Debug Tool for z/OS

Customization Guide

Version 10.1

GC14-7257-00

Debug Tool for z/OS

Customization Guide

Version 10.1

GC14-7257-00

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 115.

This edition applies to Debug Tool for z/OS, Version 10.1 (Program Number 5655-V50), which supports the following compilers:

- AD/Cycle® C/370™ Version 1 Release 2 (Program Number 5688-216)
- C/C++ for MVS/ESA Version 3 (Program Number 5655-121)
- C/C++ feature of OS/390 (Program Number 5647-A01)
- C/C++ feature of z/OS (Program Number 5694-A01)
- OS/VS COBOL, Version 1 Release 2.4 (5740-CB1) - with limitations
- VS COBOL II Version 1 Release 3 and Version 1 Release 4 (Program Numbers 5668-958, 5688-023) - with limitations
- COBOL/370 Version 1 Release 1 (Program Number 5688-197
- COBOL for MVS & VM Version 1 Release 2 (Program Number 5688-197)
- COBOL for OS/390 & VM Version 2 (Program Number 5648-A25)
- Enterprise COBOL for z/OS and OS/390 Version 3 (Program Number 5655-G53)
- Enterprise COBOL for z/OS Version 4.2 and earlier (Program Number 5655-S71)
- High Level Assembler for MVS & VM & VSE Version 1 Release 4, Version 1 Release 5, and Version 1 Release 6 (Program Number 5696-234)
- OS PL/I Version 2 Release 1, Version 2 Release 2, Version 2 Release 3 (Program Numbers 5668-909, 5668-910) - with limitations
- PL/I for MVS & VM Version 1 Release 1 (Program Number 5688-235)
- VisualAge PL/I for OS/390 Version 2 Release 2 (Program Number 5655-B22)
- Enterprise PL/I for z/OS and OS/390 Version 3.9 or earlier (Program Number 5655-H31)

This edition also applies to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

You can order publications online at www.ibm.com/shop/publications/order, or order by phone or fax. IBM Software Manufacturing Solutions takes publication orders between 8:30 a.m. and 7:00 p.m. Eastern Standard Time (EST). The phone number is (800) 879-2755. The fax number is (800) 445-9269.

You can find out more about Debug Tool by visiting the IBM Web site for Debug Tool at: http://www.ibm.com/software/awdtools/debugtool

Contents

About this document

Debug Tool combines the richness of the z/OS® environment with the power of Language Environment® to provide a debugger for programmers to isolate and fix their program bugs and test their applications. Debug Tool gives you the capability of testing programs in batch, using a nonprogrammable terminal in full-screen mode, or using a workstation interface to remotely debug your programs.

This document describes the tasks you must do to customize Debug Tool.

Who might use this document

This document is intended for system administrators who need to customize Debug Tool.

Debug Tool runs on the z/OS operating system and supports the following subsystems:

- CICS®
- DB2®
- IMS™
- JES batch
- TSO
- UNIX® System Services in remote debug mode or full-screen mode using a dedicated terminal only
- WebSphere® in remote debug mode or full-screen mode using a dedicated terminal only

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM® Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code.

To obtain your IBM Resource Link user ID and password, log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS elements and features, z/VM®, VSE/ESA, and Clusters for AIX® and Linux®:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX System Services running OMVS).
- Your Microsoft® Windows® workstation. You can install code to access IBM message explanations on the *z/OS Collection* (SK3T-4269), using LookAt from a Microsoft Windows command prompt (also known as the DOS command line).
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from a disk on your *z/OS Collection* (SK3T-4269), or from the LookAt Web site (click **Download**, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

How this document is organized

This document is divided into areas of similar information for easy retrieval of appropriate information. The following list describes how the information is grouped:

- Chapter 1 describes how to gather the information that you need that will help you decide which customization tasks to do. It provides a checklist that you can use to organize all of the information.
- Chapters 2 through 7 describe the customization tasks you must do.
- Chapter 8 and the BROWSE topic in chapter 14 describes the customization tasks to enable browse mode. Depending on how you want the enablement to work, you might do the instructions in either topic or both topics.
- Chapter 9 describes the customization tasks you must do if you are using Debug Tool Utilities.
- Chapters 10 through 12 describes the customization tasks you must do if you are using any of the following environments:
  - DB2 stored procedures
  - CICS
  - IMS

- Chapter 13 describes how to implement the EQAUEDAT user exit, which enables the library administrator or system programmer to direct Debug Tool to the location where source, listing, or separate debug files are stored.
- Chapters 14 describes the features or functions you can implement through the EQAOPTS options.
- Chapter 15 describes how to specify default and allowable values for the runtime options NATLANG, LOCALE, and LINECOUNT.
- Appendix A describes SMP/E USERMODs that are available for some customizations.
- Appendix B describes how to apply maintenance provided for Debug Tool.
- Appendix C describes all the resources available to help you find technical support information.
- Appendix D describes the features and tools available to people with physical disabilities that help them use Debug Tool and Debug Tool documents.

The last several topics list notices, bibliography, and glossary of terms.

Terms used in this document

Because of differing terminology among the various programming languages supported by Debug Tool, as well as differing terminology between platforms, a group of common terms has been established. The table below lists these terms and their equivalency in each language.

Debug Tool term	C and C++ equivalent	COBOL or non-Language Environment COBOL equivalent	PL/I equivalent	assembler
Compile unit	C and C++ source file	Program or class	• Program • PL/I source file for Enterprise PL/I • A package statement or the name of the main procedure for Enterprise PL/I[1]	CSECT
Block	Function or compound statement	Program, nested program, method or PERFORM group of statements	Block	CSECT
Label	Label	Paragraph name or section name	Label	Label

Notes:

1. The PL/I program must be compiled with and run in one of the following environments:
   - Compiled with Enterprise PL/I for z/OS, Version 3.6 or later, and run with the following versions of Language Environment:

  - Language Environment Version 1.9, or later
  - Language Environment Version 1.6, Version 1.7, or Version 1.8, with the PTF for APAR PK33738 applied
- Compiled with Enterprise PL/I for z/OS, Version 3.5, with the PTFs for APARs PK35230 and PK35489 applied and run with the following versions of Language Environment:
  - Language Environment Version 1.9, or later
  - Language Environment Version 1.6, Version 1.7, or Version 1.8, with the PTF for APAR PK33738 applied

Debug Tool provides facilities that apply only to programs compiled with specific levels of compilers. Because of this, *Debug Tool Customization Guide* uses the following terms:

assembler
Refers to assembler programs with debug information assembled by using the High Level Assembler (HLASM).

COBOL
Refers to the all COBOL compilers supported by Debug Tool except the COBOL compilers described in the term *non-Language Environment COBOL*.

disassembly or disassembled
Refers to high-level language programs compiled without debug information or assembler programs without debug information. The debugging support Debug Tool provides for these programs is through the disassembly view.

Enterprise PL/I
Refers to the Enterprise PL/I for z/OS and OS/390® and the VisualAge® PL/I for OS/390 compilers.

non-Language Environment COBOL
Refers to any of the following COBOL programs:

- Programs compiled with the IBM OS/VS COBOL compiler.
- Programs compiled with the VS COBOL II compiler with the `NOTEST` compiler option and linked with a non-Language Environment library.

As you read through the information in this document, remember that OS/VS COBOL programs are non-Language Environment programs, even though you might have used Language Environment libraries to link and run your program.

VS COBOL II programs are non-Language Environment programs when you compile them with the `NOTEST` compiler option and link them with a non-Language Environment library. VS COBOL II programs are Language Environment programs when you compile them with the `TEST` compiler option and link them with the Language Environment library.

Read the information regarding non-Language Environment programs for instructions on how to start Debug Tool and debug non-Language Environment COBOL programs, unless information specific to non-Language Environment COBOL is provided.

PL/I Refers to all levels of PL/I compilers. Exceptions will be noted in the text that describe which specific PL/I compiler is being referenced.

How to read syntax diagrams

This section describes how to read syntax diagrams. It defines syntax diagram symbols, items that may be contained within the diagrams (keywords, variables, delimiters, operators, fragment references, operands) and provides syntax examples that contain these items.

Syntax diagrams pictorially display the order and parts (options and arguments) that comprise a command statement. They are read from left to right and from top to bottom, following the main path of the horizontal line.

Symbols

The following symbols may be displayed in syntax diagrams:

Symbol	Definition
▸▸—	Indicates the beginning of the syntax diagram.
—▸	Indicates that the syntax diagram is continued to the next line.
▸—	Indicates that the syntax is continued from the previous line.
—▸◂	Indicates the end of the syntax diagram.

Syntax items

Syntax diagrams contain many different items. Syntax items include:

- Keywords - a command name or any other literal information.
- Variables - variables are italicized, appear in lowercase and represent the name of values you can supply.
- Delimiters - delimiters indicate the start or end of keywords, variables, or operators. For example, a left parenthesis is a delimiter.
- Operators - operators include add (+), subtract (-), multiply (*), divide (/), equal (=), and other mathematical operations that may need to be performed.
- Fragment references - a part of a syntax diagram, separated from the diagram to show greater detail.
- Separators - a separator separates keywords, variables or operators. For example, a comma (,) is a separator.

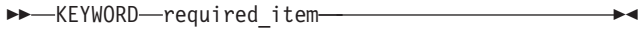
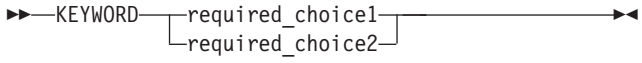
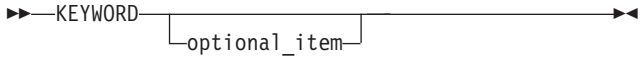
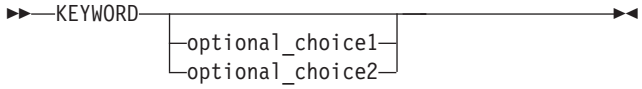
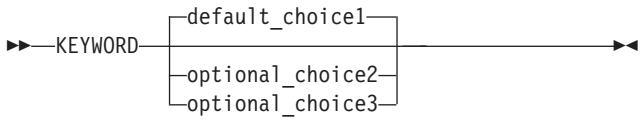
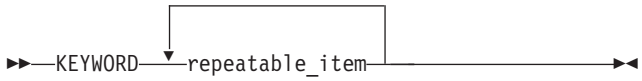
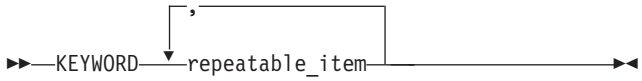
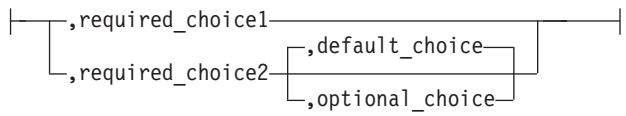
Keywords, variables, and operators may be displayed as required, optional, or default. Fragments, separators, and delimiters may be displayed as required or optional.

Item type	Definition
Required	Required items are displayed on the main path of the horizontal line.
Optional	Optional items are displayed below the main path of the horizontal line.
Default	Default items are displayed above the main path of the horizontal line.

Syntax examples

The following table provides syntax examples.

Table 1. Syntax examples

Item	Syntax example
Required item. Required items appear on the main path of the horizontal line. You must specify these items.	
Required choice. A required choice (two or more items) appears in a vertical stack on the main path of the horizontal line. You must choose one of the items in the stack.	
Optional item. Optional items appear below the main path of the horizontal line.	
Optional choice. An optional choice (two or more items) appears in a vertical stack below the main path of the horizontal line. You may choose one of the items in the stack.	
Default. Default items appear above the main path of the horizontal line. The remaining items (required or optional) appear on (required) or below (optional) the main path of the horizontal line. The following example displays a default with optional items.	
Variable. Variables appear in lowercase italics. They represent names or values.	
Repeatable item. An arrow returning to the left above the main path of the horizontal line indicates an item that can be repeated. A character within the arrow means you must separate repeated items with that character. An arrow returning to the left above a group of repeatable items indicates that one of the items can be selected, or a single item can be repeated.	 
Fragment. The ┤ fragment ├ symbol indicates that a labelled group is described below the main syntax diagram. Syntax is occasionally broken into fragments if the inclusion of the fragment would overly complicate the main syntax diagram.	 **fragment:** 

How to send your comments

Your feedback is important in helping us to provide accurate, high-quality information. If you have comments about this document or any other Debug Tool documentation, contact us in one of these ways:

- Use the Online Readers' Comment Form at www.ibm.com/software/awdtools/rcf/. Be sure to include the name of the document, the publication number of the document, the version of Debug Tool, and, if applicable, the specific location (for example, page number) of the text that you are commenting on.
- Fill out the Readers' Comment Form at the back of this document, and return it by mail or give it to an IBM representative. If the form has been removed, address your comments to:

  IBM Corporation
  H150/090
  555 Bailey Avenue
  San Jose, CA 95141-1003
  USA
- Fax your comments to this U.S. number: (800)426-7773.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Summary of changes

This section lists the key changes made to Debug Tool for z/OS.

Changes introduced with Debug Tool V10.1

The following changes, if applicable, are marked with revision bars:

- A RESTful HTTP access interface to read, create, update, and delete profiles in the DTCN profile repository has been added. The interface is described in *Debug Tool API User's Guide and Reference*. An example of a GUI interface that uses the RESTful HTTP access interface to manipulate the profiles from the workstation is also available. How to download and install this example is described in *Debug Tool API User's Guide and Reference*.
- If you are using z/OS XLC C/C++, Version 1 Release 11, Debug Tool has added support for .mdbg files that contain source. The .mdbg file contains the debug information and a copy of the source needed in your debug session. You no longer need to have access to the source while debugging your program.

  If you are not familiar with .mdbg files and how to create them, see the topics "dbgld - Create a module map for debugging" or "CDADBGLD - Create a debug side file for the module map" in *z/OS XL C/C++ User's Guide*.

  You can indicate that Debug Tool always use .mdbg files to search for source and debug information by setting the EQAXOPT option MDBG to YES in the EQAOPTS options file. To learn how to set the MDBG option, see "MDBG" on page 90 in *Debug Tool Customization Guide*. In situations where you can specify environment variables, you can set the environment variable EQA_USE_MDBG to YES or NO, which overrides any setting (including the default setting) of the EQAXOPT MDBG option.

  To learn what compiler options to choose to create .dbg files, which the dbgld command or CDADBGLD utility use to build the .mdbg files, see the following topics in *Debug Tool User's Guide*:

  – "Choosing DEBUG compiler options for C programs"
  – "Choosing DEBUG compiler options for C++ programs"

  There are several different methods of specifying the location of .dbg and .mdbg files. The following list summarizes each method:

  – While you are debugging your program, you can use the following commands:
    - "SET DEFAULT DBG"
    - "SET DEFAULT MDBG"
    - Only for full screen, batch, and line mode: "SET MDBG"

    After you specify the location of the .dbg and .mdbg files, you can use the following commands to verify the location:
    - The `QUERY SET DEFAULT DBG` and `QUERY SET DEFAULT MDBG` commands, which are described in "QUERY".
    - Only for full screen, batch, and line mode: `QUERY SET MDBG`, which is described in "QUERY"

    All of these commands are described in *Debug Tool Reference and Messages*.
  – In your JCL, you can add EQADBG and EQAMDBG DD statements and specify the data set name of the corresponding .dbg or .mdbg file. To learn

more about using these DD statements, see "Compiling your program without using Debug Tool Utilities" in *Debug Tool User's Guide*. If you are debugging in UNIX System Services on in CICS, you cannot use these DD statements.

  - In UNIX System Services, you can use the following environment variables:
    - EQA_DBG_PATH
    - EQA_MDBG_PATH

- The term *full-screen mode through a VTAM® terminal* has been changed to *full-screen mode using a dedicated terminal*. The term was changed to remove the implication that any instructions that referred to VTAM terminal applied only to those terminals connected through an SNA network.
- You can now use IBM Session Manager while debugging in full-screen mode using a dedicated terminal and using the Terminal Interface Manager.

  The following topics have been added to or updated in the *Debug Tool Customization Guide*:

  - "Example: a debugging session using the Debug Tool Terminal Interface Manager" on page 25
  - "Starting the Debug Tool Terminal Interface Manager" on page 27
  - "Configuring Terminal Interface Manager as an IBM Session Manager application" on page 30

  In the *Debug Tool User's Guide*, the topic "Starting a debugging session in full-screen mode using a dedicated terminal" has been updated.
- The `CODEPAGE(ccsid)` option has been added to the `XML` option of the `LIST CONTAINER` and `LIST STORAGE` commands to improve the display of character strings encoded in an alternate code page on a 3270 terminal.

  In the *Debug Tool Reference and Messages* and *Debug Tool Reference Summary*, the descriptions and syntax diagrams of the following commands have been updated:

  - "LIST CONTAINER command"
  - "LIST STORAGE command"
- You can now add all the variables in the Working-Storage Section of a COBOL program to the Monitor window with one command.

  In *Debug Tool User's Guide*, the topic "Displaying the Working-Storage Section of a COBOL program in the Monitor window" has been added to describe how to add these variables to the Monitor window.

  In *Debug Tool Reference and Messages*, the topic "MONITOR command" has been updated to describe the new suboption `WSS`.

  In *Debug Tool Reference Summary*, the syntax diagram in "MONITOR command" has been updated to include the new suboption `WSS`.
- In full screen mode, a new window called the Command pop-up window has been added that makes it easier to enter and edit long commands.

  In *Debug Tool User's Guide*, the following topics have been added or updated:

  - "Command pop-up window"
  - "Opening the Command pop-up window to enter long Debug Tool commands"
  - "Entering multiline commands in full-screen"

  In *Debug Tool Reference Summary* and *Debug Tool Reference and Messages*, the following topics have been added:

  - "POPUP command (full screen mode only)"

- "SET POPUP command"

- You can now enter changes to multiple variables in the Monitor window at one time.

  In *Debug Tool User's Guide*, an item on the list in "Restrictions for modifying variables in the Monitor window" has been removed.

- Debug Tool now supports automatic saving and restoring of breakpoints and settings for IMS Transaction Manager (TM) programs.

  In *Debug Tool User's Guide*, phrases that describe this limitation have been removed from the following topics:

  - "Restoring Manually"
  - "Data sets used by Debug Tool"

  In *Debug Tool Reference and Messages*, phrases that describe this limitation have been removed from the following topics:

  - "SET RESTORE command"
  - "SET SAVE command"

- In *Debug Tool Reference and Messages*, the syntax diagram for "LIST *expression* command" has been updated to include the use of the `GROUP` option for COBOL programs.

- Two commands, `POSITION` and `FINDBP`, have been added to improve the ability to scroll to a specific line. You can use `POSITION integer`, which is similar to `SCROLL TO integer`, to scroll to a particular line or statement. `FINDBP`, which is similar to `FIND`, searches for line, statement, or offset breakpoints in the Source window.

  In *Debug Tool User's Guide*, the following topics have been updated to clarify how to scroll to a particular line:

  - "Scrolling to a particular line number"
  - "Displaying the line at which execution halted"

  In *Debug Tool Reference and Messages* and *Debug Tool Reference Summary*, the commands `FINDBP` and `POSITION` are described in topics "FINDBP command (full screen mode only)" and "POSITION command (full screen mode only)".

- The way to identify, in DTCN profiles, the program you want to debug has changed.

  Previously, you identified a program through the Program ID field. This has changed to two fields: LoadMod and CU.

  In *Debug Tool Reference Summary*, the following syntax diagrams have been updated to describe the new options:

  - "DISABLE command"
  - "ENABLE command"
  - "LIST DTCN or CADP command"
  - DTCNFORCELOADMODID, which is described in "EQAOPTS options"
  - DTCNFORCECUID, which is described in "EQAOPTS options"

  In *Debug Tool Reference and Messages*, the following syntax diagrams have been updated to describe the new options:

  - "DISABLE command"
  - "ENABLE command"
  - "LIST DTCN or CADP command"

  In *Debug Tool User's Guide*, the instructions in "Creating and storing a DTCN profile" have been updated to describe the new fields.

In *Debug Tool Customization Guide*, DTCNFORCELOADMODID and DTCNFORCECUID have been added to Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

- Debug Tool now supports running in browse mode. In this mode, you cannot make modifications to storage or registers, nor modify the control flow of a program with commands like `GOTO` and `JUMPTO`. You can debug a program, but you cannot change the behavior of a program. This might be useful when you want to debug a program running in a production environment but you want to prevent unauthorized changes to a program's behavior or production data.

  In *Debug Tool Reference Summary* and *Debug Tool Reference and Messages*, the description of the following commands have been updated to describe how you cannot use them in browse mode:

  - "ALLOCATE command"
  - "Assignment command (assembler and disassembly)"
  - "Assignment command (non-Language Environment COBOL)"
  - "Assignment command (PL/I)"
  - "CALL %CECI command"
  - "CALL *entry_name* (COBOL)"
  - "CALL %FM command"
  - "CALL %HOGAN command"
  - `CLEAR LOG` command, which is described in "CLEAR command"
  - "COMPUTE command"
  - "FREE command"
  - `GO BYPASS` command, which is described in "GO command"
  - "GOTO command"
  - "GOTO LABEL command"
  - "INPUT command"
  - "JUMPTO command"
  - "JUMPTO LABEL command"
  - "MEMORY command"
  - "MOVE command"
  - "QUIT command"
  - `QUIT` *`expression`* command, which is described in "QUIT command"
  - "QQUIT command"
  - "SET INTERCEPT command (C and C++)"
  - "SET INTERCEPT command (COBOL, full-screen mode, line mode, batch mode)"
  - "SET INTERCEPT command (COBOL, remote debug mode)"
  - "SET command (COBOL)"
  - "STORAGE command"
  - "SYSTEM command (z/OS)"
  - "TRIGGER command"
  - "TSO command"

  The topic "QUERY command" has been updated to describe the new option `BROWSE MODE`.

In *Debug Tool User's Guide*, the topic "Choosing a debugging mode" has been updated to describe how browse mode works and how you control browse mode.

In *Debug Tool Customization Guide*, the following topics have been added to describe the customization tasks you must do for this feature:

- Chapter 8, "Installing the browse mode RACF facility," on page 39
- "BROWSE" on page 84

- Debug Tool now supports displaying more than 1000 lines in the Monitor window.

  In *Debug Tool Reference and Messages*, the usage note that describes this limitation has been removed from "MONITOR command".

  In *Debug Tool Reference and Messages* and *Debug Tool Reference Summary*, the following topics have been updated:

  - The topic "SET MONITOR command" has been updated to describe the new option `LIMIT`.
  - The topic "QUERY command" has been updated to describe the new option `MONITOR LIMIT`.

  In the *Debug Tool User's Guide*, the topic "Monitor window" has been updated to describe how to increase the number of lines that the Monitor window displays and the implications of monitoring large volumes of data.
- Debug Tool now supports monitoring, by using the `AT CHANGE` command, of assembler variables with dynamically updated addresses such as those in a DSECT.

  In *Debug Tool Reference and Messages*, an existing usage note has been modified and a new usage note has been added to "AT CHANGE (full screen mode, line mode, batch mode)" that describes how Debug Tool monitors these variables.
- Debug Tool now supports debugging C and C++ programs that run in the Airline Control System (ALCS). This support is available only if you debug in remote debug mode.

  In the *Debug Tool User's Guide*, the following topics have been updated:

  - "A table that lists the supported subsystems" has been updated to indicate that Debug Tool supports the ALCS subsystem.
  - "Choosing TEST or NOTEST compiler suboptions for C programs" has been updated to indicate that if you want to debug C and C++ programs running in ALCS, you must compile them with hooks.
  - "Choosing TEST or NOTEST compiler options for C++ programs" has been updated to indicate that if you want to debug C and C++ programs running in ALCS, you must compile them with hooks.
  - "Choosing a debugging mode" has been updated to indicate for the ALCS subsystem, you must choose remote debug mode.
- Debug Tool now supports using the `L` and `M` prefix commands for assembler and disassembly programs.

  In the *Debug Tool User's Guide*, the following topics have been updated to describe how to use the L and M prefix commands on assembler and disassembly programs:

  - "Displaying the value of a variable"
  - "Entering prefix commands on specific lines or statements"
  - "Displaying and monitoring the value of a variable"
  - "One-time display of the value of variables"
  - "Adding variables to the Monitor window"

Some of these topics also describe a slight change in terminology. These topics use the word "operand" to mean a variable in C, C++, COBOL, or PL/I, or the operand of an assembler instruction.

In *Debug Tool Reference and Messages*, the topics "L prefix command (full-screen mode)" and "M prefix (full-screen mode)" have been updated to describe how to identify operands or variables on a statement, describe the limitations of this support, and show a new example.

- In the *Debug Tool User's Guide*, the topic "Quick Start guide for compiling and assembling programs for use with IBM Problem Determination Tools products", which helps you choose the compiler options that work for all Problem Determination Tools, has been added.
- In *Debug Tool Customization Guide*, all of the EQAOPTS options have been organized into one topic: Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83. This will help you keep track of the changes you are making to EQAOPTS so that you can make those changes at one time.

Chapter 1. Customizing Debug Tool: checklist

This topic helps you identify which customization tasks you must do. Begin by reviewing the topic "Planning your debug session" in the *Debug Tool User's Guide* with your application programmers and library system administrator. Reviewing that topic helps you gather the following information, which you need to identify which customization tasks you must do:

- Which version of compilers you are using
- Whether you are debugging DB2, DB2 stored procedures, CICS, and IMS programs
- Whether you are using full-screen mode, full-screen mode using a dedicated terminal, batch mode, or remote debug mode
- How your programs will call Debug Tool
- Whether you will be using Debug Tool Utilities, Coverage Utility, or Problem Determination Tools
- Whether you will need to modify some of Debug Tool's behavior

After you gather this information, review the following checklists. As you read each item on the checklist, you use the information you gathered to determine if you need to do that customization task. If the task is not applicable to your site, you can skip that task.

You must do all of the following required customization tasks:

__ • Chapter 2, "Product Registration," on page 5

__ • Chapter 3, "Installing the Dynamic Debug facility," on page 7.

__ • Chapter 4, "Setting up the APF-authorized system link list data set (SEQABMOD)," on page 11

__ • Chapter 5, "Setting up the link list data set (SEQAMOD)," on page 13

__ • Chapter 6, "Enabling debugging in full-screen mode using a dedicated terminal," on page 15.

__ • Chapter 7, "Specifying the TEST runtime options through the Language Environment user exit," on page 33.

If you are using Debug Tool Utilities, you must do the following required customization tasks:

__ • "Choosing a method to start Debug Tool Utilities" on page 41.

__ • "Customizing the data set names in EQASTART" on page 43.

__ • "Adding Debug Tool Utilities to the ISPF menu" on page 43.

__ • For the JCL for Batch Debugging utility, you must specify default values for the yb1dtmod and yb1dtbin parameters. See "Customizing for JCL for Batch Debugging utility" on page 44.

If you are using any of the following utilities in Debug Tool Utilities, you must do an additional customization task:

__ • If you are using Debug Tool Setup Utility, see "Customizing Debug Tool Setup Utility" on page 44.

__ • If you are using other Problem Determination Tools (File Manager for z/OS), see "Customizing Problem Determination Tools for multiple systems" on page 47.

__ • If you are using Program Preparation, see “Customizing Program Preparation” on page 47.

__ • If you are using Coverage Utility, see “Customizing Coverage Utility” on page 50.

If you are debugging DB2 stored procedures, CICS program, or IMS programs, you must do the following required customization tasks:

__ • If your site debugs DB2 stored procedures, see Chapter 10, “Preparing your environment to debug a DB2 stored procedures,” on page 55.

__ • If your site debugs CICS programs, see Chapter 11, “Adding support for debugging under CICS,” on page 57.

__ • If your site debugs IMS programs, see Chapter 12, “Adding support for debugging under IMS,” on page 77 and implement scenario A.

__ • If your site debugs non-Language Environment IMS programs, see Chapter 12, “Adding support for debugging under IMS,” on page 77 and implement scenario C.

As you review the rest of the checklist, if you need to do an item that requires a change to the EQAOPTS options file, record your selection on the form at the beginning of Chapter 14, “Defining EQAOPTS options: checklist and instructions,” on page 83. When you are done reviewing the checklist, you can make the all changes to EQAOPTS at one time as described in “Creating EQAOPTS load module” on page 97.

For any of the following situations, see “CODEPAGE” on page 85:

- Application programmers are debugging in remote debug mode and the source or compiler use a code page other than 037.

  If your C/C++ source contains square brackets or other special characters, you might need to specify a CODEPAGE option to override the Debug Tool default code page (037). Check the code page specified when you compiled your source. The C/C++ compiler uses a default code page of 1047 if you do not explicitly specify one. If the code page used is 1047 or a code page other than 037, you need to specify a CODEPAGE option specifying that code page.
- Application programmers are debugging in full screen mode and encounter one of the following situations:
  - They use the STORAGE command to update COBOL NATIONAL variables.
  - The source is coded in a code page other than 037.
- Application programmers use the `XML(CODEPAGE(ccsid))` option on a `LIST CONTAINER` or `LIST STORAGE` command to specify an alternate code page.

Do the customization tasks in the following list only if your site needs the features described:

__ • If your site uses z/OS XLC C/C++, Version 1 Release 11, and you want Debug Tool to retrieve source and debug information from .mdbg files, see “MDBG” on page 90.

__ • If you need to debug non-Language Environment programs that start under Language Environment or your site has any host products that might use SVC screening when Debug Tool is started, see “SVCSCREEN” on page 93.

__ • If your site debugs assembler programs and you want to control whether the statements that make up a macro are displayed in the Source window, see “DEFAULTVIEW” on page 88.

__ • If your site uses a library system that uses the SUBSYS allocation parameter and your application programmers debug C, C++, or Enterprise PL/I programs, review "SUBSYS" on page 92 to determine if you need to change the SUBSYS parameter.

__ • If your site needs to debug Q++ programs, see "EQAQPP" on page 89.

__ • If you want to use RACF® to enforce one of the following situations, do the instructions in Chapter 8, "Installing the browse mode RACF facility," on page 39 and then "BROWSE" on page 84:

  – Debug programs in a production environment (or some other environment) where you want to control whether Debug Tool users can modify the contents of storage or alter program flow

  – Restrict the use of Debug Tool to certain users

Do the customization tasks in the following list only if you want to modify the behavior described:

__ • If you want to reduce Debug Tool's CPU consumption in certain cases, see "CACHENUM" on page 85.

__ • If your site wants to change the default names, which are *userid*.DBGTOOL.SAVESETS and *userid*.DBGTOOL.SAVEBPS, of the data sets that store settings, breakpoints, and monitor values, see "SAVEBPDSN, SAVESETDSN" on page 92.

__ • To modify Debug Tool's behavior when a full-screen mode using a dedicated terminal or a remote debugger is not available, see "NODISPLAY" on page 91.

__ • If your site is using the EQAUEDAT user exit to direct Debug Tool to the location of source, listing, or separate debug files, see Chapter 13, "Enabling the EQAUEDAT user exit," on page 81.

__ • If your site wants to control the appearance or settings, through Debug Tool commands, of all debugging sessions, create a global preferences file. The global preferences file is a file that is processed at the beginning of every debugging session and contains Debug Tool commands. See "GPFDSN" on page 89 for instructions on how to create a global preferences file.

__ • If your site needs to issue a NAMES command for the initial load module or any of its compile units, see "NAMES" on page 90.

__ • If your site wants Debug Tool to suppress the prompt that Language Environment displays every time statements like STOP RUN, GOBACK, or EXEC CICS RETURN are run, see "THREADTERMCOND" on page 96. These statements can occur quite frequently in an application program, creating unnecessary interruptions for a user trying to debug the application program.

__ • If your site needs to change the defaults for NATLANG, LOCALE, or LINECOUNT, see "Changing the default and allowable values in EQACUIDF" on page 99.

If your site uses any of the following functions in a Japanese or Korean environment, see "Enabling additional languages for some Debug Tool components through EQACUIDF" on page 100:

- Debug Tool Utilities ISPF panels
- Debug Tool Coverage Utility
- EQANMDBG (non-CICS non-Language Environment support)

Chapter 2. Product Registration

You must ensure that a Product Registration has been done for Debug Tool. See the "Enable/Register Debug Tool" section of the *Program Directory for IBM Debug Tool for z/OS*.

Chapter 3. Installing the Dynamic Debug facility

The Dynamic Debug facility enables the user to debug the following types of programs and code:

- Programs compiled with the `TEST(NOHOOK)` compiler option and the Enterprise PL/I for z/OS Version 3 Release 4 compiler.
- Program compiled with the `TEST(NOHOOK)` compiler option and the Enterprise COBOL for z/OS, Version 4.1 compiler.
- Programs compiled with the `TEST(NONE)` compiler option and one of the following compilers:
  - Enterprise COBOL for z/OS and OS/390, Version 3
  - COBOL for OS/390 & VM, Version 2 Release 2
  - COBOL for OS/390 & VM, Version 2 Release 1 with APAR PQ40298 installed
- Programs for which no debug data is available by using the disassembly view.
- Assembler code that complies with the requirements described in *Debug Tool User's Guide*.
- Load modules loaded by using the MVS LOAD and LINK macros.
- Programs that do not run under the Language Environment, including non-Language Environment COBOL programs.
- Programs compiled with the suboption of the `TEST` compiler option that adds compiled in hooks and with one of the following compilers:
  - Any COBOL compiler supported by Debug Tool
  - Any PL/I compiler supported by Debug Tool
  - Any C/C++ compiler supported by Debug Tool

  The Dynamic Debug facility provides performance enhancements for these programs.
- You create DTCN profiles to debug a CICS task that has already started.

The Dynamic Debug facility requires the installation of the Dynamic Debug facility SVC programs `EQA00SVC(IGC0014E)` and `EQA01SVC(IGX00051)`:

- `EQA00SVC` is a type 3 SVC with a reserved number of 145 (x'91').
- `EQA01SVC` is a type 3 using SVC number 109 (X'6D') with function code 51.

The Dynamic Debug facility SVCs from this version of Debug Tool are compatible with all previous releases of Debug Tool to Debug Tool for z/OS, Version 6 Release 1 (Program Number 5655-P14).

To install the SVCs, you can select one or both of the following alternatives:

- Install the SVCs through a system IPL. The SMP/E APPLY operation, which you run when you install Debug Tool or apply a PTF, updates the library *hlq*`.SEQALPA` with the SVCs. To place *hlq*`.SEQALPA` in the LPA list, add it to an LPALST*xx* member of parmlib that is used for IPL. If you have earlier releases of Debug Tool installed at your site, remove any other SEQALPA data sets. The next time you IPL your system, the SVCs are automatically installed.

  Check SYS1.LPALIB for the following members and, if you find them, remove them:

  - EQA00SVC

– EQA01SVC
– IGC0014E (ALIAS of EQA00SVC)
– IGX00051 (ALIAS of EQA01SVC)

These members might have been placed there by previous installations of Debug Tool. Because SYS1.LPALIB is always searched before the data sets in LPALST*xx*, these older members would be found before the newer members in LPALST*xx*.

- Install the SVCs without a system IPL. The SMP/E APPLY operation, which you run when you install Debug Tool or apply a PTF, updates the library *hlq*.SEQAAUTH with the SVCs and the dynamic SVC installer. See "Installing the SVCs without using a system IPL" for information about how to immediately install or update the SVCs.

Installing the SVCs without using a system IPL

To install the Dynamic Debug facility SVCs without using a system IPL (referred to as a dynamic installation), perform the following steps:

1. Mark the *hlq*.SEQAAUTH data set as APF-authorized[1]. This data set contains SVC installation programs; therefore, access to it must be limited to system programmers.
2. Update both places in the SVC dynamic install job EQAWISVC (shipped as a member of the data set *hlq*.SEQASAMP) with the fully qualified name for the Debug Tool *hlq*.SEQAAUTH data set. Eye-catchers (<<<<<) in the job highlight the statements that require changing. You might also need to update the job card.
3. Submit the job. The job installs both SVCs. After the job is completed, verify that the return code is 00 (RC=00).

Verifying the installation of the SVCs

To verify the installation of the SVCs, you need to check the level of the Dynamic Debug facility SVCs, then run the installation verification programs.

Checking the level of the Dynamic Debug facility SVCs

Display the level of the Dynamic Debug facility SVCs installed by entering the following command:

```
EXEC 'hlq.SEQAEXEC(EQADTSVC)'
```

Information about EQA00SVC that is similar to the following is displayed. Verify that the version and compile date that are displayed are the same or higher than what is shown here.

```
| x4.y.EQA00SVC  2009.277Licensed Materials - Property of IBM 5655-V50 Debug Tool Version 05 EQA00SVC-C7572Copyright  Copyright
| IBM Corp. All Rights Reserved
| ***> EQA00SVC is Version 05 with compile date 4 Oct 2009
```

Information about EQA01SVC that is similar to the following is displayed. Verify that the version and compile date that are displayed are the same or higher than what is shown here.

```
| x4.y.EQA01SVC  2009.2772Licensed Materials - Property of IBM 5655-V50 Debug Tool Version 08 EQA01SVC-C7572Copyright  Copyright I
| BM Corp. All Rights ReservedU
| ***> EQA01SVC is Version 08 with compile date 4 Oct 2009
```

1. To APF-authorize a data set, add an APF ADD statement for the data set to a PROG*xx* member of parmlib that is used for IPL. To immediately APF-authorize the data set, use the SETPROG APF MVS command.

```
x4.y.EQA01SV2  2009.277Licensed Materials - Property of IBM 5655-V50 Debug Tool Version 03 EQA01SV2-C7572Copyright  Copyright I
BM Corp. All Rights ReservedU
***> EQA01SV2 is Version 03 with compile date 4 Oct 2009
```

Running the installation verification programs

To help you verify the installation of the Dynamic Debug facility (that the SVCs are installed and working correctly), the *hlq*.SEQASAMP data set contains installation verification programs (IVPs) in the following members. Run the IVPs that are appropriate for the tasks that your users will be performing. Before you run any IVP, customize it for your installation as described in the member.

Table 2. Name of the installation verification program and the programming language corresponding to that installation verification program.

IVP	Task
EQAWIVP4	COBOL TEST(NONE,SYM) or TEST(NOHOOK)
EQAWIVPF	PL/I TEST(ALL,SYM,NOHOOK)
EQAWIVPI	Enterprise PL/I TEST(ALL,SYM,NOHOOK,SEPARATE)
EQAWIVPP	COBOL TEST(NONE,SYM,SEPARATE) or TEST(NOHOOK,SEPARATE)
EQAWIVPS	disassembly
EQAWIVPA	Language Environment assembler
EQAWIVPC	non-Language Environment assembler
EQAWIVPV	OS/VS COBOL
EQAWIVPX	non-Language Environment VS COBOL II

Using the Authorized Debug facility for protected programs

If your users need to use the Dynamic Debug facility to debug programs that are loaded into protected storage (located in subpool 251 or 252), your security administrator must authorize those users to use the Authorized Debug facility. Examples of reentrant programs that are loaded into protected storage are:

- Re-entrant programs loaded from an APF authorized library by MVS
- Programs loaded by CICS into RDSA or ERDSA because RENTPGM=PROTECT

Important: Before you do this task, you must have installed and verified the SVCs.

To authorize users to use the Authorized Debug facility:

1. Establish a profile for the Authorized Debug Facility in the FACILITY class by entering the RDEFINE command:

   ```
   RDEFINE FACILITY EQADTOOL.AUTHDEBUG UACC(NONE)
   ```

2. Verify that generic profile checking is in effect for the class FACILITY by entering the following command:

   ```
   SETROPTS GENERIC(FACILITY)
   ```

3. Give a user permission to use the Authorized Debug Facility by entering the following command, where *DUSER1* is the name of a RACF-defined user or group profile:

   ```
   PERMIT EQADTOOL.AUTHDEBUG CLASS(FACILITY) ID(DUSER1) ACCESS(READ)
   ```

Instead of connecting individual users, the security administrator can specify *DUSER1* to be a RACF group profile and then connect authorized users to the group.

In CICS, Debug Tool checks that the region user ID is authorized instead of an individual CICS user ID.

4. If the FACILITY class is not active, activate the class by entering the SETROPTS command:

   ```
   SETROPTS CLASSACT(FACILITY)
   ```

   Issue the SETROPTS LIST command to verify that FACILITY class is active.

5. Refresh the FACILITY class by issuing the SETROPTS RACLIST command:

   ```
   SETROPTS RACLIST(FACILITY) REFRESH
   ```

Chapter 4. Setting up the APF-authorized system link list data set (SEQABMOD)

You must make certain Debug Tool load modules available in an APF-authorized data set that is in the system link list concatenation. You can do this in one of the following ways, depending on your site policy:

- Mark and add the load modules by doing the following steps:
  1. Mark the *hlq*.SEQABMOD data set as APF-authorized.[1]
  2. Add the data set to the system link list concatenation.[2]
  3. If you have earlier releases of Debug Tool installed, remove any other SEQABMOD data sets.
  4. Do an LLA refresh to make the members in *hlq*.SEQABMOD available to Debug Tool.
- Copy the load modules and refresh the members by doing the following steps:
  1. Copy[3] all the members of the *hlq*.SEQABMOD data set into an existing APF-authorized system link list data set.
  2. Do an LLA refresh to make these members available to Debug Tool.

2. To add a data set to the link list, add a LNKLST ADD statement for the data set to a PROG*xx* member of parmlib that is used for IPL. To immediately add a data set to the link list, use the SETPROG LNKLST MVS command. Then, if the link list data set is managed by LLA, enter a F,LLA REFRESH MVS command to refresh the Library Lookaside Directories.

3. If you do this copy, you must repeat this copy after you apply any service to Debug Tool. SMP/E does not do this copy for you.

Chapter 5. Setting up the link list data set (SEQAMOD)

The *hlq*.SEQAMOD data set must be in the load module search path whenever you debug a program with Debug Tool. Except for two cases, it will be convenient for your users if you put *hlq*.SEQAMOD in the system link list concatenation. The exceptions are:

- CICS, where *hlq*.SEQAMOD must be placed in the DFHRPL concatenation. See Chapter 11, "Adding support for debugging under CICS," on page 57.
- When the Debug Tool Setup Utility component of the Debug Tool Utilities ISPF function is used to start the debugging session (where DTSU accesses *hlq*.SEQAMOD for you).

In all other cases, unless you put *hlq*.SEQAMOD in the system link list concatenation, the user will have to alter the execution environment of any program being debugged so that *hlq*.SEQAMOD is in the load module search path (such as placing it in JOBLIB, STEPLIB, ISPLLIB or via use of TSOLIB). Therefore, it is recommended that you add the *hlq*.SEQAMOD data set to the system link list concatenation[2].

Chapter 6. Enabling debugging in full-screen mode using a dedicated terminal

To enable users to debug the following types of programs while using a 3270-type terminal, you need to enable full-screen mode using a dedicated terminal:

- Batch programs
- TSO programs (using a separate terminal for debugging)
- Programs running under UNIX System Services
- DB2 stored procedures
- IMS programs

A dedicated terminal has specific set up requirements so that it can interact with Debug Tool in these environments. Thus, the terminal is dedicated for use by Debug Tool. Users do not typically use it to access other services.

How Debug Tool uses full-screen mode using a dedicated terminal

The following steps describe how a user would start a debugging session for a batch job using full-screen mode using a dedicated terminal. Study these steps to understand how Debug Tool uses full-screen mode using a dedicated terminal and to understand why you need to do the configuration steps described in "Enabling full-screen mode using a dedicated terminal" on page 16.

1. Start two terminal emulator sessions. Connect the second session to a terminal LU that can handle a full-screen mode using a dedicated terminal.
2. On the first terminal emulator session, log on to TSO.
3. Note the LU name (`LU_name`) to which the second terminal emulator session is connected.
4. Make following changes to the `PARM` string in the batch job that starts your debugging session:
   - Specify the `TEST` run time option in the following format:

     ```
     TEST(,,MFI%LU_name:*)
     ```

     LU_name is the LU name you noted in step 3.

     If your site requires that you specify the VTAM network identifier (`NETID`), specify the `TEST` run time option in the following format:

     ```
     TEST(,,MFI%NETID.LU_name:*)
     ```

     NETID identifies the network in which the second terminal emulator resides. For example, in the string `NETA.LU001`, `NETA` is the *NETID*.
5. Submit the batch job. Debug Tool completes the following tasks:
   a. Debug Tool allocates a VTAM `ACB (EQAMVnnn)` for its end of a VTAM session.
   b. Debug Tool uses VTAM to initiate a session with the terminal LU to which the second terminal emulator is connected.
   c. A VTAM session is then conducted between Debug Tool and the terminal LU.

The user does not log on to any host application through the second terminal emulator. Debug Tool initiates the connection between itself and that second terminal LU.

6. On the second terminal emulator, the emulator displays a full-screen mode debugging session. Interact with it in the same way you would with any other full-screen mode debugging session.

This technique requires you to define and configure a number of items in the z/OS Communications Server. Section "Enabling full-screen mode using a dedicated terminal" describes these definitions and configuration.

Enabling full-screen mode using a dedicated terminal

To enable full-screen mode using a dedicated terminal, do the following steps:

1. Define the VTAM APPL definition statements that Debug Tool uses for its end of the session, as described in "Defining the VTAM EQAMVnnn APPL definition statements."
2. Define the terminal LUs used by Debug Tool, as described in "Defining terminal LUs used by Debug Tool" on page 18.
3. If your terminals are connected through a SNA network, you are done. If your terminals are connected through a TN3270 network, you must continue.
4. If a TN3270 server manages the terminal, configure the TN3270 Telnet Server, as described in "Configuring the TN3270 Telnet Server to access the terminal LUs" on page 19.
5. Verify the installation of the facility to debug programs in full-screen mode using a dedicated terminal, as described in "Verifying the customization of the facility to debug full-screen mode using a dedicated terminal" on page 24.

Defining the VTAM EQAMVnnn APPL definition statements

You must define the APPL definition statements that Debug Tool uses for its end of the VTAM session with the terminal LU. You can define up to 999 APPLs for Debug Tool. You can define an APPL by using one of the following naming conventions:

- Define each APPL with the following naming convention: the first five characters of the APPL name must be `EQAMV` and the last three characters must be consecutive three digit numbers, starting with 001. Do not code an `ACBNAME` operand on the APPL definition statements for this method.
- Define each APPL name with the naming convention you use at your site. Code an `ACBNAME` operand on the APPL definition statement that uses `EQAMV` as the first five characters, and three numeric digits (starting with 001) as the last three characters.

Tip: The `EQAMV`*`nnn`* names are used internally by Debug Tool. Do not confuse these names with the terminal LU names. The user needs to know only the terminal LU name, which he specifies with the `MFI%` suboption of the `TEST` run time option.

The number of APPL names you define must be sufficient to allow for the maximum number of concurrent Debug Tool full-screen mode using a dedicated terminal sessions. (Debug Tool uses one of these APPL names for its end of each VTAM session that is initiated with a terminal LU.)

The descriptions and examples used in this book assume you defined APPL names by using the EQAMV*nnn* naming convention. Debug Tool uses the EQAMV*nnn* names for internal processing.

The EQAWAPPL member in the *hlq*.SEQASAMP data set predefines 50 APPL names, EQAMV001 to EQAMV050. You can do one of the following tasks to add this member to the VTAM definitions library (VTAMLST).

- Copy EQAWAPPL into a new member:
  1. Create a new member in the VTAM definitions library (VTAMLST). The VTAM definitions library is often stored in the data set SYS1.VTAMLST.
  2. Copy the contents of the EQAWAPPL member into the new member.
  3. Add the new member's name to the VTAM start options configuration file, ATCCON*xx*, so that VTAM activates the Debug Tool APPL definitions at initialization.
- Copy EQAWAPPL into an existing member that is already defined in VTAMLST:
  1. Select a member in the VTAM definitions library (VTAMLST) that contains the major node definitions.
  2. Copy the APPL definition statements for Debug Tool from the EQAWAPPL member into the selected member.

     Tip: The existing member has the VBUILD TYPE=APPL statement, so do not copy this statement from EQAWAPPL.

If you are running VTAM in a multi-domain environment and you require the ability to debug full-screen mode using a dedicated terminal on more than one host, edit the copy of EQAWAPPL on each system to make the names for Debug Tool major and minor nodes unique for each system.

For example, if you have hosts SYSA, SYSB, and SYSC, and need to provide definitions for up to 50 concurrent users debugging programs in full-screen mode using a dedicated terminal on each system, you can code the following entries:

- SYSA VTAMLST EQAWAPPL entry:

```
EQAAPPLA VBUILD TYPE=APPL
EQAMV001 APPL  AUTH=(PASS,ACQ),PARSESS=NO
EQAMV002 APPL  AUTH=(PASS,ACQ),PARSESS=NO
...
EQAMV050 APPL  AUTH=(PASS,ACQ),PARSESS=NO
```

- SYSB VTAMLST EQAWAPPL entry:

```
EQAAPPLB VBUILD TYPE=APPL
EQAMV051 APPL  AUTH=(PASS,ACQ),PARSESS=NO
EQAMV052 APPL  AUTH=(PASS,ACQ),PARSESS=NO
...
EQAMV100 APPL  AUTH=(PASS,ACQ),PARSESS=NO
```

- SYSC VTAMLST EQAWAPPL entry:

```
EQAAPPLC VBUILD TYPE=APPL
EQAMV101 APPL  AUTH=(PASS,ACQ),PARSESS=NO
EQAMV102 APPL  AUTH=(PASS,ACQ),PARSESS=NO
...
EQAMV150 APPL  AUTH=(PASS,ACQ),PARSESS=NO
```

You can have up to 999 unique APPL names for full-screen mode using a dedicated terminal spread across your network.

As an alternative to coding each minor node name, you can use the Model Application Names function. With this function, VTAM dynamically creates the

minor nodes. Use one of the following ways (alter these examples, if needed, to maintain unique names per system as discussed in "Defining the VTAM EQAMVnnn APPL definition statements" on page 16):

- `EQAMV??? APPL AUTH=(PASS,ACQ),PARSESS=NO`
- `ABCDE??? APPL AUTH=(PASS,ACQ),PARSESS=NO,ACBNAME=EQAMV???`

Activating the VTAM EQAMVnnn APPLs

Activate the VTAM APPLs by entering the following command from the console, where *member-name* is the member name in the VTAM library (`VTAMLST`):

```
VARY NET,ACT,ID=member-name
```

Defining terminal LUs used by Debug Tool

The terminal LUs used by Debug Tool in full-screen mode using a dedicated terminal must meet the requirements specified in the following sections:

"Terminal LU specifications"

"Terminal LU state requirements" on page 19

Terminal LU specifications

All terminal LUs that are used to debug programs in full-screen mode using a dedicated terminal must have a default log mode specified in the corresponding VTAM definitions. This log mode must match the characteristics of the terminal emulator session that is attached to this terminal LU. Use the `DLOGMOD=` operand on the APPL definition for the terminal logical unit (LU) to specify the default log mode.

To support the widest range of terminal characteristics, we recommend you use a `DLOGMOD` specification of D4C32XX3, in the IBM supplied `MODETAB` of `ISTINCLM`. If you use a `DLOGMOD` specification of D4C32XX3, you must use a TN3270E emulator that responds to a VTAM query with terminal characteristics, such as size, color, and extended graphics.

If your terminal emulator session cannot provide this information, select a log mode that matches your terminal emulator session characteristics. For example, if you have a TN3270 emulator that does not respond to a query, select one of the following log modes that matches the terminal size that the user will be using:

- D4C32782 24x80
- D4C32783 32x80
- D4C32784 43x80
- D4C32785 27x132

When you specify these types of log modes, the user must select a terminal size that matches your `DLOGMOD` specification.

An example of a set of terminal LU definitions for the terminal side of the VTAM session is *hlq*`.SEQASAMP(EQAWTRML)`. See the log mode definitions in the *IBM Communications Server SNA Resource Definition Reference* for further information about log modes. The `MODETAB` log mode table load module that contains the `DLOGMOD` default log mode specification must be available to VTAM via the VTAMLIB DD statement.

You need to `VARY` on these new terminal LU definitions, similar to the way it was done in "Activating the VTAM EQAMVnnn APPLs."

Terminal LU state requirements

When Debug Tool accesses the terminal LU, the terminal LU must be in the following state:

- It must be known to the z/OS Communications Server on the system which Debug Tool runs.
- It must be marked secondary logical unit (SLU) enabled.
- It must not be in session with any application.

You can determine whether a particular terminal LU meets these criteria by using the `DISPLAY` VTAM operator command:

1. Access the desired LU using your terminal emulator, and exit any session manager.
2. On your system console, enter the following command, where *name* is the LU name:

   ```
   DISPLAY NET,ID=name,SCOPE=ALL
   ```

3. Inspect the output of the command for the following information:
   - The IST486I message indicates `STATUS=ACTIV` and `DESIRED STATE=ACTIV`, and an IST172I NO SESSIONS EXIST message is displayed.
   - The IST597I message indicates SLU ENABLED.
   - The IST934I message indicates that a DLOGMOD was specified.

Configuring the TN3270 Telnet Server to access the terminal LUs

If you use the IBM Communications Server for z/OS TN3270 Telnet Server to manage your terminals, you must configure TN3270 Telnet Server to support terminals with the following characters:

- Terminal LUs that have a proper DLOGMOD specified must be accessed.
- The LUMAP KEEPOPEN statement needs to be specified, so that VTAM allocates the ACB for the terminal LU when a terminal emulator session is connected to it, rather than only when an application is started.
- The terminal LU name must be available to the user of the terminal emulator session.

One way to enable this support is to set up a new TN3270 telnet port. The following instructions guide you through setting up a new port and the changes you must make to the PROFILE.TCPIP data set. The examples in "Configuring the TN3270 Telnet Server" on page 22 show several variations of this support.

1. Select an unused port, such as 2023. If you have a firewall installed, ensure that this port is allowed through the firewall.
2. Do one of the following steps:
   - If you are running the TN3270 Telnet Server in the TCP/IP address space, specify a `PORT` *num* `TCP INTCLIEN` statement to reserve the new port for the TN3270 Telnet Server.
   - If you are running the TN3270 Telnet Server in a separate address space (optional on z/OS Communications Server Version 1.6 through 1.8, required on Version 1.9 or later), specify a `PORT` *num* `TCP` *jobname* `NOAUTOLOG` statement to reserve the new port for the TN3270 Telnet Server.
3. Create a new set of TELNETPARMS and BEGINVTAM blocks for the new port by copying the existing TELNETPARMS and BEGINVTAM blocks for port 23.

4. Customize the new TELNETPARMS and BEGINVTAM blocks to use this new port number. Ensure that the previous TELNETPARMS and BEGINVTAM blocks also specify a port number (typically 23).
5. Make the following changes to your new BEGINVTAM block:
   a. If you intend to use this new port for only Debug Tool in full-screen mode using a dedicated terminal, you can remove all the statements from the BEGINVTAM block that you created in step 3 on page 19, except the PORT statement. Go to step 5c.
   b. Remove any copied DEFAULTLUS, DEFAULTLUSSPEC, DEFAULTAPPL and LUMAP statements.
   c. Specify a new LUGROUP specification that indicates which terminal LUs that will be used as dedicated terminals for debugging in full-screen mode using a dedicated terminal. These terminal LUs must have a `DLOGMOD` specification in their APPL definition statement.
   d. Specify some client_identification statements (such as HNGROUP and IPGROUP).
   e. Specify a new LUMAP statement with KEEPOPEN (along with the proper LU group operand and client_identification operand).

      The KEEPOPEN operand forces the TN3270 Telnet Server to keep the access control block (ACB) for the LU open at all times (for those LUs affected by this LUMAP statement). With the ACB open, Debug Tool can acquire the LU if the LU is connected to a client terminal emulator session but is not in session.
   f. Specify a new ALLOWAPPL EQAMV* statement (or ALLOWAPPL * if site policies allow it) in the BEGINVTAM block to let Debug Tool start a session with the terminal LU.

      If you defined the name that Debug Tool uses for its side of the VTAM session with a name other than `EQAMV`*`nnn`*, then you should specify that name on the ALLOWAPPL statement, rather than `EQAMV`*`nnn`*. (Or just use * if your site policies allow it.)
   g. Specify whether the terminal is to display a session manager panel, a USSMSG10 panel, or a Telnet Solicitor Logon panel.

      The user must know what terminal LU they have acquired when they connect their terminal emulator session to this new port. If you normally use a session manager that displays the terminal LU, then you can continue to use that method. Otherwise, use one of the following panels:

      - A modified USSMSG10 panel that displays the terminal LU name
      - The Telnet Solicitor Logon panel, if the terminal emulator itself shows the terminal LU name

      To specify which panel is to be displayed, do the following steps:

      1) To display a session manager panel, specify the FIRSTONLY operand on a DEFAULTAPPL statement that defines the session manager to run. To use the LU to debug a program in full-screen mode using a dedicated terminal, the user must first exit the session manager panel and return to the Telnet Solicitor Logon panel.
      2) To display a USSMSG10 panel, specify a USSTCP statement. If your terminal emulator session supports the TN3270E protocol, the USSMSG10 panel can be customized to display the terminal LU name. See the *IBM Communication Server IP Configuration Reference* manual for information about how to create a new USS table load module that contains a USSMSG10 panel which includes the `@@LUNAME` parameter.
      3) To display a Telnet Solicitor Logon panel, code no additional statements.

If you want to restrict access for a terminal connected to this new port so that no one can use it to start any application and that no application other than Debug Tool can acquire it, then do the following steps:

1. Remove any statements from the port's BEGINVTAM block other than those recommended above.
2. Write only one ALLOWAPPL statement, specifying EQAMV*nnn* or, if you didn't use EQAMV*nnn*, the minor node name that Debug Tool uses for its side of the VTAM session.
3. Use the USSMSG10 panel or Telnet Solicitor Logon Panel display method.

After you make these changes to the TCP/IP configuration data set, you must instruct TCP/IP to use this updated definition and start the new port. The Telnet server uses the VARY command to change Telnet functions. One of the following commands can help you change Telnet functions:

VARY TCPIP,,OBEYFILE
To start, restart or change a port by updating the Telnet profile. If you are running a TN3270 Telnet Server in a separate address space, you need to include the TN3270 Telnet Server *jobname* in the command. For example, VARY TCPIP,*jobname*,OBEYFILE.

VARY TCPIP,,TELNET,STOP and VARY TCPIP,,OBEYFILE
To stop a Telnet port, and then restart that port or a new port without stopping the TCP/IP stack.

See *IBM Communication Server IP Configuration Reference* for more information about the VARY TCPIP command.

After making these changes, your users can set up a unique terminal emulator session that connects to this new port, and debug programs that require the use of full-screen mode using a dedicated terminal.

Example: Activating full-screen mode using a dedicated terminal when using TCP/IP TN3270 Telnet Server

The examples below describe how to define the Debug Tool minor node names, define the terminal LUs for use by Debug Tool, and three ways to define Telnet ports that the TN3270 Telnet server can use.

After you code these definitions, you need activate these changes by using the VARY NET and VARY TCPIP commands as described previously.

Defining Debug Tool to VTAM

These are the Debug Tool minor node names defined to VTAM through VTAMLST:

```
EQAAPPL  VBUILD TYPE=APPL
EQAMV001 APPL  AUTH=(PASS,ACQ),PARSESS=NO
EQAMV002 APPL  AUTH=(PASS,ACQ),PARSESS=NO
EQAMV003 APPL  AUTH=(PASS,ACQ),PARSESS=NO
EQAMV004 APPL  AUTH=(PASS,ACQ),PARSESS=NO
EQAMV005 APPL  AUTH=(PASS,ACQ),PARSESS=NO
...
EQAMV050 APPL  AUTH=(PASS,ACQ),PARSESS=NO
```

See *hlq*.SEQASAMP(EQAWAPPL) for a sample of these definitions.

Defining the terminals used by Debug Tool

These are the terminal LUs defined to VTAM through VTAMLST:

```
EQATRML  VBUILD TYPE=APPL
TRMLU001 APPL AUTH=NVPACE,EAS=1,PARSESS=NO,MODETAB=ISTINCLM,          *
               SESSLIM=YES,DLOGMOD=D4C32XX3
TRMLU002 APPL AUTH=NVPACE,EAS=1,PARSESS=NO,MODETAB=ISTINCLM,          *
               SESSLIM=YES,DLOGMOD=D4C32XX3
TRMLU003 APPL AUTH=NVPACE,EAS=1,PARSESS=NO,MODETAB=ISTINCLM,          *
               SESSLIM=YES,DLOGMOD=D4C32XX3
TRMLU004 APPL AUTH=NVPACE,EAS=1,PARSESS=NO,MODETAB=ISTINCLM,          *
               SESSLIM=YES,DLOGMOD=D4C32XX3
TRMLU005 APPL AUTH=NVPACE,EAS=1,PARSESS=NO,MODETAB=ISTINCLM,          *
               SESSLIM=YES,DLOGMOD=D4C32XX3
...

TRMLU050 APPL AUTH=NVPACE,EAS=1,PARSESS=NO,MODETAB=ISTINCLM,          *
               SESSLIM=YES,DLOGMOD=D4C32XX3
```

See *hlq*.SEQASAMP(EQAWTRML) for a sample of these definitions.

Note that the DLOGMOD operand is specified. Change the TRMLU*nnn* names on the terminal LU APPL definition statements to names that meet your site convention for terminal LU names. These names must match the entries in the LUGROUP statements in the BEGINVTAM blocks shown in "Example 1," "Example 2" on page 23, and "Example 3" on page 23.

Configuring the TN3270 Telnet Server

The examples below highlight the changes made to the TCP/IP TN3270 server's configuration file.

Example 1

The example defines a new port (2023). When a user connects a terminal emulator session to this port, the Netview Access Services (NVAS) menu appears when the LU is created. The user copies the LU name that appears on the NVAS screen and specifies it as the value for the MFI%*LU_name* suboption of the TEST run-time option. After the user copies the LU name, the user exits NVAS and returns to the Telnet Solicitor Logon panel to make the terminal LU available to Debug Tool.

Each change is highlighted with a number in **reverse highlighting**. This number corresponds to the step number in the list of instructions in "Configuring the TN3270 Telnet Server to access the terminal LUs" on page 19.

```
PORT
   ...
 2  2023 TCP INTCLIEN            ; Telnet Server - Debug Tool
   ...

;
; Define Telnet pool for Debug Tool
;
TELNETPARMS
 4   PORT 2023
   ... the rest of this should be a copy of port 23
ENDTELNETPARMS

BEGINVTAM
 4    PORT 2023


   LUGROUP DBGTOOL
 5c       TRMLU001..TRMLU050
```

```
   ENDLUGROUP

   IPGROUP EVERYONE
 5d       0.0.0.0:0.0.0.0
   ENDIPGROUP

 5g1 DEFAULTAPPL NVAS FIRSTONLY
 5e LUMAP DBGTOOL EVERYONE KEEPOPEN
 5f ALLOWAPPL   EQAMV*
ENDVTAM
```

See *hlq*.SEQASAMP(EQAWTTS1) for a sample of these definitions.

Example 2

The example defines a new port (2023). When a user connects a terminal emulator session to this port, a USSMSG10 panel is displayed. The USSTCP statement is coded to point to a customized USSMSG10 panel that you defined that displays the LU name. The user copies this LU name and assigns it to the MFI*%LU_name* suboption of the TEST run time option. When the USSMSG10 panel is displayed, the terminal LU is available to Debug Tool.

Each change is highlighted with a number in **reverse highlighting**. This number corresponds to the step number in the list of instructions in "Configuring the TN3270 Telnet Server to access the terminal LUs" on page 19.

```
PORT
   ...
 2  2023 TCP INTCLIEN            ; Telnet Server - Debug Tool
   ...

;
; Define Telnet pool for Debug Tool
;
TELNETPARMS
 4   PORT 2023
   ... the rest of this should be a copy of port 23
ENDTELNETPARMS

BEGINVTAM
 4    PORT 2023


   LUGROUP DBGTOOL
 5c       TRMLU001..TRMLU050
   ENDLUGROUP

   IPGROUP EVERYONE
 5d       0.0.0.0:0.0.0.0
   ENDIPGROUP

 5g2 USSTCP USS$EQAW EVERYONE
 5e LUMAP DBGTOOL EVERYONE KEEPOPEN
 5f ALLOWAPPL   EQAMV*
ENDVTAM
```

See *hlq*.SEQASAMP(EQAWTTS2) for a sample of these definitions.

Example 3

The example defines a new port (2023). When the user connects a terminal emulator session to this port, the Telnet Solicitor Logon panel is displayed, and the terminal LU is available to Debug Tool. The user copies the LU name from the terminal emulator session's information area and assigns it to the MFI*%LU_name* suboption of the TEST run time option.

Each change is highlighted with a number in **reverse highlighting**. This number corresponds to the step number in the list of instructions in "Configuring the TN3270 Telnet Server to access the terminal LUs" on page 19.

```
PORT
   ...
 2  2023 TCP INTCLIEN            ; Telnet Server - Debug Tool
   ...

;
; Define Telnet pool for Debug Tool
;
TELNETPARMS
 4   PORT 2023
   ... the rest of this should be a copy of port 23
ENDTELNETPARMS

BEGINVTAM
 4    PORT 2023


   LUGROUP DBGTOOL
 5c       TRMLU001..TRMLU050
   ENDLUGROUP

   IPGROUP EVERYONE
 5d       0.0.0.0:0.0.0.0
   ENDIPGROUP

 5e LUMAP DBGTOOL EVERYONE KEEPOPEN
 5f ALLOWAPPL   EQAMV*
ENDVTAM
```

See *hlq*.SEQASAMP(EQAWTTS3) for a sample of these definitions.

Verifying the customization of the facility to debug full-screen mode using a dedicated terminal

Connect a terminal emulator session to one of the terminal LUs setup as described previously in this chapter. Issue the DISPLAY command from your system console as shown in "Terminal LU state requirements" on page 19. Verify that the output of the DISPLAY command is correct. If the output of the DISPLAY command is not correct, you must review every step in "Enabling full-screen mode using a dedicated terminal" on page 16 and verify that you completed each step correctly. Then run one of the install verification jobs described below.

To help you verify the installation of the facility to debug full-screen mode using a dedicated terminal, the *hlq*.SEQASAMP data set contains the following installation verification program (IVP) jobs:

- EQAWIVP5 (COBOL)
- EQAWIVP6 (C)
- EQAWIVP7 (PL/I)
- EQAWIVP9 (Enterprise PL/I)
- EQAWIVPB (Language Environment assembler)
- EQAWIVPD (non-Language Environment assembler)
- EQAWIVPW (OS/VS COBOL)
- EQAWIVPY (non-Language Environment VS COBOL II)

Before you run a sample, customize it for your installation as described in the sample.

Debug Tool Terminal Interface Manager

The Debug Tool Terminal Interface Manager enables a user to debug in full-screen mode using a dedicated terminal without having to know the LU name of the dedicated terminal. Use the Debug Tool Terminal Interface Manager because it makes it easier for users to identify the terminals to use for their debugging sessions

Complete the steps in "Enabling full-screen mode using a dedicated terminal" on page 16 before you do the instructions in this section to ensure that the basic full-screen mode using a dedicated terminal function works at your site.

Example: a debugging session using the Debug Tool Terminal Interface Manager

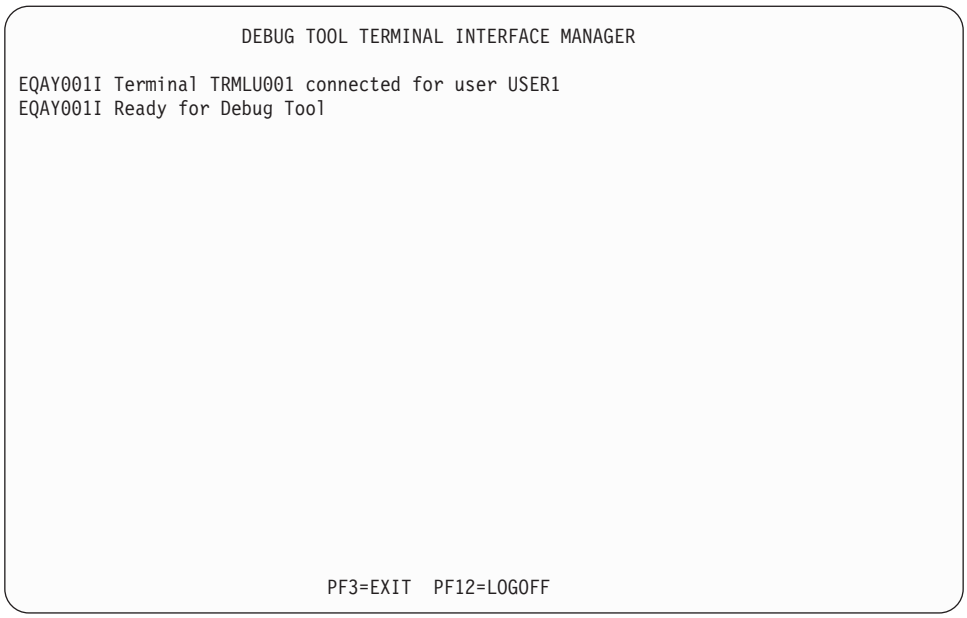
Compare the following steps with the steps shown in "How Debug Tool uses full-screen mode using a dedicated terminal" on page 15 to understand how using the Terminal Interface Manager affects the flow of work.

1. Start two terminal emulator sessions. These sessions can be either of the following situations:
   - Two separate terminal emulator sessions.
   - If you use IBM Session Manager, two sessions selected from the IBM Session Manager menu.

   In either situation, ensure that the second session connects to a terminal that can handle a full-screen mode debugging session through a dedicated terminal and that starts Debug Tool Terminal Interface Manager.
2. On the first terminal emulator session, log on to TSO.
3. On the second terminal emulator session, provide your TSO user ID and password to the Terminal Interface Manager and press Enter.

   Note: You are not logging on TSO. You are indicating that you want your user ID associated with this terminal LU.

   A panel similar to the following panel is then displayed on the second terminal emulator session:



The terminal is now ready to receive a Debug Tool full-screen mode using a dedicated terminal session.

4. Edit the PARM string of your batch job so that you specify the TEST run time parameter as follows:

   `TEST(,,,VTAM%userid:*)`

5. Submit the batch job.

   The tasks completed are similar to the tasks described in step 5 on page 15 except that first the batch job communicates with the Terminal Interface Manager to correlate the user ID to the terminal LU of the second terminal emulator session. The remaining steps are the same as described in step 5 on page 15.

6. On the second terminal emulator session, a full-screen mode debugging session is displayed. Interact with it the same way you would with any other full-screen mode debugging session.
7. After you exit Debug Tool, the second terminal emulator session displays the panel and messages you saw in step 3 on page 25. This indicates that Debug Tool can use this session again. (this will happen each time you exit from Debug Tool).
8. If you want to start another debugging session, return to step 5. If you are finished debugging, you can do one of the following tasks:
   - Close the second terminal emulator session.
   - Exit the Terminal Interface Manager by choosing one of the following options:
     - Press PF12 to display the Terminal Interface Manager logon panel. You can log in with the same ID or a different user ID.
     - Press PF3 to exit the Terminal Interface Manager.

Enabling full-screen mode using a dedicated terminal with Debug Tool Terminal Interface Manager

To enable full-screen mode using a dedicated terminal with Debug Tool Terminal Interface Manager, do the following steps:

1. Define the VTAM APPL definition statements as described in "Defining the Terminal Interface Manager APPL definition statements."
2. Start the Debug Tool Terminal Interface Manager as described in "Starting the Debug Tool Terminal Interface Manager."
3. Configure the Telnet Server as described in "Configuring the TN3270 Telnet Server to access the Terminal Interface Manager" on page 28.
4. Verify that the customizations are completed correctly by following the steps in "Verifying the customization of the Terminal Interface Manager" on page 31.

Defining the Terminal Interface Manager APPL definition statements

You must define the APPL definition statements that the Terminal Interface Manager will use for its sessions. To define the APPL definition statements, do the following steps:

1. Define the APPL definition statements as shown in the EQAWSESS member in the *hlq*.SEQASAMP data set by doing one of the following tasks:
   - Copy EQAWSESS into a new member:
     a. Create a new member in the VTAM definitions library (VTAMLST). The VTAM definitions library is often stored in the data set SYS1.VTAMLST.
     b. Copy the contents of the EQAWSESS member into the new member.
     c. Add the new member's name to the VTAM start options configuration file, ATCCON*xx*.
   - Copy EQAWSESS into an existing member:
     a. Select a member in the VTAM definitions library (VTAMLST) that contains the major node definitions.
     b. Copy the APPL definition statements for Debug Tool from the EQAWSESS member into the selected member.

To activate the new definitions, enter the following command from the console:

```
VARY NET,ACT,ID=member-name
```

member-name is the member name in the VTAM definitions library.

Starting the Debug Tool Terminal Interface Manager

The Debug Tool Terminal Interface Manager is a VTAM application that must be started (following the start of VTAM itself) before users can access it. Follow these steps to start it:

1. Copy the EQAYSESM member of the data set *hlq*.SEQASAMP to the SYS1.PROCLIB data set, making any changes required by your installation.
2. Make sure that the Debug Tool Terminal Interface Manager load module, EQAYSESM, resides in an APF authorized library (this module can be found in the *hlq*.SEQAAUTH data set). This is required to allow access to functions to validate users by TSO user ID and password.
3. Start the Debug Tool Terminal Interface Manager using the START command from the console. The START command can be added to the COMMND*xx* member of SYS1.PARMLIB to start the Debug Tool Terminal Interface Manager when the system is IPLed.

   The Debug Tool Terminal Interface Manager load module accepts three parameters, which you can provide by using the OPTS substitution variable on

the START command or in the EQAYSESM PROC definition. You can code the parameters in any sequence and all of them are optional. The following list describes the parameters:

-a *acbname*
Specifies an alternate VTAM ACB name for Terminal Interface Manager to open. For more information about this parameter, see "Running the Terminal Interface Manager on more than one LPAR on the same VTAM network" on page 30.

-s Instructs Terminal Interface Manager to supply an additional entry field on each Terminal Interface Manager panel, in which the user can enter an IBM Session Manager escape sequence. For more information about this parameter, see "Configuring Terminal Interface Manager as an IBM Session Manager application" on page 30.

+T
Turns on internal tracing for Terminal Interface Manager. Do not use this parameter unless instructed by IBM support personnel.

The following example starts the Debug Tool Terminal Interface Manager for alternate ACB EQASESS2 and instructs it to provide an extra entry field for use with IBM Session Manager:

```
START EQAYSESM,OPTS='-a EQASESS2 -s'
```

Configuring the TN3270 Telnet Server to access the Terminal Interface Manager

Select an additional unused port (for example, 2024) and then implement "Example 1" on page 22 with the following changes:

- Specify port 2024 instead of 2023 (3 times)
- Specify the following value for the DEFAULTAPPL statement:
  ```
  DEFAULTAPPL EQASESSM FIRSTONLY
  ```
- Make the following change on the ALLOWAPPL statement:
  ```
  ALLOWAPPL EQA*
  ```

Example 4

The example below shows the modified "Example 1" on page 22, with the changes highlighted with an asterisk (*).

```
PORT
 ...
 *  2024 TCP INTCLIEN            ; Telnet Server - Debug Tool
 ...

; Add a TELNETPARMS block for the new port

TELNETPARMS
 *  PORT 2024                     ; Debug Tool
  ... the rest of this should be a copy of the existing Port 23
ENDTELNETPARMS

; Add a BEGINVTAM block for the new port

BEGINVTAM
 *  PORT 2024

  ; Define the VTAM terminal LUs to use for this port (see EQAWTRML)
```

```
  LUGROUP DBGTOOL
      TRMLU001..TRMLU050
  ENDLUGROUP

  ; Allow anyone with access to this system to use the LUs above

  IPGROUP EVERYONE
      0.0.0.0:0.0.0.0
  ENDIPGROUP

  ; The Debug Tool Terminal Interface Manager will be displayed
  ; when an emulator connects

* DEFAULTAPPL EQASESSM FIRSTONLY

  ; Indicate that the ACBs always be allocated

  LUMAP DBGTOOL EVERYONE KEEPOPEN

  ; Allow only Debug Tool to use this port

* ALLOWAPPL   EQA*

ENDVTAM
```

See *hlq*.SEQASAMP(EQAWTTS4) for a sample of these definitions.

Instruct TCP/IP to use this additional definition, as described on page 21.

After you make these changes, your users can set up a unique terminal emulator session that connects to this new port, and debug programs that require the use of full-screen mode using a dedicated terminal with the Debug Tool Terminal Interface Manager. The user does the following steps:

1. Starts a terminal emulator session that connects to this new port. The Debug Tool Terminal Interface Manager is displayed.
2. The user enters his user ID and password and then presses Enter. The terminal is now ready to receive a Debug Tool full-screen mode using a dedicated terminal session.
3. On another terminal emulator session, the user starts his program with the TEST run-time option and specifies the VTAM%*userid* suboption. The terminal emulator session connected to this new port displays a full-screen mode using a dedicated terminal session.

Example: Connecting a VTAM network with multiple LPARs with one Terminal Interface Manager

This example describes the connections that need to be made in a VTAM network that has four LPARs that run Debug Tool jobs with one of the LPARs managing the terminals.

- LPAR 1 runs a TN3270E server and the Terminal Interface Manager with the default ACB name. Its VTAM also owns all the terminal LUs. Users connect their TN3270E emulator to this LPAR for the Terminal Interface Manager session. Users use the Terminal Interface Manager to create the connection between Debug Tool and the terminal LU used for their full-screen mode using a dedicated terminal debugging session.
- VTAM on LPAR1 defines the terminal LU APPL definition statements and the EQASESSM APPL definition statement for the Terminal Interface Manager.

- VTAM on LPAR 1 needs visibility to the EQAMVnnn APPL definition statements on LPARs 2, 3 and 4. This enables communication between the Terminal Interface Manager and Debug Tool.
- Each VTAM on LPAR 1, 2, 3 and 4 has a unique set of EQAMVnnn APPL definition statements. For example, LPAR 1 has APPL definition statements 001-050, LPAR 2 has APPL definition statements 051-100, LPAR 3 has APPL definition statements 101-150, and LPAR 4 has APPL definition statements 151-200.
- Each VTAM on LPAR 2, 3 and 4 needs visibility to the EQASESSM APPL definition statement on LPAR 1. This enables communication between Debug Tool and the Terminal Interface Manager.
- Each VTAM on LPAR 2, 3 and 4 needs visibility to the terminal LU APPL definition statements on LPAR 1.

Running the Terminal Interface Manager on more than one LPAR on the same VTAM network

This topic describes the modifications you need to make to the steps described in "Defining the Terminal Interface Manager APPL definition statements" on page 27, "Starting the Debug Tool Terminal Interface Manager" on page 27, and "Configuring the TN3270 Telnet Server to access the Terminal Interface Manager" on page 28 in order to make full-screen mode using a dedicated terminal with Terminal Interface Manager work in an environment where you want to run the Terminal Interface Manager on more than one LPAR in the same VTAM network.

Do the following steps for each additional instance of the Terminal Interface Manager:

1. In "Defining the Terminal Interface Manager APPL definition statements" on page 27, after you have copied EQAWSESS into a new or existing member, modify it so that you specify an ACB name other than the default EQASESSM.

   By default, Debug Tool assumes you work in an environment where you use only one instance of Terminal Interface Manager and the default ACB name used by this instance of Terminal Interface Manager and Debug Tool is EQASESSM. By specifying the ACB name used by the Terminal Interface Manager (instead of using the default name), you can create a unique ACB name for each instance of the Terminal Interface Manager.
2. In "Starting the Debug Tool Terminal Interface Manager" on page 27, after you copy the EQAYSESM member to the SYS1.PROCLIB data set, modify it to specify the new ACB name you created in step 1 by specifying `OPTS='-a XXXXXXXX'`, where *XXXXXXXX* is the new ACB name.
3. In "Configuring the TN3270 Telnet Server to access the Terminal Interface Manager" on page 28, when you modify the TCP/IP TN3270 server's configuration file, modify the DEFAULTAPPL statement to specify the ACB name you created in step 1, instead of EQASESSM.
4. Specify the EQAXOPT TIMACB option in the EQAOPTS option file, as described in "TIMACB" on page 96, using the new ACB name you created in step 1 for *ACB-name*.

Configuring Terminal Interface Manager as an IBM Session Manager application

To define Debug Tool Terminal Interface Manager as an application within IBM Session Manager, do the following steps:

1. Define a TN3270 port and a group of terminal LUs which start Terminal Interface Manager as described in "Configuring the TN3270 Telnet Server to access the Terminal Interface Manager" on page 28.
2. Enable the IBM Session Manager TCP/IP support, as described in *IBM Session Manager for z/OS: Installation and Getting Started*.
3. Define Terminal Interface Manager to IBM Session Manager as a TCP/IP application. To do this, create an APPL statement in the IBM Session Manager configuration, similar to the following statement:

```
APPL applname    APPLID TCP_1
                 DESC 'description'
                 DATA 'protocol://host-addr:port'
```

   The following list describes the variables used in this statement:

   applname
   Your choice for the application name. This is the name used when referring to the application in other IBM Session Manager definitions.

   description
   The descriptive text you want displayed on any session menus.

   protocol
   One of the following values: TELNET, TN3270 or TN3270E. For a description of these protocols, see "Session Manager and TCP/IP" in *IBM Session Manager: Facilities Reference*.

   host-addr
   The hostname or IP address of the server that hosts Terminal Interface Manager.

   port
   The port number that was configured for Terminal Interface Manager in step 1.

   For a complete description of the IBM Session Manager APPL configuration statement, see *IBM Session Manager: Technical Reference*.

   The following example shows an APPL statement:

```
APPL DTTIM
     APPLID TCP_1
     DESC 'Debug Tool Terminal Interface Manager'
     DATA 'TN3270E://mvsa.ibm.com:2024'
```

4. Start the Terminal Interface Manager started task with the `-s` parameter. This causes the Terminal Interface Manager panels to display an extra field where you can enter the IBM Session Manager escape key.

Verifying the customization of the Terminal Interface Manager

Do the following steps to verify the installation and customization:

1. Start a terminal emulator session that starts the Terminal Interface Manager. Enter your user ID and password and then press Enter.
2. On your other terminal emulator session, select the same IVP as you used above, change the run time parameter string from `MFI%LU_name:*` to `VTAM%userid:*`, submit the job and then follow the rest of the instructions in the IVP.

Chapter 7. Specifying the TEST runtime options through the Language Environment user exit

Debug Tool provides a customized version of the Language Environment user exit (CEEBXITA). The user exit returns a TEST runtime option when called by the Language Environment initialization logic. Debug Tool provides user exits for three different environments. This topic is also described in *Debug Tool User's Guide* with information specific to application programmers.

The user exit extracts the `TEST` runtime option from a user controlled data set with a name that is constructed from a naming pattern. The naming pattern can include the following tokens:

`&USERID`
Debug Tool replaces the `&USERID` token with the user ID of the current user. Each user can specify an individual `TEST` runtime option when debugging an application. This token is optional.

`&PGMNAME`
Debug Tool replaces the `&PGMNAME` token with the name of the main program (load module). Each program can have its own `TEST` runtime options. This token is optional.

Debug Tool provides the user exit in two forms:

- A load module. The load modules for the three environments are in the *hlq*.SEQAMOD data set. Use this load module if you want the default naming patterns and message display level. The default naming pattern is `&USERID.DBGTOOL.EQAUOPTS` and the default message display level is X'00'.
- Sample assembler user exit that you can edit. The assembler user exits for the three environments are in the *hlq*.SEQASAMP data set. You can also merge this source with an existing version of CEEBXITA. Use this source code if you want naming patterns or message display levels that are different than the default values.

Debug Tool provides the following user exits:

Table 3. Language Environment user exits for various environments

Environment	User exit name
DB2 stored procedures of type MAIN that run in WLM-established address spaces[1]	EQADDCXT
IMS TM and BTS[2]	EQADICXT
Batch and BTS	EQADBCXT

Notes:

1. EQADDCXT is supported for DB2 version 7 or later. If DB2 RUNOPTS is specified, EQADDCXT takes precedence over DB2 RUNOPTS.
2. For BTS, you need to specify Environment command (./E) with the user ID of the IO PCB. For example, if the user ID is ECSVT2, then the Environment command is `./E USERID=ECSVT2`.

Your users can use the user exit in the following ways:

- The user can link the user exit into his application program.
- The user can link the user exit into a private copy of a Language Environment module (CEEBINIT, CEEPIPI, or both), and then, only for the modules the user might debug, place the SCEERUN data set containing this module in front of the system Language Environment modules in CEE.SCEERUN in the load module search path.

To learn about the advantages and disadvantages of each method, see "Comparing the two methods of linking CEEBXITA" on page 36.

To prepare your site to use the Language Environment user exit, do the following tasks:

1. "Editing the source code of CEEBXITA."
2. "Linking the CEEBXITA user exit into a private copy of a Language Environment runtime module" on page 36.

To do the instructions in "Customizing for JCL for Batch Debugging utility" on page 44, you need the following information:

- If you change the naming pattern of the TEST runtime options data set, you need the new naming pattern.
- The name of the *hlq*.BATCH.SCEERUN data set you create when you do the instructions in "Linking the CEEBXITA user exit into a private copy of a Language Environment runtime module" on page 36.

Editing the source code of CEEBXITA

You can edit the sample assembler user exit that is provided in *hlq*.SEQASAMP to customize the naming patterns or message display level by doing one of the following tasks:

- Use an SMP/E USERMOD to update the copy of the exit in the *hlq*.SEQAMOD data set. Use the following sample USERMODs in *hlq*.SEQASAMP for this task:

User exit name	USERMOD name
EQADDCXT	EQAUMODC
EQADICXT	EQAUMODD
EQADBCXT	EQAUMODB

- Create a private load module for the customized exit. Copy the assembler user exit that has the same name as the user exit from *hlq*.SEQASAMP to a local data set. Edit the patterns or message display level. Customize and run the JCL to generate a load module.

Modifying the naming pattern

The naming pattern of the data set that has the `TEST` runtime option is in the form of a sequential data set name. You can optionally specify a `&USERID` token, which Debug Tool substitutes with the user ID of the current user. You can also add a `&PGMNAME` token, which Debug Tool substitutes with the name of the main program (load module).

In some cases, the first character of a user ID is not valid for a name qualifier. A character can be concatenated before the `&USERID` token to serve as the prefix

character for the user ID. For example, you can prefix the token with the character "P" to form P&USERID, which is a valid name qualifier after the current user ID is substituted for &USERID.

The default naming pattern is &USERID.DBGTOOL.EQAUOPTS. This is the pattern that is in the load module provided in *hlq*.SEQAMOD.

The following table shows examples of naming patterns and the corresponding data set names after Debug Tool substitutes the token with a value.

Table 4. Data set naming patterns, values for tokens, and resulting data set names

Naming pattern	User ID	Program name	Name after user ID substitution
&USERID.DBGTOOL.EQAUOPTS	JOHNDOE		JOHNDOE.DBGTOOL.EQAUOPTS
P&USERID.EQAUOPTS	123456		P123456.EQAUOPTS
DT.&USERID.TSTOPT	TESTID		DT.TESTID.TSTOPT
DT.&USERID.&PGMNAME.TSTOPT	TESTID	IVP1	DT.TESTID.IVP1.TSTOPT

To customize the naming pattern of the data set that has TEST runtime option, change the value of the DSNT DC statement in the sample user exit. For example:

```
* Modify the value in DSNT DC field below.
*
* Note: &USERID below has one additional '&', which is an escape
*       character.
*
DSNT_LN         DC  A(DSNT_SIZE)  Length field of naming pattern
DSNT            DC  C'&&USERID.DBGTOOL.EQAUOPTS'
DSNT_SIZE       EQU *-DSNT        Size of data set naming pattern
*
```

Modifying the message display level

You can modify the message display level for CEEBXITA. The following values set WTO message display level:

X'00'
Do not display any messages.

X'01'
Display error and warning messages.

X'02'
Display error, warning, and diagnostic messages.

The default value, which is in the load module in *hlq*.SEQAMOD, is X'00'.

To customize the message display level, change the value of the MSGS_SW DC statement in the sample user exit. For example:

```
* The following switch is to control WTO message display level.
*
*   x'00' - no messages
*   x'01' - error and warning messages
*   x'02' - error, warning, and diagnostic messages
*
MSGS_SW         DC  X'00'         message level
*
```

Comparing the two methods of linking CEEBXITA

You can link in the user exit CEEBXITA in the following ways:

- Link it into the application program.

  Advantage

  The user exit affects only the application program being debugged. This means you can control when Debug Tool is started for the application program. You might also not need to make any changes to your JCL to start Debug Tool.

  Disadvantage

  You must remember to remove the user exit for production or, if it isn't part of your normal build process, you must remember to relink it to the application program.

- Link it into a private copy of a Language Environment runtime load module (CEEBINIT, CEEPIPI, or both)

  Advantage

  You do not have to change your application program to use the user exit. In addition, you do not have to link edit extra modules into your application program.

  Disadvantage

  You need to take extra steps in preparing and maintaining your runtime environment:

  – Make a private copy of one or more Language Environment runtime routines

  – Only for the modules you might debug, customize your runtime environment to place the private copies in front of the system Language Environment modules in CEE.SCEERUN in the load module search path

  – When you apply maintenance to Language Environment, you might need to relink the routines.

  – When you upgrade to a new version of Language Environment, you must relink the routines.

If you link the user exit into the application program and into a private copy of a Language Environment runtime load module, which is in the load module search path of your application execution, the copy of the user exit in the application load module is used.

Linking the CEEBXITA user exit into a private copy of a Language Environment runtime module

The following table shows the Language Environment runtime load module and the user exit needed for each environment.

Table 5. Language Environment runtime module and user exit required for various environments

Environment	User exit name	CEE load module
DB2 stored procedures of type MAIN that run in WLM-established address spaces	EQADDCXT	CEEPIPI
IMS TM and BTS	EQADICXT	CEEBINIT

Table 5. Language Environment runtime module and user exit required for various environments (continued)

Environment	User exit name	CEE load module
Batch	EQADBCXT	CEEBINIT

Edit and run sample *hlq*.SEQASAMP(EQAWLCEE) to create these updated Language Environment runtime modules. The sample creates the following load module data sets:

- *hlq*.DB2SP.SCEERUN(CEEPIPI)
- *hlq*.IMSTM.SCEERUN(CEEBINIT)
- *hlq*.BATCH.SCEERUN(CEEBINIT)

Inform your users that you created these data sets. When you apply service to Language Environment that affects either of these modules (CEEPIPI or CEEBINIT) or you move to a new level of Language Environment, you need to rebuild your private copy of these modules by running the sample again.

Chapter 8. Installing the browse mode RACF facility

Debug Tool browse mode can be controlled by either the browse mode RACF facility or through the EQAOPTS option file. For an overview of how to control browse mode, see "Debugging in browse mode" in *Debug Tool User's Guide*.

If you want to use RACF to enforce one of the following situations, you must install the browse mode RACF facility:

- Debug programs in a production environment (or some other environment) where you want to control whether Debug Tool users can modify the contents of storage or alter program flow

  Note: C/C++ programs cannot be run in browse mode.
- Restrict the use of Debug Tool to certain users

To install the browse mode RACF facility, your security administrator must do the following tasks:

1. Choose one or both RACF facilities associated with the browse mode facility to install, then install it.
2. Set up the default user access to the facility.
3. Authorize those users that need access other than the default access.

The following RACF facilities are associated with the browse mode facility:

- EQADTOOL.BROWSE.MVS
- EQADTOOL.BROWSE.CICS

You can install either or both facilities. The first facility controls browse mode for non-CICS MVS jobs. The second controls browse mode access in CICS regions.

In most cases, if you install the browse mode RACF facility, then specify UACC(READ). However, assigning *fac_uacc* any of the following values creates the corresponding result:

NONE
: Only users specifically authorized to the facility can use Debug Tool in any way.

READ
: Only users specifically authorized to the facility can use Debug Tool in the normal (non-browse mode) way.

UPDATE
: Users specifically authorized to the facility can be limited to using Debug Tool in browse mode but all other users can use Debug Tool in either the normal (non-browse mode) way or in browse mode, depending on an entry in the EQAOPTS option file.

When you assign *usr_acc* any of the following values to grant access to a specific user, you create the corresponding result:

NONE
: The user cannot use Debug Tool in any way.

READ
: The user can use Debug Tool only in browse mode.

UPDATE (or higher)
The user can use Debug Tool in the normal (non-browse) mode by default. He can also select through EQAOPTS whether he wants the current invocation of Debug Tool to be in browse mode or normal mode.

The following instructions use EQADTOOL.BROWSE.*xxx* to represent either or both of facilities. If you choose to install both, you need to run the steps twice, once with each name.

Do the following steps to install the browse mode facility:

1. Establish a profile for the browse mode facility in the FACILITY class by entering the following RDEFINE command:

   ```
   RDEFINE FACILITY EQADTOOL.BROWSE.xxx UACC(fac_uacc)
   ```

2. Verify that generic profile checking is in effect for the class FACILITY by entering the following command:

   ```
   SETROPTS GENERIC(FACILITY)
   ```

Do the following steps to give individual users or user groups specific access to the browse mode facility:

1. Give a user permission to use the browse mode facility by entering the following command, where *DUSER1* is the name of a RACF-defined user or group profile:

   ```
   PERMIT EQADTOOL.BROWSE.xxx CLASS(FACILITY) ID(DUSER1) ACCESS(usr_acc)
   ```

   Instead of connecting individual users, the security administrator can specify *DUSER1* to be a RACF group profile and then connect authorized users to the group.
2. If the FACILITY class is not active, activate the class by entering the SETROPTS command:

   ```
   SETROPTS CLASSACT(FACILITY)
   ```

   Issue the SETROPTS LIST command to verify that FACILITY class is active.
3. Refresh the FACILITY class by issuing the SETROPTS RACLIST command:

   ```
   SETROPTS RACLIST(FACILITY) REFRESH
   ```

Refer to the following topics for more information related to the material discussed in this topic.

Related tasks

"BROWSE" on page 84

Chapter 9. Customizing Debug Tool Utilities

Debug Tool Utilities is a group of ISPF applications that provides the following tools and functions:

- **Program Preparation** to help application programmers precompile, compile, and link their programs and then start Debug Tool. This includes using COBOL and CICS Command Level Conversion Aid (CCCA) to help application programmers convert older COBOL programs to Enterprise COBOL programs.
- **Debug Tool Setup File**, which manages setup files. Setup files help application programmers prepare programs to debug them interactively or in batch mode.
- **Code Coverage** to help application programmers conduct coverage tests on their programs.
- **IMS TM Setup** to help you edit the `TEST` runtime options used by IMS programs and to create private message regions for testing.
- **Load Module Analyzer** to help users analyze load modules to determine the language translator that was used to compile or assemble each CSECT in the load module.
- **Debug Tool User Exit Data Set** to create and edit a data set that Language Environment user exits read to obtain the `TEST` runtime options string.
- **Other IBM Problem Determination Tools** to help you start IBM File Manager for z/OS.
- **JCL for Batch Debugging** to help users start a debug session when they run their application in a batch job.

The instructions in this section describe the following customization tasks:

- Choose a method to start Debug Tool Utilities
- Customize the data set names in EQASTART.
- Add Debug Tool Utilities to an ISPF menu so that your users can start Debug Tool Utilities from an ISPF menu.
- Modify Debug Tool Setup Utility so that your users can access procedure libraries.
- Customize the JCL for Batch Debugging interface.
- Customize the Problem Determination Tools interface.
- Customize Program Preparation so that users access the proper compilers and development utilities.
- Make changes so that users can access Coverage Utility and provide any defaults for Coverage Utility.
- If your users use the IMS TM Setup - Manage LE Runtime Options function in Debug Tool Utilities, make changes so that users can access this function in an IMSplex environment.

Choosing a method to start Debug Tool Utilities

Your users can start Debug Tool Utilities by doing one of the following methods:

Method 1: Enter the `EXEC '`*`hlq`*`.SEQAEXEC(EQASTART)'` command. This is the default method.

Method 2: Enter the EQASTART command. To use this method, you must do the following steps, which are described in this section:

1. Include or copy the Debug Tool Utilities data sets to your system's TSO logon data sets. To add the data sets, do one of the following alternatives:
   - Include the data sets listed in Table 6, Table 7, or Table 8 on page 43 into the DD concatenations specified in the tables.
   - Copy[3] the members of the data sets listed in Table 6, Table 7, or Table 8 on page 43 to a data set allocated to the DD concatenation specified in the table.

   For either alternative, the data sets you include into the DD concatenations must match the national language you chose in "Changing the default and allowable values in EQACUIDF" on page 99.
2. Edit the EQASTART[4] member of the *hlq*.SEQAEXEC data set and set the *Inst_NATLANG_commonlib* variable to ENU, UEN, JPN, or KOR depending on the national language you chose in "Changing the default and allowable values in EQACUIDF" on page 99.
3. Inform your users how to specify a language other than the one selected in step 2. If your users need to start Debug Tool in a language other than the default, they need to add the NATLANG(*xxx*) parameter to the EQASTART command.

Table 6. For English, data sets that need to be included or copied into the specified DD concatenations

DD concatenation	Data set name
SYSEXEC or SYSPROC	*hlq*.SEQAEXEC
ISPMLIB	*hlq*.SEQAMENU
ISPLLIB	*hlq*.SEQAMOD
ISPPLIB	*hlq*.SEQAPENU
ISPSLIB	*hlq*.SEQASENU
ISPTLIB	*hlq*.SEQATLIB

Table 7. For uppercase English, data sets that need to be included or copied into the specified DD concatenations

DD concatenation	Data set name
SYSEXEC or SYSPROC	*hlq*.SEQAEXEC
ISPMLIB	*hlq*.SEQAMENP
ISPLLIB	*hlq*.SEQAMOD
ISPPLIB	*hlq*.SEQAPENP
ISPSLIB	*hlq*.SEQASENP
ISPTLIB	*hlq*.SEQATLIB

4. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.

Table 8. For Japanese, data sets that need to be included or copied into the specified DD concatenations

DD concatenation	Data set name
SYSEXEC or SYSPROC	*hlq*.SEQAEXEC
ISPMLIB	*hlq*.SEQAMJPN
ISPLLIB	*hlq*.SEQAMOD
ISPPLIB	*hlq*.SEQAPJPN
ISPSLIB	*hlq*.SEQASJPN
ISPTLIB	*hlq*.SEQATLIB

Table 9. For Korean, data sets that need to be included or copied into the specified DD concatenations

DD concatenation	Data set name
SYSEXEC or SYSPROC	*hlq*.SEQAEXEC
ISPMLIB	*hlq*.SEQAMKOR
ISPLLIB	*hlq*.SEQAMOD
ISPPLIB	*hlq*.SEQAPKOR
ISPSLIB	*hlq*.SEQASKOR
ISPTLIB	*hlq*.SEQATLIB

Customizing the data set names in EQASTART

You must modify member EQASTART of the *hlq*.SEQAEXEC data set to specify the data set names that you chose at installation time. Edit the EQASTART[5] member and follow the directions in the member's prologue for site customization of data set names.

Adding Debug Tool Utilities to the ISPF menu

To add Debug Tool Utilities to an ISPF panel, add code that calls EQASTART to an existing panel. For example, to add Debug Tool Utilities to the ISPF Primary Option Menu panel (ISR@PRIM), insert the additional lines (**←New**) as shown below:

```
...
)BODY CMD(ZCMD)
...
9 IBM Products IBM program development products
10 SCLM SW Configuration Library Manager
11 Workplace ISPF Object/Action Workplace
F File Manager File Manager for z/OS
D Debug Tool - Debug Tool Utility functions  ←New
...
)PROC
...
&ZSEL; = TRANS( TRUNC (&ZCMD;,'.')
...
9,'PANEL(ISRDIIS) ADDPOP'
10,'PGM(ISRSCLM) SCRNAME(SCLM) NOCHECK'
```

5. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.

```
11,'PGM(ISRUDA) PARM(ISRWORK) SCRNAME(WORK)'
F,'PANEL(FMNSTASK) SCRNAME(FILEMGR) NEWAPPL(FMN)' /* File Manager */
D,'CMD(EXEC ''hlq.SEQAEXEC(EQASTART)'')' /* Debug Tool Utilities */  ←1
...
```

If you copied Debug Tool Utilities to system data sets or concatenated them to existing DDnames (as described in Method 2 in "Choosing a method to start Debug Tool Utilities" on page 41), then change line **1** to the following:

```
D,'CMD(%EQASTART)' /* Debug Tool Utilities */
```

For more information about configuring your ISPF Primary Option Menu panel, see *z/OS ISPF Planning and Customizing*.

Customizing Debug Tool Setup Utility

Debug Tool Setup Utility provides a command called COPY, which copies a JCL stream into a setup file. The EQAZPROC member of the *hlq*.SEQATLIB data set includes a list of JCL procedure libraries that Debug Tool Setup Utility uses as a source for the COPY command. You can add your own procedure libraries to the list by editing EQAZPROC and adding the procedure library names, one name per line and without trailing commas, beginning on column 1. The order in which you list procedure libraries in EQAZPROC must match the order in which you list procedure libraries in the PROCLIB concatenation.

For example, to add the LOCAL.PROCLIB procedure library name, do the following steps:

1. Edit the EQAZPROC[6] member of the *hlq*.SEQATLIB data set.
2. Add the LOCAL.PROCLIB procedure library name. The result looks like the following:

   ```
   LOCAL.PROCLIB
   SYS1.PROCLIB
   ```
3. Save and close the file.

Customizing for JCL for Batch Debugging utility

The JCL for Batch Debugging utility helps your users prepare JCL and start a debug session. You can supply your users with a number of default values.

To set the defaults, do the following steps:

1. Edit the EQAZDFLT[7] member of the *hlq*.SEQATLIB data set.
2. Modify the parameter values to match what you use at your site.
3. Add parameters required by your site. You can add parameters by doing one of the following alternatives:
   - Use the INCLUDE '*any.data.set.name*'; statement to include statements from a data set that you created.
   - Use the INCLUDE *membername*; statement to include parameters from other members in the data set *hlq*.SEQATLIB.

See the EQAZDSYS member of the *hlq*.SEQATLIB data set for the complete list of parameters and the syntax convention for these parameters.

6. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.

7. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.

If your users use terminals that cannot display mixed-case English text, enter all parameters in uppercase English.

Parameters you can set

The first 3 characters of each parameter are "yb1". The last five characters correspond to the parameter:

yb1dtmod
Debug Tool load module data set (`SEQAMOD`).

yb1dtflg
Flag to include Debug Tool load module data set in STEPLIB. Y for Yes, N for No.

If it is No, the installer must ensure that `SEQAMOD` can be found in the load module search path.

yb1dtdev
Debug session type: MFI, TIM, or GUI.

MFI
dedicated terminal identified by network and LU names.

TIM
dedicated terminal identified by user id.

GUI
Remote debugger identified by IP address.

yb1dtmtd
Debug Tool invocation method: C, E or A.

C CEEOPTS DD statement. This requires z/OS Version 1.7 or later.

E User exit module EQADBCXT in Language Environment CEEBINIT module. For instructions on how to implement this method, see Chapter 7, “Specifying the TEST runtime options through the Language Environment user exit,” on page 33.

A User exit module EQADBCXT in application module.

yb1dtprf
Data set that contains a Debug Tool preferences file.

yb1dtcmd
Data set that contains a Debug Tool commands file.

yb1dtbin
The name of the Language Environment SCEERUN(CEEBINIT) load module data set that contains the Debug Tool user exit module `EQADBCXT`. To make sure you provide the correct name, see Chapter 7, “Specifying the TEST runtime options through the Language Environment user exit,” on page 33.

yb1dtnmp
Naming pattern that identifies the Debug Tool user’s data set which contains the TEST runtime options and pattern matching information. The naming pattern must be the same as the one coded in the Debug Tool user exit module `EQADBCXT`. To make sure you provide the correct naming pattern, see Chapter 7, “Specifying the TEST runtime options through the Language Environment user exit,” on page 33.

yb1dtdfl
Debug information file. It contains a list of data sets of debug information, source, and listing files.

Customizing JCL for Batch Debugging for multiple systems

You can customize JCL for Batch Debugging utility for multiple systems by doing one of the following alternatives:

- Modify EQASTART[8] to use a fully qualified data set name or member name other than EQAZDFLT to start Debug Tool Utilities.
- Instruct your users to enter one of the following commands, depending on the customization they want to use:
  - `EXEC 'hlq.SEQAEXEC(EQASTART)' 'PUMEMBER(''data.set.name'')'`
  - `EXEC 'hlq.SEQAEXEC(EQASTART)' 'PUMEMBER(membername)'`

Customizing for the Problem Determination Tools

The Problem Determination Tools allow your users to access other IBM problem determination tools. You can supply your users with parameter values needed for accessing the tools.

To give users access to the proper tools:

1. Edit the EQAZDFLT[9] member of the *hlq*.SEQATLIB data set.
2. Modify the data set names to match what you use at your site.
3. Add parameters required by your site. You can add parameters by doing one of the following alternatives:
   - Use the `INCLUDE 'any.data.set.name';` statement to include statements from a data set that you created.
   - Use the `INCLUDE membername;` statement to include parameters from other members in the data set *hlq*.SEQATLIB.

   See the EQAZDSYS and EQAZDUSR members of the *hlq*.SEQATLIB data set for the complete list of parameters and the syntax convention for these parameters.

   If your users use terminals that cannot display mixed-case English text, enter all parameters in uppercase English.

Parameters you can set

The first two characters of each parameter are always 'pt'. The third character corresponds to the tool:

1 IBM File Manager parameters

The last five characters correspond to the parameter:

flg1 Base function availability flag: Yes or No.

flg2 DB2 function availability flag: Yes or No.

flg3 IMS function availability flag: Yes or No.

ttl Title for the tool.

elib ISPF EXEC library data set.

8. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.
9. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.

mlib ISPF message library data set.

plib ISPF panel library data set.

slib ISPF skeleton library data set.

tlib ISPF table library data set.

pnl1 ISPF panel name for the base function.

pnl2 ISPF panel name for the DB2 function.

pnl3 ISPF panel name for the IMS function.

Customizing Problem Determination Tools for multiple systems

You can customize Problem Determination Tools for multiple systems by doing one of the following alternatives:

- Modify EQASTART[10] to use a fully qualified data set name or member name other than EQAZDFLT to start Debug Tool Utilities.
- Instruct your users to enter one of the following commands, depending on the customization they want to use:
  - `EXEC 'hlq.SEQAEXEC(EQASTART)' 'PUMEMBER(''data.set.name'')'`
  - `EXEC 'hlq.SEQAEXEC(EQASTART)' 'PUMEMBER(membername)'`

Customizing Program Preparation

Program Preparation helps your users access the proper compilers and development utilities that are installed at your site. You can supply your users with default values for data set naming patterns, data set allocation parameters, and compiler and utility option strings.

To give users access to the proper compilers and development utilities, do the following steps:

1. Edit the EQAZDFLT[11] member of the *hlq*.SEQATLIB data set.
2. Modify the data set names to match what you use at your site.
3. Add parameters required by your site. You can add parameters by doing one of the following alternatives:
   - Use the `INCLUDE 'any.data.set.name';` statement to include statements from a data set that you created.
   - Use the `INCLUDE membername;` statement to include parameters from other members in the data set *hlq*.SEQATLIB.

   See the EQAZDSYS and EQAZDUSR members of the *hlq*.SEQATLIB data set for the complete list of parameters and the syntax convention for these parameters.

   If your users use terminals that cannot display mixed-case English text, you must enter all parameters in uppercase English.

   If your site uses CCCA and requires that you use the VOLUMES parameter when you define private data sets (for example, a cluster is not managed by SMS), you must include the VOLUMES parameter when you define private data sets. Modify the following variables to include the VOLUMES parameter:
   - `yccctla1`

10. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.
11. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.

- ycclcpa1
- yccchga1
- yccwrka1
- ycctkna1

The following example illustrates how the variable yccctla1 is modified to include the parameter VOLUMES(SYS166):

```
yccctla1 =          !  CONTROL FILE KSDS
                   RECORDS(10000 1000)
                   FREESPACE(30 30)
                   INDEXED
                   SPEED
                   CISZ(4096)
                   UNIQUE
                   KEYS(15 0)
                   VOLUMES(SYS166)
                   RECORDSIZE(188 188);
```

Parameters you can set

The first two characters of each parameter are always 'yc'. The third character corresponds to the compiler or development utility parameters:

1 COBOL compiler parameters

3 PL/I compiler parameters

4 C and C++ compiler parameters

5 Assembler parameters

L Link Edit parameters

c CCCA parameters

F Fault Analyzer parameters

G Fault Analyzer listing create parameters

DB2 and CICS parameters
The DB2 precompiler and CICS translator are listed by the compiler you use. You can specify a different DB2 precompiler or CICS translator for each compiler.

The last five characters correspond to the parameter:

ciclb LINKLIST or load module data set name for CICS translator.

cicmd Load module name for CICS translator.

cicps CICS translator options.

clib LINKLIST or load module data set name for the compiler.

cmod Load module name for the compiler or utility.

For the Fault Analyzer side file create and Fault Analyzer listing create utilities, the following modules are available from the Debug Tool load library or from the Fault Analyzer load library. They are functionally the same.

- Fault Analyzer side file create function:
  - Debug Tool load library: EQALANGX
  - Fault Analyzer load library: IDILANGX
- Fault Analyzer side file listing create function:

- Debug Tool load library: EQALANGP
- Fault Analyzer load library: IDILANGP

ctovr TEST compiler option override flag. Use this flag to allow or disallow the TEST or DEBUG compiler option specified in the ctst, ctst1, ctst2, ctst3, ctst4, or ctst5 parameters to be overridden by the settings in the user profile. This parameter is valid for the COBOL compiler, PL/I compiler, and C and C++ compiler.

ctst Use TEST, NOTEST, DEBUG, or NODEBUG as the main compiler debugging option. This parameter is valid for the COBOL compiler, PL/I compiler, and C and C++ compiler.

ctst1, ctst2, ctst3, ctst4, ctst5
TEST or DEBUG suboptions. These parameters are valid for the COBOL compiler, PL/I compiler, and C and C++ compiler.

cttl Title for the compiler.

db2lb LINKLIST or load module data set name for the DB2 precompiler.

db2md
Load module name for DB2 precompiler.

db2ps DB2 precompiler options.

flg Enable or disable the compiler or development utility.

lsta1 Parameters of the TSO ALLOCATE command to use when data sets for compiler listings are allocated.

lstat Data set type for the compiler listing. The type can be one of these values: PDSE, PDS, or SEQ.

lstxx Pattern to use to create a name for the compiler listing data set. The name is created by using the characters in the pattern. The special characters, which start with a slash (/), are replaced by the following values:

/1, /2, ..., /n
The nth qualifier of the fully qualified data set name that was used as input to the compiler.

/B The second to (n-1) qualifier of the fully qualified data set name that was used as input to the compiler.

/L The right-most qualifier of the fully qualified data set name that was used as input to the compiler.

/U Current® TSO user ID.

/P Current TSO profile prefix.

sds1 Shared data set prefix for CCCA.

svs1 Shared VSAM data set prefix for CCCA.

tmpa1 Parameters of the TSO ALLOCATE command to use when temporary data sets are allocated.

Customizing Program Preparation for multiple systems

You can customize Program Preparation for multiple systems by doing one of the following alternatives:

- Modify EQASTART[12] to use a fully qualified data set name or member name other than EQAZDFLT to start Debug Tool Utilities.
- Instruct your users to enter one of the following commands, depending on the customization they want to use:
  - EXEC '*hlq*.SEQAEXEC(EQASTART)' 'PUMEMBER(''*any.data.set.name*'')'
  - EXEC '*hlq*.SEQAEXEC(EQASTART)' 'PUMEMBER(*membername*)'

Customizing Coverage Utility

This section describes the steps you must do to enable Coverage Utility:

Setting up the Coverage Utility monitor interface

Do the following tasks to allow the Coverage Utility monitor interface, EQACUOCM, to be invoked:

1. Add the EQACUOCM program to the AUTHPGM entry in the member IKJTSOxx of the SYS1.PARMLIB data set.
2. Issue the PARMLIB UPDATE(xx) command from TSO or IPL your system.

Placing Coverage Utility load modules in an APF-authorized data set not accessible to general users

Certain Coverage Utility load modules must be placed in an APF-authorized data set that is accessible only to system programmers. The APF-authorized data set must not be in the link list.

To place the load modules in an APF-authorized data set, do one of the following alternatives:

- Mark the *hlq*.SEQAAUTH data set as APF-authorized[1] and do one of the following:
  - Limit access to only system programmers.
  - Create Resource Access Control Facility (RACF) profiles to restrict access to these load modules.
- Do not mark the *hlq*.SEQAAUTH data set as APF-authorized. Copy[3] the following load modules into an APF-authorized data set that only system programmers can access:
  - EQACUOIN (SVC installer)
  - EQACUOSV (SVCs)

Creating RACF profiles

If you place Coverage Utility load modules that must not be accessible to all users in an APF-authorized data set that is accessible to all users, you must create RACF profiles to prevent access to these load modules. You can add the code in the following example to the RACF profile:

```
RDEFINE PROGRAM EQACUOIN NOTIFY(notify) UACC(NONE) +
DATA('RACF profile for Coverage Utility monitor') +
ADDMEM('authlib'/'volser'/PADCHK) OWNER(owner)

RDEFINE PROGRAM EQACUOSV NOTIFY(notify) UACC(NONE) +
DATA('RACF profile for Coverage Utility monitor') +
ADDMEM('authlib'/'volser'/PADCHK) OWNER(owner)

SETROPTS WHEN(PROGRAM) REFRESH
```

12. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.

```
PERMIT EQACUOIN CLASS(PROGRAM) ID(id) ACCESS(READ)
PERMIT EQACUOSV CLASS(PROGRAM) ID(id) ACCESS(READ)

SETROPTS WHEN(PROGRAM) REFRESH
```

The commands above restrict access to EQACUOIN and EQACUOSV by granting read access to only *id*. The following list describes the operands used in this example:

notify
TSO user ID of the person who is notified of a RACF access failure.

authlib
Name of the APF-authorized data set that contains EQACUOIN and EQACUOSV.

volser
Volume serial of authlib data set or ****** to specify the current SYSRES volume.

owner
TSO user ID or RACF group name of the person or persons that own this profile.

id TSO user ID or RACF group name of the person or persons who have the ability to install the SVCs.

Installing and enabling the monitor SVCs

The EQACUOIN module installs and enables the monitor SVCs. The monitor SVCs must be installed and enabled before a user starts a monitor session. The EQACUOIN module must be run:

- When the SVCs are initially installed
- After service is applied
- Any time you IPL your system

The monitor SVCs use some common system storage, as described below. In addition, each user session uses ECSA storage. See Appendix B of the *Debug Tool Coverage Utility User's Guide and Messages* for more information about the amount of ECSA storage used by each user session.

CSA 13248 bytes

SQA 25496 bytes

Perform the following steps to:

- Install and enable the monitor SVCs immediately.
- Prepare the system so that the monitor SVCs are installed and enabled after each IPL.

1. Reserve two free *user* SVC numbers. User SVC numbers must be in the range 200 to 255 (X'C8' to X'FF'). Verify that these SVC numbers are not being used on your system. SYS1.PARMLIB(IEASVC*xx*) does not need to be updated since these user SVCs can only be installed dynamically. However, for future reference, add a comment to IEASVC*xx* to indicate that these SVCs are used.
2. Copy *hlq*.SEQASAMP(EQACUOPS) to your SYS1.PROCLIB data set as member EQACUOIN. Make the following edits to the new EQACUOIN member:
   a. Change the STEPLIB data set name to the name of the APF-authorized data set that contains the EQACUOIN and EQACUOSV modules.

b. Change the PARM operands to contain the two user SVC numbers (in hexadecimal notation) that you reserved for Coverage Utility. Verify that you typed these numbers correctly.

3. Use the PERMIT commands, as described in "Creating RACF profiles" on page 50, to give the process started by EQACUOIN access to the `EQACUOIN` and `EQACUOSV` load modules. The process started by EQACUOIN is assigned an ID by the RACF started procedures table or STARTED class. Use this ID as the value for the *id* variable of the ID parameter of the PERMIT command.
4. The `SYS1.PARMLIB(COMMND`*xx*`)` data set contains the names of programs to start at IPL time. Add the following line to the `COMMND`*xx* member of the `SYS1.PARMLIB` data set:

   ```
   COM='S EQACUOIN'
   ```

5. Run the EQACUOIN procedure by entering the following START command from the system console:

   ```
   S EQACUOIN
   ```

   Verify that the job completed with a return code of 0.

   To verify that the monitor was installed properly, run the following command from ISPF panel 6:

   ```
   ex 'hlq.SEQAEXEC(EQACUOSE) 'LEVEL'
   ```

   An ISPF Browse panel similar to the following panel is displayed:

   ```
   BROWSE    SMITH.MSGS.FILE                            Line 00000000 Col
   Command ===>                                           Scroll ==
   ***************************** Top of Data ****************************
   Monitor Release: VAR1M0 Date: 2002.245
   MAST: 00F9B8E8 PSA: 00F87000 CPU: 00F87000 SEST: 00F9BCE8 UNID: 00000000
   ```

Customizing the Coverage Utility defaults

Complete the following steps to edit *hlq*`.SEQAEXEC(EQACUDFT)`[13]:

1. Change all occurrences of EQAW to *hlq*. For example, to use the high-level qualifier EQAW.V10R1M0, change all occurrences of EQAW to EQAW.V10R1M0.
2. Enter the Coverage Utility Monitor SVC numbers (in hexadecimal notation) in the CUSVC2B and CUSVC4B entries.
3. When you create JCL, the *JOBLn lines become the first three lines of the JOB card for each respective job. Customize these lines and customize all of the *JOBJ* lines to specify any JES control information as appropriate for your site.
4. If your site requires a specification for allocation parameters such as STORCLAS or UNIT on new or temporary data set allocations, look for the word *SPACE* in this EXEC and the '*hlq*.SEQAS*' data sets and update the allocation specifications.
5. If you want Coverage Utility to generate or build each data set as sequential or partitioned, set the USEPRGNM variable to Y. To generate a data set as sequential, set the DSORG variable to SEQ. To generate a data set as partitioned, set the DSORG variable to PDS.

   Coverage Utility uses the following forms to generate data set names:
   - For sequential data sets:

     '*proj_qual.program_name.file_type*'

     For example: 'PROGA.SAMPLE.COB01.BRKTAB'

13. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.

- For partitioned data sets:

  '*proj_qual.file_type*(*program_name*)'

  For example: 'PROGA.SAMPLE.BRKTAB(COB01)'

6. If you do not want Coverage Utility to generate or build any data set names automatically, set the USEPRGNM variable to N.

Configuring for IMSplex users

To determine if you need to do the steps described in this topic, read "Preparing IMS programs" in *Debug Tool User's Guide*. If your users use the **IMS TM Setup - Manage LE Runtime Options** function in Debug Tool Utilities, you must do the following tasks:

1. Install and configure IMS Version 8 or later as an IMSplex. See *IMS Version 8: Administration Guide: System* for information about configuring an IMSplex.
2. Include the IMS RESLIB load library, which is located in the *hlq*.SDFSRESL data set, in the standard search path for load modules used by your users. *hlq* is the high level qualifier of IMS installed on your system.

If you do not include the IMS load library in the search path, your users will see one or both of the following messages and they will not be able to use the **IMS TM Setup - Manage LE Runtime Options** function in Debug Tool Utilities:

- EQAZ60E REXX IMS SPOC environment is not available. Return Code = nnn
- IKJ56500I COMMAND CSLULXSB NOT FOUND

Chapter 10. Preparing your environment to debug a DB2 stored procedures

The DB2 administrator must define the address space where the stored procedure runs. This can be a DB2 address space or a workload management (WLM) address space. This address space is assigned a name which is used to define the stored procedure to DB2. In the JCL for the DB2 or WLM address space, verify that the following data sets are defined in the STEPLIB concatenation and have the appropriate RACF Read authorization for programs to access them:

- `LOADLIB` for the stored procedure
- `SEQAMOD`[14] for Debug Tool
- `SCEERUN`[15] for Language Environment

After updating the JCL, the DB2 administrator must refresh the DB2 or WLM address space so that these updates take effect.

Refer to the following topics for more information related to the material discussed in this topic.

Related references

DB2 UDB for z/OS Application Programming and SQL Guide

14. Add *hlq*.SEQAMOD to STEPLIB only if it is not already in the system search path (for example, link list). If you create a custom EQAOPTS (as described in Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83) that is not stored in *hlq*.SEQAMOD, then place the data set containing it in STEPLIB (ahead of *hlq*.SEQAMOD if it is in STEPLIB).

15. Add CEE.SCEERUN to STEPLIB only if it is not already in the system search path (for example, link list). If you create a private copy of the Debug Tool Language Environment user exit for DB2 that is linked into CEEPIPI (as described in Chapter 7, "Specifying the TEST runtime options through the Language Environment user exit," on page 33), then place the data set containing it in STEPLIB (ahead of CEE.SCEERUN if it is in STEPLIB).

Chapter 11. Adding support for debugging under CICS

To debug applications that run in CICS, Debug Tool requires the following:

- Language Environment. Refer to the Language Environment installation and customization information for more information.
- Do the steps described in this chapter.

Note: You can use DTCN or CADP to add support for debugging, depending on the version of CICS:

- CICS version 2.2 or earlier: you must use DTCN.
- CICS version 2.3 or later: either DTCN or CADP.

To add Debug Tool support for CICS applications:

1. Verify that the current Debug Tool resources are defined in the CICS CSD and installed in the CICS region. The CICS definitions are in the `EQACCSD` and `EQACDCT` members of the *`hlq`*`.SEQASAMP` data set.
   a. If your site policy is to define the Transient Data queues by using DCT macro definitions, add the definitions in the EQACDCT member to your DCT and reassemble it.

      If your site uses COBOL or PL/I separate debug files, follow the instructions in EQACDCT to define the appropriate queues to CICS.
   b. Add the Debug Tool definitions to the CICS CSD. The following two members are provided in the *`hlq`*`.SEQASAMP` data set:
      - EQACCSD, which contains the resource definitions for the group EQA.
      - EQAWCCSD, which contains JCL to apply the definitions which are in EQACCSD.

      Review the instructions in both members and run the batch job to add the definitions to your CICS CSD.
2. Update the JCL that starts CICS:
   a. Include Debug Tool's *`hlq`*`.SEQAMOD` data set and the Language Environment run-time libraries (`SCEECICS`, `SCEERUN`, and, if required by your applications, `SCEERUN2`) in the `DFHRPL` concatenation. The `DFHRPL` concatenation is in the CICS region start-up JCL.
   b. Remove any data sets from the concatenation that refer to old releases of Debug Tool.
   c. Include `EQA00DYN` and `EQA00HFS` from Debug Tool's *`hlq`*`.SEQAMOD` data set in the `STEPLIB` concatenation by either of the following ways:
      - Use the Authorized Program Facility (APF) to authorize[1] the *`hlq`*`.SEQAMOD` data set and add the data set to the STEPLIB concatenation.
      - Copy[3] the `EQA00DYN` and `EQA00HFS` modules from the *`hlq`*`.SEQAMOD` data set to a library that is already in the `STEPLIB` concatenation.
   d. Ensure that the JCL does not include DD statements for `CINSPIN`, `CINSPLS`, `CINSPOT`, `IBMDBGIN`, or `IGZDBGIN`.
   e. See "Storing DTCN debug profiles in a VSAM file" on page 61 to determine if you want to store DTCN debugging profiles in a VSAM data set. If you do, follow the instructions in that topic to add the EQADPFMB DD statement that refers to the VSAM data set.

3. For any terminal that Debug Tool uses to display a debugging session, do the following tasks:
   - Verify that the CICS TYPETERM definition specifies a minimum value of 4096 for the `RECEIVESIZE` attribute or sets the `BUILDCHAIN` attribute to YES.
   - Enable either color or highlighting. For best usability, enable both and the ability to query the screen size. To enable these three functions, verify that the CICS TYPETERM definition specifies EXTENDEDDS. For more information, refer to the *CICS Transaction Server for z/OS Resource Definition Guide*.
   - Debug Tool can use a screen as large as 62x160, and provides automatic switching from the application's screen size to the physical screen size. Larger screens can enhance user productivity. CICS selects the TYPETERM to use from the BIND information given to it from VTAM. Ask your systems programmer to ensure that VTAM passes the screen sizes through to CICS.
4. Verify that users can run the `CDT#` transaction without receiving any errors.

   If the `CDT#` transaction runs successfully, no messages are displayed. You might see `X-SYSTEM` after you press Enter. This disappears when the transaction finishes and the keyboard unlocks.
5. If you are running your CICS programs in a multi-region CICS environment:
   a. Define the DTCN transaction name the same across all local and remote systems. If the DTCN transaction name is changed, or if a DTCN transaction is duplicated and given a different name, change the name on all systems.
   b. If a debugging session might run in a region that is different from the one where DTCN or CADP was used to save the debugging profile, use the PLTPI program EQA0CPLT with the CICS start up parameter `INITPARM=(EQA0CPLT='NWP')`.
   c. If you are using DTCN, ensure that the region shares the debug profile repository. See "Sharing DTCN debug profile repository among CICS systems" on page 62 for more information about defining the region that owns the debug profile repository. The most common multi-region debugging scenario is where the debug profile repository is shared and DTCN runs in the TOR while the application to be debugged is transaction routed to an AOR.

      One of two methods must be used in this case to start Debug Tool's new program support in the AOR. Either use EQA0CPLT to enable this support when the region starts (see step 9 on page 59 for information about EQA0CPLT), or use the Debug Tool DTCP transaction to start or stop this support as needed. In the AOR, enter DTCPO on a clear CICS screen to activate this support and enter DTCPF to deactivate it. You can activate and deactivate this support multiple times.
   d. If you are using CADP for debugging profiles, set the startup parameter DEBUGTOOL=YES for any region where a Debug Tool session might start. This parameter activates the Debug Tool new program support.
6. If users need to debug Enterprise PL/I for z/OS, Version 3 Release 4 (or later), applications under CICS:
   a. Install the following co-requisites:
      - If you are running z/OS Version 1 Release 6, you need to apply the PTF for Language Environment APAR PK03093.
      - If you are compiling with Enterprise PL/I for z/OS, Version 3 Release 4, apply the PTF for APAR PK03264.

Users can begin a debug session by using DTCN or CADP at either of the following points:

- The entry to programs invoked by EXEC CICS LINK or XCTL.
- The entry to any program, even if it is a nested program within a composite load module, invoked as a static or dynamic CALL.

b. To enable users to start debug sessions with CADP, use PLTPI program EQA0CPLT with the CICS start up parameter `INITPARM=(EQA0CPLT='NWP')`. See step 9 for information about EQA0CPLT.

7. If you are planning to debug command-level assembler application programs that do not run under or use Language Environment services, activate the CICS non-Language Environment exits as described in "Activating CICS non-Language Environment exits" on page 60.
8. If your CICS region is started with the SEC parameter set to YES and the XCMD parameter is set to YES to activate command security, review the access settings for the following resources:

   EXITPROGRAM
   Do one of the following options:

   - Verify that Debug Tool users have UPDATE authority to the EXITPROGRAM resource so that they can run EXEC CICS ENABLE PROGRAM EXIT, DISABLE PROGRAM EXIT, and EXTRACT EXIT.
   - Activate Debug Tool's single-terminal mode screen stacking user exits during CICS start up by doing the following:
     a. Verify that the user ID that runs the CICS region has UPDATE access to the EXITPROGRAM resource.
     b. Add the program EQA0CPLT to your Program List Table (PLTPI).
     c. Add `INITPARM=(EQA0CPLT='STK')` to your CICS startup parameters.

     See step 9 for instructions on using EQA0CPLT.

   TDQUEUE
   Verify that all users have UPDATE authority to the TDQUEUE resource, so that they can run EXEC CICS INQUIRE and EXEC CICS SET TDQUEUE.

   PROGRAM
   Verify that all users have READ authority to the PROGRAM resource, so that they can run EXEC CICS INQUIRE PROGRAM.

   For more information about the CICS security features, see *CICS RACF Security Guide*.
9. (Optional) Set up the CICS PLTPI program called EQA0CPLT:
   a. Add the program EQA0CPLT to your Program List Table (PLTPI). EQA0CPLT initializes parts of Debug Tool during CICS startup as indicated by a CICS INITPARM system initialization parameter. Run EQA0CPLT as a Stage 2 or Stage 3 PLTPI program. The following sample PLT includes EQA0CPLT:

```
TITLE 'DFHPLTXX - IBM Debug Tool CICS Sample PLT'
DFHPLT TYPE=INITIAL,SUFFIX=XX
*
DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
DFHPLT TYPE=ENTRY,PROGRAM=EQA0CPLT
*
DFHPLT TYPE=FINAL  END DFHPLTBA
```

b. Add the INITPARM keyword to the CICS startup parameters. Multiple parameters can be passed to EQA0CPLT in the same INITPARM. The following common parameters can be used:

   NLE
   Non-Language Environment support. See "Activating CICS non-Language Environment exits."

   STK
   Screen stack exits. This parameter is required if you are using command security.

   NWP
   New program support. This parameter is required if you are using multi-regions or Enterprise PL/I Version 3 Release 4 (or later) with CADP.

   For example, to activate the non-Language Environment support, screen stack exits, and new program support (multi-region and Enterprise PL/I Version 3 Release 4 with CADP) in a single INITPARM, add the following to your CICS startup parameters:

   ```
   INITPARM=(EQA0CPLT='NLE,STK,NWP')
   ```

   Any combination of these three can be coded on the same INITPARM.

10. If the users use COBOL or PL/I separate debug files, verify that the users specify the following attributes for the PDS or PDSE that contains the separate debug files:
    - RECFM=FB
    - LRECL=1024
    - BLKSIZE set so that the system determines the optimal size

    Important: Users must allocate files with the correct attributes to optimize the performance of Debug Tool.

11. (Optional) Increase the DSALIM and EDSALIM sizes in your CICS region so that Debug Tool functions properly with multiple concurrent users. The amount of increase is based on the current workload in the CICS region.

    Recommendation: Increase the sizes of DSALIM and EDSALIM in increments of 5% or 10%. Monitor the storage in the region as Debug Tool users are debugging for the highest amount of storage that is used at any one point.

12. If DTCN users want to start Debug Tool at a program boundary for CICS tasks that have already started, install Dynamic Debug as described in Chapter 3, "Installing the Dynamic Debug facility," on page 7.

See the *Debug Tool User's Guide* for information about how to debug CICS programs.

Activating CICS non-Language Environment exits

To debug non-Language Environment assembler programs or non-Language Environment COBOL programs that run under CICS, you must start the required Debug Tool global user exits before you start the programs. Debug Tool provides the following global user exits to help you debug non-Language Environment applications: XPCFTCH, XEIIN, XEIOUT, XPCTA, and XPCHAIR. The exits can be started by using either the DTCX transaction (provided by Debug Tool), or using a PLTPI program that runs during CICS region startup

DTCX: You can turn the exits on and off by using the transaction DTCX. To activate all of the exits, from a clear CICS terminal screen enter DTCXXO. To deactivate all of the exits, enter DTCXXF. You need to activate the exits only once. If you deactivate the exits and then want to debug a non-Language Environment program, you need to enter DTCXXO from a clear CICS terminal screen to activate the exits.

After you enter DTCXXO, a series of messages are displayed on your screen. If all exits are activated successfully, the following messages are displayed:

```
EQA9972I - DT XPCFTCH CICS exit now ON.
EQA9972I - DT XEIIN exit now ON.
EQA9972I - DT XEIOUT exit now ON.
EQA9972I - DT XPCTA exit now ON.
EQA9972I - DT XPCHAIR exit now ON.
EQA9970I - CICS exit activation successful.
```

When you enter DTCXXF, the following messages are displayed:

```
EQA9973I - DT XPCFTCH CICS exit now OFF.
EQA9973I - DT XEIIN exit now OFF.
EQA9973I - DT XEIOUT exit now OFF.
EQA9973I - DT XPCTA exit now OFF.
EQA9973I - DT XPCHAIR exit now OFF.
EQA9971I - CICS exit deactivation successful.
```

If there is a problem starting or activating one of the exits, an error message like the following is displayed:

```
EQA9974I Error enabling XPCFTCH - EQANCFTC
```

If you see this error message, verify that the CICS CSD is properly updated to include the latest Debug Tool resource definitions, and that the Debug Tool SEQAMOD data is in the DFHRPL DD concatenation for the CICS region.

You can start the exits during region initialization by using a sequential terminal or any other mechanism that runs transactions during CICS startup. You are not required to shut down the exits before or during a region shutdown.

PLT: The non-Language Environment exits can also be activated during CICS region initialization by using the CICS Program List Table (PLTPI) program EQA0CPLT (supplied by Debug Tool). In addition to adding EQA0CPLT to your CICS region PLT, you must specify the CICS startup parameter `INITPARM=(EQA0CPLT='NLE')`. EQA0CPLT supersedes the function provided earlier by PLTPI program EQANCPLT. See step 9 on page 59 for instructions on using EQA0CPLT. For more information about PLT processing, see the *CICS Resource Definition Guide*.

Storing DTCN debug profiles in a VSAM file

By default, the CICS DTCN transaction stores its debugging profiles into a CICS temporary storage queue (TSQ) called EQADTCN2. Because CICS destroys temporary storage queues at region termination, any profiles stored in EQADTCN2 are deleted when a region is stopped. To save debugging profiles across region termination and restart or after the owning terminal is disconnected, store the profiles into a VSAM data set.

Do the following steps to instruct DTCN to store its debugging profiles in a VSAM data set:

1. Create the VSAM data set by following the instructions in the EQAWCRVS member of the *hlq*.SEQASAMP data set.
2. Modify the CICS region startup JCL so that the EQADPFMB DD statement identifies the VSAM data set you created in step 1.
3. Define the VSAM file to the CICS region by following the instructions in the EQACCSD member of the *hlq*.SEQASAMP data set. "Sharing DTCN debug profile repository among CICS systems" also describes examples of CICS resource definitions.

Sharing DTCN debug profile repository among CICS systems

The DTCN debug profile repository is either a CICS temporary storage queue called EQADTCN2 or a VSAM data set identified through the EQADPFMB DD statement. If you want to share the repository among CICS systems (for example, MRO), do one of the following options:

- If you are using a temporary storage queue, do the following steps:
  1. Designate a single CICS region as the *queue-owning region* and note the SYSID of that region. In Figure 1, the SYSID of the queue-owning region is P6.
  2. For all other regions that need to access the queue-owning region, create a TSMODEL resource definition and verify that you define the following attributes:
     - For the REMOTESystem attribute, specify the SYSID of the queue-owning region.
     - For PRefix and REMOTEPrefix attribute, specify EQADTCN2.
     - To optimize the performance of Debug Tool, define the Location attribute as MAIN.

```
CEDA View TSmodel( DTCN1 )
TSmodel      ==> DTCN1
Group        ==> DTCNREM
Description  ==> TEST DTCN TSQ REMOTE
PRefix       ==> EQADTCN2
XPrefix      ==>
Location     ==> Main                Auxiliary | Main
RECOVERY ATTRIBUTES
RECovery     ==> No                  No | Yes
SECURITY ATTRIBUTES
Security     ==> No                  No | Yes
SHARED ATTRIBUTES
POolname     ==>
REMOTE ATTRIBUTES
REMOTESystem ==> P6
REMOTEPrefix ==> EQADTCN2
XRemotepfx   ==>
Group        ==>
```

Figure 1. A sample TSMODEL resource definition that gives a region access to the queue-owning region called P6.

For instructions on how to create a TSMODEL resource definition, see *CICS Resource Definition Guide*.

- If you are using a VSAM data set and want to function-ship file operations to a file-owning region (FOR), designate a single FOR and then define the file as REMOTE in the CICS FILE definition on regions that need to access it remotely.

  The following sample resource definition shows how to define the Debug Tool EQADPFMB file in a region that uses it remotely.

```
CEDA  View File( EQADPFMB )
 File           : EQADPFMB
 Group          : DTCNREM
 DEScription    : DTCN PROFILE DATASET REMOTE
VSAM PARAMETERS
 DSNAme         :
 Password       :                   PASSWORD NOT SPECIFIED
 RLsaccess      : No                Yes | No
 LSrpoolid      : 1                 1-8 | None
 READInteg      : Uncommitted       Uncommitted | Consistent | Repeatable
 DSNSharing     : Allreqs           Allreqs | Modifyreqs
 STRings        : 001               1-255
 Nsrgroup       :
REMOTE ATTRIBUTES
 REMOTESystem   : P6
 REMOTEName     : EQADPFMB
REMOTE AND CFDATATABLE PARAMETERS
 RECORDSize     :                   1-32767
 Keylength      :                   1-255 (1-16 For CF Datatable)
INITIAL STATUS
 STAtus         : Enabled           Enabled | Disabled | Unenabled
 Opentime       : Firstref          Firstref | Startup
 DIsposition    : Share             Share | Old
BUFFERS
 DAtabuffers    : 00002             2-32767
 Indexbuffers   : 00001             1-32767
DATATABLE PARAMETERS
 TABLE          : No                No | CIcs | User | CF
 Maxnumrecs     : Nolimit           Nolimit | 1-99999999
CFDATATABLE PARAMETERS
 Cfdtpool       :
 TABLEName      :
 UPDATEModel    : Locking           Contention | Locking
 LOad           : No                No | Yes
DATA FORMAT
 RECORDFormat   : V                 V | F
OPERATIONS
 Add          ==> No                No | Yes
 BRowse       ==> No                No | Yes
 DELete       ==> No                No | Yes
 READ         ==> Yes               Yes | No
 UPDATE       ==> No                No | Yes
AUTO JOURNALLING
 JOurnal      ==> No                No | 1-99
 JNLRead      ==> None              None | Updateonly | Readonly | All
 JNLSYNCRead  ==> No                No | Yes
 JNLUpdate    ==> No                No | Yes
 JNLAdd       ==> None              None | Before | AFter | ALl
 JNLSYNCWrite ==> Yes               Yes | No
RECOVERY PARAMETERS
 RECOVery     ==> None              None | Backoutonly | All
 Fwdrecovlog  ==> No                No | 1-99
 BAckuptype   ==> Static            Static | Dynamic
SECURITY
 RESsecnum      : 00                0-24 | Public
```

For the region which owns the VSAM file, omit the REMOTESYSTEM and REMOTENAME values in the FILE definition.

For details on defining a FILE resource, see *CICS Resource Definition Guide*.

- If you are using a VSAM data set and prefer to define the file locally to all CICS regions that use it, define the file on all such regions using record-level sharing (RLS). The following sample resource definition shows how to define the Debug Tool EQADPFMB file using RLS.

```
CEDA  View File( EQADPFMB )
 File           : EQADPFMB
 Group          : DTCNRLS
 DEScription    : DTCN PROFILE DATASET
VSAM PARAMETERS
 DSNAme         :
 Password       :                    PASSWORD NOT SPECIFIED
 RLsaccess      : Yes                Yes | No
 LSrpoolid      : 1                  1-8 | None
 READInteg      : Repeatable         Uncommitted | Consistent | Repeatable
 DSNSharing     : Allreqs            Allreqs | Modifyreqs
 STRings        : 010                1-255
 Nsrgroup       :
REMOTE ATTRIBUTES
 REMOTESystem   :
 REMOTEName     :
REMOTE AND CFDATATABLE PARAMETERS
 RECORDSize     :                    1-32767
 Keylength      :                    1-255 (1-16 For CF Datatable)
INITIAL STATUS
 STAtus         : Enabled            Enabled | Disabled | Unenabled
 Opentime       : Firstref           Firstref | Startup
 DIsposition    : Share              Share | Old
BUFFERS
 DAtabuffers    : 00011              2-32767
 Indexbuffers   : 00010              1-32767
DATATABLE PARAMETERS
 TABLE          : No                 No | CIcs | User | CF
 Maxnumrecs     : Nolimit            Nolimit | 1-99999999
CFDATATABLE PARAMETERS
 Cfdtpool       :
 TABLEName      :
 UPDATEModel    : Locking            Contention | Locking
 LOad           : No                 No | Yes
DATA FORMAT
 RECORDFormat   : V                  V | F
OPERATIONS
 Add            : Yes                No | Yes
 BRowse         : Yes                No | Yes
 DELete         : Yes                No | Yes
 READ           : Yes                Yes | No
 UPDATE         : Yes                No | Yes
AUTO JOURNALLING
 JOurnal        : No                 No | 1-99
 JNLRead        : None               None | Updateonly | Readonly | All
 JNLSYNCRead    : No                 No | Yes
 JNLUpdate      : No                 No | Yes
 JNLAdd         : None               None | Before | AFter | ALl
 JNLSYNCWrite   : No                 Yes | No
RECOVERY PARAMETERS
 RECOVery       : None               None | Backoutonly | All
 Fwdrecovlog    : No                 No | 1-99
 BAckuptype     : Static             Static | Dynamic
SECURITY
 RESsecnum      : 00                 0-24 | Public
```

For details on defining a FILE resource, see *CICS Resource Definition Guide*.

Deleting or deactivating debug profiles stored in a VSAM data set

If you are storing debug profiles in a VSAM data set, as described in "Storing DTCN debug profiles in a VSAM file" on page 61, the number of profiles no longer in use might become large, because the debug profiles persist across region restarts and after the terminal from which a profile was created has been disconnected. Debug Tool provides two transactions, DTCD and DTCI, to delete or deactivate debug profiles stored in a region's VSAM data set.

To delete debug profiles in the VSAM data set identified by the EQADPFMB DD statement on your region, use the DTCD transaction. The following diagram describes the syntax of the DTCD transaction:

userid
Delete the debug profile associated with a specific CICS user ID.

* Deletes debug profiles from the VSAM data set. This option requires specific RACF authority; therefore, reserve it for CICS administrators.

To deactivate all debugging profiles in the VSAM data set, use the DTCI transaction. The following diagram describes the syntax of the DTCI transaction:

The following list describes the parameters:

userid
Deactivate the debug profile associated with a specific CICS user ID.

* Deactivate debug profiles from the VSAM data set. This option requires specific RACF authority; therefore, reserve it for CICS administrators.

Refer to the following topics for more information related to the material discussed in this topic.

Related tasks
"Authorizing DTCD and DTCI transactions to delete or deactive debug profiles" on page 76

Requiring users to specify resource types

If your users use DTCN to specify debugging profiles, you can customize Debug Tool to require that your users specify some or all resource types. For example, if your users are debugging a heavily used CICS program, you can require that they specify a Terminal ID and a Transaction ID to avoid having Debug Tool started every time that CICS program is run. You can enforce these requirements by specifying the corresponding DTCNFORCE*xxxx* option in the EQAOPTS options file, described in "DTCNFORCE*xxxx*" on page 88.

Direct QSAM access through a CICS task-related user exit

Debug Tool can use two methods to access the following types of files:

- Enterprise COBOL and Enterprise PL/I separate debug files (SYSDEBUG)
- C/C++ separate debug files (.dbg and .mdbg)
- assembler and non-Language Environment COBOL EQALANGX files
- listing and source files
- command and preference files
- save settings and save breakpoints and monitor specification files
- log files

The following list describes both access methods:

- CICS Extrapartition Transient Data (default method)
- Direct QSAM access through a CICS task-related user exit

If you want the access method to avoid using CICS SPI and API to access these files, enable the QSAM access method.

To enable the QSAM access method, use the following INITPARM in your CICS start up parameters:

```
INITPARM=(DFHLETRU='USEQSAM')
```

You also need to apply the following PTFs to the appropriate products:

- For CICS Transaction Server for z/OS, Version 3.1, apply the PTF for PK67329
- For CICS Transaction Server for z/OS, Version 3.2, apply the PTF for PK68401
- For Enterprise COBOL compilers, apply the PTF for PK71852 to Language Environment, Version 1.8 through 1.10
- For Enterprise PL/I compilers, apply the PTF for PK93564 to Language Environment, Version 1.8 through 1.11

Enabling communication between Debug Tool and a remote debugger

This topic helps you activate the appropriate TCP/IP socket interface, which manages communication between your CICS region and the remote debugger. There are two TCP/IP socket interfaces: the TCP/IP Socket Interface for CICS and the CICS Socket Domain. Activating the correct interface enables the following functions:

- Communication between your CICS region and the remote debugger.
- Use of the IPv6 protocol in remote debug mode.

If you are using CICS Transaction Server Version 4, Debug Tool selects the interface according to the following rules:

- If the CICS Socket Domain is active, Debug Tool selects the CICS Socket Domain.
- If the CICS Socket Domain is inactive and the TCP/IP Socket Interface for CICS is active, Debug Tool selects the TCP/IP Socket Interface for CICS.

If you are using CICS Transaction Server Version 2 or Version 3, Debug Tool selects the interface according to the following rules:

- If the TCP/IP Socket Interface for CICS is active, then Debug Tool selects the TCP/IP Socket Interface.
- If the TCP/IP Socket Interface for CICS is inactive and the CICS Socket Domain is active, then Debug Tool selects the CICS Socket Domain.

If you are using CICS Transaction Server Version 4 and the IPv6 protocol, you must activate the CICS Socket Domain. If you are using CICS Transaction Server Version 2 or Version 3 and the IPv6 protocol, you must activate the TCP/IP Socket Interface for CICS.

To activate the TCP/IP Socket Interface for CICS, see the z/OS Communications Server *IP CICS Sockets Guide*.

To activate the CICS Socket Domain, do the following tasks:

1. Ensure that the CICS system initialization parameter TCPIP is set to YES. For more information about the CICS system initialization parameters, see the *CICS System Definition Guide*.
2. Install the IBM-supplied group DFHSO, which contains the resource definitions for External sockets support. For information about installing this group, see the CICS migration guide that is appropriate for your migration path. A list of migration guides is available in the CICS Transaction Server for z/OS information center.

Enabling the CADP transaction

Beginning with CICS Transaction Server for z/OS Version 2 Release 3, you can use the debugging profiles created by the application debugging profile manager (CADP transaction) with Debug Tool. Set the DEBUGTOOL system initialization parameter to YES to indicate that Debug Tool must use debugging profiles created by the CADP transaction. With the DEBUGTOOL system initialization parameter set to YES, you cannot use DTCN to define debugging profiles.

The default setting of DEBUGTOOL=NO indicates that Debug Tool will not use CADP profiles and will use DTCN-defined profiles. With DEBUGTOOL=NO, you can use CADP to update or add debugging profiles, but these profiles will not be used by Debug Tool.

You can dynamically switch between the CADP and DTCN debug profiles that are used by Debug Tool. After the CICS region is started, enter `CEMT SET DEBUG` to have CADP profiles used and `CEMT SET NODEBUG` to have DTCN profiles used.

Running multiple debuggers in a CICS region

Coexistence with other debuggers cannot be guaranteed since situations can occur where multiple debuggers might contend for use of storage, facilities and interfaces which are intended for only one requester.

It is suggested that if you must have multiple debuggers installed in a CICS region, then only one should be active at any given time. When another debugger is used, ensure that the Debug Tool CICS non-Language Environment user exits are deactivated and that there are no active CADP or DTCN profiles in the region. The user exits can be deactivated by issuing the DTCXXF transaction. To deactivate other debuggers, consult the documentation provided by the vendor of the other debuggers.

Running the installation verification programs

To help you verify that your CICS region has been customized properly for Debug Tool, the *hlq*.SEQASAMP data set contains installation verification programs (IVPs) in the following members. Run the IVPs that are appropriate for the tasks that your users will be performing.

IVP	Task
EQAWIVCI	Dynamic Debug facility and Enterprise PL/I `TEST(ALL,SYM,NOHOOK,SEPARATE)`
EQAWIVCP	Dynamic Debug facility and COBOL `TEST(NONE,SYM,SEPARATE)` or `TEST(NOHOOK,SEPARATE)`
EQAWIVC2	C `TEST(ALL)`

IVP	Task
EQAWIVCG	C `DEBUG(FORMAT(DWARF),HOOK(LINE,NOBLOCK,PATH),SYMBOL)`
EQAWIVC8	Enterprise PL/I `TEST(ALL)`
EQAWIVCC	Non-Language Environment Assembler

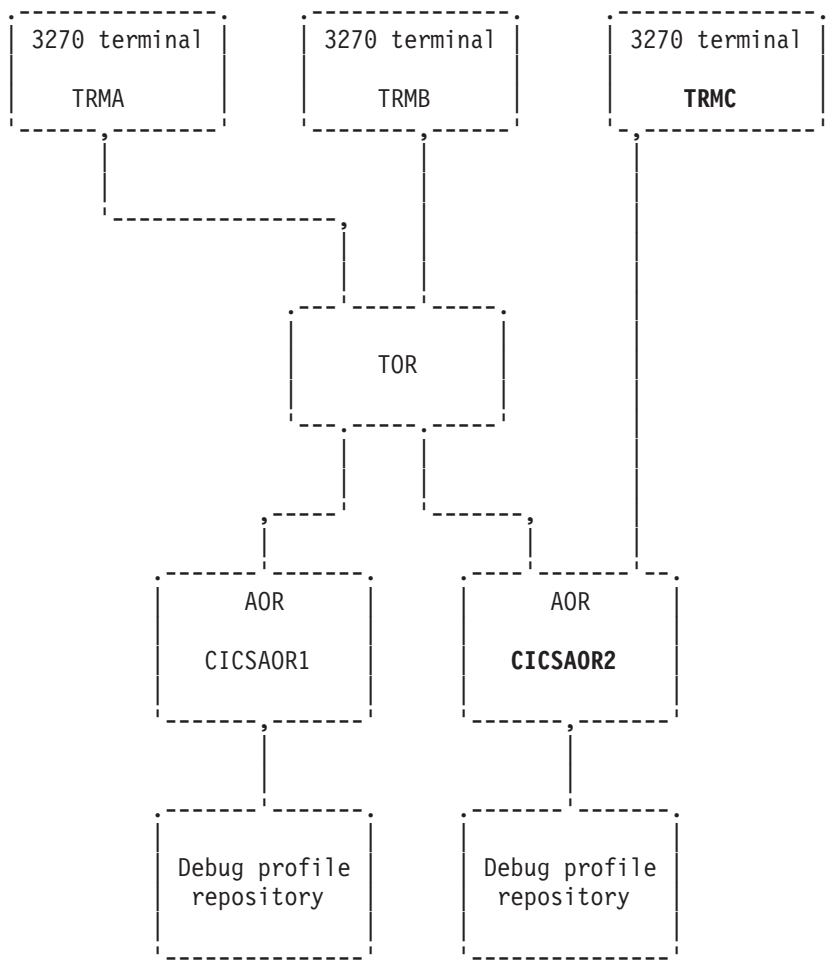
Configuring Debug Tool to run in a CICSplex environment

In a CICSplex, the application-owning regions (AORs), terminal-owning regions (TORs), queue-owning regions (QORs), repositories, and terminals can be organized in an infinite number of ways. In the following topics, we explore a finite number of scenarios and let you know what you need to do to configure Debug Tool to work in each scenario. For all of these scenarios, we assume you are working in full screen mode.

- "Terminal connects to an AOR that runs the application"
- "Terminal connects to a TOR which routes the application to an AOR; debugging profiles managed by CADP" on page 69
- "Terminal connects to a TOR which routes the application to an AOR; debugging profiles managed by DTCN" on page 70
- "Terminal connects to an AOR that runs an application that does not use a terminal" on page 72
- "Screen control mode terminal connects to a TOR and application runs in an AOR" on page 73
- "Separate terminal mode terminal connects to a TOR and application runs in an AOR" on page 73

Terminal connects to an AOR that runs the application

In this scenario, your terminal (TRMC) connects to an AOR (CICSAOR2) that runs the application you want to debug. The debugging profiles can be managed by either CADP or DTCN and they are directly accessible by the AOR.



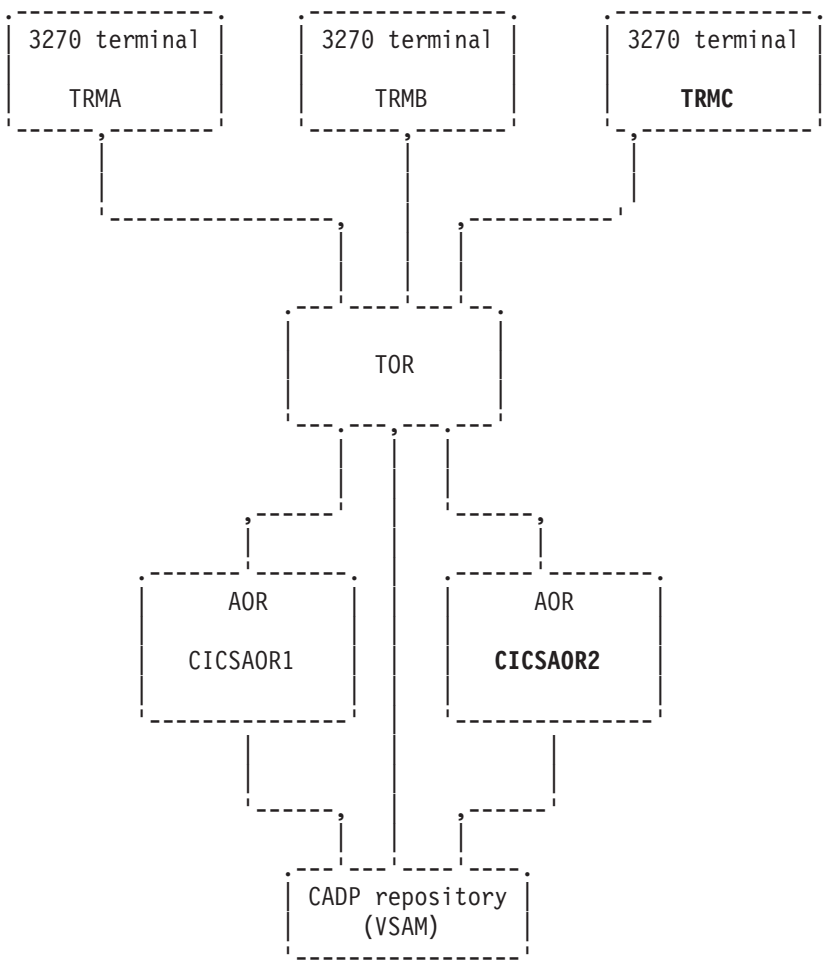
For this scenario to work, the CICS system administrator must complete the following tasks for the region CICSAOR2:

- Define Debug Tool resources in the CICS CSD and install them in the CICS region, as described in Chapter 11, "Adding support for debugging under CICS," on page 57, step 1.
- Provide access to these resources, as described in Chapter 11, "Adding support for debugging under CICS," on page 57, step 2a on page 57.

If you want to debug an application that runs in another AOR region, like CICSAOR1, you must log on to that region and verify that the system administrator completed the above tasks for that region.

Terminal connects to a TOR which routes the application to an AOR; debugging profiles managed by CADP

In this scenario, your terminal (TRMC) connects to a TOR, which uses a CICS transaction to route the application you want to debug to an AOR. The debugging profiles can be managed by either CADP or DTCN and they are directly accessible by the AOR. The CADP repository is a VSAM data set which is shared between all of the regions. You can run the CADP transaction in any of the regions.



For this scenario to work, the CICS system administrator must complete the following tasks for both AORs:

- Define Debug Tool resources in the CICS CSD and install them in the CICS region, as described in Chapter 11, "Adding support for debugging under CICS," on page 57, step 1.
- Provide access to these resources, as described in Chapter 11, "Adding support for debugging under CICS," on page 57, step 2a on page 57.
- Run the correct programs and use the correct CICS start up parameters for each type of profile, as described in the following steps:

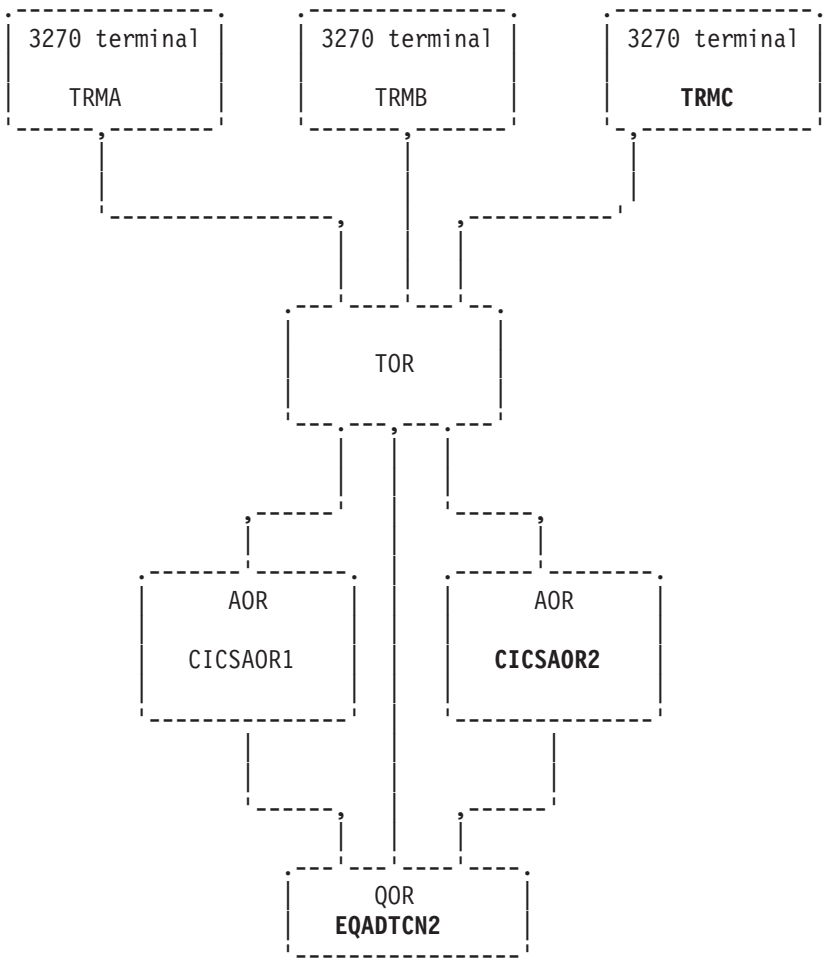
  CADP Chapter 11, "Adding support for debugging under CICS," on page 57, step 5d on page 58, 6b on page 59, 9b on page 60, and "Enabling the CADP transaction" on page 67.

  DTCN
  Chapter 11, "Adding support for debugging under CICS," on page 57, step 5a on page 58 and 9b on page 60.

Terminal connects to a TOR which routes the application to an AOR; debugging profiles managed by DTCN

In this scenario, your terminal (TRMC) connects to a TOR, which uses a CICS transaction to route the application you want to debug to an AOR. The debugging

profiles are managed by DTCN and are stored in a temporary storage queue (EQADTCN2) located in a queue-owning region (QOR). You can run the DTCN transaction in any of the regions.



For this scenario to work, the CICS system administrator must complete the following tasks for both AORs and the TOR:

- Define Debug Tool resources in the CICS CSD and install them in the CICS region, as described in Chapter 11, "Adding support for debugging under CICS," on page 57, step 1.
- Provide access to these resources, as described in Chapter 11, "Adding support for debugging under CICS," on page 57, step 2a on page 57.
- Designate a single CICS region as the QOR and define the queue accessible remotely, as described in "Sharing DTCN debug profile repository among CICS systems" on page 62.

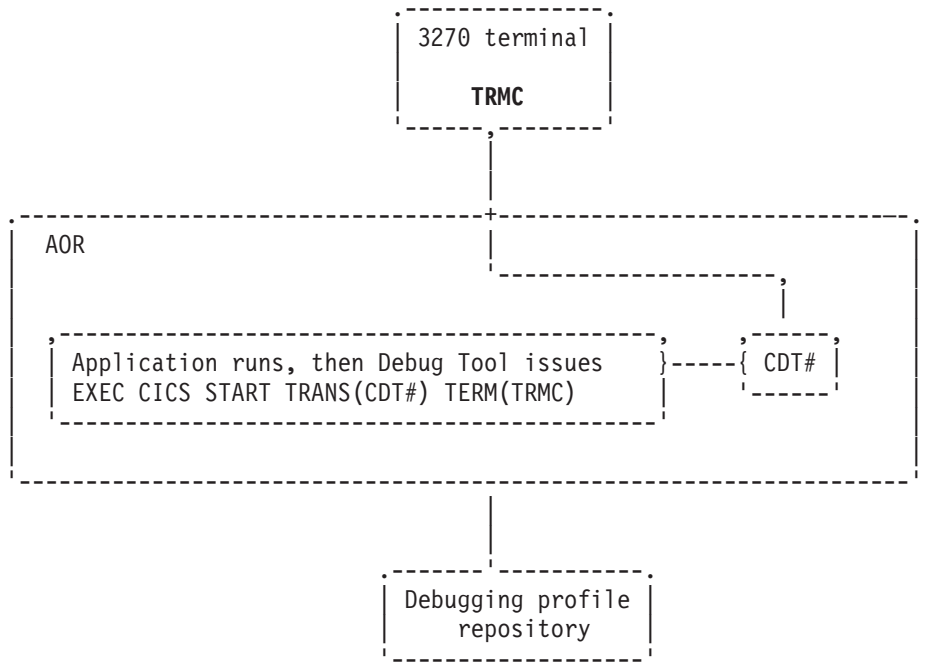
Variation on this scenario: The temporary storage queue (EQADTCN2) does not need to be located in a QOR. It can be located in the TOR, any of the AORs, or in the coupling facility. Wherever you put the temporary storage queue, keep the following considerations in mind:

- Place the queue where it can be accessed efficiently when the application programs begin, since it is referenced at that point to determine whether the program should be debugged.

- The temporary storage queue is accessed by Function Shipping, so allocate a sufficient number of connections between the regions to handle READQ requests.

Terminal connects to an AOR that runs an application that does not use a terminal

In this scenario, your terminal (TRMC) connects to an AOR, which you use to set up a debugging profile using either CADP or DTCN. When the application starts, Debug Tool is started and issues and EXEC CICS START of its display transaction (CDT#) on your terminal (TRMC). Your terminal must be connected directly to the AOR. You cannot connect through CRTE because CICS does not support issuing an EXEC CICS START to a terminal connected through CRTE.



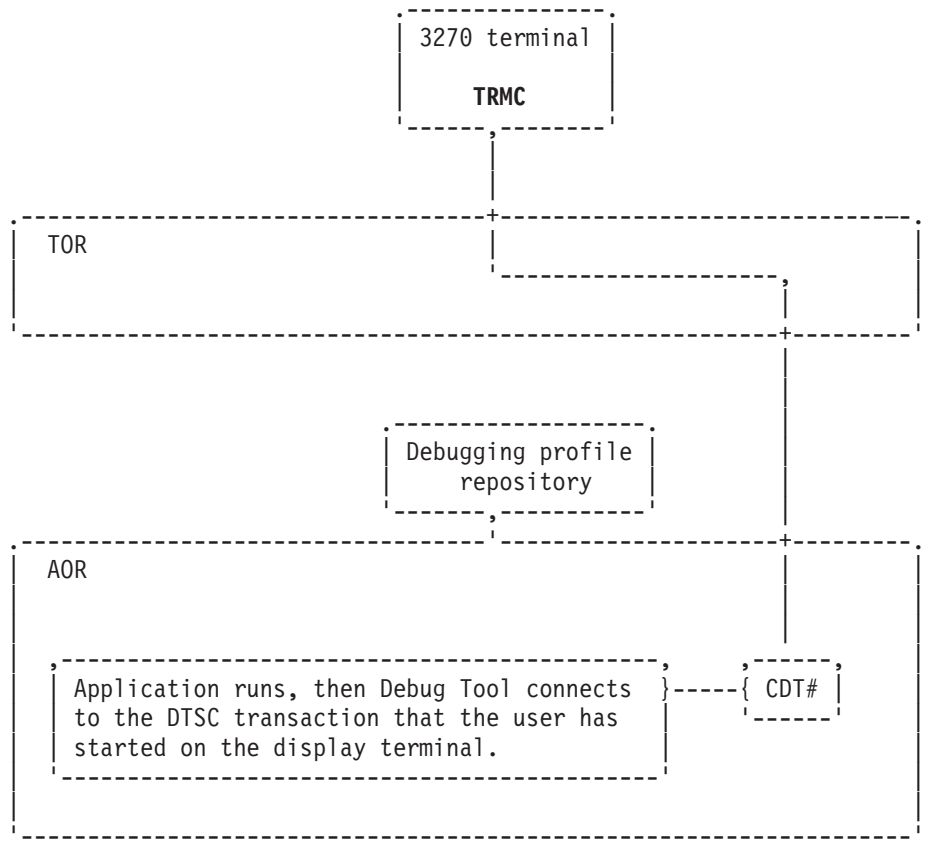
For this scenario to work, the CICS system administrator must complete the following tasks for the AOR:

- Define Debug Tool resources in the CICS CSD and install them in the CICS region, as described in Chapter 11, “Adding support for debugging under CICS,” on page 57, step 1 on page 57.
- Provide access to these resources, as described in Chapter 11, “Adding support for debugging under CICS,” on page 57, step 2a on page 57.
- If you are using CADP to manage debugging profiles, then run the correct programs and use the correct CICS start up parameters, as described in Chapter 11, “Adding support for debugging under CICS,” on page 57, Chapter 11, “Adding support for debugging under CICS,” on page 57, step 5d on page 58, 6b on page 59, 9b on page 60, and “Enabling the CADP transaction” on page 67.

Screen control mode terminal connects to a TOR and application runs in an AOR

In this scenario, the user starts the DTSC transaction on the display terminal to display the debug session. DTSC must run in the same region as the application, but could run in any of the following situations:

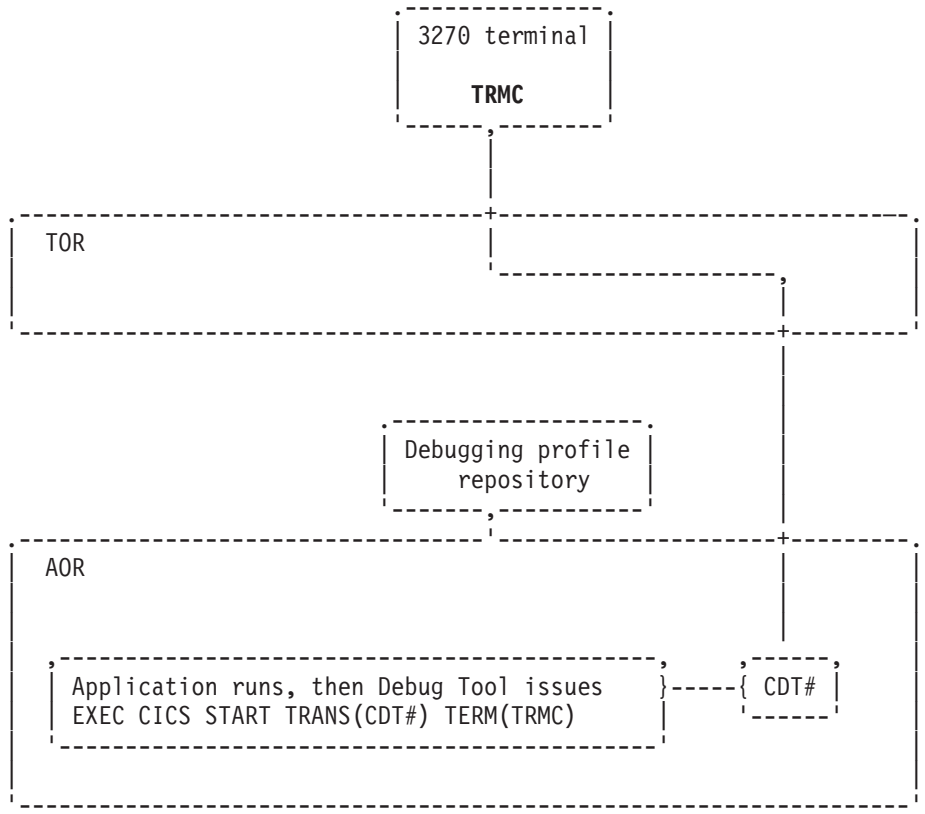
- As a Transaction-Routed transaction
- On a CRTE terminal session which was started on the AOR



Separate terminal mode terminal connects to a TOR and application runs in an AOR

In this scenario, your terminal (TRMC) connects to a TOR and the following sequence of events occurs:

1. You store a debugging profile into a repository using either DTCN or CADP.
2. The application starts. The profile matches the application so Debug Tool is started.
3. Debug Tool issues EXEC CICS START of its display transaction (CDT#) on your terminal (TRMC). However, your terminal is not found. XICTENF/XALTENF identifies the TOR as the owner of your terminal (TRMC).
4. CICS routes the START task to the TOR identified by XICTENF/XALTENF.
5. Interval Control in the TOR associates the START task with your terminal (TRMC) and then routes the START task back to the AOR.
6. CDT# establishes the communication between your terminal and the application through the TOR.



For this scenario to work, the CICS system administrator must complete the following tasks for the TOR:

- If you are using DTCN to manage debugging profiles, do the following tasks:
  –
  – Define Debug Tool resources in the CICS CSD and install them in the CICS region, as described in Chapter 11, “Adding support for debugging under CICS,” on page 57, step 1 on page 57.
  – Provide access to these resources, as described in Chapter 11, “Adding support for debugging under CICS,” on page 57, step 2a on page 57.
- If you are using CADP to manage debugging profiles, then run the correct programs and use the correct CICS start up parameters, as described in Chapter 11, “Adding support for debugging under CICS,” on page 57, Chapter 11, “Adding support for debugging under CICS,” on page 57, step 5d on page 58, 6b on page 59, 9b on page 60, and “Enabling the CADP transaction” on page 67.
- Enable routing of the terminal traffic to the correct terminal by configuring the Debug Tool transaction CDT# as DYNAMIC(YES).

For this scenario to work, the CICS system administrator must complete the following tasks for the AOR:

- If you are using DTCN to manage debugging profiles, do the following tasks:
  – Define Debug Tool resources in the CICS CSD and install them in the CICS region, as described in Chapter 11, “Adding support for debugging under CICS,” on page 57, step 1 on page 57.
  – Provide access to these resources, as described in Chapter 11, “Adding support for debugging under CICS,” on page 57, step 2a on page 57.

- If you are using CADP to manage debugging profiles, then run the correct programs and use the correct CICS start up parameters, as described in Chapter 11, "Adding support for debugging under CICS," on page 57, Chapter 11, "Adding support for debugging under CICS," on page 57, step 5d on page 58, 6b on page 59, 9b on page 60, and "Enabling the CADP transaction" on page 67.
- To locate the terminal, do the following steps:
  - Code the CICS exits XICTENF and XALTENF so that the TOR is identified as the owner of the display terminal. The *CICS Transaction Server for z/OS Customization Guide* describes these exits.
  - Run a PLT program that enables the CICS exits XICTENF and XALTENF. The *CICS Transaction Server for z/OS Customization Guide* describes how to write and run a PLT.
  - Enable routing of the terminal traffic to the correct terminal by configuring the Debug Tool transaction CDT# as DYNAMIC(YES).

Authorizing DTST transaction to modify storage

This topic describes the steps you must take to authorize the DTST transaction to modify either USER-key storage, CICS-key storage, or both. DTST does not allow users to modify Key-0 storage.

The following resources control DTST authorizations:

- EQADTOOL.DTSTMODUSERK, which controls the ability to modify USER-key storage.
- EQADTOOL.DTSTMODCICSK, which controls the ability to modify CICS-key storage.

1. Establish profiles in the FACILITY class by entering the following RDEFINE commands:

   ```
   RDEFINE FACILITY EQADTOOL.DTSTMODUSERK UACC(NONE)
   RDEFINE FACILITY EQADTOOL.DTSTMODCICSK UACC(NONE)
   ```

2. Verify that generic profile checking is in effect for the class FACILITY by entering the following command:

   ```
   SETROPTS GENERIC(FACILITY)
   ```

3. Give a user permission to modify USER-key, CICS-key storage, or both by entering one or both of the following commands, where DUSER1 is the name of a RACF-defined user or group profile:

   ```
   PERMIT EQADTOOL.DTSTMODUSERK CLASS(FACILITY) ID(DUSER1) ACCESS(UPDATE)
   PERMIT EQADTOOL.DTSTMODCICSK CLASS(FACILITY) ID(DUSER1) ACCESS(UPDATE)
   ```

   Instead of connecting individual users, the security administrator can specify DUSER1 to be a RACF group profile and then connect authorized users to the group.

4. If the FACILITY class is not active, activate the class by entering the following SETROPTS command:

   ```
   SETROPTS CLASSACT(FACILITY)
   ```

   Enter the SETROPTS LIST command to verify that FACILITY class is active.

5. Refresh the FACILITY class by entering the following SETROPTS RACLIST command:

   ```
   SETROPTS RACLIST(FACILITY) REFRESH
   ```

Authorizing DTCD and DTCI transactions to delete or deactive debug profiles

This topic describes the steps you must take to authorize the DTCD and DTCI transactions to delete or deactivate debug profiles stored in a VSAM data set.

The EQADTOOL.DTCDDELETEALL resource controls DTCD authorizations.

The EQADTOOL.DTCIINACTALL resource controls DTCI authorizations.

To authorize DTCD and DTCI users so they can delete or deactivate debug profiles stored in a VSAM data set, do the following steps:

1. Establish profiles in the FACILITY class by entering the following RDEFINE commands:

   ```
   RDEFINE FACILITY EQADTOOL.DTCDDELETEALL UACC(NONE)
   RDEFINE FACILITY EQADTOOL.DTCIINACTALL UACC(NONE)
   ```

2. Verify that generic profile checking is in effect for the class FACILITY by entering the following command:

   ```
   SETROPTS GENERIC(FACILITY)
   ```

3. Give a user permission to delete or deactivate debug profiles stored in a VSAM data set by entering the following commands, where DUSER1 is the name of a RACF-defined user or group profile:

   ```
   PERMIT EQADTOOL.DTCDDELETEALL CLASS(FACILITY) ID(DUSER1) ACCESS(UPDATE)
   PERMIT EQADTOOL.DTCIINACTALL CLASS(FACILITY) ID(DUSER1) ACCESS(UPDATE)
   ```

   Instead of connecting individual users, the security administrator can specify DUSER1 to be a RACF group profile and then connect authorized users to the group.

4. If the FACILITY class is not active, activate the class by entering the following SETROPTS command:

   ```
   SETROPTS CLASSACT(FACILITY)
   ```

   Enter the SETROPTS LIST command to verify that FACILITY class is active.

5. Refresh the FACILITY class by entering the following SETROPTS RACLIST command:

   ```
   SETROPTS RACLIST(FACILITY) REFRESH
   ```

Chapter 12. Adding support for debugging under IMS

To add support for debugging applications that run in IMS, you need to do the following steps:

1. Choose one of the following methods for specifying TEST run time options:
   - Specifying the TEST run time options in a data set, created by the application programmers, which is then extracted by a customized version of the Language Environment user exit routine CEEBXITA.
   - Specifying the TEST run time options in one of the following assembler modules:
     - CEEUOPT, which is an assembler module that uses the CEEXOPT macro to set application level defaults, and is link-edited into an application program.
     - CEEROPT, which is an assembler module that uses the CEEXOPT macro to set region level defaults.
   - Specifying the TEST run time options through the `EQASET` transaction. The transaction allows application programmers to specify a limited set of TEST run time options
2. Choose from the following scenarios that best matches your site's environment:

   Scenario A
   You run programs in IMS Transaction Manager, BTS, or DB and are managing TEST run time options with a user exit. Do the steps described in “Scenario A: Running IMS and managing TEST run time options with a user exit” on page 78 to enable this scenario.

   Scenario B
   You run programs in IMS Transaction Manager, BTS, or DB and are managing TEST run time options with CEEUOPT or CEEROPT. Do the steps described in“Scenario B: Running IMS and managing TEST run time options with CEEUOPT or CEEROPT” on page 78 to enable this scenario.

   Scenario C
   You run assembler programs without Language Environment in IMS Transaction Manager and you specify some TEST run time options with the EQASET transaction. Do the steps described in “Scenario C: Running assembler program without Language Environment in IMS TM and managing TEST run time options with EQASET” on page 78 to enable this scenario.

   Scenario D
   You run programs in an IMSplex environment and are managing TEST run time options with either a user exit, CEEUOPT, or CEEROPT. Do the steps described in “Scenario D: Running IMSplex environment” on page 79 to enable this scenario.

   You can select more than one scenario. If you select more than one scenario, some steps are repeated. Perform those steps only once.
3. After you have selected the method that your site will use to manage TEST run time options, notify your application programmers of the chosen method. Ensure that the application programmers follow the directions described in "Preparing an IMS program" in the *Debug Tool User's Guide* and choose the

correct method for specifying TEST run time options. If your application programmers are using the EQASET transaction to specify TEST run time options, ensure that they follow the directions described in "Running the EQASET transaction" in the *Debug Tool User's Guide* .

Scenario A: Running IMS and managing TEST run time options with a user exit

Do the following steps to enable this scenario:

1. Include the Debug Tool *hlq*.SEQAMOD[16] data set and the Language Environment CEE.SCEERUN[17] run-time library in the STEPLIB concatenation of your IMS region.
2. To give IMS users enough time to run and debug their applications, increase the time-out limit in the message-processing region (MPR) region to 1440.

Scenario B: Running IMS and managing TEST run time options with CEEUOPT or CEEROPT

Do the following steps to enable this scenario:

1. Include the Debug Tool *hlq*.SEQAMOD[18] data set and the Language Environment CEE.SCEERUN run-time library in the STEPLIB concatenation of your IMS region.
2. To give IMS users enough time to run and debug their applications, increase the time-out limit in the message-processing region (MPR) region to 1440.

Scenario C: Running assembler program without Language Environment in IMS TM and managing TEST run time options with EQASET

Do the following steps to enable this scenario:

1. Copy the load modules EQANIAFE and EQANISET from the *hlq*.SEQAMOD data set into the IMS.PGMLIB data set.
2. Define the following IMS transaction:

```
APPLCTN GPSB=EQANISET,PGMTYPE=TP,LANG=ASSEM   HIDAM/OSAM
TRANSACT CODE=EQASET,MODE=SNGL,                                    X
         DCLWA=NO,EDIT=UC,INQ=(YES,NORECOV),                       X
MSGTYPE=(SNGLSEG,NONRESPONSE,1)
```

3. Add the application front end parameter APPLFE=EQANIAFE to the MPR start up job.
4. Assign the EQASET transaction to a class served by the MPR that is started with the APPLFE=EQANIAFE parameter.

---

16. Add *hlq*.SEQAMOD to STEPLIB only if it is not already in the system search path (for example, link list). If you create a custom EQAOPTS (as described in Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83) that is not stored in *hlq*.SEQAMOD, then place the data set containing it in STEPLIB (ahead of *hlq*.SEQAMOD if it is in STEPLIB).
17. Add CEE.SCEERUN to STEPLIB only if it is not already in the system search path (for example, link list). If you create a private copy of the Debug Tool Language Environment user exit for IMS that is linked into CEEBINIT (as described in Chapter 7, "Specifying the TEST runtime options through the Language Environment user exit," on page 33), then place the data set containing it in STEPLIB (ahead of CEE.SCEERUN if it is in STEPLIB).
18. Add *hlq*.SEQAMOD to STEPLIB only if it is not already in the system search path (for example, link list). If you create a custom EQAOPTS (as described in Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83) that is not stored in *hlq*.SEQAMOD, then place the data set containing it in STEPLIB (ahead of *hlq*.SEQAMOD if it is in STEPLIB).

5. Include the Debug Tool *hlq*.SEQAMOD[19] data set in the STEPLIB concatenation of your IMS region.
6. To give IMS users enough time to run and debug their applications, increase the time-out limit in the message-processing region (MPR) region to 1440.

Scenario D: Running IMSplex environment

Do the following steps to enable this scenario:

1. Include the Debug Tool *hlq*.SEQAMOD[20] data set and the Language Environment CEE.SCEERUN run-time library in the STEPLIB concatenation of your IMS region.
2. To give IMS users enough time to run and debug their applications, increase the time-out limit in the message-processing region (MPR) region to 1440.

19. Add *hlq*.SEQAMOD to STEPLIB only if it is not already in the system search path (for example, link list). If you create a custom EQAOPTS (as described in Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83) that is not stored in *hlq*.SEQAMOD, then place the data set containing it in STEPLIB (ahead of *hlq*.SEQAMOD if it is in STEPLIB).

20. Add *hlq*.SEQAMOD to STEPLIB only if it is not already in the system search path (for example, link list). If you create a custom EQAOPTS (as described in Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83) that is not stored in *hlq*.SEQAMOD, then place the data set containing it in STEPLIB (ahead of *hlq*.SEQAMOD if it is in STEPLIB).

Chapter 13. Enabling the EQAUEDAT user exit

The EQAUEDAT user exit enables the library administrator or system programmer to direct Debug Tool to the location where source, listing, or separate debug files are stored. If your site policy is to control the location of these files, this user exit supports this policy by allowing your application programmers to debug their programs without knowing where these files are located.

This sample is designed to operate only under the Language Environment. If you require an exit to run at any time in a non-Language Environment environment, you must replace the `CEEENTRY` and `CEETERM` macro invocations with the proper prologue and epilogue code for your environments. If Debug Tool detects a Language Environment-enabled `EQAUEDAT` when the Language Environment is not active, the exit will not be started.

To enable this user exit, do the following steps:

1. Copy the `EQAUEDAT`[21] member from the *hlq*`.SEQASAMP` library to a private library.
2. Edit the copy, as instructed in the member. Write the logic required to implement your site policy.

   The address of the load library data set name and the length of the load libray data set name cannot be provided as input to the EQAUEDAT user exit when the loading service (provider) that loaded the module is LPA, LLA, AOS loader, or an unknown provider because this information is not available when using these loading services.
3. Submit the JCL.
4. Add the private library where the generated `EQAUEDAT` load module is located to the load module search path for the application that you are debugging and for which you want this site policy enabled, in front of *hlq*`.SEQAMOD`.

21. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.

Chapter 14. Defining EQAOPTS options: checklist and instructions

This topic describes the options you can define in the EQAOPTS options file through the EQAXOPT macro and what Debug Tool features or behaviors they affect. To determine which options to define, review the checklist in Chapter 1, "Customizing Debug Tool: checklist," on page 1. As you encounter an item that describes an option you might want to use, record the options and values you want to use for that option in the following checklist:

__ EQAXOPT BROWSE, then select one of the following options:
  __ RACF
  __ ON
  __ OFF

__ EQAXOPT CACHENUM,*number:*_____

__ EQAXOPT CODEPAGE,*code_page_number:*_____

__ EQAXOPT DEFAULTVIEW, then select one of the following options:
  __ STANDARD
  __ NOMACGEN

__ EQAXOPT DTCNFORCECUID, then select one of the following options:
  __ YES
  __ NO

  This option performs the same function as DTCNFORCEPROGID. If you select YES for DTCNFORCEPROGID, you do not need to specify this option.

__ EQAXOPT DTCNFORCEIP, then select one of the following options:
  __ YES
  __ NO

__ EQAXOPT DTCNFORCELOADMODID, then select one of the following options:
  __ YES
  __ NO

__ EQAXOPT DTCNFORCENETNAME, then select one of the following options:
  __ YES
  __ NO

__ EQAXOPT DTCNFORCEPROGID, then select one of the following options:
  __ YES
  __ NO

__ EQAXOPT DTCNFORCETERMID, then select one of the following options:
  __ YES
  __ NO

__ EQAXOPT DTCNFORCETRANID, then select one of the following options:
  __ YES
  __ NO

__ EQAXOPT DTCNFORCEUSERID, then select one of the following options:
  __ YES
  __ NO

__ EQAXOPT EQAQPP, then select one of the following options:
   __ ON
   __ OFF
__ EQAXOPT GPFDSN,'*file_name:*_____'
__ EQAXOPT MDBG, then select one of the following options:
   __ YES
   __ NO
__ EQAXOPT NAMES, then select one of the following options:
   __ EXCLUDE,LOADMOD,*pattern:*_____
   __ EXCLUDE,CU,*pattern:*_____
   __ INCLUDE,LOADMOD,*name:*_____
   __ INCLUDE,CU,*name:*_____
__ EQAXOPT NODISPLAY, then select one of the following options:
   __ DEFAULT
   __ QUITDEBUG
__ EQAXOPT SAVEBPDSN,'*file_name:*_____'
__ EQAXOPT SAVESETDSN,'*file_name:*_____'
__ EQAXOPT SUBSYS,*subsystem library name:*_____
__ EQAXOPT SVCSCREEN, then select one of the following options:
   __ ON
   __ OFF

   Select one of the following options:
   __ CONFLICT=OVERRIDE
   __ CONFLICT=NOOVERIDE

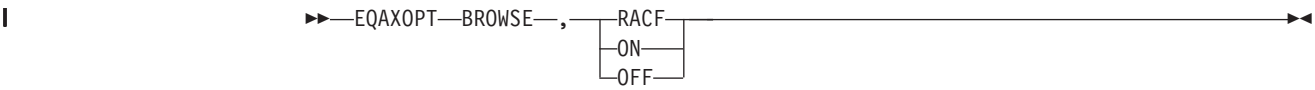
   If you want to do something about COPE, select one of the following options:
   __ NOMERGE
   __ MERGE=(COPE)
__ EQAXOPT THREADTERMCOND, then select one of the following options:
   __ PROMPT
   __ NOPROMPT
__ EQAXOPT TIMACB,*ACB-name:*_____

After you have made all of you selections, define the options as described in "Creating EQAOPTS load module" on page 97.

BROWSE

Debug Tool browse mode can be controlled by either the browse mode RACF facility, through the EQAOPTS options file, or both. For an overview of how to control browse mode, see "Debugging in browse mode" in *Debug Tool User's Guide*.

Users who have sufficient RACF authority to the applicable browse mode RACF facility (see Chapter 8, "Installing the browse mode RACF facility," on page 39) can control whether the current invocation of Debug Tool is to be in browse mode by an entry in EQAOPTS. The following diagram shows how to code this invocation of the EQAXOPT macro:



The following list describes the parameters of the EQAXOPT BROWSE macro:

RACF
Indicates that you want Debug Tool to use the browse mode access as determined by the current user's RACF access to the applicable RACF profile. If no EQAXOPT BROWSE statement is included in the EQAOPTS data set, a default of RACF is used.

ON
Indicates that unless the user's RACF access is NONE, set BROWSE MODE to ON.

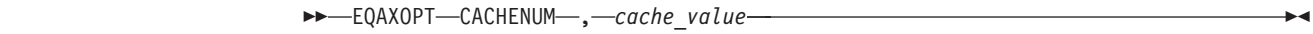
OFF
Indicates that if no RACF profile exists or if the user has UPDATE access or higher, set BROWSE MODE to OFF.

If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

CACHENUM

To reduce CPU consumption, Debug Tool uses a cache to store information about the application programs being debugged by a task. By default, for each debug session, Debug Tool stores the information for a maximum of 10 programs. Application programs that do a LINK or XCTL to more than 10 programs can degrade Debug Tool's CPU performance. You can enhance Debug Tool's CPU performance for these application programs by specifying an increased CACHENUM value in EQAOPTS. An increased value causes Debug Tool to use more storage for each debugged task.

The following diagram describes the syntax of this option:



CODEPAGE

The default code page used by Debug Tool and the remote debuggers is 037. For any of the following situations, you need to use a different code page:

- Application programmers are debugging in remote debug mode and the source or compiler use a code page other than 037.

  If your C/C++ source contains square brackets or other special characters, you might need to specify a CODEPAGE option to override the Debug Tool default code page (037). Check the code page specified when you compiled your source. The C/C++ compiler uses a default code page of 1047 if you do not explicitly specify one. If the code page used is 1047 or a code page other than 037, you need to specify a CODEPAGE option specifying that code page.
- Application programmers are debugging in full screen mode and encounter one of the following situations:
  - They use the STORAGE command to update COBOL NATIONAL variables.

  - The source is coded in a code page other than 037.
- Application programmers use the `XML(CODEPAGE(ccsid))` option on a `LIST CONTAINER` or `LIST STORAGE` command to specify an alternate code page.

Debug Tool uses the z/OS Unicode Services to process characters that need code page conversion.

The following diagram describes the syntax of the EQAXOPT CODEPAGE option:

If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

After implementing the EQAXOPT CODEPAGE option, if your application programmers using full-screen mode still cannot display some characters correctly, have them verify that their emulator's code page matches the code page of the characters they need to display.

You might need to create your own conversion images as described in "Creating a conversion image for Debug Tool."

Creating a conversion image for Debug Tool

You might need to create a conversion image so that Debug Tool can properly transmit characters in a code page other than 037 between the remote debugger and the host. A conversion image contains the following information:

- The conversion table that specifies the source CCSID (Coded Character Set Identifiers) and target CCSID. For Debug Tool, specify a pair of conversion images between the host code page and Unicode code page (UTF-8). The host code page is specified in the `VADSCP`*`nnnnn`* suboption of `TEST` run-time option or in the CODEPAGE option in the EQAOPTS file. If both the `VADSCP`*`nnnnn`* suboption and the CODEPAGE option are specified, only the CODEPAGE option is used. The following table shows the images required for CCSIDs 930, 939 (Japanese EBCDIC), 933 (Korean EBCDIC), 1141 (Germany EBCDIC), and 1047 (Latin 1/Open Systems, EBCDIC). See *Debug Tool Reference and Messages* for a detailed description of the suboption `VADSCP`*`nnnnn`*.

Table 10. Source and target CCSID to specify, depending on the code page option used

`VADSCP`*`nnnn`* suboption or CODEPAGE option	**Source CCSID**	**Target CCSID**
VADSCP930 or CODEPAGE,930	1390[1]	1208 (UTF-8)
	1208	1390[1]
VADSCP939 or CODEPAGE,939	1399[1]	1208 (UTF-8)
	1208	1399[1]
VADSCP933 or CODEPAGE,933	933	1208 (UTF-8)
	1208	933
VADSCP1141 or CODEPAGE,1141	1141	1208 (UTF-8)
	1208	1141

Table 10. Source and target CCSID to specify, depending on the code page option used (continued)

VADSCP*nnnn* suboption or CODEPAGE option	**Source CCSID**	**Target CCSID**
VADSCP1047 or CODEPAGE,1047	1047	1208 (UTF-8)
	1208	1047
Notes: 1. For compatibility with earlier versions, 1390 and 1399 are used.		

For each suboption, a pair of conversion images are needed for bidirectional conversion.

- The conversion technique, also called the technique search order. Debug Tool uses the technique search order RECLM, which means roundtrip, enforced subset, customized, Language Environment-behavior, and modified language. RECLM is the default technique search order, so you do not have to specify the technique search order in the JCL.

You might need to create a conversion image so that users debugging COBOL programs in full screen or batch mode can modify NATIONAL variables with the `STORAGE` command or to properly display C/C++ variables that contain characters in a code page other than 037. To create the conversion image, you need to do the following steps:

1. Ask your system programmer for the host's CCSID.
2. Submit a JCL job that specifies the conversion image between the host CCSID, which you obtained in step 1, and CCSID 1200 (UTF-16).

"Example: JCL for generating conversion images" describes how one JCL creates the conversion images for both situations.

Example: JCL for generating conversion images

The following JCL generates the conversions images required for Debug Tool.

This JCL is a variation of the JCL located at *hlq*`.SCUNJCL(CUNJIUTL)`, which is provided by the Unicode conversion services package.

```
//CUNMIUTL EXEC PGM=CUNMIUTL
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIMG DD DSN=UNI.IMAGES(CUNIMG01),DISP=SHR
//TABIN DD DSN=UNI.SCUNTBL,DISP=SHR
//SYSIN DD *
  /*********************************************************************/
  /* Conversion image input for Debug Tool in Remote                   */
  /* debug mode                                                        */
  /*********************************************************************/
  CONVERSION 1390,1208;   /* IBM-930 to UTF-8,RECLM */
  CONVERSION 1208,1390;   /* UTF-8 to IBM-930,RECLM */
  CONVERSION 1399,1208;   /* IBM-939 to UTF-8,RECLM */
  CONVERSION 1208,1399;   /* UTF-8 to IBM-939,RECLM */
  CONVERSION  933,1208;   /* IBM-933 to UTF-8,RECLM */
  CONVERSION 1208,933;    /* UTF-8 to IBM-933,RECLM */
  CONVERSION 1141,1208;   /* IBM-1141 to UTF-8,RECLM */
  CONVERSION 1208,1141;   /* UTF-8 to IBM-1141,RECLM */
  CONVERSION 1047,1208;   /* IBM-1047 to UTF-8,RECLM */
  CONVERSION 1208,1141;   /* UTF-8 to IBM-1141,RECLM */
  /*********************************************************************/
  /* Conversion image input for Debug Tool to modify COBOL NATIONAL   */
```

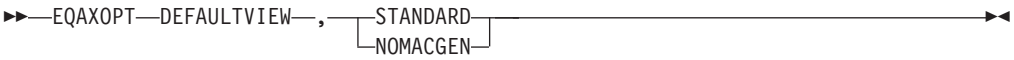
```
  /* variables with the STORAGE command while in full screen mode     */
  /*********************************************************************/
  CONVERSION 0037,1200;   /*IBM-37 to UTF-16,RECLM */
/*
```

Debug Tool uses the character conversion services but not the case conversion or the normalization services of Unicode conversion services. You do not need to include `CASE` or `NORMALIZE` control statements unless other applications require them.

DEFAULTVIEW

A user can control whether to display the statements of an assembler macro in the Source window by entering the `SET DEFAULT VIEW` command. Every time a `LOADDEBUGDATA` command is run for an assembler compile unit, Debug Tool uses the setting of this command to determine whether to display the macro's statements. You can make the setting of this command applicable to all debugging sessions by setting the DEFAULTVIEW operand of the EQAXOPT macro.

The following diagram describes the syntax of the EQAXOPT:



Each of these fields corresponds to the similar field in the SET DEFAULT VIEW command. If EQAXOPT DEFAULTVIEW is not coded, the initial setting for DEFAULTVIEW is STANDARD.

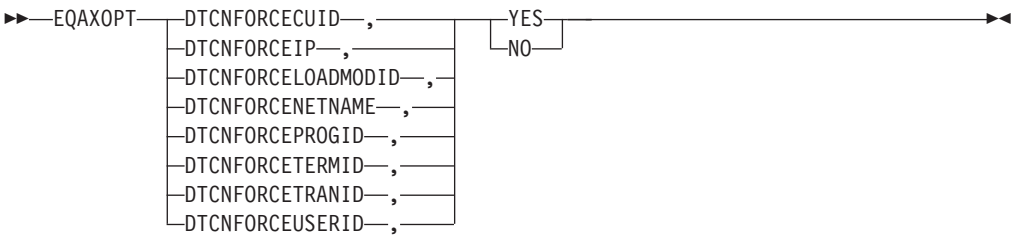
If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, “Defining EQAOPTS options: checklist and instructions,” on page 83.

DTCNFORCE*xxxx*

If your users use DTCN to specify debugging profiles, the following list describes each resource type you can require your users to specify:

- DTCNFORCECUID or DTCNFORCEPROGID, which requires the user to specify the name of a compile unit or compile units.
- DTCNFORCEIP, which requires the user to specify the IP name or address.
- DTCNFORCELOADMODID, which requires the user to specify the name of a load module or load modules.
- DTCNFORCENETNAME, which requires the user to specify the four character name of a CICS terminal or a CICS system.
- DTCNFORCETERMID, which requires the user to specify the CICS terminal.
- DTCNFORCETRANID, which requires the user to specify a transaction ID.
- DTCNFORCEUSERID, which requires the user to specify a user ID.

If any of the statements are not included, the statement defaults to NO. The following diagram describes the syntax of this option:

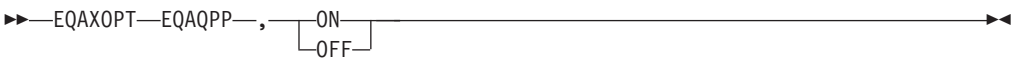


If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

EQAQPP

This topic describes one of the tasks you need to do to enable Debug Tool to debug MasterCraft Q++ programs, provided by Tata Consultancy Services Ltd. For more information about how to enable Debug Tool to support MasterCraft Q++, contact Tata Consultancy Services Ltd.

If the statement is not included, the statement defaults to OFF. The following diagram describes the syntax of the EQAXOPT EQAQPP option:



If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

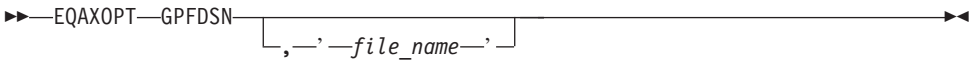
GPFDSN

You can create a *global preferences file* that runs a set of Debug Tool commands at the start of all Debug Tool sessions. For example, a global preferences file can have a command that sets PF keys to specific values. If your site uses the PF6 key as the program exit key, you can specify the `SET PF6 "EXIT" = QUIT;` command, which assigns the Debug Tool `QUIT` command to the PF6 key, in the global preferences file. (See "Customizing your full-screen session" in *Debug Tool User's Guide* for a description of the interface features you can change.) To create a global preferences file, do the following steps:

1. Create a preferences file that is stored as a sequential file or a PDS member. Refer to *Debug Tool User's Guide* for a description of preferences files.

   The rules for the preferences file are dependant on the language of the first program Debug Tool encounters. Because you might not know what language Debug Tool will encounter first, we recommend you use the following rules when you create the preferences file:

   - Put the commands in columns 8 - 72.
   - Do not put line numbers in the file.
   - Use `COMMENT` or `/* */` to delimit comments.
2. Specify the GPFDSN option in the EQAOPTS option file. The following diagram describes the GPFDSN option:



For *file_name*, specify the name of the data set where the global preferences file will be stored.

Whenever a user starts Debug Tool, the commands in the global preferences file are run first. The user can also create his or her own preferences file and a commands file. In this situation, Debug Tool processes the files in the following order:

1. Global preferences file
2. User preferences file
3. Commands file

If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

MDBG

If you are using z/OS XLC C/C++, Version 1 Release 11, you can indicate that Debug Tool always searches for .mdbg files to retrieve the source and debug information by setting the EQAXOPTS option MDBG to YES. When you set MDBG to YES, Debug Tool retrieves the debug information from an .mdbg file and does not try to find the debug information from the following sources, even if they exist:

- a .dbg file
- if the program was compiled with the ISD compiler option, the object

If you do not specify MDBG or set it to NO, Debug Tool retrieves the debug information from either the .dbg file or, if the program was compiled with the ISD compiler option, the object.

The following diagram describes the MDBG option:

If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

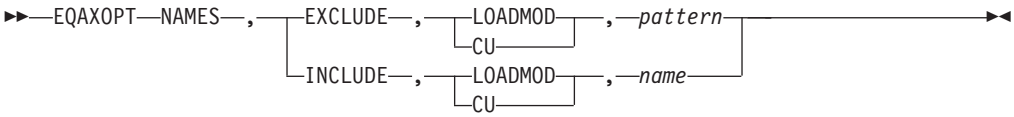
NAMES

The *Debug Tool User's Guide* describes how the NAMES command can be used to perform several specific functions dealing with load module and compile unit names recognized by Debug Tool. However, the NAMES command cannot be used to alter the behavior of load module or compile unit names that have already been seen by Debug Tool at the time the NAMES command is processed.

If it becomes necessary to perform these functions on the initial load module processed by Debug Tool or on any of the compile units contained in that load

module, you must provide the information (that would otherwise have been specified using the NAMES command) in the EQAOPTS Debug Tool customization module.

One or more invocations of the EQAXOPT macro with the NAMES operand can be used for this purpose. The syntax of this macro is shown in the following diagram:



Each of these fields corresponds to the similar field in the NAMES command.

If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

NODISPLAY

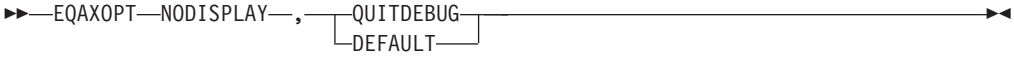
In the following two situations, in which a user can request a specific user interface, that interface may not be available:

- full-screen mode using a dedicated terminal (with or without the Terminal Interface Monitor). If the terminal is not available, the program being debugged terminates with a U4038 abend.
- remote debugger. If the remote debugger is not available, Debug Tool will use full-screen mode if the user is running under TSO. If the user is not using TSO, Debug Tool will use batch mode.

In both cases, Write To Operator (WTO) messages also appear.

You can modify these behaviors by specifying the EQAXOPT NODISPLAY option so that Debug Tool continues processing as if the user immediately entered a `QUIT DEBUG` command. This modification prevents any forced abend or the debugger from starting, which is often preferable.

The following invocation of the EQAXOPT macro with the `NODISPLAY` operand can be used for this purpose. The syntax of this macro is shown in the following diagram:



DEFAULT
Debug Tool follows the default behavior.

QUITDEBUG
Debug Tool displays a message that indicates that Debug Tool will quit, and that the user interface could not be used. Debug Tool processing continues as if the user entered a `QUIT DEBUG` command.

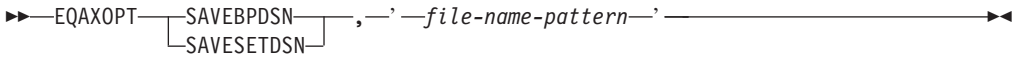
If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

SAVEBPDSN, SAVESETDSN

You can modify the default names of the data sets used to save and restore settings and breakpoints, monitor values, and LOADDEBUGDATA (LDD) specifications. The following list describes the initial default names:

- For settings: *userid*.DBGTOOL.SAVESETS
- For breakpoints, monitor values, and LOADDEBUGDATA (LDD) specifications: *userid*.DBGTOOL.SAVEBPS

To change the default name for either or both of these data sets, you need to specify the SAVESETDSN and SAVEBPDSN option in the EQAOPTS option file, along with a corresponding naming pattern for the data set. The following diagram describes the SAVESETDSN and SAVEBPDSN options:



For *file-name-pattern*, specify a naming pattern for the data set that stores this information.

In most environments, you should choose one of the following rules for the naming pattern:

- Any data set name that includes &&USERID. as one of the qualifiers. Debug Tool substitutes the user ID of the current user for this qualifier when it creates the data set.
- A DD name (Reminder: DD names are not supported under CICS)
- The string NULLFILE to indicate that saving and restoring this information is not supported

If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

SUBSYS

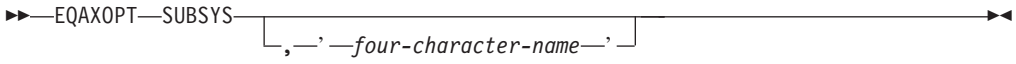
This topic describes when and how to specify the SUBSYS allocation parameter in the EQAOPTS option file.

If the following conditions apply at your site, you need specify the SUBSYS=*library_subsystem_name* allocation parameter in the EQAOPTS option file:

- The source code is managed by a library system that requires that you specify the SUBSYS=*library_subsystem_name* allocation parameter when you allocate a data set.
- Your users are debugging C, C++, or Enterprise PL/I programs compiled without the SEPARATE suboption of the TEST compiler option.

You must run Debug Tool and the specified subsystem on the same system. You cannot use this feature to debug programs that run under CICS.

The following diagram describes the EQAXOPT SUBSYS option:



If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

SVCSCREEN

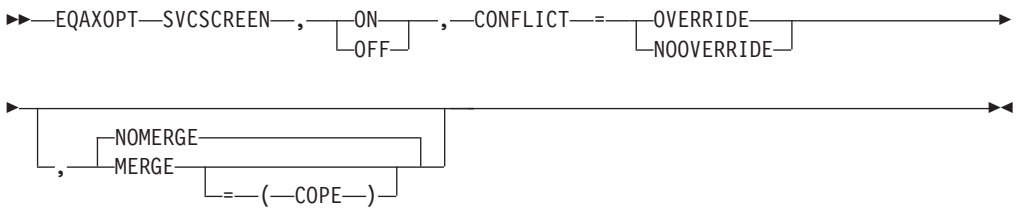
In a non-CICS environment, Debug Tool requires SVC screening for the following situations:

- Invoking Debug Tool by using EQANMDBG to debug programs that start outside Language Environment including non-Language Environment COBOL programs.
- Debugging programs that do not run in Language Environment and are started by programs that begin in Language Environment.
- Detecting services such as MVS LINK, LOAD and DELETE.

If you need to run Debug Tool in any of the following situations, you must specify the actions that Debug Tool must take regarding SVC screening:

- Start Debug Tool by using EQANMDBG in an environment that already uses SVC screening.
- Run Debug Tool when debugging programs that do not run in Language Environment and are started by programs that begin in Language Environment.
- Run Debug Tool when you need to detect services such as MVS LINK, LOAD and DELETE.
- Run Debug Tool in a situation that requires SVC screening and SVC screening is already in use by a program with which Debug Tool supports MERGE SVC screening as described by the MERGE operand that follows.

The following diagram shows how to code an invocation of the EQAXOPT macro:



The following list describes the parameters of the EQAXOPT SVCSCREEN macro:

ON
Indicates that you want Debug Tool to use SVC screening in order to support MVS LOAD, DELETE, and LINK SVCs.

OFF
Indicates that you want Debug Tool to not use SVC screening. Debug Tool will not know about programs started through MVS LOAD, DELETE, and LINK SVCs.

CONFLICT=
Specifies what you want Debug Tool to do when ON is specified or defaulted and SVC screening is already used by another program.

OVERRIDE
Indicates that you want Debug Tool to override the current SVC screening and take control of SVC screening.

NOOVERRIDE
Indicates that if SVC screening is already in use, Debug Tool does not initiate SVC screening and proceeds as if OFF were specified.

NOMERGE
Indicates that SVC screening is not to be merged with SVC screening used by any other product. `NOMERGE` is the default.

MERGE
Indicates that when SVC screening is already being used by another program when Debug Tool starts, Debug Tool saves the current SVC screening environment, then enables SVC screening for both Debug Tool and the other program. When Debug Tool terminates, it restores the original SVC screening environment.

Currently, Debug Tool supports the `MERGE` option with only one other program: COPE.

If you specify the `MERGE` option and Debug Tool does not recognize the program that is using the SVC screening, the `MERGE` option is ignored and Debug Tool starts based on the value of the `CONFLICT` option.

MERGE=(COPE)
If COPE is active, Debug Tool saves the current SVC screening environment, then enables SVC screening for both Debug Tool and COPE. When Debug Tool terminates, it restores COPE's SVC screening environment.

If COPE is not active, Debug Tool starts based on the value of the `CONFLICT` option.

The default parameters for the EQAXOPT SVCSCREEN macro is one of the following situations:

- If Debug Tool is started by using the EQANMDBG program:
  `SVCSCREEN,ON,CONFLICT=NOOVERRIDE,NOMERGE`
- If Debug Tool is started by any other method:
  `SVCSCREEN,OFF,CONFLICT=NOOVERRIDE,NOMERGE`

If Debug Tool is started by using the EQANMDBG program, the `OFF` setting is ignored.

After you review the syntax, use Table 11 on page 95 as a guide to select the appropriate suboptions.

If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

Example: Combinations of EQAXOPT SVCSCREEN suboptions

The following table shows examples of combinations of EQAXOPT SVCSCREEN suboptions:

Table 11. Combination of SVSCREEN options and their effects

SVCSCREEN options	Type of Debug Tool session	Action
`OFF,CONFLICT=NOOVERRIDE` (default)	Debug Tool started by using EQANMDBG	Same as for `ON,CONFLICT=NOOVERRIDE`.
	Debug Tool started by any other method	• Debug Tool does not enable its SVC screening. • You cannot debug programs that do not run in Language Environment which were started by programs that do run in Language Environment. • Debug Tool does not detect the MVS services LINK, LOAD and DELETE. • The CONFLICT setting is ignored when the OFF setting is specified.
`OFF,CONFLICT=OVERRIDE`	Debug Tool started by using EQANMDBG	Same as for `ON,CONFLICT=OVERRIDE`.
	Debug Tool started by any other method	Same as for `OFF,CONFLICT=NOOVERRIDE`. The CONFLICT setting is ignored when the OFF setting is specified.
`ON,CONFLICT=NOOVERRIDE`	Debug Tool started by using EQANMDBG	If SVC screening is active, Debug Tool terminates. If SVC screening is not active, Debug Tool enables its SVC screening, runs the debugging session, and disables its SVC screening after the debugging session ends.
	Debug Tool started by any other method	If SVC screening is active, Debug Tool does not enable its SVC screening. You cannot debug programs that do not run in Language Environment which were started by programs that do run in Language Environment. Debug Tool does not detect the MVS services LINK, LOAD and DELETE. If SVC screening is not active, Debug Tool enables its SVC screening, runs the debugging session, and disables its SVC screening after the debugging session ends.

Table 11. Combination of SVSCREEN options and their effects (continued)

SVCSCREEN options	Type of Debug Tool session	Action
ON,CONFLICT=OVERRIDE	Debug Tool started by using EQANMDBG	If any SVC screening is active and the NOMERGE option is in effect, Debug Tool overrides the existing SVC screening. This is also the default behavior. Debug Tool enables its SVC screening, runs the debugging session, and disables its SVC screening after the debugging session ends. If any SVC screening was active, Debug Tool restores the previous SVC screening. If you specify the MERGE option, see the following information about MERGE.
	Debug Tool started by any other method	

Each user or group can control this behavior by creating their own copy of EQAOPTS with their desired options and placing it in the load module search path before *hlq*.SEQAMOD.

THREADTERMCOND

You can indicate that Debug Tool should not prompt the user when a FINISH, CEE066, or CEE067 thread termination condition is raised by Language Environment, regardless of the suboptions used in the TEST runtime option. These conditions are raised by statements like STOP RUN, GOBACK, or EXEC CICS RETURN, which can occur frequently in an application program. Suppressing the display of these prompts can reduce the number of times your users are interrupted by this prompt during a debugging session.

If the statement is not included, the statement defaults to PROMPT. The following diagram describes the syntax of the THREADTERMCOND option:

If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, “Defining EQAOPTS options: checklist and instructions,” on page 83.

TIMACB

TIMACB identifies the name of an ACB, other than EQASESSM, that Debug Tool uses to make full-screen mode using a dedicated terminal with Terminal Interface Manager work in an environment where you want to run the Terminal Interface Manager on more than one LPAR in the same VTAM network. You specify TIMACB as the last step in “Running the Terminal Interface Manager on more than one LPAR on the same VTAM network” on page 30.

The following diagram describes the syntax of this option:

ACB-name is the new ACB name you created in step 1 on page 30.

Place this customized EQAOPTS module in the load module search path in front of *hlq*.SEQAMOD for the Debug Tool users who are using this new instance of the Terminal Interface Manager.

If you choose to implement this option, remember to record your selection on the form at the beginning of Chapter 14, "Defining EQAOPTS options: checklist and instructions," on page 83.

Creating EQAOPTS load module

After you have chosen EQAOPTS options, do the following steps:

1. Copy the EQAOPTS[22] member from the *hlq*.SEQASAMP library to a private library.
2. Edit this copy of EQAOPTS and code the EQAOPTS option or options you want. The following diagram describes the minimum assembler source required to generate the EQAOPTS load module:

```
EQAOPTS  CSECT ,
EQAOPTS  AMODE 31
EQAOPTS  RMODE ANY
         add your customized EQAXOPT statements here
         EQAXOPT  END
         END ,
```

   To this minimum source you add each EQAXOPT option you selected on the list on page 83.
3. Follow the directions in sample EQAOPTS data set to generate a new EQAOPTS load module. These directions describe how to assemble the source and link-edit the generated object into a load module named EQAOPTS.
4. Place the EQAOPTS load module in a private data set that is in the load module search path and appears before *hlq*.SEQAMOD.

22. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.

Chapter 15. Using EQACUIDF to specify values for NATLANG, LOCALE, and LINECOUNT

The EQACUIDF member of *hlq*.SEQABMOD contains the default and allowable values for the parameters NATLANG, LOCALE, and LINECOUNT. These values are used by the following Debug Tool components:

- Debug Tool Utilities ISPF dialogs: NATLANG
- EQANMDBG (non-CICS non-Language Environment support): NATLANG
- Debug Tool Coverage Utility: NATLANG, LOCALE, and LINECOUNT

This topic describes the allowable values for these parameters, how to change the default values, and how to enable additional languages for some Debug Tool components.

Changing the default and allowable values in EQACUIDF

The default and allowable values for NATLANG, LOCALE, and LINECOUNT are as follows:

- NATLANG. The national language, which can be one of the following:
  - Mixed-case English (ENU)
  - Uppercase English (UEN)
  - Japanese (JPN)
  - Korean (KOR)

  See "Enabling additional languages for some Debug Tool components through EQACUIDF" on page 100 for more information about changing the language for these Debug Tool components.
- LOCALE. The format of date, time, and numeric values. You can also create date, time, and numeric formats. The default values are as follows:
  - Date format: MM/DD/YYYY
  - Time format: HH:MM:SS
  - Numeric format: 1,234,567.89
- LINECOUNT. The number of lines (including headings) that print on a page. The default is 66 lines.

If the default values for these parameters are the values that you want to use, you can skip this section.

To change the default values:

1. Copy the EQACUIDF[23] member in the *hlq*.SEQASAMP data set into another data set.
2. Follow the instructions that are in the comment sections of the code to modify the copy that you made.
3. Assemble the modified copy by using the IBM High Level Assembler and specifying *hlq*.SEQASAMP as a SYSLIB.
4. Link edit the resulting object into the *private*.SEQABMOD data set.

23. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.

5. Copy the output load module to *hlq*.SEQABMOD.

Sample JCL is provided in the EQACUIID member of the *hlq*.SEQASAMP data set to perform steps 3 and 4.

The SEQABMOD from this version of Debug Tool is compatible with earlier versions of Debug Tool. If you have multiple versions of Debug Tool installed on your system, you need only the SEQABMOD from this version installed in your system link list concatenation.

Enabling additional languages for some Debug Tool components through EQACUIDF

If you use these components, and have installed either of the additional language features (Japanese or Korean), you must do the following steps to enable the user to specify the additional language feature with the NATLANG parameter.

To change the language to Japanese or Korean:

1. Create a private SEQASAMP data set like *hlq*.SEQASAMP.
2. Create a private SEQABMOD data set like *hlq*.SEQABMOD.
3. Copy members EQACUIDF[24], EQACUIDM[25], and EQACUIID from *hlq*.SEQASAMP to your private SEQASAMP. Any edits that are described in this section are to be done in the private SEQASAMP copies of these members.
4. Edit the EQACUIDM member and add each additional installed language feature to the line starting with &ValLang(1), using JPN for Japanese, and KOR for Korean. For example, adding Japanese would be done as follows:

   ```
   &ValLang(1) SetC 'ENU','UEN','JPN' Set valid languages
   ```

5. Edit the EQACUIDF member and add each additional installed language feature after the following line:

   ```
   UEN Language UEN
   ```

   For example:

   ```
   UEN Language UEN
   JPN Language JPN
   ```

6. If you want to change the default value for NATLANG, edit the EQACUIDF member and change the DfltLang value. For example, making JPN the default for NATLANG would be as follows:

   ```
   EQACUIDF InstDflt DfltLang=JPN,                                    +
   ```

7. Assemble and link a new copy of EQACUIDF into the private SEQABMOD by editing and submitting the JCL that is supplied in member EQACUIID.
8. Copy the EQACUIDF member from the private SEQABMOD into *hlq*.SEQABMOD.

For more information, see "Changing the default and allowable values in EQACUIDF" on page 99.

24. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.
25. See Appendix A, "SMP/E USERMODs," on page 101 for an SMP/E USERMOD for this customization.

Appendix A. SMP/E USERMODs

SMP/E USERMODs are available for a number of the customizations listed in the *Debug Tool Customization Guide* and *Debug Tool User's Guide*. The following table shows the available USERMODs and the associated names.

hlq.**SEQAEXEC**	*hlq*.**SEQATLIB**	*hlq*.**SEQASAMP**	*hlq*.**SEQAMOD**	**SMP/E USERMOD in** *hlq*.**SEQASAMP**
EQACUDFT				EQAUMOD1
EQASTART				EQAUMOD2
	EQALMPFX			EQAUMOD3
	EQALMPGM			EQAUMOD4
	EQAZDFLT			EQAUMOD5
	EQAZDSYS			EQAUMOD6
	EQAZDUSR			EQAUMOD7
	EQAZPROC			EQAUMOD8
		EQACUIDF[1]	EQACUIDF[2]	EQAUMOD9
		EQACUIDM		EQAUMODA
		EQADBCXT[1,3]	EQADBCXT[2]	EQAUMODB
		EQADDCXT[1,3]	EQADDCXT[2]	EQAUMODC
		EQADICXT[1,3]	EQADICXT[2]	EQAUMODD
		EQAOPTS[1]	EQAOPTS[2]	EQAUMODE
		EQAUEDAT[1]	EQAUEDAT[2]	EQAUMODF

Notes:

1. The source for these parts is in *hlq*.SEQASAMP. The executable (the part updated by the USERMOD) is in SEQAMOD.
2. The *Debug Tool User's Guide* and *Debug Tool Customization Guide* discussion of these parts typically shows generating a private copy of these load modules. If you want to update *hlq*.SEQAMOD so that all users see these customizations, you should use the SMP/E USERMOD method.
3. Debug Tool SMP/E USERMODs for these parts are only available if you choose the method that updates *hlq*.SEQAMOD. They are not available if you choose to update CEE.SCEERUN.

Appendix B. Applying maintenance

Appendix C, "Support resources and problem solving information," on page 105 describes all the resources available to obtain technical support information. Follow the steps in this section to apply a service APAR or PTF.

Applying Service APAR or PTF

This chapter describes how to apply service updates to Debug Tool. To use the maintenance procedures effectively, you must install the product or products by using SMP/E before doing the maintenance procedures below.

What you receive

If you report a problem with Debug Tool to your IBM Support Center, you may receive a tape containing one or more Authorized Program Analysis Reports (APARs) or Program Temporary Fixes (PTFs) that were created to solve your problem.

You may also receive a list of prerequisite APARs or PTFs, which you must apply to your system before applying the current APAR. These prerequisite APARs or PTFs might relate to Debug Tool or any other licensed product you have installed, including z/OS.

Checklist for applying an APAR or PTF

The following checklist describes the steps and associated SMP/E commands to install the APAR or PTF:

__ Step 1. Prepare to install the APAR or PTF.

__ Step 2. Receive the APAR or PTF. (SMP/E RECEIVE)

__ Step 3. Review the HOLDDATA.

__ Step 4. Accept previously applied APARs or PTFs (optional). (SMP/E ACCEPT)

__ Step 5. Apply APAR or PTF. (SMP/E APPLY)

__ Step 6. Run REPORT CROSSZONE and apply any missing requisites.

__ Step 7. Test APAR or PTF.

__ Step 8. Accept APAR or PTF. (SMP/E ACCEPT)

Step 1. Prepare to install APAR or PTF

Before you start to install an APAR or PTF, do the following:

1. Create a backup copy of the current Debug Tool libraries. Save this copy of Debug Tool until you have completed installing the APAR or PTF, and you are confident that the service runs correctly.
2. Research each service tape through the IBM Support Center for any errors or additional information. Note all errors on the tape that were reported by APARs or PTFs and apply the relevant fixes. You should also review the current Preventive Service Planning (PSP) information.

Step 2. Receive the APAR or PTF

Receive the service using the SMP/E RECEIVE command from either the SMP/E dialogs in ISPF, or using a batch job similar to EQAWRECV in *hlq*.SEQASAMP.

Step 3. Review the HOLDDATA

Review the HOLDDATA summary reports for the APAR or PTF. Follow any instructions described in the summary reports.

Step 4. Accept previously applied APAR or PTF (optional)

If there is any APAR or PTF which you applied earlier but did not accept, and the earlier APAR or PTF is not causing problems in your installation, accept the applied service from either the SMP/E dialogs in ISPF, or using a batch job similar to EQAWACPT in *hlq*.SEQASAMP.

Accepting the earlier service allows you to use the SMP/E RESTORE command to return to your current level if you encounter a problem with the service you are currently applying. You can do this either from the SMP/E dialogs in ISPF, or using a batch job.

Step 5. Apply the APAR or PTF

We recommend you first use the SMP/E APPLY command with the CHECK operand. Check the output; if it shows no conflict, rerun the APPLY command without the CHECK operand. This can be done from the SMP/E dialogs in ISPF or using a batch job similar to EQAWAPLY in *hlq*.SEQASAMP.

Step 6. Run REPORT CROSSZONE and apply any missing requisites

Run an SMP/E REPORT CROSSZONE by using the SMP/E dialogs or by using a batch job similar to EQAWRPXZ in *hlq*.SEQASAMP. Apply any missing requisites found by SMP/E.

Step 7. Test the APAR or PTF

Thoroughly test your updated Debug Tool. Do not accept an APAR or PTF until you are confident that it runs correctly.

Step 8. Accept the APAR or PTF

We recommend you first use the SMP/E ACCEPT command with the CHECK operand. Check the output; if it shows no conflict, rerun the ACCEPT command without the CHECK operand. You can do this either from the SMP/E dialogs in ISPF, or using a batch job similar to EQAWACPT in *hlq*.SEQASAMP.

Appendix C. Support resources and problem solving information

This section shows you how to quickly locate information to help answer your questions and solve your problems. If you have to call IBM support, this section provides information that you need to provide to the IBM service representative to help diagnose and resolve the problem.

For a comprehensive multimedia overview of IBM software support resources, see the IBM Education Assistant presentation "IBM Software Support Resources for System z® Enterprise Development Tools and Compilers products" at http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp?topic=/com.ibm.iea.debugt/debugt/6.1z/TrainingEducation/SupportInfoADTools/player.html.

Searching IBM support Web sites for a solution

You can search the available knowledge bases to determine whether your problem was already encountered and is already documented.

Searching the information center

You can find this publication and documentation for many other products in the IBM System z Enterprise Development Tools & Compilers information center at http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp. Using the information center, you can search product documentation in a variety of ways. You can search across the documentation for multiple products, search across a subset of the product documentation that you specify, or search a specific set of topics that you specify within a document. Search terms can include exact words or phrases, wild cards, and Boolean operators.

To learn more about how to use the search facility provided in the IBM System z Enterprise Development Tools & Compilers information center, you can view the multimedia presentation at http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp?topic=/com.ibm.help.doc/InfoCenterTour800600.htm.

Searching product support documents

Use the System z Enterprise Development Tools & Compilers information center or the product support page to search the Internet for the latest, most complete information that might help you resolve your problem.

Specific IBM Software Support sites for the System z Enterprise Development Tools and Compilers products include:

- Application Performance Analyzer for z/OS Support
- Debug Tool for z/OS Support
- Enterprise COBOL for z/OS Support
- Enterprise PL/I for z/OS Support
- Fault Analyzer for z/OS Support
- File Export for z/OS Support
- File Manager for z/OS Support
- Optim™ Move for DB2 Support
- WebSphere Developer Debugger for System z Support
- WebSphere Studio Asset Analyzer for Multiplatforms Support
- Workload Simulator for z/OS and OS/390 Support

To search multiple Internet resources for your product using the information center, click **Troubleshooting and support** in the left navigation pane and select **Searching IBM support Web sites for a solution**. You can select one or more products, specify keywords, and search a variety of resources, including the following:

- IBM technotes
- IBM downloads and fixes
- IBM problem reports (APARs) and flashes
- IBM Redbooks®, whitepapers, articles, and tutorials
- IBM developerWorks®
- Forums and newsgroups
- Google

There is also a search facility provided on the product support page. The search facility provided on the product support page allows you to narrow the search scope and search only product support documents for that product.

IBM Support Assistant

The IBM Support Assistant (also referred to as ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. It provides quick access to support-related information. You can use the IBM Support Assistant to help you in the following ways:

- Search through IBM and non-IBM knowledge and information sources across multiple IBM products to answer a question or solve a problem.
- Find additional information through product and support pages, customer news groups and forums, skills and training resources and information about troubleshooting and commonly asked questions.

In addition, you can use the built in Updater facility in IBM Support Assistant to obtain IBM Support Assistant upgrades and new features to add support for additional software products and capabilities as they become available.

For more information, and to download and start using the IBM Support Assistant for IBM System z Enterprise Development Tools & Compilers products, please visit http://www.ibm.com/support/docview.wss?rs=2300&context=SSFMHB&dc=D600&uid=swg21242707&loc=en_US&cs=UTF-8&lang=en.

General information about the IBM Support Assistant can be found on the IBM Support Assistant home page at http://www.ibm.com/software/support/isa.

Obtaining fixes

A product fix might be available to resolve your problem. To determine what fixes and other updates are available, the following information is available on the respective product support site:

- Latest PTFs for Application Performance Analyzer for z/OS
- Latest PTFs for Debug Tool for z/OS
- Latest PTFs for Fault Analyzer for z/OS
- Latest PTFs for File Export for z/OS
- Latest PTFs for File Manager for z/OS
- Latest fixes for Optim Move for DB2
- Latest PTFs for WebSphere Studio Asset Analyzer for Multiplatforms
- Latest PTFs for Workload Simulator for z/OS and OS/390

When you find a fix that you are interested in, click the name of the fix to read its description and to optionally download the fix.

For more information about the types of fixes that are available, see the *IBM Software Support Handbook* at http://techsupport.services.ibm.com/guides/handbook.html.

Receiving support updates through e-mail notification

To receive e-mail notifications about fixes and other software support news, follow the steps below. Additional information is provided at http://www.ibm.com/support/docview.wss?rs=615&uid;=swg21172598.

1. Go to the IBM Software Support Web site at http://www.ibm.com/software/support.
2. Click **My notifications** in the upper right corner of the page.
3. If you have already registered for **My notifications**, sign in and skip to the next step. If you have not registered, click **register now**. Complete the registration form using your e-mail address as your IBM ID and click **Submit**.
4. In the **My notifications** tool, click the **Subscribe** tab to specify products for which you want to receive e-mail updates.
5. To specify Problem Determination Tools products, click **Other software** and then select the products for which you want to receive e-mail updates, for example, **Debug Tool for z/OS** and **File Manager for z/OS**.
6. To specify a COBOL or PL/I compiler, click **Rational**® and then select the products for which you want to receive e-mail updates, for example, **Enterprise COBOL for z/OS** .
7. After selecting all products that are of interest to you, scroll to the bottom of the list and click **Continue**.
8. Determine how you want to save your subscription. You can use the default subscription name or create your own by entering a new name in the **Name** field. It is recommended that you create your own unique subscription name using a something easily recognized by you. You can create a new folder by entering a folder name in the **New** field or select an existing folder from the pulldown list. A folder is a container for multiple subscriptions.

9. Specify the types of documents you want and the e-mail notification frequency.
10. Scroll to the bottom of the page and click **Submit**.

To view your current subscriptions and subscription folders, click **My subscriptions**.

If you experience problems with the **My notifications** feature, click the **Feedback** link in the left navigation panel and follow the instructions provided.

Receiving support updates through RSS feeds

To receive RSS feeds about fixes and other software support news, go to one of the following web sites:

- RSS feed for Application Performance Analyzer for z/OS.
- RSS feed for Debug Tool for z/OS.
- RSS feed for Enterprise COBOL for z/OS.
- RSS feed for Enterprise PL/I for z/OS.
- RSS feed for Fault Analyzer for z/OS.
- RSS feed for File Export for z/OS.
- RSS feed for File Manager for z/OS.
- RSS feed for WebSphere Studio Asset Analyzer.
- RSS feed for Workload Simulator for z/OS and OS/390.

If you need to contact IBM Software Support

IBM Software Support provides assistance with product defects.

Before contacting IBM Software Support, your company must have an active IBM software maintenance contract, and you must be authorized to submit problems to IBM. The type of software maintenance contract that you need depends on the type of product you have:

- For IBM distributed software products (including, but not limited to, Tivoli®, Lotus®, and Rational products, as well as DB2 and WebSphere products that run on Windows, or UNIX operating systems), enroll in Passport Advantage® in one of the following ways:

  Online

  Go to the Passport Advantage Web site at http://www.lotus.com/services/passport.nsf/ WebDocs/Passport_Advantage_Home and click **How to Enroll**.

  By phone

  For the phone number to call in your country, go to the IBM Software Support Web site at http://techsupport.services.ibm.com/guides/contacts.html and click the name of your geographic region.
- For customers with Subscription and Support (S & S) contracts, go to the Software Service Request Web site at https://techsupport.services.ibm.com/ssr/login.
- For customers with IBMLink, CATIA, Linux, S/390®, iSeries®, pSeries®, zSeries®, and other support agreements, go to the IBM Support Line Web site at http://www.ibm.com/services/us/index.wss/so/its/a1000030/dt006.

- For IBM eServer™ software products (including, but not limited to, DB2 and WebSphere products that run in zSeries, pSeries, and iSeries environments), you can purchase a software maintenance agreement by working directly with an IBM sales representative or an IBM Business Partner. For more information about support for eServer software products, go to the IBM Technical Support Advantage Web site at http://www.ibm.com/servers/eserver/techsupport.html.

If you are not sure what type of software maintenance contract you need, call 1-800-IBMSERV (1-800-426-7378) in the United States. From other countries, go to the contacts page of the *IBM Software Support Handbook* on the Web at http://techsupport.services.ibm.com/guides/contacts.html and click the name of your geographic region for phone numbers of people who provide support for your location.

To contact IBM Software support, follow these steps:

1. "Determining the business impact"
2. "Describing problems and gathering information"
3. "Submitting problems" on page 110

Determining the business impact

When you report a problem to IBM, you are asked to supply a severity level. Therefore, you need to understand and assess the business impact of the problem that you are reporting. Use the following criteria:

Severity 1
: The problem has a **critical** business impact. You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.

Severity 2
: The problem has a **significant** business impact. The program is usable, but it is severely limited.

Severity 3
: The problem has **some** business impact. The program is usable, but less significant features (not critical to operations) are unavailable.

Severity 4
: The problem has **minimal** business impact. The problem causes little impact on operations, or a reasonable circumvention to the problem was implemented.

Describing problems and gathering information

When describing a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Software Support specialists can help you solve the problem efficiently.

To save time, if there is a Mustgather document available for the product, refer to the Mustgather document and gather the information specified. Mustgather documents contain specific instructions for submitting your problem to IBM and gathering information needed by the IBM support team to resolve your problem. To determine if there is a Mustgather document for this product, go to the product support page and search on the term Mustgather. At the time of this publication, the following Mustgather documents are available:

- Mustgather: Read first for problems encountered with Application Performance Analyzer for z/OS: http://www.ibm.com/support/docview.wss?rs=2300 &context=SSFMHB&q1=mustgather&uid=swg21265542&loc=en_US&cs=utf-8 ⟨=en
- Mustgather: Read first for problems encountered with Debug Tool for z/OS: http://www.ibm.com/support/docview.wss?rs=615&context=SSGTSD &q1=mustgather&uid=swg21254711&loc=en_US&cs=utf-8&lang=en
- Mustgather: Read first for problems encountered with Fault Analyzer for z/OS:http://www.ibm.com/support/docview.wss?rs=273&context=SSXJAJ &q1=mustgather&uid=swg21255056&loc=en_US&cs=utf-8&lang=en
- Mustgather: Read first for problems encountered with File Manager for z/OS: http://www.ibm.com/support/docview.wss?rs=274&context=SSXJAV &q1=mustgather&uid=swg21255514&loc=en_US&cs=utf-8&lang=en
- Mustgather: Read first for problems encountered with Enterprise COBOL for z/OS: http://www.ibm.com/support/docview.wss?rs=2231&context=SS6SG3 &q1=mustgather&uid=swg21249990&loc=en_US&cs=utf-8&lang=en
- Mustgather: Read first for problems encountered with Enterprise PL/I for z/OS: http://www.ibm.com/support/docview.wss?rs=619&context=SSY2V3 &q1=mustgather&uid=swg21260496&loc=en_US&cs=utf-8&lang=en

If the product does not have a Mustgather document, please provide answers to the following questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, and messages that are related to the problem symptoms? IBM Software Support is likely to ask for this information.
- Can you re-create the problem? If so, what steps were performed to re-create the problem?
- Did you make any changes to the system? For example, did you make changes to the hardware, operating system, networking software, and so on.
- Are you currently using a workaround for the problem? If so, be prepared to explain the workaround when you report the problem.

Submitting problems

You can submit your problem to IBM Software Support in one of two ways:

Online

Click **Open service request** on the IBM Software Support site at http://www.ibm.com/software/support/probsub.html. In the Other support tools section, select IBMLink to open an Electronic Technical Response (ETR). Enter your information into the appropriate problem submission form.

By phone

Call 1-800-IBMSERV (1-800-426-7378) in the United States or, from other countries, go to the contacts page of the *IBM Software Support Handbook* at http://techsupport.services.ibm.com/guides/contacts.html and click the name of your geographic region.

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Software Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the

Software Support Web site daily, so that other users who experience the same problem can benefit from the same resolution.

After a Problem Management Record (PMR) is open, you can submit diagnostic MustGather data to IBM using one of the following methods:

- FTP diagnostic data to IBM
- If FTP is not possible, email diagnostic data to techsupport@mainz.ibm.com. You must add PMR xxxxx bbb ccc in the subject line of your email. xxxxx is your PMR number, bbb is your branch office, and ccc is your IBM country code. Click here http://itcenter.mainz.de.ibm.com/ecurep/mail/subject.html for more details.

Always update your PMR to indicate that data has been sent. You can update your PMR online or by phone as described above.

Appendix D. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The accessibility features in z/OS provide accessibility for Debug Tool.

The major accessibility features in z/OS enable users to:

- Use assistive technology products such as screen readers and screen magnifier software
- Operate specific or equivalent features by using only the keyboard
- Customize display attributes such as color, contrast, and font size

The *IBM System z Enterprise Development Tools & Compilers* Information Center, and its related publications, are accessibility-enabled. The accessibility features of the information center are described at *http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/topic/com.ibm.help.doc/accessibility_info.html*.

Using assistive technologies

Assistive technology products work with the user interfaces that are found in z/OS. For specific guidance information, consult the documentation for the assistive technology product that you use to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces by using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Volume 1* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Accessibility of this document

Information in the following formats of this document is accessible to visually impaired individuals who use a screen reader:

- HTML format when viewed from the *IBM System z Enterprise Development Tools & Compilers* Information Center
- BookManager® format when viewed with IBM BookManager BookServer (except for syntax diagrams)

Syntax diagrams start with the word Format or the word Fragments. Each diagram is preceded by two images. For the first image, the screen reader will say "Read syntax diagram". The associated link leads to an accessible text diagram. When you return to the document at the second image, the screen reader will say "Skip visual syntax diagram" and has a link to skip around the visible diagram.

For BookManager users only: A screen reader might say the lines, symbols, and words in a diagram, but not in a meaningful way. For example, you might hear "question question dash dash MOVE dash dash plus dash dash

literal-1 dash dash plus″ for part of the MOVE statement. You can enter **Say Next Paragraph** to move quickly through syntax diagrams if your screen reader has that capability.

Notices

This information was developed for products and services offered in the U.S.A. IBM might not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with the local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Trademarks and service marks

IBM, the IBM logo, and ibm.com® are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

LINUX is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT®, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

MasterCraft is a trademark of Tata Consultancy Services Ltd.

Glossary

This glossary defines technical terms and abbreviations used in *Debug Tool Customization Guide* documentation. If you do not find the term you are looking for, refer to the *IBM Glossary of Computing Terms*, located at the IBM Terminology web site:

`http://www.ibm.com/ibm/terminology`

B

batch. Pertaining to a predefined series of actions performed with little or no interaction between the user and the system. Contrast with *interactive*.

batch job. A job submitted for batch processing. See *batch*. Contrast with *interactive*.

C

CADP. A CICS-supplied transaction used for managing debugging profiles from a 3270 terminal.

compile. To translate a program written in a high level language into a machine-language program.

compile unit. A sequence of HLL statements that make a portion of a program complete enough to compile correctly. Each HLL product has different rules for what comprises a compile unit.

compiler. A program that translates instructions written in a high level programming language into machine language.

D

data set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

debug. To detect, diagnose, and eliminate errors in programs.

DTCN. Debug Tool Control utility, a CICS transaction that enables the user to identify which CICS programs to debug.

debugging profile. Data that specifies a set of application programs which are to be debugged together.

F

full-screen mode. An interface mode for use with a nonprogrammable terminal that displays a variety of information about the program you are debugging.

H

hook. An instruction inserted into a program by a compiler when you specify the `TEST` compile option. Using a hook, you can set breakpoints to instruct Debug Tool to gain control of the program at selected points during its execution.

L

link-edit. To create a loadable computer program using a linkage editor.

load module. A program in a form suitable for loading into main storage for execution. In this document this term is also used to refer to a Dynamic Load Library (DLL).

logical window. A group of related debugging information (for example, variables) that is formatted so that it can be displayed in a physical window.

LU. See "logical unit."

logical unit. (1) A type of network accessible unit that enables users to gain access to network resources and communicate with each other. (2) A name used by VTAM to identify a terminal or other resource.

N

network identifier. In TCP/IP, that part of the IP address that defines a network. The length of the network ID depends on the type of network class (A, B, or C).

node name. The name assigned to a node during network definition. The format for the node name is *netid.cpname*.

P

parameter. Data passed between programs or procedures.

partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

PDS. See *partitioned data set*.

physical window. A section of the screen dedicated to the display of one of the four logical windows: Monitor window, Source window, Log window, or Memory window.

PLU. See *primary logical unit*.

primary logical unit. (1) In SNA, the logical unit that contains the primary half-session for a particular logical unit-to-logical unit (LU-to-LU) session. (2) In SNA, the logical unit (LU) that sends the BIND to activate a session with its partner LU.

profile. A group of customizable settings that govern how the user's session appears and operates.

program. A sequence of instructions suitable for processing by a computer. Processing can include the use of an assembler, a compiler, an interpreter, or a translator to prepare the program for execution, as well as to execute it.

S

secondary logical unit. (1) In SNA, the logical unit (LU) that contains the secondary half-session for a particular LU-LU session. An LU may contain secondary and primary half-sessions for different active LU-LU sessions. (2) A VTAM Secondary Logical Unit (i.e., terminal).

session. The events that take place between the time the user starts an application and the time the user exits the application.

SIMLOGON. A VTAM macro instruction that initiates a session in which the application program acts as the PLU.

Single Point of Control. The control interface that sends commands to one or more members of an IMSplex and receives command responses.

SLU. See "secondary logical unit."

SPOC. See "Single Point of Control."

statement. (1) An instruction in a program or procedure. (2) In programming languages, a language construct that represents a step in a sequence of actions or a set of declarations.

U

utility. A computer program in general support of computer processes; for example, a diagnostic program, a trace program, or a sort program.

V

VTAM. See "Virtual Telecommunications Access Method."

Virtual Telecommunications Access Method (VTAM). (1) IBM software that controls communication and the flow of data in an SNA network by providing the SNA application programming interfaces and SNA networking functions. An SNA network includes subarea networking, Advanced Peer-to-Peer Networking (APPN), and High-Performance Routing (HPR). Beginning with Release 5 of the OS/390 operating system, the VTAM for MVS/ESA function was included in Communications Server for OS/390; this function is called Communications Server for OS/390 - SNA Services. (2) An access method commonly used by MVS to communicate with terminals and other communications devices.

Bibliography

Debug Tool publications

Using CODE/370 with VS COBOL II and OS PL/I, SC09-1862

Debug Tool for z/OS

You can access Debug Tool publications through the IBM System z Enterprise Development Tools and Compilers information center. You can receive RSS feeds about updates to the information center by following the instructions in the topic "Subscribe to information center updates", which is in the IBM System z Enterprise Development Tools and Compilers information center.

| *Debug Tool API User's Guide and Reference*,
| SC14-7250

Debug Tool Coverage Utility User's Guide and Messages, SC14-7247

Debug Tool Customization Guide, GC14-7257

Debug Tool Reference and Messages, GC14-7255

Debug Tool Reference Summary, GC14-7256

Debug Tool User's Guide, SC14-7245

Program Directory for IBM Debug Tool for z/OS, GI11-9130

COBOL and CICS Command Level Conversion Aid for OS/390 & MVS & VM: User's Guide, SC26-9400-02

| *Program Directory for IBM COBOL and CICS
| Command Level Conversion Aid for OS/390 &
| MVS & VM*, GI10-5080-04

| *Japanese Program Directory for IBM COBOL and
| CICS Command Level Conversion Aid for OS/390
| & MVS & VM*, GI10-6976-02

High level language publications

z/OS C and C++

Compiler and Run-Time Migration Guide, GC09-4913

Curses, SA22-7820,

Language Reference, SC09-4815

Programming Guide, SC09-4765

Run-Time Library Reference, SA22-7821

User's Guide, SC09-4767

Enterprise COBOL for z/OS, Version 4

Enterprise COBOL for z/OS Compiler and Runtime Migration Guide, GC27-1409

Enterprise COBOL for z/OS Customization Guide, SC23-8526

Enterprise COBOL for z/OS Licensed Program Specifications, GI11-7871

Enterprise COBOL for z/OS Language Reference, SC23-8528

Enterprise COBOL for z/OS Programming Guide, SC23-8529

Enterprise COBOL for z/OS and OS/390, Version 3

Migration Guide, GC27-1409

Customization, GC27-1410

Licensed Program Specifications, GC27-1411

Language Reference, SC27-1408

Programming Guide, SC27-1412

COBOL for OS/390 & VM

Compiler and Run-Time Migration Guide, GC26-4764

Customization under OS/390, GC26-9045

Language Reference, SC26-9046

Programming Guide, SC26-9049

Enterprise PL/I for z/OS and OS/390

Diagnosis Guide, SC27-1459

Language Reference, SC27-1460

Licensed Program Specifications, GC27-1456

Messages and Codes, SC27-1461

Migration Guide, GC27-1458

Programming Guide, SC27-1457

VisualAge PL/I for OS/390

Compiler and Run-Time Migration Guide, SC26-9474

Diagnosis Guide, SC26-9475

Language Reference, SC26-9476

Licensed Program Specifications, GC26-9471

Messages and Codes, SC26-9478

Programming Guide, SC26-9473

PL/I for MVS & VM

Compile-Time Messages and Codes, SC26-3229

Compiler and Run-Time Migration Guide, SC26-3118

Diagnosis Guide, SC26-3149

Installation and Customization under MVS, SC26-3119

Language Reference, SC26-3114

Licensed Program Specifications, GC26-3116

Programming Guide, SC26-3113

Reference Summary, SX26-3821

Related publications

CICS

Application Programming Guide, SC34-6231

Application Programming Primer, SC34-0674

Application Programming Reference, SC34-6232

DB2 Universal Database™ for z/OS

Administration Guide, SC18-7413

Application Programming and SQL Guide, SC18-7415

Command Reference, SC18-7416

Data Sharing: Planning and Administration, SC18-7417

Installation Guide, GC18-7418

Messages and Codes, GC18-7422

Reference for Remote DRDA Requesters and Servers*, SC18-7424

Release Planning Guide, SC18-7425

SQL Reference, SC18-7426

Utility Guide and Reference, SC18-7427

IMS

IMS Application Programming: Database Manager, SC27-1286

IMS Application Programming: EXEC DLI Commands for CICS & IMS, SC27-1288

IMS Application Programming: Transaction Manager, SC27-1289

TSO/E

Command Reference, SA22-7782

Programming Guide, SA22-7788

System Programming Command Reference, SA22-7793

User's Guide, SA22-7794

z/OS

MVS JCL Reference, SA22-7597

MVS JCL User's Guide, SA22-7598

MVS System Commands, SA22-7627

z/OS Language Environment

Concepts Guide, SA22-7567

Customization, SA22-7564

Debugging Guide, GA22-7560

Programming Guide, SA22-7561

Programming Reference, SA22-7562

Run-Time Migration Guide, GA22-7565

Vendor Interfaces, SA22-7568

Writing Interlanguage Communication Applications, SA22-7563

Softcopy publications

Online publications are distributed on CD-ROMs and can be ordered through your IBM representative. *Debug Tool User's Guide*, *Debug Tool Customization Guide*, and *Debug Tool Reference and Messages* are distributed on the following collection kit:

SK3T-4269

Online publications can also be downloaded from the IBM Web site. Visit the IBM Web site for each product to find online publications for that product.

Index

Special characters

A

B

C

D

E

F

W

Readers' Comments — We'd Like to Hear from You

Debug Tool for z/OS
Customization Guide
Version 10.1

Publication No. GC14-7257-00

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: COMMENTS@US.IBM.COM

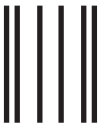
If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
GC14-7257-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Silicon Valley Laboratory
Department J87/D325
555 Bailey Ave.
San Jose, CA
95141-9989

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Program Number: 5655-V50

Printed in USA

GC14-7257-00