# CA Ideal™ for CA Datacom®

## Problem Determination Guide

**Version 14.02**

# CA Technologies Product References

This document references the following CA products:

- CA Datacom®/DB
- CA Datacom® CICS Services
- CA Ideal™ for Datacom® (CA Ideal)
- CA Ideal™ for DB2
- CA Top Secret®
- CA ACF2™

# 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: Problem Determination

The *Problem Determination Guide* describes tools and procedures that you can use to define, clarify, document, and solve problems arising from CA Ideal applications before (or without) calling CA Ideal Technical Support.

Site Administrators and systems programmers who are knowledgeable in CA Datacom/DB, teleprocessing monitor systems, and CA Ideal use and general architecture should read this guide.

This chapter introduces concepts and descriptions that are essential to effective problem solving. It includes an overview of the CA Ideal software environment, descriptions of the basic classes of problems, the format of CA Ideal error messages, and references to non-ideal problem solving tools.

For a complete overview of the CA Ideal software environment, see the “System Overview” chapter in the *Administration Guide*.

## CA InterProduct Components (CA IPCs)

The CA Inter-Product Components (CA IPCs) provide common functions to CA Ideal and an interface between CA Ideal and the external environment. These programs let CA Ideal process user requests and 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 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 and 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 (Panel Definition Facility)**

Processes the CA Ideal commands that define, paint, 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)**

Set up run-time services for acquiring, sending, receiving, and managing 327x-type terminal messages. PMS validates date and 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)**

Handle user requests from an online terminal session and a CA Ideal batch session. SCF provides CA Ideal sign-on 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.

There are two types of VLS library members. 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-Output library and printer destinations.

- IDDAT-Data member library.
- IDDVW-Dataview object library. For a DB2 site, you can include plan definitions here or in a separate plan library.
- User source, object, and panel libraries. Defaults are as follows:
  - In z/OS-ID$IDSRC, ID$IDOBJ, and ID$IDPNL
  - In VSE-IDL$IDS, IDL$IDO, IDL$IDP

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

VLS has a batch utility for library maintenance (VLSUTIL). For more information about its usage, see the *Administration Guide*.

**EDK (Editor Kernel)**

Maps fill-in panels to VLS members providing editing commands and checkpoint and rollback functions.

# CA Ideal Security

Security for CA Ideal access consists of both external and internal facilities. You can use CA Ideal internal security to control access to CA Ideal. Sites can also use the CA Standard Security Facility (SSF) to interface with external security systems, such as CA Top Secret, CA ACF2, or IBM's RACF. The SSF interface lets you use any of these security products in both online and batch environments.

During sign-on, 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 sign-on transaction (standard, express, or transparent) with external security.

For more information about internal and external security, see the *Administration Guide*.

# 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 online and in batch to build object modules of programs, panels, and reports.
- **Executor**-Control the CA Ideal run-time environment.
- **Online Services**-These panels and processors create and maintain user and system definitions and process commands such as DISPLAY INDEX, CREATE, DELETE, DUPLICATE, MARK, PRINT, DISPLAY, CATALOG DATAVIEW, and SUBMIT.
- **Editors**-A separate editor or processor is available in CA Ideal for each CA Ideal application component. For more information, see the next section, Application Components, VLS, and the Datadictionary.
- **Batch Utilities**-Used for site administration functions.

For a list of individual CA Ideal internal modules as they relate to the CA Ideal Trace Facility, see Appendix A, “Dial Trace Codes.”

## Application Components, VLS, and the Datadictionary

The following table shows how CA Ideal application components (systems, users, dataviews, data members, panels, reports, programs, and plans) are recorded in the Datadictionary, stored on a VLS library, or both. For descriptions of VLS member name formats and details of how CA Ideal commands affect VLS members, see Appendix B, “VLS Members and Utilities.”

| **VLS Libraries** | | | | | | |
|---|---|---|---|---|---|---|
| **Component** | **Dictionary** | **Source** | **Object** | **Panel** | ***IDDVW** | **IDDAT** |
| SYSTEM | x | | | | | |
| USER | X | | | | | |
| DATAVIEW | X | | | | X | |
| MEMBER | | | | | | X |
| PANEL IDENT | X | | | | | |
| LAY/SUM/EXD, etc. | | | X | X | | |
| REPORT IDENT | X | | | | | |
| PAR/HEA/DET/COL | | X | | | | |
| PROGRAM IDENT | X | | | | | |
| RESOURCE | X | | | | | |

| VLS Libraries | | | | | | |
|---|---|---|---|---|---|---|
| **Component** | **Dictionary** | **Source** | **Object** | **Panel** | ***IDDVW** | **IDDAT** |
| PARAMETER | X | X | X | | | |
| WORKING DATA | | X | X | | | |
| PROCEDURE | X | X | X | | | |
| *PLAN | X | | | | | X |

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

## Storing Application Components

Entries in the Datadictionary 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 Datadictionary 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 Datadictionary, the VLS source library, object library, and panel libraries are specified, and the appropriate relationships are created on the Datadictionary.

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

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**

Defines in CA Ideal and added to the dataview source library and the Datadictionary. The CATALOG DVW command accesses the source library and the dictionary tables and creates an object module in the IDDVW VLS library. You can create unmodeled dataviews for sequential files and VSAM files.

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

**Panels and Reports**

Are modeled in the Datadictionary 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**

Are modeled in the Datadictionary and an entity-occurrence is created in the dictionary when the program identification is entered. Relationship occurrences are added when the program resource table is entered. 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.

## Compilation and VLS Object Modules

The first time a CA Ideal application is compiled:

- The program working data, parameter data, and the attributes of each panel are compiled into separate object modules, which are not deleted after compilation.
- These objects, along with any cataloged dataviews, are merged into composite object modules.
- The program procedure and report definitions are compiled and merged with the composite object module into an executable object.

The program symbol table is also built at compile time.

Each time a program is recompiled the program procedure is recompiled into the executable object module. Other parts of the program are only recompiled if they were changed since the last compile.

## CA Ideal VLS Operations

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

## CREATE

**DVW**

Creates an unmodeled dataview that is a single member in the IDDVW library (type K).

**MEM**

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

**PGM**

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

**PNL**

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

**RPT**

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

## EDIT

**DVW**

For an unmodeled dataview, a copy of the member on IDDVW is made with an edit-indicator of E. At normal end of EDIT, the E member is deleted.

**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.

**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. When the RESOURCE definition is edited, a temporary member (type E) is created in IDDAT, as well as a copy of the member with an edit-indicator of E. At normal end of EDIT, the E members are deleted.

**PNL**

Defines 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.

**RPT**

Defines 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.

**Note:** If E members are left on the source or IDDAT libraries due to edit sessions being abnormally terminated, VLSUTIL DELETE may be used to remove old E members.

## DUPLICATE

**DVW**

For an unmodeled dataview, a copy of the IDDVW member is made with the NEWNAME or NEXT version. The DUP command internally invokes the EDIT command for the new dataview.

**MEM**

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

**PGM**

Defines 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. In addition, the DUP command internally invokes an EDIT command for the new program.

**PNL**

Defines 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. In addition, the DUP command internally invokes an EDIT command for the new panel.

**RPT**

Defines 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. In addition, the DUP command internally invokes an EDIT command for the new report.

## MARK STATUS

**PGM**

The only VLS members affected by the MARK STATUS command 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. In addition, if there is a previous PRD version of the same program, the associated members are renamed so that the PRD is changed back to the original numeric version number.

**Note:** This description of the MARK STATUS process is only valid within the CA Ideal environment. The VLS members are not affected by STATUS changes made only in the Datadictionary environment to the dictionary ENTITY.

This lets the run-time executor 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, since otherwise CA Ideal has no means of determining which of the multiple possible 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 and 007 are in TEST and 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 order shown:

```
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 order shown here):

```
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 could determine 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. Also, neither of the type V members (working data object) was affected by the MARK STATUS command. These members are used only at compilation time.

For all other entities, MARK STATUS only affects the Datadictionary definition. The corresponding VLS members are not updated.

## DELETE

- **DVW-**The IDDVW members, both K and D types, are deleted.
- **MEM-**The data member is deleted.
- **PGM-**All program source and object members are deleted.
- **PNL-**The panel source and object members are deleted.
- **RPT-**The report source 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, are missing.

## CATALOG

**DVW** Creates or replaces the dataview object member (type D) in the IDDVW library.

## Processing the Field Attribute and Symbol Tables

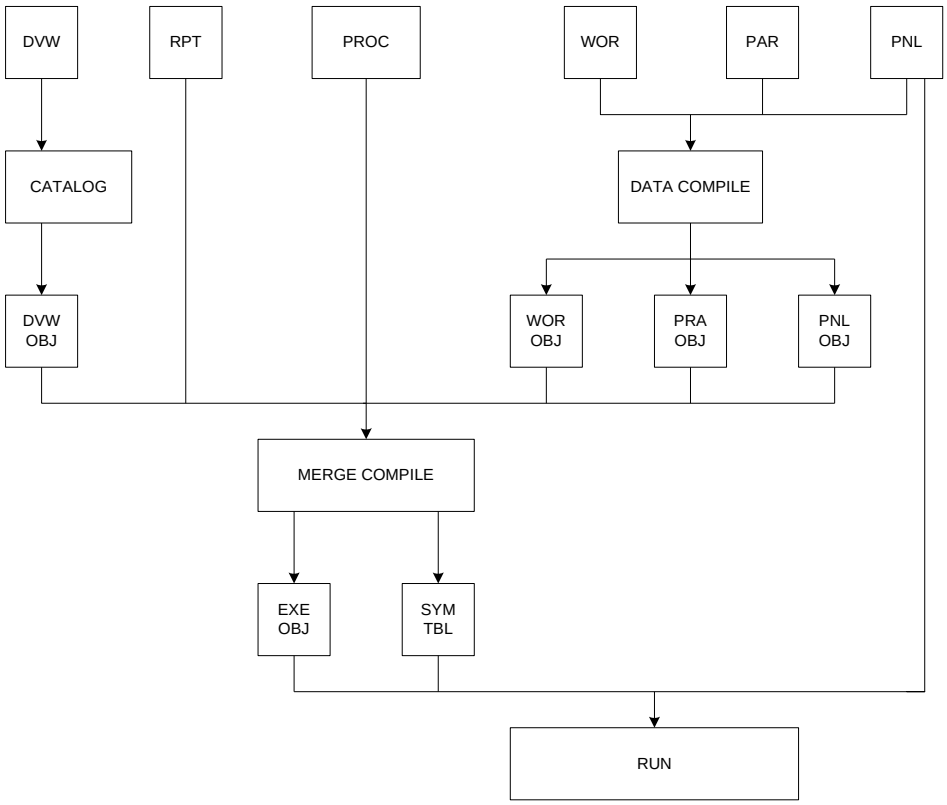
The maximum number of symbols-such as names and identifiers-in a CA Ideal application depends greatly on the amount of storage CA Ideal has available for the Field Attribute Table and the Symbol Table.

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 32KB. The FAT and Symbol Tables can span multiple blocks. A single data entity occurrence, which means a 01 level data item, dataview, or panel, cannot exceed the 32KB limit. This means that no 01 level can have more than 1,600 fields or it exceeds the FAT table limit. A 01 level can have between 800 and 1,600 fields (depending on the size of the symbols) or it exceeds the Symbol Table limit.

The following graphic shows the compilation process in CA Ideal application:




## 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.

**Note:** For the sake of convenience, load module in this guide refers to both load modules and phases.

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 Datadictionary 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.

## Execution of CA Ideal Applications in a CICS Environment

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

The executable object is loaded from the VLS object library or load library and executed. The reentrant portion (the program procedure and report) can be shared among users. The non-reentrant portion (program parameter data, working data, and dataview buffers) is not shared, and each user gets a copy.

When a panel is accessed, the panel definition (as opposed to the panel object) is retrieved from the panel library or load library.

The panel member and module is also grouped into reentrant and non-reentrant parts. The reentrant part (the panel definition) can be loaded into global storage and shared among users. The non-reentrant part (the panel data) is not shared and each user gets a copy.

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 and CICS ESDSA.

If programs and panels were converted to load modules, they are controlled by CICS, and the CA Ideal run status has no effect.

The program symbol table is loaded only when a run-time error occurs.

In a DB2 environment, execution of programs containing SQL in static mode requires an application plan. The *Administration Guide* describes how application plans are generated and used at run time.

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

# Categories of Problems

This section describes the basic categories of problems that you can encounter while in CA Ideal. Understanding these categories is useful in identifying problems and in communicating them to CA Ideal Technical Support.

## CICS or Operating System Crash

A crash is an abnormal end that brings down CICS or the operating system, including CA Ideal and any other transactions currently active. You can produce CICS or system dumps in various formats.

### Transaction Abend/Batch Abend

A transaction or batch abend is an abnormal end by CICS or the operating system that affects only a CA Ideal transaction.

VPE intercepts a CICS transaction abend encountered while CA Ideal is executing and returns control to CICS. CICS produces a dump and CICS statistics are produced at shut down.

A batch transaction abend returns control to the operating system and displays a z/OS or VSE error message.

Common abends are as follows:

| CICS Abend Code | z/OS Batch Abend Code | VSE Batch Msg. No. | Cause |
|---|---|---|---|
| ATNI, ATCH | S222, S122 | N/A | Program canceled by operator console, master terminal, or DFHNEP. |
| AICA | S322, S522 | N/A | Runaway task due to:<br>A) Exceeding time limit parameter ICVR in SIT table (DFHSIT);<br>B) Program loop;<br>C) Exceeding system default for CPU use. |
| APCT | S806 | OSO5I | Program not found due to:<br>A) Program not in PPT;<br>B) Program not in the load library;<br>C) Program disabled.<br>Causes include a user subprogram call with the subprogram not found in the STEPLIB/LIBDEF, DFHPPT, or DFHRPL. |

| CICS Abend Code | z/OS Batch Abend Code | VSE Batch Msg. No. | Cause |
|---|---|---|---|
| ASRA | SOC1...SOCF | OSO3I | Most other abends. Causes include misapplied zaps, a problem with a call to a subprogram, or storage violation. A S0C3 results from executing a CA Ideal load module. |

For more information about abend codes due to unrecoverable errors, see the *Messages and Codes Guide*.

## Internal Error

An internal error is an unexpected result produced by a CA Ideal or CA IPC process. Most internal error messages contain the identifier INTERR. CA Ideal or the CA IPC keeps control and produces an ADRLOG entry. Online, ADRLOG prints when the teleprocessing monitor comes down. You can display a panel containing information about the error using the CA Ideal command DISPLAY ERROR. For more information, see Internal Error Facility in the “Diagnostic Tools” chapter. In batch, ADRLOG prints when the batch job ends.

## Compiler Abort

A compiler abort is a system-related error encountered during a compile that terminates the compile. CA Ideal retains control. An error message displays. Online, further information is listed in the output member indicated on the screen. Use a DISPLAY OUTPUT command to view this data. In batch, further information is listed in the compile listing. This information identifies the error as an abort and lists any internal errors that caused the abort. Causes include trying to access a VLS library that is full or that cannot be accessed.

## Compiler Fatal Error

A compiler fatal error is a program-related error encountered during a compile. It is the same as a compiler abort, except that it is generally caused by a problem with the program itself, as opposed to a condition in the CA Ideal system. For example, it can be due to not finding the PDL source specified in the compile or trying to access a program that is already in use. The error message and the indicated output member, online, or the compile listing, in batch, identifies the type of problem.

## Compiler Error

A compiler error is a programming error (such as a syntax error) encountered during a compile. The error does not terminate the compile, but it does prevent the object from being created. The error is recorded in the compile listing. It is highlighted in the source if SET EDIT HIGHLIGHT ERROR is in effect and you specified the MEL (Mark Error Lines) compile option.

### Runtime Error

A run-time error is an error encountered during execution of a CA Ideal program, either:

- A system error, for example, a Class SYS, Type SYS internal error. A system error always invokes the default CA Ideal Error Procedure. This terminates the run and lists the error either in the output library (online, use DISPLAY OUT) or in the current RUNLIST DD/DLBL.
- Any other internal error, DVW type error, or illogical program condition. Such errors pass control to the current user <<ERROR>> Procedure, if one is defined, or invoke the default CA Ideal <<ERROR>> Procedure. For more information, see the "PDL ERROR Procedure" chapter.

# CA Ideal Error Messages

CA Ideal produces messages to assist you. Some of these messages are issued when you make an error, some are produced when CA Ideal requires some response from you, and some messages simply provide information about the activity you performed.

CA Ideal messages can be issued in a variety of places. Many messages appear on the message line of the terminal screen. However, messages can appear in other places too. If an error occurs during sign on, a message appears in the top line of the signon screen. Messages also are recorded in ADRLOG, and can be viewed as the result of a DISPLAY ERROR command or in the CICS job output. Finally, messages can appear in a compilation listing or at the bottom of the display that results from a CATALOG, DISPLAY, or PRINT DATAVIEW command.

## Error Message Format

The numbers that identify CA Ideal error messages display in the following format:

*#t-r-ppxxxxxxxxs* - text

**#t**

For messages generated by an asynchronous task only (asynchronous compile or network print), the asynchronous task number.

**Note:** For all other types of messages, these positions are omitted.

**r**

Displays the region number (on the display screen) to which the message applies.

**pp**

Displays the product code. This is either ID for CA Ideal, or SC for SCF (Session Control Facility).

**xxxxxxxx**

Identifies the error message. This information is extremely important because it uniquely identifies the error message. Several messages can be issued with similar text and are distinguishable only by their identification.

**s**

Displays the severity code of the message. The list is as follows:

- **E** (ERROR)-An error that does not terminate processing occurred.
- **F** (FATAL)-An error that *terminates* processing occurred.
- **I** (INFORMATION)-Informational message only-no action is required.
- **W** (WARNING)-A *condition* has been detected which may lead to incorrect or unexpected results.

Each severity code has an associated numeric code that can conditionally execute batch job steps and can be returned by the function $RETURN-CODE in a CA Ideal batch job stream or in PDL. For more information, see the *Working in the Environment Guide* and the *Programming Reference Guide*.

**text**

Displays the description of the message.

### Example

In the following error message,

```
#0-2-IDCMASYD10I - Compile for PGM PROG1 successful;...
```

#0 indicates that the message was generated from asynchronous task number 0. 2 indicates display region number 2 on the terminal. ID indicates that CA Ideal is the product in control. CMASYD10 is the message identification. I (INFORMATION) is the severity level.

**Note:** If the text of an error message includes the keyword "INTERR" (internal error), this indicates an internal system error, and CA Ideal takes additional diagnostic actions.

For complete descriptions of all CA Ideal error messages, see the *Messages and Codes Guide*.

# CA Datacom Tools

The following are aids to diagnosing an application's performance in the CA Datacom environment.

# CA Datacom/DB Return Codes

You can use CA Ideal built-in functions in <<ERROR>> Procedures to identify errors. These include several codes CA Datacom/DB returns due to errors encountered handling dataviews.

The $ERROR-TYPE function with a value of DVW indicates such errors. You can find additional information about such errors using the functions $ERROR-DVW-STATUS and $ERROR-DVW-INTERNAL-STATUS. A value returned by $ERROR-DVW-STATUS that starts with I (for example, I9) indicates a CA Ideal error. For explanations, see the *Programming Reference Guide*.

A two-digit dataview status (for example, 05) indicates a DB return code (for an 05, URT could not be opened). In this case, the value returned by $ERROR-DVW-INTERNAL-STATUS can be the DB subcode, where applicable. If the DB return code received does not have any subcodes (as documented in the *CA Datacom/DB Summary of Messages*), ignore this value. For explanations, see the *CA Datacom/DB Summary of Messages*.

For an SQL dataview, $ERROR-DVW-STATUS returns the SQLCODE from the SQLCA. This is a character string with a value in the range -999 to +999. A negative value indicates an error. If a value of -117 or -118 is detected, CA Ideal replaces this value with the CA Datacom return code. See the $ERROR-DVW-STATUS and $ERROR-DVW-INTERNAL-STATUS functions in the *Programming Reference Guide* and the *CA Datacom/DB Summary of Messages* for explanations.

CA Ideal $ERROR error functions are available only in the logical scope of <<ERROR>> procedures. For complete descriptions of CA Ideal built-in functions, see the *Programming Reference Guide*.

## Statistics and Diagnostics Area (PXX) Report

This report is a formatted dump of the CA Datacom/DB diagnostics area available for use in debugging. For more information, see the *CA Datacom/DB Utility Guide*.

CA Ideal users can trace the results of FOR statements in CBS by running the target program, and printing the PXX report, described in *CA Datacom/DB Database Administration Guide*. The command SET RUN CBSTRACE ON lets you trace the results of FOR statements in CBS during your CA Ideal session.

## CICS Service

CICS transactions available through the CICS Service Facility provide operational commands for inquiry of the CA Datacom/DB environment. For more information about CICS Service, see the *CA Datacom Option for CICS Services User Guide*.

### CA Datacom/DB Accounting Facility

This facility accumulates statistics for programs running under the Multi-User Facility. It provides statistics that can be used for tuning and problem determination.

For more information, see the *CA Datacom/DB Database Administration Guide*.

## DB2 Return Codes

You can use CA Ideal built-in functions in <<ERROR>> Procedures to identify error codes DB2 returns due to errors encountered handling dataviews or embedded SQL.

The $ERROR-TYPE function with a value of D71-D85 or DB2 indicates DB2 dataview errors. For a DB2 ERROR-TYPE, the function $ERROR-DVW-STATUS returns the SQLCODE from the SQLCA. See the *Programming Reference Guide* for explanations.

A number of $SQL functions return the current values of fields in the SQLCA for the last SQL statement executed. One identifies the last SQL statement executed.

CA Ideal $ERROR error functions are available only in the logical scope of <<ERROR>> procedures. See the *Programming Reference Guide* for complete descriptions of CA Ideal built-in functions.

The subprogram, @I$TIAR, is available to format DB2 error messages. For more information about using this utility program, see the *Programming Reference Guide*.

## VSAM Return Codes

You can use CA Ideal error functions in <<ERROR>> Procedures to identify error codes VSAM returns.

A value beginning with V from the $ERROR-TYPE function indicates a VSAM error. For a complete explanation of these functions, see the $ERROR-DVW-STATUS and $ERROR-DVW-INTERNAL-STATUS functions in the *Programming Reference Guide*.

CA Ideal $ERROR error functions are available only in the logical scope of <<ERROR>> procedures.

# CICS Debugging Aids

CICS can experience two types of abnormal ends-abends that affect one-transaction and abends that affect CICS itself.

CICS transaction and system dumps should be produced for use in the debugging process. The CSMT Log, produced when CICS is brought down, can also be useful.

The CICS Statistics Report is produced when CICS is brought down. Statistics shown include task control, storage, transactions, programs, dumps, and temporary storage. This report can be valuable in identifying what happened to CICS when a problem was encountered.

For more information, see the IBM documentation for CICS Transaction Server.

If you are serious about improving the performance of your CICS production environments, you should be running your CA Ideal programs in load module format.

Load modules provide performance benefits and an additional means of monitoring and tuning your environment. Using load modules can also reduce the impact on your system of migrating production applications.

## Monitoring

With load modules, there are many tools to aid you in gathering statistics on the performance of individual modules.

Sites which see excessive use of the U suffixed load module can identify overuse of the RELEASE command in CA Ideal. Sites can identify which modules are highly used and loaded and should possibly be made resident.

It is also important when running load modules exclusively to check your CICS statistics. You should see no I/O to the user panel or object libraries. If you do, VLS is active. You might think you are using 100 percent load modules online when you are actually running some programs in VLS format. Somehow these programs missed the conversion to load modules after they were object transported.

If this situation occurs, you can run a Datadictionary alias report. User entity type MODULE,$IP* gets a listing of all the programs and MODULE,$IM* gets a listing of all panels. These listings can be compared against a VLS library listing for missing modules.

An application load module cannot execute outside of the CA Ideal environment. For a complete description of CA Ideal load modules, see the “Application Migration Considerations” chapter of the *Administration Guide*.

## DISPLAY PCT Command

Displays the CICS Program Table entries CA Ideal uses. Enter this command in the following format:

```
DISPLAY PCT
```

The following is an example of the report this command displays:

```
TRAN  INITIAL   TYPE OF     TWA    PRD
 ID   PROGRAM   ENTRY       SIZE

scfd  SC00DISP  ACCOUNT-ID   64
ASYN  SC00NATD  ASYNC        64         WWW.ASYNC   (0001)
CWBA  DFHWBA    WEB INTF.     0         Transaction not in SCWBTRAN
DDOL  SC00INIT                12    DDO
DDOX  SC00INIT                12    DDO
DEBB  SC00INIT                64         Transaction not in SC00TRAN
DEBG  DFHWBA    WEB INTF.     64         Transaction not in SCWBTRAN
DIAL  DFHWBA    WEB INTF.     64         Transaction not in SCWBTRAN
EMPL  SC00NATD  ASYNC        64         Transaction not in SCASTRAN
HBRX  DFHWBA    WEB INTF.     0         Transaction not in SCWBTRAN
HBR1  SC00INIT  FINAL-ID      64    IDL  $ID.HBR1
HTTU  DFHWBA    WEB INTF.     0         Transaction not in SCWBTRAN
HWBA  DFHWBA    WEB INTF.     0         Transaction not in SCWBTRAN
IBIN  DFHWBA    WEB INTF.     64         WWW.WEBIMAGE(001)
IDBA  DFHWBA    WEB INTF.     64         WWW.BBSDEMOA(001)
IDEA  SC00INIT                64    IDL
IDLX  SC00INIT  FINAL-ID      64    IDL
IPCV  SC00INIT                64    IPC
IPCX  SC00INIT                64    IPC
JULC  SC00NATD  ASYNC        60         Transaction not in SCASTRAN
SAST  SC00SAST  COMP/PRINT    64
SCFD  SC00DISP  ACCOUNT-ID    64
S318  SC00NATD  ASYNC        64         Transaction not in SCASTRAN
```

## Display LMT Command

This command can display different results from the DISPLAY INDEX ALL MODULE command, which reflects only the Datadictionary MODULE entities.
The DISPLAY LMT command displays the content of the Load Module Table, which may be a composite of entries loaded from Application Module Tables, and entries retrieved from the dictionary. For more information about load module format, see the *Administration Guide*.

The following is an example of the report this command displays:

```
PGM  $ID  CICSWEB   (PRD)  IS MODULE  CICSWEB
PGM  CTH  A8842296  (PRD)  IS MODULE  ADAM1
PGM  CTH  B8842296  (PRD)  IS MODULE  ADAM2
PGM  CTH  A7698144  (PRD)  IS MODULE  A769814
PGM  CTH  B7698144  (PRD)  IS MODULE  B769814
PGM  CTH  C1198608  (PRD)  IS MODULE  CM98608
PGM  QAT  CUST      (PRD)  IS MODULE  CUST
PGM  CTH  C9370687  (PRD)  IS MODULE  C937068
PGM  CHE  DATE      (PRD)  IS MODULE  D1
PGM  SOL  ID#2192   (PRD)  IS MODULE  ID#2192
PGM  SOL  ID#2773   (PRD)  IS MODULE  ID#2773
PGM  SOL  ID#2818   (PRD)  IS MODULE  ID#2818
PGM  SOL  ID#2854   (PRD)  IS MODULE  ID#2854
```

# Chapter 2: Application Debugging Tools

This chapter describes facilities available for debugging CA Ideal PDL applications.

## DEBUG Command

This command initiates the debugging of a program. When you issue the DEBUG command, an initialization breakpoint appears on the screen that lets you specify (or alter) Debug commands. When a breakpoint is encountered, application processing stops, any commands attached to the breakpoint execute, and the Debug screen displays at the terminal. For complete syntax of debugger commands, see the *Programming Reference Guide*. For a description of use of the debugger, see the *Creating Programs Guide*.

In batch, follow the DEBUG command with Debug commands for the session and end with a GO command. When the GO command is encountered, execution of the program begins. When a breakpoint is encountered, any attached commands execute and application processing resumes. Output is written to the Debug print file.

Issuing the DEBUG command alone invokes the DEBUG prompter. DEBUG * debugs the current program.

This command has the following format:

```
      {*                          }
DEBUG {pgm-name [VERSION version]}
      {                           }

      [UPDATE [DB] {Y}]
      [            {N}]

      [{PARAMETER} 'string']
          [{MCPARM}                ]

      [            {MAIL 'email-id'      }]
      [            {LIB                  }]
```

```
[DESTINATION {SYS name             }]
[            {NET name [COPIES num]}]

[            {KEEP   }]
[DISPOSITION {RELEASE}]
[            {HOLD   }]

[MAXLINES n]

[DESCRIPTION 'string']

[         [{*                }]]
[COMMANDS [{member [USER uuu]}]]
[         [{                 }]]
```

**DEBUG *** 

Debugs the current program.

**pgm-name**

Specifies name of the program being debugged.

**Version**

Specifies version of the program being debugged.

- **nnn-**One- to three- digit version number assigned to the program.
- **PRODUCTION-**Production-status version.
- **LAST-**Latest version of the program created, that is, the version with the highest version number.

**UPDATE**

Either updates or bypasses updating the database.

- **Y-**Updates the database.
- **N-**Does not update the database.

**PARAMETER or MCPARM 'string'**

Define this parameter in the parameter definition fill-in for the program being debugged as one alphanumeric (type X) character string defined for input only. (It cannot be a group item.) A single string passed to the program.

The length of the string is limited to the space available in the prompter. When you issue the DEBUG command in the command area, the command with parameter string is limited to one command line.

**DESTINATION**

Applies to all reports this program generates unless this destination information was overridden for one or more reports with the ASSIGN REPORT command. This clause differs from the more typical DESTINATION-clause in that NAME *name* is excluded. The one- to eight-character report name becomes the output name. Select from the following:

- **MAIL 'email-id'-**Delimited 1- to 60-character name of an CA eMail+ destination.
- **LIBRARY-**Output library.
- **SYSTEM name-**System printer name.
- **NET name-**Network printer name (not available in batch).

**COPIES num**

Specifies the number of copies a system or network printer in an online environment is to print.

**Limits:** Cannot exceed site maximum.

**Note:** This clause is ignored in batch.

**DISPOSITION**

Select the disposition clause from the following:

- KEEP
- RELEASE
- HOLD

For more information about DISPOSITION, see the *Working in the Environment* guide.

**MAXLINES n**

Specifies maximum number of lines for any one report that a program can produce. The upper limit for MAXLINES is established at CA Ideal installation or by a SET OUTPUT SITE OPTIONS fill-in, and only applies to reports going to the output library. Any report reaching this maximum stops the run. This entry does not affect reports produced and printed in batch.

**DESCRIPTION 'string'**

Describes 1- to 32- character length description as an output. The description displays as part of the output status (described in Working in the Environment). The default description DEBUGGER OUT applies if the user does not provide a description in the DEBUG command.

**COMMANDS**

Specifies the name of the data member to use for debugging commands:

- *-Current member from the status line.
- **Member-**Name of the data member to use for the debugging commands. The default is DEBUG.
- **uuu-**User ID of the data member. The default is the current user.

The operands of the command when specified act as follows:

- Online-If no member name is specified, member DEBUG is used. If member DEBUG already exists, the contents of the member are saved.
- Batch-If no member name is specified, a member unique to this run is used.

In both online and batch, if a member name is specified, it is created if it does not already exist. If this member already exists, the content of the member is saved.

# LIST Statement

The CA Ideal/PDL LIST statement writes data to the output file without a report definition. For debugging, LIST is useful for:

- Displaying informative messages or the contents of fields at selected points during program execution.
- You can insert LIST statements in your program during debugging, showing the contents of significant fields or indicating which procedures executed, and remove them after any errors are corrected. For example, you could list the records retrieved by FOR statements or list the current values of variables and issue a QUIT RUN.
- Listing errors in an <<ERROR>> Procedure.

This statement has the following format:

```
LIST {list-specification}
     {ERROR             }
```

LIST writes to the CA Ideal output file. In a batch z/OS environment, this is the file with the RUNLIST DD name. In batch VSE, it is SYSLIST. Online, it is an output with an output name identical to the main program. You can browse this output or print it later.

**list-specification**

Specifies data to the list. For more information, see the Programming Reference Guide.

**ERROR**

Lists information about an error condition that occurs during a run. LIST ERROR is automatically used in the default error procedure. You can also use it in an application to handle its own error recovery. For more information, see the PDL <<ERROR>> procedure in the “Error Recovery Tools” chapter.

### Example

The following example lists records retrieved each time through a FOR construct.

```
FOR THE FIRST 5 EMPLOYEE
   WHERE NUMBER GE RLSASK.NUMBER
  .
  .
  .
  :TEST RECORDS RETRIEVED
LIST 'EMP#= ',EMPLOYEE.NUMBER,SKIP,
        'EMP NAME= ',EMPLOYEE.NAME,SKIP
ENDFOR
```

For a detailed description of the LIST statement, see the *Programming Reference Guide*.

# NOTIFY Statement

The NOTIFY statement transmits data or a message to the message line of an application panel or system panel in an online or batch environment. The message can also be sent to the operator console. The NOTIFY statement is useful for displaying a message with information about errors, instructions to continue, warnings, and so on to the user's or operator's terminal.

This statement has the following format:

```
NOTIFY list-specification [TO CONSOLE]
```

The NOTIFY message is sent from the program to the message line when the next TRANSMIT statement executes. The NOTIFY message is cleared from the message line when the next statement in the program executes after the TRANSMIT.

The TO CONSOLE clause lets the application send a message to the operator console. The message is also sent to the z/OS JESLOG or VSE POWER LOG in batch or to the CICS System Message Block online.

**Note:** The maximum length of a NOTIFY message is 79 characters. The maximum length of a NOTIFY message sent to a console is 72 characters.

For more information about NOTIFY statement, see the *Programming Reference Guide*.

# $CHARTOHEX Function

The $CHAR-TO-HEX PDL function can display non-printable fields. It returns the hexadecimal value of the specified field contents as an alphanumeric string. The length of the receiving field should be at least twice the size of the sending field to avoid truncation.

This function has the following format:

```
                {alpha-expression}
$CHAR TO HEX ( {num-field       } )
                {num-literal    }
                {flag           }
```

**alpha-expression**

Alphanumeric expression

**num-field**

Numeric field name

**num-literal**

Numeric literal

**flag**

Flag

### Example

Assume that FLD-1 is an alphanumeric field equal to 'ABCD'.

```
LIST $CHAR-TO-HEX(FLD-1)
```

Lists 'C1C2C3C4' to the output destination. If FLD-1 were defined in working data as a five-byte field, the value listed would be 'C1C2C3C440'.

For more information about $CHAR-TO-HEX function, see the *Programming Reference Guide*.

# $HEXTOCHAR Function

$HEX-TO-CHAR is an alphanumeric function that returns the display representation of the specified hexadecimal expression.

This function has the following format:

```
$HEX-TO-CHAR(alpha-expression)
```

The alphanumeric input must have an even number of valid hexadecimal characters.

### Example

Assume that a one-byte hexadecimal field is coded in the data base for each sales region. The following example sets a one-character alphanumeric working data field SALES.REGION to the non-printing hexadecimal code.

```
<SET-PROC>
SELECT
   WHEN PNL.REGION EQ 'EAST'
      SET SALES.REGION = $HEX-TO-CHAR('09')
   WHEN PNL.REGION EQ 'WEST'
      SET SALES.REGION = $HEX-TO-CHAR('0A')
ENDSEL
```

For more information about $HEX-TO-CHAR function, see the *Programming Reference Guide*.

# Compile Debugging

Based on your site and session output settings and on your use of the DESTINATION clause with CA Ideal output commands, you can direct the output of an online or a batch compilation-the compilation listing-to either the output library (for browsing online) or to a system printer. You can also direct an online compilation to a network printer.

You can use several options of the COMPILE command or the equivalent COMPILE prompter to aid in compile debugging. The *Programming Reference Guide* has the complete syntax. *Working in the Environment Guide* describes the compilation listing.

Compilation options include the following:

**IDE**

- **YES-**Includes identification information, program resources and environment sections in the compilation listing.
- **NO-**Does not include it.

**EXD**

- **YES-**Includes external data (dataview definitions, panel definitions, report definitions, and copied SQLCAs) in the compilation listing.
- **NO-**Does not include it.

**BOD**

- **YES-**Includes the body of the program (working data, parameter data, and procedure definition) in the compilation listing.
- **NO-**Does not include it.

**ADV**

- **YES-**Includes advisory messages in the compilation listing.
- **NO-**Does not include it.

**MEL**

- **YES-**Marks error lines in the original procedure definition. If SET EDIT HIGHLIGHT ERRORS is in effect, they are highlighted. They can also be found using FIND/INCL ERROR. This does not see the compilation listing as do the other options.
- **NO-**Does not mark errors.

**LSQL**

- **YES-**Lists SQL generated from any FOR constructs.
- **NO-**Does not list generated SQL.

**REF**

- **FULL-**Generates a cross reference listing.
- **SHORT-**Suppresses symbol names that are defined but not referenced.
- **NO-**Does not generate a cross reference listing.

  **Note:** You can specify this option for any batch compiles. No cross-reference listing is generated in CICS.

**PANEL**

- **FULL-**Includes all panel components in the listing.
- **SHORT-**Includes only the Identification, Summary, and Facsimile.
- **NO-**Does not include any panel components in the compilation listing.

  **Note:** This only applies if EXD is YES.

Select all of these options if you are going to communicate with CA Ideal Technical Support. However, these options do incur some overhead. Use only those needed (for example, ADV and MEL) in a normal testing environment.

### Example

```
COMPILE PGM 1 VER 2 IDE Y EXD Y BOD Y ADV Y MEL Y
```

To determine which options are on, use the DISPLAY SESSION COMPILE command. You can select any of the above options for a single compile, for a session, or for the site for the following:

- Each compile, using the COMPILE command.
- The current session, using SET COMPILE commands issued during the current session, or SET COMPILE commands executed in a CA Ideal data member.
- The site, using SET SITE commands.

# @I$SYNC Compile Debugging

The @I$SYNC keyword with the COMPILE command causes a CA Ideal compile to run as a foreground task (synchronous) rather than as a background task (asynchronous). You can use it to compile a program in synchronous mode when an asynchronous compile fails.

This keyword has the following format:

```
COMPILE name [version] [options] @I$SYNC
```

When an asynchronous compile fails, use the DISPLAY ERROR command to view the last internal error. If this does not identify the problem, re-issue the COMPILE with @I$SYNC and apply the CA Ideal Trace Facility locally to the compile. If the compile were asynchronous, a global trace is necessary.

A synchronous compile is not recommended for normal use because it locks up the terminal during the compile, bypasses the installation and systems programming controls on asynchronous tasks, and can seriously degrade system performance.

### Example

```
COMPILE DEMO1 @I$SYNC
```

For a complete description of the COMPILE command, see the *Working in the Environment Guide*.

# Chapter 3: CA Ideal Trace Facility

The CA Ideal Trace Facility is a debugging aid that traces the CA Ideal and CA IPC components executed when any command is executed in CA Ideal.

Under certain circumstances, CA Ideal Technical Support requests that you run a CA Ideal Trace. CA Technical Support uses the trace output to identify the internal processes that precede or accompany an abend, internal errors, or unexpected results from a CA Ideal program.

This chapter describes how to use the CA Ideal Trace Facility. Topics include are as follows:

- Trace Facility Functionality
- Executing a VPE service trace
- Initiating a DIAL trace
- Printing trace files
- Trace examples
- Reading trace output
- Printing an object module
- Identifying CA Ideal module versions
- JCL requirements

## Trace Facility Functionality

The CA Ideal Trace Facility provides the following functions:

**VPE service trace (@I$TRACE).**

This function records all CA Ideal service routines called and CA IPC service macros invoked during a specified part of a CA Ideal session. It generates entries for all CA Ideal online and batch services requested (except those that invoke asynchronous tasks online). It includes a facility to extract and print selected entries.

**DIAL trace (@I$DIALMASK).**

DIAL stands for Debugging Interface for Assembly Language. This facility records internal diagnostics CA Ideal modules executed as a result of a RUN, COMPILE, or CATALOG command or the Source Transport Utility produce.

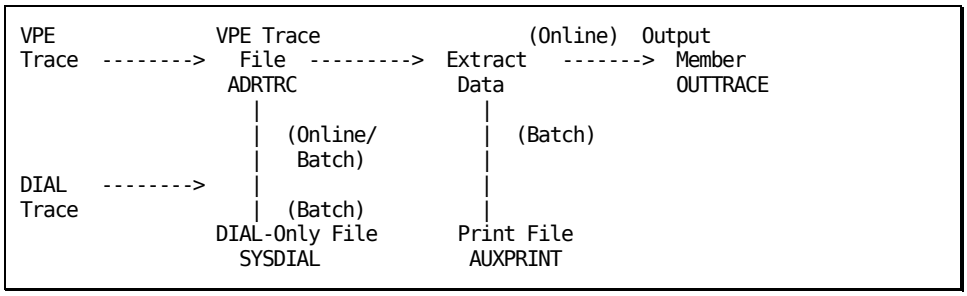
**Object Module List command (@I$CALL ADPOBJ)**

This function provides dumps of VLS object modules.

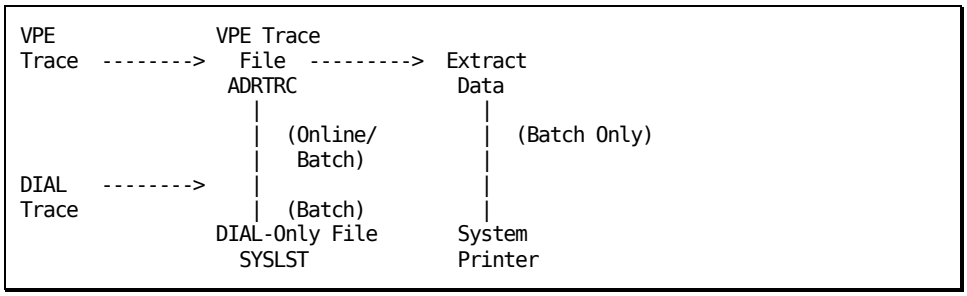
The Object Module List is described later in this chapter. The VPE and DIAL traces are described as follows.

The following graphics illustrate the data flow of the VPE and DIAL traces under z/OS and VSE. There are significant differences between the way the traces work online and in batch. These differences are described afterward.

z/OS trace facility flow is as follows:




VSE trace facility flow is as follows:



## Batch Traces

In batch, the CA Ideal traces work as follows:

- The VPE service trace writes entries to the VPE trace file (installed as ADRTRC) as the batch program makes requests. Information can then be extracted from the VPE trace file selectively (for example, by the function or by the time an error occurred).

  Under z/OS, the extracted information goes to a file (installed as AUXPRINT), which you can then print or display (as any SYSOUT file) using standard operating system spooling or print facilities.

  Under VSE, the extracted information goes to the current system printer (specified in the site-assembled CA Ideal Batch File Table (IDSYSFT) PSSPRT*nn* entries).

- The DIAL trace writes entries for specific CA Ideal internal tasks or functions to either of two places:

  - To the same trace file as the VPE trace, along with the VPE trace entries. Entries can then be extracted for printing as with the VPE trace. In this case, the VPE trace must be activated first.
  - In batch only, directly to a separate file without any VPE entries (regardless of whether the VPE trace is activated). Under z/OS, output goes to a sequential file (installed as SYSDIAL). Under VSE, output goes to a sequential file (installed as SYSLST).

## Online Traces

In an online environment, the CA Ideal traces work as follows:

- Both traces are available online. However, online, the separate DIAL-only trace output file is not supported. All information goes to the VPE trace file. The VPE trace must always be activated first.

  Online, there is only one VPE trace file for all CA Ideal users. A trace can be local to a single user or globally include all users, but all traces using a given CICS region share the same file. As in batch, information can be extracted from the VPE trace file selectively (for example, by the function or by the time an error occurred).

- Under z/OS you can print or display extracted information online using the CA Ideal output processor. Under VSE, the trace can be activated online, but trace information can be extracted only in batch.

To minimize the degradation of the online system by traces, run them in batch whenever possible.

# @I$TRACE Executing a VPE Service Trace

The VPE service traces log requests for CA Ideal online and batch services (both CA Ideal services and CA IPC services). The @I$TRACE options to activate or deactivate a VPE service trace, to extract and print data from the VPE trace file, and to reset the VPE trace file are described in the following sections.

This trace has the following format:

```
         { {ON } [LOCAL ]  }
         { {OFF} [GLOBAL]  }
@I$TRACE {                 }
         { PRINT option    }
         { RESET           }
```

## Activating or Deactivating a VPE Service Trace

The VPE service trace must be activated before any logging occurs. The service trace can be activated for an entire session or any part of a session. Entries are logged until an @I$TRACE OFF or @I$TRACE PRINT command is issued.

A DIAL trace with output directed to the VPE trace file (described in the next section) must also activate the VPE service trace first.

The following command activates or deactivates a VPE service trace.

This command has the following format:

```
         {ON } [LOCAL ]
@I$TRACE {OFF} [GLOBAL]
```

**ON**

Activates the VPE trace and begins logging entries to a sequential file (installed as ADRTRC).

**OFF**

Terminates a VPE trace.

**LOCAL (Default)**

Logs only those entries for the CA Ideal user requesting the trace. The user is identified with a particular terminal online or with a particular batch job.

**GLOBAL**

Online, logs entries for all users. Use caution with GLOBAL because the trace file can quickly reach its capacity. In batch, this option has the same effect as Local.

Online asynchronous tasks (network prints and compiles) require a global trace. Compiles can be traced locally if they are run synchronously or in batch. For more information, see the "Application Debugging Tools" chapter.

**Example**

```
@I$TRACE ON
```

Activates a VPE trace for *the current user only.*

```
@I$TRACE ON GLOBAL
```

Online only, activates a VPE trace for *all users.*

```
@I$TRACE OFF
```

Deactivates a VPE trace for *the current user.*

```
@I$TRACE OFF GLOBAL
```

Deactivates a VPE trace for *all users.*

# @I$TRACE PRINT Extracting Data from the VPE Trace File

The following command extracts entries from the VPE trace file with options to limit the information extracted. The command can also extract DIAL trace entries written to the VPE trace file (either with the VPE trace entries or alone). See the section on FUNC DIL that follows. The DIAL trace is described in the next section.

In a VSE environment, @I$TRACE PRINT command can only be issued in batch (even if the trace was run online).

This command has the following format:

```
                [USER uuu             ]
                [TIME hhmm[/hhmm]     ]
                [DATE mmdd            ]
@I$TRACE PRINT  [FUNC aaa[/aaa]...    ]
                [BLOCK nnnnn[/nnnnn]  ]
                [NAME output-name     ]
                [FILE file-name       ]
                [STATISTICS           ]
```

@I$TRACE PRINT command alone extracts *all* entries on the VPE trace file.

**USER uuu**

Limits the selection of entries in the trace file to those generated by the specified user. The value of uuu is the three-character short user ID of the user. If other users wrote to the trace file, this option lets you extract only your own entries. For example, the following selects entries for user GBS only.

```
@I$TRACE PRINT USER GBS
```

**TIME hhmm[/hhmm]**

Limits the selection of entries made in the trace file to those generated during the specified time range. The value of *hhmm* is the beginning of the time range. The range ends with another *hhmm*, if specified, or when the end of the trace file is reached. Time is in 24-hour format.

For example, the following selects all records entered in the trace file from 12:01 p.m. until the end of the trace file is reached:

```
@I$TRACE PRINT TIME 1201
```

The following, however, selects all records entered in the trace file between 1:15 am and 1:24 p.m.

```
@I$TRACE PRINT TIME 0115/1324
```

**DATE mmdd**

Limits the selection to entries made in the trace file on a specific day. The value of mmdd is a numeric value that specifies the date. If your CICS region or partition is not recycled daily (so that the trace file is not re-initialized daily), this option ensures that you select the entries for the specified day only. For example, the following selects all entries made in the trace file on April 27.

```
@I$TRACE PRINT DATE 0427
```

**FUNC aaa[/aaa]...**

Limits the selection to entries in the trace file resulting from service *aaa*. You can specify up to six of the following services. If you specify no function, all are selected.

- **ALL** All functions (default)
- **VPE** CA Virtual Processing Environment services
- **CMD** All commands (internal and external)
- **DIL** DIAL trace entries
- **DSF** Dictionary services
- **EDK** Editor Kernel services
- **ENQ** Enqueue and Dequeue requests
- **LOG** All LOG requests (for the CA Ideal error log)
- **PGM** All calls to system level subprograms

- **PMS** Panel Management System services
- **PSS** Print Sub-System services
- **SCF** Session Control Facility services
- **SVC** CA Ideal internal services
- **VLS** Virtual Library System services

For example, the following selects only VPE services.

```
@I$TRACE PRINT FUNC VPE
```

The following selects SCF, PSS, and PMS services.

```
@I$TRACE PRINT FUNC SCF/PSS/PMS
```

The following selects all functions.

```
@I$TRACE PRINT
```

You can also limit selection to all entries *except* those resulting from specified services. Use a not sign (¬) prefix.

For example, the following selects all entries except those from VPE.

```
@I$TRACE PRINT FUNC ¬VPE
```

The following selects all entries made in the trace file except those from SCF or PSS.

```
@I$TRACE PRINT FUNC ¬SCF/¬PSS
```

**BLOCK nnnnn [/nnnnn]**

Limits the selection of entries in the trace file to those in the specified range of data blocks. The value of nnnnn specifies the starting block from which entries are selected. The range ends with another nnnnn or when the end of the trace file is reached. Leading zeros are required. Data blocks on the trace file are numbered sequentially. You can find the block numbers on the Trace Listing. For example, the following selects all entries on the Trace file from block 100 to the end of the trace file.

```
@I$TRACE PRINT BLOCK 00100
```

The following selects all entries from block 5 to block 96.

```
@I$TRACE PRINT BLOCK 00005/00096
```

**NAME output-name**

Changes the name of the output member in the online output library. The default name is OUTTRACE. The output-name can be any alphanumeric name up to eight characters.

For example, the following defines MYOUTPUT as the name of the output member.

```
@I$TRACE PRINT NAME MYOUTPUT
```

**FILE file-name (Batch Only)**

Changes the name of the VPE trace file (used as input to @I$TRACE PRINT). The file-name is any valid DD or DLBL name up to eight characters in z/OS and up to seven characters in VSE. @I$TRACE PRINT first closes the default input trace file and then opens the file with the name specified in this parameter. The default trace file name is ADRTRC.

In VSE, you must update the IDSYSFT to include an entry for the new file name with the same format as the ADRT entry.

For example, the following changes the input VPE trace file name to MYDTF.

```
@I$TRACE PRINT FILE MYDTF
```

**STATISTICS**

Prints only the statistics accumulated for the sections of the trace file analyzed.

This option suppresses the printing of detailed trace entries. Instead, it summarizes the services requested and resources used during the activity that was traced. If you also use the FUNC option, only statistics that apply to the specified functions are generated. Otherwise, statistics are generated for all entries in the trace file. Other options besides FUNC are ignored. For example, the following generates statistics for the entire VPE trace file.

```
@I$TRACE PRINT STATISTICS
```

The following generates statistics for VLS entries only.

```
@I$TRACE PRINT FUNC VLS STATISTICS
```

**Note:** You can use any combination of these options in any order. For example, the following extracts all DIAL trace entries for user XYZ written on the 15th of January.

```
@I$TRACE PRINT USER XYZ DATE 0115 FUNC DIL
```

As stated above, however, if you specify STATISTICS, all other options except FUNC are ignored.

## Resetting the Online CICS VPE Trace File

The following command erases the data in the online VPE trace file and starts writing at the beginning of the file. Use it when the VPE trace file fills up (an ASRB abend occurs). In batch, if the trace file fills up, you can rerun the job with a larger trace file.

This command has the following format:

```
@I$TRACE RESET
```

You must use the following procedure with this command.

1. If you are still in CA Ideal, sign off from CA Ideal, but not off from CICS.
2. From CICS, close and reopen the VPE trace file using the CEMT CICS Operator command against the DCT. That is:

   ```
   CEMT S TDQ(ADRT) CLOSE

   CEMT S TDQ(ADRT) OPEN
   ```
3. Sign back on to CA Ideal.
4. From CA Ideal, issue @I$TRACE RESET.

# @I$DIALMASK Initiating a DIAL Trace

The following command produces internal diagnostics for a COMPILE of a CA Ideal program, a CATALOG of a CA Ideal dataview, or a RUN of a CA Ideal program. It also specifies which CA Ideal internal tasks or functions are logged. The command lets you log DIAL trace entries either to the VPE trace file or to a separate DIAL-only trace file. The previous section described how to extract entries from the VPE trace file. The next section describes how to print trace information.

This command has the following format:

```
                            {'codes'    }
SET [COMPILE] @I$DIALMASK   {'ALL'      }
    [RUN    ]               {'ONLYcodes'}
                            {'ONLYALL'  }
```

Source Transport can also process this command, but do not specify COMPILE or RUN.

**COMPILE**

Produces internal diagnostics for a COMPILE or CATALOG.

Since an online compile is normally an asynchronous task and online asynchronous tasks cannot be traced, you should produce the DIAL trace diagnostics for a compile in batch. If a batch compile is impossible, use one of the following methods to trace the compilation:

- Issue the COMPILE command with @I$SYNC. For more information about @I$SYNC, see the “Application Debugging Tools” chapter.
- Turn on the global trace (@I$TRACE ON GLOBAL) and then COMPILE the program. This can quickly fill up your trace file.

Remember that both of these methods can affect system performance for the duration of the trace.

**RUN**

Produces internal diagnostics for a RUN. You can also trace the CREATE MODULE and GENERATE PLAN or PACKAGE using the RUN option.

**codes**

The DIAL trace codes for the CA Ideal tasks or functions to log. Entries are logged with the VPE trace entries to the ADRTRC file. Commonly used codes are listed on the next page. You can find a complete list in the “DIAL Trace Codes” appendix.

You can concatenate any combination of codes. For example:

```
SET RUN @I$DIALMASK 'RSTV'
```

```
SET COMPILE @I$DIALMASK 'BLF'
```

Before using this command with this option, you must activate a VPE service trace (@I$TRACE ON). See the section titled @I$TRACE-Executing a VPE Service Trace in the “CA Ideal Trace Facility” chapter.

**ALL**

Logs all codes. For example:

```
SET RUN @I$DIALMASK 'ALL'
```

Before using this command with this option, you must activate a VPE service trace (@I$TRACE ON). See the section titled @I$TRACE-Executing a VPE Service Trace in the “CA Ideal Trace Facility” chapter.

**ONLYcodes**

The DIAL trace codes for the CA Ideal tasks or functions to log. In batch, these entries are always logged, without VPE trace entries, to SYSDIAL in z/OS or SYSLST in VSE. Commonly used codes are listed on the next page. You can find a complete list in the "DIAL Trace Codes" appendix.

You can concatenate any combination of codes. For example:

```
SET RUN @I$DIALMASK 'ONLYRSP'
```

**ONLYALL**

Logs all codes. In batch, these entries are always logged without VPE trace entries to SYSDIAL in z/OS or SYSLST in VSE. For example:

```
SET RUN @I$DIALMASK 'ONLYALL'
```

Online, ONLY has no effect. DIAL trace entries are always logged to the VPE trace file, which must first be opened with an @I$TRACE ON command.

To trace Source Transport Utility execution, use the following command syntax, omitting both the RUN and COMPILE options:

```
SET @I$DIALMASK 'ONLYcodes'
```

or

```
SET @I$DIALMASK 'ONLYALL'
```

# Common DIAL Trace Codes

This section contains DIAL trace codes for the Run, Compile, Catalog, and Source Transport commands.

## Run Codes

- Initialization
- Panel TCODES or program loads
- ASSIGN and SET ($ACCOUNT-ID/$FINAL-ID)
- CA Ideal Report Writer (PRODUCE)
- Program Symbol Table Loads
- Internal TCODES executed
- FOR constructs
- Errors or, with LIST, hex value of field

### Compile Codes

- Conditional processing
- Lexical analysis
- Data Entity Compiler
- Others
- CA Ideal Report Writer (PRODUCE)
- Group and string functions

### Catalog Codes

- Field attribute processing

### Source Transport Codes

- Command Dispatch from command table
- Export; translate to external form
- Import; translate to internal form
- Listing services
- I/O services
- Parse/verify/build command table
- Sign-on and initialization
- Tokenizing services

# Printing Trace Files

This section summarizes the information presented on outputting the results of CA Ideal traces. It describes the data flow when you print:

- VPE trace file in batch.
- Dial-only file in batch.
- VPE trace file online.

Installed default names are used here. You can change them. See the section titled JCL Requirements later in this chapter.

# Printing the VPE Trace File in Batch

In a batch environment, the @I$TRACE ON command writes VPE trace records and, if requested, DIAL trace records to the VPE trace file with the following DD/DLBL name:

```
ADRTRC
```

Under z/OS, the @I$TRACE PRINT command reads the ADRTRC file and routes the output to the file with the DD or FILEDEF name:

```
AUXPRINT
```

You can then print or display this output using a standard operating system spooling queue or printing facility.

Under VSE, the @I$TRACE PRINT command reads the ADRTRC file and routes the output to the system printer; that is, to the device address specified in the first available PSSPRT*nn* entry in the IDSYSFT.

## Printing the DIALOnly Trace File in Batch

In a batch environment, when only the DIAL trace records are written, they go directly to one of the following files.

Under z/OS, to the DD name is as follows:

```
SYSDIAL
```

Under VSE, to the DLBL name is as follows:

```
SYSLST
```

You can then print or display this output using a standard operating system spooling queue or printing facility.

## Printing a Source Transport Trace in Batch

Execution of the Source Transport Utility takes place in two phases: Command interpretation and import and export. To print the trace of the command processing only:

1. Turn on the trace.
2. Specify the commands.
3. Turn off the trace.
4. Print the trace in one batch job step.

However, to include the actual import and export:

1. Turn on the trace.
2. Specify the commands in one job step.
3. In a separate job step, print the trace.

The separate step can be either source transport or batch CA Ideal.

The DD or DLBL for ADRTRC must point to the same data set for both job steps.

## Printing the VPE Trace File Online z/OS Only

Online, the VPE trace and the DIAL trace both write entries to a data set with the DD name:

```
ADRTRC
```

To support online printing of the trace in an z/OS CICS environment, the @I$TRACE PRINT command reads the trace records from the same data set that the ddname ADRTRC points to, but using the DD name:

```
ADRINT
```

For CA Ideal running under z/OS, ADRTRC and ADRINT should be defined to your system as provided in the installation copybooks VQ14FCT and VQ14CSD. The z/OS installation process is described in the CA IPC r14 z*/OS Installation and Maintenance Guide*. The ADRTRC and ADRINT files should not be defined as temporary data sets.

The @I$TRACE PRINT command writes the output to a member in the CA Ideal online output library with the name:

```
OUTTRACE
```

You can then print or display this output using the standard CA Ideal OUTPUT commands as documented in the *Working in the Environment Guide*.

In a VSE environment, although the trace can be run online, it can only be extracted using batch CA Ideal. To do so, there must be a DLBL in the batch JCL that points to the same data set that the DLBL for ADRTRC points to in your online CICS start-up JCL. See the section titled JCL Requirements later in this chapter.

# Trace Examples

The following examples illustrate how to run VPE service traces and DIAL traces under z/OS and VSE, in both batch and online environments. For a description of the JCL required, see the section titled JCL Requirements later in this chapter.

| **z/OS** | |
|---|---|
| **Batch** | **Online** |
| **DIAL Trace of a RUN**<br>**(To DIAL-Only File)**<br>PERSON *uuu* PSW *ppp*<br>SEL SYS *sss*<br>SET RUN FILETABLE DBFLT*nnn*<br>SET RUN @I$DAILMASK 'ONLYALL'<br>RUN *pgmname*<br>OFF | **DIAL Trace of a RUN**<br>**(To VPE Trace File)**<br>SEL SYS *sss*<br>@I$TRACE ON LOCAL<br>SET RUN @I$DIALMASK 'ALL'<br>RUN *pgmname*<br>@I$TRACE OFF LOCAL<br>@I$TRACE PRINT FUNC DIL<br>PRINT OUT OUTTRACE DEST SYS *xxx* |
| **DIAL Trace of a CATALOG**<br>**(To VPE Trace File)**<br>PERSON *uuu* PSW *ppp*<br>SEL SYS *sss*<br>@I$TRACE ON LOCAL<br>SET COMPILE @I$DIALMASK 'F'<br>CATALOG DVW *name* VER *nnn*<br>@I$TRACE PRINT FUNC DIL<br>OFF | **DIAL Trace of a CATALOG**<br>**(To VPE Trace File)**<br>@I$TRACE ON LOCAL<br>SET COMPILE @I$DIALMASK 'F'<br>CATALOG DATAVIEW *name* VER *nnn*<br>@I$TRACE OFF LOCAL<br>@I$TRACE PRINT FUNC DIL<br>PRINT OUT OUTTRACE DEST SYS *xxx* |
| **VPE Service Trace**<br>**(To VPE Trace File)**<br>PERSON *uuu* PSW *ppp*<br>SEL SYS *sss*<br>@I$TRACE ON LOCAL<br>...<br>*one or more CA Ideal commands*<br>...<br>SCF/VPE<br>@I$TRACE PRINT FUNC SCF/VPE<br>OFF | **VPE Service Trace**<br>**(To VPE Trace File)**<br>@I$TRACE ON LOCAL<br>...<br>*one or more CA Ideal commands*<br>...<br>@I$TRACE OFF LOCAL<br>@I$TRACE PRINT USER *uuu* FUNC<br>PRINT OUT OUTTRACE DEST SYS *xxx* |

| VSE | |
|---|---|
| **Batch** | **Online** |
| **DIAL Trace of a RUN (To DIAL-Only File)**<br>PERSON *uuu* PSW *ppp*<br>SEL SYS *sss*<br>SET RUN FILETABLE DBFLT*nnn*<br>SET RUN @I$DIALMASK 'ONLYALL'<br>RUN *pgmname*<br>OFF | **DIAL Trace of a RUN (To VPE Trace File)**<br>SEL SYS *sss*<br>@I$TRACE ON LOCAL<br>SET RUN @I$DIALMASK 'ALL'<br>RUN *pgmname*<br>@I$TRACE OFF LOCAL |
| **DIAL Trace of a CATALOG (To VPE Trace File)**<br>PERSON *uuu* PSW *ppp*<br>SEL SYS *sss*<br>@I$TRACE ON LOCAL<br>SET COMPILE @I$DIALMASK 'F'<br>CATALOG DVW *name* VER *nnn*<br>@I$TRACE OFF LOCAL<br>@I$TRACE PRINT FUNC DIL<br>OFF | **DIAL Trace of a CATALOG (To VPE Trace File)**<br>@I$TRACE ON LOCAL<br>SET COMPILE @I$DIALMASK 'F'<br>CATALOG DATAVIEW *name* VER *nnn*<br>@I$TRACE OFF LOCAL |
| **VPE Service Trace (To VPE Trace File)**<br>PERSON *uuu* PSW *ppp*<br>SEL SYS *sss*<br>@I$TRACE ON LOCAL<br>...<br>*one or more CA Ideal commands*<br>...<br>@I$TRACE OFF<br>@I$TRACE PRINT FUNC SCF/VPE<br>OFF | **VPE Service Trace (To VPE Trace File)**<br>@I$TRACE ON LOCAL<br>...<br>*one or more CA Ideal commands*<br>...<br>@I$TRACE OFF LOCAL |
| **Print VPE Trace File (From Online Trace)**<br>PERSON *uuu* PSW *ppp*<br>@I$TRACE PRINT<br>OFF | |

# Reading Trace Output

This section describes the VPE trace output generated by the @I$TRACE PRINT command. The output has three parts:

- Environment chapter
- Trace listing
- Statistics

**Note:** The Trace Listing is not printed if the STATISTICS option is used with @I$TRACE PRINT.

Each page of the listing has a heading showing the following system information:

- ADPLOG program (@I$TRACE PRINT Command) version number
- CA Ideal version number
- Date and time of the printout
- User name

The heading also shows the following selection criteria:

- Date selected (*mmdd* if none)
- Time selected (*hhmm*/*hhmm* if none)
- User selected (*xxx* if none)
- Functions selected/rejected (ALL/*xxx*/*xxx* if none)
- Blocks selected (*nnnnn*/*nnnnn* if none)

## Environment Section

The environment page is always first. It shows the following information:

- The @I$TRACE PRINT command (generated by the ADPLOG program) as entered, including the requested options
- A description of the trace print (ADPLOG) options
- The monitor name and operating system
- The version of each major software component

An environment page sample trace is shown as follows:

```
ADPLOG   Ver: 11.0   IDEAL 11.0 print trace facility.  Date: 04/12/06 Time: 11:
 Selection criteria:  Date mmdd   Time hhmm/hhmm   User xxx   Func  ALL/ xxx/ xx
Command entered : @I$TRACE PRINT
The primary function of this module is to print IDEAL trace.
This program provides a group of commands designed to select
information from the trace file.
The commands are as follows:
- DATE  mmdd         to select a specific day in the month
- TIME  hhmm         to select from a given time
- TIME  hhmm/hhmm    to select a range of time
- BLOCK nnnnn        to select from a specific block
- BLOCK nnnnn/nnnnn  to select a range of blocks
- USER  uuu          to select a specific user-id
- FUNC  fff/fff/...  to select up to six functions
- FUNC  ^fff/^fff/.. to exclude up to six functions
- FILE  iiiiiiii     to change the name of the input file
- NAME  oooooooo     to change the name of the print file
- STATISTICS         to print only the statistics
The values for "fff" are the following:
- VPE, VLS, PMS, PSS, SCF, DIL, LOG,
- CMD, SVC, DSF, ENQ, PGM, EDK, BAS, SQI
The following software was used:
Monitor=CICS      Opsys=MVS/JES2
VPE(11.0) SCF(1100) PSS(1100) PMS(110 ) EDK(110 ) PDF(110 ) DSF(11.0) IDL(110 )
 ADPLOG   Ver: 11.0   IDEAL 11.0 print trace facility.  Date: 04/12/06 Time: 11:
```

## Trace Listing

DIAL trace entries can be interspersed with VPE trace entries in the trace listing or they can be separated. DIAL entries are free format. VPE trace entries display the following columns of information:

```
Time...usr r trace-entry......cmp function...rc..aux-data...
```

**time**

Time the entry was logged

**usr**

Three-character short user ID

**r**

Region number (1, 2, or 3) or *0 for asynchronous task

**trace-entry**

VPE trace entry in hex format

**cmp**

Component name

**function**

Function name

**rc**

Component return code

**aux-data**

Auxiliary information

For each entry, the hex trace data, time, user, and region come from the VPE trace file. @I$TRACE PRINT extracts the component, function, return code, and auxiliary data in symbolic form from the hex trace entry.

The beginning of each data block in the trace file is marked on the trace listing with a row of asterisks, with the block number, date, and time of the entry shown as follows:

```
ADPLOG  Ver: 11.0  IDEAL 11.0 print trace facility.  Date: 04/12/06 Time: 12:07:54
User: WESJO01
Selection criteria:  Date mmdd   Time hhmm/hhmm   User xxx   Func  ALL/ xxx/ xxx/
xxx/ xxx/ xxx   Blocks nnnnn/nnnnn
time.... usr r trace-entry 1st 16 bytes........ cmp function rc....
aux-data.....................
 ****************************************************************
block=0000001  Date=04/11/06 Time=15:49:06
15:49:06 WJW 1 06000000000A984E0000000000000000 VPE $EXT-n   00     pgmn=@IADTRCE
15:49:06 WJW 1 06000000000A984E0000000000000000 VPE $EXT-n   00     pgmn=@IADCMD2
15:49:06 WJW 1 06000000000A984E0000000000000000 VPE $EXT-n   00     pgmn=@IADCMDP
15:49:06 WJW 1 020000000018E1100000000000109C50 VPE $CSP-n   00     pgmn=@IADCMDP
parm=00109C50
15:49:06 WJW 1 AA0000000005C3000000001700109B19 IDL CMD-C    00     CAT DVW
SYSADM.DIR_AREA
15:49:06 WJW 1 020000000019056C000000000010A4E8 VPE $CSP-n   00     pgmn=SC00EACP
parm=0010A4E8
15:49:06 WJW 1 06000000000A984E0000000000000000 VPE $EXT-n   00     pgmn=SC00EACP
15:49:06 WJW 1 020000000019071400000000000010A4E8 VPE $CSP-n   00     pgmn=@IADENDP
parm=0010A4E8
15:49:06 WJW 1 AA00000000020000000000E0016B258 IDL entered  00     pgmn=ADSVCS
15:49:06 WJW 1 501068900001E201D30000340010930C SCF CVT-loc  00    CB name=ADMAINCB
15:49:06 WJW 1 AA1ADB6A00000000D3D6C3C30010E4E4 SVC LOCC     00     ADMAINCB
15:49:06 WJW 1 AA00000000020000000000D0016B5C8 IDL exited   00     pgmn=ADSVCS
15:49:06 WJW 1 02000000001AD8A2000000000010E5C0 VPE $CSP-a   00     pgma=00000000
parm=0010E5C0
```

## Statistics

The last two pages contain statistics summarizing the contents of the trace file. If only statistics were requested, these are the only pages printed. See the section titled @I$TRACE PRINT Function Statistics earlier in this chapter. The statistics section is divided into two parts and is shown as follows:

- The Service Requests Summary shows the number of entries for major components accessed by the traced activity.
- The Resource Utilization Summary shows the number of entries for major resources used by the traced activity, including:
  - File tables used (Filetab).
  - VLS file I/O (File ID) and sequential file I/O (where the file ID is the CICS DCT name, for example, ADRL for the CA Ideal error log, DD/DLBL name of ADRLOG).
  - VPE control blocks (VCB) used.
  - Program loads and releases ($LDM and $RLM).

- Subroutine calls ($CSP-*n* and $CSP-*a*) and program branches (Branch).

```
  SERVICE REQUEST SUMMARY
VPE-total..0003099  $AGS.......0000001  $APS.......0000004  $CSP-a.....0000086
                    $DEQ.......0000017  $DT........0000052  $ENQ.......0000018
                    $FPS.......0000004  $FS........0000174  $GS........0000157
                    $LDM.......0000094  $LGS.......0000187  $LPS.......0000087
                    $WBL.......0000031  $GSS.......0000006  $LSS.......0000009
VLS-total..0000092  ADD........0000020  AMP........0000002  BDREAD.....0000008
                    DELMEM.....0000010  INIT.......0000010  OPEN.......0000004
                    RMP........0000004
SCF-total..0000026  CVT-add....0000001  ENDTRAN....0000001  CVT-loc....0000018
IDL-total..0000277  DSF........0000127  Entered....0000134  Exited.....0000132
ADSVCS.....0000046  ALCC.......0000001  ATZ........0000001  CLVB.......0000001
                    INIT.......0000001  LOCC.......0000008  OPVB.......0000001
                    WTVB.......0000026
PMS-total..0000007  SPA........0000001  ICP........0000001  DCP........0000001
PSS-total..0000001  unknown....0000001
EDK-total..0000022  1#OPEN  ...0000002  1#FILBUF...0000002  1#LOCBUF...0000004
                    5#BUFILL...0000002  7#INILIB...0000002  7#OPNMEM...0000002
ADPLOG   Ver: 11.0   IDEAL 11.0 print trace facility.  Date: 04/12/06 Time: 12:0
Selection criteria:  Date mmdd   Time hhmm/hhmm   User xxx   Func  ALL/ xxx/ xxx
RESOURCE UTILIZATION SUMMARY
Filetab             DDURT002...0000306
Fileid              ADRPNL.....0000002  IDDVW......0000066  IDDAT......0000020
VCB                 00.........0000010  02.........0000080  01.........0000002
$LDM                DDSRTLM....0000080  @IADLBUN...0000004  @IADDDSR...0000001
                    VTPM9891...0000004  @IADOMSF...0000001  @IADDVWB...0000001
$RLM                @IADDBQI...0000001  @IADOMSF...0000001  @IADDDSR...0000001
$CSP-n              @IADCMDP...0000004  SC00EACP...0000003  @IADENDP...0000001
                    @IADDDGT...0000033  DDOLIUSP...0000040  DDSPMLM....0000092
                    DDRTVLM....0000038  DDCDBLM....0000131  DDCAMLM....0000040
                    DDUPNLM....0000007  DDRLTLM....0000001  @ICMODEO...0000001
                    DDRARLM....0000010  @ICMCDEC...0000001  @IADOMSV...0000001
                    DDENTLM....0000001  @IADDDSP...0000001  @IADOMLD...0000001
                    EDEDITOR...0000004  EDEDMEMB...0000001  EDED$VLS...0000004
                    @IADGTCP...0000001  PMSRGS1....0000001  @IADTREN...0000001
                    @IADPLOG...0000001
$CSP-a              00000000...0000086
Branch              ADSVCS.....0000046  ADDDSR.....0000022  ADDBQI.....0000006
                    ADTREN.....0000001  ADTRIN.....0000001
```

# @I$CALL ADPOBJPrinting an Object Module

The following command prints the contents of a VLS object module, formatted or unformatted, in batch.

This command has the following format:

```
@I$CALL ADPOBJ entity-type entity-name VER nnn [DUMP  ]
                                               [DELETE]
                                               [HDR   ]
```

**entity-type**

Valid entity types are as follows:

- **DVW-**Cataloged dataview
- **PNL-**Panel (that is a resource of a compiled program). This is the panel object, not the member in the panel library
- **PGM-**Object module (of a compiled program)
- **PAR-**Parameter definition (of a compiled program)
- **WOR-**Working definition (of a compiled program)
- **SYM-**Symbol table (of a compiled program). You must specify DUMP.

**Note:** This command recognizes only these spellings.

**entity-name**

Specifies the name of the entity occurrence.

**VER nnn**

Version number of the entity occurrence. PROD is only a valid version for PGM and SYM. You must specify the actual version number for all other entity types.

**DUMP**

Prints an unformatted hex dump of the object module. The default is a formatted dump.

**DELETE**

Deletes the incremental VLS object module created by data entity compiler for entity types DVW, PNL, PAR, and WOR.

**HDR**

Prints only the formatted object module heading.

This command generates output in a manner similar to the DIAL trace. That is, it writes to a separate output file in batch or to the VPE trace file online.

## Batch JCL Statements

@I$CALL ADPOBJ command generates output to one of the following files.

Under z/OS to the DD or FILEDEF name is as follows:

```
SYSDIAL
```

Under VSE, to the DLBL name is as follows:

```
SYSLST
```

You can print or display these files using standard system facilities.

### Example in z/OS batch

Output to SYSDIAL

```
PERSON uuu PSW ppp
SEL SYS sss
@I$CALL ADPOBJ PGM DEMO VER 1
OFF
```

### Example in VSE batch

Output to SYSLST

```
PERSON uuu PSW ppp
SEL SYS sss
@I$CALL ADPOBJ PGM DEMO VER 1
OFF
```

## Program and Panel Object Output

The formatted object module printout displays the logical blocks that comprise an object module. CA Ideal Technical Support can use this information in the debugging process.

The formatted object printout for a CA Ideal panel object also displays information such as the panel name, date and time last edited, and run status.

The formatted object module printout for a CA Ideal program displays data that can also aid CA Ideal users. You can find this information in the printout header and in the Dataview Control Block (DVC) entries shown as follows:

```
F O R M A T T E D    O B J E C T    M O D U L E    P R I N T O U T
 = = = = = = = = =    = = = = = =    = = = = = =    = = = = = = = =

 PROGRAM: DEMOPGM2     VERSION: 001    SYSTEM: $ID

 RUN-STATUS: SHARED   DBCS OPTION: N

 DATE/TIME  LAST COMPILED : 07/23/94 16:14:52 COMPILED UNDER IDEAL 02.1

 # OF MEMBERS ASSOCD WITH MODULE : 02

 # OF READ-ONLY MEMBERS ASSOCD WITH MODULE : 01

 # OF BLOCKS CONTAINED IN MODULE : 0027
```

The printout heading contains the following information about the program:

- Program name
- Program version
- CA Ideal system name
- Program Run-Status-private, shared, or resident
- Double-Byte Character Set Option (Y or N). This option supports alternate languages such as Kanji
- Date and time when the object module was last compiled
- CA Ideal release under which the module was last compiled (blank if pre-1.2)
- Total number of members (physical blocks) associated with the module
- Number of read-only members associated with the module
- Number of logical blocks displayed, in decimal (blocks are numbered in hex in the listing)

The following graphic shows Dataview Control Block (DVC) entry for the dataview in the resource table:

```
===> DVC HEADER   BLK  #  =  0013 (HEX)
         D8D8D8C4E5C30710005A001300000000
===> DVC CONTRL BLK
         DATAVIEW NAME= EMPLOYEE             DVW VER # = 001       DVW TYPE=  D
         DATABASE ID  = 001                  DB FILE NAME =  PMF
         INIT-AREA OFF= 0000                INIT-AREA LEN= 0000   DVW STATUS= 0000
         DVW FLAG= C (C-CLOSED,O-OPEN_OUT)   DVW POSITION= X
         ELEMENT LIST :
           ELM NAME - EMDTA    ELM SC - 00
         FILE AREA :
 03533C   004E0000 00000000 00D7D4C6 00000000   00000000 0001          * +     PMF
```

The printout contains one Dataview Control Block (DVC) entry for each dataview in the program's resource table. The following data, found in each entry, is often useful.

- Dataview name
- Dataview version number
- Dataview type, either D (CA Datacom/DB) or S (sequential file)
- Database ID and file name associated with the dataview
- Element list comprising the dataview

# JCL Requirements

This section shows the JCL, IDSYSFT, and CICS table entries that the CA Ideal trace facility requires. All of the statements and entries required to run the traces are provided as part of the installation process. You can modify them, but if you change them, they must still include all required entries.

## z/OS Batch Job Stream

A z/OS job stream to trace a program includes:

```
//jobname JOB acctinfo,'username',MSGLEVEL=1
//        EXEC IDBATCH
//SYSIN DD *
 .  .  .
CA Ideal trace commands and CA Ideal commands
 .  .  .
```

**//jobname JOB**

A standard JOB card containing information about the job for the operating system.

**// EXEC IDBATCH**

Identifies a JCL procedure containing the CA Ideal JCL procedure and the dataview records necessary for a batch session. IDLBTCH is the name of the PROC as installed. See your CA Ideal Administrator for a possible different name.

**//SYSIN DD ***

Indicates that the CA Ideal commands to executed follow. The CA Ideal commands are entered in a sequence that simulates an online session (for example, a SIGNON command is the initial command that simulates the signon screen and OFF is the final command).

## VSE Batch Job Stream

The following standard JCL records are used in a VSE environment to execute CA Ideal traces in batch.

```
* $$ JOB JNM=IDBATCH,PRI=n,USER='username',DISP=D
* $$ LST DISP=D,CLASS=L
// JOB IDBATCH IDEAL
// OPTION LOG,NODUMP
// EXEC PROC=IDLPROC
// EXEC IDBATCH,SIZE=15K
.  .  .
CA Ideal trace commands and CA Ideal commands here
.  .  .
/*
// EXEC LISTLOG
/*
/&
* $$ EOJ
```

**$$ JOB JNM=IDBATCH, PRI=n,USER='username', CLASS=c,DISP=D**

Identifies job and user, and sets the job's priority, class, and disposition.

**JNM=IDBATCH**

This jobstream is identified to POWER as 'IDBATCH'

**PRI=n**

Specifies the dispatching priority of this job is n.

**USER='username'**

Specifies the following the user name for accounting purposes:

**CLASS=c**

Defines which VSE partition this jobstream can execute in.

**DISP=D**

Defines the disposition in the reader spool for the jobstream. D indicates that the jobstream should be submitted for execution immediately and not be retained in the reader queue (spool) after execution.

**$$ LST DISP=D,CLASS=L**

Defines the output parameters for the VSE system log file.

**DISP=D**

Defines the disposition of the output in the POWER list queue. As in the DISP option for the JOB command, D indicates that the jobstream should be released immediately and not retained in the queue.

**CLASS=L**

Organizes the LIST queue. It is used with subsequent * $$ LST cards to logically separate the various CA Ideal outputs.

**// JOB IDBATCH IDEAL**

This is the first VSE job control command of the jobstream. The jobstream is identified as IDBATCH. All references to the JOB on the operator's console are to IDBATCH (not the POWER jobname).

**// EXEC PROC=IDLPROC**

This statement copies a JCL procedure into the jobstream. IDLPROC is an installed procedure to define all CA Ideal and user system files for batch.

**// EXEC IDBATCH,SIZE=15K**

IDBATCH is the CA Ideal batch processing program. The size parameter sets the amount of storage allocated to the program for execution. The rest of the partition is used for processing other CA Ideal routines. The statements that follow are CA Ideal commands, including trace commands. They are entered in a sequence that simulates an online session (for example, a SIGNON command is the initial command that simulates the signon screen and OFF is the final command).

**/*** JOB

Step delimiter.

**// EXEC LISTLOG**

LISTLOG is a VSE utility that adds all operator console messages associated with the jobstream to the end of the output for the jobstream.

**/&**

VSE job delimiter.

**$$ EOJ**

POWER job delimiter.

## z/OS Batch Procedure

The following DD statements, installed in the IDLBATCH procedure, are required to run the traces in batch.

```
//ADRTRC DD DSN=&&ADRTRC,DISP=(MOD,PASS),
//    UNIT=SYSDA,SPACE=(TRK,(15,15)),
//    DCB=BLKSIZE=2000
//SYSDIAL DD SYSOUT=*   For DIAL-Only and ADPOBJ output.
//AUXPRINT DD SYSOUT=*  For VPE Trace file output
```

Defining ADRTRC as a temporary data set is suggested. If, however, the trace runs online but prints with CA Ideal Batch, the ADRTRC DD name *must* point to the same DSN as in the CICS start-up JCL. Do not change the block size for the ADRTRC file.

The following ROSFD entries, installed in the IDSYSFT (CA Ideal System File Table) are required to run traces in batch.

```
SYSDIAL  ROSFD DDNAME=SYSDIAL,ACCMETH=SEQ,RECFM=FBA,                  X
       LRECL=133,BLKSIZE=1330,PRODUCT=IGN
AUXPRINT ROSFD DDNAME=AUXPRINT,ACCMETH=SEQ,RECFM=FBM,                 X
       LRECL=133,BLKSIZE=1330,PRODUCT=IGN
ADRT     ROSFD DDNAME=ADRTRC,ACCMETH=SEQ,RECFM=FB,                    X
       LRECL=2000,BLKSIZE=2000,PRODUCT=IGN
```

**Note:** The IDSYSFT must be reassembled if any entries are added or changed. For more information, see the *Installation Guide*.

## VSE Batch Procedure

The following JCL, installed in the IDLPROC procedure, is required to run the traces in batch.

```
// DLBL ADRTRC,'file-id'
// EXTENT SYSnnn,VOLID,,,beg-track,#-of-tracks
// ASSGN SYSnnn, DISK,VOL=xxxxxx,SHR
// ASSGN SYSLST,PRINTER   For DIAL-Only and ADPOBJ output
// ASSGN SYSaaa,PRINTER CA Ideal REPORT FORM #01
// ASSGN SYSbbb,PRINTER CA Ideal REPORT FORM #02
// ASSGN SYSccc,PRINTER CA Ideal REPORT FORM #03
// ASSGN SYSddd,PRINTER CA Ideal REPORT FORM #04
// ASSGN SYSeee,PRINTER CA Ideal REPORT FORM #05
```

SYSLST is used for DIAL-Only and ADPOBJ output.

The SYS numbers assigned to the printers (SYS*aaa-eee*) depend on the values assigned by the site at installation. System printers are used for VPE trace file output. Output goes to the device specified by the first available PSSPRT*nn* entry encountered in the IDSYSFT.

The following ROSFD entries installed in the IDSYSFT (CA Ideal System File Table) are required to run traces in batch:

```
SYSDIAL         ROSFD ACCMETH=SEQ,DTFTYPE=DTFPR,                X
                DEVADDR=SYSLST,DEVICE=1403,                     X
                CTLCHR=ASA,RECFM=F,                             X
                BLKSIZE=133,OPSYS=DOS
ADRT            ROSFD ACCMETH=SEQ,DTFTYPE=DTFSD,                X
                DEVADDR=SYSnnn,DEVICE=3340,                     X
                DTFNAME=ADRTRC,RECFM=FB,                        X
                IBLKSZ=2000,OBLKSZ=2008,                        X
                LRECL=2000,OPSYS=DOS
PSSPRT01        ROSFD ACCMETH=SEQ,DTFTYPE=DTFPR,                X
                DEVADDR=SYSaaa,DEVICE=1403,                     X
                CTLCHR=YES,RECFM=F,                             X
                BLKSIZE=133,OPSYS=DOS
PSSPRT02        ROSFD ACCMETH=SEQ,DTFTYPE=DTFPR,                X
                DEVADDR=SYSbbb,DEVICE=1403,                     X
                CTLCHR=YES,RECFM=F,                             X
                BLKSIZE=133,OPSYS=DOS
PSSPRT03        ROSFD ACCMETH=SEQ,DTFTYPE=DTFPR,                X
                DEVADDR=SYSccc,DEVICE=1403,                     X
                CTLCHR=YES,RECFM=F,                             X
                BLKSIZE=133,OPSYS=DOS
PSSPRT04        ROSFD ACCMETH=SEQ,DTFTYPE=DTFPR,                X
                DEVADDR=SYSddd,DEVICE=1403,                     X
                CTLCHR=YES,RECFM=F,                             X
                BLKSIZE=133,OPSYS=DOS
PSSPRT05        ROSFD ACCMETH=SEQ,DTFTYPE=DTFPR,                X
                DEVADDR=SYSeee,DEVICE=1403,                     X
                CTLCHR=YES,RECFM=F,                             X
                BLKSIZE=133,OPSYS=DOS
```

Do not change the block size for the ADRTRC file. The IDSYSFT must be updated and reassembled if any entries are added or changed. See the *Installation Guide*.

## z/OS CICS

The following DD statements are required as part of the CICS start-up JCL to run CA Ideal traces online.

```
//ADRTRC DD DSN=index.Ideal.TRACE,DISP=SHR              For Writing
//ADRINT DD DSN=index.Ideal.TRACE,DISP=SHR              For Reading
```

The following CICS DFHFCT entry is required:

```
DFHFCT TYPE=DATASET,            X
     DATASET=ADRINT,            X
     ACCMETH=BDAM,              X
     SERVREQ=GET,               X
     RELTYPE=BLK,               X
     LRECL=2000,                X
     BLKSIZE=2000,              X
     RECFORM=(FIXED,UNBLOCKED), X
     OPEN=INITIAL
```

**Note:** An FCT entry is *not* needed for ADRTRC.

The TDQueue ADRT must be defined.

```
CEDA  View TDqueue( ADRT )
TDqueue        : ADRT
 Group         : SIB0GRP
 DEscription   : VPE TRACE
 TYPE          : Extra             Extra | INTra | INDirect
EXTRA PARTITION PARAMETERS
 DAtabuffers   : 001               1-255
 DDname        : ADRTRC
 DSname        :
Sysoutclass    :
Erroroption    : Ignore            Ignore | Skip
 Opentime      : Initial           Initial | Deferred
 REWind        :                   Leave | Reread
 TYPEFile      : Output            Input | Output | Rdback
 RECORDSize    : 02000             0-32767
 BLOCKSize     :                   0-32767
 RECORDFormat  : Fixed             Fixed | Variable
 BLOCKFormat   : Unblocked         Blocked | Unblocked
 Printcontrol  :                   A | M
 DIsposition   : Shr               Shr | Old | Mod
INTRA PARTITION PARAMETERS
 Atifacility   :                   Terminal | File | System
 RECOVstatus   :                   No | Physical | Logical
 Facilityid    :
TRAnsid        :
TRIggerlevel   :                   0-32767
 Userid        :
INDOUBT ATTRIBUTES
 WAIT          :                   No | Yes
 WAITAction    :                   Queue | Reject
INDIRECT PARAMETERS
 Indirectname  :
REMOTE PARAMETERS
 REMOTEName    :
REMOTESystem   :
REMOTELength   :                   0-32767
```

**Note:** A TDQueue definition is *not* needed for ADRINT. Do not change the block size for the ADRTRC file.

## VSE CICS

In the VSE environment, the VPE trace file that CICS online uses can remain closed even after CICS is brought up. In this case, it must be explicitly opened and enabled with the CICS Master Terminal Operator command:

```
CEMT SET QUEUE(ADRT) OPEN ENABLE
```

Although you can create a trace online, it can be extracted using Batch CA Ideal only. Online, the DLBL file name must be ADRTRC. The file ID, EXTENT, and ASSGN must match the corresponding entries for the batch DLBL definition of ADRTRC. That is:

```
// DLBL ADRTRC,'file-id'
// EXTENT SYSnnn,VOLID,,,beg-track,#-of-tracks
// ASSGN SYSnnn,DISK,VOL=xxxxxx,SHR
```

The TDqueue definition for VSE is the same as that for z/OS mentioned in the previous section.

# Chapter 4: Diagnostic Tools

This chapter describes facilities available for accessing CA Ideal system information about internal errors, installed modules and solutions, VLS library indexes, and the output library.

## Internal Error Facility

CA Ideal provides an internal error logging and reporting facility. This facility traps internal system errors, not necessarily user errors. Any time an illogical or unexpected return code is received from an internal service in CA Ideal, the Internal Error Facility automatically gets control.

For example, if you try to delete a VLS member that does not exist, a return code indicating the problem is a reasonable condition and a simple message is issued. However, if the return code indicates “library cannot be accessed,” this is an unexpected condition, probably due to an error in generating or restoring the CA Ideal system. In the second case, an internal error message is issued.

Such internal error messages begin with the keyword INTERR.

If the message contains the keyword INTERR, the Internal Error Facility performed two actions: It wrote a hardcopy record of the internal error to the ADRLOG Log File (printed when the teleprocessing monitor is shut down or when a batch job ends) and, for an online session, it saved information about the error that you can display on the terminal by entering the following command.

This facility has the following format:

```
DISPLAY ERROR
```

The minimum abbreviation is D ERR.

If you plan to call CA Technical Support about the error, copy the information from your terminal, from a PRINT SCREEN printout, or from ADRLOG. Save the PRINT SCREEN or ADRLOG printout in case you are requested to send it to CA.

**Note:** Online, you can enter the DISPLAY ERROR command at any time. Information about the most recent internal error (if any) displays.

The following is an example of DISPLAY ERROR information.

```
1-IDADERRP18E - INTERR: LIB cannot be accessed, LIB=ID$IDOBJ

SRVC=VLS FUNC=INIT     RC=008    PGM=ADOMLD  11.0 -0E7C
CALPGM=                   CURACT=AE ACTTYP=U USER=COL ERRID=AELPGM-L20
SYS=$ID ENTTYP=PGM ENTNAM=SCROLL             ENTVER=001  ENTSTAT=TEST
LIB=ID$IDOBJ MEM=
```

Following is a description of the various fields that you can display on Internal Error panels. The Internal Error key consists of the Service (SRVC) detecting the error, the internal function (FUNC) performed by that service, and the return code (RC). Only relevant fields display on a given Internal Error panel.

**SRVC**

Internal service detecting the error. Currently, the internal services are:

- **VPE** CA Virtual Processing Environment
- **CMP** Compiler
- **DSF** Dictionary service facility
- **LBN** Linked Bundle (Storage Management)
- **LOG** General Log Message
- **PMS** Panel Management Service
- **PSS** Print Subsystem
- **SCF** Session Control Facility
- **VLS** Virtual Library System

**FUNC**

Specific function performed by the service. For example, VLS services include INIT (initialize), READ, WRITE, and so on.

**RC**

Return code from the service.

**PGM**

Identifier of the internal program in control when the error was detected. This includes the six-character program name, the CA Ideal version when the program was last compiled, the offset of the instruction detecting the error, and the date and time when the program was last compiled.

**CALPGM**

Identifier of the calling program, in cases where the program actually detecting the error is a general subroutine called from several other places. This includes the calling program name, the CA Ideal version, and the offset of the call instruction.

**CURRACT**

Current activity. Codes are as follows:

- **AD** General service commands
- **AE** Application execution
- **AP** Application Plans Editor
- **CM** Compiler
- **DE** Working-data/parameter editing
- **DV** Unmodeled sequential file dataview
- **ED** General editing
- **PD** Panel Definition Facility
- **PM** Panel Management Services
- **PS** Print subsystem
- **RP** Report editing
- **SC** Session control
- **ST** Source transport

**ACTTYP**

Type or mode of activity. Possible codes are D (display), E (edit), and U (utility).

**USER**

Three-character user ID.

**ERRID**

One- to ten-character string that the program returned detecting the error to aid in problem determination (for example, this can be the sequence number of an iterated call to the service). This field can be blank.

**DSFCMND**

Command issued to the dictionary service facility that resulted in the specified return code.

**QUALIFIER**

Dictionary service facility User Request Area qualifier field passed to the dictionary service facility.

**SYS**

Three- character short system ID currently selected.

**ENTTYP**

Current entity type of the Datadictionary component of CA Datacom/DB.

**ENTNAM**

Current Datadictionary entity name.

**ENTVER**

Current Datadictionary entity version.

**ENTSTAT**

Current Datadictionary entity status.

**PANEL-NAME**

Current panel name.

**VER**

Current panel version.

**SUB-SYS**

Current panel subsystem (PDF only).

**LIB**

Seven- or eight-character VLS library identifier (corresponds to the DD/DLBL name for the indicated VLS file).

**MEM**

An 11-to-24-character VLS member name, possibly with embedded spaces.

**LOGMSG**

Optional message written to ADRLOG by the program detecting the error. It can be in any format.

# @I$INTERNAL STATUS

The following command displays the release levels of the major CA Ideal, CA IPC, and CA Datacom components and the maintenance applied to this CA Ideal release.

This command has the following format:

```
@I$INTERNAL STATUS
```

The first part of the display shows release levels for CA Ideal, VPE, PMS, SCF, PSS, and CA Datacom/DB. The second part of the display shows the current maintenance level for CA Ideal.

Run this command, online or in batch, before and after applying each maintenance tape and before contacting CA Ideal Technical Support with a problem. CA Ideal Release Levels and Maintenance Level are also shown on the CA Ideal Trace Facility printout, as part of the environment page.

### Example

### Output (Mixed Environment)

When this command is issued in a mixed (CA Datacom/DB and DB2) environment, the following output results:

```
IDEAL 14.0  RELEASE  LEVELS                                    IDEAL OPTIONS
  VPE VERSION:  14.0  SCF VERSION:  11.0
  PMS VERSION:  14.0  PSS VERSION:  11.0                        DB:    Y
  D-D VERSION:  14.0  D-B VERSION:  11.0                        DBSQL: DATACOM
                                                                DB2:   Y
DB2/CICS ATTACH:  I710                                        VSAM:  Y
  CICS APPLID:  A11IC9P3  DB2 SUBSYS :  D71C

Maintenance:
```

The section on CA Ideal options indicates the following:

- **DB: Y**-CA Datacom/DB is used in Datacom mode
- **DBSQL: DATACOM**-CA Datacom SQL used
- **DB2: Y**-DB2 used
- **VSAM: Y**-CA Ideal VSAM option available

# Display VLS Library Index

VLS libraries are maintained with the installed utility VLSUTIL. Its functions are described in the CA IPC *Implementation Guide*. The VLSUTIL LIBRARY function lets you list the indexes of the current CA Ideal system's VLS libraries (source, object, or panel) and the space use statistics. The following command accesses the LIBRARY function online, directly from CA Ideal.

This function has the following format:

```
                              {@I$SRC   }
{DISPLAY} INDEX MEMBER USER   {@I$OBJ   }
{PRINT  }                     {@I$PNL   }
                              {@I$DVW   }
                              {@I$DAT   }
                              {@I$PLAN  }
                              {@I$ADRLIB}
```

**@I$SRC**

Displays or prints the index of the source library.

**@I$OBJ**

Displays or prints the index of the object library (including compiled panels).

**@I$PNL**

Displays or prints the index of the panel library.

**@I$DVW**

Displays or prints the index of the dataview object library (IDDVW).

**@I$DAT**

Displays or prints the index of the member library (IDDAT).

**@I$PLAN**

Displays or prints the index of the plan library (IDDVW or a site-defined plan library).

**@I$ADRLIB**

Displays or prints the index of the library containing users' job statements, product messages, and help members, both user-defined and product-defined.

You can use this command to display the index when you suspect that a library is full, when the index was corrupted, or when the library appears to be out-of-sync with the dictionary facility.

**Note:** This command does not produce exactly the same space information as VLSUTIL.

### Example

## Input

```
DISPLAY INDEX MEMBER USER @I$SRC
```

## Output

The following illustration is the resulting output:

```
IDEAL: DISPLAY INDEX MEM     MEM                                SYS: $ID   DISPLAY

NAME ....1....+....2.... RECS  DESCRIPTION             CREATED  UPDATED COMMAND
============================== T O P ======================================
$ID@I$CURSR       001P       5 PARAMETER DATA SOURCE    09/05/04 09/05/04 000001
$IDCHECK          001L       4 PDL SOURCE               09/26/04 09/26/04 000002
$IDEMPLOYEE       001L       0 PDL SOURCE               09/20/04 09/20/04 000003
$IDJUNK           001L       4 PDL SOURCE               09/31/04 09/31/04 000004
$IDSCRLLSUB       001L       1 PDL SOURCE               09/05/04 09/20/04 000005
$IDSCRLLSUB       001L E     1 PDL SOURCE               09/05/04 09/20/04 000006
$IDSCRLLSUB       001P       1 PARAMETER DATA SOURCE    09/05/04 09/05/04 000007
$IDSCROLLER       003L     553 PDL SOURCE               09/05/04 09/05/04 000008
$IDSCROLLER       003P       1 PARAMETER DATA SOURCE    09/05/04 09/05/04 000009
$IDSCROLLER       003W      31 WORKING DATA SOURCE      09/05/04 09/05/04 000010
$IDTST            001L       4 PDL SOURCE               09/25/04 09/26/04 000011
IPBIPBLOAD        001L     321 PDL SOURCE               09/06/04 09/06/04 000012
IPBIPBLOAD        001W       9 WORKING DATA SOURCE      09/06/04 09/06/04 000013
IPBIPB031IS       001L      41 PDL SOURCE               09/06/04 09/06/04 000014
IPBIPB031IS       001W       4 WORKING DATA SOURCE      09/06/03 09/06/03 000015
IPBIPB031SR       001L      38 PDL SOURCE               09/06/04 09/06/04 000016
IPBIPB031SR       001W       4 WORKING DATA SOURCE      09/06/04 09/06/04 000017
IPBIPB041         001R      20 REPORT SOURCE MEMBER     09/06/04 09/06/04 000018
IPBIPB041IS       001L      13 PDL SOURCE               09/06/04 09/06/04 000019
IPBIPB041IS       001W       1 WORKING DATA SOURCE      09/06/04 09/06/04 000020
IPBIPB051IS       001L     121 PDL SOURCE               09/06/04 09/06/04 000021
IPBIPB051IS       001W       1 WORKING DATA SOURCE      09/06/04 09/06/04 000022
IPBIPB071IS       001L      65 PDL SOURCE               09/06/04 09/06/04 000023
IPBIPB071IS       001W       2 WORKING DATA SOURCE      09/06/04 09/06/04 000024
IPBIPB072IS       001L     215 PDL SOURCE               09/06/04 09/06/04 000025
IPBIPB072IS       001W      10 WORKING DATA SOURCE      09/06/04 09/06/04 000026
IPBIPB081IS       001L      28 PDL SOURCE               09/06/04 09/06/04 000027
IPBIPB081IS       001R      47 REPORT SOURCE MEMBER     09/06/04 07/06/94 000028
```

# Display SCF Environment (CICS Only)

The DISPLAY INDEX SESSION command, issued from CA Ideal, lists the name and status of each current user operating under the Session Control Facility (SCF) environment. This command is available only under CICS (z/OS or VSE). For more information about syntax of this command, see the *Programming* Reference *Guide*.

### Example

## Input

```
DISPLAY INDEX SESSION
```

## Output

The following illustration is the resulting output:

```
DISPLAY INDEX         SES                              SYS: DMP   DISPLAY

User  OPID  Reg  User Name       Term  Product  Sys  Main PGM  Ver Sub
============================== T O P ================================
 BRR   BRR    1   BURROWS         Z000    IDEAL  WOR  WORUN     PRD 0005
 MTA   MTA    1   METELITSA       Z001    IDEAL  WOR  WORUN     PRD 0003
 AGI   AGI    1   AGIN            Z002    DDOL
 EMY   EMY    1   DEWITT          Z005    IDEAL  DEM  EDUC00    003 0002
 PRY   PRY    1   PERRY           Z006    IDEAL       **NONE**      0000
 RON   SSN    1   SANDERSON       Z008    DDOL
 ELM   ELM    1   ELMER           Z011    IDEAL  Q80  FORSQL    001 0001
 ELM   ELM    2   ELMER           Z011    IDEAL
 ELM   ELM    3   ELMER           Z011    IDEAL
 SAS   SSS    1   DSASSER         Z017    IDEAL  SAS  OEDRVR    PRD 0012
 $ID   $ID    1   $IDEAL          Z019    IDEAL       **NONE**      0000
```

# @I$UTIL PSS Online Debugging

PSS provides several online commands for maintaining the output library (ADROUT). They let you obtain information on the current status of the output library and obtain an index of all members in the output library.

The following command displays or prints the current status of the output library.

This command has the following format:

```
@I$UTIL OUT STATUS [print-name {SYSTEM } name]
                   [           {NETWORK}     ]
```

**print-name**

Contains the name identifying this display or print.

**name**

Contains the destination ID of the system printer or network printer.

The command provides the following information:

- Date and time of last initialization.
- Date and time of last clean-up maintenance.
- User ID of person submitting the clean-up request.
- Use of library space, including total blocks on library, number of free blocks, percent occupied, and maximum number of members the library can accommodate.

### Example

The command @I$UTIL OUT STATUS alone displays status information online at your terminal.

The following submits the output to print on network printer HR86 with the identifier MYLIB:

## Input

```
@I$UTIL OUT STATUS MYLIB NETWORK HR86
```

## Output

```
STATUS OF THE LIBRARY "ADROUT  "
LAST INITIALIZATION:   04/09/04  11:05
LAST RECOVERY:         06/28/04  10:15:56   PSS
SPACE:    1002 BLOCKS     FREE:    682    OCCUPANCY:  32%
NUMBER OF PRINT MEMBERS:     505
```

The output shows the information for the current PSS output directory in ADROUT. There might be more than one output directory. The output directory was built by using SCPSUTIL. See the CA IPC *Implementation Guide* for further information regarding the use of SCPSUTIL. The following command displays or prints the names of all members in the specified VLS library.

This statement has the following format:

```
@I$UTIL OUT INDEX [ADROUT  {SYSTEM } name]
                  [file-id {NETWORK}     ]
```

**ADROUT**

The output library.

**file-id**

The DD/DLBL name of a VLS library.

**name**

Destination ID of the system or network printer.

The command provides the following information:

- Description of each member.
- Total records per member.
- Number of blocks occupied by the member.
- Date created.
- Date updated.
- Length of each record.

### Example

## Input

```
@I$UTIL OUT INDEX
```

## Output

The following illustration is the resulting output:

```
IDEAL  DISPLAY OUTPUT       OUT INDEX             DISPLAY

P.S.S. - ONLINE  UTILITY  STATUS OF THE LIBRARY ADROUT  DATE 09/30/04  TIME 11.59.56
MEMBER NAME               MEMBER DESCRIPTION           NREC NBLK ADDED UPDATED RLEN
================================================================================
=====

PSS$PSSDIR$              *** PSS SPOOL DIRECTORY ***   0250 0014 070690 073199 0179
PSS$PSSDST@              PSS - DESTINATION TABLE      0012 0002 071190 071903 0024
PSSSCF#0052               COMPILE LISTING             0021 0002 071904 071904 0133
PSSSCF#0053               COMPILE LISTING             0133 0004 071904 071904 0133
PSSSCF#0054               COMPILE LISTING             0109 0004 071904 071904 0133
PSSSCF#0056               COMPILE LISTING             0350 0006 071904 071904 0133
PSSSCF#0057               DISPLAY INTERR:             0009 0002 071904 071904 0133
PSSSCF#0071               COMPILE LISTING             0118 0004 071904 071904 0133
```

The output directory in this example is the member PSS$PSSDIR$, which contains a system name of PSS and a directory name of $PSSDIR$. You can set both system name and directory name with the command SET OUTPUT OPTIONS.

# Chapter 5: Error Recovery Tools

This chapter describes CA Ideal features that you can use after an error occurs. It includes a description of <<ERROR PROCEDURES>> and the DEQUEUE command (including the format of enqueue names).

## PDL <<ERROR>> Procedure

The <<ERROR>> procedure specifies a set of actions to invoke when a run-time error occurs. The <<ERROR>> procedure can provide additional information for debugging (such as return codes) or override the default procedure and continue processing following an error.

The <<ERROR>> procedure overrides the CA Ideal default error procedure. The default procedure issues a LIST ERROR statement, performs a BACKOUT, issues a standard message, and QUITs the RUN. You can write your own <<ERROR>> procedure, for example, to display other information and to execute a PROCESS NEXT statement.

This procedure has the following format:

```
<<ERROR>> PROCEDURE
     statements
[ENDPROC     ]
[ENDPROCEDURE]
```

**<<ERROR>> PROCEDURE** The statements in the body of the <<ERROR>> procedure can consist of any CA Ideal/PDL statements and functions needed to process the error. If the last statement is not a QUIT RUN, QUIT statement, or PROCESS NEXT statement, then the default error procedure runs (following the <<ERROR>> procedure).

A program can contain only one error procedure.

The PDL internal functions used in error handling are described in the Programming Reference Guide. These functions return meaningful information about CA Ideal errors only in the <<ERROR>> procedure.

### Example

The following illustration describes the usage of <<Error>> procedure:

```
<<ERROR>> PROCEDURE
:-------------------------------------------------------------------
:  The following <<ERROR>> PROC uses a PDL function $ERROR-CLASS to
:  trap dataview errors, arithmetic errors and reference errors.
:  When the $ERROR-CLASS is 'DVW', the PDL functions $ERROR-TYPE and
:  $ERROR-DVW-STATUS further qualify the error.
:
:  The PROCESS NEXT statement is used for recoverable errors; fatal
:  errors result in the termination of the program.
:-------------------------------------------------------------------
  SELECT FIRST ACTION
  WHEN $ERROR-CLASS = 'DVW'
       SELECT FIRST ACTION
       WHEN $ERROR-TYPE = 'D71'
            NOTIFY 'Record changed; retry'
            PROCESS NEXT TRANS-LOOP
       WHEN $ERROR-TYPE = 'D72'
            NOTIFY 'Record deleted; retry'
            PROCESS NEXT TRANS-LOOP
       WHEN $ERROR-TYPE = 'DB2'
            NOTIFY 'DB2 problem:  Run Terminated'
            SET $RC = $SQLCODE
       WHEN $ERROR-TYPE = 'SQL'
            NOTIFY 'DATACOM SQL problem:  Run Terminated'
            SET $RC = $SQLCODE
       WHEN $ERROR-TYPE = 'DVW'
            SELECT FIRST ACTION
            WHEN $ERROR-DVW-STATUS = 'I3'
                 NOTIFY 'Record changed; retry'
                 PROCESS NEXT TRANS-LOOP
            WHEN $ERROR-DVW-STATUS = '16'
                 NOTIFY 'Record in use; retry'
                 PROCESS NEXT TRANS-LOOP
            WHEN OTHER
                 NOTIFY 'DATACOM CBS problem:  Run Terminated'
                 SET $RC = '66'
            ENDSEL
       ENDSEL
  WHEN $ERROR-CLASS = 'ARI'
       DO ARITHMETIC-ERR
       PROCESS NEXT MAIN-LOOP
  WHEN $ERROR-CLASS = 'REF'
       DO REFERENCE-ERR
       PROCESS NEXT MAIN-LOOP
  WHEN OTHER   : Non-recoverable errors
       NOTIFY 'System error:  Run Terminated'
       SET $RC = 12
  ENDSEL
  LIST ERROR
  BACKOUT
  QUIT RUN
ENDPROC
```

# DEQUEUE Command

CA Ideal uses enqueues to protect components including programs, panels, reports, and members, and so on. CA Ideal components are protected from simultaneous update, even by another CICS region or a batch job. This protection is at the individual component level (program, panel, report, and so on) rather than at the data set level.

Online, when a transaction abends, the program being edited might remain enqueued. When you sign on again, an attempt to re-edit the program results in the “resource busy” message. This can also happen when a system failure occurs while displaying an index, output, or jobcard. You cannot dequeue the program from another region because the operating system keeps track of where the enqueue came from. This problem does not occur for a batch job, however, because, when any batch job ends (including CICS), the operating system automatically cleans up all enqueues left outstanding by that job.

The site CA Ideal Administrator has the responsibility to DEQUEUE entities. *The site administrator should always determine that the outstanding* enqueue is a result of a *transaction abnormal termination before issuing the DEQUEUE.* For example, if one programmer is editing a panel and a second programmer tries to edit the same panel, the second programmer gets the same “resource busy” message. If the administrator issued a DEQUEUE in this situation and allowed the second programmer to begin editing the panel, serious library corruption could result.

The *Command Reference Guide* includes the format of this command. Also, see the *Administration Guide* for information regarding enqueues.

Do not try and relate CA Ideal entities enqueued and dequeued to system data sets or other system objects. CA Ideal generates a unique internal enqueue name for each object. This internal enqueue name does not correspond to any object known to the operating system.

The format for a CA Ideal enqueue name is:

```
10     20
+----+----+----+----
$Itsssxxxxxxxxebbbbbbvvv For PGM, PDL, WDT, PRM,RPT, PGMOBJ,      and PNL names.
$Itbbbyyyyyyyyyyyyyyyvvv For PER, SYS, PLA, DVW names.
$Ituuuzzzzzzzzbbbbbbbbbb For MEM names.
```

- **T** Type code are defined as follows:
- **B** Plan
- **D** Dataview
- **K** Sequential dataview
- **L** Program procedure source
- **P** Program parameter data
- **R** Report
- **S** Program DD entity
- **T** Program object data
- **U** Panel
- **W** Program working data
- **X'BB'** Person entity (user)
- **X'CE'** System entity
- **Z** Data member
- **sss** System ID.
- **x..x** Program, panel, or report name.
- **y..** Person, system, or dataview Datadictionary name.
- **vvv** Version number.

- **e** Optional enqueue code to uniquely identify names across systems. Blank if not used.
- **b..b** Spaces.
- **uuu** User ID.
- **z..z** Member name.

For z/OS the major or queue name, ADRPRDCT, is used with the minor or resource name. These names are compressed for VSE to a 12-byte format. The resulting name might not be printable.

### Example

```
DEQUEUE PGM DEMO1 VER 1 SYS DOC
```

Dequeues program DEMO1 in system DOC.

```
DEQUEUE LIBRARY $IDLIB
```

Dequeues VLS dataview library $IDLIB.

# @I$UTIL PSS Online Dequeue

The following command dequeues a print file, the display status member, or the PSS directory if the resource is caught in an enqueue due to a system failure, a CA Ideal/CA IPC failure, or another failure level.

This command has the following format:

```
                    [print-number]
@I$UTIL OUT DEQUEUE [STATUS      ]
                    [DIRECT      ]
```

**print-number**

Identifies a print file.

**STATUS**

Displays the status member.

**DIRECT**

Displays the output directory.

**Important!** Before using this command, be certain that the resource is locked due to an error condition and not due to a legitimate enqueue.

# Using DEQUEUE

Some situations in CA Ideal require the use of the DEQUEUE command. However, issuing the DEQUEUE command in the wrong place or at the wrong time can cause library corruption and system abends. Here are some tips on when to use DEQUEUE.

## Case 1: Program in Use

One of the most common situations where a dequeue might seem appropriate is when you try to edit a program and receive the message:

```
IDADPSEP03-PGM xxxxxxxx is in use, please try later.
```

The fact is, the dequeue is appropriate only under a small set of circumstances if you receive this error. You need to ask yourself the following set of questions.

- Is someone in this CA Ideal environment editing this program?

  If you have a monitoring package such as CA SYSVIEW, you CAN see the outstanding enqueues by checking the major enqueue name "ADRPDCT".

  If you do not have a monitoring package, then there is no easy and automated way to tell if an enqueue exists for that program. However, there is a good way to find out who might be using that program. The command DISPLAY INDEX SESSION tells you who is on the system and who is running programs.

  From this, you can determine who is in CA Ideal but not currently running programs. Check whether any of these people are editing the program.

  You can also check whether the program's identification component shows that it was recently updated. The indicated person might still be editing the program.

  If your answer to this question is yes, then the message you are receiving is valid and you should not dequeue.

  If your answer to this question is no, then continue to the next question.

- Is this program currently being compiled?

  You should be able to tell if an asynchronous online compile is currently being done through one of the following:

  - A DISPLAY OUTPUT ALL STATUS command.
  - The CEMT Inquire Task command.

  If there is a task with the SAST transaction, check the terminal ID. Find out who is at that terminal and see if they are compiling that program. Also check whether any batch compiles for this program are being done.

  If your answer to this question is yes, then the message you are receiving is valid and you should not dequeue.

  If your answer to this question is no, then continue to the next question.

- Does a program of the same name exist in another CA Ideal environment?

  First, find out if this CA Ideal environment has its own MUF (Multi-User Facility) and its own set of VLS libraries.

  If multiple CA Ideal environments are sharing MUF and VLS libraries, then this question is not an issue and you can continue to the next question.

  If your CA Ideal environments are completely separate and a program of the same name (in the same system) does exist in another CA Ideal environment, then check your enqueue characters (QCODE). The QCODE is what makes your enqueue names unique between CA Ideal environments.

  Since enqueues are done at the operating system level, identical QCODES in multiple regions enqueue programs with the same name (in the same system) in all regions where the QCODE is the same. To check your QCODES, issue the following command in CA Ideal:

  ```
  @I$SCF PGM=SC00OPTS OFF=8
  ```

  Offset 8 in the program SC00OPTS tells you what your QCODE is for this environment. Issue the same command in all your CA Ideal environments and note the QCODE. If your environments are completely separate, you need to make the QCODE unique among all your CA Ideal environments.

  If your answer to this question is yes, then check your QCODEs. Do not issue the DEQUEUE command.

  If your answer to this question is no, then continue to the next question.

- Was there a system abend while someone was editing or compiling this program?

  Once again, make sure that nobody is currently editing or compiling this program.

  If your answer to this question is no, then call CA Ideal Technical Support for further assistance.

  If your answer to this question is yes, then you are safe to dequeue the program. Issue the following command:

  DEQUEUE PROGRAM *xxxxxxxx* VERSION *nnn* SYSTEM *sys*

  You can find the syntax for this command in the *Command Reference Guide*.

## Notes on Enqueues During Edit Sessions

If you find the answer to the last question above to be yes and nobody is trying to edit the program, consider the following. If the abend occurred while someone was actually editing the program, then the program is the only thing that must be dequeued. If by chance the abend occurred while you were writing to the library, (the user made changes to the program and pressed the Enter key), then there is an enqueue on the library. Users probably get a VLS internal error with a return code = 4-52 when they try to access any program in that library. You can dequeue the library by issuing the command:

```
DEQUEUE LIBRARY library-name
```

Enqueues are also done on panels and reports. If an enqueue is left on these entities, you get an error message similar to the ones mentioned above. Follow the same steps as you would for a program.

## Case 2: Production Program in Use

Another situation where a DEQUEUE command might seem appropriate is when you try to mark a program to Production status and receive the message:

```
IDADXSSP24E - Prod program in use, please try later.
```

Do *not* issue a dequeue command when you get this error.

Check the following things.

- Is someone in this region currently running the Production version of this program?

  You can use the DISPLAY INDEX SESSION command to determine if anyone is running the Production version of the program. You also need to find out if anyone is running the program in batch.

If your answer to this question is yes, then the error message you received is valid. You should disable the application, then mark the new version of the program to Production. See the *Command Reference Guide* for the syntax of the DISABLE command. Remember, the disable only takes effect at a transaction boundary. If the user left his terminal with the program running, the application is not truly disabled until that user presses the Enter key.

If your answer to this question is no, then continue to the next question.

- Does a program of the same name exist in another CA Ideal environment and, if so, is someone running the Production version of this program?

  See the third question in the previous section for information on QCODE. Follow the same instructions.

  If your answer to this question is no, call CA Ideal Technical Support for further assistance. Do not attempt to dequeue the program.

  If your answer to this question is yes, check your QCODEs for all of your CA Ideal environments (as described earlier). If the QCODEs are unique and the error persists, call CA Ideal Technical Support.

## Notes on Enqueues during a MARK STATUS Command

You can receive similar error messages when marking panels and reports to Production status as when marking programs. Follow the same steps as you would for programs.

Enqueues are also done on user definitions. If you try to mark a user definition to Production status while the user is already signed on, you receive the message:

```
IDADUEDS02-User Definition In Use, Please Try Later
```

Check if that user is active online or in batch. If so, have the user sign off. Do not issue the DEQUEUE command.

The same is true with system definitions. If a user selected a system, you receive the message:

```
IDADDSEDP02-System Definition In Use, Please Try Later
```

Check if anyone is in that system, running a program from the system or editing and displaying entities in that system. If so, have them select another system definition. Do not issue the DEQUEUE command.

# Appendix A: Dial Trace Codes

This appendix contains the trace codes for each CA Ideal module and the module's description.

## Run and Debug Codes

The table that follows shows Run and Debug DIAL command codes (set by SET RUN @I$DIAMASK session command).

| Code | Module | Description |
|---|---|---|
| Dial-A | AECOLL | AEF Statistical collector |
| | AEPROM | Print a formatted object module |
| | AESTAT | Object module statistical collector |
| Dial-B | AETINT | AEF maint-code executor |
| Dial-C | AETPGM | AEF program control |
| | ADRUNP | AEF driver module |
| | ADRUPP | Run post processor |
| Dial-D | AEPLIP | PL/I batch interface |
| | AETRUN | AEF T-code interface: Run control |
| Dial-E | AEDB2D | AEF Dynamic DB2 SQL I/O mod |
| | AEFUNC | AEF String functions |
| Dial-G | AELPGM | AEF LOAD/LOCATE PGM |
| Dial-I | ADMODP | CRE/DEL/IDE MODULE commands |
| | ADPLAG | GENERATE PLAN processor |
| | AEINIT | AEF Run-time initializer |
| | AERCBP | AEF dynamic RCB processor |
| | AESTAT | Object module statistical collector |
| | ADRESR | DB2 APRES retriever |
| Dail-J | AETSQL | SQL logic module |
| | AEDB2P | AEF DB2 SQL T-code |
| | ADDBQI | DB Q-COMMAND INTERFACE PGM |
| | ADSQLE | BATCH DBSQLE |

| Code | Module | Description |
|---|---|---|
| Dial-K | SDBUGP | Symbolic debug dispatcher |
| | SDTREN | SYMBOLIC DEBUG TRANSACTION END |
| Dial-L | SDIDEN | Debug: look up identifier |
| | SDOUTP | SYMBOLIC DEBUG OUTPUT PROCESSOR |
| | SDBUGP | Symbolic debug dispatcher |
| | SDEDKP | DEBUGGER EDK SERVICE PROC |
| | SDGDAT | Debugger generator data display member |
| | SDXTAB | DEBUGGER EXTERNAL BREAKPT MGR |
| Dial-M | AETMAP | AEF panel T-code processor |
| Dial-P | AEAUXP | AEF ASSIGN commands |
| | AEPSS | AEF PSS driver |
| Dial-R | AERWDR | AEF Report Writer driver |
| | RWEXEC | RW: Execution control |
| | RWEX20 | RW: Execution control |
| Dial-S | AELSYM | AEF load and locate symbol table |
| Dial-T | AELMAP | AEF LOAD/LOCATE PANEL |
| Dial-V | AETDVV | Extension of AETDVW |
| | AETDVW | AEF T-code for dataview I/O |
| Dial-W | AETDVW | AEF T-code for dataview I/O |
| Dial-X | AERLSE | AEF cleanup processor |
| Dial-Z | AETERR | AEF error handler |
| | SDRTAB | DEBUGGER RUN-TIME STOP TBL BLD |

# Compile and Catalog Codes

The following table shows Compile and Catalog DIAL command codes (set by SET COM @I$DIAMASK session command).

| Code | Module | Description |
|---|---|---|
| Dial-A | CMAEXP | Compiler arithmetic expression parser |
| | CMPDEO | Prt data-entry object: debugging |
| | CMCDEI | Compile data-entity initializer |
| | CMFATL | Compiler fatal error processor |

| Code | Module | Description |
|---|---|---|
| Dial-B | CMCEXP | Condition analyzer main mod |
| | CMBCBS | Build CBS work area |
| | CMBCTR | COND ANAL TO BUILD COND TREE |
| | CMNORM | NORMALIZE A CONDITION TREE |
| Dial-C | CMCTRL | Compiler control module |
| | CMFINL | Compiler release resources mod |
| Dial-E | CMERR | Compiler error msg processor |
| Dial-F | CMCDET | Compile data-entity terminate |
| | CMCDEP | Compile data-entity process |
| | CMCDEC | Compile data-entity create |
| | CMODEO | Obtain data-entity object |
| | CMASST | Compiler attribute services - stow |
| | CMAS | Attribute Services |
| | CMASMR | Compiler attribute services: merge |
| | CMASMX | Comp attribute services: Auxiliary merge |
| | ADDCMP | Catalog dataview processor |
| Dial-I | CMINIT | Compiler initialization pgm |
| Dial-J | ADSQLE | BATCH DBSQLE |
| | ADDBQI | DB Q-COMMAND INTERFACE PGM |
| Dial-L | CMLX | Compiler lexical analyzer |
| | CMLXSX | Lexical select error message processor |
| | CMLXSL | Compiler lexical select |
| Dial-P | CMLOOP | Compiler loop stmt parser |
| | CMLIST | List statement parser |
| | CMMAPP | Compiler panel stmt parser |
| | CMGENP | Compiler parser control module |
| | CMGENX | Compiler-generate final blocks |
| | CMAUXP | ASS/RES/SET-ID STMT PARSER |
| | CMADSB | Compiler add/sub stmt parser |
| | CMCALP | CALL Statement Parser |
| | CMDVWS | Delete, checkpoint, backout statement |
| | CMFORE | For each p-code generator |
| | CMFORN | For next p-code generator |
| | CMFORW | P-code generator of “for new” |
| | CMPRDC | Compiler “produce” statement module |
| | CMSELX | Compiler select parser |

| Code | Module | Description |
|---|---|---|
| Dial-Q | CMTGEN | Compiler t-code generator |
| | CMPREP | Compiler SQL prepass pgm |
| | CMSCOP | Compiler scope analysis |
| | CM2GEN | SQL string generator |
| | CMDBQP | DB SQL Parser |
| | CMGENQ | SQL PASS DRIVER |
| | CM2PAR | DB2 SQL Parser |
| Dial-R | RWAEXP | RW: Arithmetic expression analyzer |
| | RWANAL | RW: analyze detail |
| | RWCF | RW: control footing |
| | RWCOL | RW: column analysis |
| | RWDET | RW: detail |
| | RWPH | RW: page heading |
| | RWTGEN | RW: t-code generator top level |
| | RWUTIL | RW: compiler utility |
| | CMRWDR | RW: Compiler driver |
| | CMRWER | COMPILER: RW ERROR HANDLER |
| Dial-S | CMTPAD | Compile Trim-Pad function |
| | CMTRAN | $translate function parser |
| | CMSEXP | Compiler string expression parser |
| | CMSSTR | $substract function parser |
| | CMGPMV | COMPILE GROUP MOVE |
| | CMNFNC | Compile numeric function |
| | CMGSTR | Compile group string |
| | CMINDX | $index function parser |
| | CMEDIT | $edit function parser |
| | CM$$DT | $date, $time parser |
| | CM$GSF | Compile General string function |
| | CM$STR | Compiler $string function parser |
| Dial-T | CMTGEN | Compiler t-code generator |
| Dial-U | ADDB2R | DB2 catalog traverse |
| Dial-Z | CMPRTP | Compiler print processor |
| | CMPREF | Print cross reference list program |

# Source Transport Codes

This table shows Source Transport DIAL codes that can be set by the command SET @I$IDIALMASK in the Source Transport job.

| Code | Module | Description |
|---|---|---|
| Dial-D | STEXPD | Export dispatcher |
| | STIMPD | Import dispatcher |
| | STPRDD | PRODUCE IMPORT COMMANDS dispatcher |
| | STTABP | Table processor |
| Dial-E | STPNLE | Panel export module |
| | STRPTE | Report export module |
| | STDVWE | Sequential DVW export |
| | STMEME | Member export module |
| | STPGME | Program export module |
| Dial-I | STMEMI | Member import module |
| | STHLPI | Import user HELP |
| | STDVWI | Sequential DVW import |
| | STRPTI | Report import module |
| | STPGMI | Program import module |
| | STPNLI | Panel export module |
| Dial-L | STLIST | Transport report module |
| Dial-O | STIOST | I/O module |
| Dial-P | STIMPP | IMPORT command processor |
| | STEXPP | EXPORT command processor |
| | STOFFP | OFF command processor |
| | STCMDP | Source transport parser |
| | STDSPP | Source transport dispatcher parser |
| | STPRDP | PRODUCE command processor |
| | STSETP | Source transport SET processor |
| Dial-Q | STDSPP | Source transport dispatcher parser |
| Dial-T | STTOKN | Tokenizer |

# Appendix B: VLS Members and Utilities

VLS supports member names of up to 40 characters. Member names can contain embedded blanks. CA Ideal program, panel, report, dataview object, and data member entities all use 24-character member names.

## CA Ideal VLS Member Name Format

Since a site can use a single VLS library for storing various types of information, CA Ideal uses the following conventions internally to identify VLS members.

A type code in the 22nd position of the name identifies the member as:

- **'A'-**Compiler encoded error messages
- **'B'-**DB2 PLAN source
- **'D'-**Dataview object
- **'H'-**Help text member
- **'J'-**Symbol table object
- **'K'-**Sequential dataview source
- **'L'-**Program PDL source
- **'N'-**Compiler panel object

  **'P'-**Parameter data source
- **'Q'-**Parameter data object
- **'R'-**Report specification
- **'T'-**Program object
- **'U'-**Panel specification
- **'V'-**Working data object
- **'W'-**Working data source
- **'Z'-**Data member

Based on the member type, the following name formats are used:

```
Types            VLS Name 10        20        30        40
------      ----+----+----+----+----+----+----+----+----+
ALPRUW           sssxxxxxxxx.......vvvt.e
JNQTV            sssxxxxxxxx......pvvvtc.
D                $DVyyyyyyyyyyyyyyyyvvvtcyyy
 -or-          $D2yyyyyyyyyyyyyyyyvvvtcaaaaaaaayyy
K                yyyyyyyyyyyyyyyyyyyvvvt.e
HZ               uuuxxxxxxxx..........t.e
B                $APppppppp...........t.e
```

- **sss-**Three-character system ID of the CA Ideal system containing the program, panel, or report.
- **uuu-**Three-character user ID of the data member's owner.
- **xxxxxxxx-**One- to eight-character program, panel, report, data member, or help member name, left-adjusted and padded to the right with blanks if necessary.
- **yyyyyyyyyyyyyyy-**15- or 18-character dataview name.
- **aaaaaaaa-**One- to eight-character DB2 authorization-ID for DB2 dataviews. Can be blank.
- **ppppppp-**One- to seven-character plan name for DB2.
- **vvv-**Three-digit version number in unsigned zoned format or the literal PRD for the PROD version of the program executable and symbol table objects.
- **t-**One-character type code, as specified above.

- **c-**For source and panel library members, a blank (X'40').

  For object library members, a one-character sequence ID:  A-Z, 0-9, X'01-0E'. For all object types except the program executable object, there can be only one object member with a sequence ID of A.

- **e-**One-byte edit-indicator to distinguish an EDIT session work- copy of a source member from the original.

  Normally, this position is blank (X'40'), indicating that this is the original member. For object members, this position is always blank.

  When editing a source member, a copy of the member is made with an E in this position. This work-copy is used when a ROLLBACK operation is done, either as a primary command or as an option when editing a member that a previous EDIT session terminated abnormally. When an EDIT session ends normally, the work-copy is deleted.

  This position can also be R if the member is being deleted.

  For data members (on library IDDAT) only, this position can be I if the member is under the control of an internal command.

  For more information about using VLSUTIL and IDUTILITY, the Library Integrity Utility, see the *Administration Guide*.