Debug Tool for z/OS

Customization Guide

Version 13.1

SC14-7601-04

Debug Tool for z/OS

Customization Guide

Version 13.1

SC14-7601-04

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 179.

Fifth Edition (October 2015)

This edition applies to Debug Tool for z/OS, Version 13.1 (Program Number 5655-Q10 with the PTF for APAR PI37275), 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 Version 1 (Program Number 5694-A01)
- C/C++ feature of z/OS Version 2 (Program Number 5650-ZOS)
- 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 (Program Number 5655-S71)
- Enterprise COBOL for z/OS Version 5 Release 1 (Program Number 5655-W32)
- High Level Assembler for MVS & VM & VSE Version 1 Release 4, Version 1 Release 5, 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 (Program Number 5655-H31)
- Enterprise PL/I for z/OS Version 4.4 and earlier (Program Number 5655-W67)

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

You can access publications online at www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss

You can find out more about Debug Tool by visiting the IBM Web site for Debug Tool at: www.ibm.com/software/products/us/en/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 the Terminal Interface Manager 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, (GI10-8928), 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.

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, and part of chapter 8, 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.
- Chapter 14 describes how to specify default and allowable values for the runtime options NATLANG, LOCALE, and LINECOUNT.
- Chapter 15 describes the features or functions you can implement through the EQAOPTS commands.
- Chapter 16 describes the customization tasks you must do for users debugging applications with remote debuggers.
- Appendix A describes SMP/E USERMODs that are available for some customizations.
- Appendix B is a copy of Chapter 6 from an earlier Customization Guide for Debug Tool.
- Appendix C describes how to apply maintenance provided for Debug Tool.
- Appendix D describes all the resources available to help you find technical support information.
- Appendix E 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 LangX COBOL equivalent**	**PL/I equivalent**	**assembler**
Compile unit	C and C++ source file	Program	• 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

Note:

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 *LangX 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.

LangX COBOL
Refers to any of the following COBOL programs supported through use of the EQALANGX (or IDILANGX) debug file:

- Programs compiled using the IBM OS/VS COBOL compiler.
- Programs compiled using the VS COBOL II compiler with the `NOTEST` compiler option.
- Programs compiled using the Enterprise COBOL compiler with the `NOTEST` compiler option.

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 link them with the non-Language Environment library. VS COBOL II programs are Language Environment programs when you link them with the Language Environment library.
Enterprise COBOL programs are always Language Environment programs. Note that COBOL DLL's cannot be debugged as LangX COBOL programs. 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 LangX 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

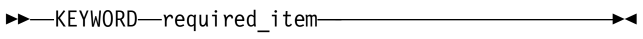
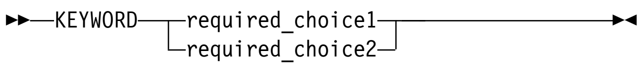
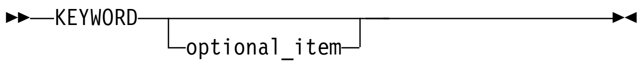
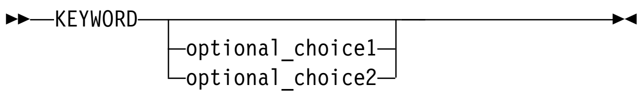
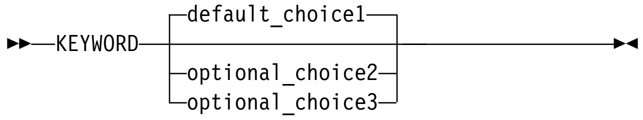
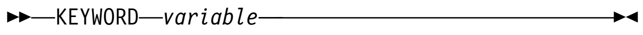
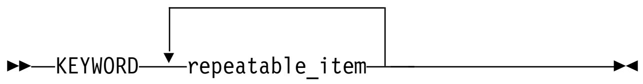
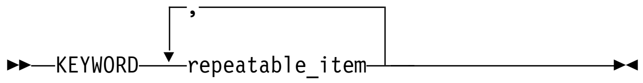
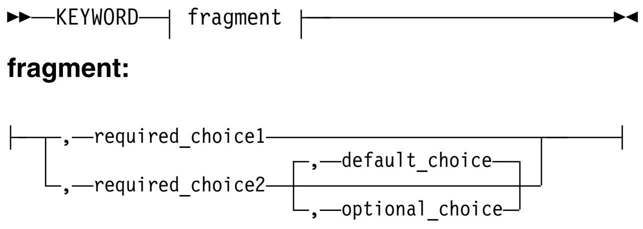
Item	Syntax example
Required item. Required items appear on the main path of the horizontal line. You must specify these items.	
Required choice. A required choice (two or more items) appears in a vertical stack on the main path of the horizontal line. You must choose one of the items in the stack.	
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.	

Table 1. Syntax examples (continued)

Item	Syntax example
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.	 KEYWORD default_choice1 optional_choice2 optional_choice3
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.	 KEYWORD fragment fragment: ,─required_choice1 ,─required_choice2 ,─default_choice ,─optional_choice

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.
- Send your comments by email to comments@us.ibm.com. Be sure to include the name of the book, the part number of the book, the version of Debug Tool, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).

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

Changes introduced with the PTF for APAR PI37275

- The IMS Transaction Isolation Facility is added to enhance the isolation of Debug Tool users in IMS message regions. Debug Tool users can register to debug certain IMS transactions in a given IMS system. Users can also start a private IMS region, which is cloned from the currently-running environment for the selected transaction.

  For more information, see "Scenario F: Enabling the Transaction Isolation Facility" on page 90.
- Support is added for deferred breakpoints for COBOL and PL/I programs. The new Debug Tool Deferred Breakpoints option in Debug Tool Utilities allows users to create and view a list of breakpoints prior to starting the debug session. It reduces the time spent on the debugging session and also the system resource usage.
- Support is added for the Debug Tool JCL Wizard. By using this new ISPF edit macro, **EQAJCL**, you can modify a JCL or procedure member and create statements to invoke Debug Tool in various environments. You can also create statements to invoke Debug Tool for the Terminal Interface Manager, Dedicated Terminal, or Remote GUI available with RD/z and PD Tools Studio.

  For more information , see "Installing and customizing the Debug Tool JCL Wizard" on page 57.
- Support is added to allow STEP and GO in the RD/z Debug Configuration Startup commands.
- Support is added for the code coverage and load module analyzer plug-ins.

  For more information , see "Adding support for the DTSP Profile, code coverage, and load module analyzer views" on page 142.
- New EQAOPTS commands DOPTACBDSN, IGNOREODOLIMIT, and MAXTRANUSER.

  For more information , see "DOPTACBDSN" on page 115, "IGNOREODOLIMIT" on page 119, and "MAXTRANUSER" on page 122.

Changes introduced with the PTF for APAR PI29800

- Support is added for debugging subtasks initiated by the ATTACH assembler macro. Multiple subtasks might be debugged concurrently, as well as the parent task.
- Delay debug mode is enhanced to support non-LE programs in certain circumstances.

Changes introduced with the PTF for APAR PI24559

- The environment specific Language Environment user exits EQADBCXT, EQADDCXT, and EQADICXT are now in sunset mode and will be removed in the next Debug Tool version. Users should move to the consolidated environment specific Language Environment user exit EQAD3CXT.

  Debug Tool Utilities option 'B Delay Debug Profile' and the DTSP Profile Manager plug-in now create an enhanced debug profile that contains a new <NM2> tag (rather than the old <PGM> tag) and will only work with the consolidated user exit.

For more information, see Chapter 7, "Specifying the TEST runtime options through the Language Environment user exit," on page 25.

- Delay debug mode is enhanced to include C functions. In addition, a load module name can now be paired with a program name or C function in the delay debug data set.
- Terminal Interface Manager now allows the same user to log on multiple times on separate terminals. This allows multiple tasks to be debugged by the same user ID simultaneously.
- Support is added for Automonitor for C/C++.
- `M/L` prefix commands are enhanced to support these prefix commands for C/C++.
- The CICS DTST transaction that is used to display, scan and modify CICS storage is enhanced to support 64 bit addresses.
- Code Coverage support is added for Enterprise COBOL for z/OS Version 5.
- `CALL %FA` command is enhanced to support remote debug mode.
- `AT GLOBAL LABEL`, `AT LABEL`, `CLEAR AT GLOBAL LABEL`, `CLEAR AT LABEL`, `LIST AT GLOBAL LABEL`, `LIST AT LABEL` and `LIST NAMES LABEL` commands are enhanced to support remote debug mode.

Changes introduced with the PTF for APAR PI16543

- Support is added for CICS TS 5.2.
- The DTCN Remote Plug-in is enhanced to support the use of SYSID (the SYSIDNT of a region in a CICSPLEX) to select CICS tasks to debug.
- The DTCN Remote Plug-in is enhanced to support additional CICS "Application" resource types that are introduced in CICS TS 5.2.

  You can use these new resource types to select a program to debug:
  - Platform
  - Application
  - Operation
  - Version
- Code Coverage support for interactive remote.
- Code Coverage support for z/OS XL C.
- SET LIST BY SUBSCRIPT ON support for COBOL for MFI. This includes QUERY LIST BY SUBSCRIPT support.
- The CLIENTID option is added to the EQAOPTS DLAYDBGXRF command. This option allows a remote Debug Tool user using the enhanced DTSP Plug-in to identify a specific DB/2 client ID, and to only trap DB/2 stored procedures executed using that client ID.

Changes introduced with the PTF for APAR PI06312

- The REPOSITORY option is added to the EQAOPTS DLAYDBGXRF command. This option instructs Debug Tool to communicate with the Terminal Interface Manager to determine whether a user requests to debug an IMS transaction or DB/2 stored procedure associated with a generic user ID. This is an alternative to using the cross reference table data set.
- The Swap IMS Transaction Class and Run Transaction utility is enhanced to allow the user to manipulate the TEST option that is used for the debug message region. This allows the user to supply commands or preference files, and to direct the Debug Tool session to the remote user interface or to an alternate Terminal Interface Manager user ID.

- For CICS, provide protection of storage that was GETMAINed in the current task by a program that is not the active program. This support is enabled via INITPARM=(EQA0CPLT='STG') in the CICS startup parameters and is only available during a remote debug session.
- <PROGRAMDSCOMPILEDATE> and <PROGRAMDSCOMPILETIME> tags are added to the XML tags for code coverage. These two tags specify the compile data and time of the program source that is contained in the program data set.
- Support is added for generating a client certificate to the remote debugger if it does not exist.
- A note is added to the Coverage Utility User's Guide and Messages about the accuracy of execution counts (frequency counts) for single statement Program Areas (PAs).

Changes introduced with Debug Tool for z/OS Version 13.1

- A method for gathering code coverage is added for the generation, viewing, and reporting of code coverage by using the Debug Tool mainframe interface (MFI) as the engine. This support is provided for applications written in Enterprise COBOL and Enterprise PL/I that are compiled with the TEST compiler option and its suboption SEPARATE. This is enabled via the new EQAOPTS CCPROGSELECTDSN, CCOUTPUTDSN, and CCOUTPUTDSNALLOC commands.
- Debug Tool is enhanced to provide the automatic start of IMS message processing program (MPP) regions and dynamic routing of transactions. This allows a developer to dynamically start an MPP region, route a transaction to that MPP region, and at the end of the transaction shutdown the MPP region created for the developer thus reducing system resources.
- To help with the ease of use of the MFI mode of Debug Tool for some users, an option is added that enables breakpoints, the current line, and the line with found text to be identified by a character indicator. This feature is enabled via a new EQAOPTS ALTDISP command.
- Debug Tool is enhanced to support the following languages and platforms:
  - Enterprise COBOL for z/OS V5.1
  - Enterprise PL/I for z/OS V4.4
  - CICS TS V5.1
  - DB2 V11
  - IMS V13
  - z/OS, V2.1
  - C/C++ V2.1
- Debug Tool is enhanced to support JCL for Batch Debugging in the DTSP plug-in. This facility is used to instrument JCL to initiate a debug session from the DTSP plug-in.
- Support is added for an IMS transaction that is associated with a generic ID. A new feature is added to the consolidated Language Environment user exit (EQAD3CXT) to search a new cross-reference table for the user ID of a user who wants to debug a IMS transaction that is started from the web and is associated with a generic ID. This enables Debug Tool to debug these transactions that use a generic ID. The user ID from the cross-reference table is used to find the user's Debug Tool user exit data set (userid.DBGTOOL.EQAUOPTS), which specifies the TEST runtime parameters and the display device address. An option is added to the Debug Tool Utilities ISPF panel, "C IMS Transaction and User ID Cross Reference Table", to allow each user to update the new cross reference table.

- Support is added for tracing load modules loaded by an application. Commands TRACE LOAD and LIST TRACE LOAD are added for Debug Tool's MFI mode. This set of commands allows the user to get a trace of load modules loaded by the application. Start the trace by issuing TRACE LOAD START. Use LIST TRACE LOAD to display the trace. The trace includes load modules known to Debug Tool at the time the TRACE LOAD START command is entered and all that are loaded while the trace is active. End the trace by issuing TRACE LOAD END. Note that when the trace is ended, all trace information is deleted.
- Support is added for terminating an idle Debug Tool session that uses the Terminal Interface Manager. Debug Tool supports a timeout option via the EQAOPTS SESSIONTIMEOUT command. This command allows the system programmer to establish a maximum wait time for debug sessions that use a dedicated terminal or the Terminal Interface Manager. If the debug session exceeds the specified time limit without any user interaction, the session will be terminated with either a QUIT or QUIT DEBUG.
- Debug Tool Coverage Utility "Create HTML Targeted Coverage Report" is enhanced to allow the user to select from a list of COBOL Program-IDs, ignore changes to non-executable code, and produce a summary of the targeted lines with selectable HTML links.
- Adds IMS information to start and stop messages generated by the EQAOPTS STARTSTOPMSG command.
- Adds EQAOPTS STARTSTOPMSGDSN command and a Debug Tool Utilities option "Non-CICS Debug Session Start and Stop Message Viewer" to collect and view Debug Tool debugger session start and stop information.
- Delay debug mode is enhanced with an EQAOPTS DLAYDBGCND command to control CONDITION trapping. In addition, an EQAOPTS DLAYDBGXRF command is added so that delay debug mode can use the "IMS Transaction and User ID Cross Reference Table". Further, NOTEST is now handled in delay debug mode.
- A confirmation message is added to Debug Tool Utilities Option 6 "Debug Tool User Exit Data Set" to indicate that the updates have been saved into the EQAUOPTS data set.
- The ON and AT OCCURRENCE commands are enhanced for Enterprise PL/I to support qualifying data.
- LIST LDD and CLEAR LDD commands are added to display and remove LDD commands known to Debug Tool. LIST CC, CC START, and CC STOP commands are added to gather and display code coverage data.
- Two EQAOPTS commands are added for remote debug mode. The EQAOPTS HOSTPORTS command specifies the specific host port number or range of host port numbers on the host for a TCP/IP connection from the host to a workstation. The EQAOPTS TCPIPDATADSN command provides the data set name for TCPIP.DATA via the SYSTCPD DD NAME when no GLOBALTCPIPDATA statement is configured.
- A timestamp is added to the EQAY999* messages that the Terminal Interface Manager issues if the +T trace flag is on.
- Debug Tool is enhanced to allow for using GOTO or JUMPTO command for programs that are compiled with OPT and NOEJPD suboptions of the Enterprise COBOL TEST compile option when SET WARNING setting is OFF.
- An updated DTST transaction is included to write messages to the operator log when a user changes storage. These messages are intended to provide an audit trail of DTST storage changes.
- Support is added for remote Playback through the Playback Toolbar in the Debug View.

- The EQALANGP and EQALANGX modules are moved from Debug Tool's EQAW.SEQAMOD library to Common Component's IPV.SIPVMODA library. They are now aliases of IPVLANGP and IPVLANGX respectively. This removes duplication between the two tools.
- Appendix "Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products" in the *Debug Tool User's Guide* is removed because this has been placed in the *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide* instead.

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 the Terminal Interface Manager, 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 compulsory customization tasks:

- Chapter 2, "Product Registration," on page 7
- Chapter 3, "Installing the Debug Tool SVCs," on page 9.
- Chapter 4, "Setting up the APF-authorized system link list data set (SEQABMOD)," on page 13
- Chapter 5, "Setting up the link list data set (SEQAMOD)," on page 15
- Chapter 6, "Enabling debugging in full-screen mode using the Terminal Interface Manager," on page 17.
- If you previously installed any of the Language Environment user exits: EQADDCXT, EQADICXT, EQADBCXT, or EQAD3CXT, rebuild these exits. Debug Tool has updated the sample assembler user exits and the load modules. To review instructions on how to build the exits, see Chapter 7, "Specifying the TEST runtime options through the Language Environment user exit," on page 25.

  If you did not previously install these user exits, then do the instructions in Chapter 7, "Specifying the TEST runtime options through the Language Environment user exit," on page 25.
- Read Chapter 8, "Installing the browse mode RACF facility," on page 33 if you want to control which users have access to Debug Tool, or control which users can access Debug Tool only through browse mode.

  Note: If you have defined a generic Facility class profile (for example, *.*), you might have to install the browse mode RACF® facilities, even if neither of the previous considerations applies. For example, if you have a generic Facility class profile of *.* with UACC(NONE) and you do not install the browse mode RACF facilities, no users would be allowed to use Debug Tool.

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

- "Choosing a method to start Debug Tool Utilities" on page 36.
- "Customizing the data set names in EQASTART" on page 38.
- "Adding Debug Tool Utilities to the ISPF menu" on page 38.
- 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 39.

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 38.
- If you are using other Problem Determination Tools (File Manager for z/OS), see "Customizing Problem Determination Tools for multiple systems" on page 41.
- If you are using Program Preparation, see "Customizing Program Preparation" on page 42.
- If you are using Coverage Utility, see "Customizing Coverage Utility" on page 44.
- For the IMS BTS Debugging option, you must specify default values for yb2* parameters. See "Customizing IMS BTS Debugging" on page 51.

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 DB2 stored procedures," on page 61.
- If your site debugs CICS programs, see Chapter 11, "Adding support for debugging under CICS," on page 63.
- If your site debugs IMS programs, see Chapter 12, "Adding support for debugging under IMS," on page 85 and implement scenario A.
- If your site debugs non-Language Environment IMS programs, see Chapter 12, "Adding support for debugging under IMS," on page 85 and implement scenario C.

In Debug Tool Version 13.1, the EQALANGP and EQALANGX modules are moved from Debug Tool's EQAW.SEQAMOD library to Common Component's IPV.SIPVMODA library, where they will be aliases of IPVLANGP and IPVLANGX respectively. This removes duplication between the two tools. If you have library build processes or other tools that reference either of these routines, and IPV.SIPVMODA is not in link list, then you will need to update your processes to point to the new location for these routines. See *Preparing a LANGX COBOL program* and *Preparing an assembler program* in the *Debug Tool User's Guide* for more information about EQALANGX.

As you review the rest of the checklist, if you need to do an item that requires that you specify an EQAOPTS command, you can print a copy of Table 17 on page 99 and use it to record the commands you need to specify and the values for any options. When you are done reviewing the checklist, you can specify all the EQAOPTS commands at one time as described in "Creating EQAOPTS load module" on page 105.

For any of the following situations, see "CODEPAGE" on page 109:

- 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 an EQAOPTS `CODEPAGE` command 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 an EQAOPTS `CODEPAGE` command 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))` parameter 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:

- These EQAOPTS commands enable certain Debug Tool functions:
  - "ALTDISP" on page 105
    You want Debug Tool to display the at sign (@) in the prefix area of a line to indicate that the line contains a breakpoint, instead of using a colored line.
  - "BROWSE" on page 106
    You want to restrict access to Debug Tool or control which users[1] must debug in browse mode.
  - "CCOUTPUTDSN" on page 107
    You want to use Code Coverage and need to specify the name of the Observation data set.
  - "CCOUTPUTDSNALLOC" on page 107
    You want to use Code Coverage and need to specify the allocation parameters for the Observation data set.
  - "CCPROGSELECTDSN" on page 108
    You want to use Code Coverage and need to specify the name of the Options data set.
  - "DLAYDBG" on page 112
    You want to allow users to use the delay debug mode.
  - | "DOPTACBDSN" on page 115
    | You want to use the IMS Transaction Isolation Facility described in
    | Chapter 12, "Adding support for debugging under IMS," on page 85.
  - "EQAQPP" on page 117
    Your site needs to debug Q++ programs.
  - | "IGNOREODOLIMIT" on page 119
    | You want to tell Debug Tool to display COBOL table items even when an
    | ODO value is out of range.
  - "LOGDSNALLOC" on page 121
    You want Debug Tool to automatically create a LOG data set for each user.

1. If you want to enforce browse mode restrictions, you must use the RACF Facility Class Profile as described in Chapter 8, "Installing the browse mode RACF facility," on page 33. You can learn how the EQAOPTS `BROWSE` command works with the RACF profiles by reviewing the table in the topic "Controlling browse mode" of the *Debug Tool User's Guide*.

  - "MDBG" on page 123
    Your site uses z/OS XL C/C++, Version 1.10, or later, and you want Debug Tool to retrieve source and debug information from `.mdbg` files.
  - "SAVESETDSNALLOC, SAVEBPDSNALLOC" on page 127
    You want Debug Tool to automatically create either of the following data sets:
    - A data set to save and restore settings
    - A data set to save and restore breakpoints, monitor values and LOADDEBUGDATA (LDD) specifications
  - "STARTSTOPMSG" on page 129
    You want Debug Tool to issue a message when each debugging session is initiated or terminated.
  - "SUBSYS" on page 131
    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 this command to determine if you need to change the SUBSYS parameter.
  - "SVCSCREEN" on page 131
    You need to debug non-Language Environment programs that start under Language Environment, Language Environment programs that use the MVS™ LINK, LOAD or DELETE services, LangX COBOL programs, or your site has any host products that might use SVC screening when Debug Tool is started.
- If your application programmers debug in remote debug mode, review Chapter 16, "Adding support for remote debug users," on page 137.
- 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 96:
  - Debug Tool Utilities ISPF panels
  - Debug Tool Coverage Utility
  - EQANMDBG (non-CICS non-Language Environment support)

Do the customization tasks in the following list only if you want to modify the behavior described:
- These EQAOPTS commands modify the behavior of certain Debug Tool functions:
  - "CACHENUM" on page 106
    You want to reduce Debug Tool's CPU consumption in certain cases.
  - "COMMANDSDSN" on page 111
    You want to change the default data set name for the user's commands file.
  - "DEFAULTVIEW" on page 112
    You want to change the default setting for SET DEFAULT VIEW so that assembler macro-generated statements are not displayed in the Source window.
  - "DLAYDBGCND" on page 113
    You want to change the default delay debug setting for monitoring condition events.
  - "DLAYDBGDSN" on page 113
    You want to change the default name of the delay debug profile data set.
  - "DLAYDBGTRC" on page 114
    You want to change the default delay debug pattern match trace message level.

– "DLAYDBGXRF" on page 114
  You want to instruct delay debug to use the cross reference file to find the user ID when it constructs the delay debug profile data set name.
– "DTCNDELETEDEADPROF" on page 116
  You want to change the default setting for controlling the deletion of dead DTCN profiles.
– "DTCNFORCE*xxxx*" on page 116
  You want to change the default DTCN behavior for certain resource types.
– "DYNDEBUG" on page 117
  You want to change the initial or default value of SET DYNDEBUG.
– "GPFDSN" on page 118
  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 this command for instructions on how to create a global preferences file.
– "HOSTPORTS" on page 119
  Your users are using the remote debugger and you need to specify a host port or range of ports for a TCP/IP connection from the host to the workstation.
– "LOGDSN" on page 120
  You want to change the default data set name for the LOG data set.
| – "MAXTRANUSER" on page 122
| You want to use the IMS Transaction Isolation Facility described in
| Chapter 12, "Adding support for debugging under IMS," on page 85, and you
| need to set the maximum number of transactions that a single user can debug
| to something less than 15.
– "MULTIPROCESS" on page 123
  You want to change the default behavior of Debug Tool when a new POSIX process is created by a fork() or exec() function.
– "NAMES" on page 124
  Your site needs to issue a `NAMES` command for the initial load module or any of its compile units.
– "NODISPLAY" on page 125
  To modify Debug Tool's behavior when a full-screen mode using the Terminal Interface Manager or a remote debugger is not available.
– "PREFERENCESDSN" on page 125
  You want to change the default data set name for the user's preferences file.
– "SAVEBPDSN, SAVESETDSN" on page 126
  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.
– "SESSIONTIMEOUT" on page 128
  You want to specify an idle session timeout for users who are using the Terminal Interface Manager.
– "STARTSTOPMSGDSN" on page 130
  You want Debug Tool to write information to a log data set when each non-CICS debugging session is initiated or terminated.
– "TCPIPDATADSN" on page 134
  Your users are using the remote debugger and your host TCP/IP does not have a specification for GLOBALTCPIPDATA.
– "THREADTERMCOND" on page 135
  Your site wants Debug Tool to suppress the prompt that Language

  Environment displays every time when the statements like STOP RUN, GOBACK, or EXEC CICS RETURN are run. These statements can occur frequently in an application program, creating unnecessary interruptions for a user trying to debug the application program.
  - "TIMACB" on page 135
    You want to change the default ACB name for the Terminal Interface Monitor.
- 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 93.
- If your site needs to change the defaults for NATLANG, LOCALE, or LINECOUNT, see "Changing the default and allowable values in EQACUIDF" on page 95.

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 Debug Tool SVCs

Debug Tool requires the installation of the Debug Tool 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 Debug Tool 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).

The Debug Tool SVCs support the Dynamic Debug facility and other necessary Debug Tool functions.

To install the SVCs, do the following steps:

1. 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.
2. Follow the instructions in "Using the Authorized Debug facility for protected programs" on page 11.

Installing the SVCs without using a system IPL

To install the 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[2]. 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).
4. Any CICS or IMS regions that are running when these SVCs are installed, and that may have had Debug Tool sessions, should be stopped and restarted.

Verifying the installation of the SVCs

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

Checking the level of the Debug Tool SVCs

Display the level of the Debug Tool 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.

```
...EQA00SVC 2015.301Licensed Materials - Property of IBM 5655-Q10 Debug Tool Version 05
EQA00SVC-C8340Copyright Copyright IBM Corp. All Rights ReservedU
***> EQA00SVC is Version 05 with compile date 28 Oct 2015
```

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.

```
...EQA01SVC 2015.301Licensed Materials - Property of IBM 5655-Q10 Debug Tool Version 18
EQA01SVC-f10521Copyright Copyright IBM Corp. All Rights Reserved
***> EQA01SVC is Version 18 with compile date 28 Oct 2015

...EQA01SV2 2015.301Licensed Materials - Property of IBM 5655-Q10 Debug Tool Version 05
EQA01SV2-f10269Copyright Copyright IBM Corp. All Rights Reserved
***> EQA01SV2 is Version 05 with compile date 28 Oct 2015

...EQA01TSR 2015.077Licensed Materials - Property of IBM 5655-Q10 Debug Tool Version 00
EQA01TSR-f8730eCopyright Copyright IBM Corp. All Rights Reserved
***> EQA01TSR is Version 00 with compile date 18 Mar 2015
```

Running the installation verification programs

To help you verify the installation of the Debug Tool SVCs (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.

2. 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.

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)
EQAWIVPJ	LangX Enterprise COBOL
EQAWIVPP	COBOL TEST(NONE,SYM,SEPARATE) or TEST(NOHOOK,SEPARATE)
EQAWIVPT	Enterprise COBOL for z/OS Version 5 TEST
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.[2]
  2. Add the data set to the system link list concatenation.[3]
  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[4] 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.

3. 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.

4. 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 63.
- 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[3]. For CICS, you also need to mark *hlq*.SEQAMOD as APF-authorized[2].

Chapter 6. Enabling debugging in full-screen mode using the Terminal Interface Manager

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

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

This topic focuses on setting up the Terminal Interface Manager to enable a terminal session that can be used to debug these types of programs in full-screen mode. To set up a dedicated terminal without using Terminal Interface Manager, see Appendix B, "Enabling debugging in full-screen mode using a dedicated terminal," on page 149.

To understand how a user and Debug Tool interact with the Terminal Interface Manager, see "How users start a full-screen mode debug session with the Terminal Interface Manager."

To enable debugging in full-screen mode using the Terminal Interface Manager, see "Enabling full-screen mode using the Terminal Interface Manager" on page 19. If you are running the Terminal Interface Manager on a VTAM® network with multiple LPARs, see "Example: Defining the VTAM EQAMVnnn and Terminal Interface Manager APPL definition statements when Debug Tool runs on four LPARs" on page 23 for variations on the instructions.

How users start a full-screen mode debug session with the Terminal Interface Manager

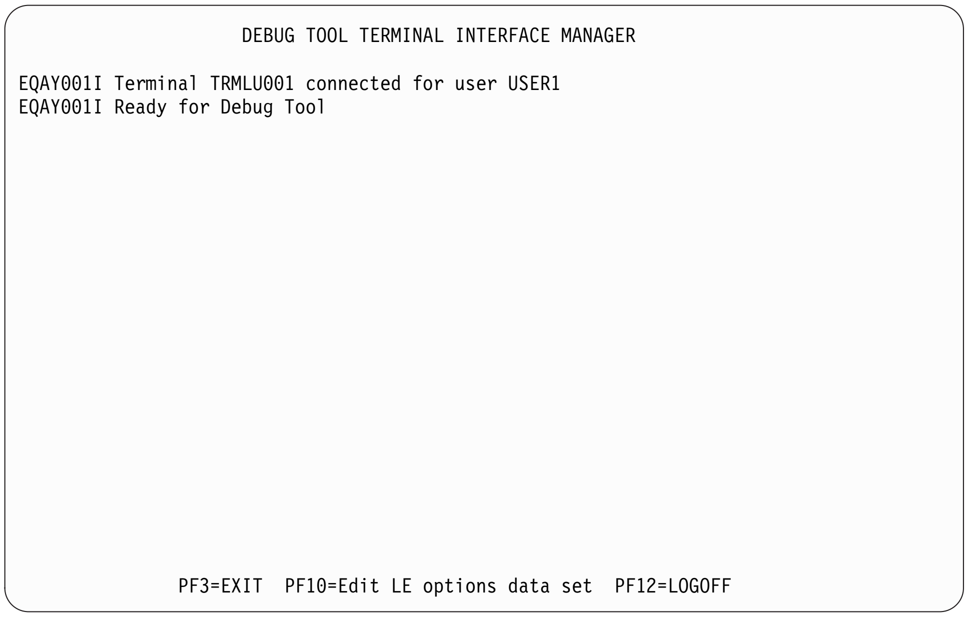
The following steps describe how a user would start a full-screen mode debugging session for a batch job with the Terminal Interface Manager. Study these steps to understand how Debug Tool uses the Terminal Interface Manager to display a full-screen mode debugging session and to understand why you need to do the configuration steps described in "Enabling full-screen mode using the Terminal Interface Manager" on page 19.

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

   In either situation, ensure that the second session connects to the Terminal Interface Manager.
2. On the first terminal session, log on to TSO.
3. On the second terminal 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 session:



```
                    DEBUG TOOL TERMINAL INTERFACE MANAGER

EQAY001I Terminal TRMLU001 connected for user USER1
EQAY001I Ready for Debug Tool


















               PF3=EXIT  PF10=Edit LE options data set  PF12=LOGOFF
```

The terminal is now ready to receive a full-screen mode debugging session.

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

   `TEST(,,,VTAM%`*`userid`*`:*)`
5. Submit the batch job. The following tasks are done:
   a. Debug Tool allocates a VTAM ACB (EQAMV*nnn*) for its end of a VTAM session.
   b. Debug Tool communicates with the Terminal Interface Manager to request a session with the terminal LU on which it is running.
   c. The Terminal Interface Manager releases the terminal LU and passes control of it to Debug Tool.
6. On the second terminal 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 session displays the panel and messages you saw in step 3 on page 17. This indicates that Debug Tool can use this session again. (This happens each time you exit 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 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.

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

Enabling full-screen mode using the Terminal Interface Manager

To enable full-screen mode using the Terminal Interface Manager, 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 VTAM APPL definition statements that the Terminal Interface Manager uses, as described in "Defining the Terminal Interface Manager APPL definition statements" on page 21.
3. Start the Terminal Interface Manager, as described in "Starting the Terminal Interface Manager" on page 21.
4. Verify that full-screen mode using the Terminal Interface Manager is enabled, as described in "Verifying the enablement of full-screen mode using the Terminal Interface Manager" on page 23.

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.

The number of APPL names you define must be sufficient to allow for the maximum number of concurrent Debug Tool full-screen mode using the Terminal Interface Manager 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 the Terminal Interface Manager 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 the Terminal Interface Manager 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 the Terminal Interface Manager spread across your network. For an example on how to configure many APPL names in a network with multiple LPARs, see "Example: Defining the VTAM EQAMVnnn and Terminal Interface Manager APPL definition statements when Debug Tool runs on four LPARs" on page 23.

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 19):

-
```
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 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 Terminal Interface Manager

The 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 Terminal Interface Manager load modules, EQAYSESM and EQAYTRMM, reside in an APF authorized library (these 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 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 Terminal Interface Manager when the system is IPLed.

   The user ID associated with the STARTED class entry in RACF for the Terminal Interface Manager started task must have an OMVS segment defined. This enables the started task to use the POSIX thread functions.

The Terminal Interface Manager load module accepts six 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 the Terminal Interface Manager to open. For more information about this parameter, see "Example: Defining the VTAM EQAMVnnn and Terminal Interface Manager APPL definition statements when Debug Tool runs on four LPARs" on page 23.

-m Instructs the Terminal Interface Manager not to fold passwords to upper case. If your security product is set up to use mixed-case passwords, and Terminal Interface Manager does not accept them, code this parameter.

-p *pattern-string*
: Specifies a naming pattern for the TEST runtime options data set. You can use this parameter to override the default naming pattern. *pattern-string* must contain the string `&USERID` for substitution purposes. Otherwise, an error will be recognized and the default naming pattern `&USERID.DBGTOOL.EQAUOPTS` will be used.

-s Instructs the Terminal Interface Manager to provide an entry field on each Terminal Interface Manager panel, in which the user can enter a session manager escape sequence.

+T Display detailed trace messages for the Terminal Interface Manager. Do not use this parameter unless instructed by IBM support personnel.

-r Start Terminal Interface Manager in repository mode. This enables users to register to debug IMS transactions or DB/2 stored procedures that are started by a generic user ID.

See "DLAYDBGXRF" on page 114 for additional customization required when using this option.

The following example starts the Terminal Interface Manager for alternate ACB EQASESS2 and instructs it to provide an extra entry field for use with a session manager:

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

Using the MODIFY operator command to display a list of Terminal Interface Manager users

Console operators can list the users who are currently logged on to the Terminal Interface Manager by using the z/OS MVS system `MODIFY` command. The list of users contains the user ID, the terminal LU that each user requests, and the job information if the user is currently in a Debug Tool session.

The syntax of the `MODIFY` command is:

```
►►──MODIFY──tim-stc-name──,──APPL=LIST──┬─ALL──┬──────────────────────────────►◄
                                        ├─SESS─┤
                                        └─IMS──┘
```

The following list describes the parameters of the `MODIFY APPL=LIST` command:

tim-stc-name
: Is the name of the started task or job that is running the Terminal Interface Manager.

ALL
: Lists all users who are logged on to the Terminal Interface Manager.

SESS
: Only lists users who are logged on to the Terminal Interface Manager and are currently in a Debug Tool session.

IMS
: Only lists users who are logged on to the Terminal Interface Manager and are currently in a Debug Tool session for an IMS message processing program.

Example

```
MODIFY EQAYSESM,APPL=LIST ALL
```

Example output

```
EQA9896I DEBUG TOOL TIM USERS
  USER ID    FLAGS    TERMINAL   JOBNAME
  --------   ------   --------   --------
  USRT002    I        TRMLU006   MPRCI10X
  USRT001    B        T1302      USRT001A
  USRT003    L        TRMLU004   **NONE**
----------------------------------------
L=LOGGED ON, NOT IN SESSION
I=IN SESSION; IMS JOB
B=IN SESSION; BATCH OR DB/2 JOB
```

Verifying the enablement of full-screen mode using the Terminal Interface Manager

To help you verify the enablement of full-screen mode using the Terminal Interface Manager, do the following steps:

1. Select which of the following installation verification jobs you want to run:
   - `EQAWIVP5` (COBOL)
   - `EQAWIVP6` (C)
   - `EQAWIVP7` (PL/I)
   - `EQAWIVP9` (Enterprise PL/I)
   - `EQAWIVPB` (Language Environment assembler)
   - `EQAWIVPD` (non-Language Environment assembler)
   - `EQAWIVPK` (LangX Enterprise COBOL)
   - `EQAWIVPW` (OS/VS COBOL)
   - `EQAWIVPY` (non-Language Environment VS COBOL II)
2. Copy the job from the *`hlq`*`.SEQASAMP` data set to your own private data set and customize it for your installation as described in the sample.
3. Connect a terminal session to the Terminal Interface Manager and log on to the Terminal Interface Manager with your TSO user ID.
4. Run the customized installation verification jobs.

Example: Defining the VTAM EQAMVnnn and Terminal Interface Manager APPL definition statements when Debug Tool runs on four LPARs

The instructions in Chapter 6, "Enabling debugging in full-screen mode using the Terminal Interface Manager," on page 17 assume that you create all the definitions and start the Terminal Interface Manager on one LPAR. 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 the Terminal Interface Manager debugging session.
- VTAM on LPAR 1 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.

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 these customized versions of the Language Environment user exit: EQADBCXT, EQADDCXT, EQADICXT, and EQAD3CXT. The following table shows the environments in which these user exits can be used. The EQAD3CXT user exit determines the runtime environment internally and can be used in multiple environments. You can use either the EQAD3CXT user exit or the environment specific user exit.

| **Note:** The environment specific Language Environment user exits (EQADBCXT,
| EQADDCXT and EQADICXT) are now in sunset mode and will be removed in the
| next Debug Tool version. Users should move to the consolidated Language
| Environment user exit (EQAD3CXT).

| Debug Tool Utilities option 'B Delay Debug Profile' and the DTSP Profile Manager
| plug-in now create an enhanced debug profile that contains a new <NM2> tag
| (rather than the old <PGM> tag) and will only work with the consolidated user
| exit.

Table 3. Language Environment user exits for various environments

Environment	User exit name
The following types of DB2 stored procedures that run in WLM-established address spaces: • type MAIN[1] • type SUB, invoked by the `call_sub` function[4]	EQADDCXT and EQAD3CXT[5]
IMS TM[2] and BTS[3]	EQADICXT and EQAD3CXT
Batch	EQADBCXT and EQAD3CXT

Note:

1. EQADDCXT or EQAD3CXT is supported for DB2 version 7 or later. If DB2 RUNOPTS is specified, EQADDCXT or EQAD3CXT takes precedence over DB2 RUNOPTS.
2. For IMS TM, if you do not sign on to the IMS terminal, you might need to run the EQASET transaction with the `TSOID` option. For instructions on how to run the EQASET transaction, see "Debugging Language Environment IMS MPPs without issuing /SIGN ON" in the *Debug Tool User's Guide*.
3. 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`.
4. Link the user exit into a private copy of the Language Environment module CEEPIPI, not to your application program.

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 29.

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

1. "Editing the source code of CEEBXITA" on page 27.
2. "Linking the CEEBXITA user exit into a private copy of a Language Environment runtime module" on page 30.

To do the instructions in "Customizing for JCL for Batch Debugging utility" on page 39, you need the following information:

- If you change the naming pattern of the TEST runtime options data set, you need the new naming pattern you set in "Modifying the naming pattern" on page 27.
- 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 30.

To do the instructions in "Customizing Debug Tool User Exit Data Set" on page 50, you need the following information:

- If you change the naming pattern of the TEST runtime options data set, you need the new naming pattern you set in "Modifying the naming pattern."

To do the instructions in "Customizing IMS Transaction and User ID Cross Reference Table" on page 54, you need the following information:

- If you change the name of the cross reference table data set, you need the data set name you set in "Activate the cross reference function and modifying the cross reference table data set name" on page 29.

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
EQAD3CXT	EQAUMODK

- 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). However, if users create and manage the TEST runtime option data set with the **DTSP Profile** view in the remote debugger, do not specify the `&PGMNAME` token because the view does not support that token.

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`. For IMS, `&USERID` token might be substituted with one of the following values:

- IMS user ID, if users sign on to IMS.
- TSO user ID, if users do not sign on to IMS.

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
*
```

Modifying the call back routine registration

You can register a call back routine to the Language Environment. The Language Environment invokes the call back routine prior to calling a type SUB program using CALL_SUB API in the CEEPIPI environment. The call back routine performs a pattern match to determine if the type SUB program is to be debugged.

To customize the registration, change the value of the RRTN_SW DC statement.

x'00'
No registration of the call back routine.

x'01'
Registration of the call back routine.

Activate the cross reference function and modifying the cross reference table data set name

You can activate the cross reference function of the IMS Transaction and User ID Cross Reference Table and provide a cross reference table data set name. When an IMS transaction is initiated from the web or MQ gateway, it runs with a generic ID. If a user wants to debug the transaction, the cross reference function provides a way to associate the transaction with his or her user ID.

To customize the activation, change the value of the XRDSN_SW DC statement.

`x'00'`
Cross reference function is not activated.

`x'01'`
Cross reference function is activated.

To customize the cross reference table data set name, change the value of the XRDSN DC statement. You must provide a fully qualified MVS sequential data set name.

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
The following types of DB2 stored procedures that run in WLM-established address spaces: • type MAIN • type SUB, invoked by the `call_sub` function[1]	EQADDCXT or EQAD3CXT[2]	CEEPIPI
IMS TM and BTS	EQADICXT or EQAD3CXT	CEEBINIT
Batch	EQADBCXT or EQAD3CXT	CEEBINIT

Note:

1. This requires that you install the PTF for APAR PM15192 for Language Environment Version 1.10 to Version 1.12.

Edit and run sample *hlq*.SEQASAMP(EQAWLCEE or EQAWLCE3) 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.

Creating and managing the TEST runtime options data set

The TEST runtime options data set is an MVS data set that contains the Language Environment runtime options. The Debug Tool Language Environment user exits (EQADDCXT, EQADICXT, EQADBCXT, and EQAD3CXT) construct the name of this data set based on a naming pattern described in "Modifying the naming pattern" on page 27.

If your site does not allow your users to create data sets, you must create the data sets manually with the following requirements:

- Sequential data set (`DSORG=PS`)
- Record format and length requirements:

  - RECFM(F) or RECFM(FB) and LRECL >=80
  - RECFM(V) or RECFM(VB) and LRECL >=84
- Not an HFS data set
- Name the data set so it follows the naming pattern defined in “Modifying the naming pattern” on page 27.

Your users can then use any of the following ways to create the data set:
- By using Terminal Interface Manager (TIM), as described in “Creating and managing the TEST runtime options data set by using Terminal Interface Manager (TIM) ” in the *Debug Tool User's Guide*.
- By using Debug Tool Utilities option 6, "Debug Tool User Exit Data Set", as described in .
- By using the **DTSP Profile** view. To learn more about this view, see “Installing the IBM Debug Tool DTCN and DTSP Profile Manager” in the *Debug Tool User's Guide*.

Chapter 8. Installing the browse mode RACF facility

Debug Tool browse mode can be controlled by either the browse mode RACF facility, through the EQAOPTS `BROWSE` command, or both. For an overview of browse mode and how to control it, see "Debugging in browse mode" in the *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
- Restrict the use of Debug Tool to certain users

Note: If you have defined a generic Facility class profile (for example, *.*), you might have to install the browse mode RACF facilities described below, even if neither of the previous considerations apply. For example, if you have a generic Facility class profile of *.* with `UACC(NONE)` and you do not install the browse mode RACF facilities described below, no users would be allowed to use Debug Tool.

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 the chosen facilities.
2. Set up the default user access to the facility.
3. Authorize those users that need access other than the default access.

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

Related tasks
"BROWSE" on page 106

Choose and install appropriate RACF facility

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 (through *usr_acc* of READ or higher) can use Debug Tool in any way.

READ Only users specifically authorized to the facility (through *usr_acc* of UPDATE or higher) 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 the option specified for the EQAOPTS `BROWSE` command.

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)
   ```

Set up user access to facility

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 specify the EQAOPTS `BROWSE` command to specify that he wants the current invocation of Debug Tool to be in browse mode or normal mode.

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
   ```

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 Functions** to help users edit the TEST runtime options that IMS programs use and to create private message regions for testing.

  The functions include the IMS Transaction Isolation facility, which allows users to select IMS transactions to debug in a private message-processing region.

  These functions also allow users to manage IMS message region templates. Message region templates are superseded by the IMS Transaction Isolation facility, and will be removed in a future release of Debug Tool.
- **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 users 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.
- **IMS BTS Debugging** to help users start a debug session when they run their IMS BTS applications.
- **Delay Debug Profile** to create a delay debug profile data set that the Debug Tool delay debug pattern matching uses to start a debug session.
- **IMS Transaction and User ID Cross Reference Table** to create a cross reference table entry that Debug Tool uses to associate a user ID to an IMS transaction.
- **Non-CICS Debug Session Start and Stop Message Viewer** to browse debug session start and stop messages.
- **Debug Tool Code Coverage** to specify code coverage options, observation selections, and do viewing and reporting.
- **Debug Tool Deferred Breakpoints** to create and view a list of breakpoints prior to starting the debug session. It reduces the time spent on the debugging session and also the system resource usage.
- **Debug Tool JCL Wizard** is an ISPF edit macro (EQAJCL) that allows you to modify a JCL or procedure member and create statements to invoke Debug Tool in various environments.

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.
- Customize IMS BTS Debugging with default values for the TEST runtime option and data set names for the Debug Tool load module, user exit module, debug information files, IMS subsystem IDs, and base JCLs.
- Customize the Debug Tool User Exit Data Set utility.
- Customize the Delay Debug Profile utility.
- Customize the IMS Transaction and User ID Cross Reference Table utility.
- Customize the Non-CICS Debug Session Start and Stop Message Viewer utility.
- Customize Debug Tool Code Coverage.
- | Install and customize Debug Tool JCL Wizard.

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 and Common Component 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 on page 37, Table 7 on page 37, or Table 8 on page 37 into the DD concatenations specified in the tables.
   - Copy[4] the members of the data sets listed in Table 6 on page 37, Table 7 on page 37, or Table 8 on page 37 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 95.
2. Edit the `EQASTART`[5] 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 95.

5. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

3. Inform your users how to specify a language other than the one selected in step 2 on page 36. 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 names
SYSEXEC or SYSPROC	*hlq*.SEQAEXEC
ISPMLIB	*hlq*.SEQAMENU and *hlq*.SIPVMENU
ISPLLIB	*hlq*.SEQAMOD and *hlq*.SIPVMODA
ISPPLIB	*hlq*.SEQAPENU and *hlq*.SIPVPENU
ISPSLIB	*hlq*.SEQASENU
ISPTLIB	*hlq*.SEQATLIB and *hlq*.SIPVTENU

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

DD concatenation	Data set names
SYSEXEC or SYSPROC	*hlq*.SEQAEXEC
ISPMLIB	*hlq*.SEQAMENP and *hlq*.SIPVMENU
ISPLLIB	*hlq*.SEQAMOD and *hlq*.SIPVMODA
ISPPLIB	*hlq*.SEQAPENP and *hlq*.SIPVPENU
ISPSLIB	*hlq*.SEQASENP
ISPTLIB	*hlq*.SEQATLIBand *hlq*.SIPVTENU

Note: PD Tools Common Components does not have an upper English set of ISPF data sets.

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

DD concatenation	Data set names
SYSEXEC or SYSPROC	*hlq*.SEQAEXEC
ISPMLIB	*hlq*.SEQAMJPN and *hlq*.SIPVMJPN
ISPLLIB	*hlq*.SEQAMOD and *hlq*.SIPVMODA
ISPPLIB	*hlq*.SEQAPJPN and *hlq*.SIPVPJPN
ISPSLIB	*hlq*.SEQASJPN
ISPTLIB	*hlq*.SEQATLIB and *hlq*.SIPVTJPN

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

DD concatenation	Data set names
SYSEXEC or SYSPROC	*hlq*.SEQAEXEC
ISPMLIB	*hlq*.SEQAMKOR and *hlq*.SIPVMKOR
ISPLLIB	*hlq*.SEQAMOD and *hlq*.SIPVMODA
ISPPLIB	*hlq*.SEQAPKOR and *hlq*.SIPVPKOR
ISPSLIB	*hlq*.SEQASKOR

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

DD concatenation	Data set names
ISPTLIB	*hlq*.SEQATLIB and *hlq*.SIPVTKOR

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[6] 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'
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 DD names (as described in Method 2 in "Choosing a method to start Debug Tool Utilities" on page 36), 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

6. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

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[7] 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[8] 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.

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.

7. See “SMP/E USERMODs” in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

8. See “SMP/E USERMODs” in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

MFI
: dedicated terminal identified by network and LU names.

TIM
: terminal identified by user id.

GUI
: Remote debugger identified by IP address.

ybldtmtd
: 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 25.

  A User exit module EQADBCXT in application module.

ybldtprf
: Data set that contains a Debug Tool preferences file.

ybldtcmd
: Data set that contains a Debug Tool commands file.

ybldtbin
: 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 25.

ybldtnmp
: 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 25.

ybldtdfl
: 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`[9] 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*`)'`

9. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

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`[10] 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.
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:

10. See “SMP/E USERMODs” in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

- Modify EQASTART[11] 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[12] 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
   - ycclcpa1
   - ycchga1
   - 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);
```

11. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

12. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

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
6	Enterprise COBOL for z/OS Version 5 compiler 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.
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.

/M The member name of the 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`[13] 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.

Notes:

- The Coverage Utility cannot be installed on the same system that the IMS Transaction Isolation Facility of Debug Tool is installed on (see "Scenario F: Enabling the Transaction Isolation Facility" on page 90). If you need a method to gather code coverage, consider the Debug Tool Code Coverage feature that is described in Appendix E. Debug Tool Code Coverage of the *Debug Tool for z/OS User's Guide*.
- The monitor in this version of Coverage Utility is not compatible with the Coverage Utility monitor in Debug Tool Version 9 Release 1 and earlier. When the SVCs for this version of Coverage Utility are installed, monitor commands, including Start Monitor from Debug Tool Version 9 Release 1 and earlier, will not run.

13. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

- Ask your users to stop any monitor sessions that are running before you install the new monitor SVCs for this release (see "Installing and enabling the monitor SVCs" on page 46). The monitor SVC installer quits any monitor sessions that it finds active, and any data from those monitor sessions will be lost.

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[2] 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[4] 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

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	12616 bytes
ESQA	210 KB

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 45, 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    GYOUNG.MSGS.FILE                          Line 00000000 Col 001 080
    Command ===>                                                  Scroll ===> CSR
   ********************************* Top of Data **********************************
   Monitor Release: VDR1M0 Date: 2012.241
   MAST: 07EC7D00 PSA: 07BCB000 CPU: 07BCB000 SEST: 07E49820 UNID: 00000000
   ```

Customizing the Coverage Utility defaults

Complete the following steps to edit *hlq*.SEQAEXEC(EQACUDFT)[14]:

1. Change all occurrences of EQAW to *hlq*. For example, to use the high-level qualifier SYS2.EQAW, change all occurrences of EQAW to SYS2.EQAW.
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'
   - 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.
7. If your users routinely instrument the same subroutine for different Monitor sessions and run these subroutines and Monitor sessions concurrently, the instrumentation (Setup) of these subroutines should be done with the following options set:
   - Performance mode (Yes | No)

     All instrumentations (Setup) should use Yes or all should use No.

14. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

- Debug mode (Yes | No)

  All should use `Yes`.
- Frequency count mode (Yes | No)

  All should use `Yes`.

For more information about these options, see section "Parameters for the setup programs" in Chapter "Preparing to monitor a program" of the *Coverage Utility User's Guide and Messages*.

To set these values for new users, set the values in `EQACUDFT` as follows:

- CUPRFMD ='Y' or ='N'
- DEBGMODE='YY'

Note: Setting Debug Mode and Frequency count mode might significantly increase their processing time when running with DTCU.

`EQACUDFT` provides the initial settings for Coverage Utility users. If you are upgrading from an older version of Debug Tool or you have changed these initial settings, users should go to ISPF Debug Tool Utilities (DTU) option **3 - Code Coverage**, then option **0 - Defaults**, and then option **2 - RESET** to reset their options to those specified in `EQACUDFT`. To recustomize the options, users can then use Coverage Utility option **0 - Defaults** and then option **1 -EDIT**.

Configuring for IMSplex users

To determine if you need to do the steps described in this topic, read "Preparing IMS programs" in the *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`

Customizing debugging by using IMS message region templates

To determine whether you need to complete the steps described in this topic, read *Using IMS message region templates to dynamically swap transaction class and debug in a private message region* in the *Debug Tool User's Guide*. If you use the IMS TM Setup - Swap IMS Transaction Class and Run Transaction function in Debug Tool Utilities, you must complete the following tasks:

1. Edit the `EQAZDFLT`[15] member of the *hlq*.SEQATLIB data set.
2. Specify a default data set name to store IMS message region templates by setting the *imstmpds* parameter value. If the data set does not exist, allocate it with the following attributes:

15. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

- Data set type LIBRARY (PDSE).
- RECFM=VB, LRECL=1280, and BLKSIZE=27998.
- Not an HFS data set.

3. Specify default job names for the three jobs that the Dynamically Swap IMS Transaction Class and Run Transaction function will start, by setting the following parameter values. The string &&user&& is replaced with up to the first 7 characters of the Debug Tool Utilities user's TSO ID.

   mprdebug
   : Job name for the private message region that is started to debug the user's transaction. The default value for this parameter is @&&user&&.

   mprnodbg
   : Job name for the private message region that is started with the debug private message region with the NOTEST parameter. This region processes all messages for the selected transaction that are scheduled, when the initial message to the transaction is being debugged at the same time. This enables users to interrupt debugging to continue running the transaction.

     The default value for this parameter is #&&user&&.

   mprbmp
   : Job name for the batch job that is submitted to run the EQANBSWT Batch Message Processing (BMP) program. The default value for this parameter is $&&user&&.

4. Customize the data sets that the EQANBSWT program will use by setting the appropriate parameters. For a list of all of the parameters that you can specify, see section *Option 4: IMS TM Functions* in the EQAZDSYS member of *hlq*.SEQATLIB. You can set the following parameters:

   us5imrsl
   : Specify the IMS RESLIB load library when you start EQANBSWT. This is the first data set in the STEPLIB concatenation for the BMP. This parameter is required. The default value is IMS.SDFSRESL.

   us5dtmod
   : Specify the Debug Tool SEQAMOD load library when you start EQANBSWT. This is in the STEPLIB concatenation for the BMP. This parameter is optional.

   us5dtce1
   : Specify the Language Environment SCEERUN load library when you start EQANBSWT. This is the STEPLIB concatenation for the BMP. This parameter is optional.

   us5slb*nn*
   : Add DD cards to the STEPLIB concatenation for the EQANBSWT BMP job. The cards must be complete lines of JCL and will be displayed in the STEPLIB after the IMS RESLIB specification and before the Debug Tool load library and Language Environment load library, if specified.

     nn is a number in the range 01 - 09.

5. Complete the steps for "Scenario E: Enabling users to launch private message regions and to assign transactions to private message regions" on page 88.

Customizing Debug Tool User Exit Data Set

The Debug Tool User Exit Data Set utility creates a user exit data set that is used by the Debug Tool Language Environment user exit to start a debug session. You can set the default value of the data set naming pattern for the users by taking the following steps:

1. Review the parameter described in Table 10. Verify that you have all the information to specify the value for the parameter. See the EQAZDSYS member of the *hlq*.SEQATLIB data set for the parameter and the syntax convention.
2. Edit the EQAZDSYS[16] member of the *hlq*.SEQATLIB data set. Modify the parameter 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 *hlq*.SEQATLIB data set.

   If your application programmers use terminals that cannot display text in mixed-case English, enter parameters and their values in uppercase English.

See Chapter 7, "Specifying the TEST runtime options through the Language Environment user exit," on page 25 for how the user exit data set is used.

Table 10. Parameter for the Debug Tool User Exit Data Set option of Debug Tool Utilities

Name of parameter	Description
uepnmp	The naming pattern of the user exit data set. The utility builds a data set name by using the naming pattern that is used when the option is selected for the first time. The data set name and modification (if user modifies it) are consistent across the Debug Tool Utilities sessions. The following list describes the rules of using the token: • The &USERID token is replaced with the value from the SYSPREF or SYSUID system variable. • The &PGMNAME token is not supported. • If the parameter does not exist, the default naming pattern is used: &USERID.DBGTOOL.EQAUOPTS

16. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

Customizing IMS BTS Debugging

The IMS BTS Debugging utility of Debug Tool Utilities helps your users prepare a BTS JCL and start a debug session in the foreground or in batch. You can supply your users with default values for the TEST runtime option and data set names for the Debug Tool load module, user exit module, debug information files, IMS subsystem IDs and base JCLs.

To set the defaults, do the following steps:

1. Review the parameters described in Table 11. Verify that you have all the information you need to specify values for each parameter. You can also view a complete list of parameters and the syntax convention for these parameters in the `EQAZDSYS` member of the *`hlq`*`.SEQATLIB` data set.
2. Edit the `EQAZDSYS`[17] member of the *`hlq`*`.SEQATLIB` data set. Modify the 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`.

   If your application programmers use terminals that cannot display text in mixed-case English, enter all parameters and their values in uppercase English.

Table 11. Parameters you can define for the IMS BTS Debugging option of Debug Tool Utilities.

Name of parameter	Description
yb2dtmod	The name of the data set that contains the Debug Tool load modules, SEQAMOD.
yb2dtce1	The name of the data set that contains Language Environment runtime library, SCEERUN1.
yb2dtce2	The name of the data set that contains Language Environment runtime library, SCEERUN2.
yb2dtbin	The name of the data set that contains the CEEBINIT load module.
yb2dtnmp	The naming pattern you stored in EQADICXT when you completed the instructions in “Modifying the naming pattern” on page 27. If you modify the naming pattern stored in EQADICXT, you must modify this parameter to match.

17. See “SMP/E USERMODs” in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

Table 11. Parameters you can define for the IMS BTS Debugging option of Debug Tool Utilities. (continued)

Name of parameter	Description
yb2dtdev	The interface type that application programmers should use to debug IMS BTS programs. You can specify one of the following values: **MFI** Interact with Debug Tool in full-screen mode or full-screen mode using a dedicated terminal without Terminal Interface Manager (TIM). If application programmers are using full-screen mode using a dedicated terminal without Terminal Interface Manager, they identify the terminal by network and LU names, as described in Appendix B, "Enabling debugging in full-screen mode using a dedicated terminal," on page 149. **TIM** Interact with Debug Tool in full-screen mode or full-screen mode using the Terminal Interface Manager (TIM). The application programmer identifies the terminal by user ID, as described in "How users start a full-screen mode debug session with the Terminal Interface Manager" on page 17. Make sure you complete the instructions in "Enabling full-screen mode using the Terminal Interface Manager" on page 19. **GUI** Interact with Debug Tool in remote debug mode, where the application programmer identifies the remote debugger by IP address.
yb2dtmtd	The method that application programmers should use to start Debug Tool. You can specify one of the following values: **C** The application programmer specifies the CEEOPTS DD statement. You must run z/OS, Version 1.7, or later to use this option. **E** The application programmer specifies the EQADICXT user exit.
yb2dtprf	The name of the data set that contains the preferences file. If your site does not use a preferences files, you can leave this field blank.
yb2dtcmd	The name of the data set that contains the commands file. If your site does not use a commands files, you can leave this field blank.
yb2dtufl	The name of a data set containing a list of data set names to be allocated by EQADEBUG DD statements.
yb2dtued	The name of a data set containing the EQAUEDAT load module.
yb2imsnm	The number of IMS subsystems that the application programmers can use to run or debug IMS applications. The maximum value is 12.
yb2iid*n*	For each IMS subsystem, create a copy of this parameter and assign *n* a unique number between 1 and 12. For example, if your site has two IMS subsystems, you create `yb2iid1` and `yb2iid2` and assign each parameter a unique IMS system name.
yb2bmp*n*	For each IMS subsystem, create a copy of this parameter and assign *n* a unique number between 1 and 12 and specify the member name of a JCL that your site uses as a base or template JCL for batch message processing (BMP) programs. Edit the EQABMPSM[1] member of *hlq*.SEQATLIB, then copy it to a new name (for example, BMPJCL1). For example, if your site has two IMS subsystems, you create `yb2bmp1` and `yb2bmp2` and assign each parameter the member name.

Table 11. Parameters you can define for the IMS BTS Debugging option of Debug Tool Utilities. (continued)

Name of parameter	Description
yb2dbb*n*	For each IMS subsystem, create a copy of this parameter and assign *n* a unique number between 1 and 12 and the member name of a JCL that your site uses as a base or template JCL for Data Language/I (DL/I) programs. Edit the EQADBBSM[1] member of ***hlq***.SEQATLIB, then copy it to a new name (for example, DBBJCL1). For example, if your site has two IMS subsystems, you create yb2dbb1 and yb2dbb2 and assign each parameter the member name.
yb2dli*n*	For each IMS subsystem, create a copy of this parameter and assign *n* a unique number between 1 and 12 and the member name of a JCL that your site uses as a base or template JCL for Data Language/I (DL/I) programs. Edit the EQADLISM[1] member of ***hlq***.SEQATLIB, then copy it to a new name (for example, DLIJCL1). For example, if your site has two IMS subsystems, you create yb2dli1 and yb2dli2 and assign each parameter the member name.

Note:
1. See "SMP/E USERMODs" for an SMP/E USERMOD for this customization.

Customizing Delay Debug Profile

The Delay Debug Profile utility creates a delay debug profile data set that is used by the Debug Tool delay debug pattern matching to start a debug session. You can set the default value of the data set naming pattern for the users by taking the following steps:

1. Review the parameter described in Table 12 on page 54. Verify that you have all the information to specify the value for the parameter. See the EQAZDSYS member of the ***hlq***.SEQATLIB data set for the parameter and the syntax convention.
2. Edit the EQAZDSYS[18] member of the ***hlq***.SEQATLIB data set. Modify the parameter 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 ***hlq***.SEQATLIB data set.

   If your application programmers use terminals that cannot display text in mixed-case English, enter parameters and their values in uppercase English.

See "Using delay debug mode to delay starting of a debug session" in the *Debug Tool User's Guide* for how to use the delay debug function.

18. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

Table 12. Parameter for the Delay Debug Profile Data Set option of Debug Tool utilities

Name of parameter	Description
ddpnmp	The naming pattern of the delay debug profile data set. The utility builds a data set name by using the naming pattern that is used when the option is selected for the first time. The data set name and modification (if user modifies it) are consistent across Debug Tool Utilities sessions. The following list describes the rules of using the token: • The `&USERID` token is replaced with the value from the `SYSPREF` or `SYSUID` system variable. • The `&PGMNAME` token is not supported. • If the parameter does not exist, the default naming pattern is used: `&USERID.DLAYDBG.EQAUOPTS`

Customizing IMS Transaction and User ID Cross Reference Table

The IMS transaction and user ID cross reference table contains the cross reference information between IMS transactions and user IDs. A web or MQ gateway initiated IMS transaction is run with a generic ID. When a user wants to debug such a transaction, he needs to add an entry in the cross reference table that contains the transaction name and his user ID. Debug Tool then uses the table to find the user ID of the user who wants to debug the transaction and to construct the name of the user's debug profile data set. You can set the values of parameters for the users by taking the following steps:

1. Review the parameters described in Table 13 on page 55. Verify that you have all the information to specify the value for the parameters. See the `EQAZDSYS` member of the *hlq*.`SEQATLIB` data set for the parameter and the syntax convention.
2. Edit the `EQAZDSYS`[19] member of the *hlq*.`SEQATLIB` data set. Modify the parameters that are 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 *hlq*.`SEQATLIB` data set.

If your application programmers use terminals that cannot display text in mixed-case English, enter parameters and their values in uppercase English.

See Chapter 7, “Specifying the TEST runtime options through the Language Environment user exit,” on page 25 for information on how to set this data set name in the exit.

19. See “SMP/E USERMODs” in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

See “DLAYDBGXRF” on page 114 for information on how to specify this data set name for IMS users who are using delay debug mode.

Table 13. Parameters for the IMS Transaction and User ID Cross Reference Table of Debug Tool utilities

Name of parameter	Description
TUXRFDSN	The data set name for the cross reference table in ISPF format. No default is provided. The data set is an MVS sequential data set with FB LRECL 80. It must be pre-allocated and accessible to the users using the utility.
TUGNRCID	One or more generic IDs separated by a blank. No default is provided. A generic ID is an ID that is used to run IMS transactions started using the MQ or web gateway. Debug Tool uses the cross reference table to identify the user who wants to debug the transaction.
TUACTPRD	Number of days that a cross reference table entry is retained from the last update date. The default is 30 days.
TUGIDMAX	Maximum number of generic IDs. The default value is 20.
TUENTMAX	Maximum number of cross reference table entries. The default value is 200.

Customizing Non-CICS Debug Session Start and Stop Message Viewer

The Non-CICS Debug Session Start and Stop Messages Viewer utility allows users to browse debug session start and stop messages. It helps you track debug sessions. You can set the value of parameters for the users by taking the following steps:

1. Review the parameters that are described in Table 14 on page 56. Verify that you have all the information to specify the value for the parameters. See the `EQAZDSYS` member of the *hlq*.`SEQATLIB` data set for the parameter and the syntax convention.
2. Edit the `EQAZDSYS`[20] member of the *hlq*.`SEQATLIB` data set. Modify the parameters that are 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 *hlq*.`SEQATLIB` data set.

If your application programmers use terminals that cannot display text in mixed-case English, enter parameters and their values in uppercase English.

See “STARTSTOPMSGDSN” on page 130 for information on how to specify this data set name for debugger sessions.

20. See “SMP/E USERMODs” in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

Table 14. Parameters for the Non-CICS Debug Session Start and Stop Message Viewer of Debug Tool utilities

Name of parameter	**Description**
SSMSGDSN	The data set name for the debug session start and stop messages. No default is provided. The data set is an MVS sequential data set with FB LRECL 80. It must be pre-allocated and accessible to the users using the utility.

Customizing Debug Tool Code Coverage

Debug Tool Code Coverage provides the following functions:

1. Observation viewer - browse code coverage observation data set.
2. Debug Tool options - create or modify the Debug Tool code coverage options data set.
3. Observation selection criteria - create or modify the observation selection criteria and source markers data set.
4. Observation extraction - extract code coverage observations by using selection criteria.
5. Report generation - create reports.

You can set the default values for the data set naming pattern for the first three functions for the users by taking the following steps:

1. Review the parameter described in the following table. Verify that you have all the information to specify the value for the parameter. See the EQAZDSYS member of the *hlq*.SEQATLIB data set for the parameter and the syntax convention.
2. Edit the EQAZDSYS[21] member of the *hlq*.SEQATLIB data set. Modify the parameter 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 *hlq*.SEQATLIB data set.

If your application programmers use terminals that cannot display text in mixed-case English, enter parameters and their values in uppercase English.

See "Debug Tool Code Coverage" in the Debug Tool User's Guide for how to use the Debug Tool code coverage function.

21. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

Table 15. Parameters for Debug Tool Code Coverage

Name of parameter	Description
cconmp	The naming pattern of the code coverage options data set. The utility builds a data set name by using the naming pattern that is used when the option is selected for the first time. The data set name and modification (if user modifies it) are persistent across Debug Tool Utilities sessions. The following list describes the rules of using the token: • The `&USERID` token is replaced with the value from the SYSPREF or SYSUID system variable. • If the parameter does not exist, the default naming pattern is used: `&USERID.DBGTOOL.CCPRGSEL`.
ccsnmp	The naming pattern of the code coverage observation selection criteria data set. The utility builds a data set name by using the naming pattern that is used when the option is selected for the first time. The data set name and modification (if user modifies it) are persistent across Debug Tool Utilities sessions. The following list describes the rules of using the token: • The `&USERID` token is replaced with the value from the SYSPREF or SYSUID system variable. • If the parameter does not exist, the default naming pattern is used: `&USERID.DBGTOOL.CCOBSSEL`.
ccxnmp	The naming pattern of the code coverage observations data set. The utility builds a data set name by using the naming pattern that is used when the option is selected for the first time. The data set name and modification (if user modifies it) are persistent across Debug Tool Utilities sessions. The following list describes the rules of using the token: • The `&USERID` token is replaced with the value from the SYSPREF or SYSUID system variable. • If the parameter does not exist, the default naming pattern is used: `&USERID.DBGTOOL.CCOUTPUT`.

Installing and customizing the Debug Tool JCL Wizard

The Debug Tool JCL Wizard is an ISPF edit macro, **EQAJCL**, that can be used by a user to modify a JCL or procedure member to create statements that will invoke Debug Tool in various environments.

Prerequisites

The Debug Tool library *hlq*.SEQAMOD is assumed to be in the z/OS link list where the batch job will run. If it is not in the link list, do one of the following actions:

- Add the library *hlq*.SEQAMOD to the link list of z/OS LPARs where the modified JCL or procedure will be run.

- Add the library *hlq*.SEQAMOD to the //STEPLIB or //JOBLIB statement of the step or job that will be debugged.

Installation of the EQAJCL ISPF macro and its ISPF panels

The Debug Tool JCL Wizard contains an ISPF edit macro and a set of ISPF panels.

Use one of the following methods to install the Debug Tool JCL Wizard:

- Installation to libraries allocated to the TSO Logon procedure.

  Use one of these two methods:
  - Use Method 2 in “Choosing a method to start Debug Tool Utilities” on page 36.
  - Use a subset of Method 2 in “Choosing a method to start Debug Tool Utilities” on page 36 where you only include or copy *hlq*.SEQAEXEC into the SYSPROC or SYSEXEC DD.
- Installation by using a local REXX exec to point to the Debug Tool libraries.

  Select a command name that you do not currently use (for example, DEBUG), and install a REXX exec by that name into an existing data set in your TSO Logon procedure's SYSEXEC or SYSPROC DDs. The REXX exec should look like this (with *hlq* being changed to the high level qualifier that you use for the Debug Tool libraries):

```
/* This REXX exec will invoke the Debug Tool EQAJCL ISPF macro */
"EXEC 'hlq.SEQAEXEC(EQAJCL)'"
EXIT
```

Customizing the data set names and other values in EQAJCL

You must modify member **EQAJCL** of the *hlq*.SEQAEXEC data set to specify the data set names that you chose at installation time. Edit the **EQAJCL** member and follow the directions in the member's prologue for site customization of data set names. [22]

Enabling Code Coverage

Debug Tool Code Coverage measures test case code coverage in application programs that are written in COBOL, PL/I and C and compiled with certain compilers and compiler options. You must define the code coverage libraries xxxx.xxxx.CCPRGSEL and xxxx.xxxx.CCOUTPUT in the EQAOPTS member residing in *hlq*.SEQAMOD, or dynamically using an EQAOPTS DD statement. If the variable CODE_COVERAGE_SETUP is set to **YES**, the Debug Tool JCL Wizard automatically adds these statements to your JCL. Therefore, system programmers do not need to change the EQAOPTS member in *hlq*.SEQAMOD.

If the variable CODE_COVERAGE_SETUP is set to **YES**, the following statements are generated:

22. See “SMP/E USERMODs” in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

```
//EQAOPTS   DD *
         EQAXOPT CCOUTPUTDSN,'&&USERID.DBGTOOL.CCOUTPUT'
         EQAXOPT CCOUTPUTDSNALLOC,'MGMTCLAS(STANDARD)            +
               STORCLAS(DEFAULT) LRECL(255) BLKSIZE(0) RECFM(V,B) +
               DSORG(PS) SPACE(2,2) CYL'
         EQAXOPT  CCPROGSELECTDSN,'&&USERID.DBGTOOL.CCPRGSEL'
         EQAXOPT END
```

For more information about the compilation requirements for Code Coverage, refer to the *Debug Tool User's Guide and Reference*.

The Debug Tool JCL Wizard creates Code Coverage commands to run either with or without an interactive debug session.

Chapter 10. Preparing your environment to debug 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`[23] for Debug Tool
- `SCEERUN`[24] 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

23. 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 15, "EQAOPTS commands," on page 97) 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).

24. 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 25), 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. If you choose to use CADP, read the following topics for information on additional installation and setup tasks:
  - "The application debugging profile manager" in *Supplied Transaction*
  - "Preparing to use debuggers with CICS applications" in *Application Programming Guide*
  - "Setting up the debugging profiles data sets" in *System Definition Guide*

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 runtime libraries (`SCEECICS`, `SCEERUN`, and, if required by your applications, `SCEERUN2`) in the `DFHRPL` concatenation. Defining the *hlq*.`SEQAMOD` data set as a CICS LIBRARY resource is not supported. It must be included in the `DFHRPL` concatenation. If you are running COBOL V5.1 or later programs compiled with the TEST compiler option on CICS, you must also add system libraries MIGLIB and SIEAMIGE in the DFHRPL DD 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[2] the *hlq*.SEQAMOD data set and add the data set to the STEPLIB concatenation.
   - Copy[4] the EQA00DYN and EQA00HFS modules from the *hlq*.SEQAMOD data set to a library that is already in the STEPLIB concatenation.
   - Place *hlq*.SEQAMOD in the system link list and use the Authorized Program Facility (APF) to authorize it[2]. For more information, see Chapter 5, "Setting up the link list data set (SEQAMOD)," on page 15.

   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 68 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 and 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*.
   - Under CICS, Debug Tool can use a screen as large as 10922 characters (for example, 68x160 can be used, but not 69x160), 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 68 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 66 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 corequisite:
      - 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 on page 66 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 67.
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 on page 66 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. IBM recommends that you place EQA0CPLT after other PLT programs. 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" on page 67.

      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.

      STG
         This parameter enables the protection of storage that was GETMAINed in the current task by a program that is not the active program. This protection is only provided when the user is using the remote debugger.

      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 four 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.

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 66 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.

Migrating a debug profiles VSAM file from an earlier release

Debug Tool Version 12 increased the record size, and changes the format of the DTCN profile records.

If you are migrating from an earlier release of Debug Tool, and you use an `EQADPFMB` VSAM file to store your profiles, you need to create a new file using the JCL sample in member `EQAWCRVS` in the ***`hlq`***`.SEQASAMP` data set.

If you want to upgrade the records from an old DTCN VSAM file to the new record format, see the JCL sample in member `EQADPCNV` in the ***`hlq`***`.SEQASAMP` data set.

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 on page 69, 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), do the following steps:
  1. Designate a single FOR.
  2. Define the EQADPFMB file as REMOTE in the CICS FILE definition on regions that need to access it remotely. To learn how to define a FILE resource, see *CICS Resource Definition Guide*. Figure 2 on page 70 shows how to define the EQADPFMB file in a region that uses it remotely.
  3. For the region which owns the VSAM data set, omit the REMOTESYSTEM and REMOTENAME values in the EQADPFMB CICS FILE definition.
  4. Start the FOR before starting any AOR that needs to read the EQADPFMB file.

```
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
```

Figure 2. An example of how to define the EQADPFMB file as REMOTE in a CICS FILE definition.

- 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 68, 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:

```
►►──DTCD──┬─userid─┬──────────────────────────────────────────────►◄
          └─*──────┘
```

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:

```
►►──DTCI──┬─userid─┬──────────────────────────────────────────────►◄
          └─*──────┘
```

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 deactivate debug profiles" on page 82

Deleting DTCN profiles with the DTCN LINK service

Debug Tool provides a service that deletes unowned profiles from the DTCN repository.

If the DTCN repository is stored in CICS Temporary Storage (EQADTCN2), profiles are owned by the terminal that created them. The service scans the repository, looking for profiles that were created in the region running the service. If the service finds a profile owned by a terminal that is no longer defined and active in the region, the service deletes the profile.

If the DTCN repository is stored in VSAM (EQADPFMB), profiles are owned by the user ID that created them. The service scans the repository, looking for profiles that were created in the region running the service. If the service finds a profile owned by a user ID that is no longer active in the region, the service deletes the profile.

Invoke the service with the following command:

```
EXEC CICS LINK PROGRAM('EQADCDEL')
```

The service does not expect a commarea.

Invoke this service during DELETE processing in the program that controls autoinstall of terminals; however, you can invoke it from any EXEC-capable program. Figure 3 and Figure 4 show how to invoke the service in DFHZATDX, the supplied, user-replaceable autoinstall control program for terminals.

```
***********************************************************************
* *              D E L E T E   P R O C E S S I N G              * *
* *              ---------------------------------              * *
* *                                                             * *
***********************************************************************
DELETE_TERMINAL DS    0H
         USING DELETE_EXIT_COMMAREA,R2 Address delete commarea
* ==> PUT DELETE CODE HERE
*
         EXEC CICS LINK PROGRAM('EQADCDEL')
*
         B     RETURN              EXIT PROGRAM
```

Figure 3. Example of invoking service in DFHZATDX

```
***********************************************************************
* Function 8 and 10 - Common delete processing for shipped definitions*
***********************************************************************
DELETE_SHIPPED_TERMINAL DS 0H                                      @D2A
         USING DELETE_SHIPPED_COMMAREA,R2 Address commarea         @D2A
* ==> PUT DELETE CODE HERE                                         @D2A
*
         EXEC CICS LINK PROGRAM('EQADCDEL') NOHANDLE
*
         B     RETURN              EXIT PROGRAM                    @D2A
         DFHEJECT                                                  @D2A
```

Figure 4. Example of invoking service in DFHZATDX

Note: To use the the DTCN LINK service, ensure that the DTCNDELETEDEADPROF EQAOPTS command is set to YES.

See "DTCNDELETEDEADPROF" on page 116 for more information.

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 EQAOPTS DTCNFORCE*xxxx* command, as described in "DTCNFORCE*xxxx*" on page 116.

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 LangX 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 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)`
EQAWIVCT	Dynamic Debug facility and Enterprise COBOL for z/OS Version 5 TEST
EQAWIVC2	C `TEST(ALL)`
EQAWIVCG	C `DEBUG(FORMAT(DWARF),HOOK(LINE,NOBLOCK,PATH),SYMBOL)`
EQAWIVC8	Enterprise PL/I `TEST(ALL)`
EQAWIVCC	Non-Language Environment Assembler
EQAWIVCJ	LangX Enterprise COBOL

Configuring Debug Tool to run in a CICSplex environment

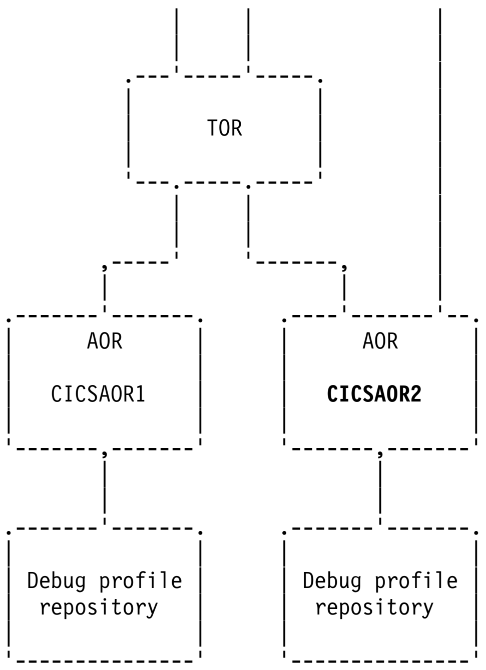
In a CICSplex, the application-owning regions (AORs), terminal-owning regions (TORs), queue-owning regions (QORs), repositories, and terminals can be organized in an infinite number of ways. In the following topics, we explore a finite number of scenarios and let you know what you need to do to configure 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 76
- "Terminal connects to a TOR which routes the application to an AOR; debugging profiles managed by DTCN" on page 77
- "Terminal connects to an AOR that runs an application that does not use a terminal" on page 78
- "Screen control mode terminal connects to a TOR and application runs in an AOR" on page 79
- "Separate terminal mode terminal connects to a TOR and application runs in an AOR" on page 80

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.

```
.---------------.    .---------------.    .---------------.
| 3270 terminal |    | 3270 terminal |    | 3270 terminal |
|               |    |               |    |               |
|     TRMA      |    |     TRMB      |    |     TRMC      |
'------,--------'    '--------,------'    '-,-------------'
       |                      |             |
       |                      |             |
       '-----------------,    |             |
```



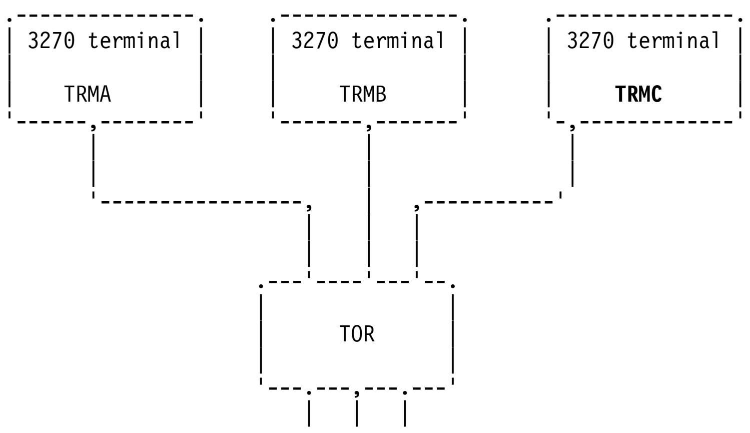
For this scenario to work, the CICS system administrator must complete the following tasks for the region CICSAOR2:

- 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 63, step 1.
- Provide access to these resources, as described in Chapter 11, "Adding support for debugging under CICS," on page 63, step 2a on page 63.

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.



```
               |     |     |
        ,------'     |     '------,
        |            |            |
.-------'-------.    |    .-------'--------.
|      AOR      |    |    |      AOR       |
|               |    |    |                |
|   CICSAOR1    |    |    |   CICSAOR2     |
|               |    |    |                |
'---------------'    |    '----------------'
        |            |            |
        |            |            |
        '------,     |     ,------'
               |     |     |
           .---'-----'-----'---.
           | CADP repository   |
           |      (VSAM)       |
           '-------------------'
```

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 63, step 1.
- Provide access to these resources, as described in Chapter 11, "Adding support for debugging under CICS," on page 63, step 2a on page 63.
- 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 63, step 5d on page 65, 6b on page 65, 9b on page 66, and "Enabling the CADP transaction" on page 74.

  DTCN
  Chapter 11, "Adding support for debugging under CICS," on page 63, step 5a on page 64 and 9b on page 66.

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.

```
.---------------.   .---------------.    .---------------.
| 3270 terminal |   | 3270 terminal |    | 3270 terminal |
|               |   |               |    |               |
|     TRMA      |   |     TRMB      |    |     TRMC      |
'------,--------'   '-------,-------'    '-,-------------'
       |                    |              |
       |                    |              |
       '----------------,   |   ,----------'
                        |   |   |
                        |   |   |
                    .---'---'---'--.
                    |              |
                    |     TOR      |
                    |              |
                    |              |
                    '---.---,---.--'
                        |   |   |
                        |   |   |
                 ,------'   |   '------,
         |          |          |
.--------'-------.  |  .-------'--------.
|      AOR       |  |  |      AOR       |
|                |  |  |                |
|   CICSAOR1     |  |  |   CICSAOR2     |
|                |  |  |                |
'----------------'  |  '----------------'
         |          |          |
         |          |          |
         '------,   |   ,------'
                |   |   |
           .----'---'---'----.
           |      QOR        |
           |   EQADTCN2      |
           '-----------------'
```

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 63, step 1.
- Provide access to these resources, as described in Chapter 11, “Adding support for debugging under CICS,” on page 63, step 2a on page 63.
- 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 68.

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.

```
                             .----------------.
                             | 3270 terminal  |
                             |                |
                             |     TRMC       |
                             '------,---------'
                                    |
                                    |
.-----------------------------------+------------------------------.
|  AOR                              |                              |
|                                   '---------------------,        |
|                                                         |        |
|  ,-----------------------------------------------,     ,------,  |
|  | Application runs, then Debug Tool issues    }-----{ CDT# |      |
|  | EXEC CICS START TRANS(CDT#) TERM(TRMC)      |     '------'      |
|  '--------------------------------------------'                   |
|                                                                   |
'-------------------------------------------------------------------'
                                   |
                                   |
                          .--------'-----------.
                          | Debugging profile  |
                          |    repository      |
                          '--------------------'
```

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 63, step 1 on page 63.
- Provide access to these resources, as described in Chapter 11, “Adding support for debugging under CICS,” on page 63, step 2a on page 63.
- 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 63, Chapter 11, “Adding support for debugging under CICS,” on page 63, step 5d on page 65, 6b on page 65, 9b on page 66, and “Enabling the CADP transaction” on page 74.

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

```
                           .---------------.
                           | 3270 terminal |
                           |               |
                           |     TRMC      |
                           '------,--------'
                                  |
                                  |
.---------------------------------+-----------------------------.
|  TOR                            |                             |
|                                 '---------------------,       |
|                                                       |       |
'-------------------------------------------------------+-------'
                                                        |
                                                        |
                          .--------------------.        |
                          | Debugging profile  |        |
                          |    repository      |        |
                          '-------,------------'        |
.---------------------------------'---------------------+-------.
|  AOR                                                  |       |
|                                                       |       |
|                                                       |       |
|  ,--------------------------------------------,   ,------,    |
|  | Application runs, then Debug Tool connects }-----{ CDT# |    |
|  | to the DTSC transaction that the user has  |   '------'    |
|  | started on the display terminal.           |               |
|  '--------------------------------------------'                   |
|                                                                   |
'-------------------------------------------------------------------'
```

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.

```
                               .----------------.
                               | 3270 terminal  |
                               |                |
                               |      TRMC      |
                               '------,---------'
                                      |
                                      |
.-------------------------------------+-----------------------------.
|  TOR                                |                             |
|                                     '---------------------,       |
|                                                           |       |
'-----------------------------------------------------------+-------'
                                                            |
                                                            |
                              .--------------------.        |
                              | Debugging profile  |        |
                              |     repository     |        |
                              '-------,------------'        |
.-------------------------------------'---------------------+-------.
|  AOR                                                      |       |
|                                                           |       |
|                                                           |       |
|  ,--------------------------------------------,     ,------,      |
|  | Application runs, then Debug Tool issues   }-----{ CDT# |      |
|  | EXEC CICS START TRANS(CDT#) TERM(TRMC)     |     '------'      |
|  '--------------------------------------------'                   |
|                                                                   |
'-------------------------------------------------------------------'
```

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 63, step 1 on page 63.
  - Provide access to these resources, as described in Chapter 11, "Adding support for debugging under CICS," on page 63, step 2a on page 63.

- 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 63, Chapter 11, “Adding support for debugging under CICS,” on page 63, step 5d on page 65, 6b on page 65, 9b on page 66, and “Enabling the CADP transaction” on page 74.
- 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 63, step 1 on page 63.
  - Provide access to these resources, as described in Chapter 11, “Adding support for debugging under CICS,” on page 63, step 2a on page 63.
- 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 63, Chapter 11, “Adding support for debugging under CICS,” on page 63, step 5d on page 65, 6b on page 65, 9b on page 66, and “Enabling the CADP transaction” on page 74.
- 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 deactivate 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 runtime options:
   - Specifying the TEST runtime 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 runtime 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 runtime options through the `EQASET` transaction. The transaction allows application programmers to specify a limited set of TEST runtime options.
   - Specifying the TEST runtime options in a private message region, created by the application programmer using Debug Tool Utilities option 4.3, "Swap IMS Transaction Class and Run Transaction".
   - Specifying the TEST runtime options in a private message region, created by the application programmer using Debug Tool Utilities option 4.5, "IMS Transaction Isolation Facility".
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 runtime options with a user exit. Do the steps described in "Scenario A: Running IMS and managing TEST runtime options with a user exit" on page 86 to enable this scenario.

   Scenario B
   You run programs in IMS Transaction Manager, BTS, or DB and are managing TEST runtime options with CEEUOPT or CEEROPT. Do the steps described in"Scenario B: Running IMS and managing TEST runtime options with CEEUOPT or CEEROPT" on page 87 to enable this scenario.

   Scenario C
   You run assembler programs without Language Environment in IMS Transaction Manager and you specify some TEST runtime options with the EQASET transaction. Do the steps described in "Scenario C: Running assembler program without Language Environment in IMS TM and managing TEST runtime options with EQASET" on page 87 to enable this scenario.

   Scenario D
   You run programs in an IMSplex environment and are managing TEST runtime options with either a user exit, CEEUOPT, or CEEROPT. Do the steps described in "Scenario D: Running IMSplex environment" on page 88 to enable this scenario.

Scenario E
You run Message Processing Programs (MPPs) in IMS Transaction Manager, running in Message Processing Regions (MPRs). You want to isolate application program debugging and to avoid scheduling delays or conflicts with programs which are not being debugged. Do the steps described in “Scenario E: Enabling users to launch private message regions and to assign transactions to private message regions” on page 88 to enable this scenario.

Scenario F
You run Message Processing Programs (MPPs) in IMS Transaction Manager, running in Message Processing Regions (MPRs). You want to isolate application program debugging and to avoid scheduling delays or conflicts with programs which are not being debugged. Do the steps described in “Scenario F: Enabling the Transaction Isolation Facility” on page 90 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 runtime 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 runtime options. If your application programmers are using the EQASET transaction to specify TEST runtime 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 runtime options with a user exit

Do the following steps to enable this scenario:

1. Include the Debug Tool *hlq*.SEQAMOD[25] data set and the Language Environment CEE.SCEERUN[26] runtime 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.
3. If you need to change the naming pattern of the data set containing the user's TEST runtime options, see the following topics for details:
   - Chapter 7, “Specifying the TEST runtime options through the Language Environment user exit,” on page 25
   - “Customizing Debug Tool User Exit Data Set” on page 50

   The user will use DTU option 'Debug Tool User Exit Data Set' to set the TEST runtime options they want.

25. 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 15, “EQAOPTS commands,” on page 97) 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).

26. 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 25), then place the data set containing it in STEPLIB (ahead of CEE.SCEERUN if it is in STEPLIB).

4. If the IMS transaction is initiated from the web or MQ gateway, it is run with a generic ID. Debug Tool supports a cross reference table to tie such a transaction to a user's ID. To set the name of that cross reference table, see the following topics for details:
   - Chapter 7, “Specifying the TEST runtime options through the Language Environment user exit,” on page 25
   - “Customizing IMS Transaction and User ID Cross Reference Table” on page 54

   The user will use DTU option 'IMS Transaction and User ID Cross Reference Table' to specify the transaction name to user ID cross reference.
5. See Chapter 7, “Specifying the TEST runtime options through the Language Environment user exit,” on page 25 and “Customizing Debug Tool User Exit Data Set” on page 50 for information about customizing the user exit (if needed).
6. If the IMS transaction is initiated from the web or MQ gateway, it is run with a generic ID. If your site has this situation, see “Activate the cross reference function and modifying the cross reference table data set name” on page 29 for information about customizing the user exit to enable a cross reference table and “Customizing IMS Transaction and User ID Cross Reference Table” on page 54 for setting up Debug Tool Utilities so that the user can access the table.

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

Do the following steps to enable this scenario:

1. Include the Debug Tool *hlq*.SEQAMOD[27] data set and the Language Environment CEE.SCEERUN runtime library in the STEPLIB concatenation of the IMS MPR or MPP region running your program.
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 runtime 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.

27. 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 15, “EQAOPTS commands,” on page 97) 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[28] data set in the STEPLIB concatenation of the IMS MPR or MPP region running your program.
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[29] data set and the Language Environment CEE.SCEERUN runtime library in the STEPLIB concatenation of the IMS MPR or MPP region running your program.
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.
3. If you are using a user exit and you need to change the naming pattern of the data set containing the user's TEST runtime options, see the following topics for details:
   - Chapter 7, “Specifying the TEST runtime options through the Language Environment user exit,” on page 25
   - “Customizing Debug Tool User Exit Data Set” on page 50

   The user can use DTU option 'Debug Tool User Exit Data Set' to set the TEST runtime options they want.

Scenario E: Enabling users to launch private message regions and to assign transactions to private message regions

Do the following steps to enable this scenario:

1. Define the following IMS batch message program:
```
APPLCTN GPSB=EQANBSWT,LANG=ASSEM,PGMTYPE=(BATCH),              X
        SCHDTYP=PARALLEL
TRANSACT CODE=(EQANBSWT),AOI=TRAN,                             X
        MSGTYPE=(SNGLSEG,NONRESPONSE)
```
2. Define the user EQANBSWT to RACF, and permit it READ access to the ASSIGN, DISPLAY, START and STOP IMS commands:
```
ADDUSER EQANBSWT NOPASSWORD DFLTGRP(SYS1)
PE ASS CLASS(CIMS) ID(EQANBSWT) ACC(UPDATE)
PE DIS CLASS(CIMS) ID(EQANBSWT) ACC(UPDATE)
PE STA CLASS(CIMS) ID(EQANBSWT) ACC(UPDATE)
PE STO CLASS(CIMS) ID(EQANBSWT) ACC(UPDATE)
SETROPTS RACLIST(CIMS) REFRESH
```
3. Enable users to create private message region templates. Users that are authorized for this function use Debug Tool Utilities option 4.4, "Manage IMS Transaction Templates" to create and edit the templates. The templates are stored in one or more Debug Tool Setup Utility data sets.

28. 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 15, “EQAOPTS commands,” on page 97) 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).

29. 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 15, “EQAOPTS commands,” on page 97) 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).

To control access to the Manage IMS Transaction Templates panel, create RACF FACILITY EQADTOOL.IMSTEMPCREATE with UACC(NONE). Then, PERMIT users who might access the panel ACC(READ) to the FACILITY.

An alternative approach is to grant UACC(READ) for EQADTOOL.IMSTEMPCREATE FACILITY. Then, access to the common IMS message region template data sets can be controlled by using RACF data set security.

4. Customize the Debug Tool Utilities ISPF interface for IMS TM Functions. See "Customizing debugging by using IMS message region templates" on page 48 for more information about this task.

After the above steps are completed, users who are authorized to EQADTOOL.IMSTEMPCREATE FACILITY can create one or more Debug Tool Setup Utility data sets that contain IMS message region templates. Application programmers use the message region templates to schedule individual transactions in private message regions. See section "Using IMS message region templates to dynamically swap transaction class and debug in a private message region" in the *Debug Tool User's Guide* for more information about how application programmers use the message region templates.

To create message region templates, complete the following steps:

1. Start Debug Tool Utilities. See "Starting Debug Tool Utilities" in the *Debug Tool User's Guide* for detailed information.
2. In the Debug Tool Utilities panel (EQA@PRIM), type 4 in the Option line and press Enter.
3. In the Manage IMS Programs panel (EQAPRIS), type 4 in the Option line and press Enter.
4. In the Create IMS MPR Templates panel (EQAPMPXS), type the name of the Debug Tool Setup Utility data set that you want to use in the Template Data Set field. Then, type an I in the Sel column of the table at the bottom of the panel, and press Enter.
5. In the Manage Message Regions - Edit Setup File panel (EQAPFORA), the Data Set Name is pre-filled from the data set name that was entered on the EQAPMPXS panel. Complete the data set name by entering a member name in parentheses, and press Enter.
6. In the Edit Setup File panel (EQAPMPRX), enter a description of the template in the Comment field. Then, type COPY on the command line and press Enter.
7. In the Debug Tool Foreground - Copy from Setup File or JCL panel (EQAPCPY), type the name of a data set that contains one of the following information:
   - Job Control Language (JCL) decks for IMS Message Processing Regions
   - Debug Tool Setup Utility (DTSU) files

   If you do not specify a member for a PDS, you can select from a member list on the ISRUDSM panel.
8. If you selected to copy from a JCL data set, the Debug Tool Foreground - Copy from JCL Dataset panel (EQAPCJ) is displayed. Select the job cards that you want to copy into the message region template, and press Enter.
9. After the JCL has been imported to your message region template, type a forward slash (/) beside Enter / to modify parameters, and press Enter.
10. In the Parameters for IMS Procedures panel (EQAPRIP1), change the Classes fields to specify a class that is reserved for application programmers by using Debug Tool. Then press PF3 to exit.

11. Adjust the data set list as needed and then press PF3 to exit. The message region template is saved.

Scenario F: Enabling the Transaction Isolation Facility

Note: The Coverage Utility cannot be installed on the same system that the IMS Transaction Isolation Facility of Debug Tool is installed on (see "Customizing Coverage Utility" on page 44). If you need a method to gather code coverage, consider the Debug Tool Code Coverage feature that is described in Appendix E. Debug Tool Code Coverage of the *Debug Tool for z/OS User's Guide*.

The IMS Transaction Isolation Facility of Debug Tool allows users to register to debug transactions in any IMS subsystem that is enabled for the facility. Each user may also launch a private message-processing region in which the selected message processing programs will run. The actions in this section will guide you through the setup of this facility.

The IMS Transaction Isolation Facility requires that the following resources be created or reserved for Debug Tool users:

1. A set of IMS message classes reserved for the private message-processing regions.
2. For each reserved message class, a set of IMS program resources with the name EQAT*cccn*, where *ccc* is the three-digit class number, and *n* is an ordinal number from 1 to the maximum number of transactions a single user can register to debug.

   Also, for each reserved message class, a set of IMS transaction resources must be defined. A set of non-conversational transactions with the names EQAT*cccn* and a set of conversational transactions with the names EQAC*cccn* must be defined, where *ccc* is the three-digit class number, and *n* is an ordinal number from 1 to the maximum number of transactions a single user can register to debug.
3. For the program resources defined above, a corresponding set of stub load modules must be created, with the names EQAT*cccn*.

Do the following steps to enable this scenario:

1. Define the following IMS batch message program:
   ```
   APPLCTN GPSB=EQANBSWT,LANG=ASSEM,PGMTYPE=(BATCH),                X
           SCHDTYP=PARALLEL
   TRANSACT CODE=(EQANBSWT),AOI=TRAN,                               X
           MSGTYPE=(SNGLSEG,NONRESPONSE)
   ```
2. Define the user EQANBSWT to RACF, and permit it READ access to the ASSIGN, DISPLAY, START and STOP IMS commands:
   ```
   ADDUSER EQANBSWT NOPASSWORD DFLTGRP(SYS1)
   PE ASS CLASS(CIMS) ID(EQANBSWT) ACC(UPDATE)
   PE DIS CLASS(CIMS) ID(EQANBSWT) ACC(UPDATE)
   PE STA CLASS(CIMS) ID(EQANBSWT) ACC(UPDATE)
   PE STO CLASS(CIMS) ID(EQANBSWT) ACC(UPDATE)
   SETROPTS RACLIST(CIMS) REFRESH
   ```
3. Enable administrative users to configure the IMS Transaction Isolation Utility. Users that are authorized for this function use Debug Tool Utilities option 4.6, "Administer IMS Transaction Isolation Environment", to reserve message classes for Debug Tool usage, and to generate other artifacts for the Transaction Isolation Utility.

To control access to the Administer IMS Transaction Isolation Environment panel, create RACF FACILITY EQADTOOL.IMSTRANISOADMIN with UACC(NONE). Then, PERMIT users who might access the panel ACC(READ) to the FACILITY.

4. Customize and run the EQAWTIVS sample member. This will create a VSAM data set that will be used as a repository to store IMS Transaction Isolation information when the IMS system is stopped.
5. Customize and run the EQAWTIMS sample member to link-edit the Debug Tool exits into an IMS system control region load library data set. Debug Tool implements the following exits for the IMS Transaction Isolation Facility:
   - DFSMSCE0 - TM and MSC Message Routing and Control User exit routine (as an alias of EQATIEXT)
   - EQATIEDT - Transaction Code (Input) edit routine for EQA**cccn* transactions.
6. Designate a library data set to contain the ACBs that will be generated for the EQAT*cccn* PSBs. If necessary, create this data set using the following characteristics:
   - DSORG=PO
   - RECFM=U
   - LRECL=0
   - BLKSIZE=32760
   - DSNTYPE=PDS
7. Customize the EQAOPTS sample and create the EQAOPTS load module.

   The EQAOPTS sample builds a set of EQAOPTS commands into a data-only load module. The EQAOPTS commands are described in detail in Chapter 15, “EQAOPTS commands,” on page 97.

   The EQAOPTS commands that apply to the IMS Transaction Isolation Facility are:

   DOPTACBDSN
   : This command specifies the data set which will contain the DOPT PSBs generated by the IMS Transaction Isolation Facility, as described in step 6.

   MAXTRANUSER
   : This command specifies the maximum number of transactions that a single user can register to debug. The default value is 15.
8. Ensure the following load libraries are in the search path for the IMS system control region:
   a. The load library with the Debug Tool user exits
   b. The library which contains the EQAOPTS load module

   Add the DOPT ACB library data set created in step 6 to the IMSACB* concatenation in the DLI and IMS address space JCL decks.

   Add the VSAM data set created in step 4 to the IMS system control region using DD name EQATIVSM.

   Then, start (or restart) the IMS system.
9. Use Debug Tool Utilities option 4, suboption 6 "Administer IMS Transaction Isolation Environment" to reserve the message classes and generate the program stubs and the IMS resource definitions.
10. Ensure that the data set you selected for the generated program stubs in step 9 is in the search path for any IMS message-processing region that will be cloned as a private IMS Transaction Isolation region.

11. Use the resource definitions from step 9 to update the IMS system definition, preferably using the Type 2 commands to update the system dynamically, if you are using Dynamic Resource Definition (DRD). Otherwise, add the Type 1 commands to the stage 1 input to your system definition process, and regenerate the IMS system definition. Restart the IMS system, if necessary, to pick up the changes.
12. Add any important IMS PROCLIBs to EQAZPROC in *hlq*.SEQATLIB. This allows Debug Tool Utilities to expand any IMS PROCs that are discovered when cloning the execution environment of the selected transaction.
13. Perform the following RACF customization:
    - Allow users of the IMS Transaction Isolation Facility to add members to and edit members in the DOPT ACBLIB defined in step 6.
    - Give the user ID for each IMS control region authority to modify the VSAM data set created in step 4.

See the section "Using IMS Transaction Isolation to create a private message-processing region and select transactions to debug" in the *Debug Tool User's Guide* for more information about how application programmers use the IMS Transaction Isolation Facility.

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.

The provided samples are designed to operate only under Language Environment. If you require an exit to run at any time in a non-Language Environment environment, you must write the exit in assembler and 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 Language Environment is not active, the exit will not be started.

Debug Tool provides two samples: `EQAUEDAT`, which is written in Language Environment-enabled assembler, and `EQAUEDAC`, which is written in Enterprise COBOL for z/OS and OS/390. Both samples generate a load module named `EQAUEDAT`.

To enable this user exit, do the following steps:

1. Copy either the `EQAUEDAT`[30] or `EQAUEDAC`[31] 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 library 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`.

30. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

31. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

Chapter 14. 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 96 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[32] 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.

32. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* 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 on page 95 and 4 on page 95.

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[33], EQACUIDM[34], 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 95.

33. See “SMP/E USERMODs” in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

34. See “SMP/E USERMODs” in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

Chapter 15. EQAOPTS commands

EQAOPTS commands are commands that alter some of the basic behavior of Debug Tool. These commands must be processed before normal Debug Tool command processing is available. You can specify most EQAOPTS commands in the following ways:

- Add dynamically at run time, as described in "Providing EQAOPTS commands at run time" on page 104, a text data set that contains the commands.
- Add to the search sequence, before the copy of EQAOPTS distributed by Debug Tool, a customized version of the EQAOPTS load module.

If you want the commands to apply to only a few debugging sessions, it might be easier to supply the EQAOPTS command dynamically at run time. If you want the commands to apply to a group of debugging sessions, it might be better to supply the EQAOPTS commands through the EQAOPTS load module.

Except for commands that can be validly specified more than once (for example, the NAMES commands), if Debug Tool finds a command more than once, it uses the first specification of the command. Debug Tool processes EQAOPTS commands specified at run time before those specified through the EQAOPTS load module. This means that commands specified at run time override duplicate commands specified in the EQAOPTS load module.

Any or all of the following people can create EQAOPTS specifications:

- The system programmer that installs Debug Tool.
- Specific groups in the organization.
- An individual user.

If you are the system programmer or you are creating EQAOPTS specifications for specific groups, you might change the EQAOPTS specifications less frequently, so specifying them by generating a new EQAOPTS load module might be more efficient. If you are an individual user, you might change the EQAOPTS specifications more frequently, so specifying them dynamically at run time might be more efficient.

Table 16 summarizes the available EQAOPTS commands and indicates whether a system programmer (S), a specific group (G), or an individual user (U) most commonly uses a command.

Table 16. A brief description of each EQAOPTS command and the type of user most likely to use that command

Command	Description	Commonly used by
ALTDISP	Controls whether to add a character indicator to the MFI screen to indicate a breakpoint, the current line, or the line with found text.	S, U
BROWSE	Allows users with the authority to use Debug Tool in normal mode to restrict their access to Browse Mode.	U
CACHENUM	Controls the size of the Debug Tool cache to minimize rereading the debug information.	U, G
CCOUTPUTDSN	Specifies the default naming pattern that Debug Tool uses to name the Code Coverage Observation file.	U, G, S

Table 16. A brief description of each EQAOPTS command and the type of user most likely to use that command (continued)

Command	Description	Commonly used by
CCOUTPUTDSNALLOC	Specifies the allocation parameters that Debug Tool uses when it creates the Code Coverage Observation file.	U, G, S
CCPROGSELECTDSN	Specifies the default naming pattern that Debug Tool uses to name the Code Coverage Options file.	U, G, S
CODEPAGE	Controls the codepage used by Debug Tool.	U, G, S
COMMANDSDSN	Specifies the default naming pattern that Debug Tool uses to name the user's commands file.	U, G, S
DEFAULTVIEW	Controls the default view of assembler programs.	U, G
DLAYDBG	Allows users to use delay debug mode.	U, G, S
DLAYDBGCND	Specifies monitoring condition events in the delay debug mode.	U, G, S
DLAYDBGDSN	Specifies delay debug profile data set naming pattern.	U, G, S
DLAYDBGTRC	Specifies delay debug pattern match trace message level.	U, G, S
DLAYDBGXRF	Specifies that Debug Tool uses a cross reference to find the user ID when Debug Tool constructs the delay debug profile data set name. This is used when an IMS transaction or DB/2 stored procedure is initiated from the web or MQ gateway, and thus the transaction is run with a generic ID. Debug Tool uses either the cross reference file or the Terminal Interface Manager repository to find the ID of the user who wants to debug the transaction or stored procedure.	U, G, S
DOPTACBDSN	Specifies the data set which will contain DOPT PSBs generated by the IMS Transaction Isolation Facility.	S
DTCNDELETEDEADPROF	Controls the deletion of dead DTCN profiles.	S
DTCNFORCE*xxxx*	Controls whether to require certain fields in DTCN.	S
DYNDEBUG	Controls the initial (default) value of SET DYNDEBUG.	U, G, S
EQAQPP	Enables Debug Tool to debug MasterCraft Q++ programs.	U, G, S
EXPLICITDEBUG	Enables explicit debug mode.	U
GPFDSN	Specifies that Debug Tool process a global preferences file.	U, G, S
HOSTPORTS	Specifies a host port or range of ports to use for a TCP/IP connection to the workstation for the remote debugger.	S
IGNOREODOLIMIT	Specifies that Debug Tool can display COBOL table data items even when an ODO value is out of range.	U, G, S
LOGDSN	Specifies the default naming pattern that Debug Tool uses to name the user's log file.	U, G, S
LOGDSNALLOC	Specifies the allocation parameters that Debug Tool uses when it creates the log file.	U, G, S
MAXTRANUSER	Specifies the maximum number of IMS transactions that a single user may register to debug using the IMS Transaction Isolation Facility.	S
MDBG	Allows users of programs compiled with z/OS XL C/C++ Version 1.10, or later, to indicate whether Debug Tool searches for .mdbg files.	U, G
MULTIPROCESS	Controls the behavior of Debug Tool when a new POSIX process is created by fork() or exec().	U, G, S

Table 16. A brief description of each EQAOPTS command and the type of user most likely to use that command (continued)

Command	**Description**	**Commonly used by**
NAMES	Controls whether Debug Tool processes or ignores certain load module or compile unit names.	U, G
NODISPLAY	Controls the Debug Tool behavior when the display requested by the Debug Tool user is not available.	U, G, S
PREFERENCESDSN	Specifies the default naming pattern that Debug Tool uses to name the preferences file.	U, G, S
SAVEBPDSN, SAVESETDSN	Specifies the default naming pattern for the data sets used to save and restore the breakpoints and monitors (SAVEBPS) and the settings (SAVESETS).	U, G, S
SAVEBPDSNALLOC, SAVESETDSNALLOC	Specifies the allocation parameters that Debug Tool uses when it creates the SAVEBPS and SAVESETS data sets.	U, G, S
SESSIONTIMEOUT	Establishes a timeout for idle Debug Tool sessions that use the Terminal Interface Manager. Timed out sessions are canceled after a specified period of no user activity.	S
STARTSTOPMSG	Controls whether to issue a message when each debugging session is initiated or terminated.	S
STARTSTOPMSGDSN	Specifies a message file for start and stop debug session messages.	S
SUBSYS	Specifies a subsystem used by certain library systems.	G, S
SVCSCREEN	Controls whether and how Debug Tool uses SVC screening to intercept LOAD and LINK SVC's. This is necessary for debugging non-Language Environment assembler and LangX COBOL programs.	S
TCPIPDATADSN	Instructs Debug Tool to dynamically allocate the specified file-name to the DDNAME SYSTCPD for the TCP/IP connection to the workstation for the remote debugger.	S
THREADTERMCOND	Controls whether Debug Tool prompts the user when it encounters a FINISH, enclave termination, or thread termination condition.	U, G
TIMACB	Specifies that the Debug Tool Terminal Interface Manager (TIM) use a name other than EQASESSM.	S
END	Specifies the end of a list of EQAOPTS commands. You must specify END.	U, G, S

Use the following list to help you record the commands and value you want to implement:

Table 17. Checklist you can print to record which EQAOPTS commands you selected and the values to use for each command.

- `EQAXOPT ALTDISP`, then select one of the following options:

  `ON`

  `OFF`

- `EQAXOPT BROWSE`, then select one of the following options:

  `RACF`

  `ON`

  `OFF`

Table 17. Checklist you can print to record which EQAOPTS commands you selected and the values to use for each command. (continued)

- EQAXOPT CACHENUM,*number:*______________________________
- EQAXOPT CCOUTPUTDSN,'*file_name_pattern:*______________________'

  Append ,LOUD if you want Debug Tool to display WTO messages, which helps you debug processing done by this command.
- EQAXOPT CCOUTPUTDSNALLOC,*allocation_parameters:*______________________

  Append ,LOUD if you want Debug Tool to display WTO messages, which helps you debug processing done by this command.
- EQAXOPT CCPROGSELECTDSN,'*file_name_pattern:*______________________'

  Append ,LOUD if you want Debug Tool to display WTO messages, which helps you debug processing done by this command.
- EQAXOPT CODEPAGE,*code_page_number:*_________________________
- EQAXOPT COMMANDSDSN,'*file_name_pattern:*______________________'

  Append ,LOUD if you want Debug Tool to display WTO messages, which helps you debug processing done by this command.
- EQAXOPT DEFAULTVIEW, then select one of the following options:

  STANDARD

  NOMACGEN
- EQAXOPT DLAYDBG, then select one of the following options:

  YES

  NO
- EQAXOPT DLAYDBGCND, then select one of the following options:

  YES

  NO
- EQAXOPT DLAYDBGDSN,'*file_name_pattern:*___________________________'
- EQAXOPT DLAYDBGTRC,*pattern_match_trace_level:*_________________
- EQAXOPT DLAYDBGXRF, then select one of the following options:

  DSN,'*file_name:*____________'

  REPOSITORY
- | EQAXOPT DOPTACBDSN,'*file_name:*____________'
- EQAXOPT DTCNDELETEDEADPROF, then select one of the following options:

  YES

  NO
- 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

Table 17. Checklist you can print to record which EQAOPTS commands you selected and the values to use for each command. (continued)

- `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 DYNDEBUG`, then select one of the following options:

  `ON`

  `OFF`
- `EQAXOPT EQAQPP`, then select one of the following options:

  `ON`

  `OFF`
- `EQAXOPT EXPLICITDEBUG`, then select one of the following options:

  `ON`

  `OFF`
- `EQAXOPT GPFDSN,'`*file_name:*____'
- `EQAXOPT HOSTPORTS,`*range_of_ports:*_____
- | `EQAXOPT IGNOREODOLIMIT`, then select one of the following options:

  | `YES`

  | `NO`
- `EQAXOPT LOGDSN,'`*file_name_pattern:*_______________________'

  Append `,LOUD` if you want Debug Tool to display WTO messages, which helps you debug processing done by this command.
- `EQAXOPT LOGDSNALLOC,`*allocation_parameters:*_______________________

  Append `,LOUD` if you want Debug Tool to display WTO messages, which helps you debug processing done by this command.
- | `EQAXOPT MAXTRANUSER,`*number:*_____________
- `EQAXOPT MDBG`, then select one of the following options:

  `YES`

  `NO`
- `EQAXOPT MULTIPROCESS`, then select one of the following options:

  `PARENT`

  `CHILD`

  `PROMPT`

  Select one of the following options to indicate what you want Debug Tool to do with a process that executes itself:

  `EXEC=ANY`

  `EXEC=NONE`

Table 17. Checklist you can print to record which EQAOPTS commands you selected and the values to use for each command. (continued)

- 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 PREFERENCESDSN,'*file_name_pattern:*_______________________'

  Append ,LOUD if you want Debug Tool to display WTO messages, which helps you debug processing done by this command.
- EQAXOPT SAVEBPDSN,'*file_name_pattern:*____________________________________'
- EQAXOPT SAVESETDSN,'*file_name_pattern:*___________________________________'
- EQAXOPT SAVEBPDSNALLOC,*allocation_parameters:*_______________________

  Append ,LOUD if you want Debug Tool to display WTO messages, which helps you debug processing done by this command.
- EQAXOPT SAVESETDSNALLOC,*allocation_parameters:*_______________________

  Append ,LOUD if you want Debug Tool to display WTO messages, which helps you debug processing done by this command.
- EQAXOPT SESSIONTIMEOUT, then select one of the following options:

  NEVER

  QUITDEBUG,*hhmmssnn*

  QUIT,*hhmmssnn*
- EQAXOPT STARTSTOPMSG, then select one of the following options:

  NONE

  ALL

  CICS

  TSO

  BATCHTSO

  IMS

  OTHER

  or any of CICS, TSO, BATCHTSO, IMS, and OTHER, or all of them enclosed in parenthesis and separated by commas.

  Append ,WTO if you want Debug Tool to display the messages via WTO.
- EQAXOPT STARTSTOPMSGDSN,'*file_name:*_______________'

  Append ,LOUD if you want Debug Tool to display WTO messages, which helps you debug processing done by this command.
- EQAXOPT SUBSYS,*subsystem_name:*____________________________

Table 17. Checklist you can print to record which EQAOPTS commands you selected and the values to use for each command. (continued)

- `EQAXOPT SVCSCREEN`, then select one of the following options:

  `ON`

  `OFF`

  `(OFF,QUIET)`

  Select one of the following options to indicate what you want Debug Tool to do if there is an existing SVC screening environment:

  `CONFLICT=OVERRIDE`

  `CONFLICT=NOOVERIDE`

  Select one of the following options to indicate whether you want Debug Tool to temporarily replace the existing SVC screening environment:

  `NOMERGE`

  `MERGE=(COPE)`

- `TCPIPDATADSN`,*'file_name: _____'*
- `EQAXOPT THREADTERMCOND`, then select one of the following options:

  `PROMPT`

  `NOPROMPT`

`EQAXOPT TIMACB,`*`ACB_name:`*______________________________________

`EQAXOPT END` Always specify this command.

After you have made all of you selections, define the options as described in "Creating EQAOPTS load module" on page 105.

Format of the EQAOPTS command

When you specify EQAOPTS commands through the EQAOPTS load module, you create them as assembler macro invocations and you must subsequently assemble and link-edit them into the EQAOPTS load module. To provide a consistent format for all forms of EQAOPTS commands, when you specify the EQAOPTS commands at run time, you must use the assembler macro invocation format. The following format rules apply to all EQAOPTS commands:

- EQAOPTS commands must be contained in fixed-length, eighty-byte records.
- The commands must be contained between columns one and seventy-one, with column seventy-two reserved for a continuation indicator. Debug Tool ignores columns seventy-three through eighty.
- Specify an asterisk (*) in column one to indicate a comment. Debug Tool ignores comments. Column one must be blank for all non-comment statements.
- The op-code for each EQAOPTS statement must be EQAXOPT and must begin in or after column two and followed by at least one blank.
- A list of one or more operands must follow the EQAXOPT op-code. Separate these operands by a comma and do not embed blanks.
- If a command exceeds the length of one line, you can continue the command in one of the following ways:
  - You can end at the comma following an operand and place a non-blank character in column seventy-two.
  - You can use all of the columns through column seventy-one and place a non-blank character in column seventy-two.

In either case, the statement that follows must be blank in columns one through fifteen and begin in column sixteen.

EQAOPTS commands that have equivalent Debug Tool commands

Some EQAOPTS commands have equivalent Debug Tool commands. Table 18 shows a few examples.

Table 18. Examples of EQAOPTS commands and their equivalent Debug Tool commands

EQAOPTS command	Debug Tool command
DEFAULTVIEW	`SET DEFAULTVIEW`
DYNDEBUG	`SET DYNDEBUG`
EXPLICITDEBUG	`SET EXPLICITDEBUG`
NAMES	`NAMES`

For these commands, specifying them as EQAOPTS commands or Debug Tool commands produces the same action. The timing (when these commands take effect) differs between EQAOPTS commands and Debug Tool commands.

Debug Tool processes Debug Tool commands after it processes the initial load module and creates the compile units contained in the initial load modules. Debug Tool processes EQAOPTS commands during Debug Tool initialization, prior to processing the initial load module. This means that when Debug Tool processes the initial load module, Debug Tool commands like `NAMES` are not in effect but the corresponding EQAOPTS commands are in effect and are applied to the initial load module.

EQAOPTS commands like `DEFAULTVIEW` provide a way of specifying a site- or group-wide default for the corresponding Debug Tool command. However, a better way to specify a site- or group-wide default for these types of commands is by putting the Debug Tool command in a global preferences file.

Providing EQAOPTS commands at run time

You can provide EQAOPTS commands to Debug Tool at run time. You must save the commands in a data set with 80-byte, fixed-length records. The following list describes the methods of specifying this data set to Debug Tool:

- In CICS, include the EQAOPTS commands through DTCN.
- In UNIX System Services, specify the name of the data set containing the EQAOPTS commands through the EQA_OPTS_DSN environment variable.
- In IMS and DB2, specify the name of the data set containing the EQAOPTS commands through the Debug Tool Language Environment user exit.
- In other environments, specify the name of the data set containing the commands through the EQAOPTS DD statement.

The following example shows what the data set might contain:

```
EQAXOPT MDBG,YES
EQAXOPT NODISPLAY,QUITDEBUG
EQAXOPT NAMES,EXCLUDE,LOADMOD,USERMOD1
EQAXOPT NAMES,EXCLUDE,LOADMOD,USERMOD7
EQAXOPT END
```

The instructions in "Creating EQAOPTS load module" contain examples with specifications for CSECT, AMODE, RMODE, and END (without EQAXOPTS) statements. Do not include these specifications if you provide EQAOPTS command at run time.

Creating EQAOPTS load module

If you have chosen to use the EQAOPTS load module to specify your EQAOPTS commands, do the following steps:

1. Copy the EQAOPTS[35] member from the *hlq*.SEQASAMP library to a private library.
2. Edit this copy of EQAOPTS and code the EQAOPTS command or commands you want. To this minimum source, add each EQAXOPT option you want to include. The following example 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.  For example:
            EQAXOPT MDBG,YES
            EQAXOPT NODISPLAY,QUITDEBUG
            EQAXOPT NAMES,EXCLUDE,LOADMOD,USERMOD1
            EQAXOPT NAMES,EXCLUDE,LOADMOD,USERMOD7
            EQAXOPT END
            END ,
   ```
3. Follow the directions in the EQAOPTS sample 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.

Descriptions of EQAOPTS commands

To learn how EQAOPTS commands work and how to specify them, see Chapter 15, "EQAOPTS commands," on page 97.

ALTDISP

You can use the EQAOPTS ALTDISP command to add a character indicator to the MFI screen to indicate a breakpoint, the current line, or the line with found text. By default, Debug Tool uses coloring to indicate these situations.

Use this command only if your 3270 color configuration and attributes make it difficult to detect the coloring in the line. It is valid only when you are using interactive MFI mode.

The following diagram describes the syntax of the ALTDISP command:

```
►►──EQAXOPT──ALTDISP──,──┬─ON──┬──────────────────────────────────────►◄
                         └─OFF─┘
```

The following list describes the parameters of the EQAOPTS ALTDISP command:

35. See "SMP/E USERMODs" in the *Debug Tool Customization Guide* for an SMP/E USERMOD for this customization.

ON Indicates to add a character indicator to indicate a breakpoint, the current line, or the line with found text.

OFF
Indicates not to add a character indicator to indicate a breakpoint, the current line, or the line with found text. This is the default value.

Example

```
EQAXOPT ALTDISP,ON
```

BROWSE

Debug Tool browse mode can be controlled by either the browse mode RACF facility, through the EQAOPTS `BROWSE` command, or both. For a description of how to control browse mode through RACF, see "Debugging in browse mode" in the *Debug Tool User's Guide*.

Users who have sufficient RACF authority can specify the EQAOPTS `BROWSE` command to indicate that the current invocation of Debug Tool be in browse mode.

The following diagram describes the syntax of the `BROWSE` command:

```
►►──EQAXOPT──BROWSE──,──┬─RACF─┬──────────────────────────────────────────►◄
                        ├─ON───┤
                        └─OFF──┘
```

The following list describes the parameters of the EQAOPTS `BROWSE` command:

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 you do not specify the `BROWSE` command, Debug Tool defaults to RACF.

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.

Examples

```
EQAXOPT BROWSE,ON
EQAXOPT BROWSE,RACF
```

CACHENUM

To reduce CPU consumption, Debug Tool stores information about the application programs being debugged in a cache. By default, for each debug session, Debug Tool stores the information for a maximum of 10 programs. Application programs that do a LINK, LOAD, 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 debugging session.

The following diagram describes the syntax of the CACHENUM command:

```
►►──EQAXOPT──CACHENUM──,──cache_value──────────────────────────────────►◄
```

cache_value
: Specifies the size of the Debug Tool cache. It must be no smaller than 10 and no larger then 999.

Example

```
EQAXOPT CACHENUM,40
```

CCOUTPUTDSN

This option provides the data set name to be used for the Code Coverage Observation file. Specify `NULLFILE` if no Observation file is to be written to.

This data set must be preallocated as a sequential data set if `CCOUTPUTDSNALLOC` is not specified. `RECFM=VB, LRECL=255` is suggested.

The following diagram describes the syntax of the `CCOUTPUTDSN` command:

```
►►──EQAXOPT──CCOUTPUTDSN──,──'──file_name_pattern──'──┬─────────┬──────►◄
                                                      └─,──LOUD─┘
```

file_name_pattern
: Specifies a naming pattern that determines the name of the data set that contains this file. Follow these guidelines when you create the naming pattern:

- Create a 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 determines the name of the data set.
- Specify `NULLFILE` to indicate that you do not want Debug Tool to process this file.

LOUD
: Specifies that Debug Tool displays WTO messages, which helps you debug processing done by this command. Debug Tool normally does not display any messages if it does not find the data set. If you are trying to determine why Debug Tool is not processing this file, specify LOUD to see if it displays a message that it can not find the data set.

CCOUTPUTDSNALLOC

This option is used to create the `CCOUTPUTDSN` data set for a new user and provides the allocation parameters (in BPXWDYN format).

The following diagram describes the syntax of the `CCOUTPUTDSNALLOC` command:

```
►►──EQAXOPT──CCOUTPUTDSNALLOC──,──'──allocation_parms──'──┬─────────┬──►◄
                                                          └─,──LOUD─┘
```

allocation_parms
: Specifies the allocation parameters you want Debug Tool to use when it creates the data set. You can specify only the keys in the following list:

- BLKSIZE
- BLOCK
- CYL

- DATACLAS
- DSNTYPE
- DSORG
- LRECL
- MGMTCLAS
- RECFM
- SPACE
- STORCLAS
- TRACKS
- UNIT
- VOL

Separate the keys by one or more blanks. Debug Tool does not provide defaults for any of the keys.

For information about the format of the keys, see the chapter "BPXWDYN: a text interface to dynamic allocation and dynamic output" in the *z/OS Using REXX and z/OS UNIX System Services* manual. Specify that the data set be sequential. To learn about other formatting rules for the log file, see "Data sets used by Debug Tool" of the Debug Tool User's Guide.

LOUD
Specifies that Debug Tool displays WTO messages, which helps you debug processing done by this command. Debug Tool normally does not display any messages when it creates this data set. If you are trying to determine why this file was not created, specify LOUD to view any messages.

CCPROGSELECTDSN

This option provides the data set name that contains the Code Coverage Options file (which specifies the Group IDs and the PROGRAM IDs of the COBOL routines that are to be processed). Specify `NULLFILE` if no Code Coverage Options file is to be read.

This dataset must be preallocated as a sequential dataset. `RECFM=VB, LRECL=255` is suggested.

The following diagram describes the syntax of the `CCPROGSELECTDSN` command:

```
►►—EQAXOPT—CCPROGSELECTDSN—,—'—file_name_pattern—'——————————————————►◄
                                                   └—,—LOUD—┘
```

file_name_pattern
Specifies a naming pattern that determines the name of the data set that contains this file. Follow these guidelines when you create the naming pattern:

- Create a 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 determines the name of the data set.
- Specify `NULLFILE` to indicate you do not want Debug Tool to process this file.

LOUD
Specifies that Debug Tool displays WTO messages, which helps you debug processing done by this command. Debug Tool normally does not display any messages if it does not find the data set or data set member. If you are trying

to determine why Debug Tool is not processing this file, specify LOUD to see if it displays a message that it can not find the data set.

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 an EQAOPTS `CODEPAGE` command 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 an EQAOPTS `CODEPAGE` command 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))` parameter 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 CODEPAGE command:

```
►►──EQAXOPT──CODEPAGE──,──nnnn─────────────────────────────────────────►◄
```

nnnn
A positive integer indicating the code page to use.

After implementing the EQAOPTS `CODEPAGE` command, if 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."

Example

```
EQAXOPT CODEPAGE,121
```

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). You can specify the host code page in the `VADSCP`*nnnnn* suboption of `TEST` runtime option or with the `CODEPAGE` command in the EQAOPTS data set. If you specify both the `VADSCP`*nnnnn* suboption and the `CODEPAGE` command, Debug Tool uses only the CODEPAGE command. 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 19. Source and target CCSID to specify, depending on the code page command used

VADSCP*nnnn* suboption or CODEPAGE command	**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
VADSCP1047 or CODEPAGE,1047	1047	1208 (UTF-8)
	1208	1047
Note: 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.

COMMANDSDSN

Indicates that you want Debug Tool to read a user's commands file (with the name of the data set containing the commands file determined by the specified naming pattern) each time it starts. This works in the following situation:

- You do not specify a data set name or DD name for the user's commands file using any other method; for example, the TEST runtime option.
- You or your site specifies the EQAOPTS `COMMANDSDSN` command.
- The data set specified by the EQAOPTS `COMMANDSDSN` command exists and contains a member whose name matches the initial load module name in the first enclave.

The following diagram describes the syntax of the `COMMANDSDSN` command:

```
►►──EQAXOPT──COMMANDSDSN──,──'──file_name_pattern──'──┬─────────┬────────────►◄
                                                      └─,──LOUD─┘
```

file_name_pattern
: Specifies a naming pattern that determines the name of the data set that contains the user's commands file. Follow these guidelines when you create the naming pattern:
  - Create a 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 determines the name of the data set.
  - Specify `NULLFILE` to indicate you do not want Debug Tool to process a commands file.

LOUD
: Specifies that Debug Tool display WTO messages, which helps you debug processing done by this command. Debug Tool normally does not display any messages if it does not find the data set or data set member. If you are trying

to determine why Debug Tool is not processing a user's commands file, specify LOUD to see if it displays a message that it cannot find the data set or the member.

If you choose to implement this option, users who want to use this function must create the commands file as a PDS or PDSE with the allocation parameters that are described in “Data sets used by Debug Tool” in the *Debug Tool User's Guide*. Then, users create a member for each program that they want to debug, with the name of the member matching the initial load module name in the first enclave.

Example

```
EQAXOPT  COMMANDSDSN,'&&USERID.DBGTOOL.COMMANDS'
```

If you log in with user ID `jsmith`, Debug Tool determines the name of the data set to be `JSMITH.DBGTOOL.COMMMANDS`.

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-generated statements. You can control the initial default for this setting by using the EQAOPTS `DEFAULTVIEW` command.

The following diagram describes the syntax of the `DEFAULTVIEW` command:

```
►►──EQAXOPT──DEFAULTVIEW──,──┬─STANDARD─┬──────────────────────────────►◄
                             └─NOMACGEN─┘
```

Each of these fields corresponds to the similar field in the `SET DEFAULT VIEW` command. If you do not code the EQAOPTS `DEFAULTVIEW` command, the initial setting for `DEFAULTVIEW` is STANDARD.

Example

```
EQAXOPT DEFAULTVIEW,NOMACGEN
```

DLAYDBG

You can specify this command only in the EQAOPTS load module. It cannot be specified at run time.

You can specify this command to enable Debug Tool to delay the starting of a debug session until Debug Tool recognizes a certain program name or C function name (compile unit) (along with an optional load module name).

This command can be used only in the non-CICS environments.

The following diagram describes the syntax of the `DLAYDBG` command:

```
►►──EQAXOPT──DLAYDBG──,──┬─NO──┬───────────────────────────────────────►◄
                         └─YES─┘
```

The following list describes the parameters of the `DLAYDBG` command:

NO Indicates that you do not want delay debug enabled; this is the default value.

YES Indicates that you want delay debug enabled.

If you choose to implement this option, users who want to use this option must create the delay debug profile as a physical sequential data set by using the option B of the Debug Tool Utilities: Delay Debug Profile.

Example

```
EQAXOPT DLAYDBG,NO
EQAXOPT DLAYDBG,YES
```

DLAYDBGCND

You can specify this command only in the EQAOPTS load module. It cannot be specified at run time.

You can use this command to indicate whether you want Debug Tool to monitor condition events in the delay debug mode.

This command can be used only in the non-CICS environments.

The following diagram describes the syntax of the DLAYDBGCND command:

```
                               ┌─ALL──┐
►►──EQAXOPT──DLAYDBGCND──,─────┴─NONE─┴─────────────────────────────────────►◄
```

The following list describes the parameters of the DLAYDBGCND command:

ALL
Indicates that you want Debug Tool to monitor all condition events. This is the default option.

NONE
Indicates that you do not want Debug Tool to monitor condition events.

Example

```
EQAXOPT DLAYDBGCND,NONE
```

DLAYDBGDSN

You can specify this command only in the EQAOPTS load module. It cannot be specified at run time.

You can specify this command to indicate that you want Debug Tool to use the specified naming pattern when it constructs the delay debug profile data set name.

This command can be used only in the non-CICS environments.

The following diagram describes the syntax of the DLAYDBGDSN command:

```
►►──EQAXOPT──DLAYDBGDSN──,──'──file_name_pattern──'─────────────────────────►◄
```

file_name_pattern
Specifies a naming pattern that determines the name of the data set that contains the delay debug profile. Follow this guideline when you create the naming pattern:

- Create a 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 determines the name of the data set.

The default naming pattern is `&&USERID.DLAYDBG.EQAUOPTS`.

Example

```
EQAXOPT DLAYDBGDSN,'&&USERID.DLAYDBG.EQAUOPTS';
```

DLAYDBGTRC

You can specify this command only in the EQAOPTS load module. It cannot be specified at run time.

You can specify this command to indicate that you want Debug Tool to generate trace during the pattern match process in the delay debug mode. Debug Tool uses the WTO (write to operator) command to output the trace.

This command can be used only in the non-CICS environments.

The following diagram describes the syntax of the `DLAYDBGTRC` command:

```
►►──EQAXOPT──DLAYDBGTRC──,──trace_level──────────────────────────────────►◄
```

trace_level
Specifies a trace level that determines the level of traces that Debug Tool generates. Valid levels are:

- 0 - no trace message; this is the default value.
- 1 - error and warning messages
- 2 - error, warning, and diagnostic messages
- 3 - error, warning, diagnostic, and internal diagnostic messages

Example

```
EQAXOPT DLAYDBGTRC,2
```

DLAYDBGXRF

You can specify this command only in the EQAOPTS load module. It cannot be specified at run time.

In this section, the term generic ID refers to a user ID that executes a task but is not the ID that debugs the task. Common examples include the user ID associated with a WebSphere MQ for z/OS trigger monitor, or the user ID that runs a web services-initiated program.

You can use the DLAYDBGXRF command to map a generic ID to a user ID that is obtained from one of the following sources:

- The Delay Debug cross reference file
- Users logged on to Terminal Interface Manager

The user ID that is obtained by this method is used in place of the current user ID when the delay debug profile data set name is constructed.

The generic ID-to-user ID mapping can occur in the following environments:

- In the IMS environment when an IMS transaction is started with a generic ID.

- In the DB/2 stored procedures environment. This environment is supported by the REPOSITORY and CLIENTID options.

The following diagram describes the syntax of the `DLAYDBGXRF` command:

```
►►──EQAXOPT──DLAYDBGXRF──,──┬─DSN──,──'file_name'─┬──────────────────────────►◄
                            ├─REPOSITORY──────────┤
                            └─CLIENTID────────────┘
```

DSN
: Specifies that the generic ID cross reference file *'file_name'* should be used to map the generic ID to the user ID of the Debug Tool user.

REPOSITORY
: Specifies that Debug Tool communicates with Terminal Interface Manager (TIM) to determine whether a user has logged on to TIM and requested to debug the current IMS transaction or DB/2 stored procedure.

  The REPOSITORY option requires that the debugging user ID be granted RACF authority to debug tasks initiated by the generic ID. This is done via the EQADTOOL.GENERICID.*generic_user_ID* facility. To set this up, use the following RACF commands:

  ```
  RDEFINE EQADTOOL.GENERICID.generic_user_ID CLASS(FACILITY) UACC(NONE)
  PERMIT EQADTOOL.GENERICID.generic_user_ID ID(user) ACC(READ)
  ```

  The *generic_user_ID* can be a pattern.

  The REPOSITORY option also requires that you start the Terminal Interface Manager started task with the REPOSITORY option. See “Starting the Terminal Interface Manager” for more information.

CLIENTID
: Specifies that Debug Tool uses the DB/2 client user ID for a stored procedure call to determine whether a remote debug user has requested to debug DB/2 stored procedures that execute with that client user ID. If such a user exists, their user ID will be used to locate the delay debug profile data set.

'file_name'
: Specifies an MVS sequential data set with FB LRECL 80 characteristics.

Example

```
EQAXOPT DLAYDBGXRF,DSN,'EQAW.TRNUSRID.XREF'
EQAXOPT DLAYDBGXRF,REPOSITORY
```

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

Related references
Debugging tasks running under a generic user ID in the *Debug Tool User's Guide*

DOPTACBDSN

You can specify this command only in the EQAOPTS load module. It cannot be specified at run time.

The DOPTACBDSN command identifies the data set that will contain DOPT PSBs that are created by Debug Tool for the IMS Transaction Isolation Facility. This data set will be used to store ACBs generated for the EQATcccn DOPT PSBs.

There is no default value for this command. If your site will use the IMS Transaction Isolation Facility, this command must be specified.

The following diagram describes the syntax of the DOPTACBDSN command:

```
►►──EQAXOPT──DOPTACBDSN──,──'data_set_name'──────────────────────────────►◄
```

'data_set_name'
Fully-qualified name of the data set that will contain the EQAT*cccn* DOPT PSBs.

DTCNDELETEDEADPROF

You can specify this command only in the EQAOPTS load module. It cannot be specified at run time.

This command controls the deletion of DTCN profiles. A dead profile is a profile whose owner has logged off or disconnected from the CICS region.

DTCN scans its repository for dead profiles from time to time, and on demand when `EQADCDEL` is called. When a dead profile is found, it deactivates the profile by default. You can specify `DTCNDELETEDEADPROF` to delete the profile instead. The dead profile is deactivated or deleted by the INACTIVATE or DELETE DTCN function.

The following diagram describes the syntax of the `DTCNDELETEDEADPROF` command:

```
                                   ┌─NO──┐
►►──EQAXOPT──DTCNDELETEDEADPROF──,──┴─YES─┴──────────────────────────────►◄
```

NO Indicates that you want DTCN not to delete the profile. `NO` is the default option.

YES
Indicates that you want DTCN to delete the profile.

Example

```
EQAXOPT DTCNDELETEDEADPROF,YES
```

DTCNFORCE*xxxx*

You can specify these commands only in the EQAOPTS load module. You cannot specify them at run time.

If your users create debugging profiles with DTCN, you can use the `DTCNFORCE`*xxxx* commands to require that certain DTCN fields are not left blank. The following list describes each resource type you can require each user 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 the DTCNFORCE*xxxx* command:

```
►►──EQAXOPT──┬─DTCNFORCECUID──,───────┬──┬─YES─┬──────────────────────────────►◄
             ├─DTCNFORCEIP──,─────────┤  └─NO──┘
             ├─DTCNFORCELOADMODID──,──┤
             ├─DTCNFORCENETNAME──,────┤
             ├─DTCNFORCETERMID──,─────┤
             ├─DTCNFORCETRANID──,─────┤
             └─DTCNFORCEUSERID──,─────┘
```

YES
Indicates that the specified field is required.

NO Indicates that the specified field is not required.

Examples

```
EQAXOPT DTCNFORCEUSERID,YES
EQAXOPT DTCNFORCETRANID,NO
```

DYNDEBUG

Debug Tool Reference and Messages describes how you use the SET DYNDEBUG command to enable or disable dynamic debug mode in Debug Tool.

The initial default setting is DYNDEBUG ON. If you want to change the initial default setting, use the EQAOPTS DYNDEBUG command.

The following diagram describes the syntax of the DYNDEBUG command:

```
►►──EQAXOPT──DYNDEBUG──,──┬─ON──┬─────────────────────────────────────────────►◄
                          └─OFF─┘
```

ON Sets the initial default to DYNDEBUG ON.

OFF
Sets the initial default to DYNDEBUG OFF.

Example

```
EQAXOPT DYNDEBUG,OFF
```

EQAQPP

You must specify this command 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.

The following diagram describes the syntax of the EQAQPP command:

```
►►──EQAXOPT──EQAQPP──,──┬─ON──┬───────────────────────────────────────────────►◄
                        └─OFF─┘
```

ON Indicates Debug Tool supports Q++ debugging.

OFF
: Indicates Debug Tool does not support Q++ debugging. If you do not specify the EQAQPP command, OFF is the default.

Example

```
EQAXOPT EQAQPP,ON
```

EXPLICITDEBUG

The *Debug Tool Reference and Messages* describes how you use the `SET EXPLICITDEBUG` command to enable explicit debug mode in Debug Tool. However, before you can enter the `SET EXPLICITDEBUG` command, Debug Tool has already processed the initial load module and loaded the debug data for the compile units it contains. If you want to enable explicit debug mode prior to processing the initial load module, use the EQAOPTS `EXPLICITDEBUG` command.

The following diagram describes the syntax of the `EXPLICITDEBUG` command:

```
►►──EQAXOPT──EXPLICITDEBUG──,──┬─ON──┬───────────────────────────────────►◄
                               └─OFF─┘
```

ON Enables explicit debug mode.

OFF
: Disables explicit debug mode. This is the default.

Example

```
EQAXOPT EXPLICITDEBUG,ON
```

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 the Debug Tool User's Guide for a description of the interface features you can change.)

Whenever a user starts Debug Tool, Debug Tool processes the commands in the global preferences file 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

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 programming language of the first program Debug Tool encounters. Because you might not know what programming language Debug Tool will encounter first, 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 command to indicate the name of the global preferences file.

   For *'file_name'*, specify the name of the data set where the global preferences file will be stored.

The following diagram describes the syntax of the GPFDSN command:

```
►►──EQAXOPT──GPFDSN──┬──────────────────────┬─────────────────────────────────►◄
                     └─,──'──file_name──'─┘
```

'file_name'
: The name of the data set where you stored the global preferences file.

Examples

```
EQAXOPT GPFDSN,'GROUP1.COMMON.DTOOL.PREFS'
EQAXOPT GPFDSN
```

HOSTPORTS

You can specify this command only in the EQAOPTS load module. It cannot be specified at run time.

You can use this command to specify a host port or range of ports for a TCP/IP connection from the host to a workstation when using remote debug mode.

The following diagram describes the syntax of the `HOSTPORTS` command:

```
►►──EQAXOPT──HOSTPORTS──┬───────────────────────────────────────┬─────────────►◄
                        ├─,──port_number────────────────────────┤
                        ├─,──port_number_range──────────────────┤
                        │         ┌─,─────────────────────┐     │
                        │         ▼                       │     │
                        └─,──(──┬─port_number───────┬─┴──)─┘
                                 └─port_number_range─┘
```

port_number
: A positive integer (1-32767) identifying a TCP/IP port number.

port_number_range
: The first and last *port_number* identifying a range of port numbers, separated by a hyphen (-).

Examples

```
EQAXOPT  HOSTPORTS,29500-30499
EQAXOPT  HOSTPORTS,(29500-30499,31500-32499)
```

IGNOREODOLIMIT

This command tells Debug Tool to display COBOL table data items even when an ODO value is out of range, and to suppress the following messages:

MFI and batch:
: EQA1471E Incorrect value for ODO variable data item

Remote:
EQA2377E Invalid data.

The following diagram describes the syntax of the IGNOREODOLIMIT command:

```
►►──EQAXOPT──IGNOREODOLIMIT──,──┬─YES─┬──────────────────────────────►◄
                                └─NO──┘
```

YES
Indicates that Debug Tool should display the requested table data item even when an ODO value is out of range, and to suppress issuing EQA1471E or EQA2377E.

NO Indicates that Debug Tool should not display the requested table data item when an ODO value is out of range, and to issue EQA1471E or EQA2377E.

Examples

```
EQAXOPT IGNOREODOLIMIT,YES
EQAXOPT IGNOREODOLIMIT,NO
```

Notes:
- The default setting is NO.
- IGNOREODOLIMIT only affects the behaviour of Debug Tool if the compilation unit was compiled with one of the following compilers:
  – COBOL for OS/390 & VM Version 2 (5648-A25)
  – Enterprise COBOL for z/OS and OS/390 Version 3 (5655-G53)
  – Enterprise COBOL for z/OS Version 4 (5655-S71)
- IGNOREODOLIMIT is ignored for LangX COBOL.

LOGDSN

By default, Debug Tool handles the log file data set in one of the following ways:
- In a non-CICS environment, when Debug Tool starts in batch mode or full-screen mode, and you allocate the INSPLOG DD name, Debug Tool runs the command `SET LOG ON FILE INSPLOG OLD` and starts writing the log to INSPLOG.
- In a CICS environment, when Debug Tool starts in full-screen mode, it runs the command `SET LOG OFF`. If you want a log file, you run the command `SET LONG ON FILE` *`fileid`* `OLD` and Debug Tool starts writing the log to *fileid*.

`LOGDSN` allows a site or a user to specify the default data set name for the log file. If you specify the `LOGDSN` command, Debug Tool handles the log file in the following way:
- In a non-CICS environment, when Debug Tool starts in batch mode or full-screen mode, and you allocate the INSPLOG DD name, Debug Tool runs the command `SET LOG ON FILE INSPLOG OLD` and starts writing the log to INSPLOG . This behavior remains the same.
- In a non-CICS environment, when Debug Tool starts in full-screen mode, if you do not allocate the INSPLOG DD name, Debug Tool runs the command `SET LOG ON FILE` *`fileid`* `OLD` and starts writing the log to the data set specified in the `LOGDSN` command.
- In CICS, when Debug Tool starts in full-screen mode, it runs the command `SET LOG ON FILE` *`fileid`* `OLD` and starts writing the log to the data set specified in the `LOGDSN` command.

This allows a user to always write the log file to a data set, whether in CICS or not, and without having to pre-allocate the log file data set.

For instructions on how to specify the allocation parameters for automatically creating the data set, see "LOGDSNALLOC." Use the EQAOPTS `LOGDSN` and `LOGDSNALLOC` commands to help a new Debug Tool user automatically create and write to the log file.

If you are an existing Debug Tool user that uses a SAVESETS data set, and you or your site specify the EQAOPTS commands `LOGDSN` and `LOGDSNALLOC`, then the SAVESETS data set contains a `SET LOG` command that overrides the EQAOPTS command `LOGDSN`.

The following diagram describes the syntax of the `LOGDSN` command:

```
►►──EQAXOPT──LOGDSN──,──'──file_name_pattern──'──┬────────┬──────────────────►◄
                                                 └─,──LOUD─┘
```

`file_name_pattern`
: Specifies a naming pattern that determines the name of the data set that contains the log file. Follow these guidelines when you create the naming pattern:
  - Create a 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 determines the name of the data set.
  - Specify `NULLFILE` to indicate you do not want Debug Tool to write to a log file.

`LOUD`
: Specifies that Debug Tool display WTO messages, which helps you debug processing done by this command. Debug Tool normally does not display any messages if it does not find the data set. If you are trying to determine why Debug Tool is not writing to this log file, specify LOUD to see if it displays any messages.

If you choose to implement this option, users who want to use the EQAOPTS `LOGDSN` command must create a log file in one of the following ways:
- Instruct Debug Tool to create the log file by specifying the EQAOPTS `LOGDSNALLOC` command, as described in "LOGDSNALLOC."
- Create the log file manually with the allocation parameters that are described in "Data sets used by Debug Tool" in the *Debug Tool User's Guide* .

Example

```
EQAXOPT  LOGDSN,'&&USERID.DBGTOOL.LOG'
```

If you log in with user ID `jsmith`, Debug Tool determines the name of the data set to be `JSMITH.DBGTOOL.COMMMANDS`.

LOGDSNALLOC

Indicates that you want Debug Tool to create the log file data set specified by EQAOPTS `LOGDSN` command if it does not exist. You specify the EQAOPTS `LOGDSNALLOC` command with the corresponding allocation parameters for the data set, which Debug Tool uses when it creates the data set.

The following diagram describes the syntax of the `LOGDSNALLOC` command:

```
►►──EQAXOPT──LOGDSNALLOC──,──'──allocation_parms──'──┬────────┬──────────────►◄
                                                      └─,──LOUD─┘
```

allocation_parms
: Specifies the allocation parameters you want Debug Tool to use when it creates the data set. You can specify only the keys in the following list:

- BLKSIZE
- BLOCK
- CYL
- DATACLAS
- DIR
- DSNTYPE
- DSORG
- LRECL
- MGMTCLAS
- RECFM
- SPACE
- STORCLAS
- TRACKS
- UNIT
- VOL

Separate the keys by one or more blanks. Debug Tool does not provide defaults for any of the keys.

For information on the format of the keys, see the chapter "BPXWDYN: a text interface to dynamic allocation and dynamic output" in the *z/OS Using REXX and z/OS UNIX System Services* manual. Specify that the data set be sequential. To learn about other formatting rules for the log file, see "Data sets used by Debug Tool" of the Debug Tool User's Guide.

LOUD
: Specifies that Debug Tool display WTO messages, which helps you debug processing done by this command. Debug Tool normally does not display any messages when it creates this data set. If you are trying to determine why the log file was not created, specify LOUD to view any messages.

Example

```
EQAXOPT  LOGDSNALLOC,'MGMTCLAS(STANDARD) STORCLAS(DEFAULT)    +
               LRECL(72) BLKSIZE(0) RECFM(F,B) DSORG(PS) SPACE(2,2) +
               CYL'
```

MAXTRANUSER

You can specify this command only in the EQAOPTS load module. It cannot be specified at run time.

The MAXTRANUSER command defines the maximum number of IMS transactions that a single user can register to debug using the IBM Transaction Isolation Facility. If this command is not specified, the default value of 15 will be used.

The following diagram describes the syntax of the MAXTRANUSER command:

```
►►──EQAXOPT──MAXTRANUSER──,──max_trans──────────────────────────────►◄
```

max_trans
: An integer value between 1 and 15 to designate the maximum number of transactions a user can register to debug.

MDBG

If you are using z/OS XL C/C++, Version 1.10 or later, you can indicate that Debug Tool always searches for .mdbg files to retrieve the source and debug information by using the MDBG command.

The following diagram describes the syntax of the MDBG command:

```
►►──EQAXOPT──MDBG──,──┬─YES─┬───────────────────────────────────────►◄
                      └─NO──┘
```

YES
: Indicates that Debug Tool searches for .mdbg files.

NO Indicates that Debug Tool does not search for .mdbg files.

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.

Example

```
EQAXOPT MDBG,YES
```

MULTIPROCESS

Controls the behavior of Debug Tool when a new POSIX process is created by a fork() or exec() function in the application.

With the `MULTIPROCESS` command, you can instruct Debug Tool to perform any of the following tasks when a new POSIX process is created:

- Continue debugging the current process. The current process is also referred to as the PARENT process.
- Stop debugging the current process and start debugging the newly created process. The newly created process is also referred to as the CHILD process.
- Prompt you to decide whether to follow the PARENT or CHILD process.

Note: The `MULTIPROCESS` command applies only to remote debug mode.

The following diagram describes the syntax of the `MULTIPROCESS` command:

```
                                      ┌─PARENT─┐
►►──EQAXOPT──MULTIPROCESS──,──────┼─CHILD──┼──┬───────────────────────┬──────────────►◄
                                      └─PROMPT─┘  │              ┌─NONE─┐  │
                                                  └─,──EXEC=─────┴─ANY──┴──┘
```

The following list describes the parameters of the `MULTIPROCESS` command:

PARENT
: Indicates that Debug Tool continues with the current debug session; that is, Debug Tool follows the PARENT process.

CHILD
: Indicates that Debug Tool stops debugging the current process and starts debugging the newly created process; that is, Debug Tool follows the CHILD process.

PROMPT
: Indicates that the remote debug GUI prompts you to decide whether to follow the PARENT or CHILD process.

EXEC=ANY
: Indicates that Debug Tool debugs any process that is reinitialized by the exec() function.

EXEC=NONE
: Indicates that Debug Tool does not debug a process that is reinitialized by the exec() function. If you do not specify the `EXEC` option, the default setting is `EXEC=NONE`.

Examples

- Specify that Debug Tool follows the PARENT process and debugs the new process that is created by the exec() function.
  ```
  EQAXOPT MULTIPROCESS,PARENT,EXEC=ANY
  ```
- Specify that Debug Tool follows the CHILD process and does not debug the new process that is created by the exec() function.
  ```
  EQAXOPT MULTIPROCESS,CHILD,EXEC=NONE
  ```
- Specify that you are prompted to choose whether to follow the PARENT or CHILD process and Debug Tool does not debug the new process that is created by the exec() function.
  ```
  EQAXOPT MULTIPROCESS,PROMPT
  ```

NAMES

The topic "Solving Problems in Complex Applications" in the Debug Tool User's Guide in the 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) through the EQAOPTS `NAMES` command.

One or more invocations of the EQAOPTS `NAMES` command can be used for this purpose.

The following diagram describes the syntax of the NAMES command:

```
►►──EQAXOPT──NAMES──,──┬─EXCLUDE──,──┬─LOADMOD─┬──,─pattern─┬──────────────────►◄
                       │             └─CU──────┘            │
                       └─INCLUDE──,──┬─LOADMOD─┬──,─name────┘
                                     └─CU──────┘
```

Each of these fields corresponds to the similar parameter in the Debug Tool NAMES command. If you use an asterisk (*) in *pattern* to indicate a wildcard, you must enclose *pattern* in apostrophes.

Examples

```
EQAXOPT NAMES,EXCLUDE,LOADMOD,'ABC1*'
EQAXOPT NAMES,EXCLUDE,CU,MYCU22
EQAXOPT NAMES,EXCLUDE,CU,'MYCU*'
EQAXOPT NAMES,INCLUDE,LOADMOD,CEEMYMOD
EQAXOPT NAMES,INCLUDE,CU,EQATESTP
```

NODISPLAY

In the following two situations, in which a user can request a specific user interface, that interface might not be available:

- Full-screen mode using the Terminal Interface Manager. 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 running under 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 EQAOPTS NODISPLAY command so that Debug Tool continues processing as if the user immediately entered a QUIT DEBUG command. This modification prevents any forced abend or prevents the debugger from starting.

The following diagram describes the syntax of the NODISPLAY command:

```
►►──EQAXOPT──NODISPLAY──,──┬─QUITDEBUG─┬─────────────────────────────────►◄
                           └─DEFAULT───┘
```

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.

Example

```
EQAXOPT NODISPLAY,QUITDEBUG
```

PREFERENCESDSN

Indicates that you want Debug Tool to read a user's preferences file (with the name of the data set containing the preferences file determined by the specified naming pattern) each time it starts. This works in the following situation:

- You do not specify a data set name or DD name for the user's preferences file using any other method; for example, the TEST runtime option.
- In a non-CICS environment, you do not allocate ISPPPREF DD.
- You or your site specifies the PREFERENCESDSN command.
- The data set specified by the PREFERENCESDSN command exists.

The following diagram describes the syntax of the PREFERENCESDSN command:

```
►►──EQAXOPT──PREFERENCESDSN──,──'──file_name_pattern──'──┬────────┬──────────►◄
                                                          └─,──LOUD─┘
```

file_name_pattern
: Specifies a naming pattern that determines the name of the data set that contains the preferences file. Follow these guidelines when you create the naming pattern:
  - Create a 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 determines the name of the data set.
  - Specify NULLFILE to indicate you do not want Debug Tool to process a preferences file.

LOUD
: Specifies that Debug Tool display WTO messages, which helps you debug processing done by this command. Debug Tool normally does not display any messages if it does not find the data set. If you are trying to determine why Debug Tool is not processing your preferences file, specify LOUD to see if it displays any messages about not finding the data set.

If you choose to implement this option, users who want to use this function must create the preferences file as a sequential data set with the allocation parameters that are described in "Data sets used by Debug Tool" in the *Debug Tool User's Guide*.

Example

```
EQAXOPT  PREFERENCESDSN,'&&USERID.DBGTOOL.PREFS'
```

If you log in with user ID jsmith, Debug Tool determines the name of the data set to be JSMITH.DBGTOOL.PREFS.

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 EQAOPTS SAVESETDSN and SAVEBPDSN commands, along with a corresponding naming pattern for the data set.

The following diagram describes the syntax of the SAVESETDSN and SAVEBPDSN commands:

```
►►──EQAXOPT──┬─SAVEBPDSN──┬─,─'─file_name_pattern─'──────────────────────►◄
             └─SAVESETDSN─┘
```

file_name_pattern
Specifies 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

Examples

```
EQAXOPT SAVESETDSN,'CICS.DTDATA.&&USERID.SAVSET'
EQAXOPT SAVEBPDSN,'&&USERID.USERDATA.DTOOL.SAVBPMON';
```

SAVESETDSNALLOC, SAVEBPDSNALLOC

Indicates that you want Debug Tool to create the data sets for SAVESETS, SAVEBPS, or both (specified by EQAOPTS `SAVESETDSN` or `SAVEBPDSN` commands) if they do not exist. You specify the EQAOPTS `SAVESETDSNALLOC` and `SAVEBPDSNALLOC` commands with the corresponding allocation parameters for the data sets, which Debug Tool uses when it creates the data sets. After creating each data set, Debug Tool runs commands that save the information (settings, breakpoints, monitors, preferences, and LDD specifications) in the corresponding data set.

The following diagram describes the syntax of the `SAVEBPDSNALLOC` and `SAVESETDSNALLOC` commands:

```
►►──EQAXOPT──┬─SAVEBPDSNALLOC──┬─,─'─allocation_parms─'─┬─────────┬───────►◄
             └─SAVESETDSNALLOC─┘                        └─,─LOUD─┘
```

allocation_parms
Specifies the allocation parameters you want Debug Tool to use when it creates the data set. You can specify only the keys in the following list:

- BLKSIZE
- BLOCK
- CYL
- DATACLAS
- DIR
- DSNTYPE
- DSORG
- LRECL
- MGMTCLAS
- RECFM
- SPACE
- STORCLAS
- TRACKS

- UNIT
- VOL

Separate the keys by one or more blanks. Debug Tool does not provide defaults for any of the keys.

For information on the format of the keys, see the chapter "BPXWDYN: a text interface to dynamic allocation and dynamic output" in the *z/OS Using REXX and z/OS UNIX System Services* manual. Specify that the data set be sequential for SAVESETS; a PDS or PDSE for SAVEBPS. To learn about other formatting rules for these files, see "Data sets used by Debug Tool" of the Debug Tool User's Guide.

LOUD
Specifies that Debug Tool display WTO messages, which helps you debug processing done by this command. Debug Tool normally does not display any messages when it creates these data sets. If you are trying to determine why the data sets were not created, specify LOUD to view any messages.

Debug Tool does the following tasks when you specify these commands:

1. If you specified the `SAVESETDSNALLOC` command, it creates the SAVESETS data set.
2. If it creates the SAVESETS data set successfully, it runs the following commands:
   ```
   SET SAVE SETTINGS AUTO;
   SET RESTORES SETTINGS AUTO;
   ```
   If it did not create the SAVESETS data set successfully, it skips the rest of these steps and does the next processing task.
3. If you specified the `SAVEBPDSNALLOC` command, it creates the SAVEBPS data set.
4. If it creates the SAVEBPS data set successfully, it runs the following commands:
   ```
   SET SAVE BPS AUTO;
   SET SAVE MONITORS AUTO;
   SET RESTORE BPS AUTO;
   SET RESTORE MONITORS AUTO;
   ```

In a CICS environment, review the performance implications discussed in the "Performance considerations in multi-enclave environments" section of the "Using full-screen mode: overview" topic in the *Debug Tool User's Guide* before choosing to implement the `SAVEBPDSNALLOC` command. If you think the performance implications might adversely affect your site, do not implement the `SAVEBPDSNALLOC` command in the EQAOPTS for CICS.

Example

```
EQAXOPT  SAVESETDSNALLOC,'MGMTCLAS(STANDARD) STORCLAS(DEFAULT)  +
         LRECL(3204) BLKSIZE(0) RECFM(V,B) DSORG(PS) SPACE(2,2) +
         TRACKS'
EQAXOPT  SAVEBPDSNALLOC,'MGMTCLAS(STANDARD) STORCLAS(DEFAULT)   +
         LRECL(3204) BLKSIZE(0) RECFM(V,B) DSORG(PO)            +
         DSNTYPE(LIBRARY) SPACE(1,3) CYL'
```

SESSIONTIMEOUT

You can specify this command only in the EQAOPTS load module. It cannot be specified at run time.

You can establish a timeout for idle Debug Tool sessions that use the Terminal Interface Manager. Timed out sessions are canceled after a specified period of no user activity.

The following diagram describes the syntax of the `SESSIONTIMEOUT` command:

```
►►──EQAXOPT──SESSIONTIMEOUT──,──┬─NEVER───────────────┬──────────────────────────►◄
                                ├─QUITDEBUG──,──hhmmssnn─┤
                                └─QUIT──,──hhmmssnn──────┘
```

NEVER
: No timeout is enforced. Unattended sessions will not be canceled.

QUITDEBUG,*hhmmssnn*
: Sessions left unattended for the specified time interval will be canceled, and the process being debugged will proceed as though the `QUIT DEBUG` command had been entered.

  The time interval is expressed as *hhmmssnn*, where
  - *hh* is the number of hours,
  - *mm* is the number of minutes,
  - *ss* is the number of seconds,
  - *nn* is the number of hundredths of seconds.

QUIT,*hhmmssnn*
: Sessions left unattended for the specified time interval will be canceled, and the process being debugged will be terminated with a U4038 abend, as though the `QUIT ABEND` command had been entered.

  The time interval is expressed as *hhmmssnn*, where
  - *hh* is the number of hours,
  - *mm* is the number of minutes,
  - *ss* is the number of seconds,
  - *nn* is the number of hundredths of seconds.

If the command is not specified in the EQAOPTS load module, the default behavior is "NEVER", which means that any full-screen mode session using Terminal Interface Manager left unattended will not be canceled.

Example

To specify a timeout interval of 1 hour and allow the debugged process to proceed after the debug session is canceled, enter the following EQAOPTS command:

```
EQAXOPT SESSIONTIMEOUT,QUITDEBUG,01000000
```

STARTSTOPMSG

This command controls whether to issue a message when each debugging session is initiated or terminated. By default, these messages are not issued.

The following diagram describes the syntax of the `STARTSTOPMSG` command:

```
►►──EQAXOPT──STARTSTOPMSG──,──┬─ALL──────────────────────┬──┬───────┬────────────►◄
                              ├─NONE─────────────────────┤  └─,─WTO─┘
                              ├─CICS─────────────────────┤
                              ├─TSO──────────────────────┤
                              ├─BATCHTSO─────────────────┤
                              ├─IMS──────────────────────┤
                              ├─OTHER────────────────────┤
                              │     ┌─,──────────────┐   │
                              │     ▼                │   │
                              └─(─────┬─CICS─────┬───┴─)─┘
                                      ├─TSO──────┤
                                      ├─BATCHTSO─┤
                                      ├─IMS──────┤
                                      └─OTHER────┘
```

The following list describes the parameters of the `STARTSTOPMSG` command:

ALL Indicates that the start/stop messages should be written in all environments.

NONE
Indicates that no start/stop messages should be written. If you do not specify `EQAOPTS STARTSTOPMSG`, the default option is `NONE`.

CICS Indicates that the start/stop messages should be written in the CICS environment.

TSO Indicates that the start/stop messages should be written in the TSO environment.

BATCHTSO
Indicates that the start/stop messages should be written in the BATCH TSO environment.

IMS Indicates that the start/stop messages should be written when running under IMS. In addition, an informational message that contains the IMS system ID, region ID, and transaction ID is written.

OTHER
Indicates that the start/stop messages should be written in all other environments, such as MVS batch.

WTO Indicates that the start/stop messages should be written to the system log using a WTO.

Examples

```
EQAXOPT STARTSTOPMSG,ALL,WTO
EQAXOPT STARTSTOPMSG,(TSO,OTHER),WTO
```

STARTSTOPMSGDSN

You can specify this command only in the EQAOPTS load module. It cannot be specified at run time.

You can use this command to indicate that you want Debug Tool to write a message in the message file when the debug session is started and stopped.

This command can be used only in the non-CICS environments.

The following diagram describes the syntax of the `STARTSTOPMSGDSN` command:

```
►►──EQAXOPT──STARTSTOPMSGDSN──,──'file_name'──┬─────────┬──────────────────►◄
                                              └─,──LOUD─┘
```

'file_name'
: Specifies an MVS sequential data set with FB LRECL 80 characteristics. It is recommended that you allocate sufficient space for the file and perform regular maintenance by removing outdated messages.

LOUD
: Specifies that Debug Tool displays WTO messages, which help you debug processing done by this command. Debug Tool normally does not display any messages when it operates on the data set. If you try to determine problems related to the data set, specify LOUD to view any related messages.

Example

```
EQAXOPT STARTSTOPMSGDSN,'EQAW.SSMSG.LOG'
```

SUBSYS

If both of the following conditions apply at your site, you need to use the EQAOPTS SUBSYS command:

- 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 or C++ programs without using the EQAOPTS MDBG command or debugging Enterprise PL/I programs compiled without the SEPARATE suboption of the TEST compiler option.

In this case, you must run Debug Tool and the specified subsystem on the same system.

You cannot use SUBSYS to debug programs that run under CICS.

The following diagram describes the syntax of the SUBSYS command:

```
►►──EQAXOPT──SUBSYS──┬──────────────────────────────────┬──────────────►◄
                     └─,──'──four_character_name──'─────┘
```

four_character_name
: Specifies the subsystem name to be used.

Examples

```
EQAXOPT SUBSYS
EQAXOPT SUBSYS,'SBSX'
```

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.
- Debugging LangX COBOL programs.

- Detecting services such as MVS LINK, LOAD, DELETE and ATTACH.

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 debugging LangX COBOL programs.
- Run Debug Tool when you need to detect services such as MVS LINK, LOAD and DELETE.
- Run Debug Tool when you debug subtasks within a multi-tasked application, where subtasks are started by using the ATTACH assembler macro.
- 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 describes the syntax of the `SVCSCREEN` command:

```
►►──EQAXOPT──SVCSCREEN──,──┬─ON──────────────────┬─,──CONFLICT──=──────────────►
                           ├─OFF─────────────────┤
                           └─(──OFF──,──QUIET──)─┘

►─┬─OVERRIDE───┬─┬──────────────────────────────────┬──────────────────────────►◄
  └─NOOVERRIDE─┘ │     ┌─NOMERGE──────────────────┐ │
                 └─,───┴─MERGE──┬───────────────┬─┴─┘
                                └─=──(──COPE──)─┘
```

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. If you start Debug Tool by using the EQANMDBG program, the OFF setting is ignored.

QUIET
Suppresses message EQA2458I, which is written to the Debug Tool log when SVC screening is disabled by default.

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` command with only one other program: COPE.

If you specify the `MERGE` command and Debug Tool does not recognize the program that is using the SVC screening, the `MERGE` command 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 EQAOPTS `SVCSCREEN` command 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`

Use Table 20 as a guide to select the appropriate suboptions.

Examples

```
EQAXOPT SVCSCREEN,ON,CONFLICT=OVERRIDE,NOMERGE
EQAXOPT SVCSCREEN,OFF,CONFLICT=NOOVERRIDE,NOMERGE
```

Combinations of suboptions for the EQAOPTS SVCSCREEN command

The following table shows examples of combinations of suboptions for the EQAOPTS `SVCSCREEN` command:

Table 20. 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.

Table 20. Combination of SVSCREEN options and their effects (continued)

SVCSCREEN options	Type of Debug Tool session	Action
`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.
`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	

TCPIPDATADSN

You can specify this command only in the EQAOPTS load module. It cannot be specified at run time.

You can use this command to instruct Debug Tool to dynamically allocate the specified *'file_name'* to DDNAME SYSTCPD (if SYSTCPD is not already allocated). This provides the data set name for TCPIP.DATA when no GLOBALTCPIPDATA statement is configured in the system TCP/IP options.

The following diagram describes the syntax of the `TCPIPDATADSN` command:

```
►►──EQAXOPT──TCPIPDATADSN──┬─────────────────┬──────────────────────────►◄
                           └─,──'file_name'──┘
```

Example

```
EQAXOPT TCPIPDATADSN,'SYS2.TCPIP.DATA'
```

If you want to use an alternative TCP/IP stack, you can add an entry to the specified `TCPIP.DATA` dataset with a `TCPIPJOBNAME` statement.

```
                            ┌─TCPIPJOBNAME TCPIP──────────┐
►►──┬────────────────┬──────┼─────────────────────────────┼─────────────►◄
    └─system_name:───┘      └─TCPIPJOBNAME──tcpip_proc────┘
```

Example

To specify TCPIPA as the name of the procedure that was used to start the TCP/IP address space, add the following code to the specified `TCPIP.DATA` dataset:

```
TCPIPJOBNAME TCPIPA
```

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.

The following diagram describes the syntax of the THREADTERMCOND command:

```
►►──EQAXOPT──THREADTERMCOND──,──┬─NOPROMPT─┬────────────────────────────►◄
                                └─PROMPT───┘
```

NOPROMPT
Suppress the display of termination prompts.

PROMPT
Prompts the user at termination. If you do nto specify the `THREADTERMCOND` command, the default PROMPT is used.

Example

```
EQAXOPT THREADTERMCOND,NOPROMPT
```

TIMACB

You can include this command only in the EQAOPTS load module. You cannot specify it at run time.

TIMACB identifies the name of an ACB, other than EQASESSM, that Debug Tool uses to make full-screen mode using the 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

“Example: Defining the VTAM EQAMVnnn and Terminal Interface Manager APPL definition statements when Debug Tool runs on four LPARs” in the Debug Tool Customization Guide.

The following diagram describes the syntax of the TIMACB command:

```
►►──EQAXOPT──TIMACB──,──ACB_name──────────────────────────────────────────►◄
```

ACB_name
The new ACB name you want used. The default name is EQASESSM.

Example

```
EQAXOPT TIMACB,EQASESS2
```

END

The END command identifies the last EQAOPTS command. You must always specify it and it must be the last command in the input stream.

The following diagram describes the syntax of the END command:

```
►►──EQAXOPT──END──────────────────────────────────────────────────────────►◄
```

Example

```
EQAXOPT END
```

Chapter 16. Adding support for remote debug users

If you have users running Debug Tool in remote debug mode, review the following list and complete the applicable tasks:

- If the source or compiler use a code page other than 037, see "CODEPAGE" on page 109.
- If you want Debug Tool to switch to full-screen mode or batch mode if the remote debugger is not available, see "NODISPLAY" on page 125.
- For your CICS users, see "Enabling communication between Debug Tool and a remote debugger (CICS)."
- For users that install the IBM Debug Tool DTCN and DTSP Profile Manager plug-ins to the remote debugger (or any Eclipse application), do the following tasks:
  - If users want to use the DTCN Profiles view, see "Adding support for the DTCN Profiles view and APIs" on page 138.
  - If users want to use the DTSP Profile, code coverage, and load module analyzer views, see "Adding support for the DTSP Profile, code coverage, and load module analyzer views" on page 142.

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

Related references
"Remote debug mode" in the *Debug Tool User's Guide*

Enabling communication between Debug Tool and a remote debugger (CICS)

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.

Adding support for the DTCN Profiles view and APIs

Your users can add the **DTCN Profiles** view to the **Debug** perspective of an Eclipse-based application, like IBM Problem Determination Tools Studio or IBM CICS Explorer® or Rational® Developer for System z®. The **DTCN Profiles** view helps users manage and store DTCN profiles stored on your z/OS system. The view uses APIs described in *Debug Tool API User's Guide and Reference*. Before your users can use the **DTCN Profiles** view or create applications that utilize these APIs, you must do the following tasks:

- Enable TCP/IP communication between the **DTCN Profiles** view (or an application utilizing the APIs) and an HTTP server running in a CICS region on the z/OS system, as described in "Defining the CICS TCPIPSERVICE resource."
- If you want users other than the profile owners to modify or delete DTCN profiles, see "Defining who can create, modify, or delete DTCN profiles" on page 141.
- Inform your users that you have enabled this support and now they can use the **DTCN Profiles** view. They can find instructions for installing it in "Appendix I: Installing the IBM Debug Tool DTCN and DTSP Profile Manager plug-ins" in the *Debug Tool User's Guide*.

Defining the CICS TCPIPSERVICE resource

Before your users can begin using the **DTCN Profiles** view or develop applications that use the APIs, you must define the CICS TCPIPSERVICE resource for every CICS region that users access.

For every region that users access, do these steps:

1. Define the TCP/IP address and host name for the z/OS system. By default, they are defined in the PROFILE.TCPIP and TCPIP.DATA data sets.
2. Add a TCP/IP listener to CICS. Use the following CEDA command to define a TCPIPSERVICE in a group:

   ```
   CEDA DEF TCPIPSERVICE(service-name) GROUP(group-name)
   ```

   Ensure that the group in which you define the service is in the startup GRPLIST, so that the listener starts when CICS starts. The following list explains what values to use for some of the key fields:

TCpipservice(*service-name*)
Create a name that is eight characters or less.

GROup(*group-name*)
Create a name that is eight characters or less.

Urm
Specify EQADCAN0.

POrtnumber
Specify an unused port number that the DTCN Profile Manager plug-in uses for the API's communication.

STatus
Specify `Open`.

PROtocol
Specify `Http`.

TRansaction
Specify `CWXN`.

Backlog
The number of TCP/IP requests that are queued before TCP/IP starts to reject incoming requests. For example, 30.

SOcketclose
Specify `No`.

Maxdatalen
Specify the maximum size, in bytes, of the body (the XML document) of the HTTP request or response. For example, `032768` represents 32K bytes.

SSl
Specify Yes if you are using SSL encryption with the HTTPS protocol.

AUthenticate
Specify `Basic`.

GRPcritical
Specify `No`.

3. Enter the following command to install the TCPIPSERVICE definition:
   ```
   CEDA INS TCPIPSERVICE(service-name) GROUP(group-name)
   ```
4. If you encounter problems with the connection from the workstation to the HTTP server in the CICS region, issue the command `CEMT INQUIRE TCPIPSERVICE(`*`service-name`*`)` on the server region to verify if your settings are correct. Your settings must be as follows:
   - `Port(`*`nnnn`*`)`, where *nnnn* is the port that the workstation connects to.
   - `Authentication(Basic)`
   - `Maxdatalen(032000)`
   - `Urm(EQADCAN0)`

Establishing secure communication between the DTCN profile view for CICS and your z/OS system

These steps help you enable secure communication via Secure Sockets Layer (SSL) between the DTCN profile view and your z/OS system. The communication between the client and server uses the HTTP protocol.

Server-side setup

To enable SSL communication, do the following tasks for the server side:

- Generate key pair and self-signed certificate.
  1. Use the RACF GENCERT command to create a key entry for the CICS region owner. The key entry contains the key pair and self-signed certificate.

     Note: The following example shows the RACF commands as they would be coded in a REXX exec. This is recommended because of the length of the commands.

     Example (Create a key entry for user USERID with label: USERID-DTCNPLG-CERT):

     ```
     /* generate key entry */
     "RACDCERT ID(USERID) GENCERT",
     " SUBJECTSDN(CN('your_host_name.com' )",
     "T ('USERID-DTCNPLG-CERT' ) ",
     "OU('IBM' ) ",
     "O ('IBM' ) ",
     "L ('San Jose' ) ",
     "SP('CA' ) ",
     "C ('US' ))",
     " NOTBEFORE(DATE(2011-02-28) TIME(20:00:00) )",
     " NOTAFTER (DATE(2031-12-31) TIME(19:59:59) )",
     " WITHLABEL('USERID-DTCNPLG-CERT' )",
     " SIZE (1024 )"
     ```
  2. Connect the key entry to a key ring that belongs to the CICS region owner ID. The common name of the subject DSN must be the host name of the server that the client uses to connect to host.

     Example (Connect it to a key ring named USERID):

     ```
     /* connect key entry to key ring */
     "RACDCERT ID(USERID )",
     "CONNECT( RING(USERID ) ",
     " LABEL('USERID-DTCNPLG-CERT' ))"
     ```
  3. Export the certificate and store it in a data set using the printable encoding format defined by the internet RFC 1421 standard.

     Example (Export the certificate to a data set: USERID.DTCNPLG.CERT):

     ```
     /* export certificate to a data set */
     "RACDCERT EXPORT(LABEL('USERID-DTCNPLG-CERT' ) ",
     " ID(USERID ) ",
     " DSN('USERID.DTCNPLG.CERT' ) ",
     " FORMAT(CERTB64 ) "
     ```
- Update system initialization parameters in CICS region.
  1. Add a KEYRING system initialization parameter to the CICS region job and point it to the key ring created for the region owner ID.
  2. The following example adds KEYRING to the CICS region's system initialization parameters:

     ```
     SIT=6$,
     START=INITIAL,
     RENTPGM=PROTECT,
     ...
     TRANISO=YES,
     KEYRING=key-ring-name,
     EDSALIM=132M,
             ...
     ```
- Modify the TCPIPSERVICE you defined above to set these two attributes:
  - SSl : Yes Yes | No | Clientauth

– CErtificate : USERID-DTCNPLG-CERT

Client-side setup

To enable SSL communication, do the following tasks for the client side:

- Install client certificate.

  Because the server certificate generated is not from an authorized CA, you need to install the certificate into the keystore that either the CICS Explorer or PD Tools Studio uses.

  1. Get a client certificate by downloading a copy of the exported server certificate (using text mode) that is created in step 3 of Server-side setup above to your workstation.
  2. Import the client certificate into the keystore. The following is an example how to import the certificate into keystore using keytool provided by Java™.

     For Java Version 1.6: `keytool –importcert –alias myprivateroot –keystore ..\lib\security\cacerts –file dtcnplg.cer`

     For Java version 1.7: `keytool –importcert –alias myprivateroot –keystore C:\YOUR_WORKSPACE_DIRECTORY\.metadata\.plugins\com.ibm.cics.core.comm\explorer_keystore.jks –file dtcnplg.cer`

     `dtcnplg.cer` is the client certificate. The initial password for the keystore is `changeit`.

  Notes:

  – For Java version 1.6, the default keystore is:

    CICS Explorer
    `C:\CICS_Explorer_HOME_DIRECTORY\jre\lib\security\cacerts`

    PD Tools Studio
    `C:\PD_TOOLS_STUDIO_HOME_DIRECTORY\jre\jre\lib\security\cacerts`

  – For Java version 1.7, the default keystore is: `C:\YOUR_WORKSPACE_DIRECTORY\.metadata\.plugins\com.ibm.cics.core.comm\explorer_keystore.jks`

  – If you use CICS Explorer, the keytool utility can be found in this Java installation bin directory, `C:\CICS_Explorer_HOME_DIRECTORY\jre\bin`.

  – If you use PD Tools Studio, the keytool utility can be found in this Java installation bin directory, `C:\PD_TOOLS_STUDIO_HOME_DIRECTORY\jre\jre\bin`.

Defining who can create, modify, or delete DTCN profiles

A profile owner can always create, modify or delete his own profile. However, you can define, through RACF profiles, other users that can modify or delete any profiles. This might be useful, for example, if you want a system administrator to delete unused or obsolete profiles owned by a user that no longer has access to those profiles.

Only the security administrator of the z/OS system can add or remove IDs to the RACF profiles. After you identify the IDs of the users you want to have this access, do these steps:

1. Establish the profile in the FACILITY class by entering the following RDEFINE command:

   ```
   RDEFINE FACILITY EQADTOOL.DTCNCHNGEANY 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 (for example, user DUSER1) permission to modify another user's profiles by entering the following command:

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

   Instead of connecting individual users, you can specify that DUSER1 be a RACF group profile and then connect authorized users to that group.
4. If the FACILITY class is not active, activate the class by entering the following command:

```
SETROPTS CLASSACT(FACILITY)
```

   Enter the `SETROPTS LIST` command to verify that the FACILITY class is active.
5. Refresh the FACILITY class by entering the following command:

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

Adding support for the DTSP Profile, code coverage, and load module analyzer views

Your users can add these 3 views to the Debug perspective of an Eclipse-based application, like IBM Problem Determination Tools Studio or Rational Developer for System z.

- The DTSP Profile view helps users update the TEST runtime options data set from the remote debugger.
- The code coverage view helps users generate code coverage reports.
- The load module analyzer view helps users analyze the contents of a load module.

Before your users can use these views, you must do the following tasks:

- Install the Problem Determination Tools Common Component Server.
- Install and configure the Debug Tool extensions for the Problem Determination Tools Common Component.
- Run the Problem Determination Tools Common Component Server.
- Inform your users that you have installed this software and now they can use these views. They can find instructions for installing it in "Appendix I: Installing the IBM Debug plug-ins" in the *Debug Tool User's Guide*.

Before you begin, verify that you have the following software installed:

- UNIX System Services
- Problem Determination Tools Common Component, as described throughout *Program Directory for Problem Determination Tools Common Component (GI10-8969).*

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

Related tasks
"Creating and managing the TEST runtime options data set" on page 30

Installing the Problem Determination Tools Common Component Server

IBM Problem Determination Tools Common Component Server is referred to as Common Component Server in this document.

You can find step-by-step procedures to run the Common Component Server in the *Problem Determination Tools for z/OS Common Component Customization Guide and User Guide*.

Installing and configuring the Debug Tool extensions for the Common Component Server

You must create a configuration file for the Debug Tool extensions for the Common Component Server, and then specify the location of the configuration file to the server to enable communication between the views and your z/OS system.

To configure the Debug Tool extensions for the Common Component Server, do the following steps:

1. Create a sequential MVS data set (`FB, LRECL 80`) that contains the following records. Replace the Debug Tool `SEQAMOD` data set name in `SPAWN_STEPLIB=EQAW.SEQAMOD` statement with the installed Debug Tool `SEQAMOD` data set name.

   Note: Do not change other statements.

   ```
   * DTSP profile viewCONFIG=DT
   SPAWN_PROGRAM=EQACMINT
   SPAWN_STEPLIB=EQAW.SEQAMOD
   SPAWN_PARMS_SECTION
   * code coverage view
   CONFIG=CC
   SPAWN_PROGRAM=EQACCINT
   SPAWN_STEPLIB=EQAW.SEQAMOD
   SPAWN_PARMS_SECTION
   * load module analyzer view
   CONFIG=LM
   SPAWN_PROGRAM=EQALMINT
   SPAWN_STEPLIB=EQAW.SEQAMOD
   SPAWN_PARMS_SECTION
   SPAWN_DD=EQASYSPF=EQAW.SEQATLIB(EQALMPFX)
   SPAWN_DD=EQAPGMNM=EQAW.SEQATLIB(EQALMPGM)
   ```

   A sample job that creates and updates the configuration data set is provided in the *hlq*`.SEQASAMP(EQAWCCFG)` data set.
2. Modify the Common Component Server started proc `IPVSRV1`.
   a. Add the configuration data set to the `CONFIG DD` statement concatenation.
   b. Set the region size parameter to 200M or 0M. For example, `//RUN EXEC PGM=IPVSRV,REGION=200M`.
   c. Stop and restart the Common Component Server.
3. When adding a host connection for Problem Determination Tools for z/OS, consult with your host administrator and choose an encoding of character set that matches what your host system uses.

Running the Common Component Server

You can find step-by-step procedures to run the Common Component Server in the *Problem Determination Tools for z/OS Common Component Customization Guide and User Guide*.

Enabling secure communication between Debug Tool and a remote debugger

This section assists you in setting up a secure communication using the z/OS Communications Server: IP Application Transparent – Transport Layer Security (AT-TLS) service. By exploiting the Secure Sockets Layer (SSL) functions of AT-TLS, Debug Tool provides a secure (encrypted) communication with the remote debugger.

The setup assumes that the remote debugger is the server and Debug Tool is the client.

Carry out the following steps:

- Add a client certificate.

  Export a copy of a client certificate from the key store that the remote debugger uses. This client certificate is used by AT-TLS to authenticate the server (remote debugger) during SSL handshaking. If the remote debugger does not have a certificate, you can use the following Java runtime utility to create a key store and certificate and to export a client certificate.

  Create a key store and certificate
  `keytool -genkeypair`

  Export a client certificate
  `keytool -exportcert`

  You must also inform the remote debugger of the location of the key store. For example, for PD Tools Studio, it is in **Preferences->Run/Debug->Debug Daemon** page.

  Upload the certificate to z/OS in text mode. Add the certificate as a SITE certificate to RACF using the RACDCERT command.

  ```
  RACDCERT SITE
     ADD('USERID1.DTPDT.CERT1') WITHLABEL('DTPDT-CERT1') TRUST
  ```

  where USERID1.DTPDT.CERT1 is a file that contains the uploaded certificate and DTPDT-CERT1 is the label of the certificate in RACF.

- Add an AT-TLS rule.

  The rule allows AT-TLS to enable SSL when Debug Tool connects to the remote debugger on a specified port. The following example shows that the remote debugger listens to the secure port 9101. In your development environment, you must determine which secure port the remote debugger listens to and use it in the `RemotePortRange` statement.

  ```
   TTLSRule DTPDTRL1
  {
   RemotePortRange 9101                     <=== secure port
   Direction Outbound                       <=== outbound direction
   TTLSGroupActionRef DTPDTRL1GrpAct
   TTLSEnvironmentActionRef DTPDTRL1EnvAct
  }
  TTLSGroupAction DTPDTRL1GrpAct
  {
   TTLSEnabled On                           <=== enable rule
   Trace 30
  }
  TTLSEnvironmentAction DTPDTRL1EnvAct
  {
   TTLSKeyRingParms
   {
  ```

```
  Keyring *SITE*/*                               <=== virtual key ring
 }
 HandShakeRole Client
}
```

AT-TLS currency

A TLS v1.2 protocol is available that uses more secured algorithms during SSL handshake operations. To use the protocol, take the following steps.

For z/OS Version 1 Release 13

1. Two APARs OA39422 and PM62905 are required to enable TLS v1.2 in z/OS v1.13.
2. AT-TLS rule update. The following code is an example for secure port 9102:

```
TTLSRule DTPDTRL2
{
 RemotePortRange 9102
 Direction Outbound
 TTLSGroupActionRef DTPDT2GrpAct
 TTLSEnvironmentActionRef DTPDT2EnvAct
}
TTLSGroupAction DTPDT2GrpAct
{
 TTLSEnabled On
 Trace 30
 TTLSGroupAdvancedParms
 {
  Envfile /etc/pagent/DTPDT2grp.env <== DTPDT2grp.env contains GSK_PROTOCOL_TLSV1_2=ON
 }                                  <== system SSL environment variable.
}
TTLSEnvironmentAction DTPDT2EnvAct
{
 TTLSKeyRingParms
 {
  Keyring *SITE*/*
 }
 HandShakeRole Client
}
```

For z/OS Version 2 Release 1

1. TLS v1.2 support is included in z/OS v2.1 base.
2. AT-TLS rule update. The following code is an example for secure port 9102:

```
TTLSRule DTPDTRL2
{
 RemotePortRange 9102
 Direction Outbound
 TTLSGroupActionRef DTPDT2GrpAct
 TTLSEnvironmentActionRef DTPDT2EnvAct
}
TTLSGroupAction DTPDT2GrpAct
{
 TTLSEnabled On
 Trace 30
}
TTLSEnvironmentAction DTPDT2EnvAct
{
 TTLSKeyRingParms
 {
  Keyring *SITE*/*
 }
 HandShakeRole Client
 TTLSEnvironmentAdvancedParms
 {
  TLSv1.2 On
 }
}
```

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
	EQABMPSM			EQAUMODG
	EQADBBSM			EQAUMODH
	EQADLISM			EQAUMODI
		EQAUEDAC[1]	EQAUEDAT[2]	EQAUMODJ
		EQAD3CXT[1,3]	EQAD3CXT[2]	EQAUMODK
EQAJCL				EQAUMODL

Note:

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

This is a copy of Chapter 6 from the Debug Tool V10.1 Customization Guide. It documents the setup for 'full-screen mode using a dedicated terminal without/with Terminal Interface Manager' (indicated via the TEST runtime sub-options %MFI and %VTAM). It should be used

- by sites that want to continue to use the TEST %MFI option to specify a dedicated terminal
- as a reference for sites that are moving from an older release of Debug Tool to Debug Tool V11 (or later).

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 150.

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` runtime 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` runtime 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 (EQAMV*nnn*) 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 152.
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 153.
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 158.

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 runtime 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 150):

-
  ```
  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 153

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" on page 152.

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 156 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 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.

- 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.

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, 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 157, and “Example 3” on page 158.

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 153.

```
PORT
   ...
 2  2023 TCP jobname NOAUTOLOG          ; 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 runtime 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 153.

```
PORT
   ...
 2  2023 TCP jobname NOAUTOLOG           ; 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` runtime 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 153.

```
PORT
   ...
 2  2023 TCP jobname NOAUTOLOG          ; 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 153. 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 150 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 150 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

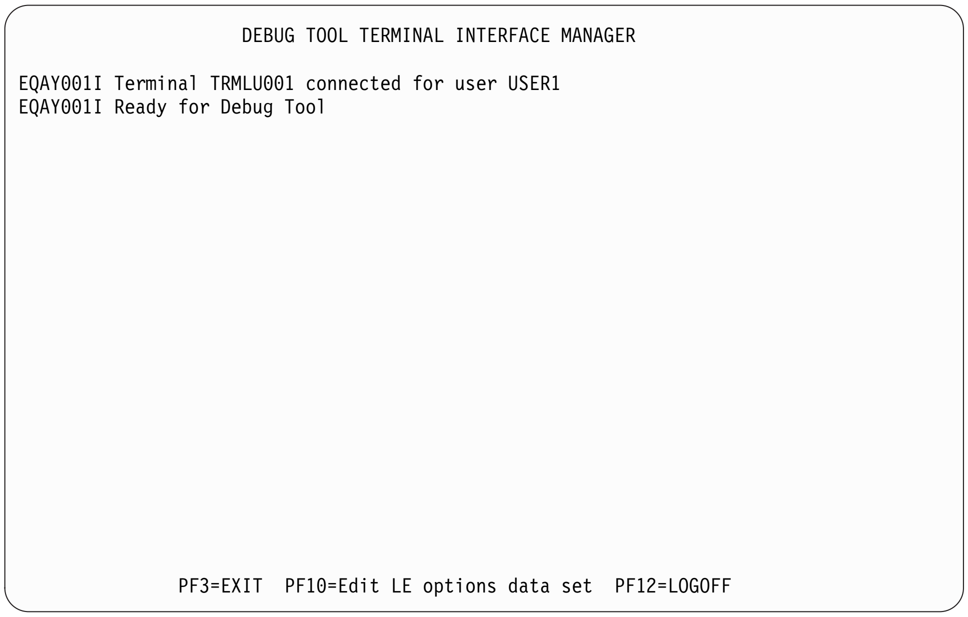
Compare the following steps with the steps shown in “How Debug Tool uses full-screen mode using a dedicated terminal” on page 149 to understand how using the Terminal Interface Manager affects the flow of work.

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:



```
                     DEBUG TOOL TERMINAL INTERFACE MANAGER

EQAY001I Terminal TRMLU001 connected for user USER1
EQAY001I Ready for Debug Tool

               PF3=EXIT  PF10=Edit LE options data set  PF12=LOGOFF
```

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 runtime 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 149 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 149.

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 159. 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 162.
4. Verify that the customizations are completed correctly by following the steps in “Verifying the customization of the Terminal Interface Manager” on page 165.

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 modules, EQAYSESM and EQAYTRMM, 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 164.

-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 164.

+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 156 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 156, with the changes highlighted with an asterisk (*).

```
PORT
 ...
*   2024 TCP jobname NOAUTOLOG            ; 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 "Configuring the TN3270 Telnet Server to access the terminal LUs" on page 153.

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 161, "Starting the Debug Tool Terminal Interface Manager" on page 161, and "Configuring the TN3270 Telnet Server to access the Terminal Interface Manager" on page 162 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 161, 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 161, 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 162, 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 EQAOPTS `TIMACB` command, as described in "TIMACB" on page 135, 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 162.
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%VTAM_LU_id:*` to `VTAM%`*`userid`*`:*`, submit the job and then follow the rest of the instructions in the IVP.
3. On your other terminal emulator session, select the same IVP as you used above, change the runtime parameter string from `MFI%`*`LU_name`*`:*` to `VTAM%`*`userid`*`:*`, submit the job and then follow the rest of the instructions in the IVP.

Appendix C. Applying maintenance

Support resources and problem solving information 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:

1. Prepare to install the APAR or PTF.
2. Receive the APAR or PTF. (SMP/E RECEIVE)
3. Review the HOLDDATA.
4. Accept previously applied APARs or PTFs (optional). (SMP/E ACCEPT)
5. Apply APAR or PTF. (SMP/E APPLY)
6. Run REPORT CROSSZONE and apply any missing requisites.
7. Test APAR or PTF.
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 D. 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 knowledge bases"
- "Getting fixes" on page 171
- "Subscribing to support updates" on page 171
- "Contacting IBM Support" on page 172

Searching knowledge bases

You can search the available knowledge bases to determine whether your problem was already encountered and is already documented.

- Searching the information center
- Searching product support documents

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

If you need to look beyond the information center to answer your question or resolve your problem, you can use one or more of the following approaches:

- Find the content that you need by using the IBM Support Portal at www.ibm.com/software/support or directly at www.ibm.com/support/entry/portal.

  The IBM Support Portal is a unified, centralized view of all technical support tools and information for all IBM systems, software, and services. The IBM

Support Portal lets you access the IBM electronic support portfolio from one place. You can tailor the pages to focus on the information and resources that you need for problem prevention and faster problem resolution.

Familiarize yourself with the IBM Support Portal by viewing the demo videos at https://www.ibm.com/blogs/SPNA/entry/the_ibm_support_portal_videos?lang=en_us about this tool. These videos introduce you to the IBM Support Portal, explore troubleshooting and other resources, and demonstrate how you can tailor the page by moving, adding, and deleting portlets.

Access a specific IBM Software Support site:

- 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
- WebSphere® Studio Asset Analyzer for Multiplatforms Support
- Workload Simulator for z/OS and OS/390 Support

- Search for content by using the IBM masthead search. You can use the IBM masthead search by typing your search string into the Search field at the top of any ibm.com® page.
- Search for content by using any external search engine, such as Google, Yahoo, or Bing. If you use an external search engine, your results are more likely to include information that is outside the ibm.com domain. However, sometimes you can find useful problem-solving information about IBM products in newsgroups, forums, and blogs that are not on ibm.com. Include "IBM" and the name of the product in your search if you are looking for information about an IBM product.
- 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.

Getting fixes

A product fix might be available to resolve your problem. To determine what fixes and other updates are available, select a link from the following list:

- 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 PTFs 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.

Subscribe to receive e-mail notifications about fixes and other IBM Support information as described in Subscribing to Support updates..

Subscribing to support updates

To stay informed of important information about the IBM products that you use, you can subscribe to updates. By subscribing to receive updates, you can receive important technical information and updates for specific Support tools and resources. You can subscribe to updates by using the following:

- RSS feeds and social media subscriptions
- My Notifications

RSS feeds and social media subscriptions

For general information about RSS, including steps for getting started and a list of RSS-enabled IBM web pages, visit the IBM Software Support RSS feeds site at http://www.ibm.com/software/support/rss/other/index.html. For information about the RSS feed for the IBM System z Enterprise Development Tools & Compilers information center, refer to the Subscribe to information center updates topic in the information center at http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/topic/com.ibm.help.doc/subscribe_info.html.

My Notifications

With My Notifications, you can subscribe to Support updates for any IBM product. You can specify that you want to receive daily or weekly email announcements. You can specify what type of information you want to receive (such as publications, hints and tips, product flashes (also known as alerts), downloads, and drivers). My Notifications enables you to customize and categorize the products about which you want to be informed and the delivery methods that best suit your needs.

To subscribe to Support updates, follow the steps below.

1. Click My notifications to get started. Click **Subscribe now!** on the page.
2. Sign in My notifications with your IBM ID. If you do not have an IBM ID, create one ID by following the instructions.

3. After you sign in My notifications, enter the name of the product that you want to subscribe in the **Product lookup** field. The look-ahead feature lists products matching what you typed. If the product does not appear, use the **Browse for a product** link.
4. Next to the product, click the **Subscribe** link. A green check mark is shown to indicate the subscription is created. The subscription is listed under Product subscriptions.
5. To indicate the type of notices for which you want to receive notifications, click the **Edit** link. To save your changes, click the **Submit** at the bottom of the page.
6. To indicate the frequency and format of the email message you receive, click **Delivery preferences**. Then, click **Submit**.
7. Optionally, you can click the RSS/Atom feed by clicking **Links**. Then, copy and paste the link into your feeder.
8. To see any notifications that were sent to you, click **View**.

Contacting IBM Support

IBM Support provides assistance with product defects, answering FAQs, and performing rediscovery.

After trying to find your answer or solution by using other self-help options such as technotes, you can contact IBM Support. Before contacting IBM Support, your company must have an active IBM maintenance contract, and you must be authorized to submit problems to IBM. For information about the types of available support, see the information below or refer to the Support portfolio topic in the Software Support Handbook at http://www14.software.ibm.com/webapp/set2/sas/f/handbook/offerings.html.

- 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 Contacts page of the *IBM Software Support Handbook* on the Web at http://www14.software.ibm.com/webapp/set2/sas/f/handbook/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 http://www.ibm.com/support/servicerequest.
- 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://www14.software.ibm.com/webapp/set2/sas/f/handbook/contacts.html and click the name of your geographic region for phone numbers of people who provide support for your location.

Complete the following steps to contact IBM Support with a problem:

1. "Define the problem and determine the severity of the problem"
2. "Gather diagnostic information"
3. "Submit the problem to IBM Support" on page 174

To contact IBM Software support, follow these steps:

Define the problem and determine the severity of the problem

Define the problem and determine severity of the problem When describing a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Support can help you solve the problem efficiently.

IBM Support needs you 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.

For more information, see the Getting IBM support topic in the Software Support Handbook at http://www14.software.ibm.com/webapp/set2/sas/f/handbook/getsupport.html.

Gather diagnostic information

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 &lang;=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.

Submit the problem to IBM Support

You can submit your problem to IBM Support in one of three ways:

Online using the IBM Support Portal
Click **Service request** on the IBM Software Support site at http://www.ibm.com/software/support. On the right side of the Service request page, expand the Product related links section. Click Software support (general) and select ServiceLink/IBMLink to open an Electronic Technical Response (ETR). Enter your information into the appropriate problem submission form.

Online using the Service Request tool
The Service Request tool can be found at http://www.ibm.com/software/support/servicerequest.

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://www14.software.ibm.com/webapp/set2/sas/f/handbook/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 Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Support website 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. For more information, refer to http://www.ibm.com/support/docview.wss?rs=615&uid=swg21154524.
- If FTP is not possible, e-mail diagnostic data to techsupport@mainz.ibm.com. You must add PMR xxxxx bbb ccc in the subject line of your e-mail. xxxxx is your PMR number, bbb is your branch office, and ccc is your IBM country code. Go to 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 E. 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 format 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

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.

Notices

Trademarks and service marks

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.

E

eXtra Performance LINKage (XPLINK)
A new call linkage between functions that has the potential for a significant performance increase when used in an environment of frequent calls between small functions. XPLINK makes subroutine calls more efficient by removing nonessential instructions from the main path. When all functions are compiled with the XPLINK option, pointers can be used without restriction, which makes it easier to port new applications to z/OS.

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.

I

index A computer storage position or register, the contents of which identify a particular element in a table.

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
A type of network accessible unit that enables users to gain access to network resources and communicate with each other.

A name used by VTAM to identify a terminal or other resource.

M

minor node
In VTAM, a uniquely defined resource within a major node.

multitasking
A mode of operation that provides for concurrent performance, or interleaved execution of two or more tasks.

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*.

O

offset The number of measuring units from an arbitrary starting point to some other point.

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
In SNA, the logical unit that contains the primary half-session for a particular logical unit-to-logical unit (LU-to-LU) session.

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
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.

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
An instruction in a program or procedure.

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)
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.

An access method commonly used by MVS to communicate with terminals and other communications devices.

X

XPLINK
See eXtra Performance LINKage (XPLINK).

Bibliography

Debug tool publications

Using CODE/370 wih 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 Tool and Compliers 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 User's Guide, SC14-7600

Debug Tool Coverage Utility User's Guide and Messages, SC27-4651

Debug Tool Reference and Messages, SC27-4652

Debug Tool Reference Summary, SC14-7602

Debug Tool API User's Guide and Reference, SC27-4654

Debug Tool Customization Guide, SC14-7601

Debug Tool Program Directory, GI13-3004

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

Problem Determination Tools Common Component Program Directory, GI10-8969

Problem Determination Tools for z/OS Common Component Customization Guide and User Guide, SC19-4159

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 5

Customization Guide, SC14-7380

Language Reference, SC14-7381

Programming Guide, SC14-7382

Migration Guide, GC14-7383

Program directory, GI11-9180

Licensed Program Specifications, GI11-9181

Enterprise COBOL for z/OS, Version 4

Compiler and Runtime Migration Guide, GC23-8527
Customization Guide, SC23-8526
Licensed Program Specifications, GI11-7871
Language Reference, SC23-8528
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, Version 4

Language Reference, SC14-7285
Licensed Program Specifications, GC14-7283
Messages and Codes, GC14-7286
Compiler and Run-Time Migration Guide, GC14-7284
Programming Guide, GI11-9145

Enterprise PL/I for z/OS and OS/390, Version 3

Diagnosis, 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 & PM

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 RDRA 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:

SK5T-8871

Online publications can also be downloaded from the IBM website. Visit the IBM website for each product to find online publications for that product.

Index

Product Number: 5655-Q10

Printed in USA

SC14-7601-04