CA Ideal™ for CA Datacom®

Administration Guide

Version 14.02

technologies

CA Technologies Product References

This document references the following CA products:

- CA Datacom®/DB
- CA Datacom® CICS Services
- CA Datacom® SQL
- CA Dataquery™ for Datacom®
- CA eMail+
- CA Ideal™ for Datacom® (CA Ideal)
- CA Ideal™ for DB2
- CA Ideal™ for VSAM
- CA Librarian® for z/VSE
- CA Dynam®/T Tape Management for z/VSE
- CA CICSORT™
- CA Sort®

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

Contents

Chapter 1: Preliminary Concepts

This document explains the administrative functions the CA Ideal site administrator performs. These functions include creating and maintaining CA Ideal systems and users, maintaining the CA Ideal environment, and creating and maintaining a production environment.

This chapter contains basic information that is necessary for an understanding of what CA Ideal is and how to use it. You need to read this chapter before you begin using this guide. You should also be familiar with the basic concepts for using CA Ideal, explained in the *Working in the Environment Guide*.

CA Ideal Overview

CA Ideal is an application development system that provides an interactive environment for the development, maintenance, and execution of complete, integrated applications. A CA Ideal application consists of the resources (dataviews, panels, report definitions, programs, and subprograms) needed to perform a wide variety of online database and business applications.

A CA Ideal application is developed by defining a series of components using the following:

- Special-purpose, fill-in-the-blank panels that display online and are processed interactively for program, dataview, report, and panel definitions, and for DB2 plan definitions
- A structured, high-level language for the application procedure

CA Ideal runs in both batch and online environments under the z/OS and VSE operating systems. CA Ideal is an "Interactive Development Environment for an Application Lifecycle." Most commands will be executed online. Although some can also be issued from batch, all editing must be done online. A few commands, such as CREATE MODULE, can only be issued in batch.

CA Ideal can access data stored in a CA Datacom/DB, a DB2 database, a sequential file, or a VSAM file. Your CA Ideal environment can include CA Datacom/DB, DB2, or both (a dual database environment). The CA Ideal application model is stored and maintained in Datadictionary. For DB2 only sites, the Datadictionary is maintained in CA Datacom/AD.

Components of a CA Ideal Application

The following sections describe the components of a CA Ideal application.

Dataview Definition

A dataview is a logical view of data that lets you make requests of the data independently of the storage structure.

Dataview definitions for accessing CA Datacom/DB tables using native command access are created and maintained in the Datadictionary.

CA Ideal maintains dataview definitions for SQL access, both for CA Datacom/DB objects and DB2 objects, in the dictionary facility. The CA Datacom/DB tables, views, and synonyms are created and maintained in the Datadictionary. The DB2 tables and views are created and maintained in the DB2 catalog.

There are two types of dataview definitions for sequential files:

- CA Datacom/DB sites create and maintain modeled sequential file dataviews in Datadictionary.
- Unmodeled sequential file dataview definitions are created and maintained in the Virtual Library System (VLS) and the dictionary facility. CA Datacom/DB and DB2 sites can use them.

CA Ideal treats modeled and unmodeled dataview definitions for sequential files identically once they are cataloged.

Like dataview definitions for unmodeled sequential files, dataview definitions for VSAM files are created and maintained in the Virtual Library System (VLS) and the dictionary facility.

Program Definition

Several facilities are provided for defining a CA Ideal program:

- The Program Identification Panel lets the application developer initiate the creation of a program and provide descriptive commentary about the program.
- The Program Resources Panel specifies the authorized resources a program uses.
- The Parameter Definition Panel describes and names data used as input parameters to a program. Parameters are only defined if the program requires them.
- The Working Data Definition Panel names and describes data items that are local to each program. This facility is used only when working data items must be defined for the program.
- The Procedure Definition Language (PDL) is a high-level language that defines procedures. It includes an integrated database sub-language, facilities for modularization, structured design and development, arithmetic capability, report production, transaction and panel processing, built-in functions, and error handling.

- The Procedure Definition Panel lets you enter the PDL statements. The Procedure Definition Panel provides a powerful editor that lets the programmer work in the same editing environment across all operating systems, and provides templates that help to ensure structured code. All other panels used in CA Ideal provide some subset of these editing capabilities.
- For sites that have the CA Datacom Database SQL Option, the Environment Panel specifies SQL precompiler options for the program.

  For complete information about the program definition panels, see the *Creating Programs Guide*. For complete descriptions of PDL statements and functions, see the *Programming Reference Guide*.

Panel Definition

The CA Ideal Panel Definition Facility (PDF) provides the facilities for creating and maintaining panel definitions that transmit data between the user and the application program. After they are created, you can print, test, and edit panel layouts and definitions online for immediate use. For information about defining panels, see the *Creating Panel Definitions Guide*.

Report Definition

The CA Ideal Report Definition Facility (RDF) creates, maintains, and tests report definitions. RDF lets the CA Ideal user define online each report's layout, parameters, fields, column headings, and other options. For information about defining report layouts, see the *Generating Reports Guide*.

Plan or Package Definition (DB2 Only)

CA Ideal provides a means for creating, maintaining, and binding plans for DB2 applications. See the chapter “Preparing DB2 Application Plans.”

Entities and Version Management

CA Ideal uses a number of special terms to describe the organization of the data the application developer uses during the creation of an application. The following sections explain these terms, and their relationship to CA Ideal.

Entity

An entity is a set of rules that governs the data structures that CA Ideal uses.

Entity Type

An entity type is the classification of an entity according to its function. CA Ideal uses six entity types: system entities, user entities, dataview entities, program entities, panel entities, and report entities.

Entity Occurrence

An entity occurrence is a uniquely identified instance of a particular entity type.

Definition

In CA Ideal, each entity occurrence has a corresponding definition. There are six types of definitions that correspond to the six entity types: system definitions, user definitions, dataview definitions, program definitions, panel definitions, and report definitions.

Note: In the remainder of this document, the CA Ideal term definition is used in place of the term entity occurrence.

Name

Each definition must have a name to distinguish it from other definitions of the same type. The application developer assigns a name to a definition when the definition is created. For more information about naming conventions, see the *Programming Reference Guide*.

Version

For all entity types except modeled and SQL dataviews

Each named definition of a given type can exist in one or more forms, called versions, each of which is identified by a number that CA Ideal assigns. There can be as many as 999 versions with the same name. CA Ideal assigns numbers to versions sequentially as they are created, starting with number 1. The application developer cannot modify version numbers.

For CA Datacom/DB CBS dataviews and sequential file dataviews modeled in the dictionary

CA Ideal assigns numbers to test versions separately from production and history versions. Test versions are identified as T1 through T999. Production and history versions are identified as 1 through 999.

For SQL dataviews

Only one version is assigned; version 1 in production status. CA Ideal generally does not display this version number or status, or require the user to specify it.

Editing a definition has no effect on the version number. No matter how many changes are made, the version number remains the same. You can only create new versions with the same name by using the DUPLICATE...NEXT VERSION command. This command makes a copy of an existing version. CA Ideal assigns each new version of a definition the next higher number than the highest previously assigned number. When a version is deleted from the system, unless it was the highest, its number is never reassigned to another version with the same name.

Each definition is uniquely identified by the combination of its type, name, and version number. You can always reference it by this combination.

There are two cases when you can reference a particular definition without using the version number:

- You can reference the production status version of a definition by replacing the version number with the term PROD (since there can be only one production version at a time).
- For any type of definition except for an SQL dataview definition, you can reference the most recently created version of a definition by replacing the version number with the term LAST.

The version clause is optional when specifying PROD or LAST. For example, if the production version of a report definition named SALARIES is version 5, you can reference it as REPORT SALARIES VERSION 5 or as REPORT SALARIES VERSION PROD. If there are seven versions of a program named UPDATE (numbered 1 through 7), you can reference the most recently created version as PROGRAM UPDATE VERSION 7 or as PROGRAM UPDATE VERSION LAST.

Status

Each version is assigned to a category that is based on the stage it reached in the production process. This category is called the status of the version. A version can be in test status, production status, or history status. Except for dataview entity types, you can change the status of a version with the MARK STATUS command.

The following is a list of status types and the rules that apply to each.

Test

All entity types except modeled and SQL dataviews-When a version is created in CA Ideal by the CREATE or DUPLICATE command, the version is in test status. You can modify a version as long as it remains in test status. There can be many versions in test status at a time. If you edit a dataview, you must recatalog it before you can use it.

CA Datacom/DB CBS dataviews and sequential file dataviews modeled in the dictionary-Test versions are created and maintained in the dictionary. There can be up to 999 test versions at a time.

For SQL-Only one version is assigned; version 1 in production status.

Production

When a version was created, edited, and tested, and is ready for use in an application, it is marked to production and becomes the production version. Only one version of a definition can be in production status at a time. A production application must consist of components that are also in production status. This serves to protect its integrity.

- **For all entity types except modeled and SQL dataviews**-You cannot edit or delete the production version. If a production version of a program is compiled, a compilation listing is produced; a new object program is not created.
- **For SQL dataviews**-There can be only one version of a definition at a time; version 1 in production status. The CA Ideal CATALOG command creates SQL dataviews. You can delete them unless they are resources of production programs. You cannot edit them.

History

History versions of a definition are former production versions of that definition.

- **For all entity types except modeled and SQL dataviews**-Marking a test version to production automatically retires the existing production version, if any, to history status.
- **For CA Datacom/DB CBS dataviews and sequential file dataviews modeled in the dictionary**-The maximum number of history status versions that you can save is recorded in the dictionary (as the ENTY-HIST-VER attribute of the FILE entity in the DATA-DICT database). When this number is exceeded, the oldest history versions are automatically deleted. You can modify the installed default of three history versions.

MARK STATUS Considerations

The MARK STATUS command changes the status of a program, panel, report, user, or system definition from test to production, or from production to history. A definition is marked from test to production when the decision is made that no further testing is necessary, and the definition is ready to be used in a production application. Production versions are subject to the following rules:

- You cannot mark a production version of a definition that was named as a resource of a production program to history without first replacing it with a new production version.
- You can mark the production version of a definition that was not named as a resource of a production program directly to history.
- Marking another test version to production automatically retires any existing production version to history status. For example, if in the following list of versions, the fifth version's status was marked to production, the status of the fourth version is automatically changed to history:

  ```
  Version    Status
        1    HISTORY
        2    TEST
        3    HISTORY
        4    PRODUCTION
        5    TEST
  ```

  Marking version 2 to production also marks version 4 to history.
- If a subprogram or unmodeled dataview in production status is listed in the resource fill-in of a program in production status, and a new version of the subprogram or dataview becomes the production version, the resource fill-in of the calling program is automatically changed to reflect the new production version. However, to provide an audit trail, the history version of the subprogram is still included in the program-to-program relationship for the calling program. Consider this relationship when you execute the source transport utility. If you do not want to transport history versions, include the command SET EXPORT RESOURCE HISTORY NO in the transport command member.

How to Modify or Delete a Production Version

Because you cannot modify or delete a version that is in production status directly, you must use the following procedures to modify or delete the production version of a definition for all entity types except dataviews.

Modify a Production Version

1. Enter the following command to make a duplicate copy of the production version:

   ```
   DUPLICATE entity-type name NEXT VER
   ```

   The next version of the definition is in test status.

2. Make the changes to the new version.
3. Make the new version the current production version as follows:

   ```
   MARK STATUS entity-type name TO PROD
   ```

4. This statement also marks the current production version to history.

Delete a Production Version

1. Enter the following command to retire the production version to history:

   ```
   MARK STATUS entity-type name TO HIST
   ```

2. Delete the history version using the DELETE command:

   ```
   DELETE entity-type name version
   ```

A version's status usually is marked from production to history just before it is deleted. If a production occurrence is replaced with a new production version, it is not necessary to first mark the current production version to history.

How to Set a Default Version

The version of a program, panel or report to use when no version clause is specified in a command or when the version entry is omitted in a prompter, is determined as follows:

- As the site administrator, you can establish a default version for the site by using the SET SITE VERSION command or by accepting the installed default version of 001.
- An application developer can use the SET VERSION command to override the site default to establish which version is selected for commands and prompters during the session. This is often done using a SIGNON member. It affects only the user who entered the command.

Note: Some commands (such as DELETE) require a version clause.

- **For CA Datacom native access dataviews and sequential and VSAM file dataviews-**The default version of a dataview used as a program resource is determined by a SET [SITE] DATAVIEW VERSION command, which usually sets PROD as the default. The default version of a dataview specified by commands such as EDIT, DISPLAY, and PRINT, is specified by the SET [SITE] VERSION command, which usually sets LAST as the default.
- **For SQL dataviews**-Only one version is assigned; version 1 in production status. You cannot set the default version.

CA Ideal Users and Systems

There are many kinds of CA Ideal users. Anyone who uses CA Ideal, from the administrator who defines CA Ideal systems and users, to the data-entry person who runs the production version of a completed application, is a CA Ideal user. Each type of user has a different set of needs and responsibilities in CA Ideal. Each user is defined on the basis of these needs and responsibilities. The CA Ideal administrator (or a user with CA Ideal administrator authorization) must define a user in CA Ideal.

Users

When you define a user, you establish the user's privileges. User privileges define the general type of privilege a user has to perform activities that affect the global environment where CA Ideal functions. You must define at least one privilege for a user.

Some privileges imply other privileges. The following table illustrates the lesser privileges implied by the selection of a specified privilege.

Specified Privilege	**IDEAL Admin**	**PRINT Admin**	**DVW Admin**	**IDEAL User**
IDEAL Administrator	X	X	X	X
PRINT Administrator		X		X
DVW Administrator			X	X
IDEAL User				X

The following list provides an explanation of the terms in the preceding table:

IDEAL Administrator

The highest authorization given to a user of CA Ideal. A CA Ideal administrator can perform any command or service, including all services governed by all other privileges. This authorization is the only one that does not restrict the user to any specified system.

PRINT Administrator

Authorizes control over the print environment. This can allow the use of privileged commands that manage outputs.

DVW Administrator

Authorizes the use of functions such as executing the CATALOG dataview command.

IDEAL User

Authorizes the user to sign on to CA Ideal and to use those commands that require the minimum privilege for a user of CA Ideal. Generally, a CA Ideal user can execute only those commands that affect the current session.

Systems

A system is a collection of applications with their associated developers and users, as defined by a CA Ideal administrator. The system definition provides the name and description of the system. It also identifies the files the system uses for storing definitions and object code.

The CA Ideal administrator (or a user with CA Ideal administrator authorization) establishes a user's activity (authorization) in a system through the User Definition fill-in. System authorizations must be established for each system the user is allowed to access.

One system authorization can imply other authorizations. The following table illustrates how the assignment of an authorization in a system extends to commands and services governed by a lesser authorization.

Specified Privilege	**CONTROL**	**UPDATE**	**UPD-PNL**	**UPD-RPT**	**READ**	**RUN-PROD**
CONTROL	X	X	X	X	X	
UPDATE		X	X	X	X	
UPDATE-PNL			X		X	
UPD-RPT				X	X	
READ					X	
RUN-PROD						X

The administrator defines specific functions included in each authorization for each site. The administrator can assign different authorizations to all CA Ideal commands and to all options in commands. You can display the authorization levels in effect with the DISPLAY AUTHORIZATION OPTIONS (D ATZ OPT) command.

The following list provides an explanation of the terms in the preceding table:

CONTROL

Authorizes a user to create, establish, and edit program resources; delete, mark status, and update identification fill-ins; and otherwise control all program, panel, and report definitions in a specific system.

UPDATE

Authorizes a user to update all the program, panel, and report definitions (except a program resource fill-in and any identification fill-ins) or display all the program, panel, and report definitions in a specific system.

READ

Authorizes a user to display and print the report, panel, and program definitions in a specific system.

UPD-PNL (Update panel)

Authorizes a user to update panel definitions in a system.

UPD-RPT (Update report)

Authorizes a user to update report definitions within a system.

RUN-PROD (Run production)

Authorizes a user to run production programs in a system.

You receive an error message if you try to execute a command or option for which you are not authorized.

When signon occurs, a user is automatically associated with a CA Ideal system. If the user is authorized to use more than one system, the first system (alphabetically by collating sequence) is selected as the current system at signon (unless the user established a different system as the default current system in his signon procedure).

A user can only display, edit, run (and so on), programs, panels, and reports in the current system. Dataviews, users, and members do not belong to a system.

Naming Conventions in Systems

Names of programs, panels, and reports need be unique only in a system. In addition, for each system, a name must be unique only in entity type. For programs and panels converted to load module format, the module name must be unique for the entire site (see the “Module Format for Program and Panels” chapter for more information).

For example, there can be a report definition and a program definition both of which are named EX1, but two different report definitions or two different program definitions in the same system cannot be named EX1.

Defaults and Abbreviations

This section explains how defaults and abbreviations are used in CA Ideal.

Defaults in CA Ideal

When this document makes reference to defaults, in most cases no actual default value is mentioned because defaults CA Ideal uses are established in a number of ways, and are often specific to the site, or even to the individual user.

CA Ideal is delivered with default values for all options. You can change some of these defaults.

Defaults that cannot be changed

Some defaults are for certain choices in CA Ideal commands and fill-ins, and cannot be changed. This type of default is underlined in the command syntax examples in this guide.

An example of this type of default is the following command, which defaults to the procedure component of the program definition.

```
              [IDENTIFICATION]
              [RESOURCES     ]
EDIT PGM name [PARAMETER     ]
              [WORK          ]
              [PROCEDURE     ]
```

Defaults that can be changed for the entire site

You can establish defaults for the entire site. You can change some site defaults only during installation, and can reset them later only by rerunning installation jobs. An example of a default in this category is a default library name.

You can reset other site defaults by using either the SET SITE commands or the fill-ins provided for setting certain session control and print options. Any default that is reset with either a SET SITE command or a site options fill-in becomes a new site default. It remains in effect unless it is reset with another SET SITE command or fill-in.

Defaults that can be set for an individual session

Finally, each CA Ideal user can set most of the defaults that can be set for the site for an individual session, using SET commands or session options fill-ins. (See the *Working* in *the Environment Guide* for a description of the SET commands and the session options fill-ins. See the *Command Reference Guide* for the complete syntax of the SET command.)

A default set with a SET command or session options fill-in is changed only for the user who issued the command. It remains in effect only for the current session. The user has the option of storing SET commands in a member called SIGNON. This executes the commands automatically each time the user signs on, and works as if that user's default settings were changed permanently.

For references to defaults in this document, see whatever default is currently in effect for any given option. The actual default for any given option, for any given user, in any given session, at any given site, depends on what choices for setting defaults were made.

You can display or print the default option settings for the current session using the following commands:

```
DISPLAY SESSION OPTIONS
PRINT SESSION OPTIONS
```

For a list and descriptions of all of the session options available under CA Ideal, see the *Working in the Environment Guide*.

Use of Abbreviations in CA Ideal Commands

The following list provides an explanation of how abbreviations are used:

- **Standard**-The standard abbreviation for a command is the first three characters of the word. Abbreviations are not shown in the syntax illustrations in this guide.
- **Exceptions**-There are a number of exceptions to the standard first three-character abbreviation. These exceptions are abbreviations for command words whose first three characters are not unique and, therefore, would conflict with an abbreviation for another command. Other abbreviations (or no abbreviation at all) are used. For a complete list of these abbreviations, see the Preliminary Concepts section o*f the Command Reference* Guide.
- **Alternates**-CA Ideal also accepts alternate abbreviations. These alternate abbreviations are included in the list of abbreviations in the *Command Reference Guide*.

Main Menu

The Main Menu is a panel that offers a selection of the CA Ideal major facilities and functions. You can invoke the same facilities and functions by specifying the equivalent commands in the command area.

```
IDEAL: MAIN MENU                                       SYS: DOC      MENU

 Enter desired option number ===>       There are 11 options in this menu:

 1. PROGRAM            Define and maintain programs
 2. DATAVIEW           Display dataview definitions
 3. PANEL              Panel Definition Facility
 4. REPORT             Report Definition Facility
 5. PLAN               Application Plan/Package Maintenance
 6. PROCESS            Compile, Run, Submit, Debug
 7. DISPLAY            Display Entities
 8. PRINT              Print Entities
 9. ADMINISTRATION     Administration functions
10. HELP               Overview of HELP facilities
11. OFF                End IDEAL Session
```

The commands equivalent to each menu selection follow, with a description of each selection and the name of the guide where the function is documented.

- **PROGRAM**-How to define and maintain program definitions is described in the *Creating Programs Guide*.
- **DATAVIEW**-How to define, display, and print dataview definitions for CA Datacom/DB tables, sequential files, VSAM files, and DB2 tables is described in the *Creating Dataviews Guide*.
- **PANEL**-The panel definition facility, PDF, is described in the *Creating Panel Definitions Guide*.
- **REPORT**-The report definition facility, RDF, is described in the *Generating Reports Guide*.
- **PLAN (DB2 Only)**-The DB2 Plan definition facility is described in the chapter "Preparing DB2 Application Plans."
- **PROCESS, DISPLAY, and PRINT**-The PROCESS, DISPLAY, and PRINT functions are described in the *Working in the Environment Guide*.
- **ADMINISTRATION**-The functions of a CA Ideal administrator are described in this guide.
- **HELP**-The CA Ideal help facility is described in the *Working in the Environment Guide*.
- **OFF**-The command to end a CA Ideal session is described in the *Working in the Environment Guide*.

Administrative Functions

This is an overview of the functions required to maintain CA Ideal. You can categorize these functions as CA Ideal administration, print administration, and dataview administration.

Some of the functions typically performed by the different administrators are illustrated in the CA Ideal Administration Maintenance Menu. To access this menu, enter the number for the Administration option on the Main Menu or enter the command ADMINISTRATION.

```
---------------------------------------------------------------------------
IDEAL: ADMINISTRATION MAINT                             SYS: DOC      MENU

Enter desired option number ===>      There are 6 options in this menu:

 1. USER                   - Define and maintain user definitions
 2. SYSTEM                 - Define and maintain system definitions
 3. DDOL                   - Enter DATADICTIONARY Online (for DD Maint.)
 4. DATAQUERY              - Transfer to Dataquery
 5. CATALOG DATAVIEW       - Catalog dataviews
 6. UTILITY                - Miscellaneous utilities
```

Note: These functions are only available to users with the appropriate authorization. All other users receive an error message. Each option on the Administration Maintenance Menu is described fully in this guide. Options shown here that do not apply to your environment do not display on your screen.

Selecting an option displays a fill-in panel. When a fill-in is completed, press the Enter key or a PF key to apply the modified data. Pressing the Enter key applies the data, but leaves the current fill-in displayed. To continue, enter the appropriate command or press the appropriate PF key. Pressing the Clear key returns the session to the CA Ideal Main Menu without applying the modified data (except in certain cases with RUN). Pressing a PA key also ignores modified data. The PA1 key issues a RESHOW. The PA2 key displays current PF/PA key assignments.

In addition to the functions on the Administration Maintenance Menu, options to various commands that are usually restricted to administrators are described in the *Command Reference Guide*. Where appropriate, these administrative commands are described in this guide.

Chapter 2: System Overview

This chapter describes the CA Ideal software environment, internal software modules, and how CA Ideal application components are stored in the dictionary facility, VLS libraries, and CICS load modules.

CA Datacom/DB Environment

CA Ideal with CA Datacom/DB runs in an online environment or as a batch process. There can be a single region of each type or multiple CICS regions and multiple batch regions. Each request for CA Datacom/DB service goes through a supervisor call (SVC) that passes the request to the MUF. The SVC is used again to pass the results back.

Environment with DB2 Option

CA Ideal Option for DB2 runs under CICS or as a batch process. There might be a single region of each type or multiple CICS regions and multiple batch regions. The CA Datacom dictionary CA Ideal accesses must be allocated to a single, separate region accessed using a single Multi-User Facility (MUF). The SVC is used again to pass the results back. The DB2 databases that CA Ideal accesses are allocated to separate DB2 regions accessed using DB2 attach facilities. The tables for the dictionary facility are accessed from the MUF region.

CA Ideal Environment Components

The following sections provide lists and explanations of the CA Ideal components.

CA Datacom/DB and Datadictionary Components

The following is a list of the CA Datacom/DB and Datadictionary components:

- **MUF**-Multi-User Facility (MUF) provides the ability to access and update databases concurrently from multiple regions or partitions.
- **CA Datacom/DB**-CA Datacom/DB includes all modules installed as the CA Datacom/DB product. A DB Master List defines the operating environment to CA Datacom/DB, including the number of CICS and batch tasks that are accessing the databases.
- **CBS**-Compound Boolean Selection (CBS) is the CA Datacom/DB access facility required for processing CA Ideal FOR statements for CBS access. It analyzes complex database requests and determines the best search strategy.
- **DB Tables**-CA Datacom/DB databases that CA Ideal applications access, are allocated to the MUF region.
- **DD Tables**-Datadictionary tables are CA Datacom/DB databases allocated to the MUF region.
- **SVC**-A supervisor call (SVC) routine handles communication between the CICS and batch CA Ideal regions and MUF.
- **CICS Service Facility**-CICS Service Facility permits CA Ideal applications running under CICS to issue requests to the database through the SVC.
- **DSF/DDOL**-The Service Facility (DSF) for the Datadictionary component of CA Datacom/DB is used whenever CA Ideal accesses the Datadictionary in MUF. DSF is used in a CA Datacom/DB or mixed CA Datacom/DB and DB2 environment.

  DDOL, the online Datadictionary component of CA Datacom/DB, is used in CICS. DDOL provides interactive Datadictionary services, using DSF and the CICS Service Facility.

Datadictionary Considerations for CA Ideal

Site administrators should use the following guidelines when using Datadictionary with CA Ideal:

- Avoid required relationship types.

  We recommend that you do not define any relationship involving an entity-type that CA Ideal maintains as required. When CA Ideal attempts to create a new entity-occurrence, the required user-specified relationship occurrences cannot be supplied since they are unknown to CA Ideal. No further processing is allowed until the relationship is satisfied. The administrator can model user-defined relationship types between entity-types in the dictionary, but must make them optional (non-required) when they involve the entity-types SYSTEM, PROGRAM, PANEL, REPORT, and PERSON.

- Avoid using batch or online Datadictionary to maintain CA Ideal entity-occurrences.

  Do not attempt to use Datadictionary (batch or online) to maintain entity-occurrences or relationship occurrences that CA Ideal maintains; for example, entity-occurrences SYSTEM, PROGRAM, PANEL, REPORT, and PERSON; and relationship definitions PER-ATZ-AUTH, PER-SYS-ACCESS, SYS-LIB-RESIDE, SYS-PGM-CONTAIN, PGM-DVW-USE, PGM-PNL-USE, PGM-RPT-PRODUCE, and PGM-PGM-CALL. The administrator can maintain entity-occurrences using Datadictionary, of any entity types that model other resources at the site, but the administrator should never tamper with those occurrences that CA Ideal creates.

  Also, the administrator should never tamper with relationship occurrences or with intersection data between occurrences that CA Ideal creates since these often contain CA Ideal control information.

  Three exceptions are:

  - Occurrences of PERSON and SYSTEM can be initially defined, if necessary, using Datadictionary rather than using CA Ideal. However, to make such occurrences valid for CA Ideal use, the CA Ideal administrator must EDIT SYSTEM and USER (PERSON) definitions using CA Ideal commands to enter additional information that CA Ideal requires.
  - The administrator can use Datadictionary directly to maintain text records.
  - Dataviews can be related to systems using the SYS-DVW-USE relationship.

- Avoid conflict with CA Ideal internal naming conventions.

  When using Datadictionary (batch or online), avoid conflicts with CA Ideal internal naming conventions. PROGRAM, PANEL, and REPORT entity-occurrences created by CA Ideal begin with $I. Therefore, users are advised not to create any occurrences beginning with a $I.

- Use entity types consistently.

  CA Ideal uses the entity types PROGRAM and SYSTEM with specific meanings: A PROGRAM represents a compilable unit of an application that can run alone or can call or be called by other programs (for example, a subprogram). A SYSTEM is a collection of related application PROGRAMs and is also related to its USERs who are developers, end users, and so on. There is no restriction on the use of these entity-types elsewhere in the dictionary; however, be sure to consider their special meaning to CA Ideal.

  Note: CA Ideal does not use the Datadictionary entity types MEMBER and PLAN.

- No entity-occurrence passwords.

  Datadictionary allows any entity-occurrence to be assigned a password. Do not specify entity-occurrence passwords (four-character PASSWORD attribute) for entity-occurrences that CA Ideal is to access. For example, if you create PERSON entity-occurrences with Datadictionary and assign them entity-occurrence passwords, CA Ideal cannot access them. Do not confuse the restrictions of not using entity-occurrence passwords with the 12-character signon password (PASS-WORD attribute of the PERSON entity type) that CA Ideal supports, that is the same attribute CA Ideal displays on the USER definition fill-in.

- Do not change LOCK attribute on CA Ideal entity-occurrences.

  All entity-occurrences that CA Ideal creates have a LOCK attribute value of 1. Do not change this attribute.

DB2 Environment Components

The following DB2 and CA Datacom/AD components support a CA Ideal Option for DB2 environment.

- **DB2 Tables**-The DB2 databases that CA Ideal applications access are allocated to the DB2 region.
- **CA Datacom/AD**-A subset of CA Datacom/DB, providing Datadictionary support in a DB2 only environment.
- **CICS Attach Facility**-The CICS DB2 Attach Facility permits CA Ideal applications running under CICS to issue requests to DB2.
- **Call Attach Facility**-The DB2 Call Attach Facility permits CA Ideal applications running in batch to issue requests to DB2.

CA Common Services for z/OS

Relevant CA Common Services for z/OS (CA Common Services) are described as follows:

CAISSF

The CA Standard Security Facility (CAISSF) interfaces with external security systems, such as CA Top Secret, CA ACF2, RACF, and other SAF-compatible products. The SSF interface lets you use any of these security products in both online and batch environments, independent of the CA Ideal code.

During signon, CA Ideal internal security checks to make sure that the user is authorized to access CA Ideal. If external security is used, users must sign onto the security system before signing onto CA Ideal. You can use any CA Ideal signon transaction (standard, express, or transparent) with external security.

For more information about internal and external security, see the “Establishing Signon Processing” chapter.

CA IPC (CA Inter-Product Components)

CA IPC provides common functions to CA Ideal and an interface between CA Ideal and the external environment. These programs allow CA Ideal to process user requests and to provide the services requested while remaining independent of the operating system and teleprocessing monitor.

- **VPE-**Virtual Processing Environment acts as a software interface between CA Ideal and the teleprocessing monitor, operating system, and database. All external services that CA Ideal requires are invoked through VPE. Other CA IPCs also use VPE for their external services.

  VPE handles requests for services such as memory management, program management, resource management, I/O services, enqueue/dequeue, dynamic batch job submission, and database accesses. It handles these requests for service through macro calls and performs the functions according to the rules and syntax of the host environment.

- **PDF-**CA Ideal Panel Definition Facility processes the CA Ideal commands used to define, print, and test panels interactively.

- **PSS-**Print Subsystem processes, routes, and manages print requests. PSS handles CA Ideal commands for maintaining the output library, browsing output, and performing other print services.

- **PMS-**Panel Management Services is a set of runtime services for acquiring, sending, receiving, and managing 327x-type terminal messages. PMS validates date/time stamps and checks input fields for violations of panel edit rules. PMS supports both CA Ideal product panels and application panels developed using PDF.

- **SCF-**Session Control Facility handles user requests, whether from an online terminal session or from a CA Ideal batch session. SCF provides CA Ideal signon processing, handles asynchronous compiles and network printing, and processes commands such as DISPLAY ERROR, SPLIT and COMBINE, SCROLL, and HELP, RETURN, and CLARIFY.

  SCF also acts as a command dispatcher, managing CA Ideal menus, commands, and PF and PA keys, separating commands and passing commands to the appropriate CA Ideal module for further analysis. In batch, SCF prints commands as though they were entered in an online environment.

- **VLS-**Virtual Library System uses the basic I/O services of VPE to store, retrieve, and modify data in both online and batch environments.

  - Two types of VLS library members are as follows:
  - **Record members:** Store source data in source libraries and output in the output library.
  - **Block data members:** Store variable-length data in panel libraries or object libraries.

  Any number of VLS libraries can exist, but they cannot be concatenated. CA Ideal requires at least eight VLS libraries:

  - **ADRLIB**-CA Ideal control information, master JOBCARD, user JOBCARDS, and system messages.
  - **ADRPNL**-CA Ideal and CA IPC product panels and session options.
  - **ADROUT**-The output library.
  - **IDDAT**-Data member library.
  - **IDDVW**-Dataview object library. For a DB2 site, plan definitions can be included here or in a separate plan library.

    User source, object, and panel libraries (defaults are: in z/OS, ID$IDSRC, ID$IDOBJ, and ID$IDPNL; in VSE, IDL$IDS, IDLIDO, IDLIDP).

  A site can define additional source, object, and panel libraries.

  In addition to being used directly by CA Ideal and CA IPC, VLS has a batch utility for library maintenance (VLSUTIL).

- **EDK**-Editor Kernel maps fill-in panels to VLS members, providing editing commands and checkpoint/rollback functions.

CA Ideal Internals

This section outlines the CA Ideal internal modules and describes how components of CA Ideal applications are stored.

CA Ideal Modules

CA Ideal is comprised of a number of Assembler modules, falling into the following categories:

- **Compiler-**CA Ideal has its own compiler used both online and in batch to build object modules of programs, panels, and reports.
- **Executor-**Control the CA Ideal runtime environment.
- **Online Services-**These panels and processors create and maintain user and system definitions and to process commands such as DISPLAY INDEX, CREATE, DELETE, DUPLICATE, MARK, PRINT, DISPLAY, CATALOG DATAVIEW, and SUBMIT.
- **Editors-**There is a separate editor or processor in CA Ideal for each CA Ideal application component.
- **Batch Utilities-**The CA Ideal batch utilities are primarily used for site administration functions.

For a list of individual CA Ideal internal modules as they relate to the CA Ideal Trace Facility, see the “Dial Trace Codes” appendix in the *Problem Determination* Guide.

Application Components, VLS, and the Dictionary Facility

The following chart shows how CA Ideal application components (systems, users, dataviews, data members, panels, reports, programs, and plans) are recorded in the dictionary facility, stored on a VLS library, or both.

Component	**Dictionary**	**Source**	**Object**	**Panel**	***IDDVW**	**IDDAT**
SYSTEM	X					
USER	X					
DATAVIEW	X				X	
MEMBER						X
PANEL IDENT	X					
PANEL LAY/SUM/ EXD, and so on			X	X		
REPORT IDENT	X					
REPORT PAR/HEA/DET/COL		X				
PROGRAM IDENT	X					
RESOURCE	X					
PARAMETER		X	X			

Component	Dictionary	Source	Object	Panel	*IDDVW	IDDAT
WORKING DATA		X	X			
PRODECURE		X	X			
*PLAN					X	

Note: * A DB2 site can define a separate plan library.

How CA Ideal Stores Application Components

Entries in the dictionary facility and VLS members for CA Ideal components are created at various times during the application development process.

The first CA Ideal entity-occurrences are stored in the dictionary facility when CA Ideal is installed. At that time, the first CA Ideal system and user are created, the appropriate entity-occurrences are added to the dictionary facility, the VLS source library, object library, and panel libraries are specified, and the appropriate relationships are created in the dictionary facility.

When additional users and systems are defined, CA Ideal adds other entries to the dictionary facility.

The database administrator defines modeled dataviews in the dictionary. In CA Ideal, the CATALOG DVW command reads the dictionary tables and creates an object module in the IDDVW VLS library.

Unmodeled dataviews

These are created in CA Ideal adding a source member to IDDVW and data to the dictionary facility. The CATALOG DVW command accesses the source member and the dictionary tables, and creates an object module in the IDDVW VLS library. Unmodeled dataviews can be created for sequential files and VSAM files.

CA Ideal data members are stored in the VLS library IDDAT. They are not modeled in the dictionary facility.

Panels and reports

Panels and reports are modeled in the dictionary facility and stored as VLS members. A dictionary entity-occurrence is created when the panel or report identification is entered. A single VLS member is added to the source library when the report parameter, header, detail, and column sections are created. This member contains all of these report components. A VLS member is added to the panel library for the panel layout, summary data, extended field definitions, input edit and validation rules, output edit rules, and facsimile. This member contains all of these panel components.

Programs

Programs are also modeled in the dictionary facility. An entity-occurrence is created in the dictionary when you enter the program identification. Relationship occurrences are added when you enter the program resource table. Separate VLS members are added to the source library for each program component. That is, members are added for the program parameter data, working data, and procedure. Each member is added when the corresponding component is first edited.

CA Ideal VLS Operations

This section describes how various CA Ideal commands affect VLS members.

CREATE

RPT

Creates a single member on the SOURCE library (type R). This member contains all the components of the report (parameters, heading, detail, and column).

PNL

Creates a single member on the PANEL library (type U). This member contains all the components of the panel (parameters, layout, summary, field, and facsimile).

PGM

CREATE PGM does not create any VLS members. See EDIT.

MEM

Creates a single member on the IDDAT library (type Z).

EDIT

RPT

Makes a copy of the report member on the source library with an edit-indicator of E. At normal end of EDIT, the E member is deleted.

PNL

Makes a copy of the panel member on the panel library with an edit-indicator of E. At normal end of EDIT, the E member is deleted.

PGM

A single program can result in up to three separate source members on the source library: procedure (type L), working data (type W), and parameter data (type P). Each member is first created when the corresponding component is first edited. In addition, any time an EDIT is issued for a given component (including when it is first created), a copy of the member is made on the SOURCE library with an edit-indicator of E. At normal end of EDIT, the E member is deleted. Deleting all of the lines of a component does not delete the corresponding member.

MEM

Makes a copy of the data member on the IDDAT library with an edit-indicator of E. At normal end of EDIT, the E member is deleted.

DUPLICATE

RPT

Makes a copy of the report member with the new name: For the NEXT option, the version number is changed to one higher than the highest version number. For the NEWNAME option, the report name is changed to the new report name, and the version is changed to 001. The DUP command internally invokes an EDIT command for the new report.

PNL

Makes a copy of the panel member with the new name: For the NEXT option, the version number is changed to one higher than the highest version number. For the NEWNAME option, the panel name is changed to the new name, and the version is changed to 001. The DUP command internally invokes an EDIT command for the new panel.

PGM

Makes a copy of any of the three possible program members that exist (work, parameter, procedure), with the new name: For the NEXT option, the version number is changed to one higher than the highest next version number. For the NEWNAME option, the program name is changed to the new name, and the version is changed to 001. The DUP command internally invokes an EDIT command for the new program.

MEM

Makes a copy of the data member with the new name: For a different user, the user ID is changed to the new user ID. The DUP command internally invokes an EDIT command for the new member.

MARK STATUS

PGM is the only VLS members that the MARK STATUS command affects are program object members. For the program executable object (type T) and symbol table (type J), the members are renamed so that the version number is changed to PRD.

This enables the runtime executor to locate the appropriate members in a system that was the target of the Transport Utility, when no Datadictionary entity-occurrence exists. This is necessary to support the command RUN *xxxxxxxx* VER PROD, otherwise, CA Ideal would have no means of determining which of the multiple versions of the object members were the production versions.

Example

Suppose that in SYS ACC, PGM UPDATE VER 005 is currently in PROD status, VER 006 is in TEST, and that VER 006 is to become the new PROD version. Before the MARK STATUS command, the following members exist in the object library (if you run a VLSUTIL LIBRARY listing, the member names are shown in strict collating sequence order, instead of the following order):

```
ACCUPDATE          PRDJA
ACCUPDATE          PRDTA
ACCUPDATE          PRDTB
ACCUPDATE          005VA
ACCUPDATE          006JA
ACCUPDATE          006TA
ACCUPDATE          006TB
ACCUPDATE          006VA
ACCUPDATE          007JA
ACCUPDATE          007TA
ACCUPDATE          007TB
ACCUPDATE          007VA
```

This program contains working data but no parameter data. We can determine this because there is a type V member (working data object) but no type Q member (parameter object). Also, the original MARK command for VER 005 changed only the names of the type J (symbol table) and T (executable object) members to reflect PRD instead of 005: The type V member was never changed. (A type Q member would not change either.)

In this example, only two members are shown for each executable object module, A and B. There can be more members (C, D, E, and so forth), but two is the minimum.

The following is now executed:

```
MARK STATUS PGM UPDATE VER 6 TO PROD
```

CA Ideal determines that there is an existing previous PROD version of the program by accessing the Datadictionary. After the above command is executed, the following members exist on the object library (if you run a VLSUTIL LIBRARY listing, the member names are shown in strict collating sequence order, instead of the following order):

```
ACCUPDATE        PRDJA
ACCUPDATE        PRDTA
ACCUPDATE        PRDTB
ACCUPDATE        006VA
ACCUPDATE        007JA
ACCUPDATE        007TA
ACCUPDATE        007TB
ACCUPDATE        007VA
```

All members for the old PROD version 005 were deleted (since CA Ideal determined from the Datadictionary that version 5 was the previous PROD version). The former 006 members were renamed to PRD since the command directed CA Ideal to mark version 6 to PROD status. Neither of the type V members (working data object) was affected by the MARK STATUS command. These members are used only at compilation time.

DELETE

- **RPT**-Report source member is deleted.
- **PNL**-Panel source and object members are deleted.
- **PGM**-All program source and object members are deleted.
- **MEM**-Data member is deleted.

The DELETE command deletes all appropriate VLS members, even if the Datadictionary entity-occurrence, or one or more of the VLS members themselves, is missing.

Compilation and VLS Object Modules

The following activities happen when the CA Ideal application is compiled for the first time:

1. The program working data, parameter data, and the attributes of each panel are compiled into separate object modules.
2. These objects, along with any panels and cataloged dataviews, are merged with composite object modules.
3. The program procedure and report definitions are compiled and merged with the composite object module into an executable object.
4. The program symbol table is also built at compile time.

Each time a program is recompiled, the program procedure and report definitions are recompiled into the executable object module. The separate object modules for panel attributes, program working data, and program parameter data only need to be recompiled if they have changed since the last compile. Any that have not changed will be merged directly into the executable object when recompiled.

Processing the Field Attribute and Symbol Tables

The maximum number of symbols-such as names and identifiers-in a CA Ideal application depends greatly on the amount of storage CA Ideal has available for the Field Attribute Table and the Symbol Table.

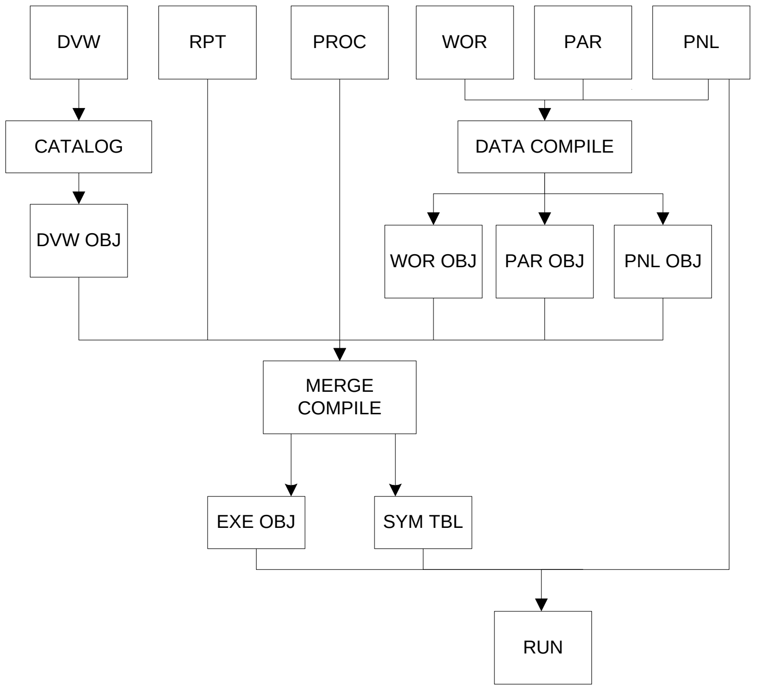
The Field Attribute Table, referred to here as the FAT, contains an entry for each dataview, dataview field, panel, panel field, working data field, literal, and FOR construct. Each entry is 20 bytes long and contains information such as the length, type, displacement, and the offset into the Symbol Table.

The Symbol Table contains an entry for each dataview name, dataview field name, panel name, panel field name, working data field name, report name, procedure name, and label name. The length of these entries varies according to the length of the symbol. Each entry contains information, such as the length of the symbol and the address of the symbol in the FAT, and the literal that identifies the symbol.

During compilation, CA Ideal creates a variety of blocks. None of these blocks can exceed 32 KB. If all the FAT or Symbol Table entries of a 01 level cannot fit in the remaining space of the current block, a new block is started. No parent (01 level) is separated from its children. The FAT and Symbol Table can span multiple blocks.

A single data entity occurrence, (which means that a 01 level data item, dataview, or panel), still cannot exceed the 32 KB limit.

This means that no 01 level can have more than 1,600 fields, or it exceeds the FAT table limit. Also, the maximum that a 01 level can have is somewhere between 800 and 1,875 fields (depending on the size of the symbols), otherwise it exceeds the Symbol Table limit. The compilation and execution process is shown in the following illustration.



Application Load Modules (Phases)

You can convert CA Ideal application programs and panels in Production status from VLS members to z/OS load modules or to VSE phases. If a program is converted to load module format, load modules are created for the executable object and the symbol table. For each program, separate load modules are created for the reentrant and non-reentrant data, respectively, and for the symbol table. If a panel is converted to load module format, a single load module is created. The panel load module contains the panel member from the panel library, comprising both reentrant and non-reentrant data.

Even when VLS members are converted to load modules, the original object members are retained and can be used again. MODULE entity occurrences in the dictionary facility record which programs or panels were converted to load modules and which were deleted as load modules. If an entity accessed from CA Ideal is not recorded as a load module, CA Ideal retrieves it from the VLS library. For more information about VLS Member Formats, see the CA Ideal *Problem Determination Guide*. For more information about load module format, see the “Module Format for Programs and Panels” chapter in this guide.

Execution of CA Ideal Applications in a CICS Environment

At runtime, for a single program, there are always three independent VLS object members (or load modules)-one reentrant, one non-reentrant, and one for the symbol table. There can be more VLS members depending on the size of the program.

The distinction between reentrant and non-reentrant is significant because the execution of CA Ideal applications online takes place in pseudo-conversational mode. At the end of each CICS transaction, some or all of the in-core resources are written to CICS auxiliary temporary storage or CICS EDSA. Exactly what is written to auxiliary temporary storage and what remains in core depends on the format of the program or panel.

If programs and panels were converted to load modules, they are controlled by CICS. The CA Ideal program or panel run status has no effect on load module format.

The program symbol table is loaded only when a runtime error occurs.

In a DB2 environment, execution of programs containing SQL in static mode requires an application plan. The “Preparing DB2 Application Plans” chapter describes how application plans are generated and used at runtime.

In CA Datacom/DB, SQL access plans are built at compile time.

Execution and the Dictionary Facility

Exactly how the dictionary facility is used at runtime depends on the status of the program.

- If the program is in Test status when the program is run, the program entity-occurrence is retrieved from the dictionary facility, and the date/time stamps are checked.
- The first time any related programs are called, the entity-occurrences of these programs and of their resources must also be retrieved from the dictionary facility, their date/time stamps checked, and table entries built.

- If the program is in Production status and the PROD keyword is specified with the RUN command (for example, RUN MYPROG VER PROD) or specified as the default (by SET VERSION PROD), no dictionary facility access takes place.
- If the program is in Production status but the PROD keyword is not specified with the RUN command or as the default, the program entity-occurrence needs to be retrieved, but no date/time checking is performed. Since all related programs and resources must also be in Production status, no further dictionary facility access is needed.

Chapter 3: Defining and Maintaining CA Ideal Users

The user definition provides multiple levels of security for CA Ideal. It secures access to CA Ideal to development and production systems in CA Ideal, and to specific activities in each system. The user definition identifies each CA Ideal user, establishes the user name, grants CA Ideal privileges, assigns the user to systems, and assigns authorization levels in each system.

Creating and Maintaining User Definitions

You can define each user individually to CA Ideal or you can define one or more user groups with access to common systems and appropriate authorizations in those systems. This section explains how to create and maintain the user definitions.

You can define and maintain user definitions either directly through the commands described in this section or by selecting options from the User Maintenance Menu. To display the User Maintenance Menu, select option 1 on the Administration Maintenance Menu or enter the CA Ideal command USER.

```
=>
=>
=>
----------------------------------------------------------------------------
IDEAL: USER MAINTENANCE      USR                         SYS: DOC      MENU

 Enter desired option number ===>      There are 7 options in this menu:

 1. EDIT/DISPLAY         - Edit or display a user definition
 2. CREATE               - Create a user definition
 3. PRINT                - Print a user definition
 4. DELETE               - Delete a user definition
 5. MARK STATUS          - Mark user status to production or history
 6. DUPLICATE            - Duplicate user definition to next version
 7. DISPLAY INDEX        - Display index of user definitions
```

Some of the options on the User Maintenance Menu display fill-in panels for data entry.

When a fill-in is complete, press Enter or a PF key to apply the modified data. Pressing the Enter key applies the data, but leaves the current fill-in displayed. To continue, enter the appropriate command or press the appropriate PF key. Pressing the Clear key returns the session to the CA Ideal Main Menu without applying the modified data. Pressing a PA key also ignores modified data. The PA1 key issues a RESHOW. The PA2 key displays current PF/PA key assignments.

For prompter panels, pressing Enter processes the command completed on the prompter.

Creating a CA Ideal User Definition

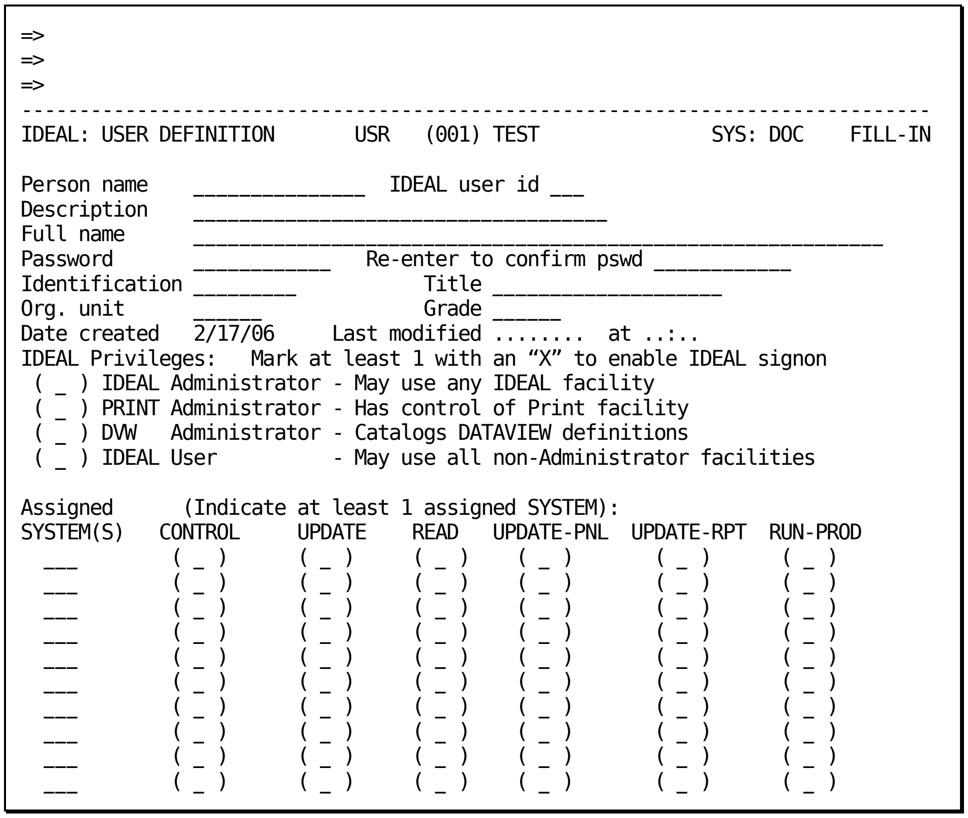
To create the first version of a user definition, enter the command CREATE USER. The CREATE USER command displays a blank user definition fill-in. The User Definition Fill-in is a panel that establishes the user, assigns a user privilege, enters descriptive information about the user, and establishes the user's authorization level in each assigned system. If a user is already defined in the dictionary (for Datadictionary or CA DataQuery, for example), duplicate the user to the next version and edit the new version to define the user to CA Ideal.

A newly created user definition is assigned a version number of 1. This version of the user definition is in test status. You can edit it at any subsequent session as long as it remains in test status.

Note:

- Before a user can sign on to CA Ideal, the user definition must be marked to production status.
- When you are using an external security system to control access to CA Ideal, you must be sure that a user definition exists for the administrator before changing the SC00OPTS table to include the option SECRTY=Y. If a user definition does not exist, you cannot sign on to CA Ideal to create the remaining user definitions.

To display the User Definition Fill-in, enter the command CREATE USER or select Option 2, CREATE, from the User Maintenance Menu.



```
=>
=>
=>
-------------------------------------------------------------------------------
IDEAL: USER DEFINITION       USR   (001) TEST              SYS: DOC    FILL-IN

Person name    _______________  IDEAL user id ___
Description    ____________________________________
Full name      ____________________________________________________________
Password       ____________   Re-enter to confirm pswd ____________
Identification _________           Title ____________________
Org. unit      ______              Grade ______
Date created   2/17/06     Last modified ........  at ..:..
IDEAL Privileges:   Mark at least 1 with an “X” to enable IDEAL signon
 ( _ ) IDEAL Administrator - May use any IDEAL facility
 ( _ ) PRINT Administrator - Has control of Print facility
 ( _ ) DVW   Administrator - Catalogs DATAVIEW definitions
 ( _ ) IDEAL User          - May use all non-Administrator facilities

Assigned      (Indicate at least 1 assigned SYSTEM):
SYSTEM(S)   CONTROL     UPDATE    READ   UPDATE-PNL  UPDATE-RPT  RUN-PROD
  ___        ( _ )      ( _ )     ( _ )    ( _ )       ( _ )      ( _ )
  ___        ( _ )      ( _ )     ( _ )    ( _ )       ( _ )      ( _ )
  ___        ( _ )      ( _ )     ( _ )    ( _ )       ( _ )      ( _ )
  ___        ( _ )      ( _ )     ( _ )    ( _ )       ( _ )      ( _ )
  ___        ( _ )      ( _ )     ( _ )    ( _ )       ( _ )      ( _ )
  ___        ( _ )      ( _ )     ( _ )    ( _ )       ( _ )      ( _ )
  ___        ( _ )      ( _ )     ( _ )    ( _ )       ( _ )      ( _ )
  ___        ( _ )      ( _ )     ( _ )    ( _ )       ( _ )      ( _ )
  ___        ( _ )      ( _ )     ( _ )    ( _ )       ( _ )      ( _ )
  ___        ( _ )      ( _ )     ( _ )    ( _ )       ( _ )      ( _ )
```

Note: Issue “MARK * PROD” or “MARK USER xxx VER n TO PROD” to enable signon

The fields on the User Definition Fill-in consist of:

- **Person name-**One- to 15-character identification of the person (user) being defined. This name must be unique at the site. CA Ideal initializes Person name to the name entered in the CREATE command, if a name was supplied there. You cannot change Person name once it is entered and accepted. If you want a new Person name, you must delete the user definition and create a new one. The Person name can be one to 15 non-blank characters and can be any combination of letters, digits, or national characters except the first character cannot be a digit.

  The Person Name does not have to be the same as the security ID or the monitor op-ID used to logon if the UIDCHK parameter of the IDOPTSCB macro is set to NO. However, unless a default user ID (DFLTUSR) is specified to determine the CA Ideal authorizations, you must still define the security ID or op-ID as an alias for the Person Name in the dictionary facility.

  If DESTINATION MAIL is used, the Person Name must be a valid CA eMail+ signon or alias.

- **IDEAL User ID-**One- to three-character user identification. You must enter this user ID to make the user valid for CA Ideal and it must be unique to the site. Once defined in a User Definition Fill-in, you cannot change the CA Ideal user ID. If you want a new CA Ideal user ID, delete the user definition and create a new one. The user ID can be one to three non-blank characters and can be any combination of letters, digits, or national characters. (This one- to three-character user identification becomes an alias for the PERSON in the dictionary facility.)

  Note: In CICS, the CA Ideal user ID must match the CICS op-ID unless either the UIDCHK parameter of the IDOPTSCB macro is set to NO or security is enabled (SECRTY=Y in the SC00OPTS table).

- **Description-**Optional description of the user being defined.

- **Full name, Identification, Title, Org unit (organization unit), Grade**-Descriptive information or organizational identification of the user. These areas are for documentation only and are optional.

- **Password**-One- to 12-character string that the user must specify on the CA Ideal signon screen. The contents of this field on the user definition panel display or are invisible, depending on the value assigned to the IDOPTSCB parameter PSWDIS=. (The user can modify this password without editing the user definition fill-in by using the ALTER SIGNON PASSWORD command described in the *Command Reference Guide*.)

- **Re-enter to confirm pswd**-A prompt and the field that only appears when the IDOPTSCB parameter PSWDIS=INVISIBLE, verifying the password field. For more information, see Customizing the CA Ideal Options Block Using IDOPTSCB, in the “Optimizing Storage Management” chapter.

- **Date created**-Initial date of the user definition. The system supplies this date. It is not modifiable.

- **Last modified at**-Date and time when this user definition was last accessed in edit mode. The system supplies the date. It is not modifiable. The system also supplies the time in the format hh:mm. It is not modifiable.
- **IDEAL Privileges**-Each user of CA Ideal is assigned (by marking with an x) one or more levels of privileges that govern the commands and services available to that user. You must select at least one privilege to make the user definition valid for signing on to CA Ideal. (To disable an existing CA Ideal user, erase all CA Ideal privileges and ignore the resulting error message.)
  - **IDEAL Administrator**-Grants the user global authorization to perform any command or service in all systems, including all services governed by all other privileges.
  - **PRINT Administrator**-Establishes a CA Ideal user with the authorization to control the printing facility (which includes the ability to define and delete print destinations).
  - **DVW Administrator**-Establishes a CA Ideal user with the authorization to catalog dataviews for CA Ideal.
  - **IDEAL User**-Specifies that the user is authorized to use only commands that affect the current session.

  The following table illustrates how the selection of a privilege also implies commands and services governed by other privileges.

Specified Privilege	IDEAL Admin	PRINT Admin	DVW Admin	IDEAL User
IDEAL Administrator	X	X	X	X
PRINT Administrator		X		X
DVW Administrator			X	X
IDEAL User				X

- **Assigned systems**-Designates which CA Ideal systems this user is authorized to use and the authorization level the user has for each system. To assign more than the ten systems allowed on the panel, scroll forward using the PF8 key.

  A user's access to a system and to the commands that can execute in a system is controlled by the authorization specified here. The user must be assigned to at least one CA Ideal system to successfully sign on to CA Ideal.

  - **CONTROL**-Authorizes full control of the specified system, including the creation and deletion of programs, panels, and reports; editing the identification fill-in for programs, panels, and reports; and editing the resource fill-in of a program definition. This authorization automatically implies all other levels in the system, except running production programs (RUN-PROD) that must be selected separately.
  - **UPDATE**-Authorizes the user to update (edit) or read (display, print, and so on) all programs, panels, and reports in the system (except the identification fill-in and resource fill-in). This authorization automatically implies all levels in the system except control (CONTROL) and running production programs (RUN-PROD).
  - **READ**-Authorizes the user to read (display, print, and so on) programs, report definitions, and panel definitions in the system.
  - **UPDATE-PNL (update panel)**-Authorizes the user to update panel definitions in the system.
  - **UPDATE-RPT (update report)**-Authorizes the user to update report definitions in the system.
  - **RUN-PROD (run production)**-Authorizes the user to run production programs in this system.

The following table illustrates how the assignment of an authorization in a system implies commands and services governed by a lesser authorization.

Specified Authorization	CONTROL	UPDATE	UPD-PNL	UPD-RPT	READ	RUN-PROD
CONTROL	X	X	X	X	X	
UPDATE		X	X	X	X	
UPD-PNL			X		X	
UPD-RPT				X	X	
READ					X	
RUN-PROD						X

Note: To enable the newly defined user to sign onto CA Ideal, issue the command MARK * PROD or MARK USER xxx VER *n* TO PROD after defining the user.

Maintaining User Definitions Online

Use the following CA Ideal commands to display, maintain, copy, and list existing user definitions. For more information about these commands, see the *Command Reference Guide*.

CREATE USER

Displays a fill-in panel that creates a user definition in the dictionary.

EDIT/DISPLAY USER

Displays an existing user definition and makes it the current entity.

PRINT USER

Prints a specific user definition.

DELETE USER

Deletes a user definition that is in history or test status. User definitions in production status must be marked to history before they can be deleted.

For important information about using the DELETE command to delete user definitions, see the notes following this table.

MARK STATUS USER

Marks a user definition's status to production or history. A user definition must be in production status before the user can sign onto CA Ideal.

DUPLICATE USER

Copies an existing user definition to the next version. The new definition becomes the current user definition, and the user fill-in displays for modification. You can modify the new user definition as long as the status is test. Until it is modified, the newly created version is identical to the previous version, including the name.

Note: You cannot copy a user definition to a new name with the DUPLICATE USER command.

DISPLAY/PRINT INDEX

Lists the name and status of each user definition currently in the dictionary. You can request an index for one or all users, with or without listing the related systems. Margin commands can be used to display, edit, delete, or mark the status of the displayed user definitions.

Note: Before deleting a user definition, be sure to delete any data members that exist for that user since the DELETE MEMBER command requires the Person Name or User ID from the user definition. If the user definition is deleted before the members that belong to that user are deleted, the members become impossible to delete with the CA Ideal DELETE command.

To determine whether the user has members, use the command:

```
DISPLAY INDEX MEMBER USER username
```

Then enter the DELETE line command for each member displayed and press Enter. (You could use the following command to delete a member but the DELETE MEMBER command must be repeated for each member.)

```
DELETE MEMBER memname USER username
```

Marking a user definition to history and deleting it using this DELETE command also removes the corresponding PERSON entity occurrence from the dictionary facility. If that user was authorized for any other CA products, those authorizations are automatically deleted.

To disable a user from CA Ideal without affecting that user's authorizations for other CA products, follow this procedure:

1. Duplicate the existing production version of the user to NEXT VERSION. This displays the next version of the user definition for editing.
2. While viewing the user definition fill-in for the new test-status version (in edit mode), erase all CA Ideal privileges and press Enter.
   The following message displays:

   ```
   ADUEDP11 - Please enter CA Ideal privilege(s)
   ```

3. Ignore the preceding message and mark the new version of the user definition to production. The CA Ideal editor does not allow the update unless there is at least one SYSTEM related to the USER definition. Without any CA Ideal privileges, signon to CA Ideal with this user ID results in the following error message:

   ```
   IDADIDIN05E - USR xxx has no signon authorization for CA Ideal
   ```

The Datadictionary batch utility, DDUPDATE, also removes CA Ideal authorization (transaction 1003 UNRL).

Creating CA Ideal Users in Batch

The usual method to create CA Ideal users is to use the CREATE USER command. However, adding large numbers of users this way can be a time-consuming process. CA Ideal users can be created in batch using Datadictionary (DDUPDATE).

A valid CA Ideal user definition consists of the following Datadictionary entity-occurrences:

- A PERSON entity-occurrence
- An ALIAS for the PERSON entity-occurrence that must match the PERSON userid attribute
- One or more RELATIONSHIP occurrences between the PERSON and the four CA Ideal AUTHORIZATION profiles and (optionally) one of the DD AUTHORIZATION profiles:
  - $$ID-ADM IDEAL Administrator
  - $$ID-DVW DVW Administrator
  - $$PR-ADM PRINT Administrator
  - $$ID-USE IDEAL User
- One or more RELATIONSHIP occurrences between the PERSON and the defined CA Ideal SYSTEM entity-occurrences. The intersection data (INTER-DATA) of this relationship occurrence must contain the following information:

  ```
  Pos Contents
  1-3 $ID (constant)
  4 One byte of bit settings for the authorization
  level in the related system, as follows:
  1... .... - Control
  .1.. .... - Update
  ..1. .... - Read
  ...1 .... - Run-Prod
  .... 1... - Not Used
  .... .1.. - Not Used
  .... ..1. - Update-Report
  .... ...1 - Update-Panel
  ```

Following is a sample set of DDUPDATE transactions to add a new user to CA Ideal:

```
-ADD PERSON,long-user-name
    1003 RELT,SYSTEM,system-name(ver),PER-SYS-ACCESS
    1003 DATA,$IDx
    1010 ADD $$ID-ADM
    1010 ADD $$ID-DVW
    1010 ADD $$PR-ADM
    1010 ADD $$ID-USE
    1014 pppppppppppp uuu
    -END
    -UPD PERSON,long-user-name(001),PROD
    -END
```

- The -ADD transaction is a header transaction. It adds the PERSON entity-occurrence for user long-user-name (from 1 to 15 characters).
- The 1003 RELT, SYSTEM transaction relates the user 'long-user-name' to the system 'system-name(ver)'. The new user can be related to a maximum of 99 SYSTEMs.
- There must be a matching 1003, DATA transaction for each 1003 RELT, SYSTEM transaction. The 1003 DATA transaction gives the user authorization within the system specified in the corresponding 1003 RELT, SYSTEM transaction. The authorization bit setting ('x') is determined using the following table:

```
/--- AUTH IN SYS: TYPE:            HEX   CHAR
|                 ---------------  ---   ----
|                 CTL + RUN-PROD    F3     3
|                 CTL               E3     T
|                 UPD + RUN-PROD    7B     #
|                 UPD               6C     %
|                 READ + RUN-PROD   30
|                 READ              20
|                 RUN-PROD          10
|
------------------------------------------
```

- The 1010 ADD transaction assigns the user a privilege title: {IDEAL-ADMIN, DATAVIEW-ADMIN, PRINT-ADMIN, IDEAL-USER}.

  For example, to define an IDEAL-USER, supply only the following transaction:

  ```
  1010 ADD $$ID-USE
  ```

- The 1014 transaction defines a password and the userid for the 'long-user-name'. If a password is not desired, the 1014 transaction must still be included to add the userid. Note that the userid MUST start in column 19.
- The -END transaction marks the conclusion of each transaction group.
- The -UPD transaction marks the 'long-user-name' definition to PROD status.
- The -END transaction marks the conclusion of each transaction group.

When these transactions have been successfully executed, the new CA Ideal user is ready to sign on.

Using Batch to Maintain CA Ideal User Definitions

It is possible to create, as well as update, CA Ideal users in batch through the Datadictionary utility program DDUPDATE.

Adding Aliases

You may want to add aliases to existing user definitions on a large scale. The following DDUPDATE transactions let you add an alias to existing users in batch:

```
+UPD PERSON,person1(PROD,,ovrd)
1103 ADD alias1
+END
+UPD PERSON,person2(PROD,,ovrd)
1103 ADD alias2
+END
```

Adding Systems

Changes to an existing user's SYSTEM authorization online require duplicating the user to the next version, modifying the user definition, and then marking the new version to PROD status. You can make the same changes in batch using DDUPDATE 1003 transactions to relate and unrelate systems to users. The following DDUPDATE transactions let you add an additional system to an existing user in batch.

```
+UPD PERSON,person1(PROD,,ovrd)
1003 RELT,SYSTEM,long-system-nme(PROD),PER-SYS-ACCESS
1003 DATA,$IDx
+END
```

$ID is a constant and *x* is the authorization. The 1003 DATA statement is the intersection data (INTER-DATA) for this RELATIONSHIP occurrence and must contain the following information:

- Positions 1 through 3 contain the $ID (constant)
- Position 4 contains a one-byte value with the following bit-settings for the authorization level in the related system:

  0x80 = Control

  0x40 = Update

  0x20 = Read

  0x10 = Run-Prod

  0x08 = Not Used

  0x04 = Not Used

  0x02 = Update-Report

  0x01 = Update-Panel

Updating System Authorizations

The following DDUPDATE transactions let you update the system authorization of a production status CA Ideal user definition in batch.

```
+UPD PERSON,person1(PROD,,ovrd)
1003 UNRL,SYSTEM,long-system-nme(PROD),PER-SYS-ACCESS
1003 RELT,SYSTEM,long-system-nme(PROD),PER-SYS-ACCESS
1003 DATA,sidx
+END
```

Changing Passwords

If you need to change many CA Ideal user passwords, the easiest way in batch is to use a 1014 transaction in DDUPDATE for each user. You do not need to issue an ALTER SIGNON PASSWORD command to change the CA Ideal password. When you change your password in DDOL, you are also changing it for CA Ideal conversely.

In this case, using DDUPDATE is more efficient than using CA Ideal batch. One execution of DDUPDATE with one 1014 transaction is required for each user, unlike CA Ideal batch, which requires that you set up one batch job or one batch step for each user that includes the SIGNON and ALTER SIGNON password transactions.

Also, remember that you need a -UPD PERSON transaction header for each user.

Chapter 4: Defining and Maintaining Systems

In CA Ideal, a system is a collection of application programs and the developers and users associated with them. The System Maintenance Menu that follows illustrates the functions provided by CA Ideal for the definition and maintenance of system definitions. To access this menu, select option 2, CREATE, on the CA Ideal Administration Maintenance Menu or enter the SYSTEM command.

```
=>
=>
=>
----------------------------------------------------------------------------
IDEAL: SYSTEM MAINTENANCE    SYS                         SYS: DOC      MENU

 Enter desired option number ===>      There are   7  options in this menu:

 1. EDIT/DISPLAY        - Edit or display a system definition
 2. CREATE              - Create a system definition
 3. PRINT               - Print a system definition
 4. DELETE              - Delete a system definition
 5. MARK STATUS         - Mark system status to production or history
 6. DUPLICATE           - Duplicate system definition to next version
 7. DISPLAY INDEX       - Display index of system definitions
```

You can define and maintain system definitions either by selecting options from the System Maintenance Menu or by entering the commands described in this section directly from the command line.

Creating a CA Ideal System

To create the first version of a system definition, enter the command CREATE SYSTEM. The CREATE SYSTEM command displays a blank system definition fill-in. This fill-in establishes the system name, provides identification information about the system, and provides the file names where the system libraries are stored.

A newly created system definition is assigned a version of 1 and is placed in test status. You can edit the new system definition at any subsequent session as long as it remains in test status. Once marked to production status, it is not modifiable.

After the system definition fill-in is presented and you enter a system name, that system becomes the current system definition.

Note: Once created, the system definition is in test status and must be marked to production status before you can access the system.

To display the System Definition Fill-in, shown in the following illustration, enter the command CREATE SYSTEM or select option 2, CREATE, from the System Maintenance Menu.

```
=>
=>
=>
---------------------------------------------------------------------------
IDEAL: SYSTEM DEFN.         SYS   (001) TEST           SYS: DOC    FILL-IN

System name                ______________
System short identifier    ___
Description                ___________________________________
System user                ______________
Date designed              __ __ __
Date implemented           __ __ __
Application identification ______________

Date created 2/21/06   Last modified ........  at ..:..

SYSTEM LIBRARIES:
  TYPE     FILE NAME

  Source   IDXXXSRC
  Object   IDXXXOBJ
  Panels   IDXXXPNL
```

Note: Issue “MARK * PROD” or “MARK SYS xxx VER n PROD” to enable SELECT SYSTEM.

The components of the System Definition Fill-in are as follows:

- **System name**-One- to 15-character name of the system being defined. CA Ideal initializes System name to the name entered on the CREATE command if you entered a name. Once defined on the system definition fill-in, you cannot change the System name. If you want a new System name, delete the system definition and create a new one.
- **System short identifier**-Three-character system identifier. This three-character identifier becomes an alias for the system in the dictionary facility.

  The System short identifier must be exactly three non-blank characters. The first character must be a letter or national character. The other two characters can be letters, digits, or national characters.

Note: Once defined on the system definition fill-in, you cannot change this identifier. If you want a new System short identifier, delete the system definition and create a new one.

- **Description (Optional)**-Area where you can provide description of the system being defined (documentation only).
- **System user**-Area where you can identify the users of the system (documentation only).
- **Date designed**-Area where you can specify the date the system was designed (documentation only).
- **Date implemented**-Area where you can specify the date the system was implemented (documentation only).
- **Application identification**-Area where you can further define the system (documentation only).
- **Date created**-Initial date of the system definition. The system supplies this date. You cannot modify it.
- **Last modified at**-Date and time when this system definition was last accessed in edit mode. The format of the time value is *hh:mm*. The system supplies the date and time. You cannot modify them.
- **System libraries**-Names of the libraries the system uses for storing source and object modules. Each system can have its own set of libraries or multiple systems can share one or more libraries.
  - **Type**-Identifies the type of each library in the system as follows:
  - **Source libraries**-Contain the source of program and report definitions.
  - **Object libraries**-Contain the object form of program and panel definitions.
  - **Panel libraries**-Contain panel definitions.
  - **File name**-For each type, an area where you must supply the file name for each library.

When the System Definition Fill-in is presented as a result of the CREATE SYSTEM command or prompter, CA Ideal provides default file names.

For z/OS, the default names are:

- ID*XXX*SRC
- ID*XXX*OBJ
- ID*XXX*PNL

For VSE, the default names are:

- IDL*XXX*S
- IDL*XXX*O
- IDL*XXX*P

Replace the *XXX* in each name with the system short identifier to relate the file names to the system definition by convention.

Since each file name becomes a DD statement (in z/OS) or a DLBL statement (in VSE), the names must conform to the rules for formation of ddnames or DLBL names. File names in VSE cannot be longer than seven characters, even though the fill-in provides eight positions (for compatibility with z/OS.)

The source, object, and panel libraries can be in separate files or any combination can be on the same file. The maximum size of a VLS library member is 60,900 blocks (regardless of the blocksize.) Consider this when planning libraries.

You can put each CA Ideal system in its own library or you can put any number of systems in the same library, as shown in the following table.

Library Types	**Possible Library Names for System INV**	**Possible Library Names for System ACT**
	IDALLSRC	IDALLSRC
Source	IDINVSRC	IDACTSRC
	IDALL	IDALL
	IDALLOBJ	IDALLOBJ
Object	IDINVOBJ	IDACTOBJ
	IDALL	IDALL
	IDALLPNL	IDALLPNL
Panel	IDINVPNL	IDACTPNL
	IDALL	IDALL

There are other steps required to allocate, format, and establish new source, object, or panel libraries for a CA Ideal system.

Note: To enable the SELECT SYSTEM command for the newly defined system, issue the command MARK * PROD or MARK SYS *xxx* VER *n* PROD.

Maintaining System Definitions

Use the following CA Ideal commands to display, maintain, print, copy, and list existing systems. For detailed information about these commands, see the Command *Reference Guide*.

CREATE SYSTEM

Displays a fill-in panel that creates a system definition in the dictionary.

EDIT/DISPLAY SYSTEM

Displays an existing system definition and makes it the current CA Ideal entity.

PRINT SYSTEM

Prints a specific system definition.

DELETE SYSTEM

Deletes a system definition that is in history or test status. Production-status system definitions must be marked to history status before they can be deleted.

MARK STATUS SYSTEM

Marks a system definition's status to production or history. A system definition must be in production status before users can select it.

DUPLICATE SYSTEM

Copies an existing system definition to the next version. The new version of the system definition becomes the current system definition. The system definition fill-in displays for modification. You can modify the new system definitions long as the status is test. Until the new definition is modified, the newly created version is identical to the previous version, including the name.

Note: You cannot copy a system definition to a new name with the DUPLICATE SYSTEM command.

DISPLAY INDEX SYSTEM

Lists the name and status of each system definition currently in the dictionary facility. Optionally, the index can include occurrences of entity-types that are related to a given system).

Note: Ignore system names that are prefixed with the characters DD-. They are not available as CA Ideal systems.

Using Batch to Create CA Ideal System Definitions

You can create systems in batch through the Datadictionary utility program, DDUPDATE.

Chapter 5: Preparing and Maintaining VLS Libraries in CA Ideal

Each CA Ideal PROGRAM, PANEL, REPORT, and DATAVIEW entity consists of one or more VLS library members and an entity-occurrence in the dictionary facility. (MEMBERS consist of a VLS member only, not a dictionary facility entity-occurrence.) CA Ideal requires that the environment (both VLS members and dictionary facility entity-occurrences) be valid and consistent. If a discrepancy is found, CA Ideal issues an internal error indication and terminates the operation.

The VLSUTIL utility manipulates VLS members to regain continuity between the VLS libraries and the dictionary facility. For more information about VLSUTIL, see the *CA IPC Implementation Guide*.

The CREATE SYSTEM command creates a new system. The resulting system definition fill-in assigns the VLS libraries. The three libraries for applications in that system are the program source library, the program object code library, and the panel library.

Any of the libraries can be shared across systems or in one system. The administrator can use one library for all systems, one library per system, three individual libraries per system, or any other combination.

This chapter describes the procedures to follow before CA Ideal can use a library (also known as a VLS file).

Library Maintenance Under z/OS or VSE

This section describes the following procedures for creating and maintaining VLS files:

- Allocating and initializing the library
- Adding the library to the JCL procedures installed with CA Ideal
- Adding the library to the CICS File Control Table for CICS
- Adding the library to the VPE Batch File Table for Batch

This section also explains the following additional maintenance procedures:

- Backing up and restoring a VLS file
- Increasing the size of a VLS library

Allocating and Initializing a VLS Library

A VLS file is organized logically as a collection of members. A member name identifies a member. The following table shows the required name length (NAMELEN) for each CA Ideal VLS file. The block size can be any size convenient to the device type, except that each file type has a minimum block size. The data is actually stored in VLS internal format in fixed length blocks. Default (installed) and minimum block sizes are shown in the following table. Data compression and space recovery are automatic.

OS	**File**	**Required Namelen**	**Installed Blksize**	**Minimum Blksize**
	IDAT	24	1960	960
	IDDVW	40	4000	4000
z/OS	ICxxxSRC	24	1960	960
	IDxxxPNL	24	4000	4000
	IDxxxOBJ	24	4000	4000
VSE	IDLxxxS	24	1960	960
	IDLxxxO	24	4000	4000
	IDLxxxP	24	4000	4000

The following files that CA Ideal uses are installed with CA IPC:

File	**Required Namelen**	**Installed Blksize**	**Minimum Blksize**
ADRLIB	24	4000	4000
ADROUT	11	4000	4000
ADRPNL	24	4000	4000

z/OS JCL to Initialize the Library

```
//INITIAL  EXEC PGM=VLSUTIL
//STEPLIB  DD   DSN=CA IPC.LOAD,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//VLSFILE  DD   DSN=library.name,DISP=(,CATLG),
//              UNIT=xxxxx,VOL=SER=xxxxxx,SPACE=(CYL,n),
//              DCB=DSORG=DA
//AUXPRINT DD   SYSOUT=*
//SYSIN           DD   *
FORMAT BLKSIZE=nnnn,NAMELEN=nn
/*
//
```

VSE JCL to Initialize the Library

```
// DLBL IDLCL,'IDEAL.IDL.LOAD'   *** CORE ***
// EXTENT ,IDL001
// LIBDEF PHASE,SEARCH=IDLCL.IPC
* *** INIT A NEW VLS LIBRARY
// ASSGN SYS004,DISK,VOL=IDL001,SHR
// DLBL VLSFILE,'IDEAL.NEW$PNL',,DA
// EXTENT SYS004,IDL001,1,0,1295,120
// EXEC VLSUTIL,SIZE=(AUTO,48K)
FORMAT BLKSIZE=nnnn,NAMELEN=nn
/*
```

Adding VLS Library JCL

z/OS

The name of the DD statement must be the same as the library name entered in the system definition fill-in panel. You must add this statement to the CICS jobstream and the CA Ideal batch jobstream. A sample JCL statement follows:

```
//ddname  DD      DSN=library.name,DISP=SHR
```

VSE

The name of the DLBL statement must be the same as the library name entered in the system definition fill-in panel. You must add the following JCL statements to the specified JCL procedures installed with CA Ideal, for a new library:

```
// DLBL dlbl-name,”dsn-name”...
// EXTENT, volume
```

The following are JCL procedures installed with CA Ideal that require the new JCL statements:

```
IDLPROC
IDLCICS
IDLDB
```

Adding a VLS Library to the CICS FCT

The VLS file name used in the CICS FCT (File Control Table) entry must be the same as the name in the DD or DLBL statement and the same as the name entered in the system definition fill-in panel. The block size and access method parameters in the FCT must match the block size and access method used when the file was initialized. The block size and record length parameters must be equal, since the record format is F (fixed).

For sample FCT entries, check the entries created during the CA Ideal installation for the default VLS libraries.

Note: You must recycle CICS before you can access a new VLS library online.

Adding a VLS Library to the CA Ideal Batch User File Table

The Virtual Processing Environment (VPE) uses the VPE File Tables for batch to isolate CA Ideal from differences between operating environments. For more information, see CA Ideal File Table for VSE in the chapter "Optimizing Storage Management."

z/OS

The following example shows how to add the appropriate entries to the IDUSRFT and reassemble these tables:

```
IDSYSFT TITLE  'SCF - BATCH FILES FOR IDEAL'
*
*                                                                 *
*  FILES USED BY IDEAL SYSTEM BATCH JOBS                          *
*  =====================================                          *
*                                                                 *
*  ANY FILE USED BY AN IDEAL BATCH RUN MUST BE IN THIS            *
*  TABLE. THIS INCLUDES CATALOGED PROCEDURES IDLBATCH,            *
*  PSSUTIL, AND IDLXPRT.                                          *
*                                                                 *
*                                                                 *
*
   .
   .
   .
*
*  IDEAL USER SYSTEM FILES                                        *
*
   .
   .
   .
*
ID$IDSRC ROSFD  DDNAME=ID$IDSRC,ACCMETH=BDAM,RECFM=F,PRODUCT=IGN
ID$IDPNL ROSFD  DDNAME=ID$IDPNL,ACCMETH=BDAM,RECFM=F,PRODUCT=IGN
ID$IDOBJ ROSFD  DDNAME=ID$IDOBJ,ACCMETH=BDAM,RECFM=F,PRODUCT=IGN
   .
   .
   .
*
  END
```

This procedure makes the new file definitions available to the batch CA Ideal and CA Datacom/DB MUF environments.

For more information about the parameters used in the ROSFD entry, see the section CA Ideal File Table for VSE in the chapter, "Optimizing Storage Management." The new library is now usable.

VSE

For more information about maintaining a CA Ideal File Table under VSE, see CA Ideal File Table for VSE in the chapter “Optimizing Storage Management.”

Backing Up and Restoring a VLS Library

The JCL step to add to the backup jobstream is shown as follows.

z/OS

```
//VLSBKUP  EXEC  PGM=VLSUTIL
//STEPLIB  DD    DSN=CA IPC.LOAD,DISP=SHR
//SYSPRINT DD    SYSOUT=*
//AUXPRINT DD    SYSOUT=*
//VLSFILE  DD    DSN=library.name,DISP=SHR
//VLSBKUP  DD    DSN=backup.name,UNIT=TAPE,
//               DISP=(NEW,CATLG),VOL=SER=xxxxxx
//SYSUDUMP DD    SYSOUT=*
//SYSIN    DD    *
BACKUP
/*
//
```

The JCL step added to the restore jobstream is shown as follows. The block size must match the block size as specified in the list of block size values.

```
//VLSREST  EXEC PGM=VLSUTIL
//STEPLIB  DD  DSN=CA IPC.LOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//AUXPRINT DD  SYSOUT=*
//VLSFILE  DD  DSN=library.name,DISP=SHR
//VLSBKUP  DD  DSN=backup.name,UNIT=TAPE,DISP=OLD
//SYSUDUMP DD  SYSOUT=*
//SYSIN    DD  *
FORMAT BLKSIZE=nnnn,NAMELEN=nn
RESTORE
LIBRARY
/*
//
```

VSE

```
/*
// PAUSE : PLEASE MOUNT "IDLBKP" TAPE ON UNIT=180
// ASSGN SYS010,180
// MTC REW,SYS010
// DLBL IDLCL,'CA IPC.IDL.LOAD' *** CORE ***
// EXTENT ,IDL001
// LIBDEF PHASE,SEARCH=IDLCL.IPC
* *** BACKUP FOR "NEW$OBJ" DSN=IDEAL.NEW$OBJ
// ASSGN SYS004,DISK,VOL=IDL001,SHR
// DLBL VLSFILE,'IDEAL.NEW$OBJ',,DA
// EXTENT SYS004,IDL001
// TLBL VLSBKUP,'NEW$OBJ',,IDLBKP,,1
// UPSI 00000011
// EXEC VLSUTIL,SIZE=(AUTO,64K)
LIBRARY
BACKUP
```

Note: UPSI switch values are documented in the CA IPC VSE *Implementation Guide*.

```
// PAUSE : PLEASE MOUNT “IDLBKP” TAPE ON UNIT=180
// ASSGN SYS010,180
// MTC REW,SYS010
// DLBL IDLCL,'CA IPC.IDL.LOAD'   *** CORE ***
// EXTENT ,IDL001
// LIBDEF PHASE,SEARCH=IDLCL.IPC
* *** BACKUP FOR “NEW$OBJ” DSN=IDEAL.NEW$OBJ
// ASSGN SYS004,DISK,VOL=IDL001,SHR
// DLBL VLSFILE,'IDEAL.NEW$OBJ',,DA
// EXTENT SYS004,IDL001
// TLBL VLSBKUP,'NEW$OBJ',,IDLBKP,,1
// UPSI 00000011
// EXEC VLSUTIL,SIZE=(AUTO,64K)
FORMAT BLKSIZE=nnnn,NAMELEN=nn
RESTORE
LIBRARY
```

Routine CA Ideal System BACKUP and RESTORE

CA Ideal maintains information about various entities in both the dictionary and in the various CA Ideal VLS files. This information is synchronized through internal date and time stamps. Therefore, CA Ideal VLS files cannot, in general, be treated as independent from the entire CA Ideal system or from the dictionary. You should schedule routine backups of the entire system.

This should be a job that includes the BACKUP function of DBUTLTY for each dictionary file (see the *CA Datacom/DB z/OS Utility Reference* for a description and sample JCL), and one VLSUTIL BACKUP step for each of the following CA IPC and CA Ideal VLS files:

- ADRLIB
- ADROUT
- ADRPNL
- IDDAT
- IDDVW

z/OS

- IDxxxSRC (*one* for each separate source file)
- ID*xxxPNL* (one for each separate panel file)
- IDxxxOBJ (one for each separate object file)

VSE

- IDL*xxx*S (one for each separate source file)
- IDL*xxx*P (one for each separate panel file)
- IDL*xxx*O (one for each separate object file)

CA Ideal generation includes the installation of a JCL stream to backup all CA Ideal files. Tailor this JCL to your site standards and add it to your current dictionary backup job.

It is essential that you run the BACKUP function of DBUTLTY and CA Ideal VLSUTIL steps at the same point (preferably as steps in one job) and that CA Ideal be completely quiesced (including batch CA Ideal runs) before and during the backup.

Sample Backup JCL

The following is sample JCL for backing up one of the CA Ideal files:

z/OS

```
//IDDVW         EXEC PGM=VLSUTIL
//STEPLIB  DD   DSN=CA IPC.LOAD,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//AUXPRINT DD   SYSOUT=*
//VLSBKUP  DD   (backup of IDDVW file)
//VLSFILE  DD   DSN=index.IDDVW,DISP=SHR
LIBRARY
BACKUP
```

VSE

```
/*
// PAUSE : PLEASE MOUNT “IDLBKP” TAPE ON UNIT=180
// ASSGN SYS010,180
// MTC REW,SYS010
// DLBL IDLCL,'CA IPC.IDL.LOAD'   *** CORE ***
// EXTENT ,IDL001
// LIBDEF PHASE,SEARCH=IDLCL.IPC
* *** BACKUP FOR “NEW$OBJ” DSN=IDEAL.NEW$OBJ
// ASSGN SYS004,DISK,VOL=IDL001,SHR
// DLBL VLSFILE,'IDEAL.NEW$OBJ',,DA
// EXTENT SYS004,IDL001
// TLBL VLSBKUP,'NEW$OBJ',,IDLBKP,,1
// UPSI 00000011
// EXEC VLSUTIL,SIZE=(AUTO,48K)
LIBRARY
BACKUP
```

Sample Restore JCL

In some instances, it might be necessary to restore all of the Datadictionary, all CA IPC, and CA Ideal VLS files to maintain synchronization of the definitions should any one of the components become damaged. (Also refer to the following section on restoring deleted entities.) You can restore these files by using the output of the previously described full CA Ideal /DDOL system backup as input to a series of DBUTLTY LOAD steps (one for each dictionary file) and VLSUTIL RESTORE steps. CA Ideal and DDOL must be completely quiesced (including batch CA Ideal runs) before attempting the full system restore.

The following is sample JCL for restoring a CA Ideal VLS file:

z/OS

```
//IDDVW          EXEC PGM=VLSUTIL
//STEPLIB  DD   DSN=CA IPC.LOAD,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//AUXPRINT DD   SYSOUT=*
//VLSBKUP  DD   (backup of IDDVW file)
//VLSFILE  DD   DSN=index.IDDVW,DISP=SHR
FORMAT BLKSIZE=4000,NAMELEN=40
RESTORE
LIBRARY
```

VSE

```
/*
// PAUSE : PLEASE MOUNT “IDLBKP” TAPE ON UNIT=180
// ASSGN SYS010,180
// MTC REW,SYS010
// DLBL IDLCL,'CA IPC.IDL.LOAD'   *** CORE ***
// EXTENT ,IDL001
// LIBDEF PHASE,SEARCH=IDLCL.IPC
* *** BACKUP FOR 'NEW$OBJ' DSN=IDEAL.NEW$OBJ
// ASSGN SYS004,DISK,VOL=IDL001,SHR
// DLBL VLSFILE,'IDEAL.NEW$OBJ',,DA
// EXTENT SYS004,IDL001
// TLBL VLSBKUP,'NEW$OBJ',,IDLBKP,,1
// UPSI 00000011
// EXEC VLSUTIL,SIZE=(AUTO,48K)
FORMAT BLKSIZE=4000,NAMELEN=24
RESTORE
LIBRARY
/*
```

Increasing the Space in a VLS File

To expand the size of a VLS file or to move it to a different device or device type, carry out the following steps:

1. Make sure the VLS file is not being accessed:

   In CICS-Shut down CICS. Under z/OS, you can deallocate the data set without shutting down CICS; however, you then need to reallocate the data set before you recycle CICS.

   You also need to quiesce all CA Ideal batch activity.

2. Use the BACKUP function of VLSUTIL to create a backup (sequential) form of the VLS library.

3. Delete the old VLS library and reallocate it (using IEFBR14) with the appropriate characteristics. The VLS file can be allocated as part of the following RESTORE step. For information regarding VLSUTIL LIBFMT=F option to format larger data sets, see the *CA IPC Implementation Guide*.

4. Run VLSUTIL again with the backup (sequential) file as input (VLSBKUP), including the following two control cards:

   ```
   FORMAT NAMELEN=nn,BLKSIZE=bbbb
   RESTORE
   ```

5. Restart CICS.

Restoring Deleted Entities

On occasion, a site might find itself in a situation where an entity was accidentally deleted from the CA Ideal environment. If the CA Ideal program, panel, report, sequential or VSAM dataview, or member exists in “external format” as a result of a Source Transport Utility EXPORT job, then you can recover the entity (both dictionary and VLS portions) using the IMPORT function of the CA Ideal Source Transport Utility. See the *Working in the Environment Guide* for more information. Otherwise, you can restore the entity if a VLS backup of the source is available. Each entity requires information from both the Datadictionary and the VLS library.

To restore an entity

1. Populate the dictionary with information regarding the entity. This can be done using either the CREATE, DUPLICATE, or the CA Ideal source transport utility. This process adds the information found in the Identification section and, in the case of programs, in the Resource section.

2. Restore the VLS library members using VLSUTIL.

Scenarios are provided below to assist in restoring your CA Ideal entities. The scenario you decide to use depends on the existence of other versions of the same entity in the system and what those versions are.

Resource Section Considerations

In many cases, you must update the resource section of your newly restored programs. You might find missing resources or resources that do not reflect the appropriate version.

VLSUTIL Considerations

1. The Dictionary entity must be present before VLSUTIL SELREST is run. This is because the CA Ideal CREATE command creates empty VLS members. If the VLS entities already exist during the create process, CA Ideal reinitializes them (or empty them).

   Note: For restoring PROGRAM entities, create the Working Data and Parameter sections of the program (if they exist) before the corresponding members are restored. If they are not, you must edit the sections (that is, overtype a single character) before you can successfully compile the program. Datadictionary does not know these sections exist until they are created or edited.

2. Remember to use the HEX x card to specify a hex delimiter for your member names during VLSUTIL functions.

   For example, to restore a program called DRIVER in system $ID, version 1, consisting of a procedure, working data and parameter data section, use the following VLSUTIL input statements. The H'40s represent blanks to pad the program name of DRIVER to 15 characters.

   ```
   HEX /
   SELREST $IDDRIVER/404040404040404040/001L
   SELREST $IDDRIVER/404040404040404040/001W
   SELREST $IDDRIVER/404040404040404040/001P
   ```

   See the CA IPC Implementation Guide for more information.

3. If you need to use the RENAME function of VLSUTIL, you need to RENAME each member using two RENAME functions because you cannot fit the old and new member name in the 80-character maximum.

   For example, to rename a program called EMPLMENU in system $ID, currently in version 22 to version 1:

   ```
   HEX /
   RENAME $IDEMPLMENU/40404040404040/022L,TEMP1
   RENAME TEMP1,$IDEMPLMENU/40404040404040/001L
   RENAME $IDEMPLMENU/40404040404040/022W,TEMP2
   RENAME TEMP2,$IDEMPLMENU/40404040404040/001W
   RENAME $IDEMPLMENU/40404040404040/022P,TEMP3
   RENAME TEMP3,$IDEMPLMENU/40404040404040/001P
   ```

Restoration Procedures

Scenario 1

An entity in version 1 was deleted and there are no other versions of that entity in the system.

1. Create the entity in CA Ideal, filling in the identification section. If the entity is a program, you must also fill in the resource
section.
2. Use the VLSUTIL SELREST function to restore all VLS source members for the entity.

Scenario 2

An entity in a version other than version 1 was deleted, and there are no other versions of that entity in the system. Plus, you want the entity returned with the original version number.

1. Create the entity in CA Ideal, filling in the identification section. If the entity is a program, you must also fill in the resource section.
2. Source transport the entity, first exporting it, then importing it back to the original system using the command SET IMPORT NEW VERSION x, where x is the version of the program you are going to restore from VLS.
3. Use the VLSUTIL SELREST function to restore all VLS source members for the entity.
4. Delete Version 1 of the entity from CA Ideal.

Scenario 3

An entity in a version other than version 1 was deleted and there are no other versions of that entity in the system. Plus, you want the entity returned in a version that is different from the original.

Option 1

1. Create the entity in CA Ideal, filling in the identification section. If the entity is a program, you must also fill in the resource section. If you want the entity returned in a version other than 1, you must either use source transport to create the correct version or duplicate the entity to the correct version.
2. Use the VLSUTIL SELREST function to restore all VLS source members for the entity.
3. Use the VLSUTIL RENAME function to rename all VLS source members for the entity from the original version to the new version.
4. Delete any unwanted versions of the entity from CA Ideal.

Option 2

1. Create the entity in CA Ideal, filling in the identification section. If the entity is a program, you must also fill in the resource section.
2. Source transport the entity, first exporting it, then importing it back to the original system using the command SET IMPORT NEW VERSION *x*, where *x* is the version of the program you are going to restore from VLS.
3. Use the VLSUTIL SELREST function to restore all VLS source members for the entity.
4. Source transport the entity, first exporting it, then importing it back to the original system using the command SET IMPORT NEW VERSION *x*, where *x* is the new version of the entity. Also, make sure that you use the command SET IMPORT DUPLICATE REPLACE.
5. Delete any unwanted versions of the entity from CA Ideal.

Scenario 4

An entity was deleted and there are lower numbered versions of the entity on the system. The entity is returned with the original version number.

Option 1

1. Source transport the entity, first exporting it, then importing it back to the original system using the command SET IMPORT NEW VERSION x, where x is the version of the program you are going to restore from VLS.
2. Use the VLSUTIL SELREST function to restore all VLS source members for the entity.

Option 2

1. Use the CA Ideal DUPLICATE command to duplicate the entity to the version you are going to restore.
2. Use the VLSUTIL SELREST function to restore all VLS source members for the entity.
3. Use the CA Ideal DELETE command to delete unwanted versions of the entity.

Scenario 5

An entity was deleted and there are higher numbered versions of the entity on the system. The entity is returned with the original version number.

1. Source transport the entity, first exporting it, then importing it back to the original system using the command SET IMPORT NEW VERSION *x*, where *x* is the version of the program you are going to restore from VLS.
2. Use the VLSUTIL SELREST function to restore all VLS source members for the utility.

Example 1

Version 18 of program ABC in system $ID was accidentally deleted. Version 19 of the same program exists in $ID. You want the program in its original version.

You follow Scenario 5 in this example. First export program ABC, version 19, then import using the command SET IMPORT NEW VERSION 18. Next use VLSUTIL SELREST to restore version 18 of the program to the source library.

Example 2

Version 6 of program TEST in system $ID was accidentally deleted. No other version of the program exists in $ID. You want the program returned in version 3.

You follow Scenario 3 for this example. Create program TEST VER 1 in system $ID. Fill in the identification and resource sections. Use source transport to export the program, then import using SET IMPORT NEW VERSION 6. Use VLSUTIL SELREST to restore version 6 of the program to the VLS library. Use source transport again, exporting version 6, then import using SET IMPORT NEW VERSION 3. You can then delete version 6 from CA Ideal.

What to Do When the Library Is Full

A VLS library has a limit of 60,900 blocks with 2-byte block numbers. If the library becomes full, you must determine why the library is full before you can correct the problem. There are four possible reasons for the library to become full:

- Library contains orphaned history members for entities that were removed from the dictionary.
- Defined block size for the VLS library is inefficient.
- VLS library contains more than one system, and the total number of entities is too large for the library.
- VLS library contains only one system, and the total number of entities is too large for the library.

The following sections describe each of these problems and the recommended solutions.

Removing Entities in History Status

VLS members in the source library can be orphaned when the number of history versions for an entity occurrence exceeds the value of the Datadictionary parameter ENTY-HIST-VERS, which is installed with a default value of 3 for all entities. When a particular entity occurrence exceeds this value, the oldest history version is deleted from the dictionary. However, because this is a dictionary function, and the dictionary is not tied to the VLS library that CA Ideal uses, the VLS source member is not deleted. Eventually this can fill the source library with members that no longer exist in the dictionary. Delete these orphaned members to free up space in the VLS library.

To delete orphaned VLS source members

1. Run the IDUTILTY utility to determine which members do not have a corresponding dictionary entry. This utility generates a report of all VLS members that do not have a corresponding dictionary entry. See Library Integrity Utility in this chapter for more information.
2. Use the CA Ideal DELETE command to delete those source members, on-line or in batch, after you determine which members do not have a corresponding dictionary entry. The DELETE command deletes the entity from VLS even when no corresponding entry is found in the dictionary.

You can also use the VLSUTIL DELETE command to delete the VLS source members.

Other Members That Can Be Deleted

If your VLS library is still full, you can consider deleting other history status programs, panels, and reports that are no longer needed. If you are concerned about losing valuable but inactive versions of entities, you can save history status entities in external source transport format. This keeps the entities available without filling up your VLS library.

Note: History status programs do not have VLS object members. CA Ideal deletes the VLS object when the program is marked to history status.

Correcting Inefficient Block Size

The most efficient block size for a VLS library depends on the size of the members contained in the library. Therefore, it is a decision that you must make for each site and for each library.

Most often a site exceeds the space of a source library. Every VLS member in a user source library consists of at least two blocks, regardless of the actual size of the member. Unless almost every member is over three blocks, increasing the block size does not solve the problem on anything but a temporary basis, and increases the use of DASD and DSA.

Creating New Libraries for Existing Systems

If the VLS library contains more than one CA Ideal system, you can create another set of VLS libraries and separate the systems across the two sets of VLS libraries.

To migrate your systems with the least impact on your programmers

1. Back up the existing source, panel, and object libraries.
2. Create the new set of VLS libraries to contain source, panel, and object code.
3. Decide which systems should be moved to the new set of libraries.
4. Move all members associated with the selected systems into the new libraries and delete them from the existing library. You can use the VLSUTIL SELREST and DELETE cards to do this.
5. Update the CA Ideal system definitions to point to the new libraries.
6. Update the CICS JCL, the batch JCL, the FCT, and IDSYSFT to reflect the new libraries.
7. Recycle CICS.

This method has no impact on the existing programs. You do not need to edit or recompile anything. The programmers are unaware of the change.

Splitting One System into Separate Systems

If a VLS library only has one system and that system is full, you can split the system into separate systems. Before you do this you should:

- Check whether there are orphan members that you can delete
- Check whether changing the block size will prove beneficial

If either of the above suggestions does not provide a solution, create new CA Ideal systems. After the new systems are created, you can move programs as necessary.

To create new systems and migrate the programs into the systems

1. Determine which programs belong together as part of an application. You can identify programs that belong to one application, and common programs that are called by many applications, by generating a report with the command

   ```
   DISPLAY INDEX PROGRAM RELATED TO PROGRAM
   ```

   You can also generate a path report from the dictionary.

2. When you determine which applications you have, you can create the CA Ideal system definitions and the VLS libraries associated with those new systems. Be sure to add the new VLS libraries to the CICS JCL, batch JCL, FCT and IDSYSFT.

3. Use the Source Transport Utility to export the programs, reports, and panels from the existing system and import them into the new CA Ideal systems. Use the SET IMPORT NEW SYSTEM source transport command to change the system for the imported entities.

4. The entities are imported in test status and you must recompile the programs. You must mark production entities to production status.

   Note: The creation and compilation dates are changed to the date of import and recompilation.

5. In the original system, use the CA Ideal DELETE command to delete the programs, reports, and panels that were moved to the new system.

Library Integrity Utility

The Library Integrity Utility is a batch utility that reads the index of a VLS library and checks its members against the dictionary facility to verify the member's consistency.

Use the following command in your batch job stream to access the utility:

```
VERIFY LIBRARY library-name
```

The value of *library-name* is the name of the VLS library to verify.

The utility verifies the type of each member against the dictionary facility to make sure that the type can reside in that library.

The batch run produces a report identifying any members that do not match. For information about message explanations, see the *CA Ideal Messages and Codes Guide*. The library Integrity report is directed to UTLBCK (OS DDname or DOS DLBL) and will go to AUXPRINT if UTLBCK is not in the JCL.

Note: For more information about messages, see the *Messages and Codes Guide.*

Sample JCL

z/OS

```
//LIBINTEG EXEC IDLBATCH,PROG=IDUTILTY
//SYSIN       DD     *
VERIFY LIBRARY lib1
VERIFY LIBRARY lib2
/*
```

VSE

```
* $$ JOB JNM=LIBINTEG,USER='IDEAL',CLASS=0
* $$ LST CLASS=L,DISP=D
// JOB LIBINTEG
// OPTION LOG
// DLBL CAI,'NNNNNNNN'
// EXTENT ,XXXXXX
// LIBDEF *,SEARCH=(CAI.IDEAL,CAI.IPC,CAI.DB)
// EXEC PROC=IDLPROC
// EXEC IDUTILTY
VERIFY LIBRARY lib1
VERIFY LIBRARY lib2
...
/*
// EXEC LISTLOG
/*
/&
* $$ EOJ
```

Verification Considerations

You cannot verify the CA Ideal message library (default name ADRLIB), the CA Ideal panel library (default name ADRPNL), or the Print Subsystem output library (default name ADROUT). If you specify the above libraries, the command is rejected. You can verify the Dataview, Plan, and Data Member libraries (IDDVW and IDDAT) and any user Source, Object, or Panel libraries.

The checks that are made for a member depend on the type code, as shown in the following. list:

- **Type B-**PLAN. Can only be present on the Plan library as defined through the CA Ideal Options block assembly. By default, this is IDDVW.
- **Types D and K-**DATAVIEW object and source. Can only be present on the Dataview library, defined in the CA Ideal Options block and defaulting to IDDVW. For these types, a matching entity definition on the dictionary facility is required.

- **Types T and J-**PROGRAM object required for RUN. For these types, a PROD status system matching the system short id found in the member name must be present on the dictionary. The object library for the system must be the one being verified. The program name is checked in the dictionary, although where version is PRD, this is not required (object-transported programs). Where applicable, the object sequence codes are checked to ensure they are consecutive.
- **Type U-**PANEL. This member is required for RUN. The system short ID in the member name must match that of a system in PROD status in the dictionary. The panel library for the system must be the one being verified. There need not be a corresponding dictionary entity for a transported panel.
- **Types A, N, Q and V-**Compiled object components. Used by the CA Ideal incremental compilation process. The system short ID in the member name must match a system in PROD status on the dictionary. The object library for that system must be the one being verified. The program or panel must also be defined on the dictionary.
- **Types L, P, R and W-**Source components. The system short ID in the member name must match a system in PROD status on the dictionary. The source library for that system must be the one being verified. The program or report must also be defined on the dictionary.
- **Type Z-**Data Member. Can only be present on the data library defined in the CA Ideal options block (default IDDAT). The user short ID in the member name must match that of a user in PROD status on the dictionary.

Other types are either unrecognized or, as in the case of help members, on the wrong library. (Help members reside on the message library, which is not eligible for verification.)

If a non-Ideal member is added to a VLS library, it can cause internal errors in the utility. These members can easily be recognized in a VLSUTIL LIBRARY listing. If you suspect that a library is corrupt, check first using VLSUTIL delete illegally named members, and then check for consistency with the dictionary, using IDUTILTY.

Chapter 6: Considerations for CA Datacom/DB Native Access

This chapter details the considerations required for accessing CA Datacom/DB.

Index-Only Processing

Index-only processing means retrieving data from the permanent index without reading the related record. Index-only is a decision that CBS makes when it can obtain all of the information it needs from the index. There are several advantages to using index-only processing:

- CA Datacom/DB can scan the index without the I/O necessary read the data in the associated rows (records).
- CA Datacom/DB can perform partial key searches when only a high order portion of the key value is known.

Designing Keys for Index-Only Processing

To gain performance improvements, design the keys so they can be used for index-only processing. To take advantage of index-only retrieval when all of the data required by an application is not defined in the key, consider adding the additional columns (fields) as low-order columns of the key, especially if these columns are not updated frequently. The following rules apply:

- Dataview must consist of an element that equals a key.
- Key must encompass all fields needed for the application.
- Element must contain the key, meaning that all elements in the element list are contained in the key.
- Dataview or FOR construct must be read-only (NO UPDATE).
- All fields in the WHERE and ORDERED BY clauses must reflect the key design.
- Database cannot have multiple keys with the same key ID.

If the key does not include all needed fields, the data record might have to be accessed. You can retrieve data rows anyway, to build a temporary index and evaluate predicates in the CA Ideal WHERE clause. You can use the CBS Diagnostic Report ($$$) to determine whether index-only processing was actually done. If no data records were read, index-only is most likely used.

Sequential Processing

CA Ideal invokes the CA Datacom/DB GETIT and GSETL commands for optimum batch performance (GETIT and GSETL are not invoked for on-line processing) in instances where this optional processing method is specified. Better performance is attributed to optimized record retrieval due to the following conditions:

- Multi-blocked read ahead if traversal is by key.
- Blocked record transfer between Multi-User Facility (MUF) and the CA Ideal batch application.

The following conditions must exist before GETIT and GSETL sequential processing is invoked:

- Traversal must be by a key that is no more than 90 bytes in length.
- All records in the key range must satisfy the WHERE criteria.
- Internal data retrieval order must match the default order or the order explicitly specified in an ORDERED BY clause. You cannot specify UNIQUE in an ORDERED BY clause. You cannot specify the DESCENDING keyword.
- Record count in a FOR FIRST *n* statement must be greater than 10.
- WHERE clause must not contain the clauses NOT EQUAL, CONTAINS, or OR.
- Parameters in the active User Requirements Table (URT) required for GETIT are listed on the following page.

CA Ideal can use the multi-block read sequential processing capabilities of CA Datacom/DB if the data is organized where the nature key can retrieve records (the key that determines the physical order of the data when the data was loaded). This type of processing uses the CA Datacom/DB GSETL and GETIT commands. Processing is restricted to a batch environment and limited to the prime data area (the area initially written during the LOAD function). The GETIT command retrieves records sequentially, starting from the record marked by the GSETL command. Multi-block reads greatly reduce the number of EXCPs required to read a table. The following also impact sequential processing:

- GETIT and GSETL acquire locks on records by blocks. To release those locks, the entire block must be processed. We recommend that you use this only with a FOR EACH/ALL construct and that you do not include any QUIT statement in the FOR construct. This is a problem only if the URT entry for the file specifies UPDATE=YES.
- If an application has substantial add and delete activity, frequently reorganizing the table keeps the data in native key sequence and maintains efficiency.

- Examine the space management option specified for the data area against the application processing requirements for the site. Option 0 and Option 2 are the most efficient for GETIT processing. (The *CA Datacom/DB Database and System Administrator Guide* fully describes the data area space management options.)

To implement sequential processing of a CA Datacom/DB database in CA Ideal running under batch, an appropriate User Requirements Table (URT) is required.

The following DBURTBL macro parameters, required for sequential access, are described in the *CA Datacom/DB Database and System Administrator Guide*.

- **ELMCHG=**
- **GBMAXR=**
- **GETBLK=**
- **SEQBUFS=** Number of data buffers the program can use for block read-ahead sequential processing. Specify an even number of buffers.
- **UPDATE=YES** Code this only if updates are required, since this causes CA Datacom/DB to acquire locks on the records.

The following example illustrates the parameter values for a URT for a CA Ideal batch job doing sequential processing:

```
TITLE 'URT FOR EMPLOYEE DVW FOR GSETL/GETIT'

DBURSTR
      CSECT=GETITURT,
      TXNUNDO=NO
DBURTBL
      DBID=1,
      SYNONYM=YES,
      ELMCHG=NO,
      SEQBUFS=8,
      GETBLK=12288,
      GBMAXR=160,
      TBLNAM=PMF,
      UPDATE=YES
DBUREND
      USRINFO=IDEAL-BATCH-URT
END
```

Before issuing the RUN command in CA Ideal batch, the following command is required:

```
SET RUN URT GETITURT
```

GETITURT is the value of the CSECT specified in the CSECT= parameter above. It must match the load module name of the URT. See the *Command Reference Guide* for more information about the SET RUN URT command.

To verify that GSETL/GETIT commands are used, trace the batch run using the CA Ideal Dial Trace with code "V" and limit the run of a test program with SET RUN LOOPLIMIT 5 (enough to see several GETIT commands). The first request to the database is a SELFR command. CA Datacom/DB sets a flag in the request area that tells CA Ideal whether GSETL/GETIT commands can be used. If this occurs, dial trace output shows "GETIT MODE ACCEPTED." For more information about using the trace facility, see the *Problem Determination Guide*.

Test and Prod Data in Datadictionary and CA Ideal

It is possible to catalog a test status database structure into the CXX. You can restrict the use of production status in Datadictionary to those versions that represent the production or "live" implementation of the database. This means that at any stage in the development of a database structure, the various versions of the database might be alike.

You can catalog the Dataview entities representing the test status databases. You can use multiple versions of a dataview with the same name, but any given program can reference only one of them. Each dataview can represent a different DBID and can differ in any other way. Since the names of the dataviews are the same, you can compile a program to use any of these dataviews just by changing the version column in the program resources table. If the different versions are incompatible, there can, of course, be errors in some of the compilations.

When a new version of a database structure is copied to production status, the existing dataview entities in Prod are re-related to the new production elements. This can require the dataview to be recataloged in CA Ideal to pick up the changes in the underlying database. You should then recompile programs using the dataview. You do not need to change the program source, even in the Resource Table, because the dataview version is unchanged. If the dataview is not affected by the database changes, no recompilation is needed.

You can avoid recompiling if the Production dataview and a Test dataview differ only in the DBID being accessed. In that case, use the CA Ideal ASSIGN or ALTER commands in the program when it is moved between database versions. For example, you can choose to have a final-test database that is a duplicate of the live database. Programs tested against that database can have their DVW DBIDs altered to run against the live copy once the tests are complete, and the programs themselves promoted to production. Alternatively, the programs can be compiled using the Production dataviews and assigned to use the test DBID for development and final testing.

Note: Datadictionary lets you update any Test status entity, regardless of whether it is part of a database definition that was cataloged into the CXX. You have no guarantee that the definition in Datadictionary corresponds to the CXX, except for the Production version. Datadictionary uses the ENABLED attribute to flag the usability of the definition. CA Ideal checks this information when a Dataview is cataloged.

When any change is made to an entity through Datadictionary, Datadictionary automatically *disables* that entity. Do not reENABLE until the new information is transferred to the CXX and the new dataview information completed. ENABLE operates on an entire BAS or DVW structure, so this is a quick and easy process.

Exercise additional care when using the ASSIGN command. For integrity reasons, this should operate at a database level (ASSIGN DBID) and not table level (ASSIGN DVW) if any updates are performed. Overall, you can compromise database integrity if you use the ASSIGN DVW to point to the table that is in another copy of the database. In this case, you could have partial updates in two different occurrences of the same database. For example, Table A is updated in DB001 but Tables B and C are updated in DB100.

Use ASSIGN DVW for read-only dataviews, for example, allowing a test execution to read live look-up tables that are essentially static.

BACKOUT Statement Considerations

The following is a list of requirements that must exist to ensure that a table can successfully back out any changes that were made to the data. These are CA Datacom/DB requirements.

- User-coded ERROR PROCEDURE must explicitly code the BACKOUT statement for situations where backing out any updates is required.
- TXNUNDO URT parameter. If TXNUNDO=YES for a given URT CA Datacom/DB file and the other requirements for transaction backout were satisfied, the BACKOUT statement functions as described. If TXNUNDO=NO, the BACKOUT statement has no effect and no error message is issued.
- Specify the RECOVERY attribute of the TABLE entity-occurrence in the dictionary as Y (yes), indicating that CA Datacom/DB recovery facilities are used. You can print a CXX report to verify that this attribute was specified correctly. If so, the CXX report indicates RECOVER - YES.
- Specify the LOGGING attribute of the TABLE entity-occurrence in the dictionary as Y (yes), indicating that CA Datacom/DB logging is used. You can print a CXX report to verify that this attribute was specified correctly. If so, the CXX report indicates LOGGING - YES.
- CA Datacom/DB logging must not be turned off. DBUTILTY allows logging to be dynamically turned off and on, using the CXXMAINT ALTER LOGGING option. (You must specify RECOVERY as Y for this option to be honored.) You can print a CXX report to verify that logging is currently turned on.
- CA Datacom/DB master list must specify all options that enable logging and recovery. CA Ideal requires these capabilities. CA Ideal uses logging and transaction backout internally, and the CHECKPOINT and BACKOUT commands.

The *CA Datacom/DB Database and System Administrator Guide* contains a chapter on using the logging facility. It describes the log file in detail and how to set up the logging facility, which is required for the CHECKPOINT and BACKOUT.

INCLUDE-NIL-KEY

When defining a key in CA Datacom/DB, there is a parameter called INCLUDE-NIL-KEY. This nil value is not related to the null value. In most cases, you should define the INCLUDE-NIL-KEY as YES. When this parameter is set to NO, an alphanumeric key value of spaces or a numeric key value of binary zeroes is considered nil and is not included in the index. Although CA Ideal has no control over the processing that takes place when this parameter is used, its improper use can be the culprit of poor performance in CA Ideal applications.

You might assume that you should use INCLUDE-NIL-KEY=NO whenever keys have a high occurrence of the nil value to free up index space. In fact, you should limit the use of this parameter to specific situations:

- Use INCLUDE-NIL-KEY when it is not necessary to access records that have an alphanumeric key value of spaces or a numeric key value of binary zeroes.
- Use two-byte binary key field designated as DAYS-LATE with INCLUDE-NIL-KEY=NO. This keeps non-delinquent accounts with zero days late (a nil value) from being indexed. If only a small portion of the table contains delinquent accounts, a considerable savings results by eliminating both the cost of indexing and storing index entries for non-delinquent accounts.

Be aware of the consequences when choosing to use the INCLUDE-NIL-KEY=NO with CA Ideal. If the range of the search condition of a WHERE clause specifies the key field that has INCLUDE-NIL-KEY=NO and that range includes the nil value, then Compound Boolean Selection is forced to do a full file traversal to retrieve the data. You cannot use the key for access because the key cannot access all possible rows that satisfy the request.

Example

```
FOR ACCOUNTS
   WHERE DAYS-LATE LE 0
   . . .
ENDFOR
```

Because the nil value is included in the key range, CBS cannot use this key to access the data records. All records must be read to return the correct data to the program.

Adding a non-keyed field to the WHERE causes the construction of a temporary index for the same reason-you cannot use the key because it does not point to all possible records that could contain the non-key value.

If you define a key as INCLUDE-NIL-KEY=NO, be sure that it reflects the way the data is accessed and that the CA Ideal programmer is aware of this key and the proper coding techniques.

Chapter 7: Maintaining Plans for CA Datacom SQL Access

Access plans for programs that access CA Datacom SQL dataviews are generated automatically when you compile the program. You can set plan options, such as the cursor isolation level, date and time format, and the ordering of table joins, in the Environment fill-in of the program definition. Or you can allow them to default to site or session DBSQL options.

The plan options are included in the program object and are stored as intersection data in the PGM-PLN-USES relationship in the dictionary. The plan entity is stored in the PLAN table in the dictionary. You can change these options in the program object by specifying the ALTER PROGRAM ENVIRONMENT command, but to include those changes in the plan itself, you must rebind the plan.

You can define multiple plans for a program by defining additional authorization IDs for the program. You can then select the plan dynamically by assigning the authorization ID at runtime.

Note: DB2 plans are generated and maintained differently. For more information, see the “Preparing DB2 Application Plans” chapter.

Generating CA Datacom SQL Access Plans

For CA Datacom SQL access, a program can run under its own default-plan or under another assigned plan. The default plan is generated automatically when you compile the program, using the plan options set in either the Environment fill-in of the program definition or the site and session DBSQL options. You can also create alternate plans for a program by entering the DEFINE AUTHORIZATION command.

The plan name is composed of the authorization ID and the program name in the following format:

```
authid-$Ivvvsssprogram
```

authid

Authorization ID specified in either the Environment fill-in or the DEFINE AUTHORIZATION command.

$I

Supplied by the system, it indicates that the plan is for a CA Ideal program.

vvv

Version of the program.

sss

System where the program resides.

program

Name of the program.

When the plan is generated, it is stored as an entity in the PLAN table in the dictionary. The plan options are stored as intersection data in the PGM-PLN-USES relationship in the dictionary. (The program object also includes the plan options.) To change the plan, you must either recompile the program or rebind the plan. When you create an alternate plan for the program with the DEFINE AUTHORIZATION command, the plan options are copied from the PGM-PLN-USES relationship. For more information about creating alternate plans, see Maintaining Access Plans for the Runtime Environment in this chapter.

Generating the Default Plan

When you compile a program that accesses a CA Datacom SQL dataview, the default plan is generated automatically. If any plans already exist for the program, they are deleted. Deleted plans are listed in the compilation listing. After the existing plans are deleted, the default plan is regenerated for the program.

Note: No matter how many plans existed for a program, only one plan, the default plan, exists for that program after it is recompiled. If you require alternate plans for the program, you must define those plans again yourself.

If a program has several alternate plans or if an alternate plan has many plan options that differ from the default plan, you might want to create a member containing all of the commands required to generate the alternate plans for that program. You can execute this member after the program is recompiled. This protects you from inadvertently leaving out one of the required plan definitions after recompiling the program.

Setting Plan Options

The programmer can set plan options in the Environment fill-in of the program definition or by changing the session options with the SET DBSQL command. As the site administrator, you can set DBSQL options for the site using the SET SITE DBSQL command. However, the session options override the site options and the options set in the Environment fill-in override both site and session options.

You can also change the plan options for a specified program after it is compiled with the ALTER PROGRAM ENVIRONMENT command (see Changing the Access Plan in this chapter). However, to incorporate those changes into the plan, you must rebind the plan.

You can set the following plan options in the program Environment fill-in. The equivalent SET command that you can use to set the option for the session or site is shown after each option.

Default Auth-ID

One- to eight-character authorization ID that identifies the program's access plan.

You can also set this option with the following command:

```
SET [SITE] DBSQL AUTH auth-id
```

SQL Mode

The mode in which CA Datacom processes the program. The values can be any of the following:

- **ANSI86**-SQL must follow ANSI86 standards
- **Datacom**-SQL must follow CA Datacom standards
- **FIPS**-SQL must follow FIPS standards
- **DB2**-SQL must follow DB2 standards (this supports CA Datacom Database Transparency Option for DB2)

You can also set this option with the following command:

```
SET [SITE] DBSQL MODE sqlmode
```

For more information about definitions of the SQL modes, see the CA Datacom SQL *Programming and Reference Guide*.

Cursor Isolation Level

Degree to which a unit of recovery is isolated from the updating operations of other units of recovery. Cursor stability is required for updates, deletes, or inserts:

- **U**-No locks are acquired
- **C**-Cursor stability
- **R**-Repeatable read (CA Datacom Release 8.1 only)

You can also set this option with the following command:

```
SET [SITE] DBSQL ISOLATION-LEVEL level
```

Optimization Mode

The mode in which CA Datacom optimizes table joins:

- **Preptime**-(Default) Order joins during bind processing
- **Manual**-Order joins as specified in FROM clauses
- **Exectime**-Order joins at execution time. (See the CA Datacom/DB documentation for more information).

You can also set this option with the following command:

```
SET [SITE] DBSQL OPTMODE optmode
```

Date Format

Display format for SQL date type items:

- **DB**-CA Datacom default. The CA Datacom site administrator can set it to ISO, USA, EUR, or JIS.
- **ISO**-International Standards Organization yyyy-mm-dd
- **USA**-U.S. standard mm/dd/yyyy
- **EUR**-European standard dd.mm.yyyy
- **JIS**-Japanese Industrial Standard yyyy-mm-dd

You can also set this option with the following command:

```
SET [SITE] DBSQL DATE date-format
```

Time Format

Display format for SQL time type items:

- **DB**-CA Datacom default. The CA Datacom site administrator can set it to ISO, USA, EUR, or JIS.
- **ISO**-International Standards Organization *hh.mm.ss*
- **USA**-U.S. standard *hh:mm* AM or PM
- **EUR**-European standard *hh.mm.ss*
- **JIS**-Japanese Industrial Standard *hh:mm:ss*

You can also set this option with the following command:

```
SET [SITE] DBSQL TIME time-format
```

CBSIO

I/O limit interrupt value for SQL statements that creates a set.

You can also set this option with the following command:

```
SET [SITE] DBSQL CBSIO nnn
```

Priority

Priority of the SQL requests:

- **1-15**-The lowest priority is 1 and the highest priority is 15.

You can also set this option with the following command:

```
SET [SITE] DBSQL PRIORITY nn
```

Wait Time

Exclusive control wait limit.

- **1-120**-Number of seconds or minutes
- **unit**-S for seconds or M for minutes.

You can also set this option with the following command:

```
SET [SITE] DBSQL WAIT nnn unit
```

Preptime Optimization Messages

Type of optimization messages CA Datacom produces during bind processing.

- **N**-None (default)
- **D**-Detail
- **S**-Summary

You can also set this option with the following command:

```
SET [SITE] DBSQL OPTMSGS PREP type
```

Exectime Optimization Messages

Type of optimization messages CA Datacom produces at runtime.

- **N**-None (default)
- **D**-Detail
- **S**-Summary

You can also set this option with the following command:

```
SET [SITE] DBSQL OPTMSGS EXEC type
```

DB/SQL Workspace

Amount of workspace available at plan execution time. (Used for error correction.) This value is multiplied by 1024 to determine the number of bytes to allocate.

- **0-128**-0 to 128 bytes, inclusive.

You cannot set this option with a SET DBSQL command. However, you can change it with the ALTER PROGRAM ENVIRONMENT command.

Maintaining Access Plans for the Run-Time Environment

When you are ready to move a program that accesses a CA Datacom SQL dataview into the production environment, you might want to change some of the plan options. You can do this without recompiling the program by entering the ALTER PROGRAM ENVIRONMENT command. You can then rebind the plan with the REBIND command to change the default plan.

If you want to be able to select a different plan dynamically at runtime, you need to create the alternate plan with the DEFINE AUTHORIZATION command. You can then enter the ALTER PROGRAM ENVIRONMENT command to change the plan options the program uses and enter the REBIND command with the new authorization ID to rebind the new plan.

To select the plan at runtime, use the ASSIGN AUTH command before running the program.

The following sections explain each of these commands.

Changing the Access Plan

To change the access plan, you need to perform the following activities.

Changing Plan Options

To change the plan options in the program object without editing and recompiling the program, you can enter the ALTER PROGRAM ENVIRONMENT command with the appropriate option.

The format of the ALTER PROGRAM ENVIRONMENT command is:

```
ALTER PROGRAM name [VERSION ver] ENVIRONMENT [option]
```

For a complete explanation of the ALTER PROGRAM ENVIRONMENT syntax, see the *Command Reference Guide*.

You can change all of the options listed in Setting Plan Options earlier in this chapter with this command, except for the authorization ID and the SQL mode. You can also specify the CLOSE option on the ALTER PROGRAM ENVIRONMENT command to determine when to close the plan. Values that you can enter for the CLOSE option follow:

RUN

Closes the plan at the end of the run-unit or CICS job. CA does not recommend it for online programs since you cannot recompile the program until the plan is closed.

TRAN

Closes the plan after each database transaction.

If you want to change more than one option, you can enter the ALTER PROGRAM ENVIRONMENT command without specifying any options. This displays a fill-in showing the current options from the program object and the CLOSE option. The current authorization ID and SQL mode also display, but you cannot change it.

Note: Do not execute this command for a program that was the object of a CREATE MODULE command. If you need to alter the plan options for a program that is executed from a load module, you must enter the ALTER PROGRAM ENVIRONMENT command before creating the load module. You can rebind the plan after you create the load module.

Rebinding the Plan

If you changed the plan options with the ALTER PROGRAM ENVIRONMENT command, you must rebind the plan to regenerate the plan using the changed options. If you need to change an alternate plan or regenerate the default plan without recompiling (for example, in a production environment where you do not have the program source), you must use the REBIND command.

The format of the REBIND command is:

```
       {  *                            }
REBIND {PROGRAM name [VERSION ver]     }{FOR AUTHORIZATION AUTHORIZATION}
```

If you do not specify AUTHORIZATION, the authorization ID of the default plan (stored in the program object) is used. For a complete explanation of the REBIND command syntax, see the *Command Reference Guide*.

The REBIND command only affects the plan identified with the specified (or implied default) authorization ID, version, and program name. If you issue an ALTER PROGRAM ENVIRONMENT command against a program associated with more than one plan, you must issue the REBIND command for each plan that requires the changed plan options.

Note: If the specified program was converted to load module format, the REBIND command uses the load module. To change plan options without recompiling the program, you must enter the ALTER PROGRAM ENVIRONMENT command before creating the load module. You can enter the REBIND command after the CREATE MODULE command to regenerate the plan.

Creating an Alternate Plan

To create an alternate plan, use the DEFINE AUTHORIZATION command. The format of the DEFINE AUTHORIZATION command is:

```
DEFINE AUTHORIZATION authid FOR PROGRAM name VERSION ver
```

For a complete explanation of the DEFINE AUTHORIZATION syntax, see the *Command Reference Guide*.

The DEFINE AUTHORIZATION command creates a new plan using the authorization ID, program name, and version specified in the command to create the new plan name. You must enter the command from the system that contains the program object. This system name is used in the new plan name.

The only thing that changes for the new plan is the authorization ID. The SQL statements are extracted from the existing program object. In the new plan, statements that reference resources defined in the program resource fill-in as QUAL=N are referenced with the authorization ID specified in the DEFINE AUTHORIZATION command. All other references in the new plan remain the same as in the original plan.

Note: The statement-IDs for each statement must match in every plan associated with the program or a runtime error occurs. This type of error can happen if the CA Datacom utility DDTRSLM exports a plan without its associated program or if a program object is transported without its associated plan.

If a plan already exists with the same name (including the authorization ID), that plan is deleted before the new plan is created. The new plan then takes the place of the old one.

If you need to change plan options for the new plan, use the ALTER PROGRAM ENVIRONMENT command to change the program object after creating the new plan. Then enter the REBIND command, specifying the authorization ID of the new plan on the REBIND command.

Selecting an Alternate Plan at Runtime

If a program was defined with more than one plan, you can select the appropriate plan dynamically at runtime by entering the ASSIGN AUTH command before executing the program.

The format of the ASSIGN AUTH command is:

`ASSIGN AUTH` *`authid1`* `NEW` *`authid2`*

- **authid1**-Authorization ID of the default plan.
- **authid2**-Authorization ID of the alternate plan. This authorization ID is sent with the program name, version, and system to identify the required plan when the SQLCA is passed to CA Datacom. As a result, the statements in the alternate plan execute instead of the statements in the default plan.

Note: The plan identified with the specified authorization ID must exist in the current dictionary for the program to run with that plan.

Chapter 8: Preparing DB2 Application Plans

You can optimize the execution of SQL in CA Ideal applications, either embedded SQL or SQL generated by FOR constructs, by using an application plan that can execute the SQL in static mode. You can execute SQL not executed as part of an application plan in CA Ideal in dynamic mode. Generally, a program is tested in dynamic mode and, when it is ready for performance testing or production, it is moved into static mode.

Program Modes

To run a program in dynamic mode, first compile the program like any other CA Ideal program and then run it. CA Ideal runs the program under its own default plan, typically IDP140DV.

To run a program in static mode, DB2 requires that you prepare an application plan that defines the program's authorizations and access paths. You can prepare the application plan from both packages and DBRMs (Database Request Modules). A package contains SQL statements that were already bound. A DBRM contains SQL statements that were not bound. If you prepare the plan from DBRMs only, you must rebind the plan when you change any program that is included in the plan. This can be extremely resource consumptive. If you prepare the plan from packages, you need only rebind the associated package when you change a program. You do not need to rebind the plan. This minimizes the impact of the bind since fewer resources are tied up during the bind.

To run in static mode, compile the CA Ideal program like any other CA Ideal program. Then complete the following steps to prepare the program for execution in static mode and bind a plan.

1. Create a package definition, if appropriate

   CA Ideal provides fill-ins and commands for creating and maintaining package definitions.

2. Generate the package

   A batch command, GENERATE PACKAGE, automatically prepares the program for execution in static mode and invokes the DB2 BIND command. This procedure includes preparing one CA Ideal SQL module (an Assembler program containing the SQL for the DB2 precompile) and performing the DB2 precompile, assembly, link-edit, and bind.

3. Create the application plan definition

   CA Ideal provides fill-ins and commands for creating and maintaining application plan definitions.

4. Generate the application plan

   A batch command, GENERATE PLAN, automatically prepares any programs included in the plan for execution in static mode and invokes the DB2 BIND command. This procedure includes preparing one or more CA Ideal SQL modules (assembler programs containing the SQL for the DB2 precompile) and performing the DB2 precompile, assembly, link-edit, and bind. If the plan includes packages, they are included only in the bind.

5. Connect the application plan to the application for the CICS environment

   In CA Ideal, you can specify which plan an application uses. This connection must also be made using DB2TRAN and DB2ENTRY elements in CICS.

   Before you run an application, you can specify whether you require all programs to run in static mode or allow programs to run in dynamic mode. Use the SET RUN SQL command to do this.

PF Key Assignments for the Plan and Package Editors

The following table shows the PF key assignments in effect when defining plans and packages. The following commands are assignments consistent throughout all facilities of CA Ideal.

HELP (PF1/13)

Displays a panel or series of panels that contain information that explains how to complete the current function.

RETURN (PF2/14)

Returns from a help panel to the plan component display or from the plan to the menu used to select the plan.

PRINT SCREEN (PF3/15)

Generates a hardcopy printout of the current screen contents.

PARAMETER (PF4/16)

Positions to the plan's parameter fill-in.

RESOURCES (PF5/17)

Positions to the plan's resources fill-in.

DBRM (PF6/18)

Positions to the plan's DBRM fill-in.

SCROLL BACKWARD (PF7/19)

Displays the previous frame in the current component.

SCROLL FORWARD (PF8/20)

Displays the next frame in the current component.

FIND (PF9/21)

Finds the next occurrence of an alphanumeric literal previously specified in a full FIND command.

SCROLL TOP (PF10/22)

Positions to the first line of the component.

SCROLL BOTTOM (PF11/23)

Positions to the bottom of the component.

INPUT (PF12/24)

Opens a window of null lines preceding the first line of the component or at the current cursor position. Unused null lines in the window are deleted when you press the Enter key after INPUT.

Defining Application Plans

This section describes the fill-ins and commands for creating and maintaining application plans.

Components of a CA Ideal Plan Definition for DB2

Each application plan definition includes:

- Plan identification fill-in that creates the plan definition and provides it with identification information
- Plan parameters fill-in that specifies DB2 BIND command options
- Plan resources fill-in that specifies the CA Ideal programs directly included in the plan
- Package list fill-in for programs that have been bound individually as packages (see later in this section)
- DBRM fill-in that specifies DBRMs for non-Ideal programs

The CREATE PLAN command creates the components of the application plan definition. The EDIT PLAN command updates the components. You can also use the commands DISPLAY, PRINT, DUPLICATE, and DELETE with plan definitions. For complete syntax, see the *Command Reference Guide*.

This section contains explanations of how to complete the fill-ins that construct each component of a plan definition presented in the Plan Menu.

The Plan Menu describes the functions that CA Ideal provides to define and maintain plan definitions. To access this menu, select option 5 on the Main Menu or enter the PLAN command as follows:

```
=>
=>
=>
----------------------------------------------------------------------------
IDEAL: PLAN MAINTENANCE      PLA               SYS: DOC       MENU
 Enter desired option number===>      There are   6 options in this menu:
 1. EDIT/DISPLAY       - Edit or display a plan
 2. CREATE             - Create a plan
 3. PRINT              - Print a plan
 4. DELETE             - Delete a plan
 5. DUPLICATE          - Duplicate a plan to new name
 6. DISPLAY INDEX      - Display index of plan names in system
```

The batch command GENERATE PLAN generates the plan and prepares the program for execution. You can use the REFRESH PLAN command in the CICS environment after a plan was regenerated to make the plan available to the CICS environment. For details, see the Generating Application Plans topic in this chapter.

How to Create a New Plan Definition

To create a new plan definition, issue the CREATE PLAN command and complete the plan identification fill-in. Until the plan is named either in the command or in the fill-in, the new definition does not exist.

Identification Fill-in

On the plan identification fill-in screen, enter descriptive information about the plan when you create or modify the plan definition.

```
=>
=>
=>
----------------------------------------------------------------------------
IDEAL: PLAN IDENTIFICATION   PLA IQPLANA                  SYS: DOC   DISPLA

PLAN IQPLANA

Created:        12/10/04               By MARCHAND
Last Modified:  01/21/05 at 13:11      By MARCHAND

Short Description:  Static QA


Description:
     ________________________________________________________________
     ________________________________________________________________
     ________________________________________________________________
     ________________________________________________________________
```

The following list provides an explanation of the fields on the preceding screen:

- **PLAN name**-The one- to seven-character name assigned to the plan definition.
- **Created ... By**-The initial creation date of the plan definition and the user ID of the creator. This is blank until the definition is accessed in edit mode. The system supplies this information. The user cannot modify it.
- **Last Modified ... at ... By**-The date, time, and user ID of the last edit access. This is blank until the definition is accessed in edit mode. The system supplies this information. The user cannot modify it.
- **Short Description**-An area where the user can supply a description of the plan definition. This description is limited to 24 characters.
- **Description**-An area where the user can supply a longer description of the plan definition. This description is limited to 5 lines.

Plan Resources Fill-in

To specify the CA Ideal programs that are included in the plan definition, issue the RESOURCE command or press the PF5/17 key. The following plan resources fill-in screen appears.

```
=>
=>
=>
-------------------------------------------------------------------------------
IDEAL: PLAN RESOURCES        PLA IQPLANA                    SYS: DOC   FILL-IN

Specify the programs, system, version and type that the Plan includes.

Command   Program    Sys   Type   Comments
-------   --------   ---   ----   -------------------------------------
======    ========   ===   ===    ======= T O P =====================
000010    MAINAP     ARI   APP    ______________________________________
000011    SQLFET     ARI   APP    ______________________________________
000012    REPORT1    ARI   PGM    ______________________________________
======    ========   ===   ===    ==== B O T T O M ===================
```

The following list provides an explanation of the fields on the preceding screen:

Program

Identifies the name of a CA Ideal program. Non-ideal programs are specified on the DBRM fill-in.

Sys (Optional)

Identifies the three-character CA Ideal system ID. (Defaults to the current system.)

Type

Specifies whether the plan includes just the program or the program and all of its CA Ideal subprograms that issue SQL statements.

- **PGM**-Includes the specified program only. (Any subprograms that perform DB2 access must be named separately in the fill-in, be part of a different application plan, be non-Ideal, or run dynamic SQL.)
- **APP**-Includes all CA Ideal subprograms of the specified program. (DBRMs for non-Ideal subprograms must be specified on the DBRM fill-in.)

Comments

Information displayed with the plan definition. It has no effect on the plan generation or on the resulting plan.

Packlist Fill-in

Issue the PACKLIST command (or PKL) to display the PACKLIST fill-information for the current plan. The packlist fill-in specifies the CA Ideal packages that are included in the plan definition.

```
=>
=>
=>

-------------------------------------------------------------------------------
IDEAL: PLAN PACKLIST        PLA PL#2806                     SYS: $ID   DISPLAY

        Specify the packages and collections that this plan includes.

Command  I   Package     Collection           Comments
-------  -   --------    ------------------   ------------------------------
======   =   ========    ==================   ===== T O P ==================
001400   I   SQL2806     __________________   ______________________________
001500   I   PK#2806     PK#COLLECTION        ______________________________
001600   _   COB#2806    COBOL#COLLECTION     ______________________________
======   =   ========    ==================   === B O T T O M =============
```

The following list provides an explanation of the fields on the preceding screen:

I

Enter I for CA Ideal or any other character for a non-Ideal package.

Package

Enter the one- to seven-character name of the CA Ideal package or the one- to eight-character name of the non-Ideal package/DBRM. To include all packages in a collection, specify “*”.

Collection

The 1- to 18-character name of the collection. This is required for a non-Ideal package but is optional for CA Ideal. Must be supplied if the package was not already successfully generated.

Comments

Any comments to appear on the plan listing. Data entered here does not affect plan generation.

Plan DBRM Fill-in

Issue the DBRM command or press PF6/18 to display the DBRM fill-in for the current plan. This fill-in lets you specify DBRMs for non-Ideal programs that are included in the plan definition.

```
=>
=>
=>
----------------------------------------------------------------------------
IDEAL: PLAN DBRM             PLA IQPLANA                 SYS: DOC   FILL-IN

Specify the external (non-Ideal) DBRM names that this plan includes.
Command  DBRM      Comments
-------  --------  ---------------------------------------------------------
======   ========  =========================================================
000100   BATCH     Optional for batch runs
======   ========  =========================================================
```

The following list provides an explanation of the fields on the preceding screen:

DBRM

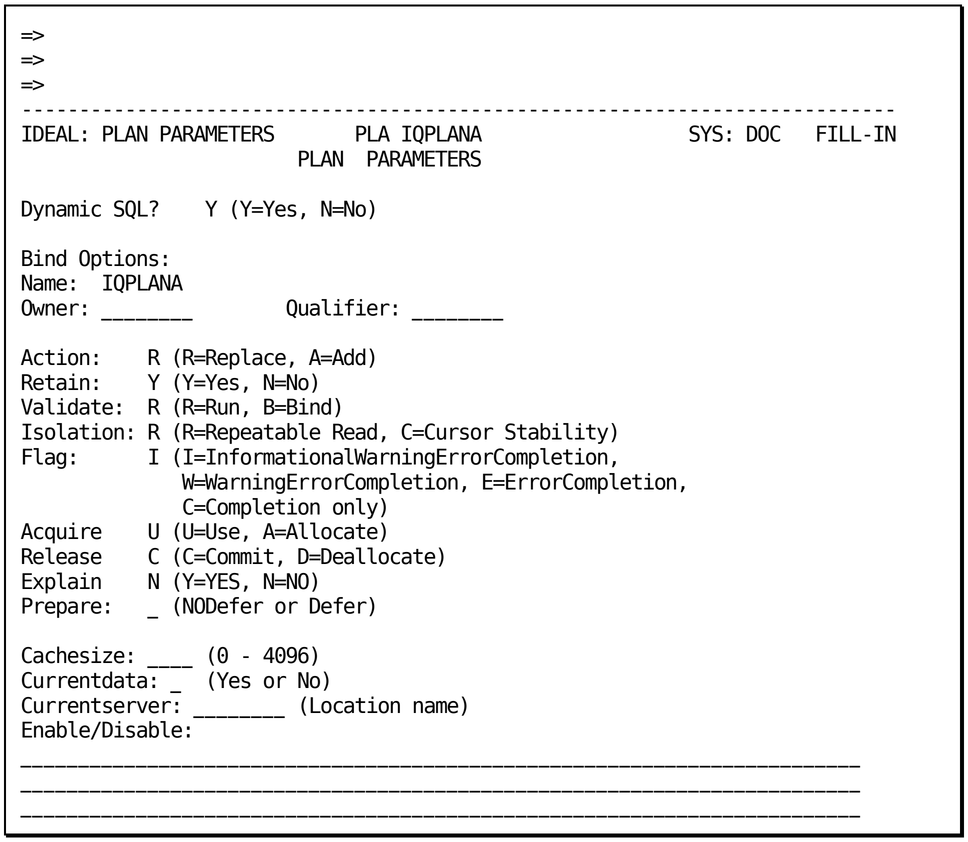
The name of a DBRM for a non-Ideal program included in this plan. Include the name of the DBRM for every non-Ideal subprogram that contains SQL and is included in this plan.

Comments

Information displayed with the plan definition. It has no effect on the plan generation or on the resulting plan. You can continue comments longer than 30 characters on the next line by ending the current line with a continuation character. The comment can be up to 52 characters long.

Parameters Fill-in

Issue the PARAMETERS command or press PF4/16 to display the parameters fill-in for the current plan. This fill-in lets you set DB2 BIND command options and allows programs in this plan to execute SQL in dynamic mode. The current settings display as one-character codes that you can change by overtyping.



```
=>
=>
=>
----------------------------------------------------------------------------
IDEAL: PLAN PARAMETERS       PLA IQPLANA                 SYS: DOC   FILL-IN
                        PLAN  PARAMETERS

Dynamic SQL?    Y (Y=Yes, N=No)

Bind Options:
Name:  IQPLANA
Owner: ________        Qualifier: ________

Action:    R (R=Replace, A=Add)
Retain:    Y (Y=Yes, N=No)
Validate:  R (R=Run, B=Bind)
Isolation: R (R=Repeatable Read, C=Cursor Stability)
Flag:      I (I=InformationalWarningErrorCompletion,
              W=WarningErrorCompletion, E=ErrorCompletion,
              C=Completion only)
Acquire    U (U=Use, A=Allocate)
Release    C (C=Commit, D=Deallocate)
Explain    N (Y=YES, N=NO)
Prepare:   _ (NODefer or Defer)

Cachesize: ____ (0 - 4096)
Currentdata: _  (Yes or No)
Currentserver: ________ (Location name)
Enable/Disable:
_________________________________________________________________________
_________________________________________________________________________
_________________________________________________________________________
```

The following list provides an explanation of the fields on the preceding screen:

Dynamic SQL?

Specifies whether this plan includes any programs executing SQL in dynamic mode. (This is a CA Ideal setting, where all other settings on this fill-in are DB2 BIND command options.)

- **N**-Application cannot execute SQL in dynamic mode.
- **Y**-Application can execute SQL in dynamic mode.

Note: It is suggested that you set this option to Y, which allows dynamic SQL to change and test one program using dynamic SQL as part of an otherwise static application without having to rebind the plan.

Setting this option to Y includes the DBRM provided with CA Ideal to handle SQL in dynamic mode.

If you specify that the plan includes dynamic SQL in this fill-in, you can override this and require that all programs be static by using the SET RUN SQL STATIC command. See the *Command Reference Guide* for complete syntax.

Name

Specifies name of the plan in plan identification fill-in screen.

Owner

Specifies authorization ID of the owner of the plan. The owner must have the privileges required to execute the statements contained in the plan.

Qualifier

Specifies the authorization ID, which is the qualifier for all unqualified DB2 objects accessed by this plan. An authorization ID specified at runtime can override this value in an ASSIGN AUTH command.

Action

Specifies whether this application plan is new or replaces an old plan.

- **R-**Replace old plan
- **A-**Add new plan

Retain

Indicates whether those users with authority to bind or execute the existing plan keep that authority over the replaced plan.

- **Y**-Retain authority
- **N**-Do not retain authority

Validate

Indicates when full validity checking is performed.

- **R**-At runtime
- **B**-At bind time

Isolation

Specifies the isolation of this application from others.

- **R**-Repeatable read
- **C**-Cursor stability

Flag

Indicates the lowest le4vel of messages to show.

- **I**- Informational, warning, error, and completion
- **W**-Warning, error, and completion
- **E**-Error and completion
- **C**-Completion only

Acquire

Indicates when you want the system to acquire the resources that your program used.

- **U**-Open table spaces and acquire locks when your application first accesses them.
- **A**-Acquire them when the application plan is allocated.

Release

Indicates when you want the system to release the resources that your program used.

- **C**-Release resources at each commit point.
- **D**-Release them when the application terminates.

Explain

Indicates whether to include the explain option.

- **Y**-Include the explain option.
- **N**-Do not include the explain option.

Prepare

Specifies when the PREPARE for an SQL statement that refers to a remote object takes place.

- **N**-PREPARE takes place in real time.
- **D**-PREPARE occurs when the first EXECUTE, OPEN, or DESCRIBE for the statement is issued.

Cachesize

Indicates the size of the authorization cache to acquire for the plan. The authorization cache is used at runtime to store the names of users authorized to run the plan.

- **0-4096**-Size in bytes for the authorization cache.

Currentdata

Specifies the data currency required for ambiguous cursors opened at remote locations. Data is considered current if the data in your host structure is identical to the data in the base table.

- **Y**-Data currency is required for ambiguous cursors.
- **N**-Data currency is not required for ambiguous cursors.

Currentserver

Specifies a connection to a location before the plan is run. The specified location receives all SQL requests until the application issues the SQL statement CONNECT TO or CONNECT RESET.

Enable/Disable

These keywords are mutually exclusive. ENABLE lists the system connection types that are enabled to use the plan. DISABLE lists the connection types that are disabled for the plan. You can use one to three lines to enter the value for this parameter.

Generating Application Plans

The application plan definition generates the plan and prepares applications for execution. If you recompile a program that is included directly in the plan, not in a package, and want it to run in static mode, you must regenerate the plan. If the program is included in a package, you must regenerate the package, but it is not necessary to regenerate the plan.

If you regenerate a plan while CICS is active, use the REFRESH PLAN command to make a new copy of the CA Ideal Static I/O modules available to CICS.

For information about the syntax of these commands, see the *Command Reference Guide*.

Issue the GENERATE PLAN command in batch. The JCL needed for plan generation is in the procedure IDPLAN, which you can use to generate both plans and packages. You must specify the library containing any
non-Ideal DBRMs in the JCL. Following is a sample jobstream:

```
//name EXEC IDPLAN
//SYSIN DD *
GENERATE PLAN AP3
/*
//
```

The IDPLAN procedure performs the following steps:

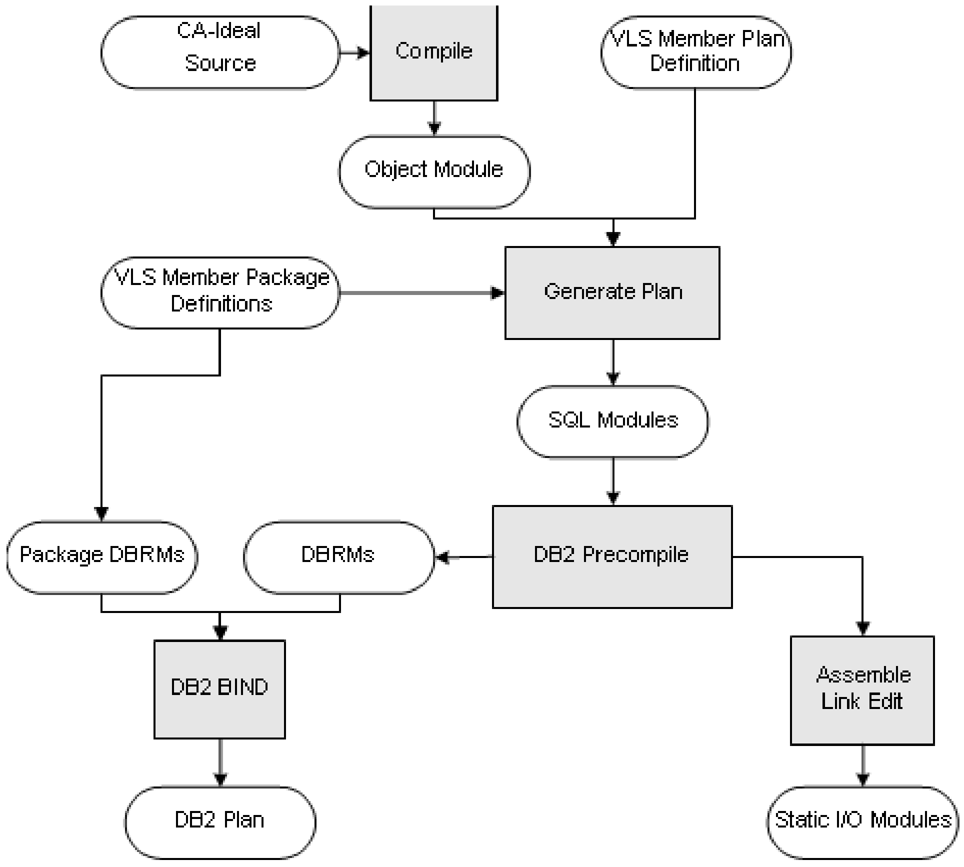
1. Generates CA Ideal Static I/O modules.
2. Precompiles DB2 SQL.
3. Assembles.
4. Link-edits.
5. DB2 BINDs.

If the plan contains any packages, the header information for the package static I/O modules is placed in the first static I/O module generated for the plan. You can bind the plan, even if packages included in the plan were not bound.

If a plan contains only packages, one static I/O module is created for the plan, but it contains only header information for the package static I/O modules, and the global COMMIT, ROLLBACK and SET CURRENT USER statements. The BIND command is generated based on the resources of the plan.

You can transport DB2 applications, package definitions, and plan definitions from the development site to the production site using the CA Ideal transport utilities (see the *Working in the Environment Guide*). You can transport DBRMs using IBM utilities.

The following diagram illustrates the plan preparation process.



Note: If dynamic SQL is allowed on the plan parameters fill-in, an application or program that would normally run in static mode runs in dynamic mode in the following circumstances:

- When an application with static SQL is executed, it runs in dynamic mode if its plan was not generated using the GENERATE command or the CA Ideal Static I/O modules cannot be loaded.
- If a program from an application with a generated plan is recompiled without re-executing the GENERATE command, that program runs in dynamic mode while the rest of the application runs in static mode.

Plan Generation JCL

The JCL required by the GENERATE PLAN command is included in the IDPLAN procedure. You can modify it for your site. The statements are grouped by the process they perform such as, precompiler, assembler, link-edit, terminal monitor. Statements you might need to modify are:

- **SQLMOD**-Data sets where source for the Static I/O modules is placed.
- **IDUDBRM**-Specifies the data set containing all user DBRMs (CA Ideal and non-Ideal).
- **ADRDBRM**-The DSN where the DBRMs CA Ideal supplies are located. These DBRMs include the dynamic SQL DBRM.
- **SYSLMOD**-The load library containing the CA Ideal SQL load modules.

CA Ideal Static I/O Modules

The Static I/O Modules, created as part of the CA Ideal GENERATE PLAN procedure, are Assembler subprograms that CA Ideal calls during static execution of SQL. These subprograms, which contain the SQL statements from the programs specified in the plan resource definition, are passed to the DB2 precompiler.

The GENERATE PLAN command creates up to 10 Static I/O Modules. At one time the DB2 precompiler imposed a limit on the number of source lines that could be processed, so CA Ideal provided a facility to limit the number of SQL statements that would be used to generate each module. When the SQL content of the next program would exceed this maximum, another module is created. This facility still serves a purpose in limiting the size of the static I/O Static I/O Modules. To enable this process, use the SET PLAN MAXSQL command. This command sets the maximum number of SQL statements allowed in each module

Note: Since the static I/O module for a package is created during a package generation, not a plan generation, any MAXSQL value specified for the GENERATE PLAN does not affect the package static I/O module, only the modules created for the plan.

The name of each Static I/O module is the one- to seven-character plan name, with an eighth character of 0 for the first module, and 1-9 for the second through tenth modules.

If the plan includes any packages, the first static I/O module contains header information for the static I/O modules for the packages. If the plan includes only packages, the (single) static I/O module contains only the global COMMIT, ROLLBACK, and SET CURRENT USER statements.

Note: The load library that the GENERATE PLAN procedure uses as SYSLMOD must be available to any CICS partition DFHRPL or batch STEPLIB where the applications associated with the plan are expected to execute in static mode.

CA Ideal Plan Tables

Another output of the GENERATE command are updates to three DB2 tables. The rows in these tables record all programs participating in the plan and any changes made by the ASSIGN AUTHID command. You can query these tables using SQL (in the same way that you query other tables) to answer questions such as, "Which plan does program x participate in?"

The "SYSADR Table Declarations for DB2" appendix illustrates the layout of these tables, named SYSADR.APTAB, SYSADR.APRES, and SYSADR.APAUT, respectively.

Impact Report

The report that the generation process produces has three parts.

- The components of the plan definition and the programs involved in the plan.
- The generated BIND command text, including all DBRMs.
- Error and completion messages.

You can produce this report without actually doing the BIND or updating the plan tables by using the VERIFY option on the PRINT PLAN command. For information about the complete syntax, see the *Command Reference Guide*.

The first section of the report shows the plan information entered in the plan definition. It is identical to the information a PRINT PLAN command produces.

```
                         IDEAL APPLICATION PLAN/PKG REPORT
          RELEASE:  14         DATE: March 10, 2010     TIME: 15:27:49
                        GENERATE APPLICATION PLAN/PKG DB2PLN
 IDEAL Application Plan/Pkg DB2PLN       March 10, 2010     15:27:49
PLAN:     DB2PLN ________
                   PLAN DB2PLN
                   Created:        11/01/08              By $ID
                   Last Modified:  03/10/09 at 15:01      By $ID
                   Short Description:  uses a package
                   Description:
                    _______________________________________________________
IDEAL Application Plan/Pkg DB2PLN       March 10, 2010     15:27:49
PLAN:     DB2PLN ________
                   Dynamic SQL?    N (Y=Yes, N=No)
                   Bind Options:
                   Owner: WESJ001             Qualifier:
                       Action:    R (R=Replace, A=Add)
                       Retain:    Y (Y=Yes, N=No)
                       Validate:  R (R=Run, B=Bind)
                       Isolation: R (R=Repeatable Read, C=Cursor Stability)
                       Flag:      I (I=InformationalWarningErrorCompletion,
                                     W=WarningErrorCompletion, E=ErrorComple
                                      C=Completion only)
                       Acquire    U (U=Use, A=Allocate)
                       Release    C (C=Commit, D=Deallocate)
                       Explain    N (Y=Yes, N=No)
                       Prepare    N (N=Nodefer, D=Defer)
                       Cachesize:  0512 (0 - 4096)
                       Currentdata:  Y (Y=Yes, N=No)
                       Currentserver:           (Location Name)
                       Enable/Disable:
 IDEAL Application Plan/Pkg DB2PLN       March 10, 2010     15:27:49
PLAN:     DB2PLN ________
                  ------    --------  ----  ---   ----   -------------------
                  SEQ       PROGRAM   VER   SYS   TYPE   COMMENTS
                  ------    --------  ----  ---   ----   -------------------
                  001200    TABSPACE  0001  $ID   PGM

 PLAN:    DB2PLN
                  ------  -   --------   ---------------   -----------------
                  SEQ     I   PACKAGE    COLLECTION        COMMENTS
                  ------  -   --------   ---------------   -----------------
                  001300  I   DB2PKG                       IDEAL package
 IDEAL Application Plan/Pkg DB2PLN       March 10, 2010     15:27:49
PLAN:     DB2PLN ________
                  ------     --------       -------------------------------
                  SEQ        DBRM                          COMMENTS
                  ------     --------       -------------------------------
                     NO DBRM LINES FOR THIS PLAN

IDEAL Application Plan/Pkg DB2PLN       March 10, 2010     15:27:49
                         Application Plan/Pkg Impact Report
SQL MOD   PROGRAM   SYS  VER   TYPE  LANG    SQL STMTS  STATUS  ERROR IND
--------  --------  ---  ----  ----  ------  ---------  ------  -----------
DB2PLN0   TABSPACE  $ID  001   PROG  IDEAL  0003
Impact Report Abbreviations:
TYPE:   APPL   - Application (program with all its subprograms)
        PROG   - Program without its subprograms
        SPGM   - Subprogram belonging to an above application
Application Plan/Pkg Statistics:
Maximum number of SQL statements in effect: 0200
01     SQL modules are included in this Plan/Pkg
001    Programs are included in this Plan/Pkg
000003 SQL statements are in this Plan/Pkg
IDEAL Application Plan/Pkg DB2PLN       March 10, 2010     15:27:49
                      Authorization ID Assignment Report
       No Authorization ID Assignments are in effect for this Plan/Pkg
```

The list of programs is shown by Static I/O module name and includes the following:

- Program name
- System ID
- Program version
- Type (application, program, or subprogram)
- Program language
- Number of SQL statements in the program
- Resource status (the first three are warnings; the rest are fatal errors that terminate the process):
  - **DUPLIC**-Program already included in the plan
  - **NOSQL**-Program does not contain any SQL
  - **EXCLIM**-Number of SQL statements exceeds the maximum
  - **NOTFND**-Program not found (for example, not compiled or not defined to the dictionary)
  - **NOTAVL**-Program not available (busy)
  - **DIFVER**-Program already included with a different version
  - **HIST**-Program in history status
  - **INCREL**-Program compiled with an incompatible (higher) CA Ideal release

The second section of the report shows the generated BIND command text. It displays all DB2 BIND options selected. The MEMBER list includes the system DBRMs that CA Ideal requires, the DBRMs the plan generation generates, and non-Ideal DBRMs specified in the plan definition. The PKLIST includes all the packages specified in the plan definition.

```
IDEAL Application Plan/Pkg DB2PLN        March 10, 2010     15:27:49
                     Application Plan/Pkg BIND command
DSN SYS (D310)
BIND PLAN (DB2PLN) OWNER (WESJO01) MEMBER (DB2PLN0) LIBRARY ('IDLD.DB23-
10.DBRMLIB','IDLD.V2R1.DBRM') PKLIST (JWCOLL.DB2PKG@). AC-
TION (REPLACE) RETAIN VALIDATE (RUN) ISOLATION (RR) FLAG (I) ACQUIRE (USE)
RELEASE (COMMIT) EXPLAIN (NO) CURRENTDATA (YES)

Application Plan Summary Messages:
001201 1-IDADPLAG02I - Plan or Package DB2PLN existed before and has been replaced
001201 1-IDADPLAG05I - Generation successfully executed

End of Plan/Package Generation Report_
```

Note: The preceding summary message shows that CA Ideal's generation process is complete. It does not show the status of the DB2 BIND.

Defining Packages

A package is an SQL object that contains a set of SQL statements that were bound and are available for processing. Before an application statically run as a Package, that Package must be bound into a Plan. To define a package for a CA Ideal program or application, you must first create the package definition and then generate the package. This section explains how to use CA Ideal fill-ins to create the package definition. For information about generating the package, see the Generating Packages topic in this chapter.

Components of a Package Definition

Each package definition includes the following:

- A package identification fill-in that creates the entry in the dictionary for the package and provides the dictionary with identification information
- A package parameter fill-in that specifies DB2 BIND command options
- A package resource fill-in that specifies the CA Ideal programs included in the package

You cannot include non-Ideal DBRMs in CA Ideal packages. Therefore, there is no DBRM fill-in for packages.

The CREATE PACKAGE command creates package definitions. The EDIT PACKAGE command can update them. You can also use the commands DISPLAY, PRINT, DUPLICATE, and DELETE with package definitions. For information about the complete syntax, see the *Command Reference Guide.*

The functions CA Ideal provides to define and maintain package definitions are presented in the Plan Maintenance menu. To access this menu, select option 5 on the Main Menu or enter the PLAN command.

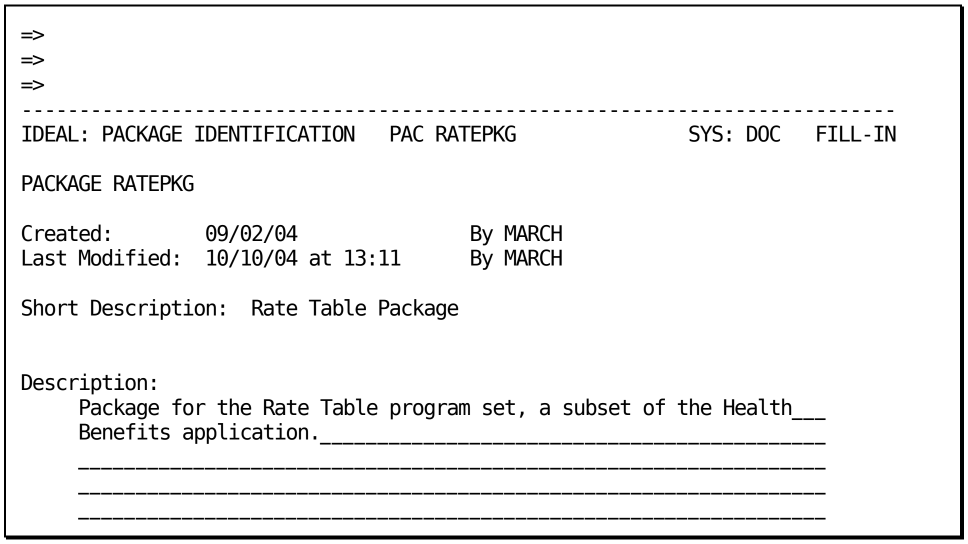
Creating a Package Definition

To create a new package definition, issue the CREATE PACKAGE command and complete the package identification fill-in. An entry for the package definition is added to the dictionary as soon as the package is named, either in the command or in the fill-in. The editors are the same as are used for Plans, but will perform their validation based on editing a Package.

Note that the name used for a Package must be the same as is used for the Module associated with the program. This common identity is used to load the correct I/O module when the application is run.

Identification Fill-in

The package identification fill-in enters descriptive information about the package when the package definition is created or when it is modified.



```
=>
=>
=>
----------------------------------------------------------------------------
IDEAL: PACKAGE IDENTIFICATION   PAC RATEPKG              SYS: DOC   FILL-IN

PACKAGE RATEPKG

Created:        09/02/04              By MARCH
Last Modified:  10/10/04 at 13:11     By MARCH

Short Description:  Rate Table Package


Description:
     Package for the Rate Table program set, a subset of the Health___
     Benefits application.____________________________________________
     _________________________________________________________________
     _________________________________________________________________
     _________________________________________________________________
```

The following list provides an explanation of the fields on the preceding screen:

PACKAGE name

Specifies the one- to seven-character name assigned to the package definition.

Created ... By

Specifies the initial creation date of the package definition and the user ID of the creator. This is blank when Identification fill-in is first displayed during package creation. The system supplies the information. It displays if you return to this fill-in. The user cannot modify it.

Last Modified ... at ... By

Specifies the date, time, and user ID of the last edit access. This is blank when the Identification fill-in is first displayed during package creation. The system supplies this information. The user cannot modify it.

Short Description

Displays an area where the user can supply a description of the package definition.

Limit: 24 characters

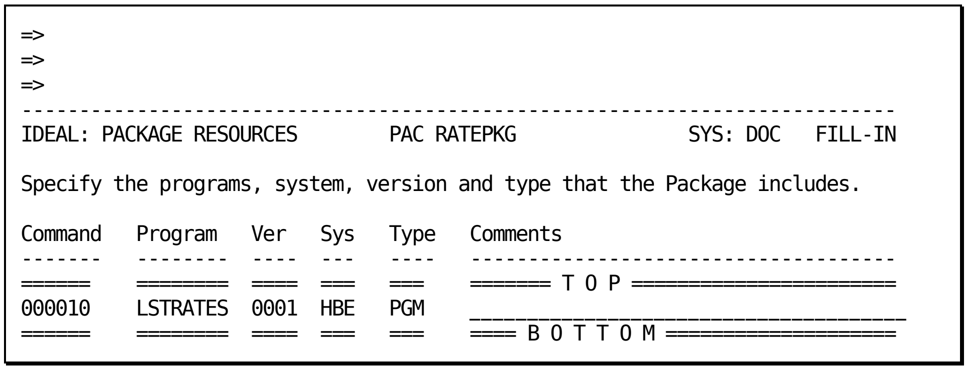
Description

Displays an area where the user can supply a longer description of the package definition.

Limit: 5 lines

Package Resources Fill-in

To specify the CA Ideal program that is included in the package definition, issue the RESOURCE command or press the PF5/17 key. This displays the package resources fill-in.



```
=>
=>
=>
----------------------------------------------------------------------------
IDEAL: PACKAGE RESOURCES        PAC RATEPKG              SYS: DOC   FILL-IN

Specify the programs, system, version and type that the Package includes.

Command   Program   Ver   Sys   Type   Comments
-------   --------  ----  ---   ----   -------------------------------------
======    ========  ====  ===   ===    ======= T O P ======================
000010    LSTRATES  0001  HBE   PGM    ______________________________________
======    ========  ====  ===   ===    ==== B O T T O M ===================
```

The following list provides an explanation of the fields on the preceding screen:

Program

Specifies the name of a CA Ideal program. (You cannot specify non-Ideal programs for a CA Ideal package.)

Ver (Optional)

The one- to three-digit version number or PROD. If you specify PROD, a version number is not substituted.

Default: 1

Sys (Optional)

The three-character CA Ideal system ID. Defaults to the current system.

Type

Not applicable to packages.

Default: PGM

Comments

Information displayed with the package definition. It has no effect on the package generation or on the resulting package.

Package Parameters Fill-in

Issue the PARAMETERS command or press PF4/16 to display the package parameters fill-in. This fill-in lets you set DB2 BIND options for the package and specify whether programs in this package can execute SQL in dynamic mode. The current settings display as one-character codes that you can change by overtyping.

```
=>
=>
=>
----------------------------------------------------------------------------
IDEAL: PACKAGE PARAMETERS      PAC RATEPKG              SYS: DOC   FILL-IN

Bind Options:
Collection id: ___________________________
Name:  RATEPKG        Location:  ________
Owner: ________       Qualifier: ________

Action:    R (R=Replace, A=Add)
Retain:    Y (Y=Yes, N=No)
Validate:  R (R=Run, B=Bind)
Isolation: R (R=Repeatable Read, C=Cursor Stability)
Flag:      I (I=InformationalWarningErrorCompletion,
              W=WarningErrorCompletion, E=ErrorCompletion,
              C=Completion only)
Acquire:   U (U=Use, A=Allocate)
Release:   C (C=Commit, D=Deallocate)
Explain:   N (Y=Yes, N=No)

Prepare:   _ (N=NODefer, D=Defer)
Currentdata: _ (Y=Yes, N=No)
SQLError:    _ (N=NoPackage or C=Continue)
Enable/Disable:
________________________________________________________________________________
```

The following list provides an explanation of the fields on the preceding screen:

Collection ID

Identifies the collection where the package belongs. This name is used as the high-order qualifier of the package name.

Name

Specifies the name of the package supplied from the Identification fill-in.

Location

Specifies the location of the DBMS where the package is bound.

Owner

Specifies the authorization ID of the package owner.

Qualifier

Specifies the authorization ID that qualifies any unqualified SQL objects accessed in this package. An authorization ID specified in an ASSIGN AUTH command can override this value at runtime.

Action

Specifies whether this application package is new or replaces an old package.

- **R**-Replace old package
- **A**-Add new package

Retain

Indicates whether those users with authority to bind or execute the existing package keep that authority over the replaced package.

- **Y**-Retain authority
- **N**-Do not retain authority

Validate

Indicates when full validity checking is performed.

- **R**-At runtime
- **B**-At bind time

Isolation

Specifies the isolation of this application from others.

- **R**-Repeatable read
- **C**-Cursor stability

Flag

Indicates the level of messages to show.

- **I**- Informational, warning, error, and completion
- **W**- Warning, error, and completion
- **E**-Error and completion
- **C**- Completion only

Acquire

Indicates when you want the system to acquire the resources your program uses.

- **U**-Open table spaces and acquire locks when your application first accesses them
- **A**-Acquire them when the application package is allocated

Release

Indicates when you want the system to release the resources your program used.

- **C**-Release resources at each commit point
- **D**-Release them when the application terminates

Explain

Indicates whether to include the explain option.

- **Y**-Include the explain option
- **N**-Do not include the explain option

Currentdata

Specifies the data currency required for ambiguous cursors opened at remote locations. Data is considered current if the data in your host structure is identical to the data in the base table.

- **Y**-Data currency is required for ambiguous cursors
- **N**-Data currency is not required for ambiguous cursors

SQLError

Indicates whether you want to continue the package bind if SQL errors are detected during the package generation.

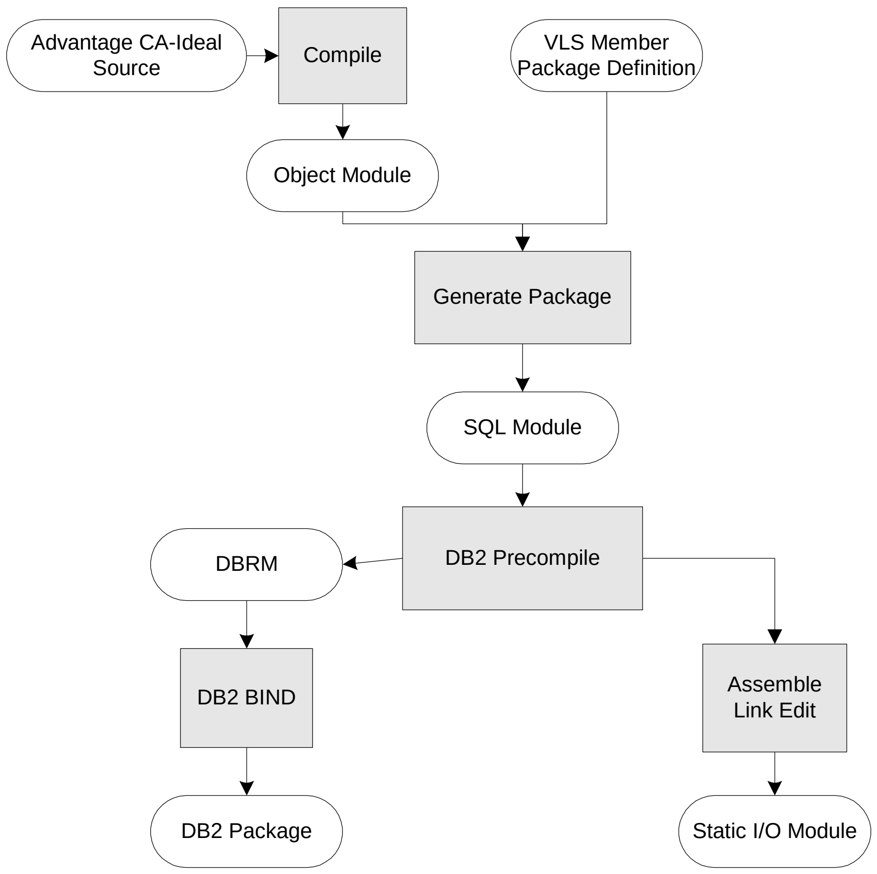
- **N**-Do not bind the package if SQL errors are detected
- **C**-Continue with the bind regardless of any SQL errors detected

Enable/Disable

These keywords are mutually exclusive. ENABLE lists the system connection types that are enabled to use the package. DISABLE lists the connection types that are disabled for the package. You can use one to three lines to enter the value for this parameter.

Generating Packages

The package definition generates the package and prepares applications for inclusion in a plan. When the plan is generated, the application is ready for execution in static mode. If you recompile a program and want it to run in static mode, you must regenerate the package, but you do not need to regenerate the plan. The package preparation process is illustrated in the following diagram:



Note: When an application with static SQL is executed, the application or a program in the application runs in dynamic mode if dynamic SQL was allowed on the plan parameters fill-in. One of the following is true:

- The package was not generated or the CA Ideal static I/O module cannot be loaded.
- A program in the application was recompiled without regenerating the package, resulting in a mismatch between the program and the I/O module.
- In either case, the affected program only runs in dynamic mode while the rest of the application runs in static mode.

If you regenerate a package while CICS is active, use the REFRESH PACKAGE command to make a new copy of the CA Ideal static I/O module available to CICS.

You can transport DB2 applications and package definitions from the development site to the production site using the CA Ideal transport utilities (see the *Working in the Environment Guide*).

Issue the GENERATE PACKAGE command in batch, as shown in the following sample jobstream. The JCL needed for package generation is in the procedure IDPLAN, which you can use for both plan and package generation.

```
//name EXEC IDPLAN
//SYSIN DD *
PERSON ...
.
.
.
GENERATE PACKAGE AP3
OFF
/*
//
```

For more information about syntax of these commands, see the *Command Reference Guide*.

The IDPLAN procedure performs the following steps:

1. Generation of CA Ideal static I/O module
2. DB2 precompile
3. Assembly
4. Link-edit
5. DB2 BIND

Note: If you specify COPY in the package parameter fill-in, the static I/O module for the original package is relinked with the name of the new package. Because the original static I/O module was already precompiled and assembled, steps 1 through 3 are not performed before the link edit, but the BIND is still performed as the last step.

Package Generation JCL

The JCL that the GENERATE PACKAGE command requires is included in the IDPLAN procedure. You can modify it for your site. The statements are grouped by the process they perform (precompiler, assembler, link-edit, terminal monitor). Statements you might need to modify are:

- **SQLMOD** -Data set where source for the static I/O module is placed.
- **IDUDBRM**-Specifies the data set to contain the DBRM that the precompiler produces.
- **SYSLMOD**-Load library containing the CA Ideal static I/O module.

CA Ideal Static I/O Modules

A static I/O module is created as part of the CA Ideal GENERATE PACKAGE procedure. CA Ideal calls this static I/O module in an assembler subprogram during static execution of SQL. This subprogram, which contains the SQL statements from the programs specified in the package resource definition, is passed to the DB2 precompiler.

The GENERATE PACKAGE command creates only one SQL module. The name of the static I/O module is the one- to seven-character package name, with an eighth character of @.

Note: The load library that the GENERATE PACKAGE procedure uses as SYSLMOD must be available to any CICS partition, DFHRPL, or batch STEPLIB, where the applications associated with the package are expected to execute in static mode.

CA Ideal Plan Tables

The GENERATE PACKAGE command also updates three DB2 plan tables. The rows in these tables record all programs participating in the package and any changes by the ASSIGN AUTHID command. You can query these tables using SQL (in the same way that you query other tables) to answer questions such as, “Which package does program x participate in?”

"SYSADR Table Declarations for DB2" appendix illustrates the layout of these tables, named SYSADR.APTAB, SYSADR.APRES, and SYSADR.APAUT, respectively.

Impact Report

The report produced by the generation process has three parts:

- The components of the package definition and the programs involved in the package.
- The generated BIND command text, including the DBRM.
- Error and completion messages.

You can produce this report without actually doing the BIND or updating the plan tables by using the VERIFY option on the PRINT PACKAGE command. (For the complete syntax, see the *Command Reference Guide*.)

The first section of the report shows the package information entered in the package definition. It is identical to the information produced by a PRINT PACKAGE command.

```
IDEAL APPLICATION PLAN/PKG REPORT
          RELEASE:  11         DATE: March 9, 2006      TIME: 17:11:00
                        GENERATE APPLICATION PLAN/PKG DB2PKG
 IDEAL Application Plan/Pkg DB2PKG        March 9, 2006      17:11:00
          DB2PKG ________                  VERSION:  __   STATUS: PROD
                   PKG  DB2PKG
                   Created:        09/29/04               By $ID
                   Last Modified:  11/01/04 at 09:29      By $ID
                   Short Description:  test
                   Description:
                        ________________________________________________________
                        ________________________________________________________
 IDEAL Application Plan/Pkg DB2PKG        March 9, 2006      17:11:00
          DB2PKG ________                  VERSION:  __   STATUS: PROD
              Collection Id: COLLID
              Bind Options:
              Owner: TESTID        Qualifier:               Location: _____
                   Action:    R (R=Replace, A=Add)  Replver: ______________
                   Retain:    Y (Y=Yes, N=No)
                   Validate:  R (R=Run, B=Bind)
                   Isolation: C (R=Repeatable Read, C=Cursor Stability)
                   Flag:      I (I=InformationalWarningErrorCompletion,
                                 W=WarningErrorCompletion, E=ErrorCompletion
                                 C=Completion only)
                   Acquire    U (U=Use, A=Allocate)
                   Release    C (C=Commit, D=Deallocate)
                   Explain    N (Y=Yes, N=No)
                   Prepare    N (N=Nodefer, D=Defer)
                   Currentdata:  Y (Y=Yes, N=No)
                   SQLerror:     N (N=NoPackage, C=Continue)
                   Enable/Disable:
                   DISABLE (IMS,BATCH,DLIBATCH)
 IDEAL Application Plan/Pkg DB2PKG        March 9, 2006      17:11:00
          DB2PKG ________                  VERSION:  __   STATUS: PROD
                  ------    --------  ----  ---   ----   -------------------
                  SEQ       PROGRAM   VER   SYS   TYPE   COMMENTS
                  ------    --------  ----  ---   ----   -------------------
                  001200    TABSPACE  0001  $ID   PGM
DEAL Application Plan/Pkg DB2PKG        March 9, 2006      17:11:00
                         Application Plan/Pkg Impact Report

SQL MOD   PROGRAM   SYS  VER   TYPE  LANG    SQL STMTS  STATUS  ERROR IND
--------  --------  ---  ----  ----  ------  ---------  ------  -----------
Impact Report Abbreviations:
TYPE:   APPL   - Application (program with all its subprograms)
        PROG   - Program without its subprograms
        SPGM   - Subprogram belonging to an above application
Application Plan/Pkg Statistics:
01     SQL modules are included in this Plan/Pkg
001    Programs are included in this Plan/Pkg
000003 SQL statements are in this Plan/Pkg
IDEAL Application Plan/Pkg DB2PKG        March 9, 2006      17:11:00
                      Authorization ID Assignment Report
       No Authorization ID Assignments are in effect for this Plan/Pkg
```

The list of programs is shown under the static I/O module name and includes:

- Program name
- System ID
- Program version
- Type (application, program, or subprogram)
- Program language
- Number of SQL statements in the program
- Resource status (the first three are warnings; the rest are fatal errors that terminate the process):
  - **DUPLIC**-Program already included in the package
  - **NOSQL**-Program does not contain any SQL
  - **EXCLIM**-Number of SQL statements exceeds the maximum
  - **NOTFND**-Program not found (for example, not compiled or not defined to the dictionary)
  - **NOTAVL**-Program not available (busy)
  - **DIFVER**-Program already included with a different version
  - **HIST**-Program in history status
  - **INCREL**-Program compiled with an incompatible (higher) CA Ideal release

The second section of the report shows the generated BIND command text. It displays all DB2 BIND options selected. The MEMBER list includes the DBRM generated by package generation.

```
IDEAL Application Plan/Pkg DB2PKG        March 9, 2006      17:11:00
                     Application Plan/Pkg BIND command
DSN SYS (DEVL)
BIND PACKAGE (COLLID) OWNER (TESTID) MEMBER (DB2PKG@) LIBRARY ('IDLD.D-
B2.DBRMLIB') ACTION (REPLACE) VALIDATE (RUN) ISOLATION (CS) FLAG (I)-
RELEASE (COMMIT) EXPLAIN (NO) CURRENTDATA (YES) DISABLE (IMS,BATCH,DLI-
BATCH)


End of Application Plan/Pkg Generation Report
```

Note: The summary message shows that CA Ideal's package generation process is complete. It does not show the status of the DB2 BIND.

Connecting Plans to Applications

This section describes how to specify the application plans to use when executing SQL in CA Ideal applications, and how to make plans known to the CICS attach facility or to the IBM Call Attach Facility in batch.

Dynamic Mode

To run in dynamic mode only, your application can run under the default transaction-ID SCFD. As part of the CA Ideal installation, CA Ideal provides a CICS RCT table entry for this transaction-ID with its associated plan name. CA Ideal's installed default plan name is IDP140DV.

The SCFD entry, or a copy with a different name, is needed to run in dynamic mode. CA Ideal uses it to support the development environment.

Static Mode

To run a CA Ideal application in static mode, you can select the application plans in the following ways:

- Under CICS, run the application under a CICS transaction-ID associated with the plan. This means that the plan must include all SQL to run under that transaction. To change plans, you must change transactions.
- Or, run under any transaction-ID associated with the RCT exit program that CA Ideal provides, and set the initial plan in CA Ideal. You can set the plan for an entire run or change plans between or in transactions.

  In batch, set the plan in CA Ideal. You can set the plan for an entire run or change plans during the run.

Associating Plans Directly with Transaction-IDs in CICS

You can associate a plan name directly with one or more transaction-IDs in the RCT, using the DB2TRAN and DB2ENTRY resources. To run using that plan, you must run under a transaction-ID associated with the plan. To select the plan in CA Ideal, you must assign a CA Ideal account-ID that is the same as the CICS transaction-ID and change transactions. It is also necessary to inform CA Ideal which plan to use through the SET RUN PLAN command.

You can assign the CA Ideal account-ID in any of the following ways:

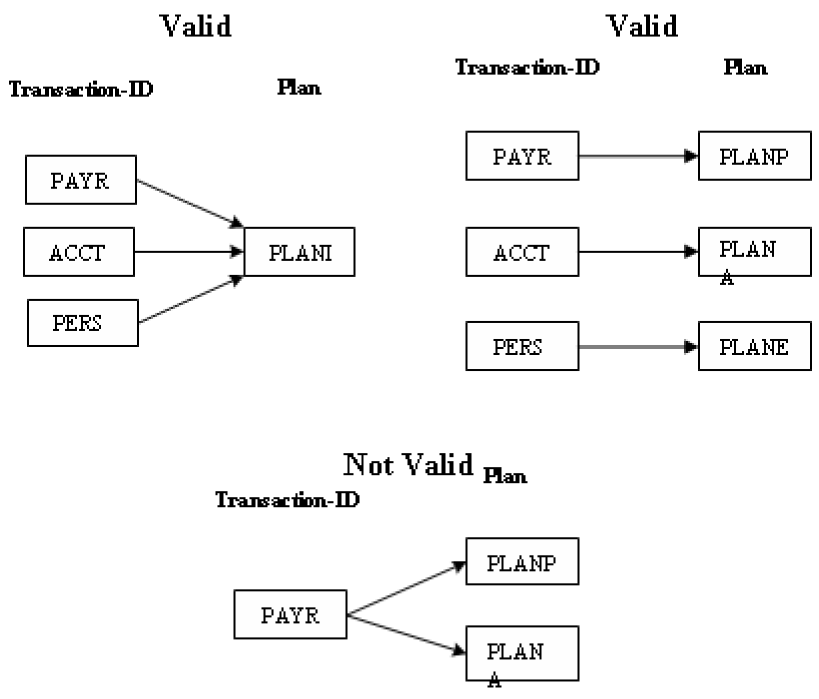
- At the terminal by entering the SET ENVIRONMENT ACCOUNT-ID command before running the application. Press Enter, to start a new transaction before the change takes place.
- In PDL using the SET $ACCOUNT-ID statement followed by a TRANSMIT statement. The change in transaction-ID takes place after the TRANSMIT starts a new transaction. You cannot execute any static SQL using the plan before these statements are executed.
- Using transparent signon to enter CA Ideal with a transaction-ID you defined in the CICS PCT and a signon member that contains a SET ENVIRONMENT ACCOUNT-ID command to specify the transaction-ID you run under. The signon member might contain:

  ```
  SET ENV ACCOUNT-ID PAY
  SET RUN PLAN PAYPLAN
  SET RUN SQL STATIC
  RUN COM-DUE
  ```

In this example, you must define transaction-ID PAY to CICS, using the plan name PAYPLAN. Changing to transaction-ID PAY does not take place until a TRANSMIT is executed in the COM-DUE program. If any SQL statements in this program are to be static and they are executed before the first transmit, they must be part of a plan associated with the previous transaction-ID.

For example, if you sign on under transaction-ID EMP using the sample member, the SQL statements in program COM-DUE executed before the TRANSMIT must be part of the plan for EMP. Thus, both transaction-IDs, EMP and PAY, must be defined as using the same plan.

An entry can associate multiple transaction-IDs with the one plan or you can have multiple entries each specifying a transaction-ID with its own plan. But you cannot associate the same transaction-ID to different plans.



For details on RDO for DB2 requirements, see the appropriate DB2 administration documentation.

Specifying Plans Independently of Transaction-IDs in CICS

CA Ideal gives you the option to switch plans in a CICS transaction. You must specify the exit that CA Ideal supplies in the entry for the transaction.

You then have several ways to select plans in CA Ideal. They are described next in reverse order of precedence; that is, the plan specified by each method overrides the plans specified by the methods described before it.

Specifying Plans as Session Options

The following CA Ideal command specifies a plan name for a session or run:

```
SET RUN PLAN plan-name
```

This name is used unless it is superseded by a new plan name specified in a CA Ideal application or a CA Ideal plan name exit program (see the Specifying Plans with a CA Ideal Plan Name Exit topic). You can include the command in your jobstream or signon member or issue it at the terminal before running an application. The following command resumes using the installed default, IDP140DV:

```
SET RUN PLAN DEFAULT
```

Specifying Plans in an Application

In a CA Ideal application, you can set a new plan name in place of the default plan name. The PDL function $PLAN sets a plan name or returns the plan name most recently set in the application.

If a plan name was not set in the application, the function returns the name set for the session. If a name was not set for the session, the function returns IDP140DV. It does not return a value set in a plan name exit (described next).

You can use $PLAN with a SET statement to set a new CA Ideal plan name for the application, for example:

```
SET $PLAN = 'INVPLAN'
```

Note: You can execute the SET $PLAN statement only before the first SQL statement in a logical unit of work. That is, executing SET $PLAN at any point except before the first SQL statement at the beginning of a CICS transaction or following a database Commit causes a runtime error.

The first SQL statement can be embedded SQL, SQL generated by a FOR construct for a DB2 dataview, or SQL in a non-Ideal subprogram. A Commit can be a CHECKPOINT, a BACKOUT statement, or an SQL COMMIT or ROLLBACK statement.

The new plan takes effect when the exit is called at the next SQL statement. For more information about $PLAN, see the *Programming Reference Guide*.

The following procedure saves the plan name that was selected in a previous procedure in a Working Data field SAV-PLAN. It sets GETPLAN as the plan name to use with the FOR construct that follows it, commits its database modifications, and resets the plan name before returning.

```
<<GET>> PROCEDURE
             SET SAV-PLAN = $PLAN
             SET $PLAN = GETPLAN
             FOR EACH DB2-DVW
                ...
             ENDFOR
             CHECKPOINT
             SET $PLAN = SAV-PLAN
ENDPROC
```

Specifying Plans with a CA Ideal Plan Name Exit

In addition to the exit required for plan switching, CA Ideal lets you specify a plan name exit program. CA Ideal passes parameters to the plan name exit, including the current installed default plan name, the session default, or the name set in the application. The exit can modify it, pass it on unchanged, or ignore it and select a new plan name.

The use of a plan name exit is optional. To use one, enter the following command in your signon member, jobstream, or at the terminal:

```
SET [SITE] ENVIRONMENT DB2PLAN-EXIT exit-name
```

You can also use the command to disable the exit:

```
SET [SITE] ENVIRONMENT DB2PLAN-EXIT NONE
```

If the plan name exit is enabled, CA Ideal calls it before the first SQL statement in a logical unit of work. That is, it is called before the first SQL statement at the beginning of each CICS transaction and before the first SQL statement following each database commit.

Note: The first SQL statement can be embedded SQL or SQL generated by a FOR construct for a DB2 dataview or in a non-Ideal subprogram. A Commit can be a CHECKPOINT or BACKOUT statement, or an SQL COMMIT or ROLLBACK statement.

Specifying @IADRCTX as the PLANEXIT

CA Ideal delivers an exit program, @IADRCTX. To allow the application to change plans in a transaction, you must specify this name in the DB2ENTRY for the transaction.

```
DB2ENTRY (name) ... PLANEXITNAME(@IADRCTX) ...
```

These entries are DB2 requirements described in the appropriate database administration documentation.

DB2ENTRY PLANEXIT Versus Plan Name Exit

In summary, two types of exits are involved in plan switching in CA Ideal.

CA Ideal supplies the Planexit named @IADRCTX for use in the CICS environment. You cannot modify it. The CICS attach facility calls it when the first SQL statement is executed in a logical unit of work. The DB2ENTRY for this transaction-ID specifies PLNEXITNAME=@IADRCTX.

The plan name exit is available under CICS. The user supplies it as needed. (You can use the sample exits described in previous pages.) CA Ideal calls it at the point when CA Ideal is about to call the CICS attach facility with the first SQL statement in a logical unit of work. This exit can also modify the plan name to use. Under CICS, CA Ideal stores its plan name in a CICS temporary storage record for later access by the exit. (The exit in turn provides the plan name to the DB2 attach facility.)

When you use a plan name exit to modify the plan name or select a new plan name, the new value is not available to the $PLAN function. This means that the plan DB2 uses might not be the same as the plan the $PLAN function returned. The plan that DB2 actually uses can display in an Error Procedure as the value of the function $ERROR-DB2-PLAN.

If an application calls a non-Ideal subprogram that executes SQL statements and if that SQL might be the first in a logical unit of work, then you must specify Y (yes) for the Update DB2? field on the non-Ideal program IDE panel, even if the SQL in the subprogram does not actually update DB2.

If an application calls a non-Ideal subprogram that is identified as containing SQL, CA Ideal assumes that the SQL is executed. Therefore, if the call is detected in a new logical unit of work before the first SQL statement in CA Ideal, it is assumed to be the first SQL request in the new logical unit of work. The CA Ideal plan name exit, if enabled, is invoked before calling the non-Ideal subprogram. The non-Ideal subprogram is identified in the plan name exit's parameter list as PLAN-SUB-PROGRAM.

Example

Consider the following scenarios to understand better.

You can use the SET $PLAN PDL statement to optionally specify a logical identifier for the next plan to use. This logical identifier is not necessarily the actual plan name. This level of indirection lets you avoid hard coding actual plan names in your programs (which require compilation for any change to plan names). This is useful, for example, if the plan name at a satellite location is slightly different from the plan name used at the development site. Since the program specifies a logical identifier, you can choose the actual plan name outside the PDL code based on site considerations. But the CA Ideal plan name exit can base the choice of actual plan name on any available criteria. Use of SET $PLAN is entirely optional.

1. At customer site ABC, program names are always six characters long. Each program has its own plan. In the Development environment, plan names are formed by prefixing a D before the program name and in production, by prefixing a P. In this example, the SET $PLAN statement is not used. The CA Ideal plan name exit program simply checks for the current program name and generates the appropriate plan name.
2. At customer site XYZ, plan names are generated by a more complex algorithm. The program can generate a character string at the start of any logical unit of work. The letter X is prefixed to form the actual plan name. In this case, the SET $PLAN statement specifies the generated string, for example, SET $PLAN = 'PLAN03'. The plan name exit then prefixes an X, resulting in an actual plan name of XPLAN03.
3. At customer site PQR, plan names are actually hard coded in the program. In this case, the program specifies SET $PLAN = 'xxxx', and there is no CA Ideal plan name exit. The string specified as xxxx becomes the actual plan name.

Specifying Plans in Batch

Any of the three methods for selecting plans (see the descriptions in Specifying Plans Independently of Transaction-IDs in this chapter) are available in batch without using an RCT exit. Plans can be selected as follows:

- As a session option.
- Specified in an application.
- In a CA Ideal plan name exit. In batch, CA Ideal passes the plan name to the IBM Call Attach Facility.

Plan Name Exit

The following procedure describes how to establish a plan name exit program.

1. Compile and link-edit the plan name exit program into your load library as a load module

   See the specifications that follow.

2. (CICS Only) Add a PPT entry for the program and, if necessary, restart CICS

   Make sure that the language parameter for the program is correct. If this module is heavily used, it may be useful to make it resident.

3. Specify the SET [SITE] ENVIRONMENT DB2PLAN-EXIT command to establish the name of the plan name exit program

   Since the CICS and non-CICS exit programs are specific to their respective environments, take care when specifying a site option. You can link-edit both versions as modules with the same name in separate libraries provided that CICS and non-CICS JCL each use the appropriate library.

   CA Ideal passes to the plan name exit the following information.

Parameters Common to Other CA Ideal Exits

```
01 ID-PARM-1.
               05  ID-EXIT-TYPE                     PIC X.
              05  ID-SYNC                          PIC X(03).
               05  ID-RELEASE-LEVEL                 PIC X(04).
               05  ID-USER-SHORT-ID                 PIC X(03).
               05  ID-USER-NAME                     PIC X(32).
               05  ID-TERMINAL-ID                   PIC X(04).
               05  ID-TRANSACTION-ID                PIC X(04).
               05  ID-TP-MONITOR-CODE               PIC X(01).
               05  ID-OPERATING-SYSTEM-CODE         PIC X(01).
               05  ID-NETWORK-ID                    PIC X(8).
```

Plan-Specific Parameters

```
01 ID-PARM-2.
              05  FILLER                          PIC X(08)
               05  PLA-RUN-SYSTEM                   PIC X(03).
               05  PLA-RUN-PROGRAM                  PIC X(08).
               05  PLA-SUB-SYSTEM                   PIC X(03).
               05  PLA-SUB-PROGRAM                  PIC X(08).
               05  PLA-SUB-TYPE                     PIC X(01).
               05  PLA-SET-PLAN                     PIC X(08).
               05  PLA-DB2-PLAN-NAME                PIC X(08).
               05  FILLER                           PIC X(16)
```

The sample plan name exit programs contain 88-level names with appropriate values for fields containing CA Ideal's internal codes.

ID-EXIT-TYPE

Type of exit invoked for this command. A 3 indicates that a plan exit is invoked.

ID-SYNC

Reserved.

ID-RELEASE-LEVEL

CA Ideal Release level (for example 1400 for Version 14.0).

ID-USER-SHORT-ID

One- to three-character CA Ideal user short ID defined for the user who is executing the command.

ID-USER-NAME

Name of the CA Ideal user who is executing the command.

ID-TERMINAL-ID

- **In CICS**-CICS ID of the terminal from which the command is executed.
- **In batch**-Hex zeros.

ID-TRANSACTION-ID

- **In CICS**-CICS signon transaction-ID.
- **In batch**-IDEA.

ID-TP-MONITOR-CODE

- **C**-Represents CICS.
- **Y**-Represents batch.

ID-OPERATING-SYSTEM-CODE

- **O**-Represents z/OS.

ID-NETWORK-ID

In CICS- VTAM LU name. If the terminal is VTAM, the system ID and terminal ID of the Terminal Owning Region (TOR). If the terminal is MRO but not VTAM, low values under any other circumstances.

PLA-RUN-SYSTEM (Input)

System of the main program (named in the RUN command).

PLA-RUN-PROGRAM (Input)

Name of the main program (named in the RUN command).

PLA-SUB-SYSTEM (Input)

System of the current subprogram.

PLA-SUB-PROGRAM (Input)

Name of the current subprogram.

PLA-SUB-TYPE (Input)

An indicator of the subprogram type:

- **I (CA Ideal)**-SQL to execute is in a CA Ideal module.
- **N (Non-Ideal)**-CA Ideal is about to call the program PLA-SUB-PROGRAM. (See the Note at end of this section.)

PLA-SET-PLAN (Input)

Value of the most recent SET $PLAN statement. It is set initially to the value of the SET RUN PLAN command or to the default plan name established at installation. (This is the plan that supports CA Ideal's development environment.)

PLA-DB2-PLAN-NAME (Update)

Value returned to CA Ideal. CA Ideal passes this value to DB2 as the actual plan name. If SPACES or LOW-VALUES is returned to CA Ideal, the installation default plan is selected. The exit is free to return any value in the PLA-DB2-PLAN-NAME field. However, invalid plan names can result in runtime errors.

Note: Non-ideal subprograms can use a plan name with up to eight characters. CA Ideal application plans are a maximum of seven characters long.

Requiring the Use of Static SQL

You can override a specification of dynamic SQL in the plan parameters fill-in, and require that all programs run in static mode, by using the SET RUN SQL STATIC command.

Chapter 9: Establishing Signon Processing

This chapter includes the following topics:

- Defining signon requirements by establishing the objective of the environment
- Explanation of the types of signon transactions, including transaction selection criteria, definition in CA Ideal, and identification to CICS
- Setting up transactions (usually used in a production environment) that automatically run a CA Ideal application
- Establishing an optional signon exit
- How signon processing actually works, specifically with and without an external security package
- The CA IPC and CA Ideal options that control customization of signon processing
- Development and production considerations that can impact user definition and signon processing setup
- Step-by-step instructions for enabling an environment to interface with external security

CA Ideal uses the CA Standard Security Facility (CAISSF) to interface with external security products such as CA Top Secret, CA ACF2, RACF, and other SAF-compatible products. This lets you use any of these security products, in both online and batch environments, to interface TP monitor signon with CA Ideal signon.

Defining Signon Requirements

There are many different options available to provide you with the method of signon that fits your processing needs. First, it is important to define those needs.

CA Ideal Environment Functionality

First you must consider the functionality of the environment for which you are setting up signon procedures. Ask yourself the following questions:

- Is this a development environment, a QA environment, an end-user production environment?
- What is the nature of the users?
- Is there a need to identify each user uniquely in the CICS and CA Ideal environment as is usually desirable in a development environment?

- Is there a group of individuals that share information for read-only purposes in a single company or department?
- Do I want to be able to uniquely track each of those users as individuals or as a group?
- Is this a public access system where it does not matter who gets into a group of read-only applications?
- Does your signon needs depend on the nature of the environment and the applications that runs?

Following are some areas to consider. There might be other areas that you need to examine when devising signon procedures.

TP Monitor Signon

How do the users identify themselves to CICS? Is the TP monitor identity important to the processing that takes place? CICS Transaction Server requires a security package for signon.

Using a Security Package

You can signon to CICS using an external security package such as CA Top Secret, CA ACF2, or RACF. In each security package, you can customize how this process is carried out. For instance, you can require that each user actually signon to the security package or you can set up a default user for the environment or a particular transaction. You can also let a group of users share the same TP monitor definition.

Using the TP Monitor ID

Based on the information you gather regarding the nature of the region, you can decide whether you want individual users to have their own CICS signons, have a group of users share CICS signons, or whether you want to define a default signon for transactions, terminal, region, and so on.

You want to provide individual CICS accounts when individuality matters in the CICS environment:

- Transaction security based on user ID
- Chargeback based on CICS user ID

CA Ideal User Definition

The next step in the process is to determine the most efficient way to define your CA Ideal users. This step requires steps that are similar to the TP monitor signon.

When you define the user signon procedures, ask yourself the following:

- Do you care who the individual CA Ideal user is?
- Do you want to enable a group of users to share a single CICS user definition and also enable multiple CICS users to share?
- Do you want to enable a single CA Ideal user definition or for each of them to have their own definition, allowing individuals to set up a default user for a transaction or an environment that anyone can use?

For example, in a development environment, it is usually very important that each person be assigned his own TP monitor user definition and CA Ideal user definition. Because a programmer is updating entities in CA Ideal, you want to be able to identify the actual person who is signed on and what entities he last updated and compiled. Whereas, if a group definition or default user was set up, some of these items would no longer be distinguishable.

It is more likely that in a production environment, where it is not important to know the exact identity of a user who is signed on, that you share CICS and CA Ideal user definitions. An example might be a read-only application of non-confidential data. On the other hand, a production application that is allowing updating of more secured data is more likely to require that the exact identity of the user be known.

Region Considerations

When you make your region considerations, ask yourself the following:

- What are your security needs?
- Do you have several CA Ideal regions sharing a common security environment and want to allow or deny access to CA Ideal on an environmental basis?
- Do you want to secure particular transactions?
- Do you want to allow access only at certain times of the day, to certain terminals, and so on?

All of the following options are possible with or without an external security package (although the external security package makes some options simpler to implement). The most important thing is to define your needs.

1. Interface CA Ideal signon with signon to the TP monitor.

   Establish a link between a security package signon-id or TP monitor ID and the CA Ideal user definition.

2. Secure each CA Ideal environment on a user basis.

   Through an interface with an external security package, you can assign a name to each CA Ideal environment or a group of environments and grant access to the environment on an individual basis. Without a security package, you need to secure CA Ideal transactions.

3. Define group definitions.

   You can create a CA Ideal user definition that behaves like a profile and you can add aliases for each TP monitor user. With this method, any time a change occurs, you only need to modify one user definition.

4. Define default users.

   You can assign a CA Ideal user definition to use with a particular CA Ideal transaction or for the entire CA Ideal environment in the event that an undefined user tries to signon.

5. Limit users to a single session.

   You can allow or deny the TP monitor user access to multiple CA Ideal sessions under a single region through the external security package or in CA Ideal.

The remainder of this chapter contains information about the options and features available to allow you to fulfill your requirements.

Establishing CA Ideal Signon Transactions

CA Ideal recognizes a signon transaction by the transaction-ID it uses. There are three basic types of CA Ideal signon transactions. However, there can be any number of signon transactions. To establish a signon transaction, you must make an entry for that transaction in the CA IPC SCF Transaction Table (SC00TRAN). For information, see the *CA IPC Implementation Guide*. The transaction also needs to be defined to CICS. For more information about JCL, see the *CA IPC Implementation Guide*.

General Transaction Types

CA Ideal facilities allow for some flexibility in the procedure necessary for a user to sign on to CA Ideal. You can institute signon procedures to provide the following methods of signing on to CA Ideal.

Standard

The standard signon signs a user on to CA Ideal from CICS using the CA Ideal signon screen where the user provides the CA Ideal signon-ID and password and executes the user's member named SIGNON, if one exists. This method is most often used in a development environment.

"IDEA" is the sample standard signon transaction ID provided with the CA Ideal installation under CICS.

Express

An express signon transaction that automatically signs a user on to CA Ideal from CICS bypasses the CA Ideal signon-screen and executes the user's member named SIGNON, if one exists. This method is most often used in a development environment.

"IDLX" is the sample express signon transaction provided with the CA Ideal installation. Under CICS the user enters an express signon transaction ID instead of entering the IDEA transaction. This transaction ID bypasses the CA Ideal signon screen and executes the user's SIGNON member, if one exists.

Transparent

In a production environment, a transparent signon is one that signs a user on to CA Ideal automatically from CICS and invokes a CA Ideal application through the execution of a common member identified in the SC00TRAN. This setup establishes an environment where the end user can enter an application with no knowledge of CA Ideal.

Additionally, with this method, an end user can be transferred into a CA Ideal application from another CA Ideal application, from CICS, or from a program in CICS. For example, from a menu driven by a native application, the user can select an entry that automatically initiates CA Ideal and an application.

External security packages can provide additional functionality to the signon process; for example, limiting the transactions a user can access, what time of day the transactions can execute, and many others.

The CA Ideal express and transparent signon facilities also allow sites with special processing or security needs additional control during the signon process. For example, you could use the signon exit to override the signon member that gets executed based on a criterion such as the term ID. A signon exit program provides this control.

SCF Transaction Table (SC00TRAN)

For all online environments, all transaction IDs used at a site with SCF-based products must be defined in the SCF Transaction Table (member SC00TRAN in the IPC/IDEAL library). The entries for the standard signon transaction (IDEA) and one express signon transaction (IDLX) are provided in the installation SC00TRAN member. To add any additional signon transactions, either express or transparent, create the entries in the member USRTRANS. When this member is assembled and linked, these entries are added to the SC00TRAN load module. Since this table is searched top down, add new transactions in order of frequency used, following the IDEA and IDLX transactions.

The following sample SCF Transaction Table. It includes the entries required for each type of CA Ideal signon.

```
SC00TRAN SCTRANTB TYPE=INITIAL
     SCTRANTB TYPE=ENTRY,TRANID=IDEA,PROD=IDL,    (STANDARD)   X
         OPTIONS=(DD,PS),                                      X
         IDENT='IDEAL:',                                       X
         XFERCMD=IDEAL
     SCTRANTB TYPE=ENTRY,TRANID=IDLX,PROD=IDL,  (EXPRESS)      X
         OPTIONS=(DD,EX,PS),                                   X
         IDENT='IDEAL:',                                       X
         XFERCMD=IDEAL
     SCTRANTB TYPE=ENTRY,TRANID=QPAY,PROD=IDL,                 X
         OPTIONS=(DD,EX,PS),                                   X
         IDENT='QPAY:',                                        X
         DFLTUSR=PAY,                                          X
         TRNDATA='N$IDQSTARTUP',                               X
         XFERCMD=IDEAL
     SCTRANTB TYPE=ENTRY,TRANID=QUIK,PROD=IDL,                 X
         OPTIONS=(DD,EX,PS),                                   X
         IDENT='QUIK:',                                        X
         XFERCMD=IDEAL
     SCTRANTB TYPE=ENTRY,TRANID=DDOL,PROD=DDO,                 X
         OPTIONS=(DD),                                         X
         IDENT='DDOL:',                                        X
         XFERCMD=DDOL
     SCTRANTB TYPE=FINAL
     END
```

The table entry for a CA Ideal signon transaction contains the following parameters:

TRANID

Identifies the four-character transaction identifier. In CICS, the TRANSID parameter value is in the CICS PCT entry.

PROD

Identifies the current CA product. In this example, IDL refers to CA Ideal.

OPTIONS

Establishes the options currently available. These options include:

- **DD**-Initializes Datadictionary access.
- **EX**-Invokes express signon.
- **PS**-Enables the print subsystem.

IDENT

The 1- to 12-character product identification displayed in the upper left-hand corner of each SCF and PSS panel. This entry must be delimited by single quotes (').

DFLTUSR

Establishes a default user ID for this transaction and can be one of the following:

- **xxx**-Indicates that the authorizations specified in User Definition *xxx* is used for all users executing this signon transaction who are not otherwise defined in the dictionary. The value of *xxx* must be the three-character user ID of a valid CA Ideal user definition.

If a default user is also specified in the IDOPTS table, the transaction-based default user overrides the one specified in IDOPTS.

- **NONE**-Indicates that no default is used for this transaction. You can still specify a generic default in the IDOPTS table. It is used for this transaction if the operator ID, terminal ID, or security ID of the user executing this transaction does not match a valid CA Ideal user definition.

If you do not specify this parameter, no default user definition is used for this transaction unless a default is specified in the IDOPTS table.

TRNDATA='xyyyzzzzzzzz'-(For transparent signon only)

An optional 12-character field that passes data to the initialization routine. The data must be delimited by single quotes (') for CA Ideal and must contain the following values:

- **x**
  - **Y**-If the site-supplied signon exit is called.
  - **N**-If no signon exit is called.
- **yyy-**The CA Ideal user-ID of the user whose member is executed at signon.
- **zzzzzzzz-**The one- to eight-character name of the member to execute at signon. This member contains a RUN command to initiate the appropriate application.

Note: If no name is specified in the TRNDATA parameter of the SCF Transaction Table entry and the signon exit program does not specify a user name and member name, no SIGNON member is executed.

If the TRNDATA parameter is set to Y but you did not specify SET SITE ENVIRONMENT SIGNON-EXIT, neither the member specified for TRNDATA nor the user's signon member is executed. The CA Ideal Main Menu displays when the signon is complete.

XFERCMD=

The command that, when issued in another product session, results in transfer to this product. The value of this parameter must be IDEAL.

If the SCF Transaction Table is updated, you must reassemble and link it to create a new SC00TRAN load module. (For details, see the *CA IPC Installation Guide*.)

CICS PCT Definitions

For CICS, an entry must be present in the PCT for each signon transaction. The entries that establish a standard CA Ideal signon and an express signon are automatically provided during the installation of CA Ideal. Add one additional entry for each additional express signon transaction ID and for each transparent signon transaction ID.

The following entry is for the standard CA Ideal signon transaction.

```
DEFINE   TRANSACTION(IDEA) PROGRAM(SC00INIT)
        GROUP(IL11GRP) TWASIZE(64) PROFILE(IL11PRF) SPURGE(YES)
```

Except for the TRANSACTION value, the CICS PCT entry is the same for all CA Ideal signon transactions. The value specified for TRANSACTION must match the value for the TRANID in the SCF Transaction Table. The TWASIZE must be 64 for all SCF transactions.

To determine which PCT entries reference SC00INIT, SC00DISP, and SC00SAST, and to see what type of entry the transaction is, enter the command DISPLAY PCT.

The DISPLAY PCT in CA Ideal command displays the following information:

```
TRAN  INITIAL   TYPE OF     TWA    PRD
ID    PROGRAM   ENTRY       SIZE
CWBA  DFHWBA    WEB INTF.     0         Transaction not in SCWBTRAN
DDOL  SC00INIT               12    DDO
IDEA  SC00INIT               64    IDL
IDLX  SC00INIT  FINAL-ID     64    IDL
IDW2  DFHWBA    WEB INTF.    64         WWW.WEBDEMO2(001)
IDW3  DFHWBA    WEB INTF.    64         WWW.WEBDEMO3(001)
IPCV  SC00INIT               64    IPC
JULA  SC00NATD  ASYNC        64         JUL.ASYNC000(1N  )
SAST  SC00SAST  COMP/PRINT   64
SCFD  SC00DISP  ACCOUNT-ID   64
SCF2  SC00DISP  ACCOUNT-ID   64
SPQR  SC00INIT  FINAL-ID     64    IDL  $ID.STARTUP
```

Startup Member

The key to a production environment is a startup member that is executed when a user signs on to CA Ideal. It contains the SET and RUN commands necessary to run the production application in a sheltered environment. You can establish this environment so that there is no CA Ideal command area (see the following example) and so that CA Ideal is totally transparent to the end user. The startup member is associated with the transaction ID and is specified in the TRNDATA= parameter of the SCF Transaction Table entry.

It executes for any user who specifies that transaction ID. The startup member overrides any signon member that can exist for a user who signs on to CA Ideal using the transparent signon.

The following example illustrates a production environment startup member for CICS:

```
MEM PROD
MEMBER:
         SEQ    DATA
         000100 SEL SYS ORD
         000200 SET CMD LINES 0
         000300 SET CMD SEP N
         000400 SET ENV ACCOUNT-ID ORDS
         000500 SET RUN QUITIDEAL YES
         000600 SET ENV FINAL-ID NONE
         000700 SET RUN CLEAR RESHOW
         000800 RUN ORDERS PROD
```

The commands are as follows:

SEQ 100

Indicates that the ORD system is selected.

SEQ 200

Suppresses the command region by specifying 0 command lines. This prevents the user from issuing any CA Ideal commands.

SEQ 300

Suppresses the line that separates the command and message area from the display area. In addition to restricting the user from using CA Ideal commands, these two commands increase the screen size available for running production applications.

SEQ 400

ORDS is designated as the transaction for which statistics are recorded (instead of SCFD, which is the default when IDLX is invoked). There must be a PCT entry for the ORDS transaction. You can model this entry from the SCFD PCT entry. (For tuning purposes, the priority assigned to ORDS can be higher or lower than SCFD.)

Note: This command is designed for use in CICS. It can be issued in batch,   or it is ignored.

SEQ 500

Issues an automatic OFF when the current RUN ends.

SEQ 600

Does not schedule a CICS transaction when CA Ideal is signed off. This returns control directly to CICS and displays a blank screen.

Note: You can issue this command in batch but it is ignored.

SEQ 700

The CLEAR key refreshes the current panel (instead of ending the RUN).

SEQ 800

The ORDERS program is invoked with a RUN command. This eliminates the need for the end user to know the name of the program or the format of the RUN command. This also places the presentation area under the control of the application at the beginning of the session. Also, since only the PROD versions of programs, subprograms, and dataviews are run in this example, you can use it in a transported environment.

When the first user signs on using a transparent signon-ID, the start-up member is read from the IDDAT member library. The member is then compressed. Leading and trailing blanks are compressed from the member. All comments are dropped. However, multiple blanks between command words are not removed.

Some tips for making the most of this optimization are:

- Use the short version of the command syntax.
- Examine your site options to determine if changing them eliminates the need to specify some commands in start-up members.
- Ensure that you use only one blank between command words

If the compressed member is less than 800 bytes, a CICS GETMAIN is issued for CICS. This eliminates I/O for subsequent transparent signons using that start-up member. To change the member, edit or delete it on the same CICS where the change is needed.

If the compressed member is more than 800 bytes, it is not built in core. If the member name is SIGNON and the user ID is the same as the user signing on, the member is not loaded into global storage.

Signon Exit Program

A site-written program that is called during signon processing can further control the form of transparent signon that enters the user into an application. The signon exit continues to be supported, although its use is not recommended. The signon exit was provided in an early release CA Ideal as a means to provide signon features that have since become part of the base product or have been provided by an external security system.

Signon Processing Execution Flow

The signon process controls the first level of security. During signon, either CA Ideal or an external security system checks to make sure that the user is authorized to access CA Ideal. In general, the CA Ideal signon process works as follows:

1. You sign on to the teleprocessing monitor system. If a security system is used, the teleprocessing monitor system passes control to the security system, which determines whether you are allowed to sign on. It then returns your security ID to the teleprocessing monitor system.
2. You sign on to CA Ideal. The signon command or transaction you enter invokes an initialization procedure, which:
   - Loads the SC00TRAN table and checks for the signon transaction.
   - Loads the SC00OPTS table to determine how to get the identity of the operator.
   - Optionally, checks to make sure you are not already signed on to CA Ideal somewhere else.
   - Loads the IDOPTS table.
   - Checks your external security resource class to make sure you can access the product.
   - Checks for the user ID in the dictionary (User Definition).
   - Executes the signon exit (if necessary).

You can use a default user definition when the user's signon-id is not found in the dictionary.

When you use an external security package to control access to CA Ideal, the security ID identifies the user instead of an operator ID or a terminal ID. There are differences in the parameter values in the SC00TRAN and IDOPTS tables, the site options have different settings, and the security ID finds the user definition in the dictionary.

Establishing the Signon Transaction (SC00INIT and SC00TRAN)

To establish a signon transaction, you must make an entry for that transaction in the SCF Transaction Table (SC00TRAN). In a CICS environment, an entry is also required in the CICS PCT (Program Control Table).

Signon transactions execute the program SC00INIT, which in turn consults the SC00TRAN table for information about how to process this transaction.

By specifying PROD=IDL in SC00TRAN, you invoke the CA Ideal version of SCF-based signon processing. This option controls which product's signon and signoff panels you see and what signon specific product module (SC00xxIN) executes. For CA Ideal, this module is called SC00IDIN. It processes the signon functions customized for CA Ideal, for example, processing the transparent signon member specified in TRNDATA.

SC00TRAN contains information about whether a signon transaction is a standard, express, or transparent signon. The OPTIONS parameter controls this. When OPTIONS = DD,PS the transaction is a standard signon. When OPTIONS=DD,PS,EX the transaction is an express signon or a transparent signon if the TRNDATA parameter is also specified.

Identifying the TP Monitor User (SC00INIT and SC00OPTS)

SC00INIT consults the site assembled table SC00OPTS to determine how the TP monitor is retrieved. The SC00OPTS table SECRTY option affects how the identity of the TP monitor user is determined.

SECRTY Option

The SECRTY option determines whether external security controls access to CA Ideal during signon. The format of this option is:

```
        {Y}
SECRTY= {N}
```

SECRTY=Y-Specifies that an external security system controls access to CA Ideal (and any other SCF-based products using the same copy of SC00OPTS). If you do not specify this parameter, this option is the default.

A call to CAISSF (CA Common Services component) is invoked, determines what external security package is installed, and returns the external security ID that was used to signon to the TP monitor.

When you enter the standard signon transaction, the CA Ideal signon screen displays with the security ID in the User ID field. (If the security ID is not the same as the CA Ideal Person Name, you must define the security ID in the dictionary as an alias for the Person Name.) The User ID and Password fields on the signon screen are protected, so users cannot enter another name or password.

The external security systems that CA Ideal supports can prevent signon if the security system is unavailable or when a user is not defined to the security system. When the external security system is not set up to prevent signon in these circumstances (as frequently happens during initial security implementation), CA Ideal continues the signon process as if security were not enabled. In this case, the CA Ideal signon screen can display with the User ID and Password fields unprotected.

When you enter an express or transparent signon transaction, the security ID must match a Person Name or an alias for a Person Name in a CA Ideal User Definition. If the signon fails because the security system is not available or the user is not defined to the security system, the CA Ideal signon screen can display with the Person Name and Password fields unprotected. If this happens, CA Ideal security facilities control signon access.

A call to CAISSF is invoked to ensure the CICS user has access to the CA Ideal environment. This process is related to the SECPRFX option in IDOPTS described in the section titled Securing User Access by Region in this chapter.

SECRTY=N-Specifies that only internal CA Ideal security features control access to CA Ideal.

This option is a static option in the SC00OPTS module, which you can maintain in a protected data set. You can enter the options in any order. To change the SC00OPTS module, enter the values in the SCBOPTCB macro. If you change the SCBOPTCB macro, you must reassemble and link the SC00OPTS module and, in CICS, recycle CICS to enable the changes.

It is possible to signon to the TP monitor through a security package and still set SECRTY=N. This is not recommended. However, if configured in this fashion, it does require that the security package or other program propagate the TP monitor data CA Ideal requires.

When you enter a standard signon transaction, the CA Ideal signon screen displays with the TP monitor ID in the user ID. See the following chart for the particular TP monitor you are using. The User ID can be optionally protected through SCF option.

When you enter an express or transparent signon, the TP monitor ID is assumed to be the same as the Person name or user ID unless a default CA Ideal definition or alias is allowed.

Signing On in Batch

When you use CA Ideal security facilities to control access to CA Ideal, the SIGNON card, which contains the PERSON and PASSWORD (or PSW) operands, it controls batch access. The SIGNON card is the first input card after the EXEC IDBATCH card in VSE and after the EXEC IDLBATCH card in z/OS. For information about batch jobstreams, see the *Working in the Environment Guide*.

When you use an external security package to control access to CA Ideal, remove the SIGNON card. The user ID is passed to the external security package in the JCL. For example, in z/OS, CA Top Secret and CA ACF2 get the user ID from the USER=parameter on the JOB card.

Checking for Duplicate Users

The SET SITE CHECK DUPLICATE USER command controls whether duplicate signons are allowed. You can also set this option through the command SET COMMAND SITE OPTIONS, which displays a fill-in screen (see Setting Site Options). The format of the SET SITE CHECK DUPLICATE command is as follows:

```
                              {YES}
                              {NO }
SET SITE CHECK DUPLICATE USER {ON }
                              {OFF}
```

YES or ON

Specifies that multiple sessions for the same user ID are allowed. A second signon to CA Ideal by the same user ID is successful, terminating the original session.

Note: This is only effective within a single CICS region. YES is not recommended for a MRO environment with multiple Application Owning Regions. External security software can be used to ensure unique CICS users.

NO or OFF

Specifies that CA Ideal allows duplicate signons. If your external security system ensures unique signons, you should set the CHECK DUPLICATE option (SET SITE CHECK DUPLICATE USER) to NO or OFF.

Securing User Access by Region (IDOPTS SECPRFX = processing when SECRTY=Y)

There are four parameters in the IDOPTS table that affect how signon security is implemented for CA Ideal:

SECPRFX

Establishes the resource class signon entity name used for CA Ideal.

UIDCHK

Determines whether the signon user ID must match the CA Ideal Person Name or User ID.

DFLTUSR

Establishes a default User Definition to use if a match is not found for the signon user ID.

Enter all these options on the FUNC=START statement in the IDOPTSCB macro, which produces the @IIDOPTS load module.

Note: The IDOPTSCB macro specifies site options related to security; therefore, keep the IDOPTSCB macro in a secure location. This macro produces a complete replacement for the site options load module @IIDOPTS.

If you change the IDOPTSCB macro, you must reassemble the source member IDOPTS and link edit @IIDOPTS to implement the changes. (In CICS, you must recycle CICS to implement the changes.)

Establishing the Resource Class Entity Name

The following section details the information about defining specific entities to specific resource classes in the security system.

SECPRFX

In the external security system, users are defined with access to specific entities in specific resource classes. When a user signs on to CA Ideal, the resource class for that user ID is checked to determine whether the user can access the signon entity for CA Ideal. Normally for CA Ideal, the signon entity name is $ISIGNON. However, you can change the name of the signon entity by including the SECPRFX parameter in the IDOPTS table. This lets you restrict access to CA Ideal systems in different CICS regions through the resource class definitions in your security system, based on the signon entities defined for each region with a different @IIDOPTS load module. See the section titled Enabling External Security in this chapter for specific information about external security.

Note: This option is used only with external security systems.

The format of the SECPRFX option is:

```
SECPRFX=xx
```

The value of *xx* is a two-character value used as the prefix to the value SIGNON to create a resource class member name for the CA Ideal environment controlled by this @IIDOPTS load module. If you do not specify this parameter, the prefix is $I.

In a production environment that uses transparent signon, users do not normally see the CA Ideal signon screen. This provides an extra measure of security for the production environment. However, the CA Ideal signon screen displays if an undefined user tries to signon and your external security system does not reject undefined users. To prevent this from happening:

- Define all users of the CA Ideal production environment to the security system, with the appropriate resource class (CACMD) and signon entity.
- Set up the security system so that a resource check prevents signon when a user is not defined or the security system is not active.

If your production region uses a separate @IIDOPTS load module, you can also use the resource class entity names to enhance security for production systems. To use the resource class as an additional security measure for your production region, use the SECPRFX parameter in the IDOPTS table to establish different resource entity names for each CICS region that has a separate @IIDOPTS load module. Then you can define your user groups in the security system with separate CACMD entity names, restricting their access to a specific CICS region.

For example, if you have separate development and production environments, you can set the SECPRFX parameter to two different values, one in each version of the @IIDOPTS load module. If the SECPRFX parameter for the development system is DI and the SECPRFX parameter for the production system is PI, you could then define two or more user groups with access to these different resource class signon entities.

In the following table, three user groups are defined:

- **Developers (DEV1)**-Access only to the development system
- **Quality assurance people (QA)**-Access to the development system for testing, and to the production system for upgrading with approved changes
- **Production system users (PROD)**-Access only to the production system

User Name	Resource Class	Signon Entity
DEV1	CACMD	DISIGNON
QA	CACMD	DISIGNON PISIGNON
PROD	CACMD	PISIGNON

The resource class name is not CACMD for all security products. The value set by the SECPRFX parameter is prefixed to the value SIGNON to create the resource class signon entity name. Specific examples for different security products are shown in the upcoming section titled Enabling External Security.

Determining Alias or Group User Definition

When SC00OPTS SECRTY=N, the UIDCHK option determines whether the TP monitor ID must match the CA Ideal Person Name for User ID.

The format for the UIDCHK parameter is:

```
       {NO }
UIDCHK={YES}
```

NO

Specifies that the TP monitor ID does not have to match the CA Ideal Person Name or user ID. It lets you define a single or multiple TP monitor under a single CA Ideal user definition by adding one or more aliases to the CA Ideal user definition.

YES

Specifies that the TP monitor ID must match the CA Ideal Person Name or User ID. An alias does not satisfy this integrity check. If you do not specify UIDCHK, YES is the default. UIDCHK=YES is ignored when SC00OPTS SECRTY=Y.

Processing Default Users

You can establish a default user definition for all users not specifically defined to CA Ideal by including the DFLTUSR parameter in the IDOPTS table. The format of this parameter is:

```
        {default-id}
DFLTUSR={NONE      }
```

default-id

Identifies the three-character CA Ideal user ID that identifies the user definition used as the default. The authorizations specified in this user definition are used for all users not otherwise defined in the dictionary. The user definition identified by default-ID must be defined in the dictionary, but no aliases are required to access the default user definition.

NONE

Indicates that no default is used. If an operator ID, terminal ID, or security ID used to sign on does not match a user ID, person name, or alias in the dictionary, the signon fails.

Note:

- If a default user is specified in the signon transaction entry in the SCF Transaction Table, it overrides any default user specified in the IDOPTS table.
- If you specify a default user when UIDCHK has a value of YES, the default user person name or user ID does not have to match the signon-ID (operator ID, terminal ID, or security ID), but any non-default user IDs are checked.

Considerations and Examples

Keep in mind the following:

- The password in the User definition is not referenced when the external security system controls signon authorization. However, if external signon fails and CA Ideal security facilities are used, the password is required.
- The security ID must be defined as either a CA Ideal Person Name or alias for a Person Name. The signon-process is more efficient when the SECURITY-ID is defined as a Person Name rather than an alias for the Person Name.
- It is not necessary to define CA Ideal users to an external security system if you are not using one to control access to CA Ideal.
- When you define CA Ideal users to the external security systems, define them with the same name as the Person Name of the CA Ideal User Definition.
- If you do not use an external security to control access to CA Ideal, UIDCHK should equal YES.
- When UIDCHK has a value of NO and no default user (DFLTUSR) is specified, you must define the security ID or the TP monitor ID in the dictionary as a Person Name or an alias for a Person Name to retrieve a User Definition for CA Ideal authorizations.
- For greatest efficiency, define the security ID as an alias when UIDCHK has a value of NO, and as the Person Name when UIDCHK has a value of YES.
- If you specify a default user when UIDCHK has a value of YES, the default user Person Name or User ID does not have to match the signon-ID (operator ID, terminal ID, or security ID), but any non-default user IDs are checked.
- If a user signs on under an alias or with a default user ID, the functions $USER-NAME and $USER-ID return the CA Ideal person name and user ID associated with the alias or default user ID. In addition, that same person name and user ID are placed in the Created by, Last Modified by, and Compiled by fields for any entities created, modified, or compiled by that user.

The SC00OPTS SECRTY parameter affects signon processing as follows:

Signon-ID source

If SC00OPTS SECRTY=Y, then Signon-ID source is the Security ID. If SC00OPTS SECRTY=N, then Signon-ID source is the TP Monitor ID **CICS:** TCTTEOI (OPERID).

SECRPFX= (IDOPTS)

Secures individual access to different regions.

UIDCHK= (IDOPTS)

If SC00OPTS SECRTY=Y, then UIDCHK assumes a value of **NO:** Aliases allowed.

If SC00OPTS SECRTY=N, then if UIDCHK is **YES:** the CA Ideal person name and user ID are valid only. If UIDCHK is **NO:** then aliases are allowed.

DFLTUSR= (IDOPTS)

Specify a default CA Ideal user ID to assign when the signon ID is not defined.

DFLTUSR= (SC00TRAN)

Specify a default CA Ideal user ID for a particular transaction to assign when the signon ID is not defined. It overrides the DFLTUSR defined in IDOPTS, if one exists.

SET CHECK DUPLICATE USER

Enables or disables the check for duplicate signon IDs. Replaces old session with new session when duplicate monitor ID is found.

IDOPTSCB Macros for Security: Samples

The following IDOPTSCB macro is a sample of how you might set up a production system using external security to control access to CA Ideal. For this region, the SCBOPTCB macro contains the parameters SECRTY=Y.

```
IDOPTS CSECT
         IDOPTSCB FUNC=START,                              X
             TYPE=DC,                                      X
             UIDCHK=NO,                                    X
             SECPRFX=IP,                                   X
             DFLTUSR=NONE,                                 X
             ATZEXIT=USERPGM1
         IDOPTSCB FUNC=ATZ,                                X
                 .
                 .
                 .
         IDOPTSCB FUNC=END
```

The following parameters were set in the IDOPTSCB macro:

- The UIDCHK parameter was set to bypass the operator or monitor ID check since the security system performs integrity checks. Your security IDs do not have to match a Person Name or User ID in a valid CA Ideal User Definition. You can use an alias to obtain the CA Ideal authorizations.
- The SECPRFX parameter was set to IP to create a resource signon entity named IPSIGNON. Production users must be defined to the security system with the resource class CACMD and signon entity IPSIGNON.
- The DFLTUSR parameter was set to NONE, which means that users cannot sign on to CA Ideal if their security ID does not match the Person Name or an alias for the Person Name of a valid CA Ideal User Definition.

If your site does not use external security to control access to CA Ideal, the SCBOPTCB macro must specify that SECRTY=N. The IDOPTSCB macro might use the following parameters:

```
IDOPTS CSECT
           IDOPTSCB FUNC=START,                         X
               TYPE=DC,                                 X
               UIDCHK=YES,                              X
               DFLTUSR=DEF,                             X
               ATZEXIT=USERPGM1
           IDOPTSCB FUNC=ATZ,                           X
                       .
                       .
                       .
           IDOPTSCB FUNC=END
```

Since external security is not used to check CA Ideal integrity, the UIDCHK parameter is set to YES. This requires the signon-ID to match the Person Name or User ID of a valid User Definition. If the signon-ID does not match the Person Name or User ID of a valid User Definition, the default User Definition specified on the DFLTUSR parameter is used. For this sample, a User Definition must be present with the User ID DEF. You should set it up to provide the minimum CA Ideal authorizations.

Enabling External Security

The following are the items to consider when enabling external security for CA Ideal. Sites using an external security product to signon to the TP monitor should implement this method that extracts the security-id and removes dependencies on TP monitor-IDs.

CA Common Services Requirements

Install the CA Common Services component, CAISSF.

For more information, see the CA Common Services for z/OS *Services Installation and Maintenance Guide.*

You must assemble CAS9SAFC with CICS=YES.

RACF users must pay special attention to the section titled Customize CAISSF for RACF and RACF-Compatible Products.

Security Product Definitions

Authorize the user for access to the CACMD signon resource.

For CA ACF2 and CA Top Secret, the resource class should already be present. RACF users should have added the resource class in during the installation of CAISSF.

The syntax of the CA Command resource class name is as follows:

```
CA Top Secret:      CACMD
CA ACF2:            CAC
RACF:               CA@MD
```

For CA Ideal, security users require access to CACMD value *sp*SIGNON, where *sp* is the two-character SECPRFX assigned in IDOPTS.

For examples of security product definitions and more information about other SCF-based product requirements, see the *CA IPC Implementation Guide*.

CA Ideal Specifications

You need to:

- Establish a link between the security ID and a CA Ideal user definition using one of the following methods.
- CA Ideal user name can match security-id. If this is not already true, you can modify existing user definitions by changing long names to match security-id names using DDUPDATE. However, it does require that you delete all test and history PERSON (USERS) occurrences first.

  To change the actual person names, you can simply run the following transactions in a single DDUPDATE batch job. Use a set of these transactions for each person name you want to change. Given a table of existing and new names, you can easily write a program to generate the transactions.

  ```
  -UPD PERSON,old-name(PROD,,ovrd)
  1000 NEWNAME,new-name
  -END
  ```

  This method also modifies the DD user signon definition, but not the DQ user in entirety.
- If CA Ideal users are already defined and the security-id does not match the current user-id, add a dictionary alias to the person equal to the security-id. You can easily add aliases in a single batch job executing DDUPDATE. You can update the prod version occurrences.
- Use the DFLTUSR option in SC00TRAN on a transaction basis or in IDOPTS as an environment option. If you specify a default user for the transaction, it takes precedence over a default user specified in IDOPTS. This is only a viable alternative if the CA Ideal user definition used to signon does not need to be distinguishable in CA Ideal.
- Reassemble a separate copy of IDOPTS for each region where you want a different SECPRFX or DFLTUSR. Set UIDCHK=NO.
- The CA Ideal SIGNON statement no longer determines the identity of a batch user since the user ID is passed to the external security package in the JCL. Each security product can function differently, depending on the security inheritance mechanisms in place. Security inheritance becomes an issue when batch jobs are submitted from online.

- Reassemble SC00OPTS with parameter SECRTY=YES. For details on maintaining SC00OPTS, see the *CA IPC Implementation Guide*.
- If you want unique signons, implement using the security product where possible. It is also possible to establish this functionality in CA Ideal by issuing the CA IPC SCF command SET COMMAND SITE OPTIONS and specifying “Check duplicate user:” as Y. This allows each security user a single session in the CA Ideal region.
- To suppress compile messages, issue the CA IPC SCF command SET COMMAND SITE OPTIONS and specify “Asynchronous messages:” as N (None).
- To suppress network print messages, issue the CA IPC PSS command SET OUTPUT SITE OPTIONS and specify “Post successful message:” as N (No).

Programming Considerations

$USER Functions

The $USER-NAME and $USER-ID PDL functions used in CA Ideal programs return the long name and short-id of the CA Ideal user definition accessed for signon. You must consider these functions when implementing the external security product interface during CA Ideal signon processing, or more simply put, when SC00OPTS parameter SECRTY is set to YES.

If the security-ID already matches the 1-15 character CA Ideal user name, it is unlikely that these functions have any impact on existing applications. You should also consider the impact of DFLTUSRs on $USER-NAME and $USER-ID if DFLTUSR is being investigated as a viable option for user definition.

Determine the results for these functions:

- Does the value of $USER-NAME match the security-ID of the user signed on?
- Does the current value of $USER-NAME or $USER-ID have a relationship to any other code or tables?
- Are unique values expected for the $USER-NAME and $USER-ID functions?

The answers to these and other similar questions can influence the type of modifications that you need to make to existing CA Ideal user definitions or existing CA Ideal code.

Submission Utilities

When external security is enabled in the CA Ideal environment, batch signon no longer processes the SIGNON command, but instead uses the JOBCARD USER assignment as the CA Ideal signon-ID. CA Ideal supports utility programs (@I$EXEC1 and pre-r2.2 @I$SUBMT) that submit batch jobs from CA Ideal programs. If programs exist that use this functionality, it is possible you need to modify the programs, depending on the way the JCL input is generated in the code.

Chapter 10: Customizing the CA Ideal Environment

SET SITE commands establish and change default values for an entire site. Any default value set with a SET SITE command becomes the site default and remains in effect until another SET SITE command resets it or until an individual user temporarily overrides it with a SET command.

SET SITE Commands

Each CA Ideal SET option is installed with a default that the CA Ideal Administrator can override using a SET SITE command. The result becomes the site default. Individual commands can set some site options. Others are changed by changing a value on a fill-in.

For consistency, you can keep and execute a member containing all the current site options after an upgrade.

For each upgrade to a CA Ideal system, the default site options are reinstalled. Rather than resetting your options manually each time there is a new release of CA Ideal or trying to remember which options were set with your last release, an alternative is available.

Create a member that contains SET SITE commands to set default site options for your production environment. The next time you upgrade to a new release of CA Ideal, sign on to CA Ideal and execute your SET SITE member. The following is an example of a SET SITE member:

```
SET SITE VERSION LAST
SET SITE EDIT MARGIN LEFT
SET SITE EDIT MULTIPLIER RIGHT
SET SITE RUN LOOPLIMIT 5000
```

After an upgrade to a new release of CA IPC or an increase in the size of your ADROUT library, the output destination commands are also reinstalled.

For a complete description of each SET SITE command's syntax, see the *Command Reference Guide*.

Session Control Facility Options

You can set the Session Control Facility options using a fill-in. The CA Ideal Administrator accesses this fill-in with the following command:

```
SET COMMAND SITE [OPTIONS]
```

The following illustration reflects how the fill-in appears when CA Ideal is first installed. Changes to these options take effect immediately.

```
=>
-------------------------------------------------------------------------------
 IDEAL:       SCF option block          SCF#OPTIONS                     FILL-IN

                        Set SCF   “site”   options
                        --------------------------

 Command “comment”   character:       :
 Command “delimiter” character:       ;
 Command “repeat”    character:       -
 Command “reshow”    character:
 Number of command lines:             3           (0-5)
 Decimal symbol:                      .           (. ,)
 Currency symbol:                     $
 Date format:                         A           (A, E, I)
 Region separator:                    -           (N=none, G=grid, other=itself)
 Asynchronous messages:               U           (U=User, N=None)
 Command reshow?                      N           (Y/N)
 UPPER CASE PANELS AND MESSAGES?      N           (Y/N)
 Translate to upper case in batch?    N           (Y/N)
 Maximum number of regions :          4           (1-4)
 Maximum number of PF keys :          48          (0-48)
 Maximum number of PA keys :          4           (0-4)
 Size of the working buffer:           4000       (4000 - 64000)
 Log file name :                      ADRL
 Trace file name :                    ADRT
 Check duplicate user:                N           (Y/N)
```

The following list provides an explanation of the fields in the previous screen shown:

Command “comment” character

Establishes the default command comment character. This character marks the beginning of a comment in the command area. You can specify any special character (except underscore). (You can use command comments to defer execution of commands online until the comment character is removed or for commands in batch.)

Note: In a CA Ideal member, the colon (:) is the only comment character recognized, regardless of the SCF option setting.

Command “delimiter” character

Establishes the default command delimiter that separates multiple commands entered on a single command line. Specify any non-national special character (except underscore). Each user can specify a different command delimiter character. The command delimiter is valid only for screen input online.

Command “repeat” character

Establishes the default command repeat character. You can enter this character to cause re-execute the most recently executed command. Specify any special character (except underscore).

Command “reshow” character

Establishes the default command reshow character. This character causes all subsequent commands to remain in the command area after execution. Specify any special character (except underscore).

Number of command lines

Establishes the default number of command lines reserved in the command region.

Limit: 0 to 5

Decimal symbol

Establishes the default decimal symbol. The digit separator character defaults to the opposite of this decimal symbol. Valid values are as follows:

- **.** Period
- **,** Comma

Currency symbol

Establishes the default currency symbol. You can specify any character. The installation default is a hex '5B', which displays as a dollar sign on United States English terminals.

Date format

Establishes the default date format. Valid values are as follows:

- **A**-American (MM/DD/YY)
- **E**-European (DD/MM/YY)
- **I**-International (YY/MM/DD)

Region separator

Establishes the default command region separator character. A separator line consists of this character, repeated across the screen, to distinguish between the command area and the execution regions. Valid values are:

- **N**-None (blanks)
- **G**-Grid (a formatted scale line)
- Any special character

Asynchronous messages

Establishes whether asynchronous messages, such as print and compile completion messages, display during the session by default. Valid values are:

- **U**-All asynchronous messages for a signed-on user display during the session. Undisplayed messages from a previous session display at signon.
- **N**-No asynchronous messages display.

You can set this option to NONE to prevent users from receiving someone else's messages from a previous session on the same terminal.

Command reshow

Establishes the default action for commands. Yes retains all commands in the command region after execution. No erases all successfully executed commands from the command area after execution.

UPPER CASE PANELS AND MESSAGES

Establishes the default system panels as uppercase panels and displays all messages in uppercase as the default.

Translate to upper case in batch

Indicates whether text submitted for printing in batch is converted to uppercase. Enter Y (yes) for translation to uppercase or N (no) for no translation.

Maximum number of regions

Establishes the maximum number of regions a user can specify.

Limit: 1 to 4

The command region (region number 0) and product region (region number 1) count as 2, therefore specify 3 or 4 to allow SPLIT to start a second or third session.

Maximum number of PF keys

Establishes a limit on the number of PF keys supported.

Limit: 0 to 48

Maximum number of PA keys

Establishes a limit on the number of PA keys supported.

Limit: 0 to 4

Size of the working buffer

Establishes the size of the buffer CA Ideal editors use for global editing. A larger value speeds up editing at the expense of main memory.

Limit: 4000 to 64000

Log file name

Establishes the CICS TDQueue name of the Log File. Specify one to four characters. The first character must be an alphabetic or national character. The rest can be alphabetic, national, or digits. This name must be the same as the name specified for the IPC DCT entries in CA Ideal's IDSYSFT and IDLCICS and in the procedure that executes IDBATCH.

Trace file name

The CICS TDQueue name of the Trace File. Specify one to four characters. The first must be an alphabetic or national character. The rest can be alphabetic, national, or digits. This name must be the same as the name specified for the CA IPC DCT entries in CA Ideal's IDSYSFT, IDLCICS, and in the procedure that executes IDBATCH.

Check duplicate user

Establishes whether duplicate signons are allowed. Specify Y to prevent users from signing on when they are already signed on with the same host user definition somewhere else. Specify N if your external security system ensures unique signons or if you want to allow duplicate signons.

Note: To check for duplicate user IDs at signon when SC00PTS SECRTY=N, CICS sites must provide a CICS op-ID in the TCTTEOI.

Site Options for Output

To change any of the PSS site options, sign in to CA Ideal and issue one of the following commands:

```
SET OUTPUT SITE OPTIONS
```

Or

```
SET OUT SITE
```

This command retrieves the current set of PSS site option values and displays them in the following panel:

```
=>
  -----------------------------------------------------------------------------
  IDEAL:        SET OUT OPT (PSS)                PSS#OPTIONS          FILL-IN

                        Set PSS  "site"  options
                        ------------------------

  Spool name:                        ADROUT
  Maximum number of lines:           64000      (1-64K)
  Default retention period:          02         (1-99)
  Default number of copies:          01         (1-99)
  Default print status:              RELEASE    (Release/Hold/Keep)
  Default output width:              120        (1-240)
  Default network printer width:     132        (0-240)
  Post successful msg:               Y          (y/n)
  Name of the batch JCL proc:        PSSUTIL
  Default destination
          Type:    LIBRARY                  (LIBrary/NETwork/SYStem/MAIl)
          Name     ___________________________________________________

  Date format                        A          (A, E or I)
  Directory name:                    $PSSDIR$
  Destination table name:            $PSSDST$
  System name:                       PSS
  Prefix name:                       PSS#
  Suppress non-display characters    N          (y/n)
  Maximum retention period:          03         (1-99)
  Maximum number of output members:  0500       (10-9999)
  Percent full occupancy warning:    99         (1-99)
  Multiple CPU environment:          N          (y/n)
```

To display this panel, you must have Print Administrator authorization. If you must change a PSS site option, and you do not have proper authorization, contact your site coordinator.

To modify a PSS site option, overtype a field entry and press Enter. Each change is stored in the PSS and remains in effect until it is modified again.

If you change the value of any of these options, the change affects all of the people using CA Ideal print files. Changes only take effect in sessions that begin after you specify new site options. A person who is using CA Ideal as you specify site options must sign out and then sign in to have the new options take effect.

A subset of the options that are contained on this fill-in is available to CA Ideal users, so that you can override some site options for the current session. The fill-in containing these session options is accessed with the SET OUTPUT SESSION OPTIONS command. For more information, see the *Working in the Environment Guide*.

Spool name

Specify a valid file name. Specifies the ddname of the output library. Before you modify this value, see How to Change Default File Names in the *IPC Implementation Guide for z/OS*.

Maximum number of lines

Specifies the maximum number of lines a report can contain when an output member is generated on-line. The valid values are numbers from 1 through 64000. However, the session option value cannot be greater than the site option value. There is no limit to the length of an output member that is generated in batch.

The following syntax shows the command to set this option:

```
►►─ SET ─┬─ OUTPUT ─┬── MAXLINES ─ nnnnn ─────────────►◄
         └─ OUT ────┘
```

Default retention period

Specifies the number of days an output is retained in the output library before it is automatically removed. The valid values are numbers from 1 through 99; however, the session option value cannot be greater than the site option value.

The following syntax shows the command to set this option:

```
►►─ SET ─┬─ OUTPUT ─┬─┬─ RETENTION ─┬─ nn ──────────────►◄
         └─ OUT ────┘ └─ RET ───────┘
```

Default number of copies

Specifies the number of copies to print if COPIES=*nn* is not specified in the DESTINATION clause of a PRINT command. The valid values are numbers from 1 through 99.

The following syntax shows the command to set this option:

```
►►─ SET ─┬─ OUTPUT ─┬─ COPIES ─ nn ──────────────►◄
          └─ OUT ────┘
```

Default print status

Specifies the default status that is assigned to all generated outputs. Specify one of the following statuses:

- HOL|HOLD retains the print member on the spool until the print status is changed to RELEASE or KEEP.
- REL|RELEASE releases the print member after it prints.
- KEE|KEEP keeps the print member on the spool after it prints.

The following syntax shows the command to set this option:

```
►►─ SET ─┬─ OUTPUT ─┬─┬─ DISPOSITION ─┬─┬─ HOLD ─────┬──────────────────►◄
         └─ OUT ────┘ └─ DISP ────────┘ ├─ HOL ──────┤
                                        ├─ RELEASE ──┤
                                        ├─ REL ──────┤
                                        ├─ KEEP ─────┤
                                        └─ KEE ──────┘
```

Default output width

Specifies the default width for all generated outputs. The valid values are numbers from 1 through 240.

The following syntax shows the command to set this option:

```
►►─ SET ─┬─ OUTPUT ─┬─ WIDTH ─ nn ───────────►◄
         └─ OUT ────┘
```

Note: CA Ideal always overrides the default width internally with the actual report width.

Default network printer width

Specifies the default network printer width for all outputs. The valid values are numbers from 0 through 240.

When an output is routed to a network printer, the printer width is determined in the following manner:

- The value is taken from the BLK/WIDTH column (associated with the network printer) of the destination table (DIS OUT DEST).
- If the value in the BLK/WIDTH column is blank, then the value is taken from the Default network printer width session setting.
- If the value in the Default network printer width option is zero, then the value is taken from the CICS TCT definition.

Post successful msg

Specifies if the user receives the informational message ICPSMSGS31I - Command successfully processed OUTPUT NUMBER=*nnn* when a print request is processed.

Y - Post the message.

N - Most likely used in an application environment when the requester does not need to know the output number.

Name of the batch JCL proc

Specifies the procedure that contains the JCL for processing batch PSS system prints.

The following syntax shows the command to set this option:

```
►►─ SET ─┬─ OUTPUT ─┬─┬─ PROCEDURE ─┬─ procname ──────►◄
         └─ OUT ────┘ └─ PROC ──────┘
```

Note: This procedure must be defined with COPIES=*nn* and DEST=*dest-id* statements. These statements are included in the EXEC SCPSUTIL statement that is generated internally when PSS processes system print requests.

Default destination type

Specifies the printout destination type as one of the following. You enter the name for the system or network name, or CA eMail+ recipient.

SYS|SYSTEM - System printer

NET|NETWORK - Network printer

LIB|LIBRARY - Output library

MAI - CA eMail+

The following syntax shows the command to set this option:

```
►►─ SET ─┬─ OUTPUT ─┬─┬─ DESTINATION ─┬─┬─ SYSTEM ─ name ─────┬────────────────────►◄
         └─ OUT ────┘ └─ DEST ────────┘ ├─ SYS ─ name ────────┤
                                        ├─ NETWORK ─ name ────┤
                                        ├─ NET ─ name ────────┤
                                        ├─ LIBRARY ───────────┤
                                        ├─ LIB ───────────────┤
                                        └─ MAI ─ /recipient/ ─┘
```

Note: Notice that you must use slashes as delimiters in the MAI keyword.

Default destination name

Specifies the name of the SYSTEM or NETWORK printer or CA eMail+ recipient that is used as the default print destination. Specify a valid system, network name, or CA eMail+ recipient.

The following syntax shows the command to set this option:

```
►►─ SET ─┬─ OUTPUT ─┬─┬─ DESTINATION ─┬─┬─ SYSTEM ─ name ─────┬────────────────────►◄
         └─ OUT ────┘ └─ DEST ────────┘ ├─ SYS ─ name ────────┤
                                        ├─ NETWORK ─ name ────┤
                                        ├─ NET ─ name ────────┤
                                        ├─ LIBRARY ───────────┤
                                        ├─ LIB ───────────────┤
                                        └─ MAI ─ /recipient/ ─┘
```

Note: Notice that you must use slashes as delimiters in the MAI keyword.

Date format

Specifies the default date format PSS uses. This value is used during RECOVERY processing. At that time, the Julian date is calculated to determine which outputs exist beyond their retention period. The valid values are:

A - American, date format: *mmddyy*

E - European, date format: *ddmmyy*

I - International, date format: *yymmdd*

Where:

mm Is the month

dd Is the day of the month

yy is the year

Directory name

Specifies the name of the directory that contains a list of all items in the output library. When this entry is changed to an existing name of a directory member name, this member closes and a directory of the same name is reopened. When a directory of the same name is reopened, it is not reinitialized. When this entry is changed to a new directory name, a new spool directory of this name is created and initialized for further processing.

Destination table name

Specifies the name of the table that contains all valid destinations that are defined in the print environment. If you change this name, the existing destination table is released from global storage and then reinitialized on the next destination table access. Also, you lose all output in the destination file.

System name

Specifies a three-character prefix that builds the internal name of each output in the output library, the directory name, and the destination table name. If you change this name, the PSS entities are internally named and referenced differently. Therefore, changing this value has the same effect as changing the destination table name or directory name. After you change this value, reinitialize ADROUT.

Prefix name

Specifies the four-character prefix for all print members. After you change this value, reinitialize ADROUT. This value should be unique for each ADROUT to avoid enqueuing conflicts between members.

Suppress non-display characters

Specifies whether to suppress non-display characters. Set this value to N if unprintable content is needed in control streams that are sent as part of the data, such as escape sequences. With this value set to Y, the data content sent to the printer has any undisplayable characters translated out. For most current printers, the control data is displayable characters. For information on non-display characters, see the *IPC Implementation Guide for z/OS*, How to Modify the PMS Conversion Tables (PMSTRUC, PMSTRND, and PMSTRNDK).

Maximum retention period

Specifies the maximum number of days that an output can be retained. Each user can set a different retention period for each output that is less than or equal to the default. However, this value must be greater than or equal to the value specified as the default retention period. A valid value is a number from 1 through 99.

Maximum number of output members

Specifies the maximum number of outputs that can exist on the output library simultaneously. A valid value is a number from 10 through 9999. This value is used when a directory is initialized to specify the maximum number of entries to create in the PSS directory. You can change this value only if the named directory does not already exist. Changing this value creates a directory in the spool with the number of entries specified.

Percent full occupancy warning

Specifies when PSS issues a library full message. A valid value is a number from 1 through 99. This value determines the point that PSS prevents more outputs from being created, so that it has enough space for outputs in progress to be completed.

Multiple CPU environment

Specifies the processor environment. If you have multiple processors with shared DASD, specify Y (yes). PSS generates a /*JOBPARM SYSAFF=* for batch jobs that are submitted to process PRINT commands directed to a system printer. Otherwise, specify N (no).

Site Control of Wide Panel Support

Wide panel support allows applications to define panels 80 to 256 characters wide. You can make it the site standard, in which case users can disable this support for their session or across sessions. You can also disable wide panel support at a site as the default. Users can then enable wide panel support for a session or the user can modify his SIGNON member to enable wide panel support across sessions for that user.

When wide panel support is disabled, the options for wide panel support on the panel parameters and layout fill-ins are suppressed, which makes the wide panel support transparent.

Enabling or Disabling Wide Panel Support

The command for enabling or disabling wide panel support for a site is:

```
SET SITE PANEL WIDEOPTION ON/OFF
```

Setting a Default Panel Width

You can use the SET SITE PANEL WIDTH command to set the default width for panels created and displayed at the site.

Panels created after this command is issued contain the number of columns specified by the command. However, to change the width of existing panels, you must change the panel parameters fill-in for each panel you want to reflect the new width.

Setting a Loop Limit

The SET SITE RUN LOOPLIMIT command establishes a site maximum for the number of times test-status program loops through a PDL FOR or LOOP construct in a test environment. A runtime error occurs if this limit is exceeded. A user can override this command for the current session.

Setting Environment Options

You can control various characteristics of CA Ideal's environment during a session. They are described in this section. For information about the syntax of the commands described in this section, see the *Command Reference Guide*.

ACCOUNT-ID Specification (CICS Only)

CA Ideal provides the ability to assign different development or production activities to separate user-specified CICS transactions. This allows CICS performance analysis packages or transaction accounting packages to isolate transactions for charge back and resource consumption analysis.

The CA Ideal command SET ENVIRONMENT ACCOUNT-ID establishes the transaction ID under which statistics for the current session is logged.

Each ACCOUNT-ID specified in a SET ENVIRONMENT command must also be defined in CICS as a TRANSACTION invoking program SC00DISP, similar to the CICS TRANSACTION definition for SCFD. In addition, the ACCOUNT-ID can be embedded in an application and can be determined dynamically. IBM standards recommend that transaction IDs not begin with the letter C.

Customizing the End of a CA Ideal Session

FINAL-IDs may be designated as the next transaction to execute at the end of a CA Ideal session. A FINAL-ID can be any valid CICS synchronous transaction, and could also be invoked directly from the terminal. Using FINAL-IDs within a CA Ideal application enables a smooth transition from one CA Ideal application to another, and for COBOL and assembler CICS applications.

FINAL-IDs that invoke CA Ideal applications must be Transparent Signon transactions that invoke the SCF initialization program, SC00INIT. Transparent Signons must have an entry in the SCF Transaction Table, SC00TRAN, which are defined to CICS as a TRANSACTION invoking the program SC00INIT.

The following rules apply to FINAL-IDs:

- A FINAL-ID can never be a transaction which invokes SC00DISP, (for example, SCFD or an ACCOUNT-ID.)
- A FINAL-ID is always a terminal-based transaction that does not require input parameters.
- A FINAL-ID can only be invoked from a terminal-based CA Ideal session.
- Using SET $FINAL-ID from within a CA Ideal program that has been invoked by the INITIATE statement to run asynchronously is not valid.

A FINAL-ID can be designated using the following command:

```
SET ENVIRONMENT FINAL-ID xxxx
```

This command can be executed from the command line or a startup MEMBER.

Like ACCOUNT-ID, there is also a PDL statement that is used to schedule the next transaction after a CA Ideal session:

```
SET $FINAL-ID = 'xxxx'
```

When a CA Ideal session ends, either at the end of a RUN with the command SET RUN QUITIDEAL YES in effect or when OFF has been entered from the command line, the FINAL-ID transaction will be started at the current terminal.

A FINAL-ID may also be set to the keyword NONE that provides a blank screen at the end of a CA Ideal session, instead of the CA Ideal Signoff panel.

Note: FINAL-ID and ACCOUNT-ID transactions can be lowercase characters corresponding to valid CICS Transaction Definitions. Lowercase FINAL-ID and ACCOUNT-ID transactions can be set in a startup MEMBER or using the PDL statement in a program, but cannot be executed from the command line. The special case of FINAL-ID NONE is only valid in uppercase characters.

Selecting an Alternate Currency Symbol

The SET ENVIRONMENT CURRENCY command displays an alternate currency symbol in panels and prints in reports generated by an application during the current session. This command does not affect the currency symbol as it is specified in the edit pattern of a report definition or panel field definition or in a $EDIT function in the procedure of a CA Ideal program.

Selecting an Alternate Date Format

The SET ENVIRONMENT DATEFOR command allows an alternate date format for the date in the heading of a CA Ideal generated listing (for example, a compile listing or print listing).

You can change date formats for a report at run time with the SET REPORT DATEFOR command, not the SET ENVIRONMENT DATEFOR command.

Selecting an Alternate Decimal Symbol

The SET ENVIRONMENT DECIMAL command lets you use an alternate convention for the digit separator and decimal point when displayed in panels or printed in reports generated by an application running during the current session. This command does not affect the decimal symbol as it is specified in the edit pattern of a report definition or panel field definition or in a $EDIT function in the procedure of a CA Ideal program.

The decimal symbol can only be one of the following:

- **. (Period)**-The decimal symbol is a period, and the digit separator is a comma.
- **, (Comma)**-The decimal symbol is a comma, and the digit separator is a period.

  For example, if the decimal symbol is set to a comma, a value set with the following edit pattern displays at runtime:

  ```
  SET ENVIRONMENT DECIMAL ,

  SET X = $EDIT(N, PIC='ZZZ,ZZZ.99')

  2.345,00
  ```

Automatic Off

Specify the SET RUN QUITIDEAL command in the startup member of an application using transparent signon to automatically sign off of CA Ideal when the application completes execution, without displaying the CA Ideal sign-off panel.

User Defined Signoff Panels

When SET RUN QUITIDEAL YES is specified, the SET RUN ERROR-PNL command specifies a panel to display in the event of a fatal CA Ideal internal error during a run. You must specify the panel name, version number, and system.

Setting an Action for the CLEAR Key

The SET RUN CLEAR command determines the action to take if you press the Clear key when a program transmits a panel. This command is only in effect while a CA Ideal application is running. During CA Ideal activities other than RUN, the Clear key continues to return the Main Menu.

Customizing the CA Ideal Options Block Using IDOPTSCB

To change the CA Ideal Options, you must use the IDOPTSCB macro to reassemble the source member IDOPTS, and then link edit @IIDOPTS to implement the changes. (In CICS, you must recycle CICS to implement the changes.) The CA Ideal install job for custom assemblies contains a model of the IDOPTS table that can be used to modify the distributed default values. It also contains JCL that can be used to assemble and link edit the IDOPTS table into @IIDOPTS.

For VSE, the IDOPTS.A book, in the distribution library, contains a model of the IDOPTS table that can be used to modify the distributed default values. The IDOPTASM.Z book, in the distribution library, is the sample for assembling and link editing the IDOPTS table into phase @IIDOPTS.

The IDOPTSCB macro with an explanation of all the parameters that are applicable to FUNC=START is shown following. See Maintaining Authorizations in this chapter for information about parameters specified for FUNC=ATZ. The IDOPTSCB macro contains default values as shown.

```
IDOPTSCB FUNC=,                                                    X
        ATZEXIT=,              Optional Global Auth Exit Name       X
        COMMAND=,              Functional Authorization Name        X
        DB2PLAN=IDP140DV,      Default DB2 Application Plan         X
        DB2SYS=DSN,            Default DB2 Subsystem ID             X
        DFLTUSR=NONE,          Default IDEAL Signon User-ID         X
        DVWLIB=IDDVW,          Dataview Library                     X
```

```
IDENT=IDEAL,          Product Panel Identifier                 X
LEVEL=,               Minimum Level Of Authorization           X
LMTBLD=DD,            Build Load Module Table?                 X
MEMLIB=IDDAT,         Member Library                           X
OBJLIB=IDXXXOBJ,      IDEAL Object Library Default             X
PLALIB=IDDVW,         Application Plan Library                 X
PLTID=IDPI,           Transaction ID For PLT                   X
PLTLOAD=NO,           Load Modules At PLT                      X
PNLLIB=IDXXXPNL,      IDEAL Panel Library Default              X
PSWDIS=VISIBLE,       VISIBLE|INVISIBLE  In DIS USR            X
SECPRFX=$I,           Security Prefix For Resource Chks        X
SORTLIB=,             VLS libs for sort work areas             X
SORTMS=IDSORTMS,      TRAN ID For Build Messages               X
SORTSZ=200704,        TRAN ID For Build  Size Parm             X
SORTWK=IDSORT01,      Batch Report Sort Work Area              X
SORTWP=,              Batch Report DOS WORK= Parm              X
SORTRC=10000000,      On-line Sort Max Number Recs             X
SRCLIB=IDXXXSRC,      IDEAL Source Library Default             X
TYPE=DC,                                                       X
UIDCHK=YES            USER-ID Integrity Check Option
```

At a minimum, you must code the IDOPTSCB macro twice with FUNC=START and FUNC=END. To alter the command authorization levels, code one or more FUNC=ATZ macros.

FUNC=

- START-Creates the beginning of the IDOPTSCB table and the default command authorization levels. It must be the first in a series of IDOPTSCB macros and can be coded only once.
- ATZ-Updates any corresponding entry in the table of command (or functional) authorizations. It is optional and should only be coded when you need to modify the default authorization level for a command (or Functional Authorization) or to invoke a user coded exit routine.

  It must be preceded by a FUNC=START and must not follow a FUNC=END. It can be coded any number of times.
- On the FUNC=ATZ the following parameter is required:

  COMMAND=
- The following parameters are optional:
  - ATZEXIT=
  - LEVEL=

  Note: The preceding parameters for changing the authorization levels are described in detail in the Maintaining Authorizations section in this chapter.

- END- Completes the table. It must be preceded by a FUNC=START and all FUNC=ATZ (if any). There are no other parameters for FUNC=END.
- HELP-Generates a print of the IDOPTSCB macro descriptions.
- ATZEXIT=Specifies the name of an optional user-coded exit module to receive control during the authorization processing of the functions listed under COMMAND=.

  When coded with FUNC=START, it becomes the exit for all listed COMMANDS. However when coded with FUNC=ATZ, the module is invoked only for that COMMAND. The FUNC=ATZ takes precedence over FUNC=START.

  ATZEXIT must be one to eight characters and follow normal load module naming conventions. See the section titled Maintaining Authorizations in this chapter for further details.
- COMMAND= Specifies the Functional Authorization to update. You must code it with FUNC=ATZ. See the Maintaining Authorizations section in this chapter.
- DB2PLAN=For sites with DB2 support, this defines the default application plan name CA Ideal uses when no other name is available. This can be any valid DB2 plan name. The default is release specific (IDP*rrr*DV).
- DB2SYS=For sites with DB2 support, this defines the default Subsystem ID CA Ideal uses to connect to DB2. This can be any valid DB2 Subsystem ID. It is padded with spaces to a length of four characters. The default is DSN.
- DFLTUSR=A three-character short ID of an existing CA Ideal user definition. This parameter specifies a default user ID for CA Ideal signons. During signon, if CA Ideal does not find a dictionary entity occurrence or alias for the supplied user ID, another signon attempt is then made with this three-character short ID. When NONE is specified, no further attempt is done if the original signon fails. The default is NONE.
- DVWLIB=A one- to eight-character name for the CA Ideal Dataview library. The default is IDDVW.
- IDENT=A five-character value that is placed on all of CA Ideal's product panels. Panels of the Inter-Product Components do not use this value nor do customer application panels. If the value CUA is supplied, CA Ideal uses the internal four-character panel name of each panel displayed. The default is IDEAL.
- LEVEL=Specifies the minimum level of authorization required for a given COMMAND=. See the section titled Maintaining Authorizations in this chapter for further explanations.
- LMTBLD= Specifies how the in-core load module table (LMT) is built. The default, DD, specifies that the LMT is built using @ILMLIST and the dictionary module entries. NO specifies that CA Ideal only processes @ILMLIST entries for this region. For more information, see the “Module Format for Programs and Panels” chapter.
- MEMLIB=A one- to eight-character name for the CA Ideal Member Library. The default is IDDAT.

- OBJLIB= A one- to eight-character name for the CA Ideal PDL Object library that appears on the System fill-in panel. The default is IDXXXOBJ.
- PNLLIB= A 1-8 character name for the CA Ideal PDF Panel library that appears on the System fill-in panel. The default is IDXXXPNL.
- PLALIB=A one- to eight-character name for the CA Ideal DB2 Application Plans library. The default is IDDVW.
- PLTID= Identifies the CICS transaction that initializes a control table to manage the optional use of load modules for PDL programs. The default is IDPI.
- PLTLOAD= (CICS only) A YES value indicates user load modules are loaded during the building of the LMT. A NO value bypasses the loading of the modules during the building of the LMT. The verification is done at the time each module is first used. The default is NO.
- PRINT= Controls the printing of the resulting assembler listing. You can use any value supported by the assembler (for example, ON, OFF, GEN, NOGEN). The default is ON.
- PSWDIS= Controls an option to make the PASSWORD field of the User Definition fill-in invisible. The keywords supported are VISIBLE and INVISIBLE. Specifying INVISIBLE also results in an additional field on the User Definition fill-in as a means of verifying the password entered. The default is INVISIBLE. See Creating a CA Ideal User Definition in this guide for more information.
- SECPRFX= A two-character prefix for the CA Ideal security resource checks. This prefix is used during signon to determine whether the user can sign on to CA Ideal. It is also used in front of all resource entity names that CA Ideal performs resource checks on. The default is $I.
- SORTLIB= Specifies the VLS library that contains members used as sort work areas for on-line sorted reports. You can use any one- to eight-character VLS library name. If a value is not specified, the value supplied for MEMLIB is used.
- SORTMS= (z/OS only) A one- to eight-character name for batch sort utility messages for compiles with cross-reference and programs run with sorted reports. The default is IDSORTMS.
- SORTRC= Maximum number of records that can be sorted on-line. If this value is exceeded, a runtime error occurs before the sort begins. The default is 10000000.
- SORTSZ= Core size parameter for the execution of a sort report. The default is 200704.
- SORTWK= A one- to eight-character name for the batch report sort work area. The default is IDSORT01.
- SORTWP= For DOS only, you can specify a value for the SORT WORK= parameter.

- SRCLIB= A one- to eight-character name for the PDL Source library that appears on the System fill-in panel. The default is IDXXXSRC.
- UIDCHK= When NO, this parameter enables the signon user ID to be different from either the CA Ideal person name or short ID that is found in the dictionary. This associates different operator IDs with a common user ID for group signons. It also lets a security ID be different from a CA Ideal person in a security signon. A value of YES requires that the user ID match either the person name or short ID of an existing CA Ideal user. The default is YES.

Example

```
IDOPTS CSECT

    IDOPTSCB  FUNC=START,                           X
    PSWDIS=INVISIBLE,                               X
    TYPE=DC
    IDOPTSCB  FUNC=ATZ,COMMAND=RUN,                 X
    ATZEXIT=RUNCHECK
    IDOPTSCB  FUNC=ATZ,COMMAND=CREATE-PGM,          X
    LEVEL=UPDATE
    IDOPTSCB  FUNC=ATZ,COMMAND=DEQUEUE,             X
    LEVEL=DISABLE
    IDOPTSCB  FUNC=ATZ,COMMAND=COPY-PGM-ACR-SYS,    X
    LEVEL=(UPDATE,CONTROL),                         X
    ATZEXIT=MYPROG
    IDOPTSCB  FUNC=END
END
```

Maintaining Authorizations

Part of the processing of commands entered in CA Ideal is to verify that the user has the proper authorization to execute the command. Each command has a minimum level of authorization required to execute it, and each user is assigned a level of authorization. A table of the currently assigned command authorizations is contained in the @IIDOPTS load module.

Not all command authorizations are controlled using @IIDOPTS. See the "Authorization Table" appendix for a complete list of those that are. In some cases, the CREATE command automatically invokes the EDIT command. Therefore, authorization for the CREATE and EDIT commands for a single entity should be the same or the EDIT-entity command should have a lower level of authorization than the corresponding CREATE-entity command.

The authorization required for CA Ideal commands is defined in the following ways:

- You can leave the default authorizations delivered by CA Ideal unchanged. You can list the delivered authorizations by assembling the IDOPTS source member. You can get additional information by including the FUNC= parameter of the IDOPTSCB macro.
- You can change the defaults during installation or later by modifying the parameters to the IDOPTSCB macro and then reassembling the IDOPTS source member.
- You can also write exit programs for additional authorization checking. Authorization exits are defined to CA Ideal by modifying the parameters to the IDOPTSCB macro and reassembling it..

  You can specify a user exit as global or local. A local exit is executed for a particular command. A global exit is invoked for all commands that do not have local exits specified.

  After CA Ideal performs its authorization checking, the exit is passed information about the user, the current site authorization level for the command, and the service requested. The exit then determines whether to allow the command to execute.

The syntax of the IDOPTSCB macro is described in the section titled IDOPTSCB Macro for Authorization. Modifying the authorization scheme with the IDOPTSCB macro during installation or after is described in the section titled Modifying Existing Authorizations in this chapter. User exits are described in the section titled Authorization Exit Programs, also found in this chapter.

IDOPTSCB Macro for Authorization

To change the @IIDOPTS load module, which contains the current command authorizations, modify the IDOPTSCB macro. (This macro is also used for other functions and contains parameters not described here. A full description is contained in the *z/OS Installation and Maintenance Guide*.)

The following syntax is used for authorization changes in the IDOPTSCB macro.

```
IDOPTSCB FUNC=START,TYPE=DC[,ATZEXIT=exit-pgm]
                    place other site options here
```

FUNC=START

Required preceding any command authorization macros.

TYPE=DC

Required.

ATZEXIT=exit-pgm(Optional)

Specifies a user-written global exit program to receive control each time a command is invoked. You can use this exit to override the action determined by CA Ideal. It does not override any local exits specified in the ATZEXIT parameters.

```
IDOPTSCB FUNC=END
```

FUNC=END

Required following all authorization macros.

```
IDOPTSCB FUNC=HELP
```

FUNC=HELP

Generates a printout of parameter documentation.

```
IDOPTSCB FUNC=ATZ,
COMMAND=function-keyword,
LEVEL=  {level                      }
        {(subject-level,object-level)}
[,ATZEXIT=exit-pgm]
```

FUNC=ATZ

ATZ is required for each command authorization.

COMMAND=function-keyword

Specifies a functional command. This description is limited to word identifying the command that the parameters of this macro apply to. "Authorization Table" appendix relates all commands and their functional keywords.

```
LEVEL=  {level                    }
        {subject-level,object-level}
```

The minimum level of authorization that is required to execute the command identified in COMMAND=. If a command refers to a system other than the current system, you must specify a subject level (for the current system) and an object level (for the system the command refers to). "Authorization Table" appendix indicates the functional keywords that have subject and object levels with asterisks. You can specify the following values:

ADMIN

Requires a CA Ideal Administrator privilege.

BYPASS

Prevents CA Ideal from checking the level, giving authorization to all users. If a user-written exit is provided, it gets control after BYPASS and can override it.

CONTROL

Requires Control authorization in the current system.

DATAVIEW-ADMIN

Requires a Dataview Administrator privilege.

DISABLE

Disables a function. However, even with DISABLE specified, a user-written exit gets control after the DISABLE and can override it.

PRINT-ADMIN

Requires a Print Administrator privilege.

READ

Requires Read authorization in the current system.

RUN-PROD

Requires Run Prod authorization in the current system.

UPDATE

Requires Update authorization in the current system.

UPDATE-PANEL

Requires Update Panel authorization in the current system.

UPDATE-REPORT

Requires Update Report authorization in the current system.

Some levels automatically include others. The levels implied by each privilege and authorization are described in "Defining and Maintaining CA Ideal."

ATZEXIT=exit-pgm(Optional)

The name of a user-written program to receive control each time this command is invoked. You can use this exit to override the action determined by CA Ideal. It is called instead of the global exit, if any, specified in the FUNC=START parameter. The next section describes requirements for exit programs.

Note: If you use a separate @IIDOPTS load module for each environment, you can put all of your authorization exit programs in one library, provided that each exit program has a different name. If you use one @IIDOPTS load module for both online and batch processing, two copies of the authorization exit program are needed in separate libraries, one for batch and one for CICS.

Example

The following example:

- Defines the storage type as DC (required to generate the options block), enforces the user ID integrity check, and specifies a global exit program USERPGM1.
- Disables online compiles for all authorization levels and specifies an exit program USERPGM2 (for example, to enforce a more precise level of authorization).

- For a COPY of a program across systems, requires CONTROL authorization in the current system, but only READ authorization in the system containing the program.
- Requires Administrator privilege for all DUPLICATE PROGRAM commands across systems.
- Requires CONTROL authorization for all batch compiles.

```
IDOPTS CSECT
      IDOPTSCB FUNC=START,                                    X
          TYPE=DC,                                            X
          UIDCHK=YES,                                         X
          ATZEXIT=USERPGM1
      IDOPTSCB FUNC=ATZ,                                      X
          COMMAND=COMPILE-ONLINE,                             X
          ATZEXIT=USERPGM2,                                   X
          LEVEL=DISABLE
      IDOPTSCB FUNC=ATZ,                                      X
          COMMAND=COPY-PROGRAM-ACROSS-SYSTEM,                 X
          LEVEL=(CONTROL,READ)
      IDOPTSCB FUNC=ATZ,                                      X
          COMMAND=DUPLICATE-PROGRAM-ACROSS-SYSTEM,            X
          LEVEL=ADMIN
      IDOPTSCB FUNC=ATZ,                                      X
          COMMAND=COMPILE-BATCH,                              X
          LEVEL=CONTROL
      IDOPTSCB FUNC=END
```

Note: Follow correct assembler syntax. (For example, a non-blank column 72 indicates that the statement is continued on the next line starting in column 16.) If conventions are not followed, other macros might not generate what was requested, even though they do not produce error messages.

Modifying Existing Authorizations

The IDOPTSCB macro produces the @IIDOPTS load module. This procedure is described next. The previous section describes the macro.

1. Code an IDOPTSCB macro with a FUNC=START parameter. (TYPE=DC and any site options not related to authorizations are required with FUNC=START.)
2. Code an IDOPTSCB macro with a FUNC=ATZ parameter for each command that requires an authorization other than the default. Keep them in a separate source module (named USERATZ), which is copied into the IDOPTS source member.
3. Code an IDOPTSCB macro with a FUNC=END parameter.
4. Assemble the source member IDOPTS and link edit @IIDOPTS.

   Member IDOPTASM in the SOURCE library on the installation tape contains either JCL (for z/OS or VSE) that you can use as an example for assembly and link edit.

Note: The IDOPTSCB macro specifies site options related to security; therefore, keep the IDOPTSCB macro in a secure location. This macro produces a complete replacement for the site options load module @IIDOPTS.

Under z/OS or VSE: If authorization exits are specified, different versions of @IIDOPTS might be required for CICS and batch. When you change authorizations, take the appropriate action to recognize the new authorizations:

- **In CICS**-Recycle CICS to recognize the new authorizations. Using the NEWCOPY function of CICS on the options block causes unpredictable results.
- **In batch**-Place the new authorizations in the correct load library before executing the CA Ideal batch job.

Specifying User Exits

Exit programs written outside of CA Ideal must be identified using the IDOPTSCB macro. The section titled Authorization Exit Programs below describes authorization exits.

- **Local User Exits**-To add a user exit program for a particular command, code an IDOPTSCB FUNC=ATZ macro with an ATZEXIT parameter for the command.
- **Global User Exits**-To add an exit program for all commands not governed by local exits, code an IDOPTSCB FUNC=START macro with an ATZEXIT parameter. (TYPE=DC is required with FUNC=START.)

Authorization Exit Programs

A user-written exit can provide an additional layer of authorization to the use of any commands in CA Ideal. When you enter a command, CA Ideal's internal authorization service checks the site option table for the authorization level, for the function, and for the presence of a user exit. If an exit is found, it is called, and the result of CA Ideal's check is passed to the exit.

You must write an authorization exit program following the rules for non-Ideal subprograms (described in the *Creating Programs Guide* and with the CALL statement in the *Programming Reference Guide*).

Under CICS

Because batch and CICS environments use different program linkages, you must create separate programs for batch and on-line or have different option blocks for batch and on-line. In either case, you must maintain separate LOAD/CORE-IMAGE libraries for each environment.

Since an authorization exit is executed frequently, consider performance implications carefully.

CA Ideal passes the exit program the information shown in the following illustration. (The sample programs contain 88-level names with appropriate values for fields containing CA Ideal's internal codes.)

```
01  ID-PARM-1.
    05  ID-EXIT-TYPE                   PIC X.
    05  ID-SYNC                        PIC X(03).
    05  ID-RELEASE-LEVEL               PIC X(04).
    05  ID-USER-SHORT-ID               PIC X(03).
    05  ID-USER-NAME                   PIC X(32).
    05  ID-TERMINAL-ID                 PIC X(04).
    05  ID-TRANSACTION-ID              PIC X(04).
    05  ID-TP-MONITOR-CODE             PIC X(01).
    05  ID-OPERATING-SYSTEM-CODE       PIC X(01).
    05  ID-NETWORK-ID                  PIC X(8).
01  ID-PARM-2.
    05  ATZ-RESULT                     PIC S9(04) COMP.
    05  FUNC-ATZ-CODE                  PIC S9(04) COMP.
    05  REQUIRED-SUBJECT-LEVEL         PIC S9(04) COMP.
    05  REQUIRED-OBJECT-LEVEL          PIC S9(04) COMP.
    05  USER-ATZ-DATA.
        10  GLOBAL-ATZ.
            15 FILLER                  PIC X(03).
            15 IDL-ADM                 PIC X.
            15 FILLER                  PIC X.
            15 IDL-USR                 PIC X.
            15 PRT-ADM                 PIC X.
            15 DVW-ADM                 PIC X.
        10  SUBJECT-ATZ.
            15  SUBJECT-SYSTEM         PIC X(3).
            15  CONTROL-ATZ            PIC X.
            15  UPDATE-ATZ             PIC X.
            15  READ-ATZ               PIC X.
            15  RUN-PROD               PIC X.
            15  FILLER                 PIC X.
            15  FILLER                 PIC X.
            15  UPDATE-REPORT          PIC X.
            15  UPDATE-PANEL           PIC X.
        10  OBJECT-ATZ.
            15  OBJECT-SYSTEM          PIC X(3).
            15  CONTROL-ATZ            PIC X.
            15  UPDATE-ATZ             PIC X.
            15  READ-ATZ               PIC X.
            15  RUN-PROD               PIC X.
            15  FILLER                 PIC X.
            15  FILLER                 PIC X.
            15  UPDATE-REPORT          PIC X.
            15  UPDATE-PANEL           PIC X.
```

```
05  SUBJECT-ENTITY-DATA.
    10  ENTITY-TYPE                PIC X(03).
    10  ENTITY-OCCUR-NAME          PIC X(32).
    10  ENTITY-OCCUR-VERSION       PIC X(03).
    10  ENTITY-OCCUR-STATUS        PIC X(04).
05  OBJECT-ENTITY-DATA.
    10  ENTITY-TYPE                PIC X(03).
    10  ENTITY-OCCUR-NAME          PIC X(32).
    10  ENTITY-OCCUR-VERSION       PIC X(03).
    10  ENTITY-OCCUR-STATUS        PIC X(04).
05  REJECT-MESSAGE                 PIC X(70).
```

ID-EXIT-TYPE

Specifies the type of exit that is invoked for this command. A 1 indicates that an authorization exit is invoked. Other values are reserved for future use.

ID-USER-SHORT-ID

The one- to three-character CA Ideal user short ID defined for the user who is executing the command.

ID-USER-NAME

The name of the CA Ideal user who is executing the command.

ID-TERMINAL-ID

- **In CICS**-The CICS ID of the terminal where the command is executed.
- **In batch**-Hex zeros.

ID-TRANSACTION-ID

- **In CICS**-The CICS signon transaction ID.
- **In batch**-The value IDEA.

ID-TP-MONITOR-CODE

- **C**-Represents CICS.
- **Y**-Represents batch.

ID-OPERATING-SYSTEM-CODE

- **O**-Represents z/OS.
- **D**-Represents DOS/VSE.

ID-NETWORK-ID

In CICS, the VTAM LU name, if the terminal is VTAM. The system ID and terminal ID of the Terminal Owning Region (TOR), if the terminal is MRO but not VTAM; low values in all other cases.

ATZ-RESULT

This field is passed to the exit program. The exit program can modify it. Possible values are:

- 0-The command is accepted.
- 4-The command is denied.
- 6-The command was disabled.

Any other values returned to CA Ideal are ignored.

FUNC-ATZ-CODE

A numeric value representing a functional keyword. "Authorization Table" appendix contains all functions and their associated values.

REQUIRED-SUBJECT-LEVEL

A numeric value representing the authorization necessary in the current system. The possible values and their respective authorization levels follow:

- **254**-BYPASS
- **255**-DISABLE
- **1**-DATAVIEW-ADMIN
- **3**-PRINT-ADMIN
- **4**-RUN-PROD
- **5**-CONTROL
- **6**-UPDATE
- **7**-READ
- **8**-UPDATE-RPT
- **9**-UPDATE-PNL

REQUIRED-OBJECT-LEVEL

A numeric value representing the authorization necessary in the object system. The possible values and their respective authorization levels are:

- **1**-ADMIN
- **2**-DATAVIEW-ADMIN
- **3**-PRINT-ADMIN
- **4**-RUN-PROD
- **5**-CONTROL
- **6**-UPDATE
- **7**-READ
- **8**-UPDATE-RPT
- **9**-UPDATE-PNL

USER-ATZ-DATA

Specifies whether the user has each global privilege and authorization level in the current and object (if needed) systems.

- **IDL-ADM**-CA Ideal Administrator (Y or N)
- **IDL-USR**-CA Ideal User (Y or N)
- **PRT-ADM**-Print Administrator (Y or N)
- **DVW-ADM**-Dataview Administrator (Y or N)
- **SUBJECT-SYSTEM**-Subject System Name
- **OBJECT-SYSTEM**-Object System Name
- **CONTROL-ATZ**-Control Authorization (Y or N)
- **UPDATE-ATZ**-Update Authorization (Y or N)
- **READ-ATZ**-Read Authorization (Y or N)
- **UPDATE-REPORT**-Update Report Authorization (Y or N)
- **UPDATE-PANEL**-Update Panel Authorization (Y or N)

SUBJECT-ENTITY-DATA

Information about the subject entity that is passed to the exit for certain functions.

- ENTITY-TYPE-Type of entity
- ENTITY-OCCUR-NAME-Name of the entity
- ENTITY-OCCUR-VERSION-Version of the entity. For all entities in Production status except panels, this is blank.

ENTITY-OCCUR-STATUS-Production (P) or Test (T) status of the entity. For panels, this is blank.

The data passed to exit programs by each function is shown in the following table.

Functional Keyword	Subject Data	Object Data
ALTER-PROGRAM	Program/Panel	None
CATALOG-DATABIEW	Dataview	None
RUN	Main Program	None
RUN-PROD	Main Program	None
RUN-PROD-USING-PANEL	Main Program	None
RUN-PROD-USING-PROGRAM	Main Program	Program

Note: For functions RUN-PROD-USING-PANEL and RUN-PROD-USING-PROGRAM, the subject entity is the program specified on the RUN command. The authorization applies only if the main program is in Production status.

OBJECT-ENTITY-DATA

Information about the object entity passed to the exit program for certain functions. Entries are the same as for SUBJECT-ENTITY-DATA.

REJECT-MESSAGE

A 70-byte field where the user exit program can specify a message that CA Ideal displays in the message line if the function is denied or disabled.

Defining a CICS Exit Program

The following procedure describes how to establish a CICS exit program. (A sample CICS COBOL authorization program called ATZCBON is available for download from the CA Support website.)

1. Define the program to CICS if CICS Program Autoinstall is not active.
2. Assemble (or compile) and link edit the exit program. Place it in a load library n the DFHRPL or Core Image Library (CIL) in the search chain.

Defining a z/OS or VSE Non-CICS Exit Program

For batch, the exit program only needs to be assembled (or compiled) and link edited into the load library or Core Image Library (CIL) for the job. An exit created for batch does not work in CICS, nor does a CICS exit work in batch.

Managing and Administering Print Services

This section describes how to establish and manage the CA Ideal print environment (also called the Print Subsystem or PSS) and the administration of print services. The print environment includes the facilities to process, route, and manage outputs.

Commands for using the print facilities of CA Ideal include:

- Commands to initiate output requests.
- Command to display the status of an output request.
- Command to delete an output.
- Command to display output destinations.
- Commands to perform many of the above functions for other users' outputs.
- Commands to define or alter print destinations.
- Commands to set options for the print environment.
- Command to define the master or another user's JOBCARD.

Output destination definitions are stored in a member in ADROUT, which is used to create an on-line destination table. When CICS is started, PSS checks for the destination table. If it does not exist, the table is created from the destination definitions stored in ADROUT. To enter destination definitions in the ADROUT member, use the DEFINE OUTPUT DESTINATION command.

Defining Printer Destinations

The printer destination table resides in the ADROUT library. Keep the following facts in mind:

- Each system and network printer must be defined using the DEFINE OUTPUT DESTINATION command.
- You can modify network printers using the ALTER OUTPUT DESTINATION NETWORK command once they are defined.
- LIBRARY is already defined as an output destination when CA IPC PSS component is installed and ADROUT is initialized.

- You do not need to define CA eMail+ destinations. They are defined using CA eMail+ facilities.
- You can define a printer as either a SYSTEM or NETWORK destination. Prints to NETWORK destinations run synchronously. Prints to SYSTEM destinations submit batch print jobs.
- CA IPC PSS supports network printers that are 328x-compatible.
- All TCT entries should exist in the CICS regions sharing an ADROUT library. For more information regarding multiple environments and the considerations for sharing ADROUT, see the "Module Format for Programs and Panels" chapter.

Network Printer Definition Considerations

When defining network printers, you can specify additional characteristics and hardware overrides. These characteristics include:

- Formfeed control
- Header and trailer pages
- Print line width

An output printed at a network printer can be logically viewed as five separate components, four of which are optional.

1. Initial Formfeed
2. Header Section
   a. Header Text
   b. Header Formfeed
3. Output
4. Trailer Section
   a. Trailer Formfeed
   b. Trailer Text
5. Final Formfeed

   `<<Formfeed>>`

   Initial Formfeed generated if FF=YES or FF=HEADER

Header Section printed if HEADER = YES

```
**************************************************************
*
*
*   Start of output listing on printer PR17,  Date 06/27/04 at  12:06:11
*
*   User name    $IDEAL           User ID       $ID
*
*   Output name          WORDNUHS       Output number 0192
*
*   Output destination   NETWORK PR17   Originating terminal 0008
*
*   Output description   LIST STATEMENT OUTPUT
*
*
**************************************************************
```

Output * (always printed)

```
<<Formfeed>>
                      (BEGINNING OF OUTPUT
                                .
                                .
                                .
                          (END OF OUTPUT)
```

Trailer Section printed if TRAILER=YES

```
<<Formfeed>>
***************************************************************
*   End of output listing on printer PR17,  Date 06/27/04 at  12:006:13
*
**************************************************************
```

<<Formfeed>>

Final Formfeed generated if FF=YES or FF=TRAILER

The following is a detailed explanation of some of the options used when defining or modifying network printer definitions. The commands that use these options are DEFINE OUTPUT DESTINATION NETWORK and ALTER OUTPUT DESTINATION. For more information about these commands, see the *Command Reference* Guide.

```
    {YES    }
FF  {NO     }
    {HEADER }
    {TRAILER}
```

Controls the PSS generated formfeeds at the beginning and end of an output routed to a network printer.

FF

- **YES**-A formfeed is issued at the start and end of the print by PSS. This option results in blank pages between PSS outputs printed consecutively. Use this option when non-PSS outputs printed to the same printer do not generate consistent formfeed control.
- **NO**-No formfeeds are issued at the start or end of the print by PSS. This option prints outputs on the same page between PSS outputs printed consecutively. Use this option when the printer does not have a fixed page size, such as rolled paper.
- **HEADER**-A formfeed is issued at the beginning of each new print. This insures that each PSS print starts at the beginning of a new page. Use this option to provide compatibility with the workings of non-PSS output that can also print to the same printer.
- **TRAILER**-A formfeed is issued at the completion of each print. This option places the printer at the top of a blank page after each PSS print is complete. Use this option to remove the printed output from the printer when it is complete without having to wait for the next printout to start or manually intervening with the printer.

  The values HEADER and TRAILER associated with the FF parameter are independent from the HEADER and TRAILER parameters.

PERTASK nnn

Determines how many outputs can process during one print transaction for a specified printer. If the network printer is dedicated to a single CICS region, define this option as 0 to allow the minimum initialization and termination overhead. Set a number greater than 0 to reduce the amount of time PSS can monopolize a printer after a printer is disabled because of physical attributes associated with the printer or the termination of the region.

WIDTH

If greater than 0, the value determines the default width of the printer. If 0, the site option Default network printer width on SET OUT SITE is used. If the Default network printer width is also defined as 0, the TCT definition for the network printer determines the printer width.

Print Service Administration Commands

You can set the print environment site options from a fill-in CA Ideal provides. This fill-in is accessed by issuing the command SET OUTPUT SITE OPTIONS. The section titled Site Options for Output earlier in this chapter describes the print environment site options and the fill-in used to set them.

You can use the following commands to control the disposition, the number of copies, the retention time, and the destination of output. In this table, the abbreviation OUT is used for OUTPUT. For more information about these commands, see the *Command Reference Guide*.

DEFINE OUT DESTINATION

Establishes the output destination of either a system (SYSTEM) or network (NETWORK) printer or the output library (LIBRARY). In the case of a system or network printer, you must also specify the name of the printer. (See Note following this table.)

DELETE OUT DESTINATION

Removes the output destination of either a system (SYSTEM) or network (NETWORK) printer or the output library (LIBRARY). In the case of a system or network printer, you must also specify the name of the printer.

DISPLAY OUT

Displays a specific output from the output library.

PRINT OUT

Prints a specific output from the output library.

ALTER OUT DISPOSITION

Changes the disposition of a specified output.

ALTER OUT COPIES

Changes the number of copies of the specified output to print.

ALTER OUT RETENTION

Changes the number of days the specified output is retained. This value cannot exceed the maximum set for the site.

ALTER OUT DESTINATION

Changes the destination of the specified output.

ALTER OUT DEST DISP

Changes the disposition of all the outputs going to a specific destination or to all outputs regardless of the destination.

ALTER OUT DEST NETWORK

Changes one of the following characteristics for the specified network printer:

- Formfeed control
- Whether a header and trailer page is printed
- The width of the printer's print line

DELETE OUT

Removes an output from the output library.

LIBRARY is already defined as an output destination when CA Ideal is first installed.

You do not need to define CA eMail+ destinations to CA Ideal. They are defined using CA eMail+ facilities.

CA Ideal supports network printers that are 328x-compatible.

When defining a network printer, you can also specify the following characteristics to override the hardware specifications:

- Whether a header and trailer page is printed.
- The width of the printer's print line.

System Printer Considerations

The following considerations are to be taken while submitting a batch job to the system printer.

Editing a Jobcard z/OS

To satisfy output requests with a destination of a system printer, the CA Ideal print environment submits a batch job to the host operating system. The CA Ideal print environment provides resources to store a jobcard for each user ID and a MASTER jobcard. The MASTER jobcard is used when the current user has no jobcard. The MASTER jobcard is also used as the initial model for editing when a user first issues the EDIT JOBCARD command (to create a user jobcard). If a CA Ideal user submits a member without a jobcard, CA Ideal uses this same user jobcard or MASTER jobcard.

The PRINT Administrator can display or edit any user jobcard or the MASTER jobcard with the command:

```
{DISPLAY}          [USER user-id]
{EDIT   } JOBCARD  [MASTER      ]
```

A jobcard fill-in consists of five 71-column lines. You can enter any JCL statements that are valid as the first one to five statements in a batch job stream. For more information about the DISPLAY and EDIT JOBCARD commands, see the *Command Reference Guide*.

Batch Output Procedure

The rest of the JCL to carry out the batch print request is identified on the SET OUTPUT SITE or SESSION panel as Name of the batch JCL proc.

BLOCKSIZE (z/OS Only)

This affects the blocksize assigned to the AUXPRINT file assigned to the batch output procedure. It is most likely that this option is used when AUXPRINT is defined as a data set rather than a SYSOUT file in the batch output procedure.

Creating User-Defined HELP Members

This section explains how to add your own HELP members to use in CA Ideal, how to disable and enable an active on-line application so you can apply changes, how to duplicate an entity across systems, and how to dequeue an entity that is erroneously enqueued.

You can extend the CA Ideal HELP system by adding your own HELP information.

You can create or edit help members outside of CA Ideal and import them into CA Ideal using the Source Transport Utility (IDUTSTRN). You can add or replace user-defined help members. You can also replace the CA Ideal help SITE member. (It contains an index into other help members that were added.) You cannot replace other CA Ideal system help members.

Adding New Help Members

To add a list of help members, follow the steps:

1. Create a sequential 80-byte input file containing one or more help members. For example:

   ```
   ->HELP MYHELP
   DESC 'User defined HELP member'
   ->HELP-DATA
   Help me if you can I'm feeling sad.
   ->END-HELP
   ```

   The control cards beginning with -> must start in column 1. The DESC is optional and cannot exceed 24 characters. When a help member displays, any text lines that begin with an equal sign (=) are highlighted.

   The end of a help member is recognized by ->END-HELP beginning in column 1, by ->HELP beginning in column 1 (which starts a new help member), or by the end of the input data set.

2. Use Source Transport to import the help member.
3. In CA Ideal, type HELP HELPNAME to display the member. For example, the following displays the member MYHELP.

   ```
   HELP MYHELP
   ```

The format of help members and the use of the Source Transport Utility are described in the *Working in the Environment Guide*.

Displaying the HELP Member Index

User-defined HELP members are stored in the VLS library ADRLIB. The member name consists of the prefix USR, the help name you defined, and an H in column 21. For example:

```
USRMYHELP           H
USRSITE            H
```

CA Ideal system HELP members start with the prefix HLP.

You can display an index of all members in ADRLIB using the command:

```
DISPLAY INDEX MEMBER USER @I$ADRLIB
```

CICS Requirements

Case Translation

To enable case translation in CICS the following changes must be made:

- On the TYPETERM definitions UCTRAN must be set to TRANID. Specifying TYPETERM UCTRAN(TRANID) causes the transaction ID to be translated to uppercase; other input is translated according to what is specified for PROFILE UCTRAN.
- On the profile definition associated with SCF-based transactions (IDEA, DDOL) UCTRAN should be set to NO. This allows the terminal to run in mixed case mode. Transactions that require uppercase should be associated with a profile that has UCTRAN set to YES. CICS supplied transactions such as CEDA, CEMT, CECI, etc. are an example. IBM distributes the profile associated with those transactions with the parameter UCTRAN=NO. If uppercase translation is desired on those transactions, then the UCTRAN parameter should be changed to YES.

See the IBM documentation for more information regarding UCTRAN.

Operating System Requirements

Administrative tasks can differ, depending on the operating system. These tasks are described in this section.

CA Ideal Batch File Table

Files accessed by CA Ideal batch that are not CA Datacom/DB files require entries in the CA Ideal File Table just as libraries accessed by CA Ideal application developers (source, object, and panel libraries) require CA Ideal File Table entries. The CA Ideal File Table is used for CA Ideal batch jobs. It is effectively a collection of DCBs DTFs for accessing any non-CA Datacom/DB files that the job needs. A file table is generated during installation with all the necessary entries for batch processing. This section describes how to add to or change the file table to provide additional site system libraries (source, panel, and object) for application developers and to support sequential file dataviews for CA Ideal batch applications.

Note: Since non-system VSAM files do not require any file table entries, it is not necessary to change the file table to support VSAM file dataviews.

The CA Ideal Batch File Table is a series of assembler macro invocations. Each entry calls the macro ROSFD and describes one file. To change the CA Ideal File Table, add the appropriate entries and reassemble. Then, execute a standalone link-edit with a phase name of @IIDSYSF with the object code in a core image library available at runtime. No other link-edits are required since the file table is dynamically loaded at runtime.

ROSFD Entry

You can specify the following parameters for a ROSFD entry:

ACCMETH= BDAM|SEQ

Identifies the type of access. You must specify BDAM for VLS files. All others must be SEQ.

ADDRESS=RELTRK

Indicates that BDAM VLS entries use relative track addressing.

BLKSIZE=nnnnn

Specifies the block size for the file. For sequential dataviews, see the *Creating Dataviews Guide* for details on how this value is overridden.

CTLCHR=ASA

Identifies ASCII control characters for printer files.

DDNAME|DTFNAME=xxxxxxx

For z/OS sequential file entries, DDNAME identifies the ddname of the file. For VSE BDAM, sequential DISK, and sequential tape entries, DTFNAME identifies the DLBL or TLBL name. For information about how CA Ideal dynamically changes this name to the actual MONITOR-NAME specified in the dictionary, see the chapter on sequential file dataviews in the *Creating Dataviews Guide.*

DEVADDR=SYSxxx-(VSE only)

Identifies the logical unit assignment for this file. It only has meaning for card input, printer, punch, and tape entries. Sequential disk and BDAM entries all use SYS000, which you can override with JCL.

DEVICE=nnnn (VSE only)

Identifies the device type such as 3390. The device must only be of the correct type, not necessarily the actual unit where it runs.

DTFTYPE=DTFPR|DTFCDP|DTFSD|DTFMT (VSE only)

Indicates the DTF type as printer (DTFPR), punch (DTFCDP), sequential disk (DTFSD), and magnetic tape (DTFMT) entries, respectively.

FILABL=STD|NO

Identifies whether a tape file has standard labels or no labels.

IBLKSZ=nnnnn

Parameter used for sequential disk dataviews. Identifies the maximum input block size. For more details about sequential dataviews, see the *Creating Dataviews Guide*.

LRECL=nnnnn

Specifies the record size for the file. For sequential dataviews, see the *Creating Dataviews Guide* for details on how this value is overridden.

OBLKSZ=nnnnn

Parameter used for sequential disk dataviews identifies the maximum output block size. It must be eight greater than the IBLKSZ for the same ROSFD entry. For more details, see the chapter on sequential file dataviews in the *Creating Dataviews Guide*.

OPSYS=DOS|VSE (VSE only)

Identifies VSE as the operating system. This is a required parameter, since if omitted the default is to generate entries for z/OS.

PRODUCT=

Obsolete.

RECFM=F|FB|FBA|FA

Identifies whether a file has fixed (F), blocked (B), or fixed block (FB) format, and whether control characters are ANSI characters (A).

Examples: VSE

The following is a sample printer entry:

```
SYSPRINT ROSFD  ACCMETH=SEQ,                                          X
                DTFTYPE=DTFPR,                                        X
                DEVICE=1403,                                          X
                DEVADDR=SYS041,                                       X
                RECFM=F,                                              X
                BLKSIZE=133,                                          X
                CTLCHR=ASA,                                           X
                OPSYS=VSE
```

SYSIN and SYSPRINT are the logical names CA Ideal uses to access these entries.

Note: When creating or modifying CA Ideal system definitions, you must specify the name of the source, object, and panel library names residing on VLS (Virtual Library System). You must describe each VLS library specified in this way with an entry in the IDSYSFT. The following is a sample VLS BDAM entry:

```
IDL$IDS  ROSFD  ACCMETH=BDAM,                                         X
                DEVICE=3390,                                          X
                                  DEVADDR=SYS000,                                X
                DTFNAME=SRC$ID,                                       X
                ADDRESS=RELTRK,                                       X
                RECFM=F,                                              X
                BLKSIZE=1960,                                         X
                OPSYS=VSE
```

IDL$IDS is the name of a library the user defined when doing SYSTEM definition. SRC$ID is the DLBL name to use in the JCL to define this file. You must override SYS000 in the JCL to assign the file to the correct unit.

Changing the CA Ideal File Batch Table

Certain changes to the CA Ideal environment require reassembling the CA Ideal Batch File Table. They include:

- Adding new source, panel, and object libraries for new application systems.
- Changing block sizes of VLS libraries.
- Changing or adding entries for sequential file dataviews processed by CA Ideal batch applications (see the *Creating Dataviews Guide*).
- For VSE, changing the logical unit assignments for unit record and tape files. During installation, the initial file table is generated based on installation parameters. The Site Administrator can change them further as long as no logical unit conflicts are introduced. Notify CA Ideal application developers of these assignments since their JCL for batch applications can depend on them, especially for report output files and sequential dataview files.

The following are the installed file table entries (except sequential dataviews, which are described in the next section). Each entry includes a brief description of what it is and what can be changed.

CA Ideal System Files

Do not change anything in this group (except the logical unit assignments for VSE).

- **ADRL**-Log file for internal errors
- **ADRT**-Internal trace file for CA support and diagnostic purposes
- **AUXPRINT**-Output for prints submitted from on-line to a system printer or batch reports if external files are not assigned
- **SYSIN**-Command card input
- **SYSDIAL**-Internal debugging print file for CA support and diagnostic purposes
- **SYSPRINT**-Standard listing file
- **XPRT**-Tape file for importing/exporting programs

CA IPC VLS Files

- **ADRLIB**-SCF and PSS (internal CA components) control information and jobcards
- **ADRPNL**-CA Ideal system panels

Application VLS Files

VSE

Block sizes can change for these entries. You can also modify DTFNAME to change the DLBL name. You can also add new entries for new source, panel, and object system libraries.

- **ADROUT**-Print Subsystem (PSS) library
- **IDDAT**-Member library
- **IDDVW**-Dataview object library
- **IDL$IDS**-Installed $ID system source library
- **IDL$IDP**-Installed $ID system panel library
- **IDL$IDO**-Installed $ID system object library

Z/OS

Block sizes can change for these entries. You can also modify the ROSFD DDNAME parameter to change the JCL DD name. You can also add new entries for new source, panel, and object system libraries.

- **ADROUT**-Print Subsystem (PSS) library
- **IDDAT**-Member library
- **IDDVW**-Dataview object library
- **ID$IDSRC**-Installed $ID system source library
- **ID$IDPNL** - Installed $ID system panel library
- **ID$IDOBJ**- Installed $ID system object library

PSS Files

You can only change logical unit assignments. The entries PSSPRT01 through PSSPRT15 are used for CA Ideal report output, LIST statement output, and PRINT command output.

VSE Report Work Files

Do not change the entries IDRWK01 through IDRWK15. RDF uses them internally when processing sorted reports. They are not SORT work files.

- **Block size**-Physical block length. This information is obtained from the dictionary when the dataview is cataloged. You can override it with an ALTER PROGRAM command or ASSIGN DATAVIEW command before the run. At runtime, the block size is stored into the ROSFD entry before opening the file.

  The block size specified in the ROSFD is the maximum block size that you can use. If a dataview is opened with a larger block size than the maximum specified, a runtime error occurs with an I7 dataview status code.

  For tape, print, and punch files use the BLKSIZE parameter. For disk files, use the IBLKSZ and OBLKSZ parameters. IBLKSZ is the maximum block size. OBLKSZ is the maximum block size plus eight.

- **Filename**-The DLBL for disk files and TLBL for standard-label tape files is obtained from the dictionary when the dataview is cataloged. This name overrides the DTFNAME parameter on the ROSFD before opening the file.

- **Logical Unit Assignment**- For standard-label tape (SLTAPE), non-labeled tape (NLTAPE), printer (PRT), and punch (PUNCH) files, the default assignment comes from the ROSFD entry. You can override it with an ASSIGN DATAVIEW before the RUN command (see the *Creating Dataviews Guide*) or with an ASSIGN statement executed before the first FOR construct referencing the dataview.

The logical assignments for disk files are made entirely with the JCL.

If a run needs to access more than one sequential file dataview of the same type (for example, two disk or three standard-label tape files), then you must add new IDSYSFT entries. To add a new entry, make a copy of an existing entry of the same type. Change the ROSFD label from *Ixxxx*1 to I*xxxxn* where *xxxx* is DISK, SLTAPE, NLTAPE, PRT, or PUNCH and *n* is the next available number for that device type, in hexadecimal sequence (1-9, A-G). For SLTAPE, NLTAPE, PRT, and PUNCH entries, change the logical unit assignment to something that does not conflict with any other ROSFD entry in the file table. Optionally, adjust the block size parameter as previously described.

If multiple sequential files of the same type in the same run are referenced and there are not enough file table entries defined, a dataview runtime error occurs with an 'I8' status code.

Requirements for a Single z/OS System

All CA Ideal tasks must run on a *single* z/OS system because the IBM z/OS operating system does not provide cross-system protection from concurrent destructive updates, except through the RESERVE macro. VLS (Virtual Library System) uses an ENQ macro to protect against concurrent destructive updates on a single system, but does not issue a RESERVE to avoid causing indefinite waits in the teleprocessing monitor. Therefore, it is imperative that any site using CA Ideal ensure that all CA Ideal tasks are run on the same system by taking the following actions:

1. Ensure that multiple teleprocessing monitors supporting CA Ideal run on the same system.
2. Ensure that all PRINT commands where the destination is a system printer direct the resulting batch print job to the same system where the teleprocessing monitor is running. For JES2 only, if you specify MCPU=YES in the session control options block (see CA Ideal installation procedures for z/OS), CA Ideal automatically generates the following statement:

   ```
   /*JOBPARM S=*
   ```

3. Ensure that all other batch jobs, including CA Ideal batch (the IDBATCH proc), VLS utility runs (VLSUTIL proc), and CA Ideal Transport Utility runs (IDLXPRT proc) run on the same system as the teleprocessing monitor. Use one or more of the following methods to ensure that batch jobs are all on the correct system:

   a. Specify a job class known only on that system for CA Ideal runs.

   b. Enforce the use of a /*JOBPARM (JES2) or //*MAIN (JES3) statement.

   c. Catalog at least one CA Ideal file only on the master catalog for the system.

Chapter 11: Optimizing Storage Management

The information contained in this chapter is intended to help you understand and optimize storage management in your CA Ideal regions. The following items are covered:

- User application storage-how to decide what format, load modules or VLS format, is best for performance.
- CICS storage use-how CA Ideal uses DSA, above-the-line storage, and temporary storage under CICS
- Releasing session storage-how to effectively free up unneeded storage through coding practices and node error recovery procedures.

Enhancing User Storage Management

A CA Ideal program or panel produces the same results regardless of its format. This includes load module and VLS object, but performance (runtime, response time, memory required) varies depending on the format that is chosen.

Load Module Format

For optimal performance in a CICS production environment, convert CA Ideal user programs and panels to load module format. Load modules provide performance benefits and a means of monitoring and tuning your environment that VLS format does not.

For more information about load modules, see the “Module Format for Programs and Panels” chapter.

Programs in VLS Format

For CA Ideal programs that run as VLS objects in a CICS environment, each user concurrently running the program has a separate copy of the entire object program. The updateable part is written and accessed in the EDSA, CICS Extended Dynamic Storage Area. At the end of each transaction, the readonly portion of the program is written to CICS temporary storage. When the next transaction for that program is executed, the object program is reloaded from CICS temporary storage.

Panels in VLS Format

Under CICS, the panel buffer is always written to temporary storage at the end of the transaction and reloaded into DSA when the next transaction for that panel is executed. Each panel has one member in load module or VLS format. However, at runtime, when a panel is loaded, two portions of the panel are built:

- The reentrant portion (non-updateable) that contains the Panel Control Block (PCB).
- The non-reentrant portion (updateable) that contains the Panel Buffer (PBF). Under CICS, the panel buffer is always written to temporary storage at the end of the transaction and reloaded into DSA when the next transaction for that panel is executed.

Each user concurrently running the panel has a separate copy of the entire panel. The panel is written to CICS temporary storage at the end of each transaction. When the next transaction for that panel is executed, the panel is reloaded from CICS temporary storage

Recommendations

Programs and panels executing in a production runtime environment should be in load module format for optimal performance.

CICS Storage Use

CA Ideal was designed as a pseudo-conversational system under CICS. From the database environment, the DBMS CICS interface issues a checkpoint at the end of a CICS transaction, dropping all locks, that is, when CA Ideal issues a TRANSMIT. CICS virtual storage resources must also be freed at task termination and saved on some storage medium.

The point of a pseudo-conversational system is to free resources at the termination of each task to allow other tasks to acquire those resources without contention. This allows more transactions to process without requiring exponentially more resources.

CA Ideal uses three basic types of CICS storage:

- Auxiliary Temporary Storage
- Extended Dynamic Storage Area (EDSA)
- Dynamic Storage Area (DSA)

In general, with the exception of certain global control blocks, CA Ideal will release DSA at the end of a task. Storage acquired from EDSA and Temporary Storage make up the CA Ideal session, which is released at the end of a session.

For more information about determining the effects that are specific to your CICS release, see the IBM CICS documentation.

Temporary Storage

Because CA Ideal executes in pseudo-conversational mode, there is significant use of CICS auxiliary temporary storage. You can define CICS auxiliary temporary storage to reside on a VSAM data set or be diverted to CICS main temporary storage. See IBM documentation for more information regarding CICS temporary storage use and its location.

Consider a user session to be the processing that occurs from the time signon to an SCF-based transaction occurs until signoff is complete. A series of control blocks keep track of each user's session.

Many of these control blocks are written to CICS temporary storage at transaction termination and retrieved into temporary storage buffers that reside in DSA at transaction initiation. In an attempt to minimize the number of writes to temporary storage, these control blocks comprise a single temporary storage record that can be as large as 32 KB.

The most significant of these control blocks include:

- SCB (Session Control Block)
- RCB (Run Control Block)
- Updateable user code copies

Important! Never modify or delete any CA temporary storage queues. This would very likely cause storage violations in CICS. See Session Storage Cleanup in this chapter for information regarding the PURGE termid command and VPE code invoked through NEP.

SCB (Session Control Block)

This piece of storage keeps track of what is occurring in an SCF-based session. This data consists of a single temporary storage record allocated during signon to an SCF-based session. Some of the information contained in this data follows:

- Products active, IDEAL, DDOL, IPCV, ...
- Current panels:
  - Command line and message line is a panel.
  - User's panel is a panel.
- User's SCF session options:
  - All SET command settings.
- SCF CVT (Communication Vector Table):
  - A directory of other control blocks.
- PMS control blocks that control presentation area parameters such as:
  - How many lines are in each region
  - How wide is the screen
  - Where is the current scroll position
- Can contain RCB if a CA Ideal run is active, if room permits; otherwise, becomes its own control block (described in the next section).

RCB (Run Control Block)

For each session that runs a program, the RCB (Run Control Block) keeps track of the processing occurring for the executing application. The run control block contains:

- Current position in the application:
  - Procedure block
  - Current instruction (t-code)
  - Compilation release
- Executor work areas
- Updateable code pointers
- A list of all programs and panels that were accessed in the current execution of the application

Updateable User Code Copies

Temporary storage stores the updateable portions of an application's active user panels for each user's session between transactions.

Temporary Storage Record Naming Conventions

Assuming the following:

tttt-CICS terminal-ID

?-Product generated character

!xx!-Hex characters

uid-CA Ideal short-ID user definition

sys-CA Ideal short-ID system definition

Most of CA Ideal Temporary Storage records are built using the following naming convention:

```
$t?ttt??
```

There are two other types of records you find that do not follow this naming convention. These two types of records represent the CA Ideal users and CA Ideal systems.

User Records

Most of these records have a length of 2048 bytes.

When SECRTY=YES: ??????!6200!

When SECRTY=NO: $I?uid!6200!

The records are not deleted even after the user signs off CA Ideal. These records eliminate the need to re-access the dictionary for signon information, thereby improving signon performance for subsequent signons to CA Ideal for manual signon or transactions invoked by $FINAL-ID.

System Records

A temporary storage record is explicitly written to main temporary storage normally consisting of 90 bytes for each CA Ideal system that was accessed in the particular CICS region.

Its naming convention is: $I!00!sys!6100!

These records are not deleted from temporary storage unless the CA Ideal system definition is updated in the CICS region. The presence of these records improves performance by eliminating the need to access the dictionary each time a system is accessed in the region.

VPE writes the following two types of temporary storage records:

A Temporary Storage record for each terminal session called the VPE Session Anchor Block (VSAB). It contains a list of all of the active control blocks, Temporary Storage records, addresses to EDSA for the user's session. Its naming convention is VSAB*tttt*.

A Temporary Storage record with the naming convention:

$VPtttt?.

Datadictionary also builds some temporary storage records to maintain information across transaction boundaries that it needs. These records are named as follows:

#tttt???

Dynamic Storage Area (DSA)

CICS normally classifies storage in the DSA into different storage types. For consistency sake, the storage CA Ideal is acquiring is classified into ISOLATED, SHARED, and PROGRAM storage.

Isolated Storage

Isolated storage is allocated during the task and released when the task is complete.

The temporary storage records that were identified are brought into DSA during active transactions in temporary storage buffers in this type of storage.

Isolated storage is also acquired and released in a single transaction for internal work areas that CA Ideal might need to acquire during a task for internal processing.

Shared Storage

Shared storage is also used for system control blocks such as the in-core Load Module Table, the Network Print Table, the SCF Signon Table, the SCF Transaction Table, and TRNDATA members.

Program Storage

Most of the CA Ideal, CA IPC, and Datadictionary system modules and all 24-bit CA Ideal user programs and panel load modules that are not marked as resident are loaded into DSA program storage.

The updateable copies of program storage are freed:

- When a RELEASE PROGRAM statement is issued
- When the application ends
- When VPUR is invoked

Extended Dynamic Storage Area

Upon initial access by a user to a CA Ideal program, a copy is made of the updateable portion of the program. VPE issues a CICS GETMAIN request for storage above the line and stores the updateable application code. This storage is freed at the end of the run.

CA Ideal Control Blocks Written to Temporary Storage

The CVT (Communication Vector Table), the Dataview Stack, the SCB (Session Control Block), the Assign Area, and (space permitting) the RCB (Run Control Block) comprise a single temporary storage record that can be as large as 32 KB. Some sites might find that a CISIZE of 32 KB gives them optimal performance. You can start at a lower CISIZE, such as 22 KB, and monitor the CICS temporary storage statistics for the WRITES GREATER THAN CI. For optimal performance, this value should be zero or a very low number to avoid excessive CICS record splitting.

31-Bit RMODE(ANY) User Program Load Modules

Load modules are generated when the CREATE MODULE ... FOR PROGRAM ... statement is issued. The objects are created in 31-bit mode.

User Panel Load Modules and RMODE

Only 24-bit (RMODE=24) is supported for user panel load modules. The CA Ideal CREATE MODULE ... FOR PANEL ... creates all panels in 24-bit mode. If a panel load module was created in 31-bit mode, it would be loaded above the line and would be inaccessible to CA IPC component PMS, which can only handle 24-bit addresses at this time.

Performance Considerations

This section details the performance considerations.

Application Design

CA Ideal will request EDSA whenever possible, but CICS above the line storage is not limitless. Programs that are no longer needed should free the storage they are no longer using. It is important to recognize, however, exactly what is meant by the term no longer needed. An example of a program that is no longer needed is an initialization program. This type of program is called at the beginning of a run and never called again. Another example is the leg of an application.

For instance, if you have a menu program that calls programs that are considered applications and if users spend a long period of time in one application before moving to another piece of the menu, you might consider using RELEASE PROGRAM statements on the programs belonging to the application. This type of application can also lend itself to setting up each leg of the application as its own transaction and using FINAL-ID to invoke the transaction. The invoked transaction can then invoke the menu transaction when it is complete. This frees up all resources because you ended the CA Ideal run.

Important! Do not follow CALL program PDL statements with RELEASE PROGRAM statements as a general rule.

Tuning Storage

Depending on the requirement, you need more XA storage than recommended.

Unused storage

CA Ideal applications should release programs that are no longer used, however, use care to not misuse the function. You do not want to sacrifice performance.

Timeouts and Line Drops

Timeouts and line drops are probably the leading cause of unused CA Ideal storage.

See information about timeouts, line drops, and VPE Node Error Program in this guide.

VLS object format

In a production environment, if programs are running in VLS object format instead of load module format, more storage than necessary is being used. This storage most definitely is DSA and extended storage if auxiliary temporary storage is defined to main storage.

Unused load modules

User load modules are loaded in extended storage. If CICS is brought up with PLTLOAD=YES, all modules are loaded and released at startup, regardless of whether they are ever used in the region. This includes all symbol table portions of every program load module, which are only referenced during error processing. Setting PLTLOAD=NO eliminates this additional processing.

If you are using storage efficiently and are still experiencing a storage shortage, you may need to increase the amount of EDSA in the region.

Session Storage Cleanup

This section presents the different ways that are available to improve performance, enforce security, and purge a session or storage.

RELEASE PROGRAM

The RELEASE PROGRAM statement can be useful and can help improve performance. If used inappropriately, however, its use can significantly increase I/O, resulting in performance degradation that can be dramatic. A standard that includes coding a RELEASE PROGRAM each time after a program is called is not a good standard. Although it controls the amount of temporary and above-the-line storage that is present at any one time, it significantly increases CPU and I/O costs.

RELEASE allows you to free a resource that is no longer needed in a CA Ideal program. Regardless of whether a program is in VLS format or load module format, the RELEASE PROGRAM statement frees the storage that is attributed to the updateable portion of a CA Ideal program.

It also marks the area in the Run Control Block (RCB) used for this program so that it is reused during the run. However, it does not reduce the size of the RCB. CA Ideal enlarges the RCB in increments of 4 KB as needed.

The following are good candidates for the RELEASE PROGRAM statement:

- Initialization programs.

  You should release any program that is called at the beginning of a run and never called again.

- The leg of an application called from a CA Ideal menu program.

  If you have a menu program that calls programs that are considered applications and if users spend a long period of time in one application before moving to another piece of the menu, you might consider using the RELEASE PROGRAM statement on the “application” program.

  This type of application can also lend itself to setting up each leg as its own transaction and using CA Ideal FINAL-ID to invoke the transaction. The invoked transaction can then invoke the menu transaction when it is complete. This frees up all resources because you ended the CA Ideal run.

Misusing RELEASE PROGRAM can increase I/Os, resulting in significant performance degradation.

For example, if you release a program in VLS format and then reuse the program, the updateable portion must be retrieved from the VLS library so that a copy can be made. The readonly portion of the program will also be read from VLS. Likewise, when a program in load module format is released and then reused, the updateable portion of the program must be retrieved from the load module library, unless it is in core because CICS has not yet released it or because it was made resident. In all situations, the existing copy of the data is deleted and then reallocated. The RELEASE statement has no effect on the CICS load module that contains the shareable portion of the code. CICS handles that storage as usual at the end of the transaction boundary.

Because the RELEASE PROGRAM statement also reinitializes the working data and parameter sections, some programmers use it for this purpose. This might not be a good practice. It can be more efficient to execute a few SET statements to reinitialize only those fields that need to be initialized.

Doing block SETS can be even more efficient: Set up a 01-level group in working data that contains all the working data fields that need to be initialized. Set the group equal to $SPACES or, for non-alpha groups or alpha groups that need to be initialized to something other than blanks, equate the group to another group containing the same setup using MOVE BY POSITION.

Timeouts and Disconnections

Coding a timeout parameter for VTAM, CICS, or a security package, can help you to enforce security. For example, it can reduce the chances of an unauthorized user taking over a session at an unattended terminal that is still signed on.

A disconnection can result because of a network problem or if the user simply closes the 3270 emulator window.

Important! Session storage is only deleted after a timeout or disconnection if the VPE Purge Invocation code has been invoked.

If the VPE Purge Invocation Code (VPUR) is not executed after a timeout or disconnection, you can expect the following to remain:

- User session Temporary Storage records allocated by CA IPC, Datadictionary, and CA Ideal.
- EDSA acquired for the updateable copy of the application program
- Use counts and enqueues performed on VLS entities.
- Storage allocated by Datadictionary for each terminal accessing DD.

If any of the preceding situations happen, it is because VPE has not been returned control, which allows it to clean up the session. Without invoking the VPE Purge Invocation code, this data can only be released if another VPE-based transaction uses the same terminal or CICS is recycled.

VPUR

If a CA Ideal session is aborted and VPE does not regain control and perform a session clean up, you can start a VPUR transaction that initiates a clean up process. Typically, this happens when a 327x terminal disappears and the CICS Node Error Program (DFHZNEP) is automatically enabled to handle the situation. To start VPUR, the VTAM codes are checked to ensure that the conditions are appropriate. VPUR invokes the VPE module VPEPURGE, and uses the input data from the CICS terminal ID that has been abnormally terminated. Using the TERMID, VPEPURGE finds the VPE Session Anchor Block (VSAB) and deletes all of the storage acquired by GETMAIN in various DSAs and Temporary Storage records that make up a CA Ideal user session. VPEPURGE will also dequeue any enqueues that have been issued for the session.

When VPUR is started, it is only known that the terminal where a CA Ideal task is executing has ceased to exist in the CICS environment. VPUR is invoked with the input data of the four-character CICS TERMID. Sometimes the task continues to execute the application, even though the TERMID has ceased to exist in CICS. If VPUR has been started, VPUR releases the storage that belonged to the task, the task is still executing and tries use its old session storage, which could by this time be reallocated by CICS to another task, resulting in abends or storage violations. Because of this potential problem, it is important to determine how VPUR should be implemented to be most effective based on a site's particular environment. For example, if long running tasks are the norm, it may be critical to first determine that the task has ended for a TERMID by querying CICS before invoking VPUR. VPUR could be invoked from the CICS NEP or the CICS terminal autoinstall exit.

Node Error Recovery

You can purge a session by invoking the VPE purge storage function from a Node Error Program (NEP). If available, CICS invokes a NEP when VTAM notifies it that a terminal was lost (LOSTERM notification). The NEP schedules a transaction that issues the purge storage function on behalf of the lost terminal.

To invoke the purge storage function from a NEP, first customize and then add the following statements to your site's CICS Node Error Program:

```
NEP0AF   DS    OH                                              @BD5021A
*-----------------BEGINNING OF SUGGESTED VPE PURGE INVOCATION CODE----*
        CLI   TWAEC,TCZxxxx       ERROR?
        BE    PURGEIT
        CLI   TWAEC,TCZxxxx       ERROR?
        BE    PURGEIT
        CLI   TWAEC,TCZxxxx       ERROR?
        B     GO_ON               NO; ELSE...
PURGEIT DS    OH
        EXEC  CICS START TRANSID('VPUR') LENGTH(4) FROM(TWANID)
GO-ON   DS    0H
*-----------------END OF SUGGESTED VPE PURGE INVOCATION CODE----------*
```

In the preceding illustration, '*xxxx*' represents the error code you intend to trap. Commonly trapped error codes include TCZTXCU (node unrecoverable), TCZNSP01 (network error 1) and TCZNSP02 (network error 2). The values that need to be trapped will depend on your network configuration. See the DFHZEQU macro for specific error code definitions. Also see the *CICS Customization Guide* for more information about Node Error Program processing.

PURGE term-id

You might need to purge storage for a CA Ideal session that is abruptly ended in CICS. For example, if a transaction is canceled, Virtual Processing Environment (VPE) can have storage areas and control blocks still allocated on behalf of the terminal. If the session is not purged, these storage areas could be allocated indefinitely since the random terminal-ID assignment of AUTOINSTALL cannot reuse the original terminal ID. The PURGE command cleans up these storage areas.

Format

```
PURGE termid
```

termid-The four-character terminal ID. You must type this value exactly as generated by the AUTOINSTALL routine since term-ID is case sensitive.

When the session is purged, the following message displays:

```
Terminal 'termid' purge complete
```

If the value supplied for *termid* is not a defined terminal ID, the following message displays:

```
Terminal 'termid' not found
```

Check to be sure you entered the terminal ID in the correct case, exactly as the AUTOINSTALL routine generated it.

VSE GETVIS Considerations

When you run applications in a VSE environment, having enough space is critical. CA Ideal under VSE is no exception.

Try some of the following suggestions if you encounter space issues:

- You might want to try decreasing the BUFSPACE parameter on your VSAM files. This degrades performance but it frees up GETVIS. It also requires that you redefine the file (you cannot change it dynamically).
- If you have several regions accessing CA Ideal, you might consider putting heavily used CA Ideal modules in the SVA. The module @IAETINT is a good candidate for moving to the SVA. It is frequently used and is approximately 100 KB in size.

  If you allow both on-line and batch compiles, the compiler modules are also a good choice. Moving those modules can save you GETVIS and free up DSA in CICS if the space in SVA is available.

  To determine which CA Ideal modules are used for compiles, compile a program using the CA Ideal Trace facility:

  ```
  @I$TRACE TRACE ON LOCAL
  COMPILE pgmname
  @I$TRACE PRINT FUNC VPE STATISTICS
  ```
- In batch, you might consider decreasing the size of your VPE file table (@IIDSYSF).

  One way to do this is to eliminate old and unused ROSFD entries for your VLS libraries. Another thing to look at is your ROSFD entries for sequential files. The blocksize parameter on those entries represents a maximum blocksize. The macro is expanded to include the space needed for that blocksize. For example, if you change the blocksize parameter on the ISLTAPE1 ROSFD entry from 4 KB to 10 KB, the size of @IIDSYSF increases by 6 KB. If you know that the largest blocksize you ever need is 4 KB, you have 6 KB of wasted space.
- Another thing to look at is your partition size. There are instances where your partition simply is not large enough.

  For batch, make sure that your SIZE= parameter is correct for the type of function you are trying to perform. If you are trying to compile a program and you are getting insufficient GETVIS messages, check if you are producing a cross-reference. If you really do not need one, SET COM XREF=NO. This frees up more GETVIS for the compile.

Chapter 12: Establishing Multiple Environments

The most common reasons for setting up multiple environments include:

- Change management
- Security
- Performance

Almost all users set up a production environment for use only by their end-users to run production applications. This separates development and production activity, which shelters and secures the production environment from a heavily used development environment. This separation also allows optimal performance for production applications.

This chapter tries to address some of the questions and concerns regarding the implementation of multiple environments. Topics covered include the following:

- DBMS regions
- Sharing libraries
- Enqueuing considerations
- Application migration
- Security and customization
- Installation

Composite Entities

The most important concept that must be considered when establishing multiple environments is that most CA Ideal entities are not simple library members, but composites of multiple members and Datadictionary entries that must be kept together.

- Program source is a composite of a Datadictionary entry and members for Procedure, Working data, and Parameter data.
- Program object is a composite of multiple members and includes the Datadictionary entry for non-Production programs.
- Panels are composites of a Datadictionary entry and Panel library members.
- Systems are composites of Datadictionary entries and VLS files.

- Members (for example, express signon) are composites of the Datadictionary entries defining the user and the VLS members themselves.
- Program or Panel Load Modules are composites of Datadictionary entries and the (multiple) executable members. If Application Module Tables are used, these can take the place of the dictionary entries.
- Regions that share these entities must therefore share the entire set of storage for them. You must consider a Datadictionary and the VLS files and load modules it describes as a unit for sharing.
- If, for example, a development environment and a production environment do *not* share the dictionary, they must not share the other files, and the Ideal entities must be transported between them using the appropriate utilities to ensure that all components are present and matched.

Enqueuing

CA Ideal will issue a single enqueue to cover all parts of a composite entity. Because entities transported between dictionaries have the same name in the sending and receiving environments, an additional component must be added to the entity name to provide a unique queue name in each. This one-byte value is obtained from the QCODE value specified as a parameter of the SC00OPTS module. It is important that all regions/partitions sharing a dictionary have the same value specified for QCODE. Conversely, regions/partitions accessing different dictionaries require different QCODE values.

Transporting Entities

There are three mechanisms used for transporting CA Ideal entities between environments.

- Source transport-Used to move programs, panels, reports, unmodeled dataviews and members. It creates (or updates) dictionary entries at the receiving end.
- Object transport-Used to move programs and their corresponding panels between environments. The programs must be in PROD status. It does not update the dictionary.
- Load modules may also be moved and identified to the dictionary in the receiving environment.

- Panels are both source and object, so they can be moved by both Source and Object transport; it is important to note the differences between the two methods. When a panel is used in a RUN, the date is checked against the value compiled into the program using it, and a discrepancy will end the run immediately. This is done to prevent buffer overruns which could result from the panel size changing.
- Source transport creates a new panel on the target environment, with a changed date of last modification. Object transport creates an exact copy of the panel, retaining the original date of last modification. This means that a panel transported as source cannot be used until the program is recompiled, so if both source and object transport are used, either the panels should only be included in the object transport, or the object should be transported after the source.
- **Note:** HELP panels may remain in TEST status for the convenience of future text updates. Use Source Transport instead of Object Transport to move TEST status HELP panels into production environments. The HELP panels left in TEST status must remain in VLS format even if the rest of the production application is in load module format.

DB2 Plans and Packages

The source of a Plan or Package for DB2 is kept on a VLS library-the default is IDDVW-but a separate file can be used by specifying a new name in IDOPTS. There is no dictionary entry for either entity.

Because Object Transport is most often used to move applications from development environments to production, it was decided that the plan and package members should also be included in the scope of this utility, even though they are considered source, in order for the transfer to use a single file. Plans and Packages must be rebound in the target DB2 subsystem, and the simplest mechanism is to regenerate from the plan/package source.

DB2 sites will therefore typically use a separate plan library for each DB2 subsystem.

CA Ideal and CA IPC Libraries

ADRPNL is used to hold the updatable SET SITE values for CA Ideal and CA IPC. A separate instance of this library is therefore needed for each environment where different settings are used. These include the PSS settings, so ADROUT and ADRPNL must be considered a pair for sharing purposes.

ADRLIB contains user jobcards; IDDAT contains all other user-related members. IDDVW contains all dataview information, so it too is specific to a dictionary. These three libraries should be kept with their related dictionaries for sharing.

CUSLIBs or VSE sub-libraries will also hold the assembled static options:

@IIDOPTS

The CA Ideal options. This includes such settings as whether to use load modules (typically NO for development regions and YES for production) which may differ between regions accessing the same dictionary.

@IIDSYSF

The batch file table. May differ between regions.

SC00TRAN, SCASTRAN, SCWBTRAN

Transaction tables for terminal, asynchronous, and Web applications. There is typically one for each CICS region.

SC00OPTS

The SCF options. Contains QCODE and other potentially region-specific settings. Multiple instances of this module will use the same QCODE value if the regions access the same dictionary.

@ILMLIST and @ILMTxxx

The list of application module tables and the tables themselves. These options are region-specific.

Chapter 13: Module Format for Programs and Panels

This chapter describes the functions associated with converting application programs and panels to load module format.

Module Definition

You should convert production status online application programs and panels from VLS format to a standard z/OS load module or a VSE phase format. In this section, the term module refers to CA Ideal programs and panels in this format. Performance is the primary reason to convert to module format.

Module Format

Module format provides significant performance benefits. In all environments, loading a module is more efficient than loading the corresponding VLS object member. Under CICS, the reentrant portion of a program or panel is loaded and released by each CA Ideal transaction that needs it. This allows CICS to optimize the use of CICS storage. CICS keeps modules specified as non-resident (CICS) with a use count of 0 in storage (DSA) only if CICS is not constrained for storage. Since CICS is in control of the whole region or partition (while CA Ideal is not), decisions about when to retain modules and when to swap them out can be made more effectively at the CICS level.

Application program load modules can be linked in 31-bit mode allowing the reentrant and non-updateable modules to reside in CICS extended storage. This decreases I/O to the libraries the modules reside in and the DSA it requires to load them.

Because CICS is managing the storage used for programs and panels, the CA Ideal RUN-STATUS of PRIVATE, SHARED, or RESIDENT is ignored for programs and panels in module format.

In addition, module format provides the following benefits:

- Module format facilitates bringing in a new version of a program or panel while CICS is up and running. The amount of time a load module application is not useable during the update process is very short compared to VLS format.
- Module format allows systems programmers to use standard monitoring and tuning tools to monitor program usage. This can result in better system tuning.
- Module format allows the use of standard IBM utilities for library maintenance, backup, and transport.

The application programs and panels cannot run outside the CA Ideal environment. If attempted, an S0C3 (z/OS) or Execute Exception (VSE) occurs. To execute a module format application, the user must still sign on to CA Ideal and execute a RUN command.

At transaction boundaries, a copy of the updateable portion of a program or panel is saved in extended storage. The original module remains in extended storage (RDSA)

The following sections describe the preceding issues in detail.

Creating Modules

Use the CREATE MODULE command to create the module form of a program or panel. This batch-only command is used the first time you convert a program or panel to module format. You also use it when you convert a new PROD version of the same program or panel to module format. As long as you use the same one- to seven-character module name, the CREATE MODULE command replaces the old module. However, if the module name for the program or panel changes, you must first delete the old module with the DELETE MODULE command.

The primary output of a CREATE MODULE command is a set of IBM-format object decks with appropriate Linkage Editor control statements, written to a sequential file (three object decks per CA Ideal program and one object deck per panel). This file is used as input to the IBM Linkage Editor. Regardless of the number of VLS object entities, there is always one object deck (module) created for each of the reentrant, non-reentrant (updateable), and symbol table portions of each CA Ideal program.

For example, a VLS object library listing can contain the following entries:

```
$IDDEMOPGM5     PRDJA
$IDDEMOPGM5     PRDJB
$IDDEMOPGM5     PRDTA
$IDDEMOPGM5     PRDTB
$IDDEMOPGM5     PRDTC
```

The VLS panel library listing can contain the following entry:

```
$IDDEMOPNL5     001U
```

- **$ID**- CA Ideal system name
- **DEMOPGM5**-CA Ideal program name
- **DEMOPNL5**-CA Ideal panel name
- **PRD**-Program status as production
- **JA,JB**-Two members for the program symbol table (used only when a runtime error occurs)
- **TA,TB,TC**-Three executable object members, two for the updateable portion and one for the non-updateable portion of the program
- **001**-Panel version as 1
- **U**-Panel VLS member

If you specify the following commands to convert the CA Ideal application to module format, the following modules are created as shown in the following table:

```
CREATE MODULE DEMPGM5 FROM PGM DEMOPGM5
CREATE MODULE DEMPNL5 FROM PNL DEMOPNL5 VERSION 1
```

VLS Object Members	Object Deck Member	Link edited Load Module Member
$IDDEMOPGM5 PRDJA $IDDEMOPGM5 PRDJB	DEMPGM5S	DEMPGM5S
$IDDEMOPGM5 PRDTA $IDDEMOPGM5 PRDTB	DEMPGM5U	DEMPGM5U
$IDDEMOPGM5 PRDTC	DEMPGM5R	DEMPGM5R
$IDDEMOPNL5 OO1U	DEMPNL5P	DEMPNL5P

Note how the multiple VLS object members for the program symbol table (PRDJA and PRDJB) were combined into one load module as were the multiple VLS members for the updateable portion of the program (PRDTA and PRDTB). The non-updateable portion of the program (PRDTC) is stored in a separate load module.

Three separate modules are created for each program:

- One for the reentrant portion of the program
- One for the updateable portion of the program
- One for the program symbol table

Only one module is created for each panel for the Panel Control Block (PCB), the reentrant portion of a panel.

The names of the created modules are the one- to seven-character modname with a one-character suffix. The suffixes are:

R-Reentrant (non-updateable) part of a program

U-Updateable (non-reentrant) part of a program

S-Symbol table of a program

P-Reentrant part of a panel

Use these suffixes in the following cases:

- The user wants to add CICS Program Definitions in order to specify RESIDENT=YES. CICS Transaction Server option for program autoinstall adds the CICS program definitions when the modules are loaded for the RUN of a CA Ideal application program. CTS PAIPGM adds the definitions with default values (RES=NO). For details, see the IBM documentation.
- The user wants to use a standard IBM utility to move, delete, and so on, the modules in the system libraries.
- The user wants to obtain performance monitor data about the use of modules in the system; for example, CICS shutdown statistics that show how many times a module is used.
- The user wants to use standard CICS services to disable, NEWCOPY, and enable the modules associated with programs and panels. There is also a CA Ideal REFRESH command to perform these same services that does not require the knowledge of the module naming conventions.

When dealing with non-Ideal services, the user must account for all three program modules and must specify the appropriate module suffix for each of these parts and for any panel modules.

You can specify many CREATE MODULE commands in the same CA Ideal batch run. The resultant object decks and appropriate Linkage Editor control statements are concatenated into the same sequential output data set. The user must specify a job step following the CA Ideal batch step to execute the Linkage Editor and read that sequential file as its input.

Execute a SELECT SYSTEM command before the CREATE MODULE command to identify the CA Ideal system that contains the program. Do not execute RUN commands in the same CA Ideal job step as CREATE MODULE commands or the results can be unpredictable.

If you need to modify the database ID associated with the program, use an ALTER PROGRAM DBID command before the CREATE MODULE, not after.

The secondary output of the CREATE MODULE command is a new or updated MODULE entity occurrence in Datadictionary. The entity name is a concatenation of the following:

```
'$I'  'PGM'  modname
'$I'  'PNL'  modname
```

The attributes maintained for the MODULE entity include the module name, version (always version 1), the date and time the module was created, entity information about the original program or panel, type, system, name, version, date and time of program compilation. In addition, the name of the user executing the CREATE command is placed in the author and controller attributes.

An alias is also created for the MODULE entity occurrence identifying the corresponding program or panel. It is a concatenation of the following:

```
'$I'  'P'  sss  pgm-name
'$I'  'M'  sss  pnl-name
```

- **sss**-System ID of the program or panel
- **P**-For program
- **M**-For panel
- *pgm-name*-Program name
- *pnl-name*-Panel name

If you are creating a module for a new version of a program or panel to which a module was previously created, then the previous MODULE entity occurrence and its related information is replaced in the dictionary. If a module is created for a program or panel and a program or panel with a different name or from a different system was already converted with the same module name, an error message is issued and the CREATE MODULE is aborted. Choose a new name or delete the old module.

The syntax of the CREATE MODULE command is described in the *Command Reference Guide*.

The program or panel must be in production status to be converted. If there are multiple versions, the one that is in production status is the one that is converted.

Production programs and panels that were transported with the CA Ideal Object Transport Utility can be converted to a module format even though there is no production program or panel dictionary facility entity occurrence for them. In this case, you must specify the panel's version number (you cannot specify PROD instead of the version number).

Sample JCL

For z/OS:

```
//IDLMODS JOB . . .
//*
//*           EXECUTE CREATE COMMAND IN IDEAL.               *
//*
//BATCH EXEC IDLBATCH
//OBJECT    DD DSN=&&TEMP,UNIT=SYSDA,DISP=(MOD,PASS,DELETE),
//          SPACE=(CYL,(2,2))
//SYSIN     DD *
PERSON $IDEAL PSW $IDEAL
SEL SYS $ID
CREATE MODULE DEMPGM5 FROM PGM DEMOPGM5
CREATE MODULE DEMPNL5 FROM PNL DEMOPNL5 VERSION 1
OFF
//*
//*      LINKEDIT MVS OBJECT DECKS PRODUCED BY CREATE.       *
//*
//LKED     EXEC PGM=IEWL,COND=(0,NE,BATCH),
//              PARM='RENT,XREF,MAP,LIST'
//*
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD UNIT=SYSDA,SPACE=(1024,(50,20))
//SYSLMOD   DD DSN=IDEAL.APPLIC.LOAD,DISP=SHR
//SYSLIN    DD DSN=&&TEMP,DISP=(OLD,DELETE)
/*
```

Note: DCB information is taken from the OBJECT entry in the CA Ideal System File Table (@IIDSYSF).

For VSE:

```
* $$ JOB JNM=IDLMODS,CLASS=0,DISP=D
* $$ LST DISP=D,CLASS=R,JSEP=0
// JOB IDLMODS
// OPTION LOG,NODUMP
// DLBL IDLPROC,'Ideal.Proclib'
// EXTENT ,vse004
// LIBDEF *,SEARCH=(IDLPROC.Sublib)
// ASSGN SYS000,DISK,VOL=vse004,SHR
// DLBL OBJECT,'Ideal.test.object',0
// EXTENT SYS000,vse004,,,76636,200
// EXEC PROC=IDLBATCH
// EXEC IDBATCH,SIZE=128K
PERSON $IDEAL PSW $IDEAL
SEL SYS $ID
CREATE MODULE DEMPGM5 FROM PGM DEMOPGM5
CREATE MODULE DEMPNL5 FROM PNL DEMOPNL5 VERSION 1
OFF
/*
// OPTION CATAL
// DLBL IJSYSIN,'Ideal.test.object'
// EXTENT SYSIPT,vse004
   ASSGN SYSIPT,DISK,VOL=vse004,SHR
// DLBL 'Ideal.sp.library'
// EXTENT ,vse004
// LIBDEF *,SEARCH=(Ideal.sublib), catalog=Ideal.sublib
   INCLUDE
// EXEC LNKEDT
/*
   CLOSE SYSIPT,05c
/&
* $$ EOJ
```

RMODE Parameter

Specification of RMODE in the parameter list of the z/OS LKED EXEC or VSE LINKEDT statements is not required. CA Ideal handles the residency mode automatically.

CA Ideal *always* automatically links panels with the specification RMODE 24. If RMODE ANY is hardcoded in the JCL, panels are placed above the line where CA Ideal cannot access them, therefore leading to abnormal terminations.

Programs are automatically linked with a specification RMODE ANY.

Deleting Modules

The DELETE MODULE command (on-line or batch) deletes the MODULE entity occurrence and alias occurrence that correspond to the specified program or panel from the dictionary only. The DELETE MODULE command does not delete the generated modules from any system module libraries. That is the user's responsibility. Be sure to delete all three generated parts of a CA Ideal program module.

Execute a SELECT SYSTEM command before the DELETE MODULE command to identify the CA Ideal system that contains the program. The syntax of the command is described in the *Command Reference Guide*.

Identifying Modules

The IDENTIFY MODULE command (batch) updates the MODULE entity occurrence and ALIAS occurrence of the specified program or panel in the dictionary. The IDENTIFY command is used after the module format of the program or panel is moved to the receiving environment.

Execute the IDENTIFY command in batch and only after a program or panel module was moved through an IBM utility. It makes the module known to CA Ideal at the receiving site. One IDENTIFY command is sufficient for all load modules associated with the specified program or panel.

The program or panel name specifies the entity to which the module corresponds. A SELECT SYSTEM command executed before the IDENTIFY command indicates the CA Ideal system that contains the entity.

The IDENTIFY MODULE command performs the following steps in batch:

1. It attempts to load the module by executing a load service call to the appropriate z/OS or VSE facility. Consequently, the module must be in the appropriate z/OS joblib or steplib or VSE core image library.
2. It verifies that the loaded module corresponds to the specified program or panel.
3. It creates or updates the MODULE dictionary entity occurrence corresponding to the module name specified on the IDENTIFY command.

The syntax of the IDENTIFY command is described in the *Command Reference Guide*.

DISPLAY/PRINT INDEX MODULE Command

The DISPLAY/PRINT INDEX MODULE command displays a list of programs and panels that have module entity occurrences in the dictionary. You can also specify an optional WITH VERIFICATION clause to verify that the modules are really in the module library and represent the appropriate application program. The syntax for the command is described in the *Command Reference Guide*.

The module name, the corresponding application program or panel identification, and the date/time of program compilation are displayed. The entries are in alphabetical order by module name with all program modules first, followed by panel modules.

The WITH VERIFICATION option attempts to load each module or set of module, to determine whether the correct modules are available to the environment. The entity type, system ID, entity name, version number, and date/time stamp in the MODULE dictionary entity are compared with the information found in the loaded module. Any discrepancies are noted.

This command does not display any modules that do not have corresponding dictionary entries in the module table.

Module Runtime Considerations

This section details the considerations of module for the runtime environment.

Building the In-Core Load Module Table

The in-core load module table (LMT) identifies the application programs and panels that are in load module format and provides their module names for the runtime environment. Whenever an application RUN loads a production program or panel for the first time or after it is reaccessed after a RELEASE statement, CA Ideal determines whether the entity is in load module format. If an entry is found, standard operating system service calls are issued, depending on the environment. If an entry is not found, a load of the program or panel is requested from the appropriate VLS library.

The contents of the LMT can be built using two sources of input:

- By scanning the MODULE dictionary entities for entries that correspond to CA Ideal application programs and panels. For each program or panel MODULE DD entity, an entry is added to the in-core LMT that identifies the program or panel name and its corresponding module name.
- By using site-maintained tables to control the contents of the LMT. This option provides two structures with CA Ideal are analogous to the CICS RDO LIST/GROUP structures.
- Application Module Table (AMT): Application Module Tables (AMTs) are physical modules/phases containing entries of programs and panels and their corresponding load module prefix. Each program and panel defined in an AMT adds an entry to the in-core LMT without requiring the presence of a module entity occurrence in the dictionary. There can be multiple AMTs in a single region. AMTs are assembled and linked with macros that CA Ideal provides. They have a naming convention of @ILMTxxx, where xxx is a name the user provides.
- @ILMLIST: @ILMLIST is a module that contains the list of AMTs to process when building the in-core LMT. @ILMLIST is assembled and linked with macros CA Ideal provides. Each region (batch or on-line) can have a separate copy of @ILMLIST, thereby allowing each region to tailor the load modules it runs.

The section following describes these two structures in more detail.

For batch, the in-core LMT is built when the first RUN command is issued. Under CICS environments, the in-core LMT is built when CICS starts, provided the program @IADPLTI was included in the PLT program startup list. If @IADPLTI is not in the PLT or in the unlikely event that the task is not successful, the user issuing the first RUN command invokes the processing to build the in-core LMT.

You can control which sources build the LMT by setting the LMTBLD parameter of the IDOPTSCB macro in the IDOPTS source (which produces the @IIDOPTS load module that controls CA Ideal site options). Enter the LMTBLD parameter on the FUNC=START statement in the IDOPTSCB macro. You can assign the following values to the LMTBLD parameter:

- **DD**-Builds the in-core LMT using entries from the @ILMLIST module and entries found in the dictionary module table.
- **NO**-Builds the in-core LMT from processing @ILMLIST only.

The building of the in-core LMT in all environments begins with the loading of @ILMLIST. If @ILMLIST contains entries, the in-core LMT is built based on the contents of each AMT table specified in @ILMLIST. If @ILMLIST is empty, the in-core LMT is empty at this point.

Under CICS, if LMTBLD=DD has been specified, the in-core table will be updated to include any module entries in the dictionary that do not exist in the in-core LMT.

In batch where LMTBLD=DD, each time a program or panel is accessed and is not found in the in-core LMT, the dictionary is accessed to determine if a module occurrence is present. If a module occurrence is found, it is added to the in-core LMT.

The in-core LMT is always built, but can be empty. An empty in-core LMT is built when @ILMLIST is empty and either LMTBLD=NO or there are no module entity occurrences in the dictionary. When both of these conditions are true, CA Ideal uses only VLS members in the environment.

Displaying the In-Core Load Module Table

With the introduction of AMT processing and the elimination of the requirement for creating load module entries in the dictionary, the DISPLAY INDEX MODULE command does not necessarily reflect an accurate picture of the in-core LMT. The following command produces a display of the current contents of the in-core LMT in the region the command is issued.

```
DISPLAY LMT
```

It does not produce an alphabetical listing. You can use FIND and INCLUDE commands to locate specific entries. For additional information regarding this command, see the *Command Reference Guide*.

Tailoring the LMT

The introduction of site-defined AMTs and @ILMLIST let you designate the exact contents of the in-core LMT. This can be useful in a variety of situations:

- Multiple CICS regions sharing a single dictionary and each of those CICS regions run different programs
- Creating separate AMTs for batch only applications
- Simplification of processing

AMT processing removes the PPT and LMT entries for load modules that are not run in a specified CICS region.

AMT processing can also simplify the processing that must otherwise execute at the receiving site of migrated applications. You can do this by simply supplying the load modules and AMTs to the receiving site. In some cases, you can also supply @ILMLIST. In other cases, the receiving site executes the steps necessary to update, reassemble, and link @ILMLIST.

Sites using AMT processing should be aware of the following restrictions. For more information see the “Application Migration Considerations” chapter.

- REFRESH command deletes the LMT entry if no dictionary module entry is present. Execute the CICS NEWCOPY command to refresh an existing program.
- Modifications to @ILMLIST and AMTs require recycling. CICS NEWCOPY commands do not affect the contents of the LMT once CICS is active.
- New subprograms require you add additional entries to the corresponding AMT. You must either recycle CICS or issue the IDENTIFY MODULE command before a REFRESH command so that the LMT can be updated. CICS NEWCOPY commands do not make the module known to the CA Ideal LMT.

Preparing Application Module Tables (AMTs)

The following is the format of an AMT. Specify each program or panel module entry on a single line.

```
AMT             TITLE 'title'
                LMHDR TABLEID=xxx
modname         LMDEF ent,sys,entname,ver
modname         LMDEF ent,sys,entname,ver
                        .
                        .
                        .
                TITLE 'title'
modname         LMDEF ent,sys,entname,ver
modname         LMDEF ent,sys,entname,ver
                        .
                        .
                        .
                END
```

- **TABLEID**-Three-character literal that distinguishes the application LMT and builds a header in the application LMT.
- **modname**-One- to seven-character module prefix identifier.
- **ent**-Literal PGM for programs and PNL for panels.
- **sys**-Three-character system name of the program or panel.
- **entname**-One- to eight-character program or panel name.
- **ver**-Literal PRD for programs and the explicit version number for panels.

The following is an example of entries and JCL that assemble and link an AMT.

z/OS

```
//IDLMTXXX JOB ...
//ASMOPT EXEC PGM=IEV90,PARM=(DECK,NOOBJ),REGION=500K
//SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=Ideal.maclib,DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSPRINT DD SYSOUT=*
//SYSPUNCH DD DSN=Ideal.objlib(LMTXXX),DISP=SHR
//SYSIN    DD *
****  SAMPLE INPUT  ****
AMT      TITLE 'Sample Application LMT:  Application 1'
         LMHDR TABLEID=999
MODNAM1  LMDEF PGM,$ID,PROGRAM1,PRD
MODNAM2  LMDEF PGM,$ID,PROGRAM2,PRD
MODNAM3  LMDEF PGM,$ID,PROGRAM3,PRD
MODNAM4  LMDEF PGM,$ID,PROGRAM4,PRD
MODNAM5  LMDEF PNL,$ID,PANEL1,001
         END
//LNKOPT EXEC PGM=IEWL,PARM=(RENT,XREF,LIST,NCAL),COND=(4,LT)
//SYSLMOD  DD DSN=loadmod.lib,DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSPRINT DD SYSOUT=*
//OBJLIB   DD DSN=Ideal.objlib,DISP=SHR
//SYSLIN   DD *
  INCLUDE  OBJLIB(LMTxxx)
  NAME     @ILMT999(R)
/*
```

VSE

```
* $$ JOB JNM=@ILMT$ID,CLASS=A,LDEST=(*,VMUID)
* $$ LST CLASS=R
// JOB    @ILMT$ID
// ON     $RC>0 GOTO $EOJ
// LIBDEF *,SEARCH=(CAI.IDEAL,CAI.IPC),CATALOG=CAI.USER
// OPTION CATAL
   PHASE  @ILMTxxx
// EXEC   ASSEMBLY
         PRINT NOGEN
LMT      TITLE 'LOAD MODULE DEFINITIONS FOR $ID SYSTEM'
         LMHDR TABLEID=$ID
MODNAM1  LMDEF PGM,$ID,PROGRAM1,PRD
MODNAM2  LMDEF PGM,$ID,PROGRAM2,PRD
MODNAM3  LMDEF PGM,$ID,PROGRAM3,PRD
MODNAM4  LMDEF PGM,$ID,PROGRAM4,PRD
MODPNL1  LMDEF PNL,$ID,PANEL1,001
         END
/*
// EXEC   LNKEDT
/*
/&
* $$ EOJ
```

Note: In both z/OS and VSE, a CICS PPT entry is required for the @ILMT*xxx* module or phase, where *xxx* is $ID by default. You can change it as long as the PPT name corresponds to (is the same as) the module or phase name.

Establishing @ILMLIST AMT Lists

The following is the input for the assembly of @ILMLIST. Specify on the LMTGEN macro TABLEID parameter the list of AMT tableids that reflect the contents for @ILMLIST.

```
LMLIST                TITLE '@ILMLIST Application LMTs'
                      LMTGEN TABLEID=(tid,tid,...)
                      END
Tid
```

Represents the list of AMT tableids. All @ILMTsss modules and the @ILMLIST module must be in a library accessible to the CICS or batch region.

To produce an empty @ILMLIST, specify the following for the assembly input:

```
LMLIST                TITLE '@ILMLIST Application LMTs'
                      LMTGEN TYPE=DUMMY
                      END
```

The following is an example of entries and JCL that assemble and link @ILMLIST.

z/OS

```
//IDLMLIST JOB ...
//ASMOPT EXEC PGM=IEV90,PARM=(DECK,NOOBJ),REGION=500K
//SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=Ideal.maclib,DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSPRINT DD SYSOUT=*
//SYSPUNCH DD DSN=Ideal.objlib(LMLIST),DISP=SHR
//SYSIN    DD *
LMLIST   TITLE '@ILMLIST Application LMTs'
         LMTGEN TABLEID=(000,999,DRA)
         END
//LNKOPT EXEC PGM=IEWL,PARM=(RENT,XREF,LIST,NCAL), COND=(4,LT)
//SYSLMOD  DD DSN=Ideal.cailib,DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSPRINT DD SYSOUT=*
//OBJLIB   DD DSN=Ideal.objlib,DISP=SHR
//SYSLIN   DD *
  INCLUDE  OBJLIB(LMLIST)
  NAME     @ILMLIST(R)
//
```

VSE

```
* $$ JOB JNM=@ILMLIST,CLASS=A,LDEST=(*,VMUID)
* $$ LST CLASS=R
// JOB    @ILMLIST
// ON     $RC>0 GOTO $EOJ
// LIBDEF *,SEARCH=(CAI.IDEAL,CAI.IPC),CATALOG=CAI.USER
// OPTION CATAL
   PHASE  @ILMLIST
// EXEC   ASSEMBLY,PARM='VSE'
         TITLE '@ILMLIST APPLICATION LMTS'
         PRINT NOGEN
         LMTGEN TABLEID=($ID,DRA)
         END
/*
// EXEC   LNKEDT
/*
/&
* $$ EOJ
```

Note: A CICS PPT entry is required for the @ILMLIST phase.

Automating AMT Generation

You can generate an AMT manually or automatically. AMTGEN is a sample CA Ideal program available from the CA Support website. This program generates an AMT that includes an entry for each production program and panel in a system that you specify. You can modify this program to meet other specifications you might want to adopt at your site or you can use an entirely different program that generates this data.

The program AMTGEN is provided in external source format. You must use the CA Ideal source transport utility to import this application.

You can automate the generation of an AMT. CA Ideal distributed a sample application to generate AMTs for all the programs and panels in an entire CA Ideal system definition. CA Ideal member AMTGEN contains JCL to run the application.

Execute the CA Ideal program, AMTGEN, passing the name of the selected system as a parameter to the AMTGEN program, as shown in the following RUN command:

```
RUN AMTGEN  VERSION PROD PARM '@IC'
```

The AMTGEN program produces the following output:

- A list of all program and panel module entries in the dictionary that are in the specified system.

```
09/29/94           AMTGEN Audit Listing         PAGE 1
       AMT and PPT Source Generator for System: HBS
SEQUENCE                OUTPUT                  MODULE
  NUMBER                GENERATED               NOT FOUND
________                _________               _________
       1                ADDEMPL
       2                                        ADDHMOS
       3                                        CHGCARR
       4                RATTGEN
       5                RATTUPD
       6                UPDEMPL
                           .
                           .
                           .
      23                ADDEMPL
    Total Entities Encountered:             23
    Entities Processed                      15
    Module-Not-Found Entities                8
```

- You can use card-image files to assemble and link the list of module entries into an application module table.

```
AMT      TITLE 'LOAD MODULE DEFINITIONS FOR AMT: PROGRAMS'       00000001
         LMHDR TABLEID=@IC                                       00000002
RATTGEN  LMDEF PGM,@IC,RATTGEN,PRD                               00000003
RATTUPD  LMDEF PGM,@IC,RATTUPD,PRD                               00000004
UPDEMPL  LMDEF PGM,@IC,UPDEMPL,PRD                               00000005
ADDEMPL  LMDEF PGM,@IC,ADDEMPL,PRD                               00000006
                     .
                     .
                     .
         TITLE 'LOAD MODULE DEFINITIONS FOR AMT: PANELS'         00000012
ADDEMPL  LMDEF PNL,HBO,ADDEMPL,0001                              00000013
                     .
                     .
                     .
         END                                                     00000019
 NAME @ILMT@IC(R)                                                00000001
```

To assemble the load module table @ILMLIST, perform the following tasks:

- Assemble the card-image file of module entries produced in Step 1 to create the AMT (application module table). The application module table is named @ILMT*sss*, where *sss* is the system ID of the program specified to the AMTGEN program. To specify a different suffix, set the parameter to the appropriate three-character name on the control card for the AMTGEN program.
- Repeat steps 1 and 2 for each application in the CICS region.
- Edit the LMTGEN statement in the jobstream in source member LMLSTASM to include the three-character suffixes of the application load module tables and then submit the member LMTASM to assemble the load module table @ILMLIST.

Notes:

- All @ILMTsss modules and the @ILMLIST module must be in a library accessible to the CICS or batch region.
- If your site has CA Datacom Database Resource Analyzer Option, the Application Module Table @ILMTDRA must be an entry in @ILMLIST.
- For ease of maintenance, each application LMT should match one RDO group for PPT entries which lets you isolate changes and group them easily.

CICS Considerations

This section details the CICS considerations as follows.

PPT Entries

The CICS system initialization parameter PGAIPGM must specify active. This lets the program autoinstall function so that CICS dynamically adds PPT entries for the CA Ideal application load MODULEs. For more information, see IBM documentation.

Loading the Modules at Startup

You can control whether all the load modules are loaded when the in-core LMT is built at PLT startup by setting the PLTLOAD parameter of the IDOPTSCB macro in the IDOPTS source.

You can assign the following values to the PLTLOAD parameter.

- **YES**-Determines whether the load modules associated with an entry added to the in-core LMT during CICS PLT processing are loaded into CICS. An error message will be issued for any missing or invalid modules.
- **NO**-Bypasses the loading of the modules when the entry is added to the in-core LMT during CICS PLT processing.

In most instances, you want to set the PLTLOAD parameter to NO. Because it bypasses the loading of the user program and panel load modules, it significantly reduces the amount of time CICS becomes available to your users after startup. A value of NO does not affect how the in-core LMT is built or the PPT entries.

If PLTLOAD=YES, all program modules, including the symbol table of every program that is only loaded in the case of a runtime error, remains resident in RDSA . This could amount to a lot of storage never accessed by the environment.

Updating the In-Core LMT-REFRESHing Modules Online

Changes to MODULE entity occurrences in the dictionary after the in-core LMT is built do not affect the in-core LMT. CREATE MODULE, DELETE MODULE, and IDENTIFY MODULE commands consequently do NOT modify the in-core LMT.

The REFRESH command updates the in-core LMT and synchronizes it with the dictionary entity occurrence as follows:

- When a module entity for the specified program or panel is found in the dictionary:

  If an in-core LMT entry exists, it is updated; otherwise, an entry is added. In a CICS environment, if PPT entries do not exist, they are added and a CICS DISABLE, NEWCOPY, and ENABLE for each module associated with the program or panel, is performed.

- When a module entity for the specified program or panel is not found in the dictionary:

  If an in-core LMT entry exists, it is deleted; otherwise, an error message is issued.

A SELECT SYSTEM command executed before the REFRESH command identifies the CA Ideal system that contains the program or panel. The syntax of the REFRESH command is described in the *Command Reference Guide*.

Execute the REFRESH command for each program or panel that was replaced. You should also execute it in each environment using the replaced program or panel.

Carry out this process with standard CICS commands. However, using the CA Ideal REFRESH command with the program name means that all three modules are refreshed for application programs; you do not need to remember the proper module suffixes, and you do not need to remember the seven-character module name.

If a CICS NEWCOPY service fails because the module is still in use, CA Ideal waits five seconds and tries again. If it still fails, CA Ideal continues waiting and tries again in one minute. If, after a minute, the NEWCOPY still fails, the load modules are left disabled and a message indicating the problem is sent to the user. This usually implies that the CA Ideal DISABLE or ENABLE command was not correctly issued or that the ENABLE command was issued before the REFRESH command. For more information about the REFRESH command, see the "Application Migration Considerations" chapter.

Chapter 14: Application Migration Considerations

This chapter describes the methods available for moving application programs and panels from the development center to the production center. Each method has certain advantages and disadvantages. For each site, determine the best method that suits its specific needs.

Although it is not required, you should convert all programs and panels to load module format at the production center for optimal performance.

Development Considerations

When you are ready to release an application to the end users:

1. All programs and their resources should be in production status.
2. Move the programs from the development center to the production center.
3. Add or update the application at the production center.

Testing Programs Before Marking Them to Prod

After the application is initially marked to prod, you can modify one or more programs in the application.

You can modify the application by first duplicating the programs that require modification to the next version in the development environment and making the necessary changes. When the TEST status programs are ready to test, you can run them with the unchanged PROD status programs by issuing the command:

```
ASSIGN PROGRAM pgm-name VER test-version
```

You should then issue the RUN statement for the current production application. (For the command syntax, see the *Command Reference Guide*.)

After the program is tested with the new TEST versions of the modified programs and the results are correct, you can MARK these programs to Prod status. The current Prod status calling programs are automatically related to the new production versions. No change or recompile is necessary for the calling programs.

When issuing the MARK STATUS command, you can receive the error message IDADXSSP24 “PROD” program in use, please try later. This can happen if the SCOOPTS QCODE parameter is not properly set.

Mark Programs to Prod Status

The CA Ideal MARK STATUS command changes the status of an entity. For information about entity status and version, see Preliminary Concepts in the *Programming Reference Guide*. The MARK STATUS command is documented in the *Command Reference Guide*.

Putting an application in Prod status cuts down processing at runtime. When you run a program that is in Test status, CA Ideal checks the dictionary and verifies that:

- The program entity occurrence exists.
- The program's procedure, working data, and parameter sections were not edited since it was last successfully compiled.
- The resources of the program were not modified since the last successful compile.

When you run a program that is in Prod status, the dictionary checking is minimal because:

- Moving the program to Prod status ensures that all of the program's resources must exist and are in Prod status.
- The edit dates must be in sync because you cannot edit the program.

When VER PROD is specified on the RUN command, CA Ideal goes directly to the object code and does not check the dictionary at all. PROD status is a requirement for the Object Transport Utility and the CREATE MODULE facility.

Application Migration Considerations

The first decision that you need to make for the production environment is the format under which the applications run. Load modules are highly recommended in a CICS production environment over VLS format.

Runtime Configurations

When load modules are the format of choice, you must also decide what sources to use to build the in-core LMT. You can find more information about the in-core LMT in the “Module Format for Programs and Panels” chapter.

There are three choices for building the in-core LMT:

- You can build the contents of the in-core LMT by extracting the data found only in the dictionary MODULE table. To implement this method, all modules must be created or identified. Specify IDOPTS LMTBLD=DD and the @ILMLIST must be empty.

  Advantages:

  DIS INDEX [ALL] MODULE is an accurate list of the programs and panels running in load module format.

  Disadvantages:

  If running multiple CICS regions against the same dictionary, LMT entries are built for programs and panels in all regions regardless of whether they run there.

- You can build the contents of the LMT by extracting the data only from the AMTs listed in the @ILMLIST. To implement this method, specify IDOPTS LMTBLD=NO. Dictionary entries might exist, but they are ignored when the LMT is initially built.

  Advantages:

  - You can customize the contents of the LMT to include only the programs and panels accessed in a particular region. This can be especially useful when multiple CICS regions run different sets of programs but share the same dictionary.
  - This method results in the quickest processing time to build the LMT because there is only access to the AMT modules, therefore access to the MODULE dictionary table was eliminated.
  - Minimal work is executed at each production center. You can create the modules in the development region along with an AMT and move it to the production center. This can be a very useful method of distributing programs in a vendor situation.
  - IDENTIFYs are optional.

  Disadvantages:

  - DIS INDEX MODULE is not an accurate list of the programs and panels running in load module format because it only references dictionary and not the LMT.
  - You cannot execute the REFRESH command if no dictionary module occurrence exists. The LMT entry is deleted.

- The contents of the LMT are built from both the programs found in the AMTs and the dictionary. Duplicates are allowed. However, they incur additional overhead when the in-core LMT is built. To implement this method, IDOPTS LMTBLD=DD and the @ILMLIST should not be empty.

  Advantages:

  - You can define applications common to all CICS regions in the dictionary.
  - You can define applications executed in one CICS region using an AMT.

  Disadvantages:

  - DIS INDEX MODULE is not an accurate list of the programs and panels running in load module format because it only references dictionary and not the LMT.
  - You cannot execute the REFRESH command if a dictionary module occurrence does not exist. The LMT entry is deleted.
  - Using the two methods together can be a source of confusion.

Application Migration Processes

You can use a variety of methods to move the application programs and panels from the development center to the production center.

Using the Object Transport Utility with Load Modules

One possible method of migrating an application is to first use the CA Ideal Object Transport Utility to move the program and panel object modules in VLS format to the production center. After this is done, use the CREATE MODULE command in the production environment to convert the VLS object members to module format. There are several advantages to using the Object Transport Utility approach:

- You can easily unload and load related collections of programs and panels with just a few commands.
- The Object Transport Utility output identifies all the programs and panels that were loaded. This provides a list for determining which programs and panels are converted to module format.
- If any ALTER PROGRAM commands are required at a later time, the transported VLS object modules are available to ALTER. Then new load modules created from them without having to retransport. ALTER PROGRAM has no effect on module format applications.
- You are not required to be aware of module suffixes for the three modules per program.
- If a subset of all programs and panels is converted to module format, the Object Transport Utility is still needed to move those that still execute in VLS format. It is, therefore, advisable to move the whole application and then convert those programs and panels as needed to module format.

Using the CA Ideal Source Transport Utility with Load Modules

Sites can use the source transport utility to move an application. Some sites like to have the source available in case of an emergency change and the sending and target sites are not connected. It might also be possible that the target site needs to make modifications to the source.

Once the application is source transported, compile and mark the entities to production before load modules are created.

EXPGEN Sample Application

CA Ideal distributes a sample application that is called EXPGEN that generates all the entity EXPORT statements in the application substructure of the specified high-level program. Use this application as delivered or customize it according to individual requirements. Sample code can be downloaded from the CA Support website.

Source Transport Commands

There are several commands of the CA Ideal Source Transport Utility that you should carefully consider when using the source transport for production application migration. You can find these commands in *Working in the Environment Guide*. The most helpful of these commands for application migration include:

- **SET IMPORT RESOURCE**

  Overrides the versions specified for the resources of a program at import time without having to modify the external source.

- **SET IMPORT DUPLICATE**

  Specifies the action to take when an imported entity already exists.

- **SET IMPORT NEW VERSION**

  Specifies the version for the entities created during the import process.

Important! If you are using the source transport utility to migrate applications to a production environment, do not use it with the object transport utility. Panels can cause problems.

CA Ideal application panels are stored in two parts: the layout is stored in a source member in the panel library and the compiled attributes are stored in each program's object.

Both parts are used at runtime. The object transport utility moves both parts because they are both needed at runtime. The object transport copies the source and object members “as is” into the receiving region. The date/time stamps showing the last modification to the panel are not changed.

The source transport, however, moves just the source member. It changes the date/time stamp of the transported member to the date/time of the actual transport because the source transport is in effect creating a new entity at the receiving site.

At runtime, CA Ideal compares the date/time of the last edit of the panel (as it was recorded in the program object at compile time) to the edit date found in the panel source member to make sure that the panel was not modified since the last compile. In this scenario, the panel date/time stamp is later than the compile date/time because the source transport overlaid the panel source and changed the date. The runtime processor assumes that this panel was modified and issues the PANEL MODIFIED message.

The best thing to do if you want to shelter the source code is to export it using source transport and save it in external source format using CA Librarian (or another library management system) or a data set.

If you want to maintain both the source and object code for panels in a production region, do one of the following:

- Maintain separate system IDs for source and object so that there is no chance of overlaying the panels.
- Use the source transport only and recompile everything in the production region.

Using Utilities to Move Load Modules

The last option requires the least amount of processing in the production center. You can move whole libraries containing CA Ideal user program and panel load modules to the production center. CA Ideal IDENTIFY commands are optional if an application module table (AMT) is available to define the entities to the LMT. For additional information regarding this process, see the “Module Format for Programs and Panels” chapter.

Replacing Online Applications

The basic requirement of any online line application system is the ability to install new application software or replace existing applications with updated software while the online line environment is running. These steps differ slightly based on whether you are running VLS or load module format.

Using load module/phase format:

1. Migrate the application using one of several methods resulting in load modules available to the production environment.
2. Use the DISABLE RUN command to force active users off the affected application and suppress potential users from gaining access while you are installing the application.
3. REFRESH the load modules.
4. Use the ENABLE RUN command to allow users to start running their applications again.

With load modules, you can execute Step 1 only and wait until CICS is recycled to bring in the new versions of the application programs.

Using VLS format:

1. Use the DISABLE RUN command to force active users off the affected application and suppress potential users from gaining access while you are installing the application.
2. Transport the application using the object transport utility.
3. Use the ENABLE RUN command to allow users to start running their applications again.

If there is a mixture of load modules and VLS format for a single application, it is still necessary to execute the object transport utility for VLS run programs between the DISABLE and ENABLE of the application and panels in VLS format.

Disabling and Enabling an Active Online Application

You can use the DISABLE and ENABLE commands to force active users to exit from an active application to import a new version of that application by using the transport utility or by creating a new version of a load module. You can use the DISABLE command against applications that are in VLS object format or load module format.

The syntax of these commands is described in the *Command Reference Guide*.

The DISABLE command forces users to exit from an application. The DISABLE command must specify the highest level of an application (that program executed by a RUN command). It has no effect if issued for a subprogram. Precede the DISABLE command with the appropriate SELECT SYSTEM command.

Important Considerations

The following considerations while disabling an active online application are:

- If the application to disable is running in multiple CICS regions or partitions, execute the DISABLE command in each region. This is true for each standalone CICS region and each AOR region in an MRO environment.
- When the DISABLE time arrives, the users running the specified application are not immediately purged. Their runs are aborted at the next transaction boundary (when the user presses a PF key or the Enter key or when a panel is transmitted) after the disable time arrives. For programs and panels in VLS format, the enqueues on those programs and panels are not released until the run is actually aborted. If a user left the terminal before the disable time arrives, those enqueues are not released until the user returns and presses the Enter key or a PF key.

  These enqueues prevent the transport utility from running or a new version of a program from being marked to PROD status since an exclusive enqueue on those programs and panels is requested from the operating system. In this case, eliminating these enqueues requires a DEQUEUE command be issued for each program and panel enqueued before running the object transport utility or issuing the MARK STATUS command. This is not a problem for application programs and panels in load module format since the runtime executor uses no enqueues in this case.
- It is not possible to disable a particular subprogram or portion of a run-unit. Specifying a subprogram name instead of the main program name is ineffective. The program name specified in the DISABLE command must be the same program name specified in the RUN command.
- If a new version of a program is replaced and multiple applications use it, each of those applications must be disabled.
- If a new version of a program is replaced without issuing a DISABLE command or in some way making sure that all users are off the application, different kinds of internal errors can occur. The most common error, “IDADOMLD07 - Global storage DT discrepancy for PGM pgm-name VER ver found” indicates that the updateable and reentrant portion of a program is not synchronized. If the updateable portion of a program is in CICS temporary storage while the reentrant portion of the program is replaced, this synchronization error is detected the next time the reentrant part of the program is loaded into memory.
- The DISABLE command stays in effect until it is explicitly released with the ENABLE command or until CICS is shut down and restarted.

The ENABLE RUN command allows users to start running applications that were previously disabled.

If DISABLE and ENABLE commands are executed after an end user left an active run of that application, the run still aborts when the user returns to continue the run. This is true even if the application was enabled before the session was continued. This is because the user was running a program that is replaced and the updateable portion of the program is from the old version of the program while the reentrant portion of the program is from the NEWCOPY copy. To synchronize all program components, the run must still be aborted and the user must restart the application.

Replacing Programs and Panels in an Active Online Environment

You can choose among five basic configurations options to process a single environment.

Option 1: Load Modules - LMTBLD=DD with empty @ILMLIST

This represents the traditional method used in past releases of CA Ideal before AMTs became available with CA Ideal 2.2. Using this option requires that the modules be defined to the dictionary. You can do this by CREATing the modules in the production center from the VLS object or moving the modules and issuing an IDENTIFY command for each program or panel.

The syntax of the CREATE and IDENTIFY command are described in the *Command Reference Guide*. You must execute the commands in batch.

Since the CREATE and IDENTIFY commands do not update any in-core module table, the new or updated modules are not known to any online line CA Ideal environments in operation when the commands are executed. After the load modules and dictionary entries are defined to the region, you can disable the application and refresh each of the new or replaced programs and panels before they are disabled. Waiting to recycle the region is also an optional means of bringing in the new application.

Option 2: Load Modules - LMTBLD=NO, without dictionary entries

This setup means the LMT is built only from the entries pointed to by @ILMLIST. By including an AMT in the same library as the application load modules, it is only necessary that the target site include the name of the AMT in the @ILMLIST module.

This method requires careful consideration when you attempt to replace modules in an active online environment. Because there are no MODULE dictionary entries for any of the programs or panels when using this method, the REFRESH command deletes the LMT entry and does not perform the NEWCOPY.

If it is necessary to bring in new programs, panels, or applications, you can issue IDENTIFY commands in batch for each of the entities that need to be activated. Once that is complete, you can REFRESH each entity so that the LMT entries are built. You can then DELETE each module. Be sure to make the necessary modification to the AMT and @ILMLIST so that the LMT is built correctly when CICS is recycled at a later time.

Existing programs and panels in load module format can be CICS NEWCOPIED to bring in new versions. Always remember to NEWCOPY all three modules associated with a program if you select this method. You can also IDENTIFY and REFRESH the modules if you do not want to use the CICS NEWCOPY commands and optionally DELETE the modules, depending on whether you want to maintain any module entries in the dictionary.

If it is not necessary to add a new program, panel, or application to the environment while CICS is up; consider disabling the REFRESH command to avoid LMT entries from accidental deletion. Disabling the REFRESH command eliminates the ability to add a new program or panel to the environment while CICS is active. Any new programs and panels must be added to an AMT. You must add new AMT names to @ILMLIST. A recycle of CICS must be performed for the changes to become active. NEWCOPYing an AMT and @ILMLIST has no effect on the in-core LMT while CICS is up.

If the REFRESH command is erroneously executed, it can delete the program or panel from the LMT. If this occurs, execute the IDENTIFY command in batch and the affected application is disabled before the REFRESH is executed again. Although you are executing load modules in this environment, DISPLAY INDEX MODULE does not return an index of modules. Remember that this command is a listing according to the dictionary.

Option 3: Load Modules - LMTBLD=DD and entries in @ILMLIST

This setup builds the LMT from two sources. It is important to only issue REFRESH commands for those entities that exist in the dictionary. Programs and panels that are only defined to an AMT are deleted.

Option 4: Load Modules - LMTBLD=NO with dictionary entries

This setup implies that you are building the in-core module table from the entries pointed to by @ILMLIST, but still maintaining the dictionary MODULE occurrences. The dictionary entries let you execute the REFRESH command in CA Ideal without having the adverse effect of deleting the LMT entry.

Option 5: VLS format only or with load modules

You can run load modules with one of the four options previously mentioned and you can run some of your programs in VLS format.

You should be aware of the following enqueuing considerations if you plan to replace VLS format programs in an active online environment.

Enqueuing Considerations

- An enqueue remains active against a program in VLS format from the time the program is first called until the RUN of the application that accessed the program terminates.
- The object transport utility LOAD obtains exclusive enqueues against the portions of the programs loaded into the VLS libraries.
- The DISABLE RUN command forces users off an active application only after the transaction boundary following the disable time. This means some programs could still remain enqueued even after the disable time arrives. Review Disabling and Enabling an Active Online Application earlier in this chapter.
- Another CICS region running VLS programs and panels of the same name in physically different VLS libraries can lead to failure of the object transport utility ADXSSP24 'PROD' program in use. Please try later. This happens when the SC00OPTS QCODE parameters in each region are not properly set.
- If you replace active programs and panels without disabling the application first, the results can be unpredictable. Under most circumstances, you receive IDADOMLD07 - Global storage DT discrepancy for PGM pgm-name VER ver found, however, other messages are possible.

It is also possible that the transport or mark status fails if a physically different program of the exact same name is running in another region and both regions have the same QCODE setting. For more information about enqueuing considerations for multiple regions, see the “Establishing Multiple Environments” chapter.

Upgrading from an Earlier Version

CA Ideal is "upwardly compatible" meaning that a program that is compiled with an earlier release can still run in the most current release of CA Ideal. However, programs compiled with a higher release of CA Ideal cannot run in a CA Ideal environment at lower release. Therefore, the programs that are updated in your newly upgraded development environment can be re-compiled in batch with the old, lower version of CA Ideal to run in these environments until the production environment is upgraded.

Note: Always upgrade and test a development environment before upgrading your production environment.

Use the following procedure for running and testing programs in a development environment on a new release of CA Ideal and running the same programs in your current production environment that uses a different release of CA Ideal.

- Save your current release installed software load libraries for continued use by production. Save the ADRLIB, ADRPNL, and ADROUT data sets used by the current release development environment for the batch procedure.
- Upgrade the development system to the new release. Plan for an interim period with the new release in development and the current release in production.
- Developers can continue development with the following warning: During the interim period, ongoing maintenance of production applications should not use any new release features. These features would not be recognized in the current release production environment. New applications that are moved into production after the interim test period can use new release features. This should thoroughly test the new release code in the development environment.
- For any maintained programs that must be moved from development to production during the interim period, the following steps should be followed:

1. Develop the changes to the application on the new release development system.
2. Run a CA Ideal batch job to compile the application program(s) and mark status to production.

   This batch job must use the load libraries of the current release and ADRLIB, ADRPNL, and ADROUT data sets. Ensure that the new release development source, object, and panel VLS libraries are included.
3. Run the CA Ideal object transport utility unload next or as the second step of the batch compile and mark. The new release load library for the object transport should be used for the unload process and the current release load library should be used for the loading of the application into the current release production environment.

   This loads the application into the current release production environment.

Note: If the QCODE value in the SCF Options Block (SC00OPTS) was changed in the current release development environment to modify the enqueue names, the same value must be used for the new release development environment. When transporting, ensure that the correct load library is supplied so that the enqueue names match the system with which they are running.

Chapter 15: Asynchronous Execution

This chapter describes how a CA Ideal application can execute as an asynchronous task in CICS using the Session Control Facility (SCF) module SC00NATD. A program can call a subprogram to execute as an asynchronous task using the PDL statement, INITIATE.

An asynchronous transaction does not require user interaction to complete. All or most resources are released at the end of the task. All database updates are fully committed or rolled back at the completion of the asynchronous process.

An asynchronous execution using SC00NATD combines the functions of SC00SAST, which is used for asynchronous compiles, and SC00INIT, the terminal-based driver, to provide a true asynchronous run of a CA Ideal program in CICS. A CICS Command Level program (assembler, COBOL, or PL/I) starts the transaction that runs the application program asynchronously.

Setting Up an Asynchronous Execution

To use the SCASTRAN feature, the SCF Asynchronous Transaction Table must be assembled and linked. The entries in SCASTRAN specify the CA Ideal program to be run.

An entry in SCASTRAN is defined using the following parameters:

- TRANID=*xxxx*
  - *xxxx*-Four-character transaction-id defined in CICS with TWASIZE=64 to invoke program SC00NATD.
- TRNDATA=*ssspppppppppvvvvx*
  - sss-Three-character SYSTEM name where the program resides.
  - pppppppp-Eight-character program name. The name must be padded with blanks if less than eight characters.
  - vvvv-Four-digit version number or PROD for production status programs.
  - x- Code N indicates that the RUN PARAMETER clause is not used when the RUN command for the program is executed, even if data is specified in the FROM option of the EXEC CICS START command invoking the asynchronous run (for example, when starting a process from MQSeries). The default value is Y.

  Note: If no FROM data is specified in the EXEC CICS START command, no run parameter data is expected.

- DFLTUSR=*uuu*
  - *uuu*-Three-character user ID that runs the program. This user ID must be defined with specific authorization for the system specified in the TRNDATA parameter. This applies even if the user ID has ADMINISTRATOR authority. If a user has a SIGNON member, it is executed prior to the RUN. This allows SET RUN commands to be processed for the environment.

A CICS transaction definition must be added for the transaction defined in the SCASTRAN entry. Transaction definitions must specify PROGRAM(SC00NATD) and TWASIZE(64). The command DISPLAY PCT will verify if the transaction has a CICS definition and the corresponding SCASTRAN entry.

Any messages other than internal errors (which are always logged in ADRLOG) are not available for an asynchronous run. If there are errors in a user signon member that fail before the program executes, there is no place to send an error or informational message since there is no terminal. Programs that run asynchronously are not attached to a terminal; therefore any TRANSMIT or NOTIFY statements are invalid for this type of run.

RUNLIST and REPORT output directed to destination LIBRARY will be routed to the user-id specified by the DFLTUSR parameter in the SCASTRAN entry.

This asynchronous run is similar to a run started within CA Ideal by the INITIATE statement. A single 01-level run parameter can be passed to the program (a group field cannot be passed). The run parameter is limited to a maximum of 43 characters. Internally, the RUN command is executed with the parameter option, which leaves a maximum of 43 characters for the actual run parameter data.

To start a transaction that executes SC00NATD, the CICS 'FROM' option on the EXEC CICS START command can be used to specify the data for the CA Ideal run parameter. If the length of this data is more than 43 characters, the option to ignore the run parameter can be specified in the SCASTRAN entry (TRNDATA='*sssppppppppvvvv*N').

The QUITIDEAL option may not be used for asynchronous runs. If the QUITIDEAL option has been set to YES for the site, then there must be a SIGNON member for the DFLTUSR user-id that contains SET RUN QUITIDEAL NO. The command DISPLAY SESSION RUN will show the QUITIDEAL option.

Sample Application

An example of how this feature might be used would be in a CA Ideal program to produce a report. The program CUSTLIST produces a report of all CUSTOMER rows for a particular state determined by the RUN parameter. To run CUSTLIST asynchronously, under the CUST transaction invoked from a COBOL application, perform the following tasks:

1. Define the CUST transaction in CICS with TWASIZE=64 and PROGRAM=SC00NATD.
2. Assemble SCASTRAN.

```
SCASYNTB TYPE=INITIAL
 SCASYNTB TYPE=ENTRY,
    TRANID=CUST,
    TRNDATA='ORDCUSTLIST0001Y',
    DFLTUSR=ASY
 SCASYNTB TYPE=FINAL
```

   The CUSTLIST program version 1 in the system ORD expects a RUN parameter. The ASY user ID must be defined with specific authorization to run a program in system ORD, even if the ASY user ID has been defined with administrator authority.
   Note: Continuation characters must be in column 72.

3. Start transaction CUST and provide a RUN parameter from a COBOL CICS Command Level program.

```
 ...
       DATA DIVISION.
       WORKING-DATA SECTION.
       01  WOR-DATA   PIC X(2) VALUE 'TX'.
       PROCEDURE DIVISION.
       ...
       START-IDEAL.
         EXEC CICS START
           TRANSID('CUST')
           FROM(WOR-DATA)
           LENGTH(+2)
         END-EXEC.
       ...
```

If the report output was directed to DEST LIB, it would be under the ASY user ID as specified by the DFLTUSR parameter in the SCASTRAN entry.

Appendix A: SYSADR Table Declarations for DB2

The appendix describes about the resources of SYSADR table for DB2.

SYSADR.APTAB Table

The SYSADR.APTAB table contains an entry for every generated application plan.

```
EXEC SQL DECLARE SYSADR.APTAB TABLE
                        (APNAME        CHAR(7)       NOT NULL,
                        DTSTAMP        CHAR(12)      NOT NULL,
                        MODNUM         SMALLINT      NOT NULL,
                        PGMNUM         SMALLINT      NOT NULL,
                        TOTSQL         INTEGER       NOT NULL)
```

The SYSADR.APTAB table has the following columns:

- **APNAME**-Application plan name
- **DTSTAMP**-Date and time of the plan generation
- **MODNUM**-Number of static I/O modules comprising the plan
- **PGMNUM**-Total number of programs in the plan
- **TOTSQL**-Total number of SQL statements in the plan

SYSADR.APRES Table

The SYSADR.APRES table presents the resources included in the application plan. It contains an entry for every program participating in the application plan.

```
EXEC SQL     DECLARE SYSADR.APRES TABLE
          (APNAME        CHAR(7)         NOT NULL,
          SQLMOD         CHAR(8)         NOT NULL,
          SYSID          CHAR(3)         NOT NULL,
          PGMNAME        CHAR(8)         NOT NULL,
          PGMVER         CHAR(4)         NOT NULL,
          RESTYP         CHAR(4)         NOT NULL,
          PGMLAN         CHAR(6)         NOT NULL,
          PGMOCC         SMALLINT        NOT NULL,
          SQLNUM         SMALLINT        NOT NULL,
          COLLID         CHAR(18)        NOT NULL)
```

The SYSADR.APRES table has the following columns:

- **APNAME**-Application plan name
- **SQLMOD**-Name of the static I/O module or package containing the program
- **SYSID**-Program system ID
- **PGMNAME**-Program name
- **PGMVER**-Program version (nnn or PROD)
- **RESTYP**-Resource type (APPL, PROG, or SPGM)
- **PGMLAN**-Program language (IDEAL or NONID)
- **PGMOCC**-Number of times the program occurs in the plan
- **SQLNUM**-Number of SQL statements in the program
- **COLLID**-Collection ID of the package

SYSADR.APAUT Table

The SYSADR.APAUT table contains the DB2 table authorization IDs that were replaced by the assigned authorization IDs during the generation process. There is an entry for each assigned ID in the plan.

```
EXEC SQL     DECLARE SYSADR.APAUT TABLE
          (APNAME       CHAR(7)      NOT NULL,
          PGMAUTH       CHAR(8)      NOT NULL,
          PLANAUTH      CHAR(8)      NOT NULL)
```

The SYSADR.APAUT table has the following columns:

- **APNAME**-Application plan name
- **PGMAUTH**-Authorization ID as it appears in the program
- **PLANAUTH**-New authorization ID superseding the program authorization ID

SYSADR Indexes

The SYSADR plan tables are indexed by some of their columns. The SYSADR.APTAB table is indexed by APNAME and the index is unique. The SYSADR.APRES table is indexed by APNAME, SYSID, and PGMNAME. The SYSADR.APAUTH table is indexed by APNAME. The index names coincide with the table names.

```
CREATE UNIQUE INDEX SYSADR.APTAB ON SYSADR.APTAB
          (APNAME)
CREATE INDEX SYSADR.APRES ON SYSADR.APRES
          (APNAME,SYSID,PGMNAME)
CREATE SYSADR.APAUT ON SYSADR.APAUT
          (APNAME)
```

Appendix B: Double-Byte Character Set Support

This appendix describes double-byte character set and 5550 terminal support implemented by CA Ideal and the related CA IPC.

The CICS specification of the SOSI feature in the CICS Terminal Control Table (TCT) provides recognition of the IBM 5550 terminal and network printer.

Failure to specify SOSI in the TCT displays normal characters on the 5550, but DBCS data displays as question marks (?) on the 5550 network printer.

To allow expansion to languages that support a double-byte representation of symbols, CA Ideal requires a level of sensitivity to the way in which you define double-byte character set data. This document uses the following terms to describe double-byte character set data.

DBCS

Double-byte character set.

S/D

Shift to double character (hex 'OE') that precedes DBCS data (on a 5550, known as the shift-out character, where shift-out means shift out of EBCDIC mode).

S/S

Shift to single character (hex 'OF') that follows DBCS data (on a 5550, known as the shift-in character, where shift-in means shift into EBCDIC mode).

I-DBCS

Implicit DBCS, DBCS data without S/D and S/S characters.

E-DBCS

Explicit DBCS, DBCS data with S/D and S/S characters.

The following areas were changed in CA Ideal to accommodate DBCS data:

- The Panel Definition Facility (PDF) extends to display and enter DBCS data in panel fields.
- Panel Definition Language (PDL) adds new string functions for handling DBCS data.
- A new compiler option makes the compiler sensitive to DBCS data when it is moved between fields.
- The Report Definition Facility (RDF) allows DBCS data in certain areas.
- CA Ideal now supports user exits for modifying DBCS data before it is output on a system printer. The CA IPC *Installation Guide* documents this PSS user exit.
- A new CA IPC SCF option displays all CA Ideal system panels and messages in uppercase since the lowercase keyboard positions are used for native character-set support (for example, Katakana in Japan).
- A new CA IPC PMS translate table, PMSTRNDK, was added to the system. It translates non-displayable characters on a 5550. It differs from the regular PMS translate table, PMSTRND, in that the 5550 has a different set of accepted terminal values.

Panel Definition Facility (PDF)

The basic extension to PDF displays and enters DBCS data in Type X panel fields. A Type X panel field can contain DBCS data from any one of the following sources:

- Initial DBCS data defined during editing of the panel layout
- DBCS data entered by the application user
- DBCS data entered or modified through PDL statements in the application program

Important! If a statement in the PDL program modifies a panel field, it is the responsibility of the application program to guarantee that the DBCS data is surrounded with S/D and S/S characters as appropriate.

PDF prevents DBCS data in panel defined fields from accidentally displaying and modifying on non-5550 terminals by first translating the DBCS data to question marks (?). However, the DBCS data is properly preserved in the panel definition, and you can modify it on a 5550 terminal.

Program Definition Language (PDL)

The primary impact on PDL for the handling of DBCS data is that it is the application's responsibility to be aware of and to manipulate the S/D and S/S characters.

For example, if DBCS data is stored in a dataview field without the S/D and S/S characters (implicit-DBCS or I-DBCS) and that field is moved to a panel field, the application program must surround the DBCS data with S/D and S/S characters (explicit-DBCS or E-DBCS).

CA Ideal provides special string functions to facilitate this operation, but the application must instruct PDL to do it. A reverse operation must then be done when the panel field is moved back to the dataview field.

String Functions

$DBCS-ATTACH(alpha-field or 'literal')

This function converts I-DBCS data to E-DBCS data. It builds a temporary string value by concatenating an S/D character, the contents of the referenced alpha field or literal (ignoring any trailing spaces), and an S/S character. You can use the function in any context where you can use a $STRING function. Some examples are:

```
SET A = $DBCS-ATTACH(B)
IF A = $DBCS-ATTACH(B)
LIST $DBCS-ATTACH(B)
```

$DBCS-DETACH(alphanumeric-field or 'literal')

This function converts E-DBCS data to I-DBCS data. It builds a temporary string value by removing the S/D character at the left (ignoring leading spaces), removing any trailing spaces, and then removing the S/S character. A runtime error occurs if the S/D and S/S characters are not found when this function is used or if there are any S/D or S/S characters remaining. You can use an alpha-literal as an operand instead of an alpha-field. You can use the function in any context where you can use a $STRING function, however, it cannot have other functions imbedded in the $DBCS functions.

For $DBCS-DETACH to work successfully, at least one DBCS character must appear between the S/D and S/S characters and only spaces can follow the S/S character. For this reason, do not specify LOW VALUES as an INPUT FILL character for panel fields that have $DBCS-DETACH applied to them.

$DBCS-TYPE(alpha-field)

This string function returns a one-byte code to indicate whether a field is E-DBCS. The codes indicate:

- ' '-There is no S/D character in the field.
- 'L'-There is an S/D character on the left, an S/S character on the right (ignoring right blanks), and no other S/D or S/S characters.
- 'E'-Anything else that implies DBCS and non-DBCS data.

For the $DBCS-TYPE function to return an L, at least one DBCS character must appear between the S/D and S/S characters and only spaces can follow the S/S character. For this reason, do not specify LOW VALUES as an INPUT FILL character for panel fields that the $DBCS-TYPE function tests. You can use this function anywhere you can use a $STRING function. For example:

```
IF $DBCS-TYPE(A) = 'L'
```

$CHAR-TO-HEX(alpha-field)

This string function returns the hexadecimal value of the specified field's contents. The length of the receiving field should be twice the size of the sending field to avoid truncation. An example follows:

```
X = ABCD
SET Y = $CHAR-TO-HEX(X)
Y = C1C2C3C4
```

If X was defined in working data as a 5-byte field, the value of Y is C1C2C3C440.

Data Manipulation

The following rules apply when one Type X field is moved to another under normal PDL processing (with non-DBCS data):

- When the fields are the same length, the data is moved with no inspection of the data values.
- When the sending field is shorter than the receiving field, the sending field is left justified in the receiving field and padded with spaces on the right.
- If the sending field is longer than the receiving field, the sending field is truncated on the right.

The normal PDL rules for comparing Type X fields also applies to Type X fields that contain DBCS data without regard for the value of the SET COMPILE DBCS command.

The manipulation of DBCS data, however, requires an added level of sensitivity. To truncate DBCS data without removing necessary S/D and S/S characters and without leaving half of a double-byte character, you can use a new option during a compile that makes CA Ideal sensitive to DBCS data. The option for a session is:

```
                 {YES|Y}
SET COMPILE DBCS {NO |N}
```

The option for a site is:

```
                      {YES|Y}
SET SITE COMPILE DBCS {NO |N}
```

When you specify YES, the data contents are examined to determine if a S/S character was removed; leaving an unmatched S/D character. If this occurs, further truncation is performed to add the S/S character to the end of the field. CA Ideal also keeps an even number of bytes between the S/D and S/S characters (this ensures that only complete DBCS characters are removed).

When you specify NO, Type X data fields are moved under the normal PDL rules without regard for S/D, S/S, or DBCS characters.

Alphanumeric Literals

You can define alpha-literals containing DBCS data either in the PDL source or as initial values in working data. The form of the literals must be an EBCDIC quote or apostrophe, followed by any number of DBCS characters (surrounded by S/D and S/S characters) or non-DBCS characters, followed by a matching EBCDIC quote or apostrophe. The literal value is everything between the matching EBCDIC quotes or apostrophes, including any S/D or S/S characters. If the application wants to move to or compare with a field that contains implicit DBCS, you must use the $DBCS-DETACH function.

Example

```
SET A = $DBCS-DETACH('dbcs data surrounded by S/D and S/S')
IF A = $DBCS-DETACH('dbcs data surrounded by S/D and S/S')
```

There is no way to put implicit DBCS initial values into working data. They must be initialized through SET statements.

Report Definition Facility

Explicit DBCS data can appear in the following places in the report definition:

- Report title on the parameter fill-in.
- Report page on the parameter fill-in.
- You can specify alphanumeric fields, string expressions, and alpha-literals that contain DBCS data in the Field definitions for headings and detail. The DBCS data must be surrounded by S/D and S/S characters (E- DBCS) or specified with $DBCS-ATTACH functions.
- The column definition for the detail definition.

Session Control Facility

You are required to specify a SCF option to display all CA Ideal system panels and messages in upper-case and to reserve the lower-case keyboard positions for native character set support. You can specify this option in two ways:

- With the following set command:

```
            {UPC      }
SET COMMAND {UPPERCASE} ON
```

- With the site option:

```
UPPER CASE PANELS AND MESSAGES ON
```

This option appears on the Set SCF site options fill-in that is accessed by specifying the command:

```
SET COMMAND SITE [OPTIONS]
```

Dataviews

If a field is found with a type code of K or Y (which implies I-DBCS data) during a CATALOG DATAVIEW, CA Ideal treats it as if there was an X type code. Although Datadictionary supports types K and Y for DBCS data, the CA Ideal dataview compiler does not support these types. They are translated to type X. It is up to the application program to ensure that the data is I-DBCS.

Print Subsystem (PSS)

CA Ideal provides a user exit capability that lets you tailor certain Print Subsystem (PSS) functions in an individualized manner, such as the printing of DBCS data on a system printer. For more information, see the
CA IPC Installation Guide.

Installation Considerations

Increasing the size of ADRPNL

When you specify the SET COMMAND UPPERCASE command, a set of upper-case CA Ideal system panels appears. This additional set of panels is loaded into ADRPNL during installation.

Miscellaneous

- CA Ideal lets you enter DBCS data into:
  - CA Ideal data members that are not executed
  - Short descriptions and text for CA Ideal report, program, and panel identification fill-ins
- CA Ideal does not let you enter DBCS data into:
  - CA Ideal data members that are executed
  - Command area (this implies no DBCS in CHANGE and FIND commands)
  - Any identifiers (names) for fields, programs, reports, panels, systems, users, and so on
  - Other panel fields where DBCS data is not reasonable or where system-defined edits preclude the use of DBCS data
- If DBCS data is entered when editing a report, working data, program parameter data, procedures, or members, use the SET EDIT CASE MIXED command. If you specify UPPER, the DBCS characters are shifted. However, you can use SET PANEL LAYOUTCASE UPPER when editing a panel layout without shifting DBCS characters to upper case.
- Use SET SITE ENVIRONMENT DATEFOR 'MONTH DD, YEAR' instead of the default LCMONTH DD, YEAR.
- Do not use DBCS data in the edit pattern specified in a $EDIT function.
- When using the $SUBSTR function with DBCS data, use caution to ensure that the resulting string contains S/D and S/S characters.

Appendix C: Authorization Table

The following table contains all functions, their associated values, and the numeric value representing a functional keyword that is used in CA Ideal.

Keywords

Functional Keywords and Synonyms	FUNC-ATZ-CODE	CA Ideal Commands
ALTER-PROGRAM ALTER-PGM	1	ALTER PROGRAM and ALTER PANEL commands
ALTER-SIGNON-PANEL ALT-SIGNON-PNL ALT-SIG-PNL	73	ALTER SIGNON PANEL command
CATALOG-DATAVIEW CATALOG_DVW	5	CATALOG DVW command
COMPILE-BATCH	7	COMPILE command, batch
COMPILE-ONLINE	6	COMPILE command on-line
COMPILE-SYNC	82	COMPILE command, synchronous
COPY-MEM-ACROSS-USER COPY-MEM-ACROSS-USR COPY-MEM-ACR-USR	9	COPY/DUP MEMBER *xxxxxxx* USER *yyy* commands
COPY-PGM-ACROSS-SYSTEM* COPY-PGM-ACROSS-SYS COPY-PGM-ACR-SYS	8	COPY PROGRAM *xxxxxxx* SYSTEM *yyy* COMMAND
CREATE-DATAVIEW	93	CREATE DATAVIEW command

CREATE-EDIT-SYSTEM
CREATE-EDIT-SYS
	10	CREATE SYSTEM or EDIT SYSTEM commands

CREATE-EDIT-USER
CREATE-EDIT-USR
	11	CRE USER or EDIT USER commands

CREATE-MEM-ACROSS-USR
CREATE-MEM-ACR-USR
	12	CREATE MEMBER *xxxxxxx* USER *yyy* command

CREATE-MODULE
CREATE-MOD
	13	CREATE MODULE command

Functional Keywords and Synonyms	FUNC-ATZ-CODE	CA Ideal Commands

CREATE PACKAGE
CREATE-PKG
CRE-PKG
	124	CREATE PACKAGE command

CREATE-PANEL
CREATE-PNL
	15	CREATE PANEL command

CREATE-PLAN	84	CREATE PLAN command

CREATE-PROGRAM
CREATE-PGM
	14	CREATE PROGRAM command

CREATE-REPORT
CREATE-RPT
	16	CREATE REPORT command

DEBUG-TEST	160	DEBUG program command (for test programs)
DEBUG-PROD	109	DEBUG program command (for PROD programs)
DELETE-DATAVIEW DEL-DVW	94	DELETE DATAVIEW command
DELETE-MEMBER-ACROSS-USER DELETE-MEM-ACR-USR	79	DELETE MEMBER *xxxxxxx* USER *yyy* command
DELETE-MODULE DELETE-MOD	17	DELETE MODULE command
DELETE-PACKAGE DELETE-PKG DEL-PKG	125	DELETE PACKAGE command
DELETE-PANEL DELETE-PNL	19	DELETE PANEL command
DELETE-PLAN DELETE-PLA	87	DELETE PLAN command
DELETE-PROGRAM DELETE-PGM	18	DELETE PROGRAM command
DELETE-REPORT DELETE-RPT	20	DELETE REPORT command
DELETE-SYSTEM DELETE-SYS	21	DELETE SYSTEM command

Functional Keywords and Synonyms	FUNC-ATZ-CODE	CA Ideal Commands
DELETE-USER DELETE-USR	22	DELETE USER command
DEQUEUE DEQUE	23	DEQUEUE command
DISPLAY-AUTHORIZATIONS DISPLAY-ATZ DIS-ATZ	28	DISPLAY ATZ OPTIONS
DISPLAY-DATAVIEW DIS-DATAVIEW DISPLAY-DVW DIS-DVW	27	DIS DATAVIEW command
DISPLAY-INDEX-ALL-MODULE DIS-IND-ALL-MODULE DIS-IND-ALL-MOD	24	DIS INEX ALL MODULE command
DISPLAY-INDEX-ALL-PROGRAM DIS-IND-ALL-PROGRAM DIS-IND-ALL-PGM	111	DIS IND ALL PGM command
DIS-INDEX-MEM-ACROSS-USER DIS-IND-MEM-ACROSS-USR DIS-IND-MEM-ACR-USER DIS-IND-ACR-USR	25	DISPLAY INDEX MEMBER USER *yyy* command
DISPLAY-INDEX-RELATED DIS-INDEX-RELATED DIS-IND-REALTED DIS-IND-REL	26	DISPLAY INDEX with RELATED clause command
DISPLAY-MEMBER-ACROSS-USER DISPLAY-MEM-ACROSS-USER DISPLAY-MEM-ACR-USR DIS-MEMBER-ACROSS-USER DIS-MEM-ACROSS-USER DIS-MEM-ACR-USR	29	DISPLAY MEMBER *xxxxxxx* USER *yyy* command
DISPLAY-OUTPUT-ACROSS-USER DIS-OUT-ACR-USR	113	DISPLAY OUTOUT nnn produced by another user (output remains on ADROUT)

Functional Keywords and Synonyms	**FUNC-ATZ-CODE**	**CA Ideal Commands**
DISPLAY-PACKAGE DISPLAY-PKG DIS-PKG	126	DISPLAY PACKAGE command
DISPLAY-PANEL DISPLAY-PNL DIS-PNL	31	DISPLAY PANEL command
DISPLAY-PLAN DIS-PLA	86	DISPLAY PLAN command
DISPLAY-PROGRAM DISPLAY-PGM DIS-PGM	30	DISPLAY PROGRAM command
DISPLAY-USER DISPLAY-USR DIS-USR	34	DISPLAY USER command (your own definition)
DISPLAY-REPORT DISPLAY-RPT DIS-RPT	32	DISPLA Y REPORT command
DISPLAY-SYSTEM DISPLAY-SYS DIS-SYS	33	DISPLAY SYSTEM command
DISPLAY-USER-ACROSS-USER DISPLAY-USR-ACROSS-USR DISPLAY-USR-ACR-USR DIS-USR-ACR-USR	35	DISPLAY USER command (other than your own definition)
DUPLICATE-DATAVIEW DUPLICATE-DVW DUP-DVW	95	DUP DATAVIEW command
DUPLICATE-MEMBER-ACROSS-USER DUPLICATE-MEM-ACROSS-USR DUPLICATE-MEM-ACR-USR DUP-MEM-ACR-USR	83	DUP MEMBER *xxxxxx* USER *uuu* command
DUPLICATE-PACKAGE DUPLICATE-PKG DUP-PKG	128	DUPLICATE PACKAGE command

Functional Keywords and Synonyms	FUNC-ATZ-CODE	CA Ideal Commands
DUPLICATE-PANEL DUPLICATE-PNL DUP-PNL	38	DUPLICATE PANEL command
DUPLICATE-PANEL-ACROSS-SYSTEM DUPLICATE-PNL-ACROSS-SYS DUPLICATE-PNL-ACR-SYS DUP-PNL-ACR-SYS	39	DUP PANEL *xxxxxxx* SYSTEM *yyy* command
DUPLICATE-PLAN DUP-PLA	89	DUPLICATE PLAN command
DUPLICATE-PROGRAM DUPLICATE-PMG DUP-PGM	36	DUP PROGRAM command
DISPLAY-PLAN DIS-PLA	86	DISPLAY PLAN command
DISPLAY-PROGRAM DISPLAY-PGM DIS-PGM	30	DISPLAY PROGRAM command
DUPLICATE-PROGRAM-ACROSS-SYSTEM DUPLICATE-PGM-ACROSS-SYS DUPLICATE-PGM-ACR-SYS *DUP-PGM-ACR-SYS	37	DUP PROGRAM *xxxxxxx* SYSTEM *yyy* command
DUPLICATE-REPORT DUPLICATE-RPT DUP-RPT	40	DUPLICATE REPORT command
DUPLICATE-REPORT-ACROSS-SYSTEM DUPLICATE-RPT-ACROSS-SYS DUPLICATE-RPT-ACR-SYS DUP-RPT-ACR-SYS	41	DUP REPORT *xxxxxxx* SYSTEM *yyy* command
DUPLICATE-SYSTEM DUPLICATE-SYS DUP-SYS	42	DUP SYSTEM command
DUPLICATE-USER DUPLICATE-USR DUP-USR	80	DUPLICATE USER command

Functional Keywords and Synonyms	**FUNC-ATZ-CODE**	**CA Ideal Commands**
EDIT-DATAVIEW EDIT-DVW	96	EDIT DATAVIEW command
EDIT-MEMBER-ACROSS-USER EDIT-MEM-ACROSS-USR EDIT-MEM-ACR-USR	43	EDIT MEMBER *xxxxxxx* USER *yyy* command
EDIT-PACKAGE EDIT-PKG EDI-PKG	128	EDIT PACKAGE command
EDIT-PANEL EDIT-PNL	49	EDIT PANEL command
EDIT-PLAN EDIT-PLA	85	EDIT PLAN command
EDIT-PROGRAM EDIT-PGM	45	EDIT PROGRAM command
EDIT-PROGRAM-IDE EDIT-PGM-IDE	44	EDIT PROGRAM IDENTIFICATION command
EDIT-PROGRAM-RES EDIT-PGM-RES	46	EDIT PROGRAM RESOURCE command
*EDIT-PROGRAM-RES-SUBPGM EDIT-PRG-RES-SUBPGM EDIT-PRG-RES-SUB	47	EDIT PROGRAM RESOURCE (for specifying subprograms to be called)
EDIT-REPORT EDIT-RPT	48	EDIT REPORT command
ENABLE-DISABLE-RUN ENA-DIS-RUN	50	ENABLE RUN and DISABLE RUN commands
EXPLAIN-PLAN EXP-PLA	92	EXPLAIN PLAN command
EXPORT DATAVIEW	107	EXPORT DATAVIEW command
EXPORT-MEMBER	105	EXPORT MEMBER command
EXPORT-PANEL	101	EXPORT PANEL command
EXPORT-PROGRAM	99	EXPORT PROGRAM command
EXPORT-REPORT	103	EXPORT REPORT command

Functional Keywords and Synonyms	FUNC-ATZ-CODE	CA Ideal Commands
GENERATE-PACKAGE GENERATE-PKG GEN-PKG	129	GENERATE PACKAGE command
GENERATE-PLAN GEN-PLA	88	GENERATE PLAN command
IDENTIFY-MODULE IDENTIFY-MOD	51	IDENTIFY MODULE command
IMPORT-DATAVIEW	106	IMPORT DATAVIEW command
IMPORT-HELP	108	IMPORT HELP command
IMPORT-MEMBER	104	IMPORT MEMBER command
IMPORT-PANEL	100	IMPORT PANEL command
IMPORT-PROGRAM	98	IMPORT PROGRAM command
IMPORT-REPORT	102	IMPORT REPORT command
MARK-STATUS-DATAVIEW MARK-STATUS-DVW MARK-DATAVIEW MARK-DVW	97	MARK STATUS DATAVIEW command
MARK-STATUS-PANEL MARK-STATUS-PNL MARK-PANEL MARK-PNL	53	MARK STATUS PNL command
MARK-STATUS-PROGRAM MARK-STATUS-PGM MARK-PROGRAM MARK-PGM	52	MARK STATUS PROGRAM command
MARK-STATUS-REPORT MARK-STATUS-RPT MARK-REPORT MARK-RPT	54	MARK STATUS REPORT command
MARK-STATUS-SYSTEM MARK-STATUS-SYS MARK-SYSTEM MARK-SYS	55	MARK STATUS SYSTEM command

Functional Keywords and Synonyms	FUNC-ATZ-CODE	CA Ideal Commands
MARK-STATUS-USER MARK-STATUS-USR MARK-USER MARK-USR	56	MARK STATUS USR command
PRINT-DATAVIEW PRINT-DVW PRI-DVW	57	PRINT DATAVIEW command
PRINT-INDEX-ALL-MODULE PRI-IND-ALL-MODULE PRI-IND-ALL-MOD	77	PRINT INDEX ALL MODULE command
PRINT-INDEX-ALL-PROGRAM PRINT-IND-ALL PROGRAM PRI-IND-ALL-PGM	112	PRINT INDEX MEMBER USER *xxx* command
PRINT-INDEX-MEM-ACROSS-USER PRINT-IND-MEM-ACR-USR PRI-IND-MEM-ACR-USR	64	PRI INDEX with RELATED clause command
PRINT-INDEX-RELATED PRINT-IND-RELATED PRINT-INDEX-REL PRINT-IND-REL	66	PRINT MEMBER (your own members)
PRINT-MEMBER PRINT-MEM PRI-MEM	58	PRINT MEMBER (your own members)
PRINT-MEMBER-ACROSS-USER PRINT-MEM-ACROSS-USER PRINT-MEM-ACR-USR PRI-MEMBER-ACROSS-USER PRI-MEM-ACROSS-USER PRI-MEM-ACR-USR	81	PRINT MEMBER xxxxxxxx USER yyy command
PRINT-OUTPUT-ACROSS-USER PRINT-OUT-ACR-USR	114	PRINT OUTPUT nnn produced by another user (output remains on ADROUT in LEAVE status)

Functional Keywords and Synonyms	FUNC-ATZ-CODE	CA Ideal Commands
PRINT-PACKAGE PRINT-PKG PRI-PKG	130	PRINT PACKAGE command
PRINT-PROGRAM PRINT-PGM PRI-PGM	59	PRINT PROGRAM command
PRINT-PANEL PRINT-PNL PRI-PNL	60	PRINT PANEL command
PRINT-PLAN PRI-PLA	91	PRINT PLAN command
PRINT-REPORT PRINT-RPT PRI-RPT	61	PRINT REPORT command
PRINT-SYSTEM PRINT-SYS PRI-SYS	62	PRINT SYSTEM command
PRINT-USER PRINT-USR PRI-USR	65	PRINT USER (your own definitions)
PRINT-USER-ACROSS-USER PRINT-USR-ACR-USR PRI-USR-ACR-USR	63	PRIT USER (definition of another user)
REBIND REB	74	REBIND command
REFRESH	78	REFRESH command
RUN	67	RUN (test version)
RUN-PROD RUN-PRD	68	RUN (production version)
RUN-PROD-USING-PANEL* RUN-PROD-USE-PNL RUN-PRD-USE-PGM	69	RUN (production program when a panel is accessed)

Functional Keywords and Synonyms	FUNC-ATZ-CODE	CA Ideal Commands
RUN-PROD-USING-PROGRAM* RUN-PROD-USE-PGM RUN-PRD-USE-PGM	70	RUN (production program when a subprogram is called)
SELEECT-SYSTEM SELECT-SYS SEL-SYS	71	SELECT SYSTEM command

SET-SITE-OPTIONS
SET-SITE-OPTS
SET-SITE-OPT
SET-SITE
	72	SET SITE commands

SUBMIT-MEMBER SUBMIT-MEM	75	SUBMIT (your own members)
SUBMIT-MEMBER-ACROSS-USER SUBMIT-MEM-ACROSS-USR	76	SUBMMIT (another user's members)