat-count must be an integer in the range 0 through 32767. Repeat-count 0 means no cloning. If a non-zero repeat-count is specified, the physical and logical terminal name should end on a sequence number and the sum of that sequence number and the repeat count should not cause a digit overflow.

SCReen type is device-table-name

Associates a device independence table with a visual-display teletypewriter terminal (glass TTY). The device independence table provides support for mapping operations to and from the TTY.

Device-table-name must be a 3-character alphanumeric name suffix of a device independence table, as specified in the NAME= parameter of the #TTYDIT macro.

Note: Users can override this parameter at runtime with the DCUF SET SCREEN command.

TYPe is physical-terminal-device-type

Specifies the physical terminal device type.

Physical-terminal-device-type must be a valid device type for the line with which the physical terminal is associated.

The TYPE parameter is required to create an executable system.

additional parameters applicable to physical-terminal-device-type

Specify optional characteristics that complete the physical terminal definition.

Syntax for the additional parameters applicable to each physical terminal device type is presented under the heading for the associated line type later in this section.

PTERM Statement Usage

Cloning PTERM/LTERM uses a naming convention: the PTERM/LTERM name ends on a number of digits called the sequence number. This sequence number is incremented for each cloned PTERM and for its associated LTERM if it exists. SYSGEN makes sure that enough digits are available. It is the DBA's responsibility to ensure that a name conflict does not exist. For example, an existing PTERM/LTERM is defined with a name that equals the name of a cloned PTERM/LTERM. If a name conflict is encountered, a warning message is output, but cloning continues.
Note: A single record in the dictionary represents cloned PTERM/LTERMs. Cloning starts after all dictionary PTERM/LTERM records are read and their associated control blocks built. If there is a name conflict, the PTERM/LTERM with a conflicting name is built as defined by the dictionary record and the cloned PTERM/LTERM is discarded.

Example PTERM Statement

ADD PTERM BULKP01
REPEAT COUNT 98
TYPE IS BULK.

ADD LTERM BULKL01
PTERM BULKP01.

This definition results in the creation of 99 PTERM/LTERM pairs:

- BULKP01/BULKL01,
- BULKP02/BULKL02,
- BULKP03/BULKL03
- ... until
- BULKP99/BULKL99

If a PTERM with name BULKP21 is also defined in the dictionary, this occurs:

- The PTERM BULKP21 and its associated LTERM (if any) is built according to the dictionary definition of BULKP21.
- At run time, during startup, warning message DC391009 is output.
- The clone pair BULKP21/BULKL21 is not built, but cloning proceeds with BULKP22.

Examples of the PTERM statement for various device types can be found under Device Definitions (see page 297).

More Information

For more information about the DESTINATION entity type, see DESTINATION Statement (see page 195).

For more information about mapping support for glass TTYs, see the CA IDMS Mapping Facility section.
LTERM Statement

The LTERM statement is used to define a logical terminal and to associate the logical terminal with a physical terminal.

LTERM Statement Syntax

ADD/MODIFY/DELETE LTERM Syntax

```
ADD/MODIFY/DELETE LTERM Syntax

ADD
MODify
DELETE

LTErm logical-terminal-id

Version is 1 version-number

AUTotask is task-code NULL

BATch COMmand initiated TERMINal initiated

PRInter BANner NOBanner

CHECKpoint is page-count SYSTEM OFF

CLASS is ADD DELete

( printer-class )

NOPrinter

ENAbled

DISabled

PRInter CONtrol is SYStem - print-ctrl-options

PRIority is 0 logical-terminal-priority

PTErm is physical-terminal-id Version 1 version-number
```

Expansion of print-ctrl-options

```
SCS FORMfeed is FF-NL FF

ON REPort STArt FORMfeed for NON-native print is INSert SUPpress NO-Insert

FORMfeed for NATive print is INSert SUPpress NO-Insert
```
LTERM Statement Parameters

**LTerm logical-terminal-id**

Specifies the logical terminal identifier.

*Logical-terminal-id* must be a one- through eight-character value.

**Version is version-number**

Qualifies the logical terminal definition with a version number.

*Version-number* must be an integer in the range 1 through 9999. The default is 1.

Although multiple versions of a logical terminal can exist in the data dictionary, only one version of a logical terminal can be used by a system at runtime. If multiple ADD LTERM statements specify the same logical terminal identifier, the system uses the version that appears first in the system definition.

**AUTotask is**

Specifies whether the system is to initiate a task automatically when the logical terminal is enabled.

**Note:** Users can override the AUTOTASK specification at runtime with the DCMT VARY LTERM ONLINE command.
BATch

Specifies the logical terminal is a batch device.

- **COMMAND initiated**
  Specifies the logical terminal will process batch data by using the task code specified in the AUTOTASK parameter.

- **TERminal initiated**
  Specifies the logical terminal will process data in response to a task code entered from the batch terminal. The system leaves a read request outstanding for the terminal. If TERMINAL INITIATED is specified, the system prompt (ENTER NEXT TASK CODE) is not displayed at the terminal.

**Note:** Users can override the COMMAND/TERMINAL INITIATED specification at runtime with the DCMT VARY LTERM command.

PRInter

Specifies the terminal is a printer.

- **BANner**
  Indicates a banner page is printed with each report on the printer device defined by the PRINTER parameter. A banner page separates each report and contains information about when the report was created and who created it.
  BANNER is the default when you specify neither BANNER nor NOBANNER with the PRINTER parameter of the LTERM statement.

- **NOBanner**
  Indicates that no banner page will be printed.

- **CHECKpoint is**
  Specifies the printer checkpoint value to use when printing reports. At runtime, printer checkpoints are taken for all active reports at a specified interval. If printing is interrupted by a system abend or a hardware problem (for example, out-of-paper), the system resumes printing the report at the last checkpoint.

  - **page-count**
    Specifies the printer checkpoint interval.
    *Page-count* must be an integer in the range 0 through 32,767. A value of 0 is synonymous with OFF.

- **SYstem**
  Directs the system to use the value specified in the PRINTER CHECKPOINT parameter of the current SYSTEM statement.
  SYSTEM is the default when you omit the CHECKPOINT parameter from the PRINTER parameter of the LTERM statement.
**OFF**
Disables the printer checkpoint facility. The system resumes printing interrupted reports from the beginning of the file.

**CLASS**
Defines the printer class list for the logical terminal.

⚠️ **Note:** Users can override the CLASS specification at runtime with the DCMT VARY PRINTER command.

**ADD**
Adds one or more printer classes to the printer class list. ADD is the default when you specify neither ADD nor DELETE in the CLASS parameter of the PRINTER parameter of the LTERM statement.

**DELETE**
Deletes one or more printer classes from the printer class list. DELETE is valid only for MODIFY operations.

**(printer-class)**
Specifies one or more printer classes. *Printer-class* must be an integer in the range 1 through 64. Multiple printer classes must be separated by blanks.

**ALL**
Assigns all printer classes (that is, 1 through 64) to the logical terminal.

**NOPRINTER**
For MODIFY statements only, changes a logical terminal previously defined as a printer back to an interactive device. When you specify NOPRINTER, the system generation compiler automatically clears all other printer-related parameters such as CLASS, CHECKPOINT, and BANNER. The LTERM specification reverts to the default, INTERACTIVE BREAK.

**ENABLED**
Indicates the logical terminal is enabled at system startup.

ENABLED is the default when you specify neither ENABLED nor DISABLED in the LTERM statement.

**DISABLED**
Indicates the logical terminal is not enabled at system startup.

⚠️ **Note:** Users can override the ENABLED/DISABLED specification at runtime with the DCMT VARY LTERM command.
PRINTER CONTROL is

Specifies the printer form feed options.

SYSTEM

Specifies the printer control options established for the system are used for this printer. This is the default.

- **SCS FORMFEED is**
  Specifies the form feed sequence sent to SCS devices.
  - **FF-NL**
    Specifies the form feed sequence is "FF" (form feed) followed by "NL" (new line). FF-NL is the default.
  - **FF**
    Specifies the form feed sequence contains "FF" only.

- **ON REPORT START**
  Specifies what happens when a report starts printing.

  - **FORMFEED FOR NATIVE PRINT IS**
    Specifies the form feed processing for native mode print reports.

  - **FORMFEED FOR NON-NATIVE PRINT IS**
    Specifies the form feed processing for non-native mode print reports.

  Parameters for the two FORMFEED options are:

  - **INsert**
    A form feed is inserted, if needed.

  - **SUPpress**
    The report never starts with a form feed, i.e., if the report starts with a form feed, it is removed.

  - **NO-Insert**
    No change to the report contents is made.

- **SCS CR-LF IS**
  Suppresses the Carriage Return/Line Feed sequence at the beginning of a report transmitted to an SCS printer.

ON REPORT END

Specifies what happens when a report finishes printing.

- **FORMFEED FOR NATIVE PRINT IS**
  Specifies the form feed processing for native mode print reports.

- **FORMFEED FOR NON-NATIVE PRINT IS**
  Specifies the form feed processing for non-native mode print reports.

  Parameters for the two FORMFEED options are:

  - **INsert**
    A form feed is inserted.

  - **NO-Insert**
    No form feed is inserted.

PRiority is *logical-terminal-priority*
Specifies the logical terminal's dispatching priority. For tasks associated with a logical terminal, the specified priority is added to the priorities specified in the TASK statement (described in TASK Statement (see page 259)) to determine the priority assigned to the task at runtime.

If the sum of the above values is greater than 240, a dispatching priority of 240 is assigned at runtime. Terminal-priority must be an integer in the range 0 (lowest priority) through 240. The default is 0.

For more information about how terminal priorities are assigned, see Specifying a TERMINAL PRIORITY.

PTErm is physical-terminal-id

Specifies the physical terminal with which the logical terminal is associated.

Physical-terminal-id must be the identifier of a physical terminal previously defined in the data dictionary with a PTERM statement.

This parameter can be omitted if the logical terminal is to be associated with the current physical terminal.

⚠️ Note: Users can override the PTERM IS specification at runtime with the DCMT VARY LTERM command.

LTERM Statement Usage

Command-initiated Batch Terminals

When a system starts up, the status of a command-initiated batch terminal is offline. To activate the terminal, the user must enter a DCMT VARY LTERM ONLINE command from another terminal. The command-initiated terminal is then marked online and the AUTOTASK task executes. When the task completes execution, the terminal is marked offline, awaiting the next DCMT VARY LTERM ONLINE request. The DCMT VARY LTERM ONLINE command allows the user to specify parameters to be passed to the AUTOTASK task and to override the AUTOTASK value specified in the LTERM statement.

Specifying a TERMINAL PRIORITY

For tasks associated with a logical terminal, the terminal-priority specified on the PRIORITY parameter is added to the values specified in the PRIORITY parameter of the TASK statement to determine the priority assigned to the task at runtime. If the sum of the above values is greater than 240, a dispatching priority of 240 is assigned at runtime.

A task initiated in one of the following ways is associated with a logical terminal:

- The task code is entered by the terminal operator in response to the "Enter Next Task Code" prompt
The task is initiated as a result of a previous task specifying a next task code parameter on a DC RETURN statement or #RETURN macro.

- The task is an LTERM autotask

**Printer Classes**

A logical terminal receives print output from a physical terminal if the physical terminal's printer class appears in the logical terminal's printer class list. For example, if PRINTER CLASS 5 is specified in the PTERM statement for physical terminal PT001 and in the LTERM statement for logical terminal LT064, print output from PT001 will be routed to LT064, as illustrated next.

When multiple print classes are assigned to one logical terminal, reports are printed in order by class, starting with the lowest class and proceeding to the highest.

One printer class can be assigned to multiple logical terminals. For example, if class 2 is assigned to terminals A, B, C, and D, the system will route reports assigned to class 2 to the first printer that is not being used when the print request is issued. If all printers to which class 2 is assigned are in use, the system will queue the report and route it to the first available printer.

Printer classes assigned to no logical terminal can be used to hold reports for future printing. Users can use the DCMT VARY PRINTER or DCMT VARY REPORT command to route held reports to a printer.

### Example LTERM Statement

**Defining a Logical Terminal for a Printer**

The following statement directs the system generation compiler to add logical terminal LT02 to the data dictionary and associate it with physical terminal PR02:

```
ADD LTERM LT02
    PRINTER CLASS (5 18 11 63)
    CHECKPOINT 100
    BANNER
    PRIORITY IS 15
    PTERM IS PR02.
```

**Defining a Logical Terminal for an Interactive Device**

The following statement directs the system generation compiler to add logical terminal LT01 to the data dictionary and associate it with physical terminal PR01:
ADD LTERM LT01
   AUTOTASK IS NULL
   ENABLED
   PRIORITY IS 0
   PTERM IS PR01.

Modifying a Logical Terminal

The following statement modifies logical terminal LT01 by changing its priority to 35:

MODIFY LTERM LT01
   PRIORITY IS 35.

Deleting a Logical Terminal

The following statement deletes logical terminal LT01:

DELETE LTERM LT01.

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

Device Definitions

A device is defined by using a combination of LINE and PTERM statement parameters that uniquely identify the device and supply device-specific characteristics. The LINE and PTERM statement parameters used to define each supported device type supported are presented next in alphabetic order, based on the value specified in the LINE statement TYPE parameter.

**Note:** These parameters must be coded in addition to the parameters presented under LINE Statement (see page 281) and PTERM Statement (see page 284).

The following table summarizes the model number values associated with each physical terminal. You should refer to this table when coding the MODEL parameter of the PTERM statement.

<table>
<thead>
<tr>
<th>Physical Terminal Type</th>
<th>Associated Line Type</th>
<th>Default Value for MODEL Parameter</th>
<th>Valid Values for MODEL Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR33</td>
<td>ASYNC</td>
<td>0</td>
<td>A through Z, 0 through 9</td>
</tr>
<tr>
<td>CRT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R033</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2741</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R3275S</td>
<td>BSC2</td>
<td>2</td>
<td>1, 2</td>
</tr>
<tr>
<td>R3275</td>
<td>BSC3</td>
<td>2</td>
<td>1, 2</td>
</tr>
<tr>
<td>R3277</td>
<td></td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>Physical Terminal Type</td>
<td>Associated Line Type</td>
<td>Default Value for MODEL Parameter</td>
<td>Valid Values for MODEL Parameter</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------</td>
<td>-----------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>R3278</td>
<td></td>
<td></td>
<td>1 through 5</td>
</tr>
<tr>
<td>R3279</td>
<td></td>
<td></td>
<td>2, 3</td>
</tr>
<tr>
<td>R3284</td>
<td></td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>R3286</td>
<td></td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>R3287</td>
<td></td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>R3288</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>R3289</td>
<td></td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>L3277</td>
<td>L3270B</td>
<td>2</td>
<td>1, 2</td>
</tr>
<tr>
<td>L3278</td>
<td></td>
<td></td>
<td>1 through 5</td>
</tr>
<tr>
<td>L3279</td>
<td></td>
<td></td>
<td>2, 3</td>
</tr>
<tr>
<td>L3284</td>
<td>L3280B</td>
<td>2</td>
<td>1, 2</td>
</tr>
<tr>
<td>L3286</td>
<td></td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>L3287</td>
<td></td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>L3288</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>L3289</td>
<td></td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>S3277 S3278 S3279 S3270Q</td>
<td></td>
<td>2</td>
<td>1, 2 1 through 5 2, 3</td>
</tr>
<tr>
<td>TCAMTRM</td>
<td>TCAMLIN</td>
<td>2</td>
<td>1 through 5</td>
</tr>
<tr>
<td>V3277</td>
<td>VTAMLIN</td>
<td>2</td>
<td>1, 2</td>
</tr>
<tr>
<td>V3278</td>
<td></td>
<td></td>
<td>1 through 5</td>
</tr>
<tr>
<td>V3279</td>
<td></td>
<td></td>
<td>2, 3</td>
</tr>
<tr>
<td>V3284</td>
<td></td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>V3286</td>
<td></td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>V3287</td>
<td></td>
<td></td>
<td>1, 2</td>
</tr>
<tr>
<td>V3288</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1, 2</td>
</tr>
</tbody>
</table>
ASYNC

The ASYNC syntax is used for the LINE and PTERM statement parameters to define asynchronous start/stop devices.

ASYNC Syntax

LINE Syntax

```
  TYPE is ASYNC

  CONnect
  NOConnect

  DDname is ddname filename
```

PTERM Syntax

```
  TYPE is ASR33
  CRT
  RO33
  2741

  MODEL is model-specification

  UNIT is hex-line-address
```

ASYNC Parameters

LINE Statement Parameters

- **TYPE is ASYNC**
  Supplies the line type.

- **CONnect**
  Specifies the line is to be treated as a connect-type line.
  CONNECT is the default when you specify neither CONNECT nor NOCONNECT in the LINE statement.

- **NOConnect**
  Specifies the line is not treated as a connect-type line.

- **DDname is**
  Associates the line with an external file name.
  The DDNAME parameter is required to create an executable system.

  - **ddname**
    For z/OS systems only, identifies the ddname for the line, as specified in the system startup JCL.
filename
For z/VSE systems only, identifies the filename for the line, as specified in the system startup JCL.

PTERM Statement Parameters

• **TYPe is**
  Supplies the device type for the physical terminal. Valid values are:
  - ASR33
  - CRT
  - RO33
  - 2741

• **MODel is model-specification**
  Specifies the model of the physical terminal.
  *Model-specification* must be a single alphanumeric character.

• **UNIt is hex-line-address**
  Specifies the 3-character hexadecimal address of the line associated with the physical terminal.
  The UNIT parameter is required to create an executable system.

ASYNC Usage

Terminal Type Codes

When dialing into DC/UCF, the system prompts the dial-up asynchronous terminal operator to type in the terminal identifier. The terminal identifier is a one-character uppercase terminal type code followed by a one- through seven-character name:

- **The terminal type code** varies according to device characteristics and PTERM statement parameters
- **The name** can be chosen arbitrarily

The following table lists the PTERM type, terminal type code, and device characteristics for asynchronous terminals.

<table>
<thead>
<tr>
<th>Terminal type code</th>
<th>Device type</th>
<th>Line length</th>
<th>Formfeed character</th>
<th>Parity</th>
<th>Character set</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTERM TYPE IS ASR33 or CRT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Hardcopy teletype</td>
<td>133 char.</td>
<td>Even</td>
<td>ASCII</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>Hardcopy teletype</td>
<td>80 char.</td>
<td>Even</td>
<td>ASCII</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>Hardcopy teletype</td>
<td>158 char.</td>
<td>Even</td>
<td>ASCII</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Glass teletype</td>
<td>80 char.</td>
<td>X'0C'</td>
<td>Even</td>
<td>ASCII</td>
</tr>
<tr>
<td>D</td>
<td>Glass teletype</td>
<td>40 char.</td>
<td>X'0C'</td>
<td>Even</td>
<td>ASCII</td>
</tr>
</tbody>
</table>
Example ASYNC

The following LINE, PTERM, and LTERM parameters define a teletype device:

```
ADD LINE SDL60F
   TYPE ASYNC
   DDNAME = TTY60F.
ADD PTERM TTY001
   TYPE ASR33
   UNIT IS 233.
ADD LTERM LTE60F
   PTERM TTY001.
```

BSC2

The BSC2 syntax is used by the LINE and PTERM parameters to define binary synchronous switched devices.

BSC2 Syntax

**LINE Syntax**

```
   TYPE is BSC2
   BUFFER size is buffer-size
   CU is 2781
   DDname is ddname
   DEVaddr is SYSnnn
   ID is 'hex-response'
   NOID
   SWItched
   NONswitched
```

**PTERM Syntax for Remote BTAM Switched Devices**

```
   TYPE is R3275S
   ASCII
   EBCdic
```

### Terminal type code

<table>
<thead>
<tr>
<th>Terminal type code</th>
<th>Device type</th>
<th>Line length</th>
<th>Formfeed character</th>
<th>Parity</th>
<th>Character set</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>Glass teletype</td>
<td>80 char.</td>
<td>X'18'</td>
<td>Odd</td>
<td>ASCII</td>
</tr>
</tbody>
</table>

**PTERM TYPE IS 2741**

<table>
<thead>
<tr>
<th>Device type</th>
<th>Line length</th>
<th>Formfeed character</th>
<th>Parity</th>
<th>Character set</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>IBM 2741</td>
<td>Even</td>
<td>PTTC/BCD</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>IBM 2741</td>
<td>Even</td>
<td>PTTC/EBCDIC</td>
<td></td>
</tr>
</tbody>
</table>
PTERM Syntax for Remote 3741 Devices

TYPE is R3741S

ECF

NOECF

FORMfeed

NOFormfeed

ID is 'hex-identifier'

NOId

INPut

NOINput

OUTput

NOOutput

PHOne number is 'telephone-number'

NOPhone

PTERM Syntax for Remote 3780 Devices

TYPE is R3780S

ASCII

EBCdic

COMpression

NOCompression

ECF

NOECF

EXClusive

NOExclusive

FORMfeed

NOFormfeed

ID is 'hex-identifier'

NOId

INPut

NOINput

OUTput

NOOutput

PHOne number is 'telephone-number'

NOPhone

SELection address is 'hex-address'

NOSelection
BSC2 Parameters

LINE Statement Parameters

- **TYPe is BSC2**
  Supplies the line type.

- **BUFfer size is buffer-size**
  Specifies the line I/O page buffer size, in bytes, for the line. Buffer-size must be an integer in the range 1 through 32,767.

- **CU is**
  For z/VSE systems only, specifies the control unit device type. Valid values are 2701 and 2703. The default is 2701.

- **DDname is ddname**
  For z/OS systems only, specifies the ddname for the line, as specified in the system startup JCL. The DDNAME parameter is required under z/OS to create an executable system.

- **DEVaddr is SYSnnn**
  For z/VSE systems only, specifies the logical unit assignment of the device, as specified in the system startup JCL. The DEVADDR parameter is required under z/VSE to create an executable system.

  - **THRough SYSnnn**
    Specifies a range of logical unit assignments, beginning with the value specified in the DEVADDR parameter and ending with the value specified in the THROUGH parameter.

- **ID is 'hex-response'**
  Specifies the system is to send a response to terminals when they dial into the line. Hex-response must be the 2- through 30-character hexadecimal response enclosed in site-standard quotation marks.

- **NOID**
  Specifies the system will not send a response to terminals when they dial into the line. NOID is the default when you specify neither ID nor NOID in the LINE statement.

- **SWItched**
  Indicates the connections to physical terminals associated with the line are switched (that is, connections are established by dialing). SWITCHED is the default when you specify neither SWITCHED nor NONSWITCHED in the LINE statement.

- **NONswitched**
  Indicates the connections to physical terminals associated with the line are nonswitched (that is, leased).

PTERM Statement Parameters for Remote BTAM Switched Devices

- **TYPe is R3275S**
  Specifies the device type for the physical terminal.
- **ASCII**
  Specifies the physical terminal supports the ASCII character set.

- **EBCDIC**
  Specifies the physical terminal supports the EBCDIC character set.

- **ID IS 'hex-identifier'**
  Specifies the physical terminal identifier that will be transmitted to the DC/UCF system when connection is established.
  *Hex-identifier* must be the 2- through 30-character hexadecimal identifier enclosed in site-standard quotation marks.
  The ID parameter is required to create an executable system.

- **MODE**
  Specifies the model number of the physical terminal.
  Valid values are 1 and 2. The default is 2.

- **PHONE number is telephone-number**
  Specifies the system will dial up the terminal automatically, using the specified number.
  *Telephone-number* must be a one- through 16-character value enclosed in site-standard quotation marks. Blanks and hyphens can be inserted when applicable. At least one numeric character must be specified.

- **NOPhone**
  Specifies the system will not dial up the terminal automatically. Connection must be established manually.
  NOPHONE is the default when you specify neither PHONE NUMBER nor NOPHONE in the PTERM statement.

**PTERM Statement Parameters for Remote 3741 Devices**

- **TYPE is R3741S**
  Specifies the device type for the physical terminal.

- **ECF**
  Specifies logical records are blocked.

- **NOECF**
  Specifies logical records are not blocked.
  NOECF is the default when you specify neither ECF nor NOECF in the PTERM statement.

- **FORMFEED**
  Specifies the physical terminal has formfeed capabilities.

- **NOFORMFEED**
  Specifies the physical terminal does not have formfeed capabilities.
  NOFORMFEED is the default when you specify neither FORMFEED nor NOFORMFEED in the PTERM statement.

- **ID IS 'hex-identifier'**
  Specifies the terminal will transmit an identifier to the system when connection is established.
  *Hex-identifier* must be the 2- through 30-character hexadecimal identifier enclosed in site-standard quotation marks.
- **NOId**
  Specifies the terminal will not transmit an identifier to the system when connection is established. NOID is the default when you specify neither ID nor NOID in the PTERM statement.

- **INPut**
  Specifies the system will read input from the physical terminal.

- **NOINput**
  Specifies the system will not read input from the physical terminal. NOINPUT is the default when you specify neither INPUT nor NOINPUT in the PTERM statement.

- **OUTput**
  Specifies the system will write output to the physical terminal.

- **NOOutput**
  Specifies the system will not write output to the physical terminal. NOOUTPUT is the default when you specify neither OUTPUT nor NOOUTPUT in the PTERM statement.

- **PHOne number is ‘telephone-number’**
  Specifies the system will dial up the terminal automatically, using the specified number. Telephone-number-q must be a one- through 16-character value enclosed in site-standard quotation marks. Blanks and hyphens can be inserted as desired. At least one numeric character must be specified.

- **NOPhone**
  Specifies the system will not dial up the terminal automatically. Connection must be established manually. NOPHONE is the default when you specify neither PHONE NUMBER nor NOPHONE in the PTERM statement.

**PTERM statement parameters for remote 3780 devices**

- **TYPe is R3780S**
  Specifies the device type for the physical terminal.

- **ASCii**
  Specifies the physical terminal supports the ASCII character set.

- **EBCdic**
  Specifies the physical terminal will support the EBCDIC character set. EBCDIC is the default when you specify neither ASCII nor EBCDIC in the PTERM statement.

- **COMpression**
  Specifies the system will use the space compression feature when writing output to the physical terminal.

- **NOCompress**
  Specifies the system will not use the space compression feature when writing output to the physical terminal. NOCOMPRESSION is the default when you specify neither COMPRESSION nor NOCOMPRESSION in the PTERM statement.
ECF
Specifies logical records are blocked.
NOECF is the default when you specify neither ECF nor NOECF in the PTERM statement.

NOECF
Specifies logical records are not blocked.

EXClusive
Specifies the remote 3780 device will have exclusive use of the line for the duration of input data-stream transmission.

NOExclusive
Specifies the remote 3780 device will not have exclusive use of the line for the duration of input data-stream transmission.
NOEXCLUSIVE is the default when you specify neither EXCLUSIVE nor NOEXCLUSIVE in the PTERM statement.

FORmfeed
Specifies the physical terminal has formfeed capabilities.

NOFormfeed
Specifies the physical terminal does not have formfeed capabilities.
NOFORMFEED is the default when you specify neither FORMFEED nor NOFORMFEED in the PTERM statement.

ID is 'hex-identifier'
Specifies the physical terminal will transmit an identifier to the system when connection is established.
Hex-identifier must be the 2- through 30-character hexadecimal response enclosed in site-standard quotation marks.

NOId
Specifies the terminal will not transmit an identifier to the system when connection is established.
NOID is the default when you specify neither ID nor NOID in the PTERM statement.

INPut
Specifies the system will read input from the physical terminal.

NOINput
Specifies the system will not read input from the physical terminal.
NOINPUT is the default when you specify neither INPUT nor NOINPUT in the PTERM statement.

OUTput
Specifies the system will write output to the physical terminal.

NOOuput
Specifies the system will not write output to the physical terminal.
NOOUTPUT is the default when you specify neither OUTPUT nor NOOUTPUT in the PTERM statement.
- **PHOne number is 'telephone-number'**
  Specifies the system dials up the terminal automatically, using the specified number. 
  *Phone-number-q* must be a one- through 16-character value enclosed in site-standard quotation marks. Blanks and hyphens can be inserted when applicable. At least one numeric character must be specified.

- **NOPhone**
  Specifies the system will not dial up the terminal automatically. Connection must be established manually. 
  NOPHONE is the default when you specify neither PHONE NUMBER nor NOPHONE in the PTERM statement.

- **SELection address is 'hex-address'**
  Specifies the physical terminal uses the 3780 component selection feature. 
  *Hex-address* must be a 2-character hexadecimal literal enclosed in site-standard quotation marks.

- **NOSelection**
  Specifies the physical terminal will not use the 3780 component selection feature. 
  NOSELECTION is the default when you specify neither SELECTION ADDRESS nor NOSELECTION in the PTERM statement.

### BSC3

The BSC3 syntax for the LINE and PTERM parameters is used to define binary synchronous leased multipoint or remote 3270 devices.

#### BSC3 Syntax

**LINE Syntax**

```plaintext
<table>
<thead>
<tr>
<th>Type is BSC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer size is buffer-size</td>
</tr>
<tr>
<td>CU is 2701, 2703</td>
</tr>
<tr>
<td>DDname is ddname</td>
</tr>
<tr>
<td>DEVaddr is SYSnnn, THroughe SYSnnn</td>
</tr>
<tr>
<td>Polling delay is 0 delay-time</td>
</tr>
</tbody>
</table>
```

**PTERM Syntax for Remote 3270 Devices**

```plaintext
| Type is R3275, R3277, R3278, R3279 |
```
PTERM Syntax for Remote 3280 Devices

- **TYPE is**
  - R3284
  - R3286
  - R3287
  - R3288
  - R3289

PTERM Syntax for 3741 Devices

- **TYPE is** R3741

PTERM Syntax for Remote 3780 Devices

- **TYPE is** R3780
BSC3 Parameters

LINE Statement Parameters

- **TYPe is BSC3**
  Specifies the line type.

- **BUFFer size is buffer-size**
  Specifies the line I/O page buffer size, in bytes, for the line.
  *Buffer-size* must be an integer in the range 1 through 32,767.

- **CU is**
  *For z/VSE systems only, specifies the control unit device type.*
  Valid values are 2701 and 2703. The default is 2701.

- **DDname is ddname**
  *For z/OS systems only, specifies the ddname for the line, as specified in the system startup JCL.*
  The DDNAME parameter is required under z/OS to create an executable system.

- **DEVaddr is SYSnnn**
  *For z/VSE systems only, specifies the logical unit assignment of the device, as specified in the system startup JCL.*
  The DEVADDR parameter is required under z/VSE to create an executable system.

  - **THRough SYSnnn**
    Specifies a range of logical unit assignments, beginning with the value specified in the DEVADDR parameter and ending with the value specified in the THROUGH parameter.

- **POlling delay is delay-time**
  Specifies the interval, in wall-clock seconds, at which the line will be polled.
  *Delay-time* must be an integer in the range 0 through 2,147,483,647. The default, 0, specifies the line is polled continuously.

PTERM Statement Parameters for Remote 3270 Devices

- **TYPe is**
  Specifies the device type for the physical terminal. Valid values are:

  - R3275
  - R3277
  - R3278
  - R3279
- **ASCII**
  Specifies the physical terminal supports the ASCII character set.

- **EBCDIC**
  Specifies the physical terminal supports the EBCDIC character set.
  EBCDIC is the default when you specify neither ASCII nor EBCDIC in the PTERM statement.

- **MODEL is `model-number`**
  Specifies the model number of the physical terminal. Valid model numbers for each physical terminal type are:

<table>
<thead>
<tr>
<th>Physical terminal type</th>
<th>Model number</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3275</td>
<td>1 and 2</td>
</tr>
<tr>
<td>R3277</td>
<td>1 and 2</td>
</tr>
<tr>
<td>R3278</td>
<td>1 through 5</td>
</tr>
<tr>
<td>R3279</td>
<td>2 and 3</td>
</tr>
</tbody>
</table>

The default is 2 for all physical terminal types.

- **POLing address is `'hex-address-q'`**
  Specifies the address the system uses to read input from the physical terminal.
  `Hex-address` must be a 4-character hexadecimal value enclosed in site-standard quotation marks.
  The POLLING ADDRESS parameter is required to create an executable system.

- **RLN is `relative-line-number`**
  Specifies the line to which the physical terminal is connected in a BTAM BSC3 line group.
  Under z/OS, `relative-line-number` must be an integer in the range 1 through 255. The default is 1.
  Under z/VSE, `relative-line-number` must be an integer in the range 0 through 255. The default is 0.

**PTERM Statement Parameters for Remote 3280 Device Types**

- **TYPE is**
  Specifies the device type for the physical terminal. Valid values are:

  - R3284
  - R3286
  - R3287
  - R3288
  - R3289

- **ASCII**
  Specifies the physical terminal supports the ASCII character set.

- **EBCDIC**
  Specifies the physical terminal supports the EBCDIC character set.
  EBCDIC is the default when you specify neither ASCII nor EBCDIC in the PTERM statement.
FORmfeed
Specifies the physical terminal has formfeed capabilities.
The FORMFEED parameter is invalid for physical terminal type R3284.

NOFormfeed
Specifies the physical terminal does not have formfeed capabilities.
NOFORMFEED is the default when you specify neither FORMFEED nor NOFORMFEED in the
PTERM statement.

MODel is model-number
Specifies the model number of the physical terminal. Valid model numbers for each physical
terminal type are:

<table>
<thead>
<tr>
<th>Physical terminal type</th>
<th>Model number</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3284</td>
<td>1 and 2</td>
</tr>
<tr>
<td>R3286</td>
<td>1 and 2</td>
</tr>
<tr>
<td>R3287</td>
<td>1 and 2</td>
</tr>
<tr>
<td>R3288</td>
<td>2</td>
</tr>
<tr>
<td>R3288</td>
<td>1 and 2</td>
</tr>
</tbody>
</table>

The default is 2 for all physical terminal types.

POLling address is 'hex-address-q'
Specifies the address the system uses to read input from the physical terminal.
Hex-address must be a 4-character hexadecimal value enclosed in site-standard quotation marks.
The POLLING ADDRESS parameter is required to create an executable system.

RLN is relative-line-number
Specifies the line to which the physical terminal is connected in a BTAM BSC3 line group.
Under z/OS, relative-line-number must be an integer in the range 1 through 255. The default is 1.
Under z/VSE, relative-line-number must be an integer in the range 0 through 255. The default is 0.

PTERM Statement Parameters for 3741 Devices

TYPe is R3741
Specifies the device type for the physical terminal.

FORmfeed
Specifies the physical terminal has formfeed capabilities.

NOFormfeed
Specifies the physical terminal does not have formfeed capabilities.
NOFORMFEED is the default when you specify neither FORMFEED nor NOFORMFEED in the
PTERM statement.

POLling address is 'hex-address'
Specifies the system will read input from the physical terminal at the specified address.
Hex-address must be a 4- through 8-character value enclosed in site-standard quotation marks.
• **NOPolling**
  The system will not read input from the physical terminal.
  NOPOLLING is the default when you specify neither POLLING ADDRESS nor NOPOLLING in the PTERM statement.

• **RLN is relative-line-number**
  Specifies the line to which the physical terminal is connected in a BTAM BSC3 line group.
  Under z/OS, relative-line-number must be an integer in the range 1 through 255. The default is 1.
  Under z/VSE, relative-line-number must be an integer in the range 0 through 255. The default is 0.

• **SElection address is 'hex-address'**
  Specifies the system will write output to the physical terminal.
  Hex-address must be a 4- through 8-character hexadecimal value enclosed in site-standard quotation marks.

• **NOSelection**
  Specifies the system will not write output to the physical terminal.
  NOSELECTION is the default when you specify neither SELECTION ADDRESS nor NOSELECTION in the PTERM statement.

**PTERM Statement Parameters for 3780 Devices**

• **TYPe is R3780**
  Specifies the device type for the physical terminal.

• **ASCii**
  Specifies the physical terminal supports the ASCII character set.

• **EBCdic**
  Specifies the physical terminal supports the EBCDIC character set.
  EBCDIC is the default when you specify neither ASCII nor EBCDIC in the PTERM statement.

• **COMpression**
  Specifies the system will use the space compression feature when writing output to the physical terminal.

• **NOCompression**
  Specifies the system will not use the space compression feature when writing output to the physical terminal.
  NOCOMPRESS is the default when you specify neither COMPRESSION nor NOCOMPRESS in the PTERM statement.

• **EXClusive**
  Specifies the remote 3780 device has exclusive use of the line for the duration of input data-stream transmission.

• **NOExclusive**
  Specifies the remote 3780 device does not have exclusive use of the line for the duration of input data-stream transmission.
  NOEXCLUSIVE is the default when you specify neither EXCLUSIVE nor NOEXCLUSIVE in the PTERM statement.
- **FORMfeed**
  Specifies the physical terminal has formfeed capabilities.

- **NOFormfeed**
  Specifies the physical terminal does not have formfeed capabilities.
  NOFORMFEED is the default when you specify neither FORMFEED nor NOFORMFEED in the PTERM statement.

- **INQury**
  Specifies the terminal will transmit data to the system in inquiry mode.

- **NOInqury**
  Specifies the terminal will not transmit data to the system in inquiry mode.
  NOINQUIRY is the default when you specify neither INQUIRY nor NOINQUIRY in the PTERM statement.

- **POLling address is 'hex-address'**
  Specifies the system will read input from the physical terminal at the specified address.
  *Hex-address* must be a 4- through 8-character value enclosed in site-standard quotation marks.

- **NOPolling**
  The system will not read input from the physical terminal.
  NOPOLLING is the default when you specify neither POLLING ADDRESS nor NOPOLLING in the PTERM statement.

- **RLN is relative-line-number**
  Specifies the line to which the physical terminal is connected in a BTAM BSC3 line group.
  Under z/OS, *relative-line-number* must be an integer in the range 1 through 255. The default is 1.
  Under z/VSE, *relative-line-number* must be an integer in the range 0 through 255. The default is 0.

- **SELecton address is 'hex-address'**
  Specifies the system will write output to the physical terminal.
  *Hex-address* must be a 4- through 8-character hexadecimal value enclosed in site-standard quotation marks.

- **NOSelection**
  Specifies the system will not write output to the physical terminal.
  NOSELECTION is the default when you specify neither SELECTION ADDRESS nor NOSELECTION in the PTERM statement.

### Example BSC3

The following LINE, PTERM, and LTERM statements define a remote 3270 line with two 3277 devices:

```plaintext
ADD LINE R3270B1
  TYPE IS BSC3
  DDNAME IS R3270B1.

ADD PTERM PR3270B1
  TYPE IS R3277
  MODEL IS 2
  POLLING ADDRESS IS '4040'.
ADD LTERM LR3270B1
  PTERM IS PR3270B1.
```
Add PTERM PR3270B2
    TYPE IS R3277
    MODEL IS 2
    POLLING ADDRESS IS '40C1'.
Add LTERM LR3270B2
    PTERM IS PR3270B2.

CCI

CCI lines (CA IDMS DDS and CA IDMS Server users only) are used to connect a DC/UCF system (host node) with another DC/UCF system (target node) located on a different CPU in the CA IDMS communications network. To include a DC/UCF system in the DDS network, you must explicitly define the system as a node with an access type of CCI on the system generation NODE statement. Additionally, you define the resources (databases and systems) that can be accessed by DDS using the system generation RESOURCE TABLE statement. The NODE (NODE Statement) and RESOURCE TABLE (RESOURCE TABLE Statement) statements are described in System Generation Statements (see page 180).

You define one CCI line for the DC/UCF system you are defining.

You define a PTERM and LTERM pair for the maximum number of concurrent DDS user sessions you will allow within a DC/UCF system.

To define an LTERM, simply add the LTERM and the associated PTERM. There are default LTERM options the DC/UCF system uses at runtime.

The CCI line manages the physical terminals. The physical terminals perform the actual communication between CPUs.

CCI Syntax

LINE Syntax

►►─── TYPe is CCI ────►◄

PTERM Syntax

►►─── TYPe is BULk ────►◄

CCI Parameters

LINE Statement Parameters

- **TYPe is CCI**
  
  Identifies the line as a CCI line that is used to connect two DC/UCF systems (nodes). This parameter is required for ADD operations.
• **TYPe is BULK**
  Specifies the data transfer between DC/UCF systems is bulk.
  The TYPE parameter is required for ADD operations.

**CONSOLE**

The CONSOLE syntax for the LINE and PTERM parameters is used to define the operator's console.
The teleprocessing network can include only one console definition. To define the operator's console, the LINE identifier must be CONSOLE, the PTERM identifier must be OPERATOR, and the LTERM identifier must be CONSOLE.

**CONSOLE Syntax**

**LINE Syntax**

►►─── TYPe is CONSOLE ───────────────────────────────────────────────────►◄

**PTERM Syntax**

►►─── TYPe is OPERATOR ────────────────────────────────────────────────►◄

**CONSOLE Parameter**

**LINE Statement Parameter**

• **TYPe is CONSOLE**
  Specifies the line type.
  The LINE identifier, as described in LINE Statement (see page 281), must be CONSOLE to define the operator's console.

**PTERM Statement Parameter**

• **TYPe is OPERATOR**
  Specifies the device type for the physical terminal.
  The PTERM identifier, as described in PTERM Statement (see page 284), must be OPERATOR to define the operator's console.

**Example 1**

The following LINE, PTERM, and LTERM statements define a simulated 3270 line and terminal:

```
ADD LINE S3270Q1
  TYPE IS S3270Q
  INPUT DDNAME IS SIMIN1
  OUTPUT DDNAME IS SIMOUT1.

ADD PTERM PS3270Q1
  TYPE IS S3277
  MODEL IS 2.
```
ADD LTERM LS3270Q1
PTERM IS PS3270Q1.

DDS

DDS lines are used to connect a DC/UCF system with another DC/UCF system in the DDS network, using the VTAM access method.

A DDS line, together with a physical terminal, creates a port through which the system passes request and response packets to another node. In order for communication to occur between two DDS nodes, a port must be defined for each node.

Syntax for the LINE and PTERM parameters used to define DDS lines is presented below. An LTERM statement is not required to define a VTAM PTERM.

DDS Syntax

LINE Syntax

►►─── TYPe is DDS ─────────────────────────────────────────────────────────────►
►─┬──────────────────────────────────┬───────────────────────────────────────►◄
└─ SOUrce is application-id ─┘

PTERM Syntax

►►─────── TYPE ─┬─ BULK ─┬───────────────────────────────────────────────────►
└─ VTAM ┘
►───┬───────────────────────────────┬────────────────────────────────────────►
└─ BLOcksize ─┬─ 8192 ◄───────┬─┘
└─ block-size ┘
►───┬──────────────────────────────────────┬─────────────────────────────────►◄
└─ TARget application application-id ┘
►───┬──────────────────────────────────────┬─────────►◄
└─ WEIght factor ─┬─ 0 ◄─────────────┬─┘
└─ weight-factor ┘

DDS Parameters

LINE Statement Parameters

- **TYPe is DDS**
  Specifies DDS as the line type.

- **SOUrce is application-id**
  Specifies the VTAM LUNAME of the DDS application to be created by the line driver.

PTERM Statement Parameters

- **TYPe**
BULK
Specifies the data transfer between DC/UCF systems is BULK. You must define at least two PTERMS with a TYPE of BULK.

VTAM
Specifies the PTE that will be used to transmit packets through a VTAM communications link. Specify VTAM if the connection being established is between two nodes residing on the same or different machines connected by VTAM. You must define at least one PTERM with a TYPE of VTAM.
If you specify VTAM, you must specify the TARGET parameter and optionally the BLOCKSIZE and WEIGHT parameters, which are described below. For additional information about DDS VTAM considerations, see VTAM Considerations (see page 359) and the CA IDMS DDS Design and Operations section.

BLOCKsize
Specifies the size of the buffer to use to pass run unit request and response packets between nodes. If two connected nodes specify different block sizes, DDS uses the smaller size.

8192
Specifies the default block size.

block-size
Specify an integer in the range 1 through 32,767.

TARGET application application-id
Specifies the VTAM LUNAME of the destination application program. Application-id is a unique one- through eight-character name that corresponds to an ACB name or label specified in an APPL statement. This name must correspond to the application-id specified on the DDS LINE statement SOURCE parameter for the target node.

WEIGHT factor
Specifies the priority to assign to the port defined by this physical terminal and its associated line.

0
Specifies the default weight-factor.

weight-factor
Specify an integer in the range 0 through 999,999,999.

DDS Usage

Defining DDS Lines
One LINE statement must be coded for each DDS node to which the system is to be directly attached, except under VTAM. Under VTAM, one LINE statement can be used to directly connect a DDS node to one or more nodes.

One PTERM statement must be coded for each DDS line, with the exception of VTAM connections. Under VTAM, one or more physical terminals can be defined for each line.
Specifying a Weight Factor

When selecting a port through which to pass request and response packets, the node bases its selection on the weight factor specified for the port that connects the system to the target node. A port can directly connect the system to the target node or to a node that, in turn, is connected to the target node either directly or indirectly. If such multiple paths exist, the host node selects the path having the lowest cumulative weight factor.

For more information about weight factors, see the CA IDMS DDS Design and Operations section.

Example DDS

Defining a DDS Port that Communicates using VTAM

The following statements define a DDS port that communicates through a VTAM line. In the system in New York, define:

ADD LINE NY2BOS
  TYPE DDS
  SOURCE DDSVTM01.

ADD PTERM PT001
  TYPE VTAM
  BLOCKSIZE 8192
  TARGET DDSVTM02
  WEIGHT FACTOR 10.

In the system located in Boston, define:

ADD LINE BOS2NY
  TYPE DDS
  SOURCE DDSVTM02.

ADD PTERM PT001
  TYPE VTAM
  TARGET DDSVTM01.

Modifying the Weight Factor

The following statement modifies physical terminal PT001, increasing its weight factor to 25:

MODIFY PTERM PT001
  WEIGHT FACTOR 25.

Deleting a DDS Port

The following statement deletes the DDS port that communicates through line NY2BOS:

DELETE PTERM PT001.
DELETE LINE NY2BOS.

INOUTL

The INOUTL syntax for the LINE and PTERM parameters is used to define SYSIN/SYSOUT devices.
INOUTL Syntax

LINE Syntax

```
►►─── TYPe is INOUTL ──────────────────────────────────────────────────────────►
►─┬───────────────────────────────────┬───────────────────────────────────────►
└─ INput ddname is ─┬─
   │  ddname ──┬─
   │   filename ──┐
   │   linkname ─┘
   └─ filename ┘

◄─┬───────────────────────────────────┬──────────────────────────────────────►◄
└─ OUTput ddname is ─┬─
   │  ddname ──┬─
   │   filename ──┐
   │   linkname ─┘
   └─ filename ┘
```

PTERM Syntax

```
►►─── TYPe is INOUTT ──────────────────────────────────────────────────────────►
►─┬─────────────────────────────────────────────────────┬────────────────────►
└─ DELete character is ─┬─ C' delete-character' ──┬─ X' hex-delete-character' ┘
   └─ hex-delete-character ┘

◄─┬───────────────────────────────────────────────────┬──────────────────────►◄
└─ FLUsh character is ─┬─ C' flush-character' ──┬─ X' hex-flush-character' ┘
   └─ hex-flush-character ┘

►─┬────────────────────────────────────┬─────────────────────────────────────►◄
└─ PAGe length is ─┬─ 24 ┘
   └─ page-length ┘
```

INOUTL Parameters

LINE Statement Parameters

- **TYPe is INOUTL**
  Specifies the line type.

- **INput ddname is**
  Specifies the external name of the input file for the line, as specified in the system startup JCL. The INPUT DDNAME IS parameter is required to create an executable system.

  - **ddname**
    For z/OS systems, specifies the ddname of the input file for the line.

  - **filename**
    For z/VSE systems, specifies the filename of the input file for the line.

- **OUTput ddname is**
  Specifies the external name of the output file for the line, as specified in the system startup JCL. The OUTPUT DDNAME IS parameter is required to create an executable system.

  - **ddname**
    For z/OS systems, specifies the ddname of the output file for the line.

  - **filename**
    For z/VSE systems, specifies the filename of the output file for the line.
PTERM Statement Parameters

- **TYPE is INOUTT**
  Specifies the device type for the physical terminal.

- **DELETE character is**
  Specifies the control character to be used to delete characters.
  - `'delete-character'`
    Specifies the delete character as a one-character literal.
  - `'hex-delete-character'`
    Specifies the delete character as a 2-digit hexadecimal literal.

- **FLUSH character is**
  Specifies the control character used to delete lines.
  - `'flush-character'`
    Specifies the flush character as a one-character literal.
  - `'hex-flush-character'`
    Specifies the flush character as a 2-digit hexadecimal literal.

- **PAGE length is page-length**
  Specifies the maximum page size, in text lines, for the physical terminal.
  *Page-length* must be an integer in the range 1 through 32,767. The default is 24.

### INOUTL Usage

**z/OS DD Statements for INOUTL Lines**

For z/OS systems, the DD statement for the input file for an INOUTL line must include the following DCB specifications:

RECFM=F, BLKSIZE=80, LRECL=80

The DD statement for the output file for an INOUTL line must include the following DCB specifications:

RECFM=VBA, BLKSIZE=141, LRECL=137

### INOUTL Example

The following LINE, PTERM, and LTERM statements define a simulated 3270 line and terminal:

ADD LINE S3270Q1
  TYPE IS S3270Q
  INPUT DDNAME IS SIMIN1
  OUTPUT DDNAME IS SIMOUT1.

ADD PTERM PS3270Q1
  TYPE IS S3277
LAPPCEMU

The following syntax for LINE and PTERM parameters is used to define emulated APPC.

LAPPCEMU Syntax

LINE Syntax

►►─── TYPe is LAPPCEMU ────────────────────────────────────────────────────────►◄

PTERM Syntax

►►─── TYPe is PAPPCEMU ────────────────────────────────────────────────────────►◄

LAPPCEMU Parameters

LINE Statement Parameters

- TYPe is LAPPCEMU
  Specifies the line type.

PTERM Statement Parameters

- TYPe is PAPPCEMU
  Specifies the physical terminal type as PAPPCEMU.
  You should associate only one physical terminal with an LAPPCEMU line.

Example LAPPCEMU

Sample Line and Terminal Definitions

The following DC/UCF system generation statements define a line and associated physical terminal to support emulated APPC:

ADD LINE APPCLIN
  TYPE IS LAPPCEMU
  ENABLED.

ADD PTERM APPCPTE
  TYPE IS PAPPCEMU
  ENABLED.

ADD LTERM APPCLTE
  ENABLED.
L3270B

The following syntax for LINE and PTERM parameters is used to define local 3270 devices using BTAM.

L3270B Syntax

LINE Syntax

```
>>> TYPE is L3270B
```

```
BUFFfer size is buffer-size
```

```
DDname is ddname
```

PTERM Syntax

```
>>> TYPE is
```

```
L3277
L3278
L3279
```

```
DEVaddr is SYSnnn
```

```
MODel is 2 model-number
```

L3270B Parameters

LINE Statement Parameters

- **TYPE is L3270B**
  Specifies the line type.

- **BUFFfer size is buffer-size**
  Specifies the line I/O page buffer size, in bytes, for the line.
  *Buffer-size* must be an integer in the range 1 through 32,767.

- **DDname is ddname**
  For z/OS systems only, specifies the ddname for the line, as specified in the system startup JCL. The DDNAME parameter is required under z/OS to create an executable system.

PTERM Statement Parameters

- **TYPE is**
  Specifies the device type for the physical terminal. Valid values are:

  - L3277
  - L3278
  - L3279
• **DEVaddr is SYSnnn**
  For z/VSE systems only, specifies the logical unit assignment of the device, as specified in the system startup JCL. This parameter is required under z/VSE to create an executable system.

• **MODel is model-number**
  Specifies the model number of the physical terminal. Valid model numbers for each physical terminal type are:

<table>
<thead>
<tr>
<th>Physical terminal type</th>
<th>Model number</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3277</td>
<td>1 and 2</td>
</tr>
<tr>
<td>L3278</td>
<td>1 through 5</td>
</tr>
<tr>
<td>L3279</td>
<td>2 and 3</td>
</tr>
</tbody>
</table>

The default is 2 for all physical terminal types.

**Example 3**

The following LINE, PTERM, and LTERM statements define a simulated 3270 line and terminal:

```plaintext
ADD LINE S3270Q1
   TYPE IS S3270Q
   INPUT DDNAME IS SIMIN1
   OUTPUT DDNAME IS SIMOUT1.

ADD PTERM PS3270Q1
   TYPE IS S3277
   MODEL IS 2.
ADD LTERM LS3270Q1
   PTERM IS PS3270Q1.
```

**L3280B**

The following syntax for LINE and PTERM statements is used to define local 3280 devices using BTAM.

**L3280B Syntax**

**LINE Syntax**

```
  TYPE is L3280B
  DDname is ddname
```

**PTERM Syntax**

```
  TYPE is
    L3284
    L3286
    L3287
    L3288
    L3289
```
L3280B Parameters

LINE Statement Parameters

- **TYPE** is **L3280B**
  Specifies the line type.

- **DDname**
  Associates the line with an external file name.
  The DDNAME parameter is required to create an executable system.

  - **ddname**
    For z/OS systems only, identifies the ddname for the line, as specified in the system startup JCL.

  - **filename**
    For z/VSE systems only, identifies the filename for the line, as specified in the system startup JCL.

PTERM Statement Parameters

- **TYPE** is
  Specifies the device type for the physical terminal. Valid values are:

  - L3284
  - L3286
  - L3287
  - L3288
  - L3289

- **DEVaddr is SYSnnn**
  For z/VSE systems only, specifies the logical unit assignment of the device, as specified in the system startup JCL.
  This parameter is required to create an executable system.

**Note:** The z/VSE IPL ADD statement must define the printer device as a 3277 without a mode '01'.
FORmfeed
Specifies the physical terminal has formfeed capabilities.

NOFormfeed
Specifies the physical terminal does not have formfeed capabilities.
NOFORMFEED is the default when you specify neither FORMFEED nor NOFORMFEED in the PTERM statement.

MODel is model-number
Specifies the model number of the physical terminal. Valid model numbers for each physical terminal type are:

<table>
<thead>
<tr>
<th>Physical Terminal Type</th>
<th>Model Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3284</td>
<td>1 and 2</td>
</tr>
<tr>
<td>R3286</td>
<td>1 and 2</td>
</tr>
<tr>
<td>R3287</td>
<td>1 and 2</td>
</tr>
<tr>
<td>R3288</td>
<td>2</td>
</tr>
<tr>
<td>R3289</td>
<td>1 and 2</td>
</tr>
</tbody>
</table>

The default is 2 for all physical terminal types.

Example L3280B

The following LINE, PTERM, and LTERM statements define a local 3280-type printer:

ADD LINE PRTLINE
  TYPE IS L3280B
  DDNAME IS BTAM3280.

ADD PTERM PRTPT1
  TYPE IS L3287
  FORMFEED.
ADD LTERM PRRTL1
  PRINTER CLASS (2,11,13,42).

SOCKET

SOCKET lines are used to support Generic Listening and DDS using TCP/IP. Generic Listening is enabled by defining one or more LISTENER PTERMs and a number of BULK PTERMs. DDS using TCP/IP is enabled by defining one DDSTCPIP PTERM for each remote node. Associate an LTERM with each PTERM.

Each connection request to a LISTENER PTERM requires the use of a BULK PTERM/LTERM pair within the TCP/IP line. Define as many BULK PTERM/LTERM pairs as the expected maximum number of concurrent connections.

Notes:
1. If you are exploiting the suspend/resume capability (in other words, tasks terminate without closing their socket), the required number of BULK PTERM/LTERM pairs can become rather large.

2. You can use the REPEAT COUNT clause of the PTERM statement to facilitate the definition of multiple PTERM/LTERM pairs.

**SOCKET Syntax**

**LINE Syntax**

```plaintext
►►TyP e is SOCKET
  ▼MODule is plug-in
```

**LISTENER PTERM Syntax**

```plaintext
►► TyP e is LISTENER
  ▼PORT is listener-port-identifier
  ▼BACKlog is backlog
  ▼IP STAck NAME is stack-ip-name
  ▼ADDress is 'stack-ip-address'
  ▼NAMe is stack-host-name
  ▼TASk is task-code
  ▼PARM is 'string'
  ▼MAXimum NUMber of CONnections is OFF max-number-conn
```

**DDSTCPIP PTERM Syntax**

```plaintext
►► TyP e is DDSTCPIP
  ▼TARget ADDress is 'target-ip-address'
  ▼NAMe is target-host-name
  ▼TARget PORT is target-port-identifier
  ▼IP STack NAME is stack-ip-name
  ▼PORT RANge is OFF start-port-number THRu end-port-number
  ▼IDLe INTerval is idle-interval
  ▼MAXimum NUMber of CONnections is OFF max-number-conn
  ▼NUMber of PERmanent CONnections is perm-conn-number
```

**BULK PTERM Syntax**

```plaintext
►► TyP e is BULK
```
SOCKET Parameters

LINE Statement Parameters

- **TYPe is SOCKET**
  Identifies the line as a TCP/IP line. An active SOCKET line is required to run any program that uses TCP/IP sockets.

- **plug-in**
  Name of the plug-in module that implements support for specific TCP/IP stack implementations. Possible values are:
  - **RHDCD1IP**
    Supports most common TCP/IP stack implementations, including CA TCPaccess CS for z/OS, IBM TCP/IP and their run time equivalents.
  - **RHDCD2IP, RHDCD3IP and RHDCD4IP**
    Names reserved for future implementations.

**Note:** As of r17, the plug-in parameter, if specified, is ignored to provide backward compatibility with r16. It is defined through the PLUGIN MODULE clause in the new TCP/IP SYSGEN entity.

LISTENER PTERM Statement Parameters

- **TYPE is LISTENER**
  Indicates that the physical terminal is a TCP/IP listener.

- **listener-port-identifier**
  Specifies the number of the listener port or a service name. If listener-port-identifier is a port number, it must be a positive number between 1 and 65535. If listener-port-identifier is a service name, it is limited to 32 characters and must be the name of a service in the services file with an associated protocol of TCP.

- **backlog**
  The value defines the maximum length for the queue of pending connections TCP/IP allows before disallowing connection requests. Backlog is a positive number between 1 and 1,147,483,647. The default value is 5.
  The value specified for backlog is not necessarily the value accepted by the LISTEN call. Each TCP/IP implementation has a limit of its own. CA IDMS uses the lesser of the implementation’s limit and the value specified for the backlog parameter.

- **stack-ip-name**
  The job name of the TCP/IP stack. The name is limited to 8 characters. Specifying *ALL on a multi-homed system (z/OS only) causes listening to all active TCP/IP stacks. Specifying *DEFAULT causes listening to the default TCP/IP stack.
- **stack-ip-address**
  IP address of the host. The limit of an IP address depends on whether IPv4 or IPv6 is used: the limit in IPv4 is 15 characters; in IPv6, it is 45 characters.

- **stack-host-name**
  Name of the host. The maximum host name length is 64 characters.

- **task-code**
  Name of the task code to start when a connection request arrives.

- **MODE IS USER or SYSTEM**
  Indicates whether the task attached by the listener executes in SYSTEM or USER MODE. To execute in SYSTEM mode, the program must be fully reentrant and be written in either of the following:
  - Assembler using DC calling conventions
  - COBOL or PL/I and compiled with an LE-compliant compiler

- **string**
  A string is passed to the task attached by the generic listener. The string is limited to 80 characters.

- **max-number-conn**
  Defines the maximum number of active connections that can be started from the corresponding listener program, that is, the maximum number of active BULK PTERMs allocated by the specific LISTENER. When the number of connections reaches the value specified for **max-number-conn**, any new connection accepted by the listener program will be rejected. **max-number-conn** is a positive number between 1 and 65535. The default value is OFF, indicating that the maximum number of connections is unlimited.

For information about the options that can be specified for the CA IDMS Server listener, see the following Usage section.

**DDSTCPIP PTERM Parameters**

- **TYPe is DDSTCPIP**
  Defines the target system for the DDS connection.

- **TARget**
  The **target-ip-address** and **target-host-name** parameters are mutually exclusive. You must specify at least one of these parameters in the definition of a DDSTCPIP type PTERM.

- **target-ip-address**
  Specifies the IP address of the target system enclosed in single quotes. The IP address limit depends on whether IPv4 or IPv6 is used: IPv4 is 15 characters; IPv6 is 45 characters.

- **target-host-name**
  Specifies the host name of the target system. The maximum host name length is 64 characters.
- **target-port-identifier**
  Specifies the number of the target port or a service name. If `target-port-identifier` is a port number, it must be a positive number between 1 and 65535. If `target-port-identifier` is a service name, it is limited to 32 characters and must be the name of a service in the services file with an associated protocol of TCP.

- **stack-ip-name**
  Specifies the job name of the TCP/IP stack to use in the local system. The job name is limited to 8 characters. Specify an empty string value (two single-quotes) to remove an IP STACK NAME definition.

- **start-port-number and end-port-number**
  Defines a range of port numbers that are used to BIND the local sockets explicitly. Each time a new connection is established, the first free port from the range is selected and associated (bound) with the corresponding socket. If no free port is found, the request is aborted. The default value is OFF, indicating that the operating system will select a free port from the pool and bind the socket implicitly during the connect processing. `start-port-number` and `end-port-number` are positive numbers between 1 and 65535. `start-port-number` must be lower than or equal to `end-port-number`.

- **idle-interval**
  Defines the time interval a non-permanent connection stays in an idle state after the corresponding DDS request has finished. This allows the same connection to be reused if a new DDS request comes in before the timeout expires. `idle-interval` is a positive number between 0 and 32767. The default value is 0.

- **max-number-conn**
  Defines the maximum number of active connections allowed from the local system. `max-number-conn` is a positive number between 1 and 65535. The default value is OFF, indicating that the maximum number of connections is unlimited.

  **Note:** The maximum number of connections depends on the number of free BULK PTERMs in the SOCKET line on the target (remote) system.

- **perm-conn-number**
  Defines the number of permanent connections that can exist between the host and the target systems. `perm-conn-number` is a positive number between 0 and 65535. The default value is 0, indicating that permanent connections are not needed. In this case, the connections are always established dynamically when a new DDS request arrives.

**BULK PTERM Parameters**

- **TYPE is BULK**
  Specifies that the type of terminal is BULK.

**Notes for LISTENER PTERMS Only**
Parameters `stack-ip-name`, `stack-ip-address`, and `stack-host-name` are mutually exclusive. Specifying any of these parameters only makes sense in a multi-homed environment.

If one of these parameters is specified:

- Listening is restricted to the named TCP/IP stack.
- A CV can be tied to an operating system image.

If none of these parameters are specified, the generic listener processes connection requests from the default TCP/IP stack.

⚠️ **Note:** For more information about TCP/IP and SOCKET programming, see the CA IDMS Callable Services section. For more information about CA IDMS Server, see the CA IDMS Server r5 User section. For more information about the Services File and the Services Resolver, see the CA IDMS System Operations section.

### SOCKET Usage

To use the CA IDMS Server "wire protocol" drivers, define a listener PTERM/LTERM pair for the built-in server program, IDMSJSRV. This PTERM must specify the RHDCNP3J task defined during installation, SYSTEM mode, and the port used by the driver. The default port, 3709, is used by the drivers and registered with the Internet Assigned Number Authority (IANA) for CA IDMS. This can be used if only a single DC/UCF system is used on the host machine. Otherwise, a recommended convention is to append the system number to 37.

⚠️ **Note:** Be advised that all CA IDMS systems that have task-level security turned ON must turn off security for the RHDCNP3J task. The following SECRRT entry example displays how this can be performed. Tailor this entry to meet your site-specific requirements in your CA IDMS SRTT table.

```
#SECRRT  TYPE=OCCURRENCE, RESNAME='RHDCNP3J',
          RESTYPE=TASK,
           SECBY=OFF
```

Additional listener options parameters can be specified as keyword-values pairs in the PARM string on the PTERM definition. The following are the PARM keyword guidelines:

- Options are not case-sensitive.
- Options can appear in any order, separated by commas, with no embedded spaces in the string (leading spaces are ignored).
- If an error occurs, the server writes a message to the DC log and ignores the rest of the options.
The parameters are:

- **ACCt=profile-key-name**
  Profile key name for accounting information. If specified, any accounting information supplied by
  the client as part of the signon is added to the user profile. This allows SQL statements access to
  the accounting information using the PROFILE function. Procedures written in COBOL or CA ADS
  can use the IDMSIN01 GETPROF callable service to access the accounting information.

- **BUFLen=default-length**
  Length of the default buffer used for data received from and sent to the client. The default buffer
  is allocated when the task starts and freed when the task ends. Larger buffers are allocated
  dynamically as needed for individual requests and freed immediately after results are returned.
  The default is 1024 bytes. Specifying a larger buffer can minimize storage allocation requests
  while using more of the storage pool. The client can override this value.

- **BUFMax=max-length**
  Maximum length the client interface can specify for the default buffer.

- **TASK=task-code**
  Specifies the default task code used for signon security, timeouts, and as the next task after a
  pseudo converse. The IDMSJSRV task is defined during installation as a model task for this
  purpose. The client can override this task code.

- **ATTACH=Y**
  Causes IDMSJSRV to end the initial task immediately after processing the signon request. The
  specified task is attached on the next request from the client interface. If not specified (or No),
  IDMSJSRV copies the timeouts from specified task to the current task, which ends when (and if)
  explicitly requested by the client interface.

- **TIMeout=seconds**
  Specifies the maximum timeout the client can request for the external wait or resource timeout
  interval. A value of -1 allows the client interface to specify any value. A value of 0, the default,
  takes the value from the task or system default.

---

**Example SOCKET**

```plaintext
** SAMPLE LINE DEFINITION
ADD LINE TCPIP
  TYPE IS SOCKET.

** WIRE PROTOCOL DRIVER LISTENER
  USING THE CONVENTION
  PORT = 3700 + SYSTEM NUMBER
ADD PTERM TCPJSRV
  TYPE IS LISTENER
  PORT IS 3799
  BACKLOG IS 100
  TASK IS RHDCNP3J
  MODE IS SYSTEM
  PARM IS 'TASK=IDMSJSRV'.

ADD LTERM TCLJSRV
  PTERM IS TCPJSRV.

** USER IMPLEMENTED LISTENER
ADD PTERM TCP_LIS_1
  TYPE IS LISTENER
  TASK IS TSKLIS_1
```

---
MODE IS USER
PORT IS 12345
BACKLOG IS 999
PARM IS 'Parameters for TSKLIS_1'.

ADD LTERM TCPLIS_1
PTERM TCPLIS_1.

** PTERM DEFINITION FOR A REMOTE DDS NODE SYSTEM

ADD PTERM PDDSNOD1
  TYPE IS DSTCPIP
  TARGET NAME IS 'SYSNODE1'
  TARGET PORT IS 15901
  PORT RANGE IS 15950 THROUGH 15999
  IDLE INTERVAL IS 60
  NUMBER OF PERMANENT CONNECTIONS IS 3.

ADD LTERM LDSDSNOD1
  PTERM IS PDDSNOD1.

** PTERM DEFINITION FOR 100 SOCKETS

ADD PTERM TCP0001
  TYPE IS BULK
  REPEAT COUNT IS 99.

ADD LTERM TCL0001
  PTERM IS TCP0001.

SYSOUTL

The following syntax for LINE and PTERM parameters is used to define the following device types:

- SYSOUT devices using QSAM (z/OS systems only)
- Printers that will receive spooled output (z/VM systems only)

⚠️ Note: To implement user exit 21, include at least one SYSOUTL line in the system definition. For more information about user exit 21, see the CA IDMS Installation and Maintenance section -- z/VM and the CA IDMS System Operations section.

SYSOUT Syntax

** LINE Syntax **

```
  TYPE is SYSOUTL
```

```
  DDname is ┌─┐
    ddname └─┘

  linkname ┌─┐
    linkname └─┘
```

** PTERM Syntax **

```
  TYPE is SYSOUTT
```

```
  PAGE LENGTH is ┌─┐
    page-length └─┘

  PAGE WIDTH is ┌─┐
    page-width └─┘
```
SYSOUT Parameters

LINE Statement Parameters

- **TYPe is SYSOUTL**
  Specifies the line type.

- **DDname is**
  Associates the line with an external file name.
  The DDNAME parameter is required to create an executable system.

  - **ddname**
    For z/OS and z/VM systems only, identifies the ddname for the line, as specified in the system startup JCL.

PTERM Statement Parameters

- **TYPe is SYSOUTT**
  Specifies the device type for the physical terminal.

- **PAGe LENgth is page-length**
  Specifies the maximum page size, in text lines, for the physical terminal.
  Page-length must be an integer in the range 1 through 32,767. The default is 60.

- **PAGe WiDth is page-width**
  Specifies the maximum page width, in characters, for the physical terminal.
  Page-width must be an integer in the range 1 through 32,767. The default is 132.

**Note:** The SYSOUTL line can be used in conjunction with the WRITE PRINTER command to submit batch jobs via the z/OS internal reader. For more information, see the section "Writing JCL to a JES2 Internal Reader" in the *Navigational DML Programming section*.

S3270Q

The following syntax for LINE and PTERM parameters is used to define simulated 3270 devices using QSAM.

S3270Q Syntax

LINE Syntax

```
<table>
<thead>
<tr>
<th>TYPE is S3270Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFER size is buffer-size</td>
</tr>
<tr>
<td>INPut ddname is ddname</td>
</tr>
<tr>
<td>filename</td>
</tr>
<tr>
<td>linkname</td>
</tr>
</tbody>
</table>
```
S3270Q Parameters

LINE Statement Parameters

- **TYPe is S3270Q**
  Specifies the line type.

- **BUFfer size is buffer-size**
  Specifies the line I/O page buffer size, in bytes, for the line.
  Buffer-size must be an integer in the range 1 through 32,767.

- **INput ddname is**
  Specifies the external name of the input file for the line, as specified in the system startup JCL.
  The INPUT DDNAME IS parameter is required to create an executable system.

  - **ddname**
    For z/OS and z/VM systems, specifies the ddname of the input file for the line.

  - **filename**
    For z/VSE systems, specifies the filename of the input file for the line.

- **OUTput ddname is**
  Specifies the external name of the output file for the line, as specified in the system startup JCL.
  The OUTPUT DDNAME parameter is required to create an executable system.

  - **ddname**
    For z/OS and z/VM systems, specifies the ddname of the output file for the line.

  - **filename**
    For z/VSE systems, specifies the filename of the output file for the line.

PTERM Statement Parameters

- **TYPe is**
  Specifies the device type for the physical terminal. Valid values are:

  - S3277
  - S3278
  - S3279
• **MODel is *model-number***
  Specifies the model number of the physical terminal. Valid model numbers for each physical terminal type are:

<table>
<thead>
<tr>
<th>Physical Terminal Type</th>
<th>Model Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3277</td>
<td>1 and 2</td>
</tr>
<tr>
<td>S3278</td>
<td>1 through 5</td>
</tr>
<tr>
<td>S3279</td>
<td>2 and 3</td>
</tr>
</tbody>
</table>

The default is 2 for all physical terminal types.

**S3270Q Usage**

**z/OS DD Statements for S3270Q Lines**

For z/OS systems, the DD statement for the input file for an S3270Q line must include the following DCB specifications:

- **LRECL=80**

The DD statement for the output file for an S3270Q line must include the following DCB specifications:

- **LRECL=133**

**Example 4**

The following LINE, PTERM, and LTERM statements define a simulated 3270 line and terminal:

```plaintext
ADD LINE S3270Q1
  TYPE IS S3270Q
  INPUT DDNAME IS SIMIN1
  OUTPUT DDNAME IS SIMOUT1.

ADD PTERM PS3270Q1
  TYPE IS S3277
  MODEL IS 2.
ADD LTERM LS3270Q1
  PTERM IS PS3270Q1.
```

**TCAMLIN**

The following syntax for LINE and PTERM parameters is used to define 3270 or 3280 devices using TCAM.

**TCAMLIN Syntax**

**LINE Syntax**
TCAMLIN Parameters

LINE Statement Parameters

- **TYPE is TCAMLIN**
  Specifies the line type.

- **BUFfer size is buffer-size**
  Specifies the line I/O page buffer size, in bytes, for the line. 
  *Buffer-size* must be an integer in the range 1 through 32,767.

- **CONnect**
  Specifies the line is treated as a connect-type line.
  *CONNECT* is the default when you specify neither *CONNECT* nor *NOCONNECT* in the LINE statement.

- **NOConnect**
  Specifies the line is not treated as a connect-type line.

- **INput ddname is**
  Specifies the external name of the input file for the line, as specified in the system startup JCL.

  - **ddname**
    For z/OS systems, specifies the ddname of the input file for the line.

  - **filename**
    For z/VSE systems, specifies the filename of the input file for the line.
OUTput ddname is
Specifies the external name of the output file for the line, as specified in the system startup JCL.

- **ddname**
  For z/OS systems, specifies the ddname of the output file for the line.

- **filename**
  For z/VSE systems, specifies the filename of the output file for the line.

**PTERM Statement Parameters**

- **TYPe is TCAMTRM**
  Specifies the device type for the physical terminal.

- **FORmfeed**
  Specifies the physical terminal has formfeed capabilities.

- **NOFormfeed**
  Specifies the physical terminal does not have formfeed capabilities.
  NOFORMFEED is the default when you specify neither FORMFEED nor NOFORMFEED in the PTERM statement.

- **MODel is model-number**
  Specifies the model number of the physical terminal.
  *Model-number* must be an integer in the range 1 through 5. The default is 2.

- **NAME is tcam-terminal-id**
  Reserves the physical terminal element (PTE) for the TCAM terminal identified by *tcam-terminal-id*.
  PTEs to be used by 3280-type printers should always be defined with a TCAM terminal identifier.

- **NOName**
  Defines a generic PTE that can be used by any TCAM terminal not explicitly associated with another PTE.
  NONAME is the default when you specify neither NAME nor NONAME in the PTERM statement.

---

**TCAMLIN Usage**

### z/OS DD Statements for TCAMLIN Lines

For z/OS systems, the DD statement for the input file for a TCAMLIN line must include the following DCB specifications:

LRECL=80

The DD statement for the output file for a TCAMLIN line must include the following DCB specifications:

LRECL=133
The UCFLINE statement is used to define the UCF line driver to support UCF terminal communications.

The following syntax for LINE and PTERM parameters is used to define a UCF line. The DC/UCF system definition can contain only one UCF line definition.

**UCFLINE Syntax**

**LINE Syntax**

```
TYPe is UCFLINE
MODule is system-table-module-name
```

**PTERM Syntax**

```
TYPe is UCFTERM
NAME is front-end-terminal-id
NOName
```

**UCFLINE Parameters**

**LINE Statement Parameters**

- **TYPe is UCFLINE**
  Specifies the line type.

- **MODule is system-table-module-name**
  Specifies the name of the UCF system table module created by the assembly and link edit of the #FESTDEF and #FESTENT macros.
  The MODULE parameter is required to create an executable system.

**PTERM Statement Parameters**

- **TYPe is**
  Specifies the device type for the physical terminal.

  - **UCFTERM**
    Specifies a 3270-type for the physical terminal.

- **NAME is front-end-terminal-id**
  Specifies, for UCFTERM PTERMS only, the physical terminal is connected with only the specified front-end terminal.
  *Front-end-terminal-id* must be the identifier of a UCF front-end terminal defined in the UCF front-end table.
- **NOName**
  Defines the physical terminal as a generic terminal to which any UCF front-end terminal can be connected.
  NONAME is the default when you specify neither NAME nor NONAME in the PTERM statement.

**Note:** For more information about creating the UCF front-end system, see the *CA IDMS System Operations section*. For more information about the UCF front-end table, see the *CA IDMS System Operations section*. For more information about batch external request units, see External User Sessions (see page 63).

### Example UCFLINE

#### Reserving Physical Terminals for UCF Front-End Terminals

The following LINE, PTERM, and LTERM statements define a UCF line with two physical terminals, each associated with a UCF front-end terminal:

```
ADD LINE UCF
  TYPE IS UCFLINE
  MODULE RHDCFSTB.

ADD PTERM UCFPT001
  TYPE IS UCFTERM
  NAME = TO15.
ADD LTERM UCFLT001.
ADD PTERM UCF002
  TYPE IS UCFTERM
  NAME = TO32.
ADD LTERM UCFLT002.
ADD PTERM UPRT001
```

#### Defining Prototype Terminals and a Printer Terminal

The following statements define a UCF line with three prototype physical terminals (that is, terminals available to any front-end terminal) and one printer terminal:

```
ADD LINE UCF
  TYPE IS UCFLINE
  MODULE RHDCFSTB.

ADD PTERM UCFPT001
  TYPE IS UCFTERM
  NONAME.
ADD LTERM UCFLT001.
ADD PTERM UCFPT002
  TYPE IS UCFTERM
  NONAME.
ADD LTERM UCFLT002.
ADD PTERM UCFPT003
  TYPE IS UCFTERM
  NONAME.
ADD LTERM UCFLT003.
```
VTAMLIN

The following syntax for LINE and PTERM parameters is used to define 3270 or asynchronous VTAM devices.

⚠️ Note: For more information about using VTAM with DC/UCF, see VTAM Considerations (see page 359).

VTAMLIN Syntax

LINE Syntax

- **TYPE** is VTAMLIN
- Application identification is *vtam-application-id*
- **CONnect** / NOConnect
- **PASsword** is *vtam-password*
- **GENeric RESource** is *generic-vtam-resource*
- **PERMREADBUF** / NOPERMREADBUF
- **RPL count** is *rpl-count*

PTERM Syntax

- **TYPE** is V3277 - V3289
- **ACQuire** / NOAcquire
- **FORmfeed** / NOFormfeed
- **HOLd** / NOHold
- **MODel** is 2
VTAMLIN Parameters

LINE Statement Parameters

- **TYPE is VTAMLIN**
  Specifies the line type.

- **APPLICATION identification is vtam-application-id**
  Specifies the VTAM application identifier used by the DC/UCF system at runtime to sign on to VTAM.
  `vtam-application-id` must match the system identifier specified for the DC/UCF system in the VTAM system definition.
  The `APPLICATION IDENTIFICATION` parameter is required to create an executable system.

- **CONnect**
  Specifies the line is treated as a connect-type line.
  `CONNECT` is the default when you specify neither `CONNECT` nor `NOCONNECT` in the `LINE` statement.

- **NOConnect**
  Specifies the line is not treated as a connect-type line.

- **PASSWORD is vtam-password**
  Specifies the password used by the system to sign on to VTAM.
  `vtam-password` must be the password specified for the DC/UCF application in the VTAM system definition.

- **GENeric RESource is generic-vtam-resource**
  Specifies the name of a VTAM Generic Resource name that allows access to the CA IDMS CV. The value specified in the line definition is in addition to the value specified in the “application identification is”.
  The Application ID is the VTAM node while the Generic Resource name allows access across SYSPLEX LPARs.
  Either value can be used to connect to the VTAM lines in an IDMS system. The Generic VTAM resource name is not defined to VTAM via standard “VTAMLST” entries. Additionally, one or more IDMS systems can specify the same Generic VTAM resource value. The VTAM Resource name is implemented in the SYSGEN for each VTAM line.
  For more information see [Administrating CA IDMS System Operations](https://docops.ca.com/display/IDMS19/Administrating+CA+IDMS+System+Operations).

- **Limits**: The Generic Resource name has a maximum length of 8.

- **PERMREADBUF**
  Specifies that a read buffer is permanently allocated for the duration of a session. The buffer is allocated at connection time and released at session termination.
- **NOPERMREADBUF**
  Specifies that a read buffer is dynamically allocated only for the duration of an input operation. NOPERMREADBUF is the default when neither PERMREADBUF nor NOPERMREADBUF is specified in the LINE statement.

- **RPL count is rpl-count**
  Specifies the number of entries in the VTAM request parameter list (RPL). This number represents the maximum number of write I/O requests that can be handled concurrently by the VTAM line.  
  `Rpl-count` must be an integer in the range 1 through 32,767. The default is 5. Typically, `rpl-count` should be 20 percent of the number of physical terminals in the line, plus the number of printers. An insufficient RPL count value will result in degraded response time.

**PTERM Statement Parameters**

- **TYPe is**
  Specifies the device type for the physical terminal. Valid values are:
  - V3277
  - V3278
  - V3279
  - V3284
  - V3286
  - V3287
  - V3288
  - V3289

  This parameter is for documentation only. The actual device type is determined from information provided by VTAM when the user logs on.

- **ACQuire**
  Specifies the physical terminal is connected to the system automatically. The user does not have to log on through VTAM to access the system. 
  If you specify ACQUIRE, you must also specify the NAME parameter.

- **NOAcquire**
  Specifies the physical terminal is not connected to the system automatically. The user must log on through VTAM, supplying the DC/UCF system identifier. NOACQUIRE is the default when you specify neither ACQUIRE nor NOACQUIRE in the PTERM statement.

- **FORmfeed**
  Specifies the physical terminal has formfeed capabilities.

*Note: FORMFEED is invalid for physical terminal types V3277, V3278, and V3279.*
- **NOFormfeed**
  Specifies the physical terminal does not have formfeed capabilities.
  NOFORMFEED is the default when you specify neither FORMFEED nor NOFORMFEED in the PTERM statement.

- **HOLD**
  Specifies the device is not released when VTAM notifies DC/UCF that another system is requesting a session with this device.

- **NOHold**
  Specifies the device is released when VTAM notifies DC/UCF that another system is requesting a session with this device. If the request specifies a printer, DC/UCF will release the printer when all currently queued reports have been printed.
  NOHOLD is the default when you specify neither HOLD nor NOHOLD in the PTERM statement.

- **MODEL is model-number**
  Specifies the model number of the physical terminal. Valid model numbers for each physical terminal type are:

<table>
<thead>
<tr>
<th>Physical terminal type</th>
<th>Model number</th>
</tr>
</thead>
<tbody>
<tr>
<td>V3277</td>
<td>1 and 2</td>
</tr>
<tr>
<td>V3278</td>
<td>1 through 5</td>
</tr>
<tr>
<td>V3279</td>
<td>2 and 3</td>
</tr>
<tr>
<td>V3284</td>
<td>1 and 2</td>
</tr>
<tr>
<td>V3286</td>
<td>1 and 2</td>
</tr>
<tr>
<td>V3287</td>
<td>1 and 2</td>
</tr>
<tr>
<td>V3288</td>
<td>2</td>
</tr>
<tr>
<td>V3289</td>
<td>1 and 2</td>
</tr>
</tbody>
</table>

The default is 2 for all physical terminal types.

- **NAME is vtam-minor-node-name**
  Reserves the physical terminal element for the specified VTAM minor node.
  Vtam-minor-node-name must be the name of a VTAM minor node (logical unit) specified in the VTAM definition.
  You must specify the NAME parameter for printers.

- **NONAME**
  Defines the physical terminal as a generic terminal to which any VTAM minor node can be connected.
  NONAME is the default when you specify neither NAME nor NONAME in the PTERM statement.

- **RElease**
  For printer terminals only, specifies the printer is automatically released after each report is printed. If other reports are queued to the printer, DC/UCF will subsequently reacquire the printer.
• **NOReslease**  
  For printer terminals only, specifies the printer is not released automatically after a report is printed. The printer will be held by the DC/UCF system until notified by VTAM that another system is requesting the device. Subsequent action by DC/UCF depends on the HOLD/NOHOLD specification. NORERELEASE is the default when you specify neither RELEASE nor NORERELEASE in the PTERM statement.

![Note: For more information about sample DC/UCF configuration using VTAM, see DC/UCF System Generation Statements (see page 361). For more information about VTAMLST entries, see VTAMLST Entries (see page 364).]

### VTAMLIN Usage

#### Calculating the Size of a Permanent Read Buffer

The size of a permanent read buffer is calculated on the basis of terminal dimensions returned by VTAM in the bind area. Buffers for dynamic logmode usage are eventually resized depending on the terminal dimensions returned by the read partition operation.

#### Removing the Generic VTAM Resource

The following Syntax statements REMOVE the Generic VTAM resource from the LINE statement:

- MOD LINE VTAM01
  - GENERIC RESOURCE ID IS ““.
- MOD LINE VTAM01
  - GENERIC RESOURCE ID IS ‘OFF’.
- MOD LINE VTAM01
  - GENERIC RESOURCE ID IS ‘NONE’.

### Example VTAMLIN

The following LINE, PTERM, and LTERM statements define a VTAM line with three 3270-type terminals:

ADD LINE VTAM90  
  TYPE IS VTAMLIN  
  APPLICATION ID IS IDMSDC.

ADD PTERM PTVTM001
IN LINE VTAM90
  TYPE IS V3277.
ADD LTERM LTVTM001
  PTERM IS PTVTM001.
ADD PTERM PTVTM002
  IN LINE VTAM90
  TYPE IS V3284
  NAME = CUL08
  ACQUIRE.
ADD LTERM LTVTM002
  PTERM IS PTVTM002.
ADD PTERM PTVTM003
  IN LINE VTAM90
  TYPE IS V3277
  NAME = CUL18.
ADD LTERM LTVTM003
  PTERM IS PTVTM003.

The following LINE statement defines using Generic VTAM resources. MODIFY can be used for existing lines.

ADD LINE VTAM01
  ENABLED
  COMPACT
  PROTOCOL IS EXPRESP
  TYPE IS VTAMLIN
  APPLICATION ID IS U31ISY90
  GENERIC RESOURCE IS IDMSGVR1
  CONNECT
  PERMREADBUF
  RPL COUNT IS 5

VTAMLU

The following syntax for LINE and PTERM parameters is used to define SNA/VTAM logical unit (LU) types 0 through 4 and 6.2. For a discussion of the DC SNA/VTAM line driver, see SNA and LU 6.2 Considerations (see page 399).

Note: You should define only one VTAMLU line, regardless of the number of LUs to be supported. You must define a single physical terminal for each logical unit. No pool of physical terminals is used.

VTAMLU Syntax

LINE Syntax

  ➤ TYPE is VTAMLU
  ➤ APPLICATION identification is vtam-application-id
  ➤ CONnect
  ➤ NOConnect
  ➤ PASsword is vtam-password
VTAMLU Parameters

LINE Statement Parameters

- **TYPE is VTAMLU**
  Specifies the line type.

- **APPLICATION identification is vtam-application-id**
  Specifies the VTAM application identifier used by DC/UCF at runtime to open a VTAM access method control block (ACB) and prepare for communication with logical units. `vtam-application-id` must match the system identifier specified for the DC/UCF system in the VTAM system definition. The APPLICATION IDENTIFICATION parameter is required to create an executable system.
• **CONnect**
  Specifies the line is treated as a connect-type line.
  CONNECT is the default when you specify neither CONNECT nor NOCONNECT in the LINE statement.

• **NOConnect**
  Specifies the line is not treated as a connect-type line.

• **PASSword is vtam-password**
  Specifies the password used by DC/UCF to open a VTAM ACB.
  *Vtam-password* must be the password specified for the DC/UCF application in the VTAM system definition.

• **RPL count is rpl-count**
  Specifies the number of request parameter list (RPL) storage areas to be used by the SNA/VTAM line driver to support all logged-on logical units. This number represents the maximum number of write I/O requests that can be handled concurrently by the SNA/VTAM line.
  *Rpl-count* must be an integer in the range 1 through 32,767. The default is 5.
  Typically, *rpl-count* should be lower than the expected number of logical units in the line. An insufficient RPL count value will result in degraded response time.

**PTERM Statement Parameters**

• **TYPe is**
  Specifies the device type for the physical terminal. Valid values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LU</td>
<td>General category; any logical unit, including non-pipeline LUs</td>
</tr>
<tr>
<td>LU62</td>
<td>LU 6.2 support</td>
</tr>
<tr>
<td>3600LU</td>
<td>3600 full-function LU support</td>
</tr>
<tr>
<td>3600PL</td>
<td>3600 pipeline LU support</td>
</tr>
<tr>
<td>3614</td>
<td>3614 devices</td>
</tr>
</tbody>
</table>

*Note:* For more information about SNA and LU 6.2 system generation considerations, see SNA and LU 6.2 Considerations (see page 399).

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

• **ACQuire**
  Specifies the physical terminal is connected to the system automatically. The user does not have to log on through VTAM to access the system.
NOAcquire
Specifies the physical terminal is not connected to the system automatically. The user must log on through VTAM, supplying the DC/UCF system identifier.
NOACQUIRE is the default when you specify neither ACQUIRE nor NOACQUIRE in the PTERM statement.

CONtention is
Specifies whether a session bound using the physical terminal will be a bracket contention winner or loser.

Note: For LU 6.2 sessions, the multiple session service manager negotiates the numbers of contention winners and losers to be allowed with the remote LU. The CONTENTION parameter specification is used only for the initial count.

HOLd
Specifies the physical terminal is not released when VTAM sends notification that another system is requesting a session with the logical unit.

NOHold
Specifies the physical terminal is released when VTAM sends notification that another system is requesting a session with the logical unit only if a conversation is not currently allocated to the session. If a conversation is allocated, the physical terminal is released after the conversation terminates.
NOHOLD is the default when you specify neither HOLD nor NOHOLD in the PTERM statement.

INFmh is
Specifies whether an input function management header (FMH) is expected as a prefix to a request/response unit (RU).

- YES
  Indicates that an FMH is expected. The FMH is not removed.

- NO
  Indicates the FMH is not expected. The FMH is automatically removed.
  NO is the default when you omit the INFMH parameter from the PTERM statement.

LIMit on input is input-limit-size
Specifies the maximum amount of data, in bytes, the DC/UCF system is to buffer before the user requests that data.
Input-limit-size must be an integer in the range 0 through 32,767. The default, 0, directs the DC/UCF system not to limit the amount of data buffered.

MODEent is vtam-modeent-name
Identifies the mode table entry that contains the bind parameters to be used when initiating a session with the physical terminal.
Vtam-modeent-name must match the name specified by the LOGMODE parameter of the appropriate VTAM mode table entry.
Note: For real APPC support, vtam-modeent-name must match the name specified in the MODEENT parameter of a PTERM statement used to define a physical terminal for APPC.

You must name a mode table entry for logical unit type 6.2. For more information about real APPC support, see Real APPC Support Considerations (see page 398).

- **NOModeent**
  Specifies that VTAM is to use the default mode table entry for the logical unit. NOMODEENT is the default when you specify neither MODEENT nor NOMODEENT in the PTERM statement.

- **NAME is vtam-minor-node-name**
  Specifies the name of the VTAM node that will be associated with the terminal. Vtam-minor-node-name must be the name of a VTAM minor node (logical unit) specified in the VTAM definition. This parameter is required to create an executable system.

- **OUTfmh is**
  Specifies a default function management header (FMH) to be sent with all outbound transmissions for this logical unit.
    - **'hex-fm-header-q'**
      Identifies the default FMH to be sent with all outbound transmissions. Hex-fm-header-q is a 4- through 8-character hexadecimal literal enclosed in site-standard quotation marks that is included in every transmission.
    - **NO**
      Indicates that no default FMH exists. NO is the default when you omit the OUTFMH parameter from the PTERM statement.

- **RELease**
  Specifies the physical terminal is disconnected automatically whenever a conversation on the session terminates.

- **NORElease**
  Specifies the physical terminal is not disconnected automatically whenever a conversation on the session terminates; that is, the session will remain available for subsequent conversations. NORELEASE is recommended for greater efficiency. NORELEASE is the default when you specify neither RELEASE nor NORELEASE in the PTERM statement. NORELEASE should be specified for logical unit type 6.2.

- **SYNclevel is**
  For LU 6.2 sessions only, specifies the maximum level of synchronization to be allowed for a session that uses the physical terminal.
    - **CONFirm**
      Directs the DC/UCF system to accept a bind with a synchronization level of CONFIRM or NONE.
- **SYNcpoint**
  Directs the DC/UCF system to accept a bind with a synchronization level of SYNCPOINT, CONFIRM, or NONE.

- **NONE**
  Directs the DC/UCF system *not* to accept a bind with a synchronization level of SYNCPOINT or CONFIRM.

- **OFF**
  Directs the DC/UCF system to accept the synchronization level specified in the bind parameters.
  OFF is the default when you omit the SYNCELEVEL parameter from the PTERM statement.

**VTAM LU Usage**

**Real APPC Support**

DC uses the VTAM LU line type to support real APPC.

To provide support for real APPC in a DC/UCF system:

- Add the appropriate physical and logical terminal definitions to the VTAM LU line definition in the DC/UCF system definition.
- Have your VTAM system programmer code a VTAM mode table entry to define the SNA protocols (that is, the bind parameters) to be used for sessions with other type 6.2 logical units.

The PTERM statement used to define a physical terminal for real APPC must specify:

- TYPE IS LU
- NOACQUIRE (default)
- NAME IS vtam-node-name
- INFMH IS NO (default)
- OUTFMH IS NO (default)
- CONTENTION IS WINNER
- NOHOLD (default)
- LIMIT ON INPUT IS 0 (default)
- MODEENT IS vtam-modeent-name
- NORELEASE (default)
- SYNCELEVEL IS OFF (default)
Example VTAMLU

Sample Statements for a VTAM Line with Three Terminals

The following LINE, PTERM, and LTERM statements define a VTAM line with three terminals:

```
ADD LINE VTAM91
   TYPE IS VTAMLU
   APPLICATION ID IS IDMSDC.

ADD PTERM PTVTM001
   IN LINE VTAM91
   TYPE IS LU.
ADD LTERM LTVTM001
   PTERM IS PTVTM001.

ADD PTERM PTVTM002
   IN LINE VTAM91
   TYPE IS 3600LU
   NAME IS CUL09
   ACQUIRE.
ADD LTERM LTVTM002
   PTERM IS PTVTM002.

ADD PTERM PTVTM003
   IN LINE VTAM91
   TYPE IS 3614
   NAME = CUL18.
ADD LTERM LTVTM003
   PTERM IS PTVTM003.
```

Sample Statements for Real APPC Support

The following system generation statements define two terminals to be used for real APPC:

```
ADD LINE SNALU1
   TYPE IS VTAMLU
   APPLICATION IDENTIFICATION IS IDMSSNA
   ENABLED.

ADD PTERM LU001
   TYPE IS LU
   NAME IS LU620001
   CONTENTION IS WINNER
   MODEENT IS SNAAPPC1
   ENABLED.
ADD LTERM LU001
   ENABLED.
ADD PTERM LU002
   TYPE IS LU
   NAME IS LU620001
   CONTENTION IS WINNER
   MODEENT IS SNAAPPC1
   ENABLED.
ADD LTERM LU002
   ENABLED.
```
Teleprocessing Network Example

The following statements, define a teleprocessing network consisting of four lines: an operator’s console, a local 3270 BTAM line, a local 3280 BTAM printer line, and a SYSIN/SYSOUT line. The following example lists the required LINE statements and their associated PTERM and LTERM statements:

ADD LINE CONSOLE
    TYPE IS CONSOLE.

ADD PTERM OPERATOR
    TYPE IS OPERATOR.
ADD LTERM CONSOLE
    PRIORITY IS 220.

ADD LINE BTAMLINE
    TYPE IS L3270B
    DDNAME IS BTAM3271.

ADD PTERM BTAMPT1
    TYPE IS L3277
    MODEL IS 2.
ADD LTERM BTAML1.

ADD PTERM BTAMPT2
    TYPE IS L3277
    MODEL IS 2.
ADD LTERM BTAML2.

ADD PTERM BTAMPT3
    TYPE IS L3277
    MODEL IS 2.
ADD LTERM BTAML3.

ADD LINE PRTLINE
    TYPE IS L3280B
    DDNAME IS BTAM3280.

ADD PTERM PRTPT1
    TYPE IS L3287
    FORMFEED.
ADD LTERM PRTL1
    PRINTER CLASS (2,11,13,42).
ADD LINE IOLINE
    TYPE IS INOUTL
    INPUT DDNAME IS SYSINA
    OUTPUT DDNAME IS SYSOUTA.

ADD PTERM IOPT1
    TYPE IS INOUTT.
ADD LTERM IOLT1.

ADD PTERM IOPT2
    TYPE IS INOUTT.
ADD LTERM IOLT2.
System Generation Data Dictionary Structure

This section outlines the data dictionary structure created and maintained by the system generation compiler.

System Generation Dictionary Records

The following table lists the data dictionary records associated with each system generation statement. The statements are listed alphabetically.

<table>
<thead>
<tr>
<th>System Generation Statement</th>
<th>Data Dictionary Record Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADSO</td>
<td>CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
</tr>
<tr>
<td>AUTOTASK</td>
<td>CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>PROG-051</td>
<td>Prototype program definition; record name is 0DEFAULT, where 0 is hexadecimal 00</td>
</tr>
<tr>
<td>DESTINATION</td>
<td>DEST-028</td>
<td>Destination source record</td>
</tr>
<tr>
<td></td>
<td>SENDLST-021</td>
<td>Logical extension of the DEST-028 record; source record defining the users, logical terminals, or printers that constitute the destination</td>
</tr>
<tr>
<td></td>
<td>DESTLST-027</td>
<td>Destination object record</td>
</tr>
<tr>
<td></td>
<td>USERDST-150</td>
<td>Logical extension of the DESTLST-027 record for destinations composed of users</td>
</tr>
<tr>
<td></td>
<td>DESTLTRM-117</td>
<td>Logical extension of the DESTLST-027 record for destinations composed of logical terminals or printers</td>
</tr>
<tr>
<td>IDD</td>
<td>CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
</tr>
<tr>
<td>KEYS</td>
<td>CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
</tr>
<tr>
<td>LINE</td>
<td>LINE-109</td>
<td>Line source record</td>
</tr>
<tr>
<td></td>
<td>LINELST-103</td>
<td>Line object record</td>
</tr>
<tr>
<td>LOADLIST</td>
<td>CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
</tr>
<tr>
<td>LTERM</td>
<td>LTRM-106</td>
<td>Logical terminal source record</td>
</tr>
<tr>
<td>System Generation Statement</td>
<td>Data Dictionary Record Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>LTRMLST-105</td>
<td>Logical terminal object record</td>
<td></td>
</tr>
<tr>
<td>MAPTYPE CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
<td></td>
</tr>
<tr>
<td>NODE CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
<td></td>
</tr>
<tr>
<td>OLM CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
<td></td>
</tr>
<tr>
<td>OLQ CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
<td></td>
</tr>
<tr>
<td>PROGRAM PROG-051</td>
<td>Program source record</td>
<td></td>
</tr>
<tr>
<td>PROGRAM PROGLST-049</td>
<td>Program object record</td>
<td></td>
</tr>
<tr>
<td>PTERM PTRM-074</td>
<td>Physical terminal source record</td>
<td></td>
</tr>
<tr>
<td>PTERM PTRMLST-104</td>
<td>Physical terminal object record</td>
<td></td>
</tr>
<tr>
<td>QUEUE QUEUE-030</td>
<td>Queue source record</td>
<td></td>
</tr>
<tr>
<td>QUEUE QUEUELST-029</td>
<td>Queue object record</td>
<td></td>
</tr>
<tr>
<td>RESOURCE TABLE CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
<td></td>
</tr>
<tr>
<td>RUNUNITS CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
<td></td>
</tr>
<tr>
<td>SQL CACHE CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
<td></td>
</tr>
<tr>
<td>STORAGE POOL CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
<td></td>
</tr>
<tr>
<td>SYSTEM SYS-041</td>
<td>System source record; all occurrences are named DCSYSTEM</td>
<td></td>
</tr>
<tr>
<td>SYSTEM SYSMO-170</td>
<td>System object record</td>
<td></td>
</tr>
<tr>
<td>TASK TASK-025</td>
<td>Task source record</td>
<td></td>
</tr>
<tr>
<td>TASK TASKLST-023</td>
<td>Task object record</td>
<td></td>
</tr>
<tr>
<td>TCP/IP CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object. One different sub-type record for each occurrence of an INCLUDE STACK, EXCLUDE STACK or EXCEPT clause in the TCP/IP entity statement.</td>
<td></td>
</tr>
<tr>
<td>System Generation Statement</td>
<td>Data Dictionary Record Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>XA STORAGE POOL</td>
<td>CVGDEFS-142</td>
<td>Logical extension of the SYS-041 record type; one record type for both source and object</td>
</tr>
</tbody>
</table>

**Note:** For more information about the system generation data dictionary structure, see the *CA IDMS Dictionary Structure Reference section*.

**System Programs and Tasks**

**CA IDMS System Programs**

<table>
<thead>
<tr>
<th>Product</th>
<th>Source Member</th>
<th>IDD Module Name(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA ADS</td>
<td>DxODADSA</td>
<td>ADSA-COMPILER</td>
</tr>
<tr>
<td></td>
<td>DxODADSC</td>
<td>ADSA-COMPILER-DYN 1</td>
</tr>
<tr>
<td></td>
<td>DxODADSO</td>
<td>ADSC-COMPILER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADSC-COMPILER-DYN 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADSO-DYN 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADSO-STATEMENT</td>
</tr>
<tr>
<td>CA ADS OPTION</td>
<td>DxASFIDB</td>
<td>ASFIDB</td>
</tr>
<tr>
<td></td>
<td>DxODASF</td>
<td>ASFIDB-DYN 1</td>
</tr>
<tr>
<td></td>
<td>DxODMAPB</td>
<td>ASF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASF-DYN 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAPB-COMPILER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAPB-COMPILER-DYN 1</td>
</tr>
<tr>
<td>CA ADS Batch</td>
<td>DxODMAPB</td>
<td>MAPB-COMPILER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAPB-COMPILER-DYN 1</td>
</tr>
<tr>
<td>CA ICMS</td>
<td>DxASFIDB</td>
<td>ASFIDB</td>
</tr>
<tr>
<td></td>
<td>DxODIDB</td>
<td>ASFIDB-DYN 1</td>
</tr>
<tr>
<td>Product</td>
<td>Source Member</td>
<td>IDD Module Name(s)</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>CA IDMS</td>
<td>DxODDB</td>
<td>IDMS-DB</td>
</tr>
<tr>
<td></td>
<td>DxODCVM</td>
<td>CV-MONITOR</td>
</tr>
<tr>
<td></td>
<td>DxODFACM</td>
<td>IDMS-FACOM (MSP CLIENTS ONLY)</td>
</tr>
<tr>
<td></td>
<td>DxODIDDO</td>
<td>ONLINE-IDD</td>
</tr>
<tr>
<td></td>
<td>DxODMAPC</td>
<td>ONLINE-IDD-DYN 1</td>
</tr>
<tr>
<td></td>
<td>DxODOCF</td>
<td>MAPC-COMPILER</td>
</tr>
<tr>
<td></td>
<td>DxODOLM</td>
<td>ONLINE-COMMAND-FACILITY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ONLINE-MAPPING-STATEMENT</td>
</tr>
<tr>
<td>CA ADS APPC</td>
<td>DxODDNSS</td>
<td>SEND-RECEIVE</td>
</tr>
<tr>
<td>CA IDMS DBCS Option</td>
<td>DxODDBCS</td>
<td>DBCS-OPTION</td>
</tr>
<tr>
<td>CA IDMS Performance Monitor</td>
<td>DxODPERF</td>
<td>PERFORMANCE-MONITOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORMANCE-MONITOR-DYN 1</td>
</tr>
<tr>
<td>CA IDMS Presspack</td>
<td>DxODPRES</td>
<td>PRESSPACK</td>
</tr>
<tr>
<td>CA IDMS Server</td>
<td>DxODSERV</td>
<td>IDMS-SERVER</td>
</tr>
<tr>
<td>CA IDMS vsam Transparency</td>
<td>DxODEVSM</td>
<td>ESCAPE-VSAM</td>
</tr>
<tr>
<td>CA OLQ</td>
<td>DxODOLQ</td>
<td>ONLINE-QUERY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ONLINE-QUERY-DYN 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ONLINE-QUERY-STATEMENT</td>
</tr>
</tbody>
</table>

**Note:** Two sets of source modules for the system programs are supplied with CA IDMS Release 15.0. One set of modules specifies the system programs are installed with storage protect on when they are defined to the system; the other set specifies the system programs are installed with storage protect off. The second character of the module name designates the storage protect option, *x* is either:

- **L** -- the programs are installed with storage protect on
- **N** -- the programs are installed with storage protect off

For your operating system for the source module's location and additional information about this installation option, see the *CA IDMS Installation section*. 

---

26-Dec-2017 356/432
1  -- Some members may contain definitions for programs that are eligible to use null PDEs and therefore are not pre-defined to the system. These members are designated by the '-DYN' suffix and can be added to the system if your site requires all programs be defined.

### CA IDMS Tools System Programs

<table>
<thead>
<tr>
<th>Product</th>
<th>Source Member</th>
<th>IDD Module Name(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA ADS Alive</td>
<td>USGSYSGN</td>
<td>ADS-ALIVE</td>
</tr>
<tr>
<td>CA ADS Trace</td>
<td>UATSYSxN</td>
<td>ADS-TRACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADS-TRACE-DYN 1</td>
</tr>
<tr>
<td>CA IDMS Extractor</td>
<td>USVSYxN</td>
<td>DBX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBX-DYN 1</td>
</tr>
<tr>
<td>CA IDMS/DC Sort</td>
<td>TPSSYSGN</td>
<td>DC-SORT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DC-SORT-DYN 1</td>
</tr>
<tr>
<td>CA IDMS Dictionary Migrator Assistant</td>
<td>XDMSYSxN</td>
<td>DMA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DMA-DYN 1</td>
</tr>
<tr>
<td>CA IDMS DME</td>
<td>USESYxN</td>
<td>DME</td>
</tr>
<tr>
<td>CA IDMS DQF</td>
<td>DADSYSGN</td>
<td>DQF-DYN 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DQF</td>
</tr>
<tr>
<td>CA IDMS DMLO</td>
<td>USDSYSxN</td>
<td>DMLO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DMLO-DYN 1</td>
</tr>
<tr>
<td>CA IDMS Enforcer</td>
<td>ESXSYxN</td>
<td>ENFORCER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENFORCER-DYN 1</td>
</tr>
<tr>
<td>CA IDMS Masterkey</td>
<td>SSKSYxN</td>
<td>MASTERKEY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MASTERKEY-DYN 1</td>
</tr>
<tr>
<td>CA IDMS Online Log Display</td>
<td>USKSYSGN</td>
<td>LOG-DISPLAY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOG-DISPLAY-DYN 1</td>
</tr>
<tr>
<td>CA IDMS SASO</td>
<td>ESSSYxN</td>
<td>SASO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SASO-DYN 1</td>
</tr>
<tr>
<td>CA IDMS Task Analyzer</td>
<td>USFSYSGN</td>
<td>TASKA</td>
</tr>
<tr>
<td>GENERAL EDITOR</td>
<td>USXSYxN</td>
<td>GEN-EDITOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GEN-EDITOR-DYN 1</td>
</tr>
</tbody>
</table>
### CA IDMS - 19.0

<table>
<thead>
<tr>
<th>Product</th>
<th>Source Member</th>
<th>IDD Module Name(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERAL IDMS</td>
<td>GSISYSxN</td>
<td>GEN-IDMS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GEN-IDMS-DYN 1</td>
</tr>
<tr>
<td>GENERAL MAPPER</td>
<td>GSMSYSxN</td>
<td>GEN-MAPPER</td>
</tr>
<tr>
<td>GENERAL SERVICES</td>
<td>GSSSYSGN</td>
<td>GEN-SERVICES</td>
</tr>
<tr>
<td>GENERAL SORT</td>
<td>TPRSYSGN</td>
<td>GEN-SORT</td>
</tr>
</tbody>
</table>

**Note:** Two sets of source modules for the system programs are supplied with CA IDMS Tools Release 15.0. One set of modules specifies the system programs are installed with storage protect on when they are defined to the system; the other set specifies the system programs are installed with storage protect off. The seventh character of the module name designates the storage protect option, x is either:

- **G** -- the programs are installed with storage protect on
- **N** -- the programs are installed with storage protect off

For your operating system for the source module's location and additional information about the installation options, see the *CA IDMS Installing section*.

1 -- Some members may contain definitions for programs that are eligible to use null PDEs and therefore are not pre-defined to the system. These members are designated by the 'DYN' suffix and can be added to the system if your site requires all programs be defined.

## CA Endevor/DB System Programs

<table>
<thead>
<tr>
<th>Product</th>
<th>Source Member</th>
<th>IDD Module Name(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA Endevor/DB</td>
<td>NxVRTASK</td>
<td>CA-ENDEVOR-DB-SYSGEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CA-ENDEVOR-DB-SYSGEN-DYN 1</td>
</tr>
</tbody>
</table>
Note: Two sets of source modules for the system programs are supplied with CA Endevor /DB Release 15.0. One set of modules specifies the system programs are installed with storage protect on when they are defined to the system; the other set specifies the system programs are installed with storage protect off. The second character of the module name designates the storage protect option, x is either:

D -- the programs are installed with storage protect on

N -- the programs are installed with storage protect off

For the source module's location and additional information about the installation options, see the CA IDMS Installation and Maintenance section -- z/OS.

1 -- Some members may contain definitions for programs that are eligible to use null PDEs and therefore are not pre-defined to the system. These members are designated by the '-' DYN' suffix and can be added to the system if your site requires all programs be defined.

VTAM Considerations

The following VTAM entries are included in the VTAMLST data set for VTAM1:

VTAM Entries

| APPL MAJOR NODE: SYSTEM01 APPL AUTH=(ACQ,NOPASS,NVPACE,NOTSO,SPO) DDSN ODE1 APPL AUTH=(NOACQ,NOPASS,NVPACE,NOTSO,SPO), PARSSESS=YES CDRSC MAJOR NODE: DDSNODE2 CDRSC CDRM=CDRM02 |

The following VTAM entries are included in the VTAMLST data set for VTAM2:
System Generation Entries 1

The following examples show the CA IDMS system generation statements required to define the systems.

**SYSTEM01 sysgen**

```
SYSTEM 01

LINE VTAM01
  ENABLED
  TYPE IS VTAMLIN
  APPL ID IS SYSTEM01...
  PTERM...

LINE DDS01
  ENABLED
  TYPE IS DDS
  SOURCE=DDSNODE1.

PTERM PT01DDS1
  TYPE VTAM
  BLOCKSIZE 8192
  TARGET DDSNODE2
  WEIGHT 20.

PTERM PT02DDS1
  TYPE VTAM
  BLOCKSIZE 8192
  TARGET DDSNODEX (NOT SHOWN ABOVE)
  WEIGHT 30.

PTERM PT01BULK
  TYPE BULK.

LTERM LT01BULK
  PTERM IS PT01BULK.
```

**SYSTEM02 sysgen**

```
SYSTEM02

LINE VTAM02
  ENABLED
  TYPE IS VTAMLIN
  APPL ID IS SYSTEM02....
  PTERM....

LINE DDS02
  ENABLED
  TYPE IS DDS
  SOURCE = DDSNODE2

PTERM PT01DDS2
  TYPE VTAM
```
Note: VTAM minor node names must be unique throughout the VTAM network. Therefore, the APPL ID IS parameter on the VTAMLIN LINE statement (used to define the 3270 terminals) and the SOURCE parameter on the DDS LINE statement cannot be the same even though these LINE statements participate in the same CA IDMS system.

DC/UCF System Generation Statements

The following statements must be included in the DC/UCF system definition to establish communication between DC/UCF and VTAM:

- One **LINE statement** that specifies TYPE IS VTAMLIN. The LINE statement does not define an actual teleprocessing line but rather a control block (PLE). Actual line definitions and addressing are managed by the Network Control Program (NCP) and VTAM. In the LINE statement, you specify the application identifier (and, optionally, the password) that the DC/UCF system uses to sign on to VTAM. DC/UCF stores the application identifier and the password in the access method control block (ACB) used for communication with VTAM. When the DC/UCF system opens the ACB, VTAM searches the VTAMLST data set for a matching identifier and password.

- One **PTERM statement** for each physical terminal to be associated with the VTAM line.

- One **LTERM statement** for each PTERM statement.

Syntax for the LINE, PTERM, and LTERM statements is presented in *Teleprocessing Network Statements* (see page 280).

The sample DC/UCF configuration shown in the previous diagram consists of two systems, SYSTEM01 and SYSTEM02, running on a single machine. The DC/UCF system definitions include the following statements:

**SYSTEM01**

```
ADD SYSTEM 01
  .
  .
ADD LINE VTAM01
  ENABLED
```
TYPE IS VTAMLIN
APPLICATION ID IS DCSYS01

ADD PTERM PV01001
ENABLED
IN LINE VTAM01
PRINTER CLASS IS 1
TYPE IS V3277

ADD PTERM PV01002
ENABLED
IN LINE VTAM01
PRINTER CLASS IS 1
TYPE IS V3277

ADD PTERM PV01017
ENABLED
IN LINE VTAM01
PRINTER CLASS IS 1
TYPE IS V3277

ADD PTERM PR01001
ENABLED
TYPE IS V3286
LINE VTAM01
FORMFEED
NAME=DEMO001
ACQUIRE

ADD PTERM PR01002
ENABLED
TYPE IS V3286
LINE VTAM01
FORMFEED
NAME=DEMO002
ACQUIRE

ADD PTERM PR01003
ENABLED
TYPE IS V3286
LINE VTAM01
FORMFEED
NAME=DEMO003
ACQUIRE

ADD PTERM PR01004
ENABLED
TYPE IS V3286
LINE VTAM01
FORMFEED
NAME=DEMO004
ACQUIRE

ADD LTERM LV01001 ENABLED NOPRINTER PTERM IS PV01001.
ADD LTERM LV01002 ENABLED NOPRINTER PTERM IS PV01002.
ADD LTERM LV01003 ENABLED NOPRINTER PTERM IS PV01003.

ADD LTERM LV01017 ENABLED NOPRINTER PTERM IS PV01017.
ADD LTERM PR01001 PTERM IS PR01001 ENABLED PRINTER CLASS=(1).
ADD LTERM PR01002 PTERM IS PR01002 ENABLED PRINTER CLASS=(2).
ADD LTERM PR01003 PTERM IS PR01003 ENABLED PRINTER CLASS=(3).
ADD LTERM PR01004 PTERM IS PR01004 ENABLED PRINTER CLASS=(4).

SYSTEM02
ADD SYSTEM 02
.
.
ADD LINE VTAM02
   ENABLED
   TYPE IS VTAMLIN
   APPLICATION ID IS DCSYS02 PASSWORD IS YOURPASS
.
ADD PTERM PV02001
   ENABLED
   IN LINE VTAM02
   PRINTER CLASS IS 1
   TYPE IS V3277
   .
.
ADD PTERM PV02002
   ENABLED
   IN LINE VTAM02
   PRINTER CLASS IS 1
   TYPE IS V3277
   .
.
ADD PTERM PV02017
   ENABLED
   IN LINE VTAM02
   PRINTER CLASS IS 1
   TYPE IS V3277
   .
.
ADD PTERM PR02001
   ENABLED
   TYPE IS V3286
   LINE VTAM02
   FORMFEED
   NAME=DEMO001
   ACQUIRE

ADD PTERM PR02002
   ENABLED
   TYPE IS V3286
   LINE VTAM02
   FORMFEED
   NAME=DEMO002
   ACQUIRE

ADD PTERM PR02003
   ENABLED
   TYPE IS V3286
   LINE VTAM01
   FORMFEED
   NAME=DEMO003
   ACQUIRE

ADD PTERM PR02004
   ENABLED
   TYPE IS V3286
   LINE VTAM01
   FORMFEED
   NAME=DEMO004
   ACQUIRE

ADD LTERM LV02001 ENABLED NOPRINTER PTERM IS PV02001.
ADD LTERM LV02002 ENABLED NOPRINTER PTERM IS PV02002.
ADD LTERM LV02003 ENABLED NOPRINTER PTERM IS PV02003.
   .
   .
ADD LTERM LV02017 ENABLED NOPRINTER PTERM IS PV02017.
VTAMLST Entries

The VTAMLST data set contains information about the network configuration and network resources. The two members of the VTAMLST data set that contain relevant information are the APPL type major node and the LOCAL type major node.

APPL Type Major Node Definition

<table>
<thead>
<tr>
<th>VBUILD TYPE=APPL</th>
<th>DCYS01 APPL</th>
<th>AUTH=(ACQ,NOPASS,NVPACE,NOTSO,NPO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCYS02 APPL</td>
<td>AUTH=(ACQ,NOPASS,NVPACE,NOTSO,NPO), PRTCT=YOURPASS</td>
<td></td>
</tr>
<tr>
<td>TESTDC1 APPL</td>
<td>AUTH=(ACQ,NOPASS,NVPACE,NOTSO,NPO)</td>
<td></td>
</tr>
<tr>
<td>TESTDC2 APPL</td>
<td>AUTH=(ACQ,NOPASS,NVPACE,NOTSO,NPO)</td>
<td></td>
</tr>
<tr>
<td>INTERACT APPL</td>
<td>AUTH=(NOACQ,NOPASS,NVPACE,NOTSO,NPO)</td>
<td></td>
</tr>
<tr>
<td>INTERTST APPL</td>
<td>AUTH=(NOACQ,NOPASS,NVPACE,NOTSO,NPO)</td>
<td></td>
</tr>
</tbody>
</table>

Sample LOCAL Type Major Node

The following description shows a LOCAL type major node used to define local (that is, channel-attached) terminals.

<table>
<thead>
<tr>
<th>LOCGRP1 LBUILD</th>
<th>CUL0A2 LOCAL</th>
<th>TERM=3277, CUADDR=0A2, ISTATUS=INACTIVE, FEATUR2=(ANKEY, MODEL2, PFK, NOSELPEN), USSTAB=CULUSS, MODETAB=CULLMODE, DLOGMOD=N3277M2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CUL0A3 LOCAL</td>
<td>TERM=3277, CUADDR=0A3, ISTATUS=ACTIVE, FEATUR2=(ANKEY, MODEL2, PFK, NOSELPEN), USSTAB=CULUSS, MODETAB=CULLMODE, DLOGMOD=N3277M2</td>
</tr>
<tr>
<td></td>
<td>CUL0A4 LOCAL</td>
<td>TERM=3277, CUADDR=0A4, ISTATUS=ACTIVE, FEATUR2=(ANKEY, MODEL2, PFK, NOSELPEN), USSTAB=CULUSS, MODETAB=CULLMODE, DLOGMOD=N3277M2</td>
</tr>
<tr>
<td></td>
<td>CUL0A5 LOCAL</td>
<td>TERM=3277, CUADDR=0A5, ISTATUS=ACTIVE, FEATUR2=(ANKEY, MODEL2, PFK, NOSELPEN), USSTAB=CULUSS, MODETAB=CULLMODE, DLOGMOD=N3277M2</td>
</tr>
<tr>
<td></td>
<td>CUL0A6 LOCAL</td>
<td>TERM=3277, CUADDR=0A6, ISTATUS=ACTIVE, FEATUR2=(ANKEY, MODEL2, PFK, NOSELPEN), USSTAB=CULUSS, MODETAB=CULLMODE, DLOGMOD=N3277M2</td>
</tr>
<tr>
<td></td>
<td>DEMO001 LOCAL</td>
<td>TERM=3284, CUADDR=0BD, ISTATUS=ACTIVE, FEATUR2=(MODEL2), DLOGMOD=N3280T3, MODETAB=CULLMODE</td>
</tr>
</tbody>
</table>

Mode Table

The LOCAL type major node references a mode table that describes different 3270-type devices.

The following table shows a mode table that defines both non-SNA and SNA devices. The mode table contains recommended values for each device; however, other values can be used when appropriate.
MODE TITLE 'MODE TABLE FOR CA'
   AIF $(SNASW).LBL1
   *
   MNOTE *,'THIS MODE TABLE IS FOR NON-SNA DEVICES'
   *
   CULNMODE
   CULNMODE MODETAB
   *
   NON-SNA DEFINITIONS -- TERMINALS
   *
   * N3278M2
     MODEENT LOGMODE=N3278M2,FMPROF=X'02',TSPROF=X'02',
        - PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000',
        - RUSIZES=X'8989',
        - PSERVIC=X'00000000000185018500200'
   *
   * N3277M1
     MODEENT LOGMODE=N3277M1,FMPROF=X'02',TSPROF=X'02',
        - PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000',
        - RUSIZES=X'8787',
        - PSERVIC=X'000000000000C280C280100'
   *
   * N3277M2
     MODEENT LOGMODE=N3277M2,FMPROF=X'02',TSPROF=X'02',
        - PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000',
        - RUSIZES=X'8787',
        - PSERVIC=X'000000000000C280C507F00'
   *
   * N3278M1
     MODEENT LOGMODE=N3278M1,FMPROF=X'02',TSPROF=X'02',
        - PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000',
        - RUSIZES=X'8989',
        - PSERVIC=X'00000000000000000000200'
   *
   * N3278M3
     MODEENT LOGMODE=N3278M3,FMPROF=X'02',TSPROF=X'02',
        - PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000',
        - RUSIZES=X'8989',
        - PSERVIC=X'00000000000000000000200'
   *
   * N3278M4
     MODEENT LOGMODE=N3278M4,FMPROF=X'02',TSPROF=X'02',
        - PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000',
        - RUSIZES=X'8989',
        - PSERVIC=X'00000000000000000000200'
   *
   * N3278M5
     MODEENT LOGMODE=N3278M5,FMPROF=X'02',TSPROF=X'02',
        - PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000',
        - RUSIZES=X'8989',
        - PSERVIC=X'00000000000000000000200'
   *
   * N3279M2
     MODEENT LOGMODE=N3279M2,FMPROF=X'02',TSPROF=X'02',
        - PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000',
        - RUSIZES=X'8989',
        - PSERVIC=X'00000000000185018500200'
   *
   * N3279M3
     MODEENT LOGMODE=N3279M3,FMPROF=X'02',TSPROF=X'02',
        - PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000',
        - RUSIZES=X'8989',
        - PSERVIC=X'00000000000185018500200'
   *
   * DSILGMD
     MODEENT LOGMODE=DSILGMD,FMPROF=X'02',TSPROF=X'02',
        - PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000',
        - RUSIZES=X'8787',
        - PSERVIC=X'000000000000000000000200'
   *
   * D4B32782
     MODEENT LOGMODE=D4B32782,FMPROF=X'02',TSPROF=X'02',
        - PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000',
        - RUSIZES=X'0000',
        - PSERVIC=X'0000000000018500007E00'
   *
   * NON-SNA DEFINITIONS -- PRINTERS
   *
   * N3280T1
     MODEENT LOGMODE=N3280T1,FMPROF=X'02',TSPROF=X'02',
        -
PRIPROT=X'71', SECPROT=X'40', COMPROT=X'2000', RUSIZES=X'8089', PSERVIC=X'01000000000000000000000000000000'

*N3280T3
MODEENT LOGMODE=N3280T3, FMPROF=X'02', TSPROF=X'02',
PRIPROT=X'71', SECPROT=X'40', COMPROT=X'2000', RUSIZES=X'8089', PSERVIC=X'0300000000018501850200'

*N32804K
MODEENT LOGMODE=N32804K, FMPROF=X'02', TSPROF=X'02',
PRIPROT=X'71', SECPROT=X'40', COMPROT=X'2000', RUSIZES=X'8787', PSERVIC=X'0300000000018502B507F00'

.LBL1  ANOP
     AIF (NOT &SNASW).LBL2
     *
     MNOTE *, 'THIS MODE TABLE IS FOR SNA DEVICES'
     *
     * CULSMODE
     CULSMODE MDEDAT
     *
     * SNA DEFINITIONS -- TERMINALS
     *
     * S3278M1
     MODEENT LOGMODE=S3278M1, FMPROF=X'03', TSPROF=X'03',
     PRIPROT=X'B1', SECPROT=X'90', COMPROT=X'3080', RUSIZES=X'8785', PSERVIC=X'020000000000C280C507F00'
     *
     * S3278M2
     MODEENT LOGMODE=S3278M2, FMPROF=X'03', TSPROF=X'03',
     PRIPROT=X'B1', SECPROT=X'90', COMPROT=X'3080', RUSIZES=X'88C7', PSERVIC=X'02000000000185018507F00'
     *
     * S3278M3
     MODEENT LOGMODE=S3278M3, FMPROF=X'03', TSPROF=X'03',
     PRIPROT=X'B1', SECPROT=X'90', COMPROT=X'3080', RUSIZES=X'88C7', PSERVIC=X'0200000000018502B507F00'
     *
     * S3278M4
     MODEENT LOGMODE=S3278M4, FMPROF=X'03', TSPROF=X'03',
     PRIPROT=X'B1', SECPROT=X'90', COMPROT=X'3080', RUSIZES=X'88C7', PSERVIC=X'02000000000185025050F00'
     *
     * S3278M5
     MODEENT LOGMODE=S3278M5, FMPROF=X'03', TSPROF=X'03',
     PRIPROT=X'B1', SECPROT=X'90', COMPROT=X'3080', RUSIZES=X'88C7', PSERVIC=X'0200000000018501B847F00'
     *
     * S3279M2
     MODEENT LOGMODE=S3279M2, FMPROF=X'03', TSPROF=X'03'
     PRIPROT=X'B1', SECPROT=X'90', COMPROT=X'3080', RUSIZES=X'88C7', PSERVIC=X'02000000000185028507F00'
     *
     * S3279M3
     MODEENT LOGMODE=S3279M3, FMPROF=X'03', TSPROF=X'03',
     PRIPROT=X'B1', SECPROT=X'90', COMPROT=X'3080', RUSIZES=X'88C7', PSERVIC=X'0200000000018501B847F00'
     *
     * SNA DEFINITIONS -- PRINTERS
     *
     * S3280256
     MODEENT LOGMODE=S3280256, FMPROF=X'03', TSPROF=X'03'
     PRIPROT=X'B1', SECPROT=X'90', COMPROT=X'3080', RUSIZES=X'8585', PSERVIC=X'0100000000018501850200'
     *
     * S3280T3
     MODEENT LOGMODE=S3280T3, FMPROF=X'03', TSPROF=X'03'
     PRIPROT=X'B1', SECPROT=X'90', COMPROT=X'3080',
### Runtime Considerations

If VTAM fails or is taken down during DC/UCF system execution, you do not have to take the system down. To resume processing, the VTAM ACB must be closed, then reopened after VTAM has been restarted.

The following example illustrates the above procedure:

- The VTAM line XYZ is varied offline after failure with the following command:
  
  ```
  DCMT VARY LINE XYZ OFFLINE
  ```

- VTAM is started up (if not done previously).

- The VTAM line XYZ is varied online with the following command:
  
  ```
  DCMT VARY LINE XYZ ONLINE
  ```

DCMT commands are described in the CA IDMS System Tasks and Operator Commands section.

### Runtime Flow of Control

Flow of control at runtime determines which dialog in an application is executed and, within a dialog, which premap or response process is executed and whether a mapin or mapout operation is performed. Most batch flow of control considerations are similar to those for online applications.

An overview of batch runtime flow of control is provided under "Batch Dialog Structure" and "Application Structure" in CA ADS Batch Concepts (https://docops.ca.com/display/IDMS19/CA+ADS+Batch+Concepts). A detailed description of the flow of control on a mapin operation is provided below.

The diagram below shows the flow of control on a mapin operation for an application not defined using the application compiler.
Notes:

1. A response process is selected by the occurrence of a batch control event (BCE) or response field value (RFV). If the BCE or RFV do not invoke a response process and if a physical input error did not occur, the dialog's default response process, if any, is selected.

2. The response process is not executed if the record contains input errors and EXECUTE ON EDIT ERRORS is NO for the response process. Instead, the record is written to the suspense file, error messages are sent to the log file, and another record is read and mapped in.

3. The application aborts if the same dialog encounters an end-of-line condition for a second time.

The following two diagrams show the runtime flow of control on a mapin operation for an application that is defined using the application compiler, as follows:

- The first diagram below shows the examination of the input record's response field and the mapin of the record. At runtime, when a mapin operation is to be performed, the runtime system first examines the record's response field, if any. If the response field value invokes an immediately executable function that is different from the current function, control passes to that function and the record is not mapped in. The next time a mapin operation is performed for that file, the record is mapped in immediately.
The runtime system maintains a MAPIN-READY flag for each input file in the application. The flag is set ON when the response field for a record has been examined, but the record has not yet been mapped into variable storage. When the flag is OFF, a mapin operation begins with an examination of the record’s response field. When the flag is ON, a mapin operation begins with the record being mapped into variable storage.

- The following diagram shows the selection of an application function or dialog response process. The order of precedence in function or response process selection is as follows:

1. A valid immediately executable function that is not the same as the current function.

2. A valid dialog response process. If a response process is selected and the current record has not yet been mapped in, it is mapped in and the MAPIN-READY flag is reset OFF.

3. A valid function.
Notes:

1 A function is selected by the occurrence of a batch control event (BCE) or a response field value (RFV). If the BCE or RFV is associated with an application response that is valid for the current function, then the function invoked by the response is selected. The function invoked by the current function's default response is selected if no valid function is selected, if a physical input error did not occur, and if the RFV is spaces (or the map has no response field or an end-of-file condition has occurred).

2 By default, all functions in the batch environment are immediately executable.

3 A response process is selected by the occurrence of a BCE or RFV. If the BCE and RFV do not invoke a response process, and if a physical input error did not occur, the dialog's default response process, if any, is selected.

4 The response process is not executed if the record contains input errors and EXECUTE ON EDIT ERRORS is NO for the response process. Instead, the record is written to the suspense file, error messages sent to the log file, and another record is examined and/or mapped in.

5 If a deferred function was also selected, it can be executed by an EXECUTE NEXT FUNCTION command.

6 In this case, the selected function may be the same as the current function. If so, the associated dialog is reexecuted, beginning with its premap process or mapin operation, as applicable.
The application abends if the same dialog encounters an end-of-file condition for a second time.

Runtime Flow of Control

The runtime flow of control is as follows:

1. A dialog performs a mapin operation that examines the response field value of the next input record. The record’s response field value invokes an immediately executable function that is not the same as the current function. Control passes to that function, which executes its associated dialog. The record has not yet been mapped into variable storage.

2. The second dialog performs a mapin operation on the same file. The record is immediately mapped into variable storage.

3. The response field value for the record mapped in is associated with an immediately executable function; this time, however, the function is the same as the current function, so the function is not reexecuted.

4. The runtime system selects and executes a response process. (If no response process can be selected, the current function is re-executed from the beginning of the dialog).

Specifying File Characteristics

You specify file characteristics for each file that the application accesses. File characteristics include record format, block size, logical record length, file type, and device type.

The following topics are discussed separately below:

- Where to specify file characteristics
- Specifying a record length
- Specifying a block size
- Default file characteristics
- Specifying characteristics of concatenated data sets
- Specifying characteristics of suspense files
- Tape labels
Where to Specify File Characteristics

*z/OS*

Under *z/OS*, you can specify a file's characteristics at any of the following levels:

- **In the file's associated IDD file entity** -- Any characteristics you specify here do not have to be specified in the JCL or data set label. However, characteristics specified in IDD are enforced at the other levels; if a characteristic is specified both in IDD and at another level, the IDD specifications take precedence.

⚠️ **Note**: Do not specify a file type or device type at the IDD level. If you specify a characteristic in IDD and you want to change it later, you must recompile all maps that use the file. Therefore, you may not want to provide any characteristics for a file at the IDD level.

- **In the JCL** -- Characteristics you specify in the JCL are enforced at the data set label level.

  - **As part of the data set's label**.

*z/VSE*

Under *z/VSE*, you can specify a file's characteristics at either of the following levels:

- **In the file's associated IDD file entity** -- Any characteristics you specify here do not have to be specified in IDMSFILE control statements. However, characteristics specified in IDD are enforced at the IDMSFILE level; if a characteristic is specified both in IDD and at the IDMSFILE level, the IDD specifications take precedence.

⚠️ **Note**: Do not specify a file type or device type at the IDD level. If you specify a characteristic in IDD and you want to change it later, you must recompile all maps that use the file. Therefore, you may not want to provide any characteristics for a file at the IDD level.

- **In IDMSFILE control statements** -- IDMSFILE is a program that enables you to specify file characteristics at runtime in the form of control statements. For a detailed description of IDMSFILE, see *z/VSE File Characteristics Program* ([https://docops.ca.com/pages/viewpage.action?pageId=328580516](https://docops.ca.com/pages/viewpage.action?pageId=328580516)).

**Defaults**

CA ADS Batch applies defaults to characteristics that are not specified at any level. For physical sequential files, the record format defaults to undefined, the logical record length to 0, and the block size to 6233 for disk files and 32760 for tape files. For VSAM ESDS files, CA ADS Batch uses the information supplied from the catalog.
Specifying a Record Length

The record length you specify for a file must be consistent with the record lengths of the IDD record entities that are associated with the file and that are used in the application, as follows.

**Fixed Record Format**

If the record format is fixed, the record length of the file must be equal to the record length of each associated record entity.

**Variable Record Format**

If the record format is variable, the record length of the file must be greater than or equal to the record length of the longest associated record entity plus four.

**Undefined Record Format**

If the record format is undefined, the record length of the file must be greater than or equal to the record length of the longest associated record entity.

The RECORD SIZE specification for a file can be removed by using IDD to set it to 0 (zero).

Specifying a Block Size

The block size you specify for a file must be consistent with the logical record length of the file, as follows.

**Fixed Blocked Record Format**

If the record format is fixed blocked, the block size must be an integral multiple of the logical record length.

**Fixed Unblocked Record Format**

If the record format is fixed unblocked, the block size must equal the logical record length.

**Variable Blocked Record Format**

If the record format is variable blocked, the block size must be greater than or equal to the logical record length plus four.

**Variable Unblocked Record Format**

If the record format is variable unblocked, the block size must equal the logical record length plus four.

**Undefined Record Format**

If the record format is undefined, the block size must be greater than or equal to the logical record length.

The RECORD SIZE specification for a file can be removed by using IDD to set it to 0 (zero).
Default File Characteristics

If you do not specify a file's record format, logical record length, and/or block size, CA ADS Batch supplies defaults for the omitted characteristics.

If the record format is not specified, CA ADS Batch supplies one of the following characteristics:

- **U** (undefined), for a physical sequential file
- **F** (fixed), for a SYSIN file
- **FA** (fixed, with control characters), for a SYSOUT file

Once the record format is determined, CA ADS Batch supplies defaults for block size and logical record length based on the record format:

**Fixed or Fixed-block Record Format**

- **If the logical record length and the block size are zero (not specified),** CA ADS Batch uses a block size of 6233 for physical sequential files, 80 for SYSIN files, and 133 for SYSOUT files, and uses a logical record length equal to the block size.

- **If the logical record length is not zero and the block size is zero,** CA ADS Batch uses a block size equal to the logical record length.

- **If the logical record length is zero and the block size is not zero,** CA ADS Batch uses a logical record size equal to the block size.

**Variable or Variable-blocked Record Format**

- **If the logical record length and the block size are zero (not specified),** CA ADS Batch uses a block size of 6233 for physical sequential files, 80 for SYSIN files, and 133 for SYSOUT files, and uses a logical record length equal to the block size minus 4.

- **If the logical record length is not zero and the block size is zero,** CA ADS Batch uses a block size equal to the logical record length plus 4.

- **If the logical record length is zero and the block size is not zero,** CA ADS Batch uses a logical record size equal to the block size minus 4.

**Undefined Record Format**

- **If the logical record length is not zero and the block size is zero,** CA ADS Batch uses a block size equal to the logical record length.

- **If the block size is zero,** CA ADS Batch uses a block size of 6233 for physical sequential files, 80 for SYSIN files, and 133 for SYSOUT files.

**Specifying Characteristics of a Concatenated Data Set**

You define a concatenated data set (z/OS) by coding multiple DD statements in your JCL for a single ddname. For CA ADS Batch, characteristics that you specify for the file at the IDD level are enforced for every data set.
Rules for Concatenated Data Sets

Concatenated data sets must conform to the following rules:

- Data sets can be of different device types; for example, disk, tape, or scheduler (SYSIN).
- The data set with the largest block size must appear first in the concatenation.
- The record formats must be equivalent; that is:
  - All fixed or fixed blocked, or a mixture of the two
  - All variable or variable blocked, or a mixture of the two
  - All undefined
- The record lengths must correspond to the record format:
  - If the record format is fixed or fixed blocked, the logical record lengths must be equal.
  - If the record format is variable or variable blocked, the logical record lengths must be less than or equal to the first logical record length.
  - If the record format is undefined, the logical record length is not applicable.

In addition, the data sets must conform to standard z/OS rules for concatenated data sets; for example, the maximum number of data sets allowed in a concatenation is 255.

Specifying Characteristics of a Suspense File

Suspense files use the IDD-level characteristics specified for the file's associated input file. Suspense file characteristics are enforced when they are defined in the IDD file entity associated with the map. For example, if an input file's IDD description specifies a record format of fixed, that characteristic is enforced for the suspense file.

All characteristics supplied in the JCL or data set label of a suspense file must match those of the associated input file.

Tape Labels

Tape labeling options include the following:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL</td>
<td>IBM standard labels</td>
</tr>
<tr>
<td>SUL</td>
<td>IBM standard and user labels</td>
</tr>
<tr>
<td>AL</td>
<td>ANSI labels</td>
</tr>
<tr>
<td>AUL</td>
<td>ANSI and user labels</td>
</tr>
<tr>
<td>NSL</td>
<td>Nonstandard labels</td>
</tr>
<tr>
<td>NL</td>
<td>No labels (unlabeled)</td>
</tr>
</tbody>
</table>
CA ADS Batch processes IBM standard and ANSI labels, but bypasses user labels and nonstandard labels on input. On output, user labels and nonstandard labels cannot be written to a tape file.

**Application Components and Physical Data Sets**

The relationship between application components and physical data sets begins with IDD file entities and ends with the description of the files in the JCL or data set labels.

**IDD File Entities**

You define IDD file entities to represent the input and output files in the application. You do not define file entities for suspense and log files. Suspense files use the file entity descriptions of their associated input files. Log files are not described in the data dictionary.

A single file entity can represent several files, as long as the characteristics included in the file entity are common to all files that the entity represents. For example, you do not have to specify any characteristics as part of the file entity; therefore, you can define one file entity to represent all files in an application.

More information on specifying file characteristics is provided earlier in this section.

**IDD Record Entities**

You define record entities that describe the layout of data for all input and output files in an application. You must associate each record entity with the appropriate file entity; if you are using the IDD DDDL compiler, you do this by specifying the INCLUDE WITHIN FILE clause of the RECORD statement.

**Defining File Maps**

You define file maps that associate file record entities with dialog variable storage.

**Defining Dialogs**

You define dialogs that associate file maps with the application.

Described below are some different combinations you can use when associating maps with dialogs in an application:

- You can associate a map with several dialogs.

- You can specify that a map is one dialog's input map and is the same or another dialog's output map.

- In an application that accesses a file with multiple record layouts, you define a map for each layout and associate each map with its own dialog.

You can associate a dialog with one input map, one output map, or both. You assign a suspense file to a dialog that is associated with an input file by specifying the runtime label (z/OS ddname, z/VSE filename) of the suspense file either during dialog definition or in a runtime control statement.
Note that IDD elements in records used by an CA ADS Batch application should not specify an external picture that includes P (for numeric data).

Runtime Labels

Runtime labels (z/OS ddname, z/VSE filename) associate file maps and suspense files with physical data sets. You specify runtime labels during dialog definition or in runtime control statements. Specifications made in control statements supersede those made during dialog definition.

You can specify the same runtime label for different file maps. For example, in an application that accesses a file with multiple record layouts, you define a file map for each record layout. During dialog definition or at runtime, you specify the same runtime label for each file map.

You can specify different runtime labels for the same file map. For example, in an application that reads records from one file and writes them to another, you can define a single file map for both files, since they have the same record layout. You associate the file map with one dialog as its input file map, and specify the runtime label of the input file. You associate the file map with the same dialog or another dialog as its output file map, and specify the runtime label of the output file.

Data Sets

You describe the application's physical data sets in the JCL and identify the data sets to the application by means of runtime labels.

Application Restartability

Contents

- Checkpoint Record (see page 377)
- Taking a Checkpoint at Runtime (see page 378)
- Restarting the Application After an Abend (see page 379)

An application is restartable if it can be resumed at some checkpoint after an abend occurs. A restartable application must be able to reestablish the database, input, and output files, and the general application environment that existed at the checkpoint.

The following topics on application restartability are discussed separately below:

- The checkpoint record
- Taking a checkpoint at runtime
- Restarting an application after an abend

Checkpoint Record

The application should keep a checkpoint record that holds information on the status of the application. A checkpoint record should include:

- A checkpoint identification number
Counts of the number of records, by dialog, that have been written to or read from each input, output, and suspense file

A copy of all variable storage information that will have to be restored if the application abends, such as:

- Condition flags
- Subscripts
- Report totals

Taking a Checkpoint at Runtime

Your application should take its checkpoints with the object of being able to restart after an abend at the highest level dialog.

The highest level dialog should perform the checkpoint or should pass control to a dialog whose purpose is to take the checkpoint. The checkpoint should do the following:

1. Update the input, output, and suspense file record count fields of the checkpoint record.
2. Close all files in order to clear the file buffers; the disposition of each file should be MOD.
3. Perform a database commit. Between checkpoints, you should leave the run unit open by using the NOSAVE parameter in LINK and INVOKE commands, and the NOFINISH parameter in TRANSFER commands. Committing the database at any point other than the checkpoint can jeopardize restartability.
4. Move required variable and status information to the checkpoint record.
5. Increment the checkpoint record identification number.
6. Write the checkpoint record to an output file or a queue file.
7. If the record is being written to an output file, close the file to ensure that the record is physically written.

The first two tasks must be performed by the dialogs that access the files. The dialog performing the checkpoint can link to each of these dialogs, with a flag set that indicates that a checkpoint is being performed. The dialogs can have premap processes that perform special processing when the flag is set. The processing includes incrementing the record count fields of the checkpoint record and closing the files.

Considerations

The following considerations apply:

- Dialogs to which control will be transferred during a checkpoint must not have as their entry points the mapin operation. If such an entry point is desired, the premap process can issue a READ TRANSACTION command immediately, unless a checkpoint is being performed.
If a file has been accessed by more than one dialog, you must link to and issue a CLOSE command in each dialog that has accessed the file. If you do not, the file will not be physically closed.

The record counts for a dialog's input, output, and suspense files are contained in the system-supplied fields $INPUT-COUNT, $OUTPUT-COUNT, and $ERROR-COUNT.

You update the checkpoint record's record count fields by adding the system-supplied data fields for each dialog to the checkpoint record fields. Each time you issue a CLOSE command in a dialog, the system-supplied data fields for the closed files are reset to zero. At a checkpoint, therefore, the value of a data field equals the number of records read or written only since the last checkpoint.

Your application should have a checkpoint interval, based on a value such as the number of input records read or the time elapsed since the last checkpoint was taken. The checkpoint interval should not be so frequent as to substantially increase processing time, nor be so infrequent as to leave uncommitted more transactions than you would be willing to have to reprocess should the application abend.

Restarting the Application After an Abend

In central version, if the application abends, the runtime system automatically rolls back the database to its state at the time of the last database commit, which should have occurred at the time of the last checkpoint. (In local mode, you must use journals and rollback utilities to roll back the database.) Therefore, the database should already be set for restart.

Restart at the Highest Level Dialog

Your application should be restarted at the highest level dialog; the dialog can link to a dialog that performs the restart and then returns control to the highest level dialog.

Restart Steps

The restart should include the following steps:

1. **Read the checkpoint record** -- Read the record, then, if the record is being stored in a sequential file, close the file.

2. **Restore the variable fields** -- Move the saved variable fields from the checkpoint record to the appropriate records in dialog variable storage. You do not (and, in fact, cannot) restore the record count fields.

3. **Set the input files' pointers** -- For each input file to be restored, link to a dialog whose sole function is to read the file until the file pointer is set at the proper position. The dialog determines how many records to read by using the record count field stored in the checkpoint record. After setting the file pointers, the dialog should close the file. The disposition of the restored file should be MOD.

**Note:** If the file was accessed by several dialogs, you may have several checkpoint record fields for the file. If so, you should use the total count of the fields to determine how many records to read.
If the file has multiple record layouts, you will have to use multiple dialogs to read the record and you must have an application structure that determines which dialog reads the various types of records.

4. **Set the output and suspense files' pointers** -- For each output and suspense file to be restored, link to a dialog whose sole function is to copy the file to a new file, until the proper number of records have been written. Consider the following scenario. An application writes 100 records to an output file before taking a checkpoint. The application writes another 10 records to the file, then abends. The runtime system empties the buffer; as a result, 110 records have been physically written to the file. Your restart procedure must create a new file that consists of only the first 100 records. The dialog responsible for restoring the output file reads the file's records and writes them to a new file. The dialog determines how many records to copy by using the record count field stored in the checkpoint record. After the file has been restored, the dialog should close both files. The disposition of the new file should be MOD. The dialogs that will be writing to the new output or suspense file should specify the runtime label of the new file.

![Note: If the file was accessed by several dialogs, you may have several checkpoint record fields for the file. If so, you should use the total count of the fields to determine how many records to read.](image)

If the file has multiple record layouts, you will have to use multiple dialogs to copy the record and you must have an application structure that determines which dialog reads the various types of records.

5. **Return to the highest level dialog and resume processing.**

---

**CA ADS Batch Trace Facility**

The CA ADS Batch trace facility is a debugging aid that traces the flow of control and commands executed in an application at runtime.

**Steps in the Trace Facility**

To use the trace facility, perform the following steps:

1. **During dialog definition**, specify that a symbol table is to be included in the dialogs you want to trace. Symbol tables are required by the facility to list the process commands executed at runtime. If you are using the online dialog compiler, you request a symbol table on the Options and Directives screen.

2. **At runtime**, include TRACE statements as part of your control statements. TRACE statements specify:
   - **The dialogs to be traced** -- One, several, or all dialogs in an application can be traced.
When the dialogs are to be traced -- By default, a selected dialog is traced every time it becomes operative in the application thread. Alternatively, you can specify that tracing begin only after the dialog has become operative for the nth time, that tracing end when the dialog becomes operative for the nth time, or that tracing occur only every nth time that the dialog becomes operative.

The level of detail of the trace -- You can specify one of three levels of detail for the trace of a dialog:

a. Dialog and process module execution -- The trace facility documents the beginning of dialog and premap and response process execution.

b. Subroutine execution -- The trace facility documents dialog and process module execution, as in level one, and additionally documents the beginning of subroutine execution.

c. Command execution -- The trace facility documents dialog, process module, and subroutine execution, as in level two, and additionally documents all commands executed within the dialog. Additionally, detailed information is provided on database commands and on database currency saves and restores.

Syntax for the TRACE statement is provided under Runtime Control Parameters (see page 385).

The trace facility writes trace records to the log file as DEBUG records. You can print the records by using the CA ADS Batch print log utility, as discussed in CA ADS Batch Print Log Utility (https://docops.ca.com/display/IDMS19/CA+ADS+Batch+Print+Log+Utility). Alternatively, you can send all log file output directly to the printer at runtime.

Sample Output

Sample output from a trace facility run is shown in the report below. The output columns are as follows:

- Execution number (first column) -- The execution occurrence for the dialog. The lines in the sample were created while the dialog was operative for the first time (00000001).

- Dialog name (second column) -- The name of the dialog currently executing. The sample shows that dialog REPTEDIT executes first, then dialog HDRLINE, and then dialog REPTEDIT again.

- Process name (third column) -- The name of the premap or response process currently executing. JEB-EMPL-REPT-EDIT is the first process executed in the sample trace facility run.

Note: A line that documents the beginning of a dialog has the following literal in the process name field:

********* DIALOG-ENTRY *********

- Subroutine name (fourth column) -- The name of the process subroutine currently executing. PAGHDR is the first subroutine executed in the sample trace facility run.
Note: A line that documents the beginning of a dialog, the beginning of a process, or a process command that is not within a subroutine has the following literal in the subroutine name field:

**MAIN**

- **Sequence number field** (fifth column) -- The IDD sequence number of the command being executed. The first four commands executed in the sample trace facility run begin at lines 100, 300, 400, and 500. These line numbers are the same ones used by ADSORPTS in its report of a dialog's processes.

Note: For more information on ADSORPTS, see the CA ADS Reference Section. The line that documents the beginning of a dialog, process, or subroutine has 00000000 in this field.

- **Process command** (sixth column) -- The process command currently executing. The first command executed in the sample trace facility run is IF.

Note: A line that documents the beginning of a dialog, process, or subroutine has the literal ENTRY in this field.

- **Command offset** (last column) -- The hexadecimal offset of the command from the beginning of the dialog's fixed dialog block (FDB). These offsets are the same ones used by ADSORPTS in its list of command elements (CMEs) for a process.

Note: For more information on ADSORPTS, see the CA ADS Reference Section.

The process being traced in the sample trace facility report does not include database commands or database currency saves and restores. However, if a process did and if tracing were being performed at the command execution level, the report would include detailed information on these database functions.

If the module is an included module, the included module name appears in the report after the last column.

```
00000001 REPTEDIT ****** DIALOG-ENTRY ****** **MAIN** 00000000 ENTRY...
00000001 REPTEDIT JEB-EMPL-REPT-EDIT **MAIN** 00000000 ENTRY...
00000001 REPTEDIT JEB-EMPL-REPT-EDIT **MAIN** 0000100 IF 004F8
00000001 REPTEDIT JEB-EMPL-REPT-EDIT **MAIN** 0000300 DC ACC 00524
00000001 REPTEDIT JEB-EMPL-REPT-EDIT **MAIN** 0000400 MOVE 00538
00000001 REPTEDIT JEB-EMPL-REPT-EDIT **MAIN** 0000500 CALL 00574
00000001 REPTEDIT JEB-EMPL-REPT-EDIT CVRTDATE 00000000 ENTRY...
00000001 REPTEDIT JEB-EMPL-REPT-EDIT CVRTDATE 0013700 MOVE 0019DC
00000001 REPTEDIT JEB-EMPL-REPT-EDIT CVRTDATE 0013800 MOVE 001A94
00000001 REPTEDIT JEB-EMPL-REPT-EDIT CVRTDATE 00000000 MOVE 00184C
00000001 REPTEDIT JEB-EMPL-REPT-EDIT CVRTDATE 00000000 GOBACK 001C04
00000001 REPTEDIT JEB-EMPL-REPT-EDIT **MAIN** 00000600 MOVE 00058C
```
Multiple External Run Units Under CA ADS Batch

Preparing the system to handle multiple external run units (ERUs) for CA ADS Batch is a DBA or system administrator issue, not a programmer issue.

However, an CA ADS Batch application run at a Distributed Database System (DDS) site should be designed so that a single database run unit is maintained for the application's duration. It is particularly important that the application avoid saving and then restoring database currencies if the application's run unit is accessing a database at a different DDS node than the default node for central version operations.

An CA ADS Batch application's database requests are routed to the batch interface module (IDMSINTB). IDMSINTB passes the request through the CA IDMS SVC (for central version requests) or directly to DC/UCF (for local mode requests). In DC/UCF, the database request is handled as an external run unit as is any other batch database request.

At system generation, it is necessary to specify a value for the MAXIMUM ERUS parameter in the SYSTEM statement. An CA ADS Batch application occasionally requires up to five ERUs at the same time:

- **User database run unit**, required if a dialog is accessing the database
- **Message run unit**, required to retrieve message text from the dictionary
- **Loader run unit**, required to retrieve load modules from the load area
If the MAXIMUM ERUS count is exceeded at runtime, your CA ADS Batch will be unable to load modules (for maps, dialogs, and so forth). If you receive a message to this effect, verify that the MAXIMUM ERUS specification for the system is at least 3. This specification is made in the system generation SYSTEM statement.

**Note:** For more information, see the *CA IDMS System Generation Section*.

## Operator Shutdown

CA ADS Batch enables the console operator to instruct the runtime system to terminate an application.

If operator shutdown is enabled at runtime, the runtime system begins by issuing a WTOR (write-to-operator with reply) macro. At any time during application execution, the operator can issue a SHUTDOWN command, described below, that causes the runtime system to terminate the application with an optional dump. If shutdown is disabled, the operator can terminate the application only by canceling the job.

The system default for operator shutdown is determined using the ADSOBSYS utility; the ADSOBSYS default is DISABLED. Control statements specified at runtime can override the system default. For more information on enabling operator shutdown, see [ADSOBSYS](https://docops.ca.com/display/IDMS19/ADSOBSYS) and Control Statements (see page 386).

If operator shutdown is enabled, the operator can issue a SHUTDOWN command to terminate the application. Syntax for the SHUTDOWN command is shown below:

```
[SHUTDOWN][DUMP][SD]
```

- **SHUTDOWN**
  Specifies that the runtime system is to terminate the application.

- **DUMP**
  Specifies that a system dump of the application is to be produced.

- **SD**
  Is equivalent to SHUTDOWN DUMP.

**Note:** Under BS2000/OSD, the SHUTDOWN command must be entered by means of the BS2000/OSD /INFORM-PROGRAM command. For example, if the application runs as an interactive task, enter:

```
/INFORM-PROGRAM MSG='SHUTDOWN DUMP'
```
Runtime Control Parameters

Contents
- PARM Field Parameters (see page 385)
- Control Statements (see page 386)

Runtime control parameters determine the CA ADS Batch runtime environment. These parameters can be specified at system generation using ADSOBSYS and/or at runtime using PARM field parameters and control statements.

When a runtime control parameter is specified in more than one place, the following order of precedence applies:

1. Control statements
2. PARM field parameters
3. ADSOBSYS

PARAM field parameters and control statements are discussed separately below. ADSOBSYS is discussed in ADSOBSYS (https://docops.ca.com/display/IDMS19/ADSOBSYS).

PARM Field Parameters

Input parameters allow you to specify the following:

- **A mainline dialog or application task code** -- The runtime system begins application execution at the specified dialog or at the function associated with the specified task code. The function must be associated with a mainline dialog.
  Alternatively, you can specify a dialog or task code using the ENTRY POINT control statement, as described later in this section. The ENTRY POINT statement overrides the input parameter specification.

- **Runtime parameter** -- You can optionally specify an alphanumeric character string as a parameter that is to be passed to the application. Dialog processes can read the string into variable storage by issuing an ACCEPT RUN PARAMETERS command, as described in Process Command Language (https://docops.ca.com/pages/viewpage.action?pageId=309429410).

Specifying Parameters

Under z/OS and z/VSE, you specify input parameters in the PARMS field parameter of the EXEC statement. The mainline dialog or application task code is specified as the first parameter. The runtime parameter is specified as the third parameter. The second parameter is reserved for future use. Parameters must be separated by commas. The value assigned to the job variable may not exceed 10 characters. For example:

```
PARM='DIALOG1'
PARM='TASKA,,ABCDEFG'
PARM=',,ABCDEFG'
```
Control Statements

Control statements help to establish the runtime environment for an CA ADS Batch application. Each control statement must be terminated by a period. Control statements can appear in any order.

Syntax for the control statements is shown below:

```
PRImary pool  primary-pool-size .
               IS =

SECondy pool  secondary-pool-size .
               IS =

DIAlog  STATistics  OFF
       DLG                      ON
          ALL
       /FOR ( dialog-name )

. CHEckpoint  interval  checkpoint-interval
       CKPT  IS =

ROUting codes ( route-code ).
       ARE
       IS =

DEScriptor codes ( descriptor-code ).
       ARE
       IS =

DIAlog  dialog-name  has  MAXERR
       DLG  MAXimum  ERRor

count of  maximum-errors .

OPerator SHUtdown  ENAbled .
       IS  YES
       =  DIEnabled
       NO

MAXimum LINks  maximum-links .
       IS =

COMmit when files open causes  NOACTion .
       IS  WARNing
```
LOG FILE1 run-time-label IS =
COUNT1 file1-maximum-records IS =
FILE2 run-time-label IS =
COUNT2 file2-maximum-records IS =
ARCHive
NOARCHive
COMpress
NOCOMpress
PREfix
NOPREfix

DIAlgo dialog-name INput run-time-lbl
DLG OUTput runtime LABEL IS
SUSpense File name = DDName

DEFault print CLAss print-class IS
DESTination print-destination IS =

DBName database-name IS =

DBNode database-node-name IS =

DICTName dictionary-name IS =

DICTNode dictionary-node-name IS =

REQUESTor user-id
USER IS =

ENTRY point DIAlgo dialog-name
DLG IS =
TASK application-task-code
IS

TRAce dialog-name .
   ALL BEFore MAXimum execution-count
   AFTER 0 execution-count
   EVEry 1 execution-count
   SUBRoutines COMmands

• PRIMARY POOL IS primary-pool-size
  Specifies the size, in bytes, of the primary buffer pool. Primary-pool-size is an integer in the range 0 through 2,147,483,647. The default is determined by specifications made using ADSOBSYS; the ADSOBSYS default is set at system generation.

Note: For more information on specifying primary pools, see the discussion of the ADSO statement in the CA IDMS System Generation Section.

• SECONDARY POOL IS secondary-pool-size
  Specifies the size, in bytes, of the secondary buffer pool. The secondary pool is allocated from the system storage pool when the primary buffer pool becomes full. Secondary-pool-size is an integer in the range 0 through 2,147,483,647. The default is determined by specifications made using ADSOBSYS; the ADSOBSYS default is set at system generation.

• DIALOG STATISTICS OFF/ON
  Specifies whether dialog statistics are to be collected during application execution.
  OFF specifies that statistics are not to be collected.
  ON specifies that statistics are to be collected. The default is determined by specifications made using ADSOBSYS; the ADSOBSYS default is OFF.
  Parameters included in the ON clause are as follows.

  • ALL/FOR (dialog-name)
    Specifies whether dialog statistics are maintained for all dialogs (ALL) or for only selected dialogs in an application. Multiple dialogs must be separated by commas.
    The default is determined by specifications made using ADSOBSYS. ADSOBSYS allows you to specify ALL or SELECTED. If you specify SELECTED, you must list the selected dialogs in the DIALOG STATISTICS control statement at runtime.

  • CHECKPOINT INTERVAL IS checkpoint-interval
    Specifies the frequency with which dialog statistics are to be written to the log file. Statistics are written to the log file once every checkpoint-interval time that statistics are accumulated.
    CA ADS Batch accumulates statistics every time a dialog issues a control command. The default checkpoint interval is determined by specifications made using ADSOBSYS; the ADSOBSYS default is set at system generation.

  • ROUTING CODES ARE (route-code)
    (z/OS only) specifies the z/OS operator-message routing codes, as described in the applicable operating system supervisor services and macro instructions. This parameter supplies the value of
the ROUTCDE parameter for WTO (write-to-operator) macros issued by the system. Route-code is an integer in the range 1 through 16. Multiple routing codes must be separated by commas. The default routing codes are determined by specifications made using ADSOBSYS; the ADSOBSYS default is 11.

- **DESCRIPTOR CODES ARE (descriptor-code)**
  (z/OS only) specifies the z/OS operator-message description code, as described in the applicable operating system supervisor services and macro instructions. This parameter supplies the descriptor code to the DESC parameter for WTO (write-to-operator) macros issued by the system. Descriptor-code is an integer in the range 1 through 16. The default descriptor codes are determined by specifications made using ADSOBSYS; the ADSOBSYS default is 7.

- **DIALOG dialog-name HAS MAXERR COUNT OF maximum-errors**
  Specifies the maximum number of error records that can be sent to the suspense file of the specified dialog. If this number is exceeded at runtime, the runtime system terminates the application. For example, if the maximum error count is 1, the runtime system terminates the application when the second error record is to be written. If the maximum error count is 0, the runtime system allows an unlimited number of error records to be sent to a suspense file. Maximum-errors is an integer in the range 0 through 32,767. The default is determined by specifications made using ADSOBSYS; the ADSOBSYS default is 0.

- **OPERATOR SHUTDOWN IS ENABLED/DISABLED**
  Specifies whether the operator can send a request to the runtime system to terminate an application. If operator shutdown is enabled, the runtime system begins application execution by issuing a WTOR (write-to-operator with reply) macro. At any time during application execution, the operator can issue a SHUTDOWN command, as described earlier in this section. The SHUTDOWN command causes the runtime system to terminate the application with an optional dump. If shutdown is disabled, the operator can terminate the application only by canceling the job. The default is determined by specifications made using ADSOBSYS; the ADSOBSYS default is DISABLED.

- **COMMIT WHEN FILES OPEN CAUSES NOACTION/WARNING/ABEND**
  Specifies the action to be taken when a run unit is committed before all files used in the application have been closed, as follows:
  - **NOACTION** specifies that no action is taken.
  - **WARNING** specifies that a warning message is sent to the log file.
  - **ABEND** specifies that the application is abended.

  The default is determined by specifications made using ADSOBSYS; the ADSOBSYS default is ABEND. Committing a run unit when files are open can jeopardize application restartability after an abend, as described under Application Restartability (see page 377).

- **LOG**
  LOG parameters specify defaults for the log file; all LOG parameters are optional.
  - **FILE1 IS run-time-label**
    Specifies the runtime label (z/OS ddname, z/VSE filename) of the primary log file. The default is determined by specifications made using ADSOBSYS; the ADSOBSYS default is ADSLOGA.
COUNT1 IS file1-maximum-records
Specifies the number of log records that are written to the primary log file before the file is considered full. This parameter is meaningless for spool or tape log files.
If file1-maximum-records is reached at runtime, the runtime system switches to the secondary log file or, if no secondary log file is allocated, wraps around to the beginning of the primary log file.
If 0 is specified, no predefined limit is placed on the number of records written to the primary log file. If space for a disk log file is exceeded at runtime, the runtime system abnormally terminates the application.
File1-maximum-records is an integer in the range 1 through 99999. The default is determined by specifications made using ADSOBSYS; the ADSOBSYS default is 0.

FILE2 IS run-time-label
Specifies the runtime label of the secondary log file. The default is determined by specifications made using ADSOBSYS.

COUNT2 IS file2-maximum-records
Specifies the number of log records that are written to the secondary log file before the file is considered full.
If file2-maximum-records is reached at runtime, the runtime system switches back to the primary log file.
If 0 is specified, no predefined limit is placed on the number of records written to the secondary log file. If space for a disk log file is exceeded at runtime, the runtime system abnormally terminates the application.
File2-maximum-records is an integer in the range 1 through 99999. The default is determined by specifications made using ADSOBSYS; the ADSOBSYS default is 0.

ARCHIVE/NOARCHIVE
(z/OS only) specifies whether log file archiving is to be performed at runtime when a log file is full. For more information on log file archiving, see "Log files" in CA ADS Batch Concepts (https://docops.ca.com/display/IDMS19/CA+ADS+Batch+Concepts). The default is determined by specifications made using ADSOBSYS; the ADSOBSYS default is NOARCHIVE.

COMPRESS/NOCOMPRESS
Specifies whether log records are compressed in the log file to save space at runtime. The default is determined by specifications made using ADSOBSYS. If neither ADSOBSYS nor a runtime control statement specifies COMPRESS or NOCOMPRESS, the runtime system uses the z/OS, or z/VSE device-type assignment to determine whether to compress the records:

- In z/OS, records are not compressed if the assignment is SYSOUT; otherwise, records are compressed.
- In z/VSE, records are not compressed if the device type is PRINTER (as specified at system generation using ADSOBSYS or at runtime); otherwise, records are compressed.

PREFIX/NOPREFIX
Specifies whether a prefix is to precede each log record. Prefixes are used by the print log utility to select records. The default is determined by specifications made using ADSOBSYS. If neither ADSOBSYS nor a runtime control statement specify PREFIX or NOPREFIX, the runtime system uses the z/OS or z/VSE device type assignment to determine whether to include a prefix:
In **z/OS**, a prefix is not included if the assignment is SYSOUT; otherwise, a prefix is included.

In **z/VSE**, a prefix is not included if the device type is PRINTER (as specified at system generation using ADSOBSYS or at runtime); otherwise, a prefix is included.

**DIALOG**
The DIALOG statement associates a dialog's input, output, and suspense files with runtime labels (**z/OS ddnames, z/VSE filenames**). Each input, output, and suspense file of each dialog must be associated with a runtime label.

**Note:** You can associate a file with a runtime label during dialog definition as well. In such a case, you do not have to specify a runtime label in a control statement at runtime. If a runtime label is specified in both places, the control statement specification takes precedence.

DIALOG statement parameters are described below.

- **dialog-name**
The name of the dialog.

- **INPUT/OUTPUT/SUSPENSE**
  Specifies whether the file identification information that follows pertains to the dialog's input, output, or suspense file.

- **RUNTIME LABEL/FILE NAME/DDNAME IS run-time-label**
  Specifies the runtime label input, output, or suspense file. RUNTIME LABEL, FILE NAME, and DDNAME are synonymous.

**Note:** To request that a suspense file not be allocated for the input file, even though a suspense file runtime label was specified during dialog definition, specify blanks within single quotation marks, as in the example below:

```
DIALOG XXX INPUT RUNTIME LABEL INFILE
   SUSPENSE RUNTIME LABEL ' '.
```

The following example specifies information on a **z/OS** dialog's input, output, and suspense files:

```
DIALOG DIALOG1
   INPUT DDNAME IS INPUT1
   OUTPUT DDNAME IS OUTPUT1
   SUSPENSE DDNAME IS SUS1.
```

**DEFAULT PRINT CLASS IS print-class/DESTINATION IS print-destination**
Specifies the default print class to which a report is assigned or the default print destination to which a report is routed. A report is created using WRITE PRINTER commands. **Print-class** must be an integer in the range 1 through 64. **Print-destination** is a 1- to 8-character destination not enclosed in quotation marks.
• **DBNAME IS** *database-name*
  Optionally specifies the CA IDMS/DB database to be accessed by the application.

• **DBNODE IS** *database-node-name*
  Optionally specifies the DDS node that controls the CA IDMS/DB database to be accessed by the application.

• **DICTNAME IS** *dictionary-name*
  Optionally specifies the data dictionary in which the definitions and load modules for the requested application are stored.

• **DICTNODE IS** *dictionary-node-name*
  Optionally specifies the DDS node that controls the data dictionary in which the definitions and load modules for the requested application are stored.

• **REQUESTOR IS** *user-id*
  Specifies the 1- to 32-character id of a user. The user id can be accepted into dialog variable storage by issuing an ACCEPT USER ID command, and is automatically moved to the prefix of all log file records written during the application. *User-id* is not enclosed in quotation marks.

• **ENTRY POINT**
  Specifies the entry point into the application.
  If the ENTRY POINT statement is not included, an entry point must be specified as an input parameter, as discussed earlier in this section. The ENTRY POINT statement overrides the input parameter.

  - **DIALOG IS** *dialog-name*
    Specifies the name of a mainline dialog.

  - **TASK IS** *application-task-code*
    Specifies a task code associated with an application function. The function must be associated with a mainline dialog.

• **TRACE**
  Activates the CA ADS Batch trace facility for a dialog.
  For a detailed discussion of the trace facility, see **CA ADS Batch Trace Facility (see page 380)**.
  TRACE statement parameters are described below.

  - **dialog-name/ALL**
    Specifies the dialog for which tracing is activated; the parameters BEFORE, AFTER, EVERY, COMMANDS, and SUBROUTINES apply to the named dialog. ALL specifies that tracing is performed for all dialogs; the parameters BEFORE, AFTER, EVERY, COMMANDS, and SUBROUTINES apply to all dialogs that have not been named explicitly in the TRACE statement. To selectively activate tracing for several dialogs, repeat the parameters of the TRACE statement for each dialog.

  - **BEFORE MAXIMUM/execution-count**
    Specifies that tracing is to be performed each time the dialog becomes operative, up to but not including the *execution-count*th time. MAXIMUM (default) specifies tracing is to be performed every time the dialog becomes operative, unless restricted by AFTER and EVERY specifications.
- **AFTER 0/execution-count**
  Specifies that tracing is to be performed each time the dialog becomes operative beyond the execution-count time. Zero (default) specifies tracing is to be performed every time the dialog becomes operative, unless restricted by BEFORE and EVERY specifications.

- **EVERY 1/execution-count**
  Specifies that tracing is to be performed each time the execution count for the dialog reaches a multiple of execution-count. One (default) specifies tracing is to be performed every time the dialog becomes operative, unless restricted by AFTER and BEFORE specifications. specifies that tracing is to be performed only every execution-countth time that the dialog becomes operative. One (default) specifies that tracing is performed each time the dialog becomes operative, unless restricted by BEFORE and AFTER specifications.

- **SUBROUTINES**
  Specifies that subroutine entry points are to be documented, in addition to dialog and process entry points.

- **COMMANDS**
  Specifies that process commands are to be documented, in addition to dialog, process, and subroutine entry points.

If neither SUBROUTINES nor COMMANDS is specified, only dialog and process entry points are documented for the specified dialog. Multiple TRACE statements can be specified for the same dialog. The last BEFORE, AFTER, and EVERY specifications coded for a dialog apply to that dialog for the entire execution of the application. The most detailed level of documentation coded applies to the dialog for the entire execution; COMMANDS is more detailed than SUBROUTINES, which is more detailed than no specification at all.

**Loading Runtime Components**

**Order of Precedence**

Runtime components (such as subschema load modules, dialog load modules, application definition blocks, and map load modules) can be stored in load areas and load libraries. The runtime system uses the following order of precedence when loading a component:

1. Main memory of the batch region, in the event that the component has already been loaded
2. The load area of the secondary dictionary, if any, specified in a control statement
3. The load area of the system's primary dictionary
4. A load library specified in the STEPLIB statement, if any

**Note:** If an application snaps the application definition block (ADB) or fixed dialog block (FDB) of an application or dialog that was loaded from STEPLIB, only the header portion is snapped.
An error status of 1474 may be returned if an CA ADS Batch program running in local mode attempts to use a subschema that was compiled for use on CA IDMS/DC without a DMCL name.

CA ADS Batch can link to a user-written program. The program can perform database commands. The program, however, cannot perform QUEUE, WRITE PRINTER, or other DC/UCF-oriented commands.

A user-written COBOL, or PL/I program accessed by an CA ADS Batch application must have been submitted to the DML preprocessor with a mode of BATCH.

**JCL**

The JCL for using the CA ADS Batch runtime system (ADSBATCH) to execute an application is shown below.

**Sample z/OS JCL Under Central Version ADS BATCH (z/OS)**

```plaintext
//ADSBATCH EXEC PGM=ADSBATCH,REGION=1024K,parm=parameter
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DSN=idms.loadlib,DISP=SHR
//IDMSSNAP DD SYSOUT=A
//sysctl DD DSN=idms.sysctl,DISP=SHR
//adsloga DD DSN=log.file,DISP=OLD
//file.descriptions
//SYSLST DD SYSOUT=A
//SYSIDMS DD *
DMCL=dmcl-name
DICTNAME=dictionary-name
Other SYSIDMS parameters, as appropriate
/*
//SYSIPT DD *
control statements
/*
//SYSUDUMP DD SYSOUT=A
parm=par an optional PARM parameter
idms.dba. loadlib data set name of the load library containing the DMCL and database name table load modules
idms. loadlib data set name of the load library containing the CA IDMS executable modules
sysctl ddname of SYSCTL file
idms. sysctl data set name of SYSCTL file
adsloga ddname of the log file
log.file data set name of the log file
file. descriptions file descriptions of all input, output, and suspense files. The ddnames specified must match those specified in control statements or dialog definitions.
```

Note: To get a snap dump when an abend occurs during a mapping operation, you must
Note: To get a snap dump when an abend occurs during a mapping operation, you must specify a DD statement with the name IDMSSNAP. The IDMSSNAP file stores the snap dump. An example of a mapping abend is when an input record that contains errors cannot be written to a suspense file because the MAXIMUM ERRORS for the dialog, which is specified at system generation or at runtime, has been exceeded.

Archive Log File

For an archive log file, specify a ddname of ARCLOG\textit{n}, where \textit{n} is 1 for the first archive file, 2 for the second archive file, and so forth, up to 9. The following sample JCL allocates the first archive log file:

```jcl
//ARCLOG1 DD DSN=MEN.CC.RQE.ARCTAP1,DISP=(NEW,CATLG,DELETE),
//        UNIT=(TAPE,,DEFER),LABEL=(,SL),VOL=(,RETAIN),
//        DCB=(RECFM=VB,LRECL=320,BLKSIZE=12804)
```

Local mode

To execute ADSBATCH in local mode, the following steps are required:

1. Remove the sysctl DD statement.
2. Add the following statements:

   ```jcl
   //sysjrnl DD DSN=idms.tapejrnl,DISP=NEW,UNIT=tape
   //dictdb DD DSN=idms.dictdb,DISP=OLD
   //dloddb DD DSN=idms.dloddb,DISP=SHR
   //dmsgdb DD DSN=idms.dmsgdb,DISP=SHR
   //dlogdb DD DSN=idms.dlogdb,DISP=SHR
   additional journal file assignments, as required
   ```

Note: Under local mode, if you are running a multi-level application which transfers control using a LINK command without the NOSAVE option, you must code the LOCKING=ON parameter in the SYSIDMS parameter file.

<table>
<thead>
<tr>
<th>ddname</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysjrnl</td>
<td>ddname of journal file</td>
</tr>
<tr>
<td>idms.tapejrnl</td>
<td>data set name of journal file</td>
</tr>
<tr>
<td>tape</td>
<td>symbolic device name of journal file</td>
</tr>
<tr>
<td>dictdb</td>
<td>ddname of data dictionary</td>
</tr>
<tr>
<td>idms.dictdb</td>
<td>data set name of data dictionary</td>
</tr>
<tr>
<td>dloddb</td>
<td>ddname of data dictionary load area</td>
</tr>
<tr>
<td>idms.dloddb</td>
<td>data set name of data dictionary load area</td>
</tr>
<tr>
<td>dmsgdb</td>
<td>ddname of data dictionary message area</td>
</tr>
<tr>
<td>idms.dmsgdb</td>
<td>data set name of data dictionary message area</td>
</tr>
<tr>
<td>dlogdb</td>
<td>ddname of data dictionary log area</td>
</tr>
<tr>
<td>idms.dlogdb</td>
<td>data set name of data dictionary log area</td>
</tr>
</tbody>
</table>
Sample z/VSE JCL Under Central Version ADSBATCH (z/VSE)

// DLBL SYSTCL,'idms.sysctl', 1999/365,SD
// DLBL ciln,'idms.cilib'
// EXTENT nnnnnn
// LIBDEF *,SEARCH=(ciln),TEMP
// DLBL adsloga,'log.file'
// EXTENT sysnnn,VOLSER,,start-track,end-track
// ASSGN sysnnn,DISK,VOL=VOLSER,SHR
// DLBL SYSIDMS,'#SYSIPT'
// file.descriptions
// EXEC IDMSFILE,SIZE=(AUTO,128K)
SYSIDMS control statements
IDMSFILE control statements
ADSBATCH control statements
DIALOG control statements

<table>
<thead>
<tr>
<th>idms.sysctl</th>
<th>data set name of SYSCTL file</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciln</td>
<td>filename of core-image library</td>
</tr>
<tr>
<td>idms.cilib</td>
<td>file-id of DC/UCF core-image library</td>
</tr>
<tr>
<td>nnnnnn</td>
<td>volume serial number</td>
</tr>
<tr>
<td>adsloga</td>
<td>filename of the log file</td>
</tr>
<tr>
<td>log.file</td>
<td>file-id of the log file</td>
</tr>
<tr>
<td>sysnnn</td>
<td>logical unit of the device on which the log file is located</td>
</tr>
<tr>
<td>start-track</td>
<td>starting track of the log file data set</td>
</tr>
<tr>
<td>end-track</td>
<td>ending track of the log file data set</td>
</tr>
<tr>
<td>file.descriptions</td>
<td>file descriptions of all input, output, and suspense files used in the application. Filenames specified must match those specified in control statements.</td>
</tr>
</tbody>
</table>

SYSIDMS control statements used by SYSIDMS to describe physical runtime environments as described in CA IDMS Common Facilities Section -- Volume 2.

IDMSFILE control statements used by IDMSFILE to describe the characteristics of the input, output, suspense, and log files used in the application, as described in z/VSE File Characteristics Program (https://docops.ca.com/pages/viewpage.action?pageId=328580516). This must include the 'RUN PROGRAM ADSBATCH' card.

DIALOG control statements required if any of the dialogs to be executed have an input map with the 'SYSIPT' label.

ADSBATCH control statements used by ADSBATCH to establish the runtime environment.

CH control statements
What to Consider

A /* must follow each SYSIPT file. If an abend occurs during a file mapping operation, the snap dump is written automatically to SYSLST.

Local Mode

To execute ADSBATCH in local mode, the following steps are required:

1. Remove the SYSCTL DLBL card

2. Add the following statements:

   // DLBL    dictdb,'idms.dictdb',,DA  
   // EXTENT  sys005,nnnnnn  
   // ASSGN   sys005,DISK,VOL=nnnnnn,SHR  
   // DLBL    dloddb,'idms.dloddb',,DA  
   // EXTENT  sys017,nnnnnn  
   // ASSGN   sys017,DISK,VOL=nnnnnn,SHR  
   // DLBL    dmsgdb,'idms.dmsgdb',,DA  
   // EXTENT  sys016,nnnnnn  
   // ASSGN   sys016,DISK,VOL=nnnnnn,SHR  
   // DLBL    dlogdb,'idms.dlogdb',,DA  
   // EXTENT  sys019,nnnnnn  
   // ASSGN   sys019,DISK,VOL=nnnnnn,SHR  
   // TLBL    sysjrnl,'idms.tapejrnl',,nnnnn,,f  
   // ASSGN   sys009,TAPE,VOL=nnnnnn  

additional journal file assignments, as required

<table>
<thead>
<tr>
<th>dictdb</th>
<th>filename of data dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.dictdb</td>
<td>file-id of data dictionary</td>
</tr>
<tr>
<td>sys005</td>
<td>logical unit assignment for data dictionary</td>
</tr>
<tr>
<td>nnnnnnn</td>
<td>volume serial number</td>
</tr>
<tr>
<td>dloddb</td>
<td>filename of data dictionary load area</td>
</tr>
<tr>
<td>idms.dloddb</td>
<td>file-id of data dictionary load area</td>
</tr>
<tr>
<td>sys017</td>
<td>logical unit assignment for data dictionary load area</td>
</tr>
<tr>
<td>dmsgdb</td>
<td>filename of data dictionary message area</td>
</tr>
<tr>
<td>idms.dmsgdb</td>
<td>file-id of data dictionary message area</td>
</tr>
<tr>
<td>sys016</td>
<td>logical unit assignment for data dictionary message area</td>
</tr>
<tr>
<td>dlogdb</td>
<td>filename of data dictionary log area</td>
</tr>
<tr>
<td>idms.dlogdb</td>
<td>file-id of data dictionary log area</td>
</tr>
<tr>
<td>sys019</td>
<td>logical unit assignment for data dictionary log area</td>
</tr>
<tr>
<td>sysjrnl</td>
<td>filename of tape journal file</td>
</tr>
<tr>
<td>idms.tapejrnl</td>
<td>file-id of tape journal file</td>
</tr>
<tr>
<td>f</td>
<td>file number of tape journal file</td>
</tr>
<tr>
<td>sys009</td>
<td>logical unit assignment for tape journal file</td>
</tr>
</tbody>
</table>

Local Mode
To execute ADSBATCH in local mode, the following steps are required:

1. Remove the ADD-FILE-LINK statement for sysctl

2. Add the following statements:

```
/ADD-FILE-LINK L-NAME=dictdb,F-NAME=idms.appldict.dictdb,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=dloddb,F-NAME=idms.appldict.dloddb,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=dcmsg,F-NAME=idms.sysmsg.ddldcmsg,SHARED-UPD=*YES
/ADD-FILE-LINK L-NAME=sysjrnl,F-NAME=idms.tapejrnl
```

additional user database file assignments
additional journal file assignments, as required

<table>
<thead>
<tr>
<th>dictdb</th>
<th>linkname of data dictionary file</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.appldict.dictdb</td>
<td>filename of data dictionary file</td>
</tr>
<tr>
<td>dloddb</td>
<td>linkname of data dictionary load area file</td>
</tr>
<tr>
<td>idms.appldict.dloddb</td>
<td>filename of data dictionary load area file</td>
</tr>
<tr>
<td>dcmsg</td>
<td>linkname of data dictionary message area file</td>
</tr>
<tr>
<td>idms.sysmsg.ddldcmsg</td>
<td>filename of data dictionary message area file</td>
</tr>
<tr>
<td>sysjrnl</td>
<td>linkname of tape journal file</td>
</tr>
<tr>
<td>idms.tapejrnl</td>
<td>filename of tape journal file</td>
</tr>
</tbody>
</table>

Real APPC Support Considerations

The VTAM mode table entry for the logical unit used for APPC must specify:

- LOGMODE=vtam-modeent-name
- TYPE=0
- PSNDPAC=X'03'
- SSNDPAC=X'03'
- SRCVPAC=X'03'
- FMPROF=X'13'
- TSPROF=X'07'
- PRIPROT=X'B0'
- SECPROT=X'B0'
- COMPROT=X'S081'
- RUSIZE=X'8585'
- PSERVIC=X'060200000000000000102F00'
In the parameters listed above, **vtam-modeent-name** must match the name specified in the MODEENT parameter of a PTERM statement used to define a physical terminal for APPC.

**Mode Table Entry**

The following VTAM mode table entry defines the bind parameters for APPC:

```
MODEENT LOGMODE=SNAAPPC1, -
      TYPE=0, -
      PSNDPAC=X'03', -
      SSNDPAC=X'03', -
      SRCVPAC=X'03', -
      FMPROF=X'13', -
      TSPROF=X'07', -
      PRIPROT=X'B0', -
      SECPROT=X'B0', -
      COMPROT=X'50B1', -
      RUSIZE=X'8585', -
      PSERVIC=X'060200000000000000102F00'
```

**SNA and LU Considerations**

**SNA Terminology**

The following SNA terms appear in the descriptions of the LINE and PTERM statement parameters used to implement SNA support:

- **A logical unit (LU)** is a port through which users access the SNA network. For example, an LU can be a CRT terminal, a device such as a displaywriter, or a program such as CICS or the DC/UCF system. Each LU is a single network addressable unit (NAU) to VTAM.

- **A session** is a logical connection between two LUs. Multiple sessions can exist between two LUs that share a single physical connection. Each session is represented in the DC/UCF environment by a physical terminal element (PTE)/logical terminal element (LTE) pair. DC/UCF treats each session as a local resource.

- **A conversation** is a complete transaction between LUs. A conversation is delineated on a session by a begin bracket and an end bracket. Only one conversation can be allocated to a session at a time.

**Bind Parameters for LU 6.2 Sessions**

The mode table entry for an LU 6.2 should include the following parameter specifications:

```
LOGMODE=vtam-modeent-name
COS=class-of-service-name
TYPE=0
FMPROF=X'13'
TSPROF=X'07'
PRIPROT=X'B0'
SECPROT=X'B0'
```
In the parameter specifications listed above:

- **Vtam-modeent-name** must match the mode table entry identified by the MODEENT parameter of a PTERM statement.

- **Class-of-service-name** must match a name in the VTAM class-of-service table.

- **Meme** specifies the maximum amount of data that the DC/UCF system passes to VTAM. Meme represents two pairs of numbers, each of which consists of a mantissa and an exponent.

### LU 6.2 Restrictions

The DC SNA/VTAM driver does not provide support for the following LU 6.2 functions:

- **MAP_NAME on read and write verbs.** This function is not yet supported by IBM.

- **SYNCPOINT and ROLLBACK support for LU 6.2.**

- **FLUSH verb.** The DC/UCF system buffers read data but does not buffer write data.

- **Program initialization parameters (PIP) on allocate requests.**

- **COBOL and PL/I support for LU 6.2 verbs.**

### Multiple Session Support

**DC SNA Line Driver Functions as SNA Logical Unit**

The DC SNA/VTAM line driver functions as an SNA logical unit. As such, the driver can maintain multiple sessions with one or more other LUs to facilitate program-to-program communication (that is, distributed processing). For example, if twenty sessions exist between a DC/UCF system and CICS, up to twenty tasks executing under DC/UCF can communicate concurrently with twenty tasks executing under CICS.

**How to Define Multiple Sessions**

To define multiple sessions, you include multiple physical terminals with the same VTAM node name in the DC/UCF system definition. You specify the VTAM node name in the NAME parameter of the system generation PTERM statement. The maximum number of sessions that can exist between two LUs is limited by the number of physical terminals you define. The number of sessions available at any given time is limited by the number of physical terminals that are in service.

Multiple sessions between two LUs can use different classes of service or different protocols. The protocols for a session are established by the mode table entry containing the parameters used to bind the session. You use the MODEENT parameter of the system generation PTERM statement to associate a physical terminal with a mode table entry.
Multiple Session Service Manager

To support multiple LU 6.2 sessions, the DC/UCF system must include the multiple session service manager. The service manager negotiates the number of sessions to be allowed with the remote LU.

You define the multiple session service manager with system generation PROGRAM and TASK statements:

- Code an ADD PROGRAM statement for program RHDCCNOS that specifies ASSEMBLER, NOPROTECT, and REENTRANT.
- Code an ADD TASK statement for task 06F1 that specifies INVOKES PROGRAM RHDCCNOS. Set the task priority as high as possible.

To ensure that a session is always available to the LU 6.2 service manager, you must reserve two physical terminals:

- The system generation PTERM statement for each physical terminal must specify MODEENT IS SNASVCMG.
- The CONTENTION parameter of the system generation PTERM statement must specify WINNER for one physical terminal and LOSER for the other physical terminal.

Sample Definitions for SNA Support

The following example shows sample system generation statements and VTAM definitions that implement SNA support for LU 6.2 sessions in a DC/UCF system. The example creates two LUs, each with four sessions. For each LU, two sessions are reserved for the multiple session service manager (mode table entry SNASVCMG).

The system generation statements consist of:

- **One LINE statement** with line type VTAMLU. You need only one LINE statement to support all LUs regardless of LU type.
- **Eight PTERM statements** with physical terminal type LU. All but one of the physical terminals are defined as ACQUIRE. The sessions that use these physical terminals are automatically connected at DC/UCF system startup.
- **One PROGRAM statement** for the multiple session service manager.
- **One TASK statement** for the multiple session service manager.

The VTAM definitions consist of:

- **The VTAMLST entry** for the APPL type major node
- **The mode table entries** with bind parameters for LU 6.2 sessions

System Generation Statements
ADD LINE SNALU1  
  TYPE VTAMLU  
  APPL ID IDMSSNA  
  RPL 8  
  ENABLED.

ADD PTERM LU10  
  TYPE LU  
  ENABLED  
  ACQUIRE  
  NAME IS LU620001  
  MODEENT IS SNASVCMG  
  CONTENTION IS WINNER.

ADD LTERM LU10  
  ENABLED  
  PRIORITY IS 240.

ADD PTERM LU11  
  TYPE LU  
  ENABLED  
  ACQUIRE  
  NAME IS LU620001  
  MODEENT IS SNASVCMG  
  CONTENTION IS LOSER.

ADD LTERM LU11  
  ENABLED  
  PRIORITY IS 240.

ADD PTERM LU12  
  TYPE LU  
  ENABLED  
  ACQUIRE  
  NAME IS LU620001  
  MODEENT IS APPC01  
  CONTENTION IS WINNER.

ADD LTERM LU12  
  ENABLED.

ADD PTERM LU13  
  TYPE LU  
  ENABLED  
  ACQUIRE  
  NAME IS LU620001  
  MODEENT IS APPC01  
  CONTENTION IS LOSER.

ADD LTERM LU13  
  ENABLED.

ADD PTERM LU20  
  TYPE LU  
  ENABLED  
  ACQUIRE  
  NAME IS LU620002  
  MODEENT IS SNASVCMG  
  CONTENTION IS WINNER.

ADD LTERM LU20  
  ENABLED  
  PRIORITY IS 240.

ADD PTERM LU21  
  TYPE LU  
  ENABLED  
  ACQUIRE  
  NAME IS LU620002  
  MODEENT IS SNASVCMG  
  CONTENTION IS LOSER.
ADD LTERM LU21
   ENABLED
   PRIORITY IS 240.

ADD PTERM LU22
   TYPE LU
   ENABLED
   ACQUIRE
   NAME IS LU620002
   MODEENT IS APPC01
   CONTENTION IS WINNER.

ADD LTERM LU22
   ENABLED.

ADD PTERM LU23
   TYPE LU
   ENABLED
   NAME IS LU620002
   MODEENT IS APPC01
   CONTENTION IS OFF.

ADD LTERM LU23
   ENABLED.

ADD PROGRAM RHDCCNOS
   LANGUAGE IS ASSEMBLER
   REENTRANT
   NOPROTECT.

ADD TASK 06F1
   INVOKES PROGRAM RHDCCNOS
   PRIORITY IS 240
   NOINPUT
   INACTIVE INTERVAL IS OFF.

VTAMLST Entry

   VBUILD TYPE=APPL

   IDMSSNA  APPL  ACBNAME=IDMSSNA,
   AUTH=(ACQ,NOPASS,NVPACE,NOTS0,NOP0),
   EAS=20,PARSESS=YES,
   MODETAB=LU62TAB

---

**Important!** Do not specify APPC=YES on the VTAM Application Definition.

Mode Table Entries

<table>
<thead>
<tr>
<th>MODEENT</th>
<th>LOGMODE=APPC01,</th>
<th>(Parallel sessions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE=0,</td>
<td>FMPROF=X'13',</td>
<td>-</td>
</tr>
<tr>
<td>TSPROF=X'07',</td>
<td>PRIPROT=X'B0',</td>
<td>-</td>
</tr>
<tr>
<td>SECPROT=X'B0',</td>
<td>COMPROT=X'50B1',</td>
<td>-</td>
</tr>
<tr>
<td>RUSIZES=X'8787',</td>
<td>PSERVIC=X'060200000000000000002F00'</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MODEENT</th>
<th>LOGMODE=APPC02,</th>
<th>(Single sessions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE=0,</td>
<td>FMPROF=X'13',</td>
<td>-</td>
</tr>
<tr>
<td>TSPROF=X'07',</td>
<td>PRIPROT=X'B0',</td>
<td>-</td>
</tr>
<tr>
<td>SECPROT=X'B0',</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Sample System Definition

This section provides a sample of a complete system definition for a DC/UCF system.

System Definitions Provided at Installation

During installation, definitions of the following DC/UCF systems can be added to the system dictionary:

- **System 90** -- A demonstration system
- **System 99** -- A base system from which definitions can be copied into other systems

Copying Sample System Definitions

You can use the system generation compiler to display the definitions of the systems listed above once they have been added to the data dictionary. Using the COPY statement, you can copy all or part of the base system definition to the site-specific system that you are defining. The COPY statement is described in COPY Statement.

Sample System Definition for SYSTEM 12

The following definition of a DC/UCF system is provided as an example of a complete system definition. The definition is shown as entered by the user rather than as displayed by the compiler. As a result, defaults accepted by the user are not listed.

The sample system is assigned version number 12 and is defined to run under z/OS.

⚠️ **Note:** Parameter values in the systems you define will differ from the values shown for the sample system based on site-specific requirements.

```plaintext
ADD SYSTEM 12
ABEND STORAGE IS 200
```
ABRU SNAP
CHKUSER TASKS IS 4
CUSHION IS 20
CVNUMBER IS 12
CWA SIZE IS 504
DPE COUNT IS DEFAULT
DUMP
ECB LIST IS DEFAULT
GENERATION ID IS SYS12
INACTIVE INTERVAL IS 300
NOJOURNAL RETRIEVAL
CALL LIMIT FOR ONLINE TASKS IS 1000
DBIO LIMIT FOR ONLINE TASKS IS 500
LIMITS FOR ONLINE ARE ENABLED
LOADLIST IS SYSLOAD
LOG DATABASE
MAXIMUM ERUS IS 4
MAXIMUM TASKS IS 22
NEW COPY IS AUTOMATIC
OPERATING SYSTEM IS MVS
PAGE RELEASE IS YES
PRINT KEY IS PF12
PRINTER CHECKPOINT IS 50
PROGRAM POOL IS 100
NOPROTECT
QUEUE JOURNAL BEFORE
RCE COUNT IS DEFAULT
REENTRANT POOL IS 2200
RELOCATABLE THRESHOLD IS 85
REPORT RETENTION IS 5
RESOURCE TIMEOUT INTERVAL IS 1800 PROGRAM IS RHDCBYE VERSION 1
RETRIEVAL NOLOCK
RLE COUNT IS DEFAULT
RUNAWAY INTERVAL IS 45
RUNUNITS FOR LOADER = 1
RUNUNITS FOR MSGDICT = 1
RUNUNITS FOR QUEUE = 1
RUNUNITS FOR SECURITY = 1
RUNUNITS FOR SIGNON = 1
RUNUNITS FOR SYSTEM/DEST = 1
STACKSIZE IS 1200
STATISTICS INTERVAL OFF LINE TASK WRITE NOUSER TRANSACTION
STORAGE KEY IS 9
STORAGE POOL IS 1500
SVC IS 173
SYSCTL IS SYSCTL
SYSLOCKS IS 1200
SYSTRACE ON ENTRIES = 250
TICKER INTERVAL IS 5
UNDEFINED PROGRAM COUNT IS (23 23) FOR (ALL)
UPDATE NOLOCK
USERTRACE ON ENTRIES = 250

************************************************************************
* SECONDARY STORAGE POOLS FOR SYSTEM 12
************************************************************************

ADD STORAGE POOL 127
SIZE IS 1000
CUSHION IS 12
CONTAINS TYPES (USER USER-KEPT)
RELOCATABLE THRESHOLD IS 85

ADD STORAGE POOL 126
SIZE IS 500
CUSHION IS 10
CONTAINS TYPES (SHARED SHARED-KEPT)
RELOCATABLE THRESHOLD IS 85
************************************************************************
*          PRODUCT STATEMENTS FOR SYSTEM 12            *
************************************************************************

ADD ADSO
MAXIMUM LINKS IS 4
PRIMARY POOL IS 4084
SECONDARY POOL IS 4084
RESOURCES ARE RELOCATABLE
DIALOG STATISTICS ON SELECTED

ADD OLM
NEW COPY IS YES
PAGING STORAGE IS 50
QUEUE RETENTION IS 60

ADD OLQ
REPORT RETENTION IS 7
MAXIMUM REPORT RETENTION IS 60
MAXIMUM REPORT PAGES IS 8
MAXIMUM REPORT COUNT IS 15
MAXIMUM INTERRUPT COUNT IS 500
REPORT DICTNAME IS ASF1DICT
MAXIMUM SORT STORAGE IS 500
BATCH CLASS IS 9.

************************************************************************
*          KEYS TABLE FOR SYSTEM 12            *
************************************************************************

INCLUDE MODULE DC-PFKEY-DEF

************************************************************************
*          TASK AND PROGRAM STATEMENTS FOR ONLINE COMPONENTS            *
*          (EXCEPT THE SYSTEM GENERATION COMPILER)            *
************************************************************************

*    CA ADS

INCLUDE MODULE ADSO

*    AUTOMATIC SYSTEM FACILITY

INCLUDE MODULE ASFIDB

INCLUDE MODULE ASF

*    ONLINE COMPILERS (EXCEPT SYSGEN)

INCLUDE MODULE IDD

MODIFY TASK SSC

MODIFY TASK SSCT

*    ONLINE MAPPING

INCLUDE MODULE ONLINE-MAPPING
INCLUDE MODULE ONLINE-QUERY

**************************************************************************
** TASK AND PROGRAM STATEMENTS FOR SYSTEM TASKS                       **
** AND THE SYSTEM GENERATION COMPILER                                   **
**************************************************************************

COPY TASKS FROM SYSTEM 99

COPY PROGRAMS FROM SYSTEM 99

**************************************************************************
** SUBSCHEMA PROGRAM STATEMENTS                                       **
**************************************************************************

DEFAULT TEMPORARY PROGRAM

SUBSCHEMA
NOPROTECT
NODYNAMIC

* * SUBSCHEMAS FOR CORPDATA

ADD PROGRAM PERS001
ADD PROGRAM PERS002
ADD PROGRAM ACCSS001
ADD PROGRAM ACCSS002
ADD PROGRAM ACCSS003
ADD PROGRAM ADVSS001
ADD PROGRAM SALSS001

* * SUBSCHEMAS FOR CUSTDATA

ADD PROGRAM ADVSSC01
ADD PROGRAM SALSSC01
ADD PROGRAM SALSSC02

DEFAULT TEMPORARY PROGRAM

PROGRAM
PROTECT
DYNAMIC

**************************************************************************
** TASK AND PROGRAM STATEMENTS FOR CORPORATE APPLICATIONS             **
** PERSONNEL DEPARTMENT                                                 **
**************************************************************************

* * EMPLOYEE TRACKING

ADD TASK EMPTKRK

INVOKES PROGRAM PERP001

ADD PROGRAM PERP001
ADD PROGRAM PERP002
ADD PROGRAM PERP003
ADD PROGRAM PERP004
ADD PROGRAM PERMP01 MAP
ADD PROGRAM PERMP02 MAP
ADD PROGRAM PERMP03 MAP
* 
* SALARY TRACKING
*
ADD TASK EMPSAL
  INVOCKES PROGRAM PERP030
  NOINPUT
  ...
ADD PROGRAM PERP030
ADD PROGRAM PERP031
ADD PROGRAM PERMP30 MAP
ADD PROGRAM PERMP31 MAP
ADD PROGRAM PERMP32 MAP
************************************************************************
* * TASK AND PROGRAM STATEMENTS FOR CORPORATE APPLICATIONS * *
* * ACCOUNTING DEPARTMENT * *
* *
************************************************************************
* *
* BUDGET PLANNING AND IMPLEMENTATION *
*
ADD TASK BUDGET
  INVOCKES PROGRAM ACCP001
ADD PROGRAM ACCP001
ADD PROGRAM ACCP002
ADD PROGRAM ACCP003
ADD PROGRAM ACCMP01 MAP
ADD PROGRAM ACCMP02 MAP
************************************************************************
* * TASK AND PROGRAM STATEMENTS FOR CORPORATE APPLICATIONS * *
* * ADVERTISING DEPARTMENT * *
* *
************************************************************************
* *
* CAMPAIGN TRACKING *
*
ADD TASK ADVTRK
  INVOCKES PROGRAM ADVP001
ADD PROGRAM ADVP001
ADD PROGRAM ADVP002
ADD PROGRAM ADVP003
ADD PROGRAM ADVMP01
   MAP
ADD PROGRAM ADVMP02
   MAP

************************************************************************
* TASK AND PROGRAM STATEMENTS FOR CORPORATE APPLICATIONS
* SALES AND MARKETING DEPARTMENT
************************************************************************
* CUSTOMER TRACKING
* ADD TASK CUSTRK
   INVOKES PROGRAM SALP001
   ADD PROGRAM SALP021
   ADD PROGRAM SALP022
   ADD PROGRAM SALP023
   ADD PROGRAM SALMP21
      MAP
************************************************************************
* UTILITY PROGRAMS FOR CORPORATE APPLICATIONS
************************************************************************
* ADD PROGRAM COMPQTR
* ADD PROGRAM CHEKCUST
* ADD PROGRAM CUSTLOC
   TABLE
   ADD PROGRAM EXP001
      LANGUAGE ASSEMBLER
      REENTRANT
   ADD PROGRAM EXP002
      LANGUAGE ASSEMBLER
      REENTRANT
   ADD PROGRAM EXP003
      LANGUAGE ASSEMBLER
      REENTRANT
************************************************************************
* TELEPROCESSING NETWORK FOR SYSTEM 12
************************************************************************
* OPERATOR'S CONSOLE
* ADD LINE CONSOLE
   TYPE IS CONSOLE
ADD PTERM OPERATOR
   TYPE IS OPERATOR

ADD LTERM CONSOLE
   PRIORITY IS 240
   PTERM IS OPERATOR

* * * VTAM LINE
* * *
ADD LINE VTAM12
   COMPACT
   TYPE IS VTAMLIN
   APPLICATION ID IS SYSTEM12
   RPL COUNT IS 10

ADD PTERM PV12001
   TYPE IS V3277

ADD LTERM LT12001

ADD PTERM PV12002
   TYPE IS V3277

ADD LTERM LT12002

ADD PTERM PV12003
   TYPE IS V3277

ADD LTERM LT12003

ADD PTERM PV12004
   TYPE IS V3277

ADD LTERM LT12004

ADD PTERM PV12005
   TYPE IS V3277

ADD LTERM LT12005

ADD PTERM PV12006
   TYPE IS V3277

ADD LTERM LT12006

ADD PTERM PV12007
   TYPE IS V3277

ADD LTERM LT12007

ADD PTERM PV12008
   TYPE IS V3277

ADD LTERM LT12008

ADD PTERM PV12009
   TYPE IS V3277

ADD LTERM LT12009

ADD PTERM PV12010
   TYPE IS V3277

ADD LTERM LT12010

ADD PTERM PV12011
   TYPE IS V3277

ADD LTERM LT12011
ADD PTERM PV12012
   TYPE IS V3277
ADD LTERM LT12012
ADD PTERM PV12013
   TYPE IS V3277
ADD LTERM LT12013
ADD PTERM PV12014
   TYPE IS V3277
ADD LTERM LT12014
ADD PTERM PV12015
   TYPE IS V3277
ADD LTERM LT12015
ADD PTERM PV12016
   TYPE IS V3277
ADD LTERM LT12016
ADD PTERM PV12017
   TYPE IS V3277
ADD LTERM LT12017
ADD PTERM PV12018
   TYPE IS V3277
ADD LTERM LT12018
ADD PTERM PV12019
   TYPE IS V3277
ADD LTERM LT12019
ADD PTERM PV12020
   TYPE IS V3277
ADD LTERM LT12020
ADD PTERM PV12021
   TYPE IS V3277
   NAME IS CT112001
ADD LTERM LT12021
ADD PTERM PV12022
   TYPE IS V3277
   NAME IS CT084020
ADD LTERM LT12022
ADD PTERM PV12023
   TYPE IS V3277
   NAME IS CT084122
ADD LTERM LT12023
ADD PTERM PV12024
   TYPE IS V3277
   NAME IS CT064100
ADD LTERM LT12024
ADD PTERM PV12025
TYPE IS V3277
NAME IS CT112124

ADD LTERM LT12025

ADD PTERM PV12026
TYPE IS V3277
NAME IS CT084102

ADD LTERM LT12026

ADD PTERM PRT12001
TYPE IS V3286
ACQUIRE
NAME IS FT064007
FORMFEED

ADD LTERM PRT12001
PRINTER NOBANNER CLASS IS (8)

ADD PTERM PRT12002
TYPE IS V3286
ACQUIRE
NAME IS CT116015
FORMFEED

ADD LTERM PRT12002
PRINTER NOBANNER CLASS IS (10)

ADD PTERM PRT12003
TYPE IS V3286
ACQUIRE
NAME IS FT068007
FORMFEED

ADD LTERM PRT12003
PRINTER CLASS IS (25)

*     UCF LINE
*     UCF LINE
ADD LINE UCFLINE
TYPE IS UCFLINE
MODULE IS RHDCFSTB

ADD PTERM UCFTP01
TYPE IS UCFTERM
ADD LTERM UCFLT01

ADD PTERM UCFTP02
TYPE IS UCFTERM
ADD LTERM UCFLT02

ADD PTERM UCFTP03
TYPE IS UCFTERM
ADD LTERM UCFLT03

ADD PTERM UCFTP04
TYPE IS UCFTERM
ADD LTERM UCFLT04

ADD PTERM UCFTP05
TYPE IS UCFTERM
NAME IS CLASS33
ADD LTERM UCFLT05
PRINTER CLASS IS (33 35)
Tailoring the Banner Page

This section describes how to tailor the banner page on CA IDMS reports.

Specifying a Banner Page

Reports printed by CA IDMS optionally can begin with a banner page. You use the BANNER /NOBANNER parameter of the system generation LTERM statement to specify whether reports printed on a given printer have a banner page. The LTERM statement is described in LTERM Statement.

RHDCBANR Module

The text of the banner page is determined by the RHDCBANR module. The print program calls RHDCBANR repeatedly to obtain each line of the banner page.
You can tailor the banner page to meet site-specific needs. To do this, you modify the RHDCBANR source module. Then you assemble the source module and link edit the resulting module into the load (core-image) library.

To assemble and link-edit RHDCBANR, use the sample job control that is provided in the CAGJSAMP library. The RHDCBANR source is located in the CAGJSRC library.

How the Banner Page is Built

When the print program calls RHDCBANR:

- **Register 1** points to the logical terminal element (LTE) of the printer on which the report is to be printed.
- **LTEQBUFA** contains the address of a 180-byte buffer.
- **LTERPEA** contains the address of the report element (RPE) for the report being printed.
- **RPEBCTR** contains the number of the banner page line that RHDCBANR is to supply. For each report, the print program sets RPEBCTR to zero before the first call to RHDCBANR.

RHDCBANR must build the next line of the banner page in the buffer identified by LTEQBUFA. Each banner page line should contain only the data to be printed. RHDCBANR should not place device-dependent line control characters in the buffer.

Additionally, RHDCBANR must set:

- **LTEPRLEN** to the length of the banner page line.
- **LTEPROPT** to X'00' for normal line spacing or to X'01' to cause a page eject before the line is printed.
- **RPEBCTR** to the number of the banner page line to be built the next time the print program calls RHDCBANR. If no more banner page lines are to be printed, RPEBCTR should be set to 255.

TCAM Considerations

This section describes considerations that apply to DC sessions initiated from terminals on TCAM lines (that is, lines defined as TYPE IS TCAMLIN).

Initial Task Execution

You can define a DC/UCF task to be executed whenever a user initiates a DC/UCF session on a TCAM line. You can use the task to invoke a program such as the signon program (RHDCSNON), CA ADS runtime system (ADSORUN1), or a user defined map.

The DC/UCF task code must match the TCAM Message Control Program (MCP) application id that the user enters to initiate the DC/UCF session. Additionally, the DC/UCF task must be defined with the INPUT parameter.
For example, assume the TCAM MCP application id used to initiate a DC/UCF session is IDMS. If you include the following TASK statement in the DC/UCF system definition, the signon program is invoked automatically whenever a user initiates a session on a TCAM line:

```
ADD TASK IDMS
    INVOKES PROGRAM RHDCSNON
    INPUT
```

**Disconnect Messages**

DC/UCF issues either of two messages when it wants to terminate a session on a TCAM line:

- **DC080101 LOGOFF - DISCONNECTED FROM IDMS-DC**
  
  IDMS-DC/UCF issues this message when:
  
  - The terminal operator executes the BYE task.
  - The resource timeout interval for the terminal expires.
  - A user at another terminal issues a DCMT VARY PTERM DISCONNECT or VARY PTERM OFFLINE command for the terminal.

- **DC080102 LOGOFF - NO AVAILABLE IDMS-DC PTERMS**
  
  DC/UCF issues this message when a user attempts to initiate a session but all of the TCAM physical terminals are either in use, reserved for other terminals, or offline.

DC/UCF sends the message to the terminal from which the session was initiated. In the case of message DC080101, DC/UCF also disconnects the terminal from the physical terminal element (PTE).

You should write the TCAM MCP to scan for messages DC080101 and DC080102 in the output message handler. The TCAM MCP should terminate the DC/UCF session when the message handler detects either message.

---

**IDMSLBLS Procedure for z/VSE JCL**

This section lists the IDMSLBLS procedure referenced in z/VSE JCL in this document.

**What is the IDMSLBLS Procedure**

IDMSLBLS is a procedure provided during a CA IDMS z/VSE installation. It contains file definitions for the CA IDMS components listed below. These components are provided during installation:

- Dictionaries
- Sample databases
- Disk journal files
- SYSIDMS file
Tailor the IDMSLBLS procedure to reflect the file names and definitions in use at your site and include this procedure in z/VSE JCL job streams.

The sample z/VSE JCL provided in this document includes the IDMS.LBLS procedure. Therefore, individual file definitions for CA IDMS dictionaries, sample databases, disk journal files, and SYSIDMS file are not included in the sample JCL.

**IDMSLBLS Procedure Listing**

```plaintext
* LIBDEFS
  // LIBDEF *,SEARCH=idslib.sublib/* LIBDEF *,CATALOG=user.sublib/* LABELS
  // DLBL idmslib,'ids.libname',2099/365
  // EXTENT nnnnnn,,ssss,1500
  // DLBL dccat,'ids. system.dccat',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,31
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dccatl,'ids. system.dccatiod',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dccatx,'ids. system.dccatx',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcdml,'ids. system.dcdml',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,101
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcdcl,'ids. system.dcdclod',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,21
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcdcm,'ids. system.dcdcm',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,68
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcdsc,'ids. system.dcdsc',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,135
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcdmsg,'ids. sysmsg. dcdmsg',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,201
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcdscr,'ids. sysloc. dcdscr',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrex,'ids. sycsec. dcrex',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,6
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
  // DLBL dcrem,'ids. sycsec. dcrem',2099/365,DA
  // EXTENT SYSnnn,nnnnnn,,ssss,11
  // ASSGN SYSnnn,DISK, VOL=nnnnnnn, SHR
```
```plaintext
// EXTENT  SYSnnn, nnnnnn, ssss, 6
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    sysctl, 'idms.sysctl', 2099/365, SD
// EXTENT  SYSnnn, nnnnnn, ssss, 2
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    secadd, 'idms.sysuser.dlsec', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 26
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    dictdb, 'idms.appldict.dldml', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 51
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    dloddb, 'idms.appldict.dlldclod', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 201
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    sqldd, 'idms.syssql.ddlcat', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 101
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    sqllod, 'idms.syssql.ddlcatl', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 201
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    sqlwdx, 'idms.syssql.ddlcatx', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 26
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    asfdml, 'idms.asfdict.dlldml', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 201
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    asfldl, 'idms.asfdict.asflod', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 401
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    asfdata, 'idms.asfdict.asfdata', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 201
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    ASFDEFN, 'idms.asfdict.asfdefn', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 101
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    jjjrnl, 'idms.jjrnl', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 54
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    j2jrn1, 'idms.j2jrn1', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 54
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    j3jrn1, 'idms.j3jrn1', 2099/365, DA
// EXTENT  SYSnnn, nnnnnn, ssss, 54
// ASSGN   SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL    SYSIDMS, '#SYSIPT', 0, SD
/+ */
```

<p>| <code>&lt;idmslib.sublib&gt;</code> | Name of the sublibrary within the library containing CA IDMS modules |
| <code>&lt;user.sublib&gt;</code> | Name of the sublibrary within the library containing user modules |
| <code>&lt;idmslib&gt;</code> | Filename of the file containing CA IDMS modules |
| <code>&lt;idms.library&gt;</code> | File-ID associated with the file containing CA IDMS modules |
| <code>&lt;SYSnnn&gt;</code> | Logical unit of the volume for which the extent is effective |
| <code>&lt;nnnnnn&gt;</code> | Volume serial identifier of appropriate disk volume |
| <code>&lt;ssss&gt;</code> | Starting track (CKD) or block (FBA) of disk extent |
| <code>&lt;dccat&gt;</code> | Filename of the system dictionary catalog (DDLCAT) area |
| <code>&lt;idms.system.dccat&gt;</code> | File-ID of the system dictionary catalog (DDLCAT) area |
| <code>&lt;dccatlod&gt;</code> | Filename of the system dictionary catalog load (DDLCATLOD) area |
| <code>&lt;idms.system.dccatlod&gt;</code> | File-ID of the system dictionary catalog load (DDLCATLOD) area |
| <code>&lt;dccatx&gt;</code> | Filename of the system dictionary catalog index (DDLCATX) area |</p>
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.system.dccatx</td>
<td>File-ID of the system dictionary catalog index (DDLCATX) area</td>
</tr>
<tr>
<td>dcml</td>
<td>Filename of the system dictionary definition (DDLDML) area</td>
</tr>
<tr>
<td>idms.system.ddldml</td>
<td>File-ID of the system dictionary definition (DDLDML) area</td>
</tr>
<tr>
<td>dclod</td>
<td>Filename of the system dictionary definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td>idms.system.ddldclod</td>
<td>File-ID of the system dictionary definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td>dclog</td>
<td>Filename of the system log area (DDLDCLOG) area</td>
</tr>
<tr>
<td>idms.system.ddldclog</td>
<td>File-ID of the system log (DDLDCLOG) area</td>
</tr>
<tr>
<td>dcrun</td>
<td>Filename of the system queue (DDLDCRUN) area</td>
</tr>
<tr>
<td>idms.system.ddldcrun</td>
<td>File-ID of the system queue (DDLDCRUN) area</td>
</tr>
<tr>
<td>dcscr</td>
<td>Filename of the system scratch (DDLDCSCR) area</td>
</tr>
<tr>
<td>idms.system.ddldcscr</td>
<td>File-ID of the system scratch (DDLDCSCR) area</td>
</tr>
<tr>
<td>dcmmsg</td>
<td>Filename of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td>idms.sysmsg.ddldcmmsg</td>
<td>File-ID of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td>dclscr</td>
<td>Filename of the local mode system scratch (DDLDCSCR) area</td>
</tr>
<tr>
<td>idms.sysloc.ddlocscr</td>
<td>File-ID of the local mode system scratch (DDLDCSCR) area</td>
</tr>
<tr>
<td>dirldb</td>
<td>Filename of the IDMSDIRL definition (DDLDML) area</td>
</tr>
<tr>
<td>idms.sysdirl.ddldml</td>
<td>File-ID of the IDMSDIRL definition (DDLDML) area</td>
</tr>
<tr>
<td>dirllod</td>
<td>Filename of the IDMSDIRL definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td>idms.sysdirl.dirllod</td>
<td>File-ID of the IDMSDIRL definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td>empdemo</td>
<td>Filename of the EMPDEMO area</td>
</tr>
<tr>
<td>idms.empdemo1</td>
<td>File-ID of the EMPDEMO area</td>
</tr>
<tr>
<td>insdemo</td>
<td>Filename of the INSDEMO area</td>
</tr>
<tr>
<td>idms.insdemo1</td>
<td>File-ID of the INSDEMO area</td>
</tr>
<tr>
<td>orgdemo</td>
<td>Filename of the ORGDEMO area</td>
</tr>
<tr>
<td>idms.orgdemo1</td>
<td>File-ID of the ORGDEMO area</td>
</tr>
<tr>
<td>empldem</td>
<td>Filename of the EMPLDEMO area</td>
</tr>
<tr>
<td>idms.sqldemo.empldemo</td>
<td>File-ID of the EMPLDEMO area</td>
</tr>
<tr>
<td>infodem</td>
<td>Filename of the INFODEMO area</td>
</tr>
<tr>
<td>idms.sqldemo.infode</td>
<td>File-ID of the INFODEMO area</td>
</tr>
<tr>
<td>projdem</td>
<td>Filename of the PROJDEMO area</td>
</tr>
<tr>
<td>idms.projseg.projdemo</td>
<td>File-ID of the PROJDEMO area</td>
</tr>
<tr>
<td>indxdem</td>
<td>Filename of the INDXDEMO area</td>
</tr>
<tr>
<td>idms.sqldemo indxdemo</td>
<td>File-ID of the INDXDEMO area</td>
</tr>
<tr>
<td>sysctl</td>
<td>Filename of the SYSCTL file</td>
</tr>
<tr>
<td>idms.sysctl</td>
<td>File-ID of the SYSCTL file</td>
</tr>
<tr>
<td>secdd</td>
<td>Filename of the system user catalog (DDLSEC) area</td>
</tr>
<tr>
<td>File Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>idms.sysuser.ddlsec</td>
<td>File-ID of the system user catalog (DDLSEC) area</td>
</tr>
<tr>
<td>dictdb</td>
<td>Filename of the application dictionary definition area</td>
</tr>
<tr>
<td>idms.appldict.ddldml</td>
<td>File-ID of the application dictionary definition (DDLDML) area</td>
</tr>
<tr>
<td>dloddb</td>
<td>Filename of the application dictionary definition load area</td>
</tr>
<tr>
<td>idms.appldict.ddldclod</td>
<td>File-ID of the application dictionary definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td>sqld</td>
<td>Filename of the SQL catalog (DDLCAT) area</td>
</tr>
<tr>
<td>idms.syssql.ddlcat</td>
<td>File-ID of the SQL catalog (DDLCAT) area</td>
</tr>
<tr>
<td>sqlld</td>
<td>Filename of the SQL catalog load (DDLCLATL) area</td>
</tr>
<tr>
<td>idms.syssql.ddlcatl</td>
<td>File-ID of SQL catalog load (DDLCLATL) area</td>
</tr>
<tr>
<td>sqlxld</td>
<td>Filename of the SQL catalog index (DDLCATX) area</td>
</tr>
<tr>
<td>idms.syssql.ddlcatx</td>
<td>File-ID of the SQL catalog index (DDLCATX) area</td>
</tr>
<tr>
<td>asfdml</td>
<td>Filename of the ASF dictionary definition (DDLDML) area</td>
</tr>
<tr>
<td>idms.asfdict.ddldml</td>
<td>File-ID of the ASF dictionary definition (DDLDML) area</td>
</tr>
<tr>
<td>asflod</td>
<td>Filename of the ASF dictionary definition load (ASFLOD) area</td>
</tr>
<tr>
<td>idms.asfdict.asflod</td>
<td>File-ID of the ASF dictionary definition load (ASFLOD) area</td>
</tr>
<tr>
<td>asfdata</td>
<td>Filename of the ASF data (ASFDATA) area</td>
</tr>
<tr>
<td>idms.asfdict.asfdata</td>
<td>File-ID of the ASF data area (ASFDATA) area</td>
</tr>
<tr>
<td>ASFDEFN</td>
<td>Filename of the ASF data definition (ASFDEFN) area</td>
</tr>
<tr>
<td>idms.asfdict.asfdefn</td>
<td>File-ID of the ASF data definition area (ASFDEFN) area</td>
</tr>
<tr>
<td>j1jrnl</td>
<td>Filename of the first disk journal file</td>
</tr>
<tr>
<td>idms.j1jrnl</td>
<td>File-ID of the first disk journal file</td>
</tr>
<tr>
<td>j2jrnl</td>
<td>Filename of the second disk journal file</td>
</tr>
<tr>
<td>idms.j2jrnl</td>
<td>File-ID of the second disk journal file</td>
</tr>
<tr>
<td>j3jrnl</td>
<td>Filename of the third disk journal file</td>
</tr>
<tr>
<td>idms.j3jrnl</td>
<td>File-ID of the third disk journal file</td>
</tr>
<tr>
<td>SYSIDMS</td>
<td>Filename of the SYSIDMS parameter file</td>
</tr>
</tbody>
</table>

**IDMSLBLS Procedure**

**What is the IDMSLBLS procedure**

IDMSLBLS is a procedure provided during a CA IDMS z/VSE installation. It contains file definitions for the CA IDMS components listed below. These components are provided during installation:

- Dictionaries
- Sample databases
- Disk journal files
**SYSIDMS file**

Tailor the IDMSLBLS procedure to reflect the filenames and definitions in use at your site and include this procedure in z/VSE JCL job streams.

The sample z/VSE JCL provided in this document includes the IDMSLBLS procedure. Therefore, individual file definitions for CA IDMS dictionaries, sample databases, disk journal files, and SYSIDMS file are not included in the sample JCL.

**IDMSLBLS procedure listing**

```jcl
* ---------- LIBDEFS ----------
// LIBDEF *,SEARCH=idmslib.sublib
// LIBDEF *,CATALOG=user.sublib
/* ------------------------- LABELS -------------------------
// DLBL idmslib,'idms.library',2099/365
// EXTENT ,nnnnnn,,ssss,1500
// DLBL dccat,'idms.system.dccat',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,31
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dccatl,'idms.system.dccatlod',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dccatx,'idms.system.dccatx',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,11
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dcdml,'idms.system.ddldml',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,101
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dclod,'idms.system.ddldclod',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,21
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dclog,'idms.system.ddldclog',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,401
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dcmgr,'idms.system.ddldemgr',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,68
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dcmpl,'idms.system.ddldempl',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,135
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dirldb,'idms.sysdirl.ddldml',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,201
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dirllod,'idms.sysdirl.ddldclod',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,2
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL empdemo,'idms.empdemo1',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,11
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL insdemo,'idms.insdemo1',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL orgdemo,'idms.orgdemo1',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL emplogdem,'idms.sqldemo.emplogdem',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,11
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL infodem,'idms.sqldemo.infodemo',2099/365,DA
// EXTENT SYSnnn,nnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
```
// DLBL projdem,'idms.projseg.projdemo',2099/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL indxdem,'idms.sqldemo.indxdemo',2099/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL sysctl,'idms.sysctl',2899/365,SD
// EXTENT SYSnnn,ssss,ssss,ssss,2
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL secdd,'idms.sysuser.ddlsec',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,2
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dictdb,'idms.appldict.ddldml',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,51
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dloddb,'idms.appldict.ddldclod',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,51
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL sqldd,'idms.syssql.ddlcat',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,101
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL sqllod,'idms.syssql.ddlcatl',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,51
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL sqlxdd,'idms.syssql.ddldcatx',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,26
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL asfdml,'idms.asfdict.ddldml',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,51
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL asflod,'idms.asfdict.asflod',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,201
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL asfdata,'idms.asfdict.asfdata',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,201
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL ASFDEFN,'idms.asfdict.asfdefn',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,101
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL j1jrnl,'idms.j1jrnl',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,54
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL j2jrnl,'idms.j2jrnl',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,54
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL j3jrnl,'idms.j3jrnl',2899/365,DA
// EXTENT SYSnnn,ssss,ssss,ssss,54
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL SYSIDMS,'#SYSIPT',0,SD

idmslib.sublib Name of the sublibrary within the library containing CA IDMS modules
user.sublib Name of the sublibrary within the library containing user modules

idmslib Filename of the file containing CA IDMS modules
idms.library File-ID associated with the file containing CA IDMS modules
SYSnnn Logical unit of the volume for which the extent is effective
nnnnnn Volume serial identifier of appropriate disk volume
ssss Starting track (CKD) or block (FBA) of disk extent
dccat Filename of the system dictionary catalog (DDLCAT) area
ids.system.dccat File-ID of the system dictionary catalog (DDLCAT) area
dccatl Filename of the system dictionary catalog load (DDLCATLOD) area
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>idms.system.dccatlod</code></td>
<td>File-ID of the system dictionary catalog load (DDLCATLOD) area</td>
</tr>
<tr>
<td><code>dccatx</code></td>
<td>Filename of the system dictionary catalog index (DDLCATX) area</td>
</tr>
<tr>
<td><code>idms.system.dccatx</code></td>
<td>File-ID of the system dictionary catalog index (DDLCATX) area</td>
</tr>
<tr>
<td><code>dcdml</code></td>
<td>Filename of the system dictionary definition (DDLDML) area</td>
</tr>
<tr>
<td><code>idms.system.ddldml</code></td>
<td>File-ID of the system dictionary definition (DDLDML) area</td>
</tr>
<tr>
<td><code>dclod</code></td>
<td>Filename of the system dictionary definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td><code>idms.system.ddldclod</code></td>
<td>File-ID of the system dictionary definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td><code>dclog</code></td>
<td>Filename of the system log area (DDLDCLOG) area</td>
</tr>
<tr>
<td><code>idms.system.ddldclog</code></td>
<td>File-ID of the system log (DDLDCLLOG) area</td>
</tr>
<tr>
<td><code>dcrun</code></td>
<td>Filename of the system queue (DDLDCRUN) area</td>
</tr>
<tr>
<td><code>idms.system.ddldcrun</code></td>
<td>File-ID of the system queue (DDLDCRUN) area</td>
</tr>
<tr>
<td><code>dcscr</code></td>
<td>Filename of the system scratch (DDLDCSCR) area</td>
</tr>
<tr>
<td><code>idms.system.ddldcscr</code></td>
<td>File-ID of the system scratch (DDLDCSCR) area</td>
</tr>
<tr>
<td><code>dcmcg</code></td>
<td>Filename of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td><code>idms.sysmsg.ddldcmsg</code></td>
<td>File-ID of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td><code>dclocr</code></td>
<td>Filename of the local mode system scratch (DDLOCSCR) area</td>
</tr>
<tr>
<td><code>idms.sysloc.ddlococr</code></td>
<td>File-ID of the local mode system scratch (DDLOCSCR) area</td>
</tr>
<tr>
<td><code>dirldb</code></td>
<td>Filename of the IDMSDIRL definition (DDLDML) area</td>
</tr>
<tr>
<td><code>idms.sysdir.ddldml</code></td>
<td>File-ID of the IDMSDIRL definition (DDLDML) area</td>
</tr>
<tr>
<td><code>dirlod</code></td>
<td>Filename of the IDMSDIRL definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td><code>idms.sysdir.dirllod</code></td>
<td>File-ID of the IDMSDIRL definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td><code>emple demo</code></td>
<td>Filename of the EMPDEMO area</td>
</tr>
<tr>
<td><code>idms.empdemo1</code></td>
<td>File-ID of the EMPDEMO area</td>
</tr>
<tr>
<td><code>insdemo</code></td>
<td>Filename of the INSDEMO area</td>
</tr>
<tr>
<td><code>idms.insdemo1</code></td>
<td>File-ID of the INSDEMO area</td>
</tr>
<tr>
<td><code>orgdemo</code></td>
<td>Filename of the ORGDEMO area</td>
</tr>
<tr>
<td><code>idms.orgdemo1</code></td>
<td>File-ID of the ORGDEMO area</td>
</tr>
<tr>
<td><code>empldem</code></td>
<td>Filename of the EMPLDEMO area</td>
</tr>
<tr>
<td><code>idms.sqldemo.empldemo</code></td>
<td>File-ID of the EMPLDEMO area</td>
</tr>
<tr>
<td><code>infodem</code></td>
<td>Filename of the INFODEMO area</td>
</tr>
<tr>
<td><code>idms.sqldemo.infodem</code></td>
<td>File-ID of the INFODEMO area</td>
</tr>
<tr>
<td><code>projdem</code></td>
<td>Filename of the PROJDEMO area</td>
</tr>
<tr>
<td><code>idms.projseg.projdemo</code></td>
<td>File-ID of the PROJDEMO area</td>
</tr>
<tr>
<td><code>indxdem</code></td>
<td>Filename of the INDXDEMO area</td>
</tr>
<tr>
<td><code>idms.sqldemo.indxdem</code></td>
<td>File-ID of the INDXDEMO area</td>
</tr>
<tr>
<td><code>sysctl</code></td>
<td>Filename of the SYSCTL file</td>
</tr>
</tbody>
</table>
SYSGEN User Exit Program

The SYSGEN user-exit module is called by the system generation compiler when it encounters any of these four points:
When a User Exit is Called

- **SIGNON/SIGNOFF/COMMIT**
  After the signon procedure is complete and the compiler’s security checks have been passed, or immediately after signoff or COMMIT processing.

- **Major command**
  After an ADD, MODIFY, DELETE, DISPLAY or PUNCH request has been issued. The program is invoked just after the system generation compiler has identified the entity that is the object of the request and has successfully checked authorization requirements. Object entities can be any standard system generation entity type.

- **Card image**
  After each input statement (card image) is passed to the user-exit control block after the statement has been:
  - Scanned and printed on the SYSGEN Compiler Activity List
  - Displayed at the terminal
  - Written to the print file (online SYSGEN compiler interface only)

  The administrator can build an audit trail of accesses and updates to the dictionary.

- **End of converse**
  When one of the following occurs, the user can perform a termination activity, such as a write-to-log:
  - The user presses ENTER during an online SYSGEN compiler session
  - A batch run of the SYSGEN compiler processes a SIGNOFF statement
  - A batch run of the SYSGEN compiler detects an end-of-file condition.

Rules for Writing the User-exit Program

This section describes the rules that apply to writing the user-exit program.

- **Language**
  You can write the user-exit module in any language that supports OS calling conventions. However, it is recommended that you write user-exit modules in Assembler to allow the online SYSGEN compiler to remain reentrant.

  **Note:** User-exit modules cannot be CA ADS dialogs.
CA IDMS - 19.0

- **Versions**
  You can code and maintain separate versions of user-exit modules for the batch and online SYSGEN compilers, or you can code modules that can be executed both in batch mode and online.

- **Macros**
  The user-exit facility supports all CA IDMS/DB macros for exits to be used with the online SYSGEN compiler. For exits to be used with the batch SYSGEN compiler, the only CA IDMS/DB macros supported are: #WTL, #ABEND, #GETSTG, #FREESTG, #LOAD, and #DELETE; under DOS/VS(E), the only valid form of #DELETE is EPADDR=.

- **Run units**
  You can start a run unit within an exit, however you should ensure that the run unit does not deadlock with the SYSGEN compiler run unit. If a user-exit run unit accesses a dictionary area, the run unit should ready the object area in a retrieval usage mode.

- **Entry point**
  The entry point of the user exit invoked by the batch and online compilers differ.

<table>
<thead>
<tr>
<th>Compiler Name</th>
<th>Description</th>
<th>User Exit Entry Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHDCSGEN</td>
<td>Batch system generation compiler</td>
<td>SGNEXITB</td>
</tr>
<tr>
<td>RHDCSGDGC</td>
<td>Online system generation compiler</td>
<td>SGNEXITO</td>
</tr>
</tbody>
</table>

Although each exit has a unique entry point name, you can use the same exit code for more than one compiler by assigning multiple entry point names to the same set of code.

- **Enabling a compiler exit**
  To enable a user exit for the system generation compiler, link your exit module with IDMSUXIT.

  **Note:** For more information on how to enable user exits by linking them with IDMSUXIT, refer to the "User Exits" section in the CA IDMS Systems Operations section.

- **Interface**
  User exits written in COBOL to run under the online SYSGEN compiler require a user-exit interface, written in Assembler with an entry point of SGNEXITO, to be link edited with IDMSUXIT. This interface should issue a #LINK to the COBOL program (with an entry point other than RHDCSGDGC) to isolate it from RHDCSGDGC, which is storage protected.

- **Register conventions**
  User-exit modules are called using the following OS register conventions:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R15</td>
<td>Entry point of user exit module</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R13</td>
<td>18 fullword SAVEAREA</td>
</tr>
<tr>
<td>R1</td>
<td>Fullword parameter list</td>
</tr>
</tbody>
</table>

- **Parameters 3 and 4**
  For all four types of user exits, parameter 1 points to a user-exit control block and parameter 2 points to a SIGNON element block. The information addressed in parameters 3 and 4 varies based on the type of user exit, as follows:
For the SIGNON/SIGNOFF/COMMIT and end-of-conversation exits, parameter 3 points to a SIGNON block.

For the major command user exit, parameter 3 points to an entity control block.

For the card-image user exit, parameter 3 points to a card-image control block.

For all user exits except the card-image user exit, parameter 4 is reserved for use by the SYSGEN compiler and should be defined as a PIC X(80) field in the user-exit module.

For the card-image user exit, parameter 4 points to the input card image, which is defined as a PIC X(80) field.

The user-exit control blocks are described later in this section.

Information modification
With the exception of the fields identified within the user-exit control block described below, a user-exit module should not modify any of the information passed.

Return codes
On return from a user-exit module, the user must set a return code and, optionally, specify a message ID and message text to be issued by the system generation compiler, as follows:

<table>
<thead>
<tr>
<th>Code</th>
<th>SYSGEN action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No message is issued; SYSGEN continues with normal processing.</td>
</tr>
<tr>
<td>1</td>
<td>An informational message is issued; SYSGEN continues with normal processing.</td>
</tr>
<tr>
<td>4</td>
<td>An error message is issued; SYSGEN initiates error processing.</td>
</tr>
</tbody>
</table>

Control Blocks and Sample User-exit Programs

This section presents the formats of these five control blocks:

- User-exit control block
- SIGNON element block
- SIGNON block
- Entity control block
- Card-image control block

User-exit Control Block

The following table shows how to define the user-exit control block:
### Field Usage Size Picture Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Usage</th>
<th>Size</th>
<th>Picture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Char</td>
<td>8</td>
<td>X(8)</td>
<td>Compiler name: RHDCSGEN</td>
</tr>
<tr>
<td>2</td>
<td>Char</td>
<td>8</td>
<td>X(8)</td>
<td>Compiler start date: mm/dd/yy</td>
</tr>
<tr>
<td>3</td>
<td>Char</td>
<td>8</td>
<td>X(8)</td>
<td>Compiler start time: hhmmssmm</td>
</tr>
<tr>
<td>4</td>
<td>Binar</td>
<td>4</td>
<td>S9(8)</td>
<td>COMP</td>
</tr>
<tr>
<td>5</td>
<td>Binar</td>
<td>4</td>
<td>S9(8)</td>
<td>COMP</td>
</tr>
<tr>
<td>6</td>
<td>Char</td>
<td>8</td>
<td>X(8)</td>
<td>Message ID returned by user, in the range DC900000 through DC999999, or any 6-digit number; blank if no message is returned</td>
</tr>
<tr>
<td>7</td>
<td>Char</td>
<td>80</td>
<td>X(80)</td>
<td>Message text returned by user</td>
</tr>
</tbody>
</table>

### SIGNON Element Block

The following table shows how to define the SIGNON element block:

<table>
<thead>
<tr>
<th>Field</th>
<th>Usage</th>
<th>Size</th>
<th>Picture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Binary</td>
<td>1</td>
<td>X</td>
<td>Length of user ID for #WTLs (not addressable by COBOL)</td>
</tr>
<tr>
<td>2</td>
<td>Char</td>
<td>32</td>
<td>X(32)</td>
<td>SIGNON user ID</td>
</tr>
</tbody>
</table>

### SIGNON Block

The following table shows how to define the SIGNON block:

<table>
<thead>
<tr>
<th>Field</th>
<th>Usage</th>
<th>Size</th>
<th>Picture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Char</td>
<td>16</td>
<td>X(16)</td>
<td>SIGNON, SIGNOFF, COMMIT or END-OF-CONVERSE statement</td>
</tr>
<tr>
<td>2</td>
<td>Char</td>
<td>8</td>
<td>X(8)</td>
<td>SIGNON dictionary name</td>
</tr>
<tr>
<td>3</td>
<td>Char</td>
<td>8</td>
<td>X(8)</td>
<td>SIGNON node name</td>
</tr>
<tr>
<td>4A</td>
<td>CHAR</td>
<td>32</td>
<td>X(32)</td>
<td>User ID</td>
</tr>
<tr>
<td>5</td>
<td>Binary</td>
<td>2</td>
<td>S9(4)</td>
<td>DDLDM area usage mode: 36=UPDATE; 38=PROTECTED UPDATE; 37=RETRIEVAL</td>
</tr>
<tr>
<td>6</td>
<td>Binary</td>
<td>2</td>
<td>S9(4)</td>
<td>DDLDCLOD area usage mode</td>
</tr>
<tr>
<td>7</td>
<td>Binary</td>
<td>2</td>
<td>S9(4)</td>
<td>DDLDCMSG area usage mode</td>
</tr>
<tr>
<td>8</td>
<td>Binary</td>
<td>10</td>
<td>X(10)</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

⚠️ **Note:** Each bit in flag 0 and flag 1 must be tested separately. More than one bit may be on at any one time.
Entity Control Block

The following table shows how to define the entity control block:

<table>
<thead>
<tr>
<th>Field</th>
<th>Usage</th>
<th>Size</th>
<th>Picture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Char</td>
<td>16</td>
<td>X(16)</td>
<td>Major command (ADD, MODIFY, DELETE, DISPLAY, or PUNCH).</td>
</tr>
<tr>
<td>2</td>
<td>Char</td>
<td>32</td>
<td>X(32)</td>
<td>Entity type</td>
</tr>
<tr>
<td>3</td>
<td>Char</td>
<td>40</td>
<td>X(40)</td>
<td>Entity occurrence</td>
</tr>
<tr>
<td>4</td>
<td>Binary</td>
<td>2</td>
<td>S9(4)</td>
<td>Entity version number or number of records requested</td>
</tr>
<tr>
<td>5</td>
<td>Char</td>
<td>64</td>
<td>X(64)</td>
<td>Additional Qualifier (not used)</td>
</tr>
<tr>
<td>6</td>
<td>Char</td>
<td>32</td>
<td>X(32)</td>
<td>PREPARED BY user ID</td>
</tr>
<tr>
<td>7</td>
<td>Char</td>
<td>32</td>
<td>X(32)</td>
<td>REVISED BY user ID</td>
</tr>
</tbody>
</table>

Card-image Control Block

The following table shows how to define the card-image control block:

<table>
<thead>
<tr>
<th>Field</th>
<th>Usage</th>
<th>Size</th>
<th>Picture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Char</td>
<td>16</td>
<td>X(16)</td>
<td>'CARD IMAGE' command</td>
</tr>
<tr>
<td>2</td>
<td>Binary</td>
<td>2</td>
<td>S9(8)</td>
<td>Input low-card column</td>
</tr>
<tr>
<td>3</td>
<td>Binary</td>
<td>2</td>
<td>S9(8)</td>
<td>Input high-card column</td>
</tr>
</tbody>
</table>

Sample User-exit Program

The following sample user-exit program can be used to enforce naming conventions for elements in the batch and online versions of the system generation compiler. The source code for this program can be found in the installation source library under member name IDDSUXIT.

********************************************************************
IDDUXIT CSECT
#REGEQU
ENTRY SGNEXITO
SGNEXITO DS 0H
********************************************************************
ENTRY SGNEXITB
SGNEXITB DS 0H Batch SYSGEN compiler entry
*********************************************************************
* SET UP ADDRESSABILITY *
*********************************************************************
STM R14,R12,12(R13) SAVE CALLERS REGISTERS
LR R12,R15
USING I/DDUXIT,R12
L R4,12(R1) GET THE
L R3,8(R1) CORRECT
L R2,4(R1) PARAMETER
L R1,0(R1) ADDRESSES
*
ID/DUXITR DS 0H BASE THE CONTROL BLOCKS
*
USING UXITCB,R1 USER EXIT CONTROL BLOCK
MVC UXITCDE,F0 ZERO OUT THE RETURN CODE
MVC UXITMID(8),BLANKS BLANK OUT THE MESSAGE ID
MVC UXITMTXT(80),BLANKS BLANK OUT THE MESSAGE
*
*********************************************************************
* INTERROGATE THE MAJOR COMMAND *
*********************************************************************
*
SPACE
UXIENTY EQU * ENTITY CONTROL BLOCK
*
CLC UXITEVRB,UXICON IS IT AN SIGNON?
BE USIGNON YES, CHECK THE USER NAME
*
CLC UXITEVRB,UXICARD IS IT AN CARD IMAGE EXIT?
BE UCARD YES, CHECK THE CARD
*
CLC UXITEVRB,UXICADD IS IT AN ADD?
BE UXIECHK YES, CHECK THE ENTITY-NAME
*
CLC UXITEVRB,UXICMOD IS IT A MODIFY?
BE UXIECHK YES, CHECK THE ENTITY-NAME
*
CLC UXITEVRB,UXICDEL IS IT A DELETE?
BE UXIECHK YES, CHECK THE ENTITY-NAME
*
MVC UXITMID(8),ELSEID MOVE IN 'ELSE' MESSAGE ID
MVC UXITMTXT(80),ELSEMSG MOVE IN 'ELSE' MESSAGE
B UXIEBYE
*
*********************************************************************
* CHECK THE CARD IMAGE *
*********************************************************************
*
SPACE
UCARD EQU *
*
MVC UXITMID(8),CARDID FILL IN THE MESSAGE ID
MVC UXITMTXT(80),CARDMSG FILL IN THE MESSAGE TEXT
B UXIEBYE BACK TO THE COMPILER
*
*********************************************************************
* CHECK THE USER NAME FOR ME *
*********************************************************************
*
SPACE
USIGNON EQU *
*
USING UXITSEB,R2 SIGNON ELEMENT BLOCK
USING UXITSB,R3 SIGNON BLOCK
*
CLC UXITUSER(3),WHOME IS IT ME
BE UXIEDC YES GO CHECK FOR DC NAME
*
NO, GO TO JAIL, GO DIRECTLY TO
* JAIL, DO NOT PASS GO DO NOT COLLECT $200.
USNAME EQU *
MVC UXITRCDE,F8 FILL IN THE RETURN CODE
MVC UXITMID(8),NOSNID FILL IN THE MESSAGE ID
MVC UXITMTXT(80),NOSNMSG FILL IN THE MESSAGE TEXT
B UXIEBYE BACK TO THE COMPILER
*
UXIHD EQU *
TM UXITFLG1,UXITIDC ARE WE RUNNING DC
BZ UXIEBYE NO, SKIP DC ID CHECK
*
CLC UXITUSER,UXITIUSR IS THE USER THE SAME AS DC
BE UXIEBYE YES, OK LET IT PASS
*
MVC UXITRCDE,F8 FILL IN THE RETURN CODE
MVC UXITMID(8),NODCID FILL IN THE MESSAGE ID
MVC UXITMTXT(80),NODCMG FILL IN THE MESSAGE TEXT
B UXIEBYE BACK TO THE COMPILER
*
*********************************************************************
* CHECK THE ENTITY-NAME FOR VALID NAMING CONVENTION                  *
*********************************************************************
*
SPACE UXIECHK EQU *
USING UXITECB,R3 ENTITY CONTROL BLOCK
*
CLC UXITENME(3),NAMECHK DOES THE NAME FOLLOW THE RULES?
BE UXIEBYE YES, LET THIS ONE PASS.
*
MVC UXITRCDE,F8 FILL IN THE RETURN CODE
MVC UXITMID(8),NODCID FILL IN THE MESSAGE ID
MVC UXITMTXT(80),NODCMG FILL IN THE MESSAGE TEXT
*
*********************************************************************
* RETURN BACK TO THE COMPILER                                       *
*********************************************************************
*
SPACE UXIEBYE EQU *
LM R14,R12,12(R13) RELOAD CALLER’S REGISTERS
BR R14 RETURN TO CALLER
EJECT
*********************************************************************
* CONSTANTS AND LITERALS                                           *
*********************************************************************
UXICADD DC CL16 'ADD'
UXICMOD DC CL16 'MODIFY'
UXICDEL DC CL16 'DELETE'
UXICSON DC CL16 'SIGNON'
UXICARD DC CL16 'CARD IMAGE'
NAMECHK DC CL3 'XYZ'
WHOME DC CL3 'XYZ'
WKLEN DC F'100'
NONOID DC CLB 'DC999001'
NONOMSG DC CLB 'NAMING CONVENTION VIOLATED - ACTION NOT ALLOWED'
NOSNID DC CLB 'DC999002'
NOSNMSG DC CLB 'SIGNON ERROR - USER NOT ALLOWED ACCESS'
NODCID DC CLB 'DC999003'
NODCMG DC CLB 'SIGNON ERROR - USER NAME NOT DC USER NAME'
CARDID DC CLB 'DC999004'
CARDMSG DC CLB 'MESSAGE PRODUCED BY CARD IMAGE EXIT'
ELSEID DC CLB 'DC999005'
ELSEMSG DC CLB 'MESSAGE PRODUCED BY CARD IMAGE EXIT'
BLANKS DC CLB ' '
F0 DC F'0' NORMAL RETURN CODE - NO ERRORS
F2 DC F'1' INFORMATION MESSAGE
F4 DC F'4' WARNING MESSAGE
F8 DC F'8' ERROR MESSAGE
CA IDMS - 19.0

********************************************************************
* USER EXIT CONTROL BLOCK *
********************************************************************
UXITCB DSECT
UXITCLRLR DS CL8  COMPILE NAME 'RHDCSGEN'
UXITDATE DS CL8  COMPILER START DATE MM/DD/YY
UXITTIME DS CL8  COMPILER START TIME HHMMSSMM
UXITWORK DS F  USER FULLWORD INITIALIZED TO 0
UXITRCDE DS 0F  RETURN CODE RETURNED BY USER
DS XL3  UNUSED
UXITRC DS X
UXITRC00 EQU X'00'  NORMAL RETURN CODE - NO ERRORS
UXITRC01 EQU X'01'  INFORMATION MESSAGE
UXITRC04 EQU X'04'  WARNING MESSAGE
UXITRC08 EQU X'08'  ERROR MESSAGE
UXITMID DS CL8  USER MESSAGE ID RETURNED BY USER
UXITMTXT DS CL80  USER MESSAGE TEXT RETURNED BY USER
UXITCBLN EQU -*-UXITCB  USER EXIT CONTROL BLOCK LENGTH

********************************************************************
* USER EXIT SIGNON ELEMENT BLOCK *
********************************************************************
UXITSEB DSECT
UXITIDLN DS X  LENGTH OF USERID FOR #WTL'S
UXITUSER DS CL32  USER ID
DS 0A  ROUND UP TO FULLWORD
UXITSLN EQU -*-UXITSEB  LENGTH OF SIGNON ELEMENT

********************************************************************
* USER EXIT SIGNON BLOCK *
********************************************************************
UXITSB DSECT
UXITTYPE DS CL16  VERB
UXITDICT DS CL8  DICTIONARY NAME
UXITNODE DS CL8  NODE NAME
UXITIUSR DS CL32  USER ID
UXITIPSW DS CL8  USER'S PASSWORD
UXITFLGO DS CL1  ENVIRONMENT FLAG
UXIT0DOS EQU X'80'  COMPILER RUNNING UNDER DOS
UXIT0MEN EQU X'40'  RUNNING UNDER 'MENU' MODE
UXITFLGO DS CL1  ENVIRONMENT FLAG
UXIT1CLL EQU X'80'  RUNNING UNDER INTERNAL SUBROUTINE MODE
UXITIDC EQU X'40'  COMPILER RUNNING UNDER DC
DS CL2  RESERVED FOR FUTURE FLAGS
DS CL20  RESERVED
UXITDMMLM DS H  DDLDMML USAGE MODE
* 36=UPDATE
* 37=PROTECTED UPDATE
* 38=RETRIEVAL
UXITDMMLM DS H  DDLDMML USAGE MODE
UXITMSGM DS H  DDLDCMDG USAGE MODE
DS CL10  RESERVED
UXITSELEN EQU -*-UXITSB  LENGTH OF USER EXIT SIGNON BLOCK

********************************************************************
* USER EXIT ENTITY CONTROL BLOCK *
********************************************************************
UXITECB DSECT
UXITEVRB DS CL16  VERB
UXITENTSY DS CL32  ENTITY-TYPE
UXITENAME DS CL40  ENTITY NAME
UXITEVER DS H  VERSION
UXITEQAO DS CL64  ADDITIONAL QUALIFIER
UXITPREP DS CL32  PREPARED BY USER NAME
UXITREV DS CL32  REVISED BY USER NAME
UXITELLEN EQU -*-UXITECB  LENGTH OF USER EXIT ENTITY CONTROL BLK

END