Debug Tool for z/OS
Debug Tool Utilities and Advanced Functions for z/OS

Customization Guide

Version 8.1

SC19-1200-02

Debug Tool for z/OS
Debug Tool Utilities and Advanced Functions for z/OS

Customization Guide

Version 8.1

SC19-1200-02

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 87.

| This edition applies to Debug Tool for z/OS, Version 8.1 (Program Number 5655-S17), with the PTF for PK58192
| applied, which supports the following compilers:

- AD/Cycle® C/370™ Version 1 Release 2 (Program Number 5688-216)
- C/C++ for MVS/ESA Version 3 (Program Number 5655-121)
- C/C++ feature of OS/390 (Program Number 5647-A01)
- C/C++ feature of z/OS (Program Number 5694-A01)
- OS/VS COBOL, Version 1 Release 2.4 (5740-CB1) - with limitations
- VS COBOL II Version 1 Release 3 and Version 1 Release 4 (Program Numbers 5668-958, 5688-023) - with limitations
- COBOL/370™ Version 1 Release 1 (Program Number 5688-197
- COBOL for MVS & VM Version 1 Release 2 (Program Number 5688-197)
- COBOL for OS/390 & VM Version 2 (Program Number 5648-A25)
- Enterprise COBOL for z/OS and OS/390 Version 3 (Program Number 5655-G53)
- Enterprise COBOL for z/OS Version 4.1 (Program Number 5655-S71)
- High Level Assembler for MVS & VM & VSE Version 1 Release 4, Version 1 Release 5 (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 3.7 or earlier (Program Number 5655-H31)

Parts of this edition apply to Debug Tool Utilities and Advanced Functions for z/OS, Version 8.1 (Program Number 5655-S16).

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

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

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

Contents

About this document

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

This document describes the tasks you must do to customize Debug Tool and, if you purchased and installed Debug Tool Utilities and Advanced Functions, the tasks you must do to customize it.

The Debug Tool Utilities and Advanced Functions Coverage Utility is referred to throughout this document as the Debug Tool Coverage Utility or Coverage Utility.

Who might use this document

This document is intended for system administrators who need to customize Debug Tool and (if purchased) Debug Tool Utilities and Advanced Functions after they have been installed.

The following operating systems and subsystems are supported:

- z/OS
  - CICS®
  - DB2®
  - IMS™
  - JES batch
  - TSO
  - UNIX® System Services in remote debug mode or full-screen mode through a VTAM terminal only
  - WebSphere® in remote debug mode or full-screen mode through a VTAM terminal only

Accessing z/OS licensed documents on the Internet

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

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

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

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

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

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

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

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

Printed licensed documents are not available from IBM.

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

Using LookAt to look up message explanations

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

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

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

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

How this document is organized

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

- Chapter 1 gives an overview of all the customization steps. It provides a checklist of all the steps, which you can print and use as a guide as you complete the customization.
- Chapter 2 describes customizations you are required to make and how to modify defaults.
- Chapter 3 describes how to customize the part of Debug Tool Utilities that comes with Debug Tool.
- Chapter 4 describes how to customize the part of Debug Tool Utilities that comes with Debug Tool Utilities and Advanced Functions.

- Chapter 5 describes how to activate the facility to debug in full-screen mode through a VTAM terminal. If your users want to debug any of the following programs by using full-screen mode through a VTAM terminal, you must do the steps described in this chapter:
  - batch programs
  - DB2 stored procedures
  - IMS programs
  - programs running under UNIX System Services
- Chapter 6 describes how to add the EQAUEDAT user exit.
- Chapter 7 describes what you need to do to prepare your environment so that users can debug DB2 stored procedures.
- Chapter 8 describes how to add support for CICS. If your users must debug CICS programs, you must do the instructions described in this chapter.
- Chapter 9 describes how to add support for IMS programs. If your users must debug IMS programs, you must do the instructions described in this chapter.
- Chapter 10 describes what you need to do so that users can specify additional languages (for example, Japanese) with the NATLANG parameter.
- Chapter 11 describes how to create Unicode conversion images.
- Appendix A describes what you need to do to make EQAOPTS options effective at your site. Chapters 1 through 11 contain several topics that describe each EQAOPTS option; this appendix helps you keep track of all of these options so that you can make all these options effective at one time.
- Appendix B describes how to apply maintenance provided for Debug Tool.
- Appendix C describes all the resources available to help you find technical support information.
- Appendix D describes the features and tools available to people with physical disabilities that help them use Debug Tool and Debug Tool documents.

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

Terms used in this document

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

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

Notes:

1. The PL/I program must be compiled and running with one of the following environments:
   - Compiled with Enterprise PL/I for z/OS, Version 3.6 or later, and running in Language Environment Version 1.4 through 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 running in Language Environment Version 1.4 through 1.8 with the PTF for APAR PK33738 applied.

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

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

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

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

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

full-screen mode through a VTAM terminal
Refers to the debugging mode that requires a second terminal, a VTAM® terminal, be started and used to debug an application. After the VTAM terminal has been started, you can optionally use the Debug Tool Terminal Interface Manager to identify that terminal to Debug Tool by using a user ID instead of a LU name.

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

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

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

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

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

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

separate debug file
Refers to the following files:

- Enterprise COBOL for z/OS, Version 4 separate debug file
- Enterprise COBOL for z/OS and OS/390, Version 3, separate debug file
- COBOL for OS/390 & VM, Version 2, separate debug file
- Enterprise PL/I for z/OS, Version 3.5 or later, separate debug file
- file generated by the DWARF suboption of the C/C++ compiler

How to send your comments

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

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

IBM Corporation
H150/090
555 Bailey Avenue
San Jose, CA 95141-1003
USA

- Fax your comments to this U.S. number: (800)426-7773.

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

Summary of changes

This section lists the key changes made to Debug Tool for z/OS and Debug Tool Utilities and Advanced Functions for z/OS that affect this document.

Changes introduced with the PTF for APAR PK58192

You can now debug VS COBOL II programs that are compiled with the NOTEST compiler option and linked with a non-Language Environment library in the same manner that OS/VS COBOL programs are debugged. With the introduction of this support, a new term (*non-Language Environment COBOL*) has been introduced to refer to both OS/VS COBOL programs and these VS COBOL II programs. The following changes have been made to this document:

- A definition of the new term has been added to "Terms used in this document" on page vii.
- Most instances of the term *OS/VS COBOL* have been changed to *non-Language Environment COBOL*. These changes are marked with revision bars.
- The table in "Running the installation verification programs" on page 5 has been updated to include the name of the new installation verification program for non-Language Environment COBOL programs.
- The list in "Verifying the customization of the facility to debug full-screen mode through a VTAM terminal" on page 42 has been updated to include the name of the new installation verification program for non-Language Environment COBOL programs.

The processing for the Debug Tool user exit (EQADBCXT, EQADDCXT, EQADICXT) is enhanced to include an optional program name token in the user exit data set name pattern. This provides a mechanism to have different TEST runtime options for different programs.

The processing for the Debug Tool user exit (EQADBCXT, EQADDCXT, EQADICXT) is enhanced with a recovery routine that handles the s913 exception, which occurs when you cannot access the data set for security reasons.

Changes introduced with the PTF for APAR PK53826

- You can now debug Enterprise PL/I DLL programs.
- The Monitor window has been changed to align the beginning of data along the same column.
- Debug Tool now supports the changes to the TEST compiler option introduced in Enterprise COBOL for z/OS, Version 4.1. See *Enterprise COBOL for z/OS Programming Guide* for more information about the changes made to the TEST compiler option, including the addition of the EJPD suboption.
- Debug Tool now supports the changes to the TEST compiler options introduced in Enterprise PL/I for z/OS, Version 3.7. See *Enterprise PL/I for z/OS Programming Guide* for a description of the changes.
- You can now specify the IPv6 format for TCP/IP addresses. For CICS users, see "Enabling communication between Debug Tool and a remote debugger" on page 59 for instructions on ensuring that users connect with the proper socket mechanism.
- You now have better control over how Debug Tool handles invalid comparisons.

- Miscellaneous improvements to text to improve clarity and accuracy. Chapter 9, "Adding support for debugging under IMS," on page 69 has been improved to describe all the methods available to specify TEST run time options and guidance on selecting the appropriate method for your environment.

Changes introduced with Debug Tool V8.1

- When a FINISH, CEE066 or CEE067 thread termination condition is raised by Language Environment, your system administrator can now prevent Debug Tool from prompting the user by specifying the `THREADTERMCOND` option in the EQAOPTS options file. To learn more about how the option works, see "Suppressing the prompt Debug Tool displays for FINISH, CEE066, or CEE067 conditions" on page 16.
- Debug Tool now supports only single socket connection types with remote debuggers. The `VADTCPIP&` and `TCPIP&` suboptions of the TEST runtime option will both start single socket connections.
- Debug Tool has added an option called CODEPAGE to the EQAOPTS options file for specifying a code page to use during a debugging session. This new option allows you to specify a code page for full screen mode and remote debug mode. See "Specifying a code page" on page 16 for instructions on how to specify the CODEPAGE option of the EQAOPTS option file to enable this feature.
- The `NAMES EXCLUDE` command has been enhanced so that you can more easily indicate the specific types of compile units that you do not want to debug.
- Debug Tool now handles characters that cannot be displayed in their declared data type in a more consistent manner across all programming languages.

  Previously, Debug Tool did not allow you to modify COBOL characters, which could not be displayed in their declared data type, unless you display those characters in hexadecimal format. This behavior was different from the other programming languages. Changes have been made so that Debug Tool handles these characters in the same manner across all programming languages.
- The instructions on how to prepare a DB2 stored procedure have been improved. See Chapter 7, "Preparing your environment to debug a DB2 stored procedures," on page 51 for instructions.
- Debug Tool has made the following enhancements to help make it easier to use command sequences across different programming languages:
  - You can use the `BEGIN` and `END` command with all supported programming languages.
  - You can use a single syntax (X'xxxxxxxx') for hexadecimal addresses in all supported programming languages.
  - You can use quotation marks (") for strings in all supported programming languages.

  This improvement makes it easier to write command sequences that can be used in an environment where several programming languages are used. It also helps improve the ability to restore these command sequences into programs that are written in programming languages other than the one in which they were created.
- Debug Tool now supports the display of the 64-bit general purpose registers when running on hardware that supports this feature. New symbols, `%GPRG`*n* and `RG`*n*, are provided for referencing the 64-bit general purpose registers in disassembly expressions. No support is provided for 64-bit addressing.

- In previous releases, Debug Tool assumed that all floating point data items and registers were in hexadecimal floating point format. Now, for disassembly programs, Debug Tool also supports floating point register in binary (IEEE) and decimal floating point format.
- Debug Tool has a new window called the Memory window, which provides you with better navigation and display of memory in the full screen mode.

  In order to understand how to navigate through all the windows in a Debug Tool session panel, you need to understand the difference between a physical window and a logical window.
- The following enhancements have been made to prefix commands that you enter in the Monitor window:
  - You can use the `DEF` prefix command like the `HEX` prefix command so that it operates on individual array and structure elements.
  - You can use the `HEX` prefix command on Enterprise PL/I variables.
- For CICS programs, you can specify a non-Language Environment assembler program, which is loaded through an EXEC CICS LOAD command, in the Program Id(s) field of the main DTCN screen.
- The following enhancements have been made for remote debug mode:
  - The default port has been changed to 8001.
  - You can now enter the `SET DEFAULT LISTINGS` command through the debug console.
  - You can now use a preferences file, global preferences file, and commands file in remote debug mode.
  - The remote debugger's interface has been enhanced so that you can filter variables. See the remote debugger's online help for a description of this improvement.

Changes introduced with Debug Tool Utilities and Advanced Functions V8.1

- In CICS, Debug Tool Utilities and Advanced Functions now enhances Debug Tool to provide a way to disable or re-enable DTCN and CADP pattern match breakpoints from within a debugging session. You can use `DISABLE CADP`, `DISABLE DTCN`, `ENABLE CADP` and `ENABLE DTCN` commands to control pattern-match breakpoints.
- Debug Tool Utilities and Advanced Functions has expanded the number of ways you can link in the CEEBXITA Language Environment user exit routine, which is provided by Debug Tool.
- Debug Tool Utilities and Advanced Functions provides a new CICS transaction (called DTST), which runs separately from Debug Tool. The DTST transaction helps you view and modify CICS storage. The system programmer can control whether a user can modify storage with this transaction. See “Authorizing DTST transaction to modify storage” on page 66 for instructions.
- Debug Tool Utilities and Advanced Functions adds to Debug Tool the ability to check for specific types of storage violations in CICS.
- Debug Tool Utilities and Advanced Functions enhances Debug Tool so it can now display CICS channels and containers.

  By using the `DESCRIBE CHANNEL` and `LIST CONTAINER` commands, you can now display the contents of CICS channels and containers. For more information about CICS channels and containers, see the section "Enhanced inter-program data transfer: channels as modern-day COMMAREAs" in the *CICS Application Programming Guide*.

- Debug Tool Utilities and Advanced Functions has enhanced the information Debug Tool displays in the automonitor section of the Monitor window for assembler programs.

  Whenever possible, Debug Tool displays user variable and register names. In other cases, it appends comments so you can easily see how _STORAGE operands are associated with the user-coded operands.
- The following enhancements have been made to the automonitor section of the Monitor window:
  - You can use the `HEX` and `DEF` prefix commands in the prefix area of the automonitor section of the Monitor window.
  - You can use the `MONITOR n HEX` command to display the value of a variable in hexadecimal format.
  - You can use the `MONITOR n DEF` command to display the value of a variable in the variable's declared data type.
  - For COBOL characters displayed in the automonitor section of the Monitor window, Debug Tool displays the values in character format, regardless of whether the string contains characters that cannot be displayed in their declared format. You can modify these values by typing over the existing values.
- Debug Tool Utilities and Advanced Functions adds new symbols to Debug Tool, `%GPRGn` and `RGn`, to reference the 64-bit general purpose registers in assembler expressions. You can now display and modify 64-bit arithmetic data items and (on hardware that supports 64-bit) the 64-bit general purpose registers. These 64-bit items can also be used in assembler arithmetic expressions. No support is provided for 64-bit addressing.
- In previous releases, Debug Tool assumed that all floating point data items and registers were in hexadecimal floating point format. Now, for assembler and disassembly programs, Debug Tool Utilities and Advanced Functions enhances Debug Tool so that it also supports floating point data items in binary (IEEE) and decimal floating point format. You can reference and display the floating point registers in any of the three formats. Debug Tool correctly displays all floating-point data items for all three data types (hexadecimal, binary, or decimal). You can use the assembler assignment commands to assign constant values to binary and decimal floating point data items and registers.
- Support for MasterCraft Q++ has been added. For information on MasterCraft Q++, contact Tata Consultancy Services Ltd. See “Configuring for debugging Q++ programs” on page 31 for information about how to add this support.

Chapter 1. Customization checklist

Use the following checklist to customize Debug Tool.

After installing Debug Tool for z/OS, do the following steps:

__ 1. "Product Registration" on page 3

__ 2. "Installing the Dynamic Debug facility" on page 3.

__ 3. "Setting up the APF-authorized system link list data set (SEQABMOD)" on page 7

__ 4. "Setting up the link list data set (SEQAMOD)" on page 7

__ 5. "Changing the default and allowable values in EQACUIDF" on page 7. If your site does not need to change the defaults for NATLANG, LOCALE, or LINECOUNT, you can skip this step.

__ 6. "Specifying global preferences" on page 8. If your site does not need to specify global preferences for Debug Tool, you can skip this step.

__ 7. "Modifying the name of the default data sets that store settings, breakpoints, and monitor values" on page 9. If your site does not need to change the default data set names for these data sets, you can skip this step.

__ 8. "SVC screening option" on page 10. If you do not need to debug non-Language Environment programs that start under Language Environment and your site does not have any host products that might use SVC screening when Debug Tool is started, you can skip this step.

__ 9. "Supplying NAMES commands for the initial load module" on page 13. If your site does not need to issue a `NAMES` command for the initial load module or any of its compile units, you can skip this step.

__ 10. "Setting the initial value for SET DEFAULT VIEW" on page 14. If your site does not debug assembler programs, you do not need to review this section to determine if you need to change this default.

__ 11. "Modifying Debug Tool behavior when requested user interface is not available" on page 14. If your site does not use either full-screen mode through a VTAM terminal or a remote debugger, you do not need to review this section to determine if you need to change this default.

__ 12. "Specifying SUBSYS to access source code in a library system" on page 15. If your site does not use a library system that uses SUBSYS, you do not need to review this section to determine if you need to change this default.

__ 13. "Suppressing the prompt Debug Tool displays for FINISH, CEE066, or CEE067 conditions" on page 16. If your site does not want to suppress the prompt from Language Environment when it terminates an enclave, you can skip this step.

__ 14. "Specifying a code page" on page 16. If your site does not need to use a code page other than the default 037, you can skip this step.

__ 15. "Choosing a method to start Debug Tool Utilities" on page 19. If your site does not use Debug Tool Utilities, you can skip this step.

__ 16. "Customizing the data set names in EQASTART" on page 21. If your site does not use Debug Tool Utilities, you can skip this step.

__ 17. "Adding Debug Tool Utilities to the ISPF menu" on page 21. If your site does not use Debug Tool Utilities or you do not want to add Debug Tool Utilities to your ISPF panel, you can skip this step.

__ 18. "Customizing Debug Tool Setup Utility" on page 22. If your site does not use Debug Tool Setup Utility, you can skip this step.

__ 19. Chapter 5, "Enabling debugging in full-screen mode through a VTAM terminal," on page 33. If your site does not need to debug programs in full-screen mode through a VTAM terminal, you can skip this step.

__ 20. Chapter 6, "Enabling the EQAUEDAT user exit," on page 49. If your site does not need to use the EQAUEDAT user exit, you can skip this step.

__ 21. Chapter 7, "Preparing your environment to debug a DB2 stored procedures," on page 51. If your site does not need to debug DB2 stored procedures, you can skip this step.

__ 22. Chapter 8, "Adding support for debugging under CICS," on page 53. If your site does not need to debug CICS programs, you can skip this step.

__ 23. Chapter 9, "Adding support for debugging under IMS," on page 69. If your site does not need to debug IMS programs, you can skip this step.

__ 24. Chapter 10, "Enabling additional languages for some Debug Tool components via EQACUIDF," on page 71. Skip this step if your site does not use any of the following functions in a Japanese or Korean environment:
- Debug Tool Utilities ISPF panels
- EQANMDBG (non-CICS non-LE)
- Debug Tool Coverage Utility

__ 25. Chapter 11, "Enabling support for display of NLS characters and modification of COBOL NATIONAL variables," on page 73. If your site does not do one of the following, you can skip this step:
- Use a remote debugger to display NLS characters.
- Use the `STORAGE` command to update COBOL NATIONAL variables.
- Properly display C/C++ variables that contain NLS characters

If you are installing Debug Tool for z/OS ***and*** Debug Tool Utilities and Advanced Functions for z/OS, do all of the previous steps, then the following task:

__ 1. Chapter 4, "Customizing Debug Tool Utilities functions shipped with Debug Tool Utilities and Advanced Functions," on page 25. If your site does not use Debug Tool Utilities, you can skip this step.

Chapter 2. Customizing Debug Tool

You are required to make some of the customizations described in this chapter to install Debug Tool. Other customizations are optional.

Use the instructions in this section to complete the following customization tasks:

- Product Registration
- Install the Dynamic Debug facility
- Set up an APF-authorized link list data set
- Set up a link list data set
- Change the default and allowable values in EQACUIDF
- Change the EQAOPTS customization module

Product Registration

You can purchase Debug Tool and Debug Tool Utilities and Advanced Functions in several ways. You must ensure that a Product Registration has been done that is appropriate for the way in which you purchased the tools. The following list describes where you will find the information for doing this Product Registration. Select only one method for registration. This registration must be done before you can run any component of the tools.

Debug Tool Utilities and Advanced Functions
See the "Enable/Register Debug Tool Utils & AF" section of the *Program Directory for Debug Tool Utilities and Advanced Functions for z/OS*.

Debug Tool (stand alone)
See the "Enable/Register Debug Tool section" section of the *Program Directory for Debug Tool for z/OS*.

Debug Tool as a feature of the full-function compiler of Enterprise PL/I for z/OS, Version 3.7
See the *Program Directory* for Enterprise PL/I for z/OS, Version 3.7.

Installing the Dynamic Debug facility

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

- Programs compiled with the `TEST(NOHOOK)` compiler option and the Enterprise PL/I for z/OS Version 3 Release 4 compiler.
- Program compiled with the `TEST(NOHOOK)` compiler option and the Enterprise COBOL for z/OS, Version 4.1 compiler.
- Programs compiled with the `TEST(NONE)` compiler option and one of the following compilers:
  - Enterprise COBOL for z/OS and OS/390, Version 3
  - COBOL for OS/390 & VM, Version 2 Release 2
  - COBOL for OS/390 & VM, Version 2 Release 1 with APAR PQ40298 installed
- Programs for which no debug data is available by using the disassembly view.
- Assembler code that complies with the requirements described in *Debug Tool User's Guide*. You must also install Debug Tool Utilities and Advanced Functions, Version 8 Release 1.

- Load modules loaded by using the MVS™ LOAD and LINK macros.
- Programs that do not run under the Language Environment, including non-Language Environment COBOL programs. You must also install Debug Tool Utilities and Advanced Functions, Version 8 Release 1.
- Programs compiled with the suboption of the TEST compiler option that adds compiled in hooks and with one of the following compilers:
  - Any COBOL compiler supported by Debug Tool
  - Any PL/I compiler supported by Debug Tool
  - Any C/C++ compiler supported by Debug Tool

  The Dynamic Debug facility provides performance enhancements for these programs.

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

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

The Dynamic Debug facility SVCs from this version of Debug Tool are backwards compatible to Debug Tool for z/OS and OS/390, Version 3 Release 1 (Program Number 5655-H32).

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

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

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

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

  These members might have been placed there by previous installations of Debug Tool. Because SYS1.LPALIB is always searched before the data sets in LPALST*xx*, these older members would be found before the newer members in LPALST*xx*.
- Install the SVCs without a system IPL. The SMP/E APPLY operation, which you run when you install Debug Tool or apply a PTF, updates the library *hlq*.SEQAAUTH with the SVCs and the dynamic SVC installer. See "Installing the SVCs without using a system IPL" for information on how to immediately install or update the SVCs.

Installing the SVCs without using a system IPL

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

1. 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.

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

Verifying the installation of the SVCs

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

Checking the level of the Dynamic Debug facility SVCs

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

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

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

```
x4.y.EQA00SVC  2007.255Licensed Materials - Property of IBM 5655-S17 Debug Tool
Version 05 EQA00SVC-F6355aCopyright  Copyright IBM Corp. All Rights Reserved
***> EQA00SVC is Version 05 with compile date 12 Sep 2007
```

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

```
x4.y.EQA01SVC  2007.255Licensed Materials - Property of IBM 5655-S17 Debug Tool
Version 06 EQA01SVC-F5398Copyright  Copyright IBM Corp. All Rights ReservedU
***> EQA01SVC is Version 06 with compile date 12 Sep 2007

x4.y.EQA01SV2  2007.255Licensed Materials - Property of IBM 5655-S17 Debug Tool
Version 02 EQA01SV2-F6355aCopyright  Copyright IBM Corp. All Rights Reserved
***> EQA01SV2 is Version 02 with compile date 12 Sep 2007
```

Running the installation verification programs

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

Table 1. Name of the installation verification program and the programming language corresponding to that installation verification program.

IVP	Task
EQAWIVP4	COBOL TEST(NONE,SYM) or TEST(NOHOOK)
EQAWIVPF	PL/I TEST(ALL,SYM,NOHOOK)
EQAWIVPI	Enterprise PL/I TEST(ALL,SYM,NOHOOK,SEPARATE)
EQAWIVPP	COBOL TEST(NONE,SYM,SEPARATE) or TEST(NOHOOK,SEPARATE)
EQAWIVPS	disassembly

Table 1. Name of the installation verification program and the programming language corresponding to that installation verification program. (continued)

IVP	Task
EQAZIVPA[1]	Language Environment assembler
EQAZIVPC[1]	non-Language Environment assembler
EQAZIVPV[1]	OS/VS COBOL
EQAZIVPX[1]	non-Language Environment VS COBOL II

Notes:

1. This IVP is available only if you installed Debug Tool Utilities and Advanced Functions.

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:

- 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
   ```

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

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

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

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 8, "Adding support for debugging under CICS," on page 53.
- When the Debug Tool Setup Utility component of the Debug Tool Utilities ISPF function is used to start the debugging session (where DTSU accesses *hlq*.SEQAMOD for you).

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

Changing the default and allowable values in EQACUIDF

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 and the Debug Tool Utilities and Advanced Functions components:

- Debug Tool Utilities ISPF dialogs: NATLANG
- EQANMDBG (non-CICS non-LE support): NATLANG
- Debug Tool Coverage Utility: NATLANG, LOCALE, and LINECOUNT

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

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

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 Chapter 10, "Enabling additional languages for some Debug Tool components via EQACUIDF," on page 71 for more information on 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 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.
5. Copy the output load module to *hlq*.SEQABMOD.

Sample JCL is provided in the EQACUIID member of the *hlq*.SEQASAMP data set to perform steps 3 and 4.

The SEQABMOD from this version of Debug Tool is backwards 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.

Specifying global preferences

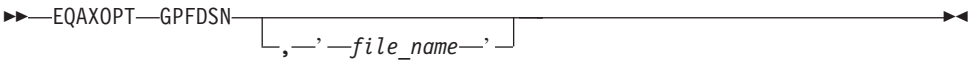
You can define settings or preferences for Debug Tool that apply to all Debug Tool sessions in a global preferences file. For example, if your site uses the PF6 key as the program exit key, you can assign the Debug Tool exit key to be the PF6 key. To create a global preferences file, do the following steps:

1. Create a preferences file that is stored as a sequential file or a PDS member. Refer to *Debug Tool User's Guide* for a description of preferences files.

   The rules for the preferences file are dependant on the language of the first program Debug Tool encounters. Because you might not know what language Debug Tool will encounter first, we recommend you use the following rules when you create the preferences file:

- Put the commands in columns 8 - 72.
- Do not put line numbers in the file.
- Use `COMMENT` or `/* */` to delimit comments.

2. Specify the GPFDSN option in the EQAOPTS option file. The following diagram describes the GPFDSN option:



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

   There are several options that you can specify in the EQAOPTS option file. To help you organize all the options you are selecting , check GPFDSN on the checklist in step 2 on page 75 of the instructions in topic Appendix A, "Defining EQAOPTS options," on page 75. When you are done selecting all the options you want to use, follow the instructions in Appendix A, "Defining EQAOPTS options," on page 75.

Whenever a user starts Debug Tool, the commands in the global preferences file are run first. The user can also create his or her own preferences file and a commands file. In this situation, Debug Tool processes the files in the following order:

1. Global preferences file
2. User preferences file
3. Commands file

Modifying the name of the default data sets that store settings, breakpoints, and monitor values

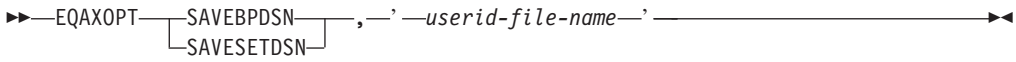
You can modify the default names of the data sets used to save and restore the following information:

- settings (default name: *userid*.DBGTOOL.SAVESETS)
- breakpoints, monitor values, and LOADDEBUGDATA (LDD) specifications (default name: *userid*.DBGTOOL.SAVEBPS)

In most environments, you can modify the name so that it complies with any of the following naming conventions:

- Any other data set name that includes *userid*
- 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

To change the default name for either or both of these data sets, you need to specify the SAVESETDSN and SAVEBPDSN option in the EQAOPTS option file. The following diagram describes the SAVESETDSN and SAVEPDSN options:



For *userid-file-name*, specify the name of the data set where this information will be stored.

There are several options that you can specify in the EQAOPTS option file. To help you organize all the options you are selecting , check SAVESETDSN and SAVEBPDSN on the checklist in step 2 on page 75 of the instructions in topic Appendix A, “Defining EQAOPTS options,” on page 75. When you are done selecting all the options you want to use, follow the instructions in Appendix A, “Defining EQAOPTS options,” on page 75.

SVC screening option

In a non-CICS environment, Debug Tool requires SVC screening for the following situations:

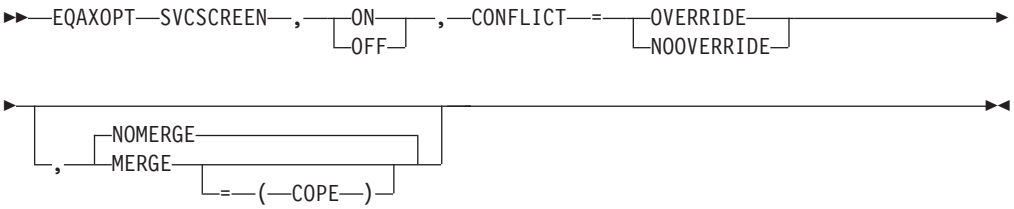
- Invoking Debug Tool by using EQANMDBG to debug programs that start outside Language Environment including non-Language Environment COBOL programs.
- Debugging programs that do not run in Language Environment and are started by programs that begin in Language Environment.
- Detecting services such as MVS LINK, LOAD and DELETE.

If you need to run Debug Tool in any of the following situations, you must specify the actions that Debug Tool must take regarding SVC screening:

- Start Debug Tool by using EQANMDBG in an environment that already uses SVC screening.
- Run Debug Tool when debugging programs that do not run in Language Environment and are started by programs that begin in Language Environment.
- Run Debug Tool when you need to detect services such as MVS LINK, LOAD and DELETE.
- Run Debug Tool in a situation that requires SVC screening and SVC screening is already in use by a program with which Debug Tool supports MERGE SVC screening as described by the MERGE operand that follows.

Syntax of the invocation of the EQAXOPT SVCSCREEN macro

The following diagram shows how to code an invocation of the EQAXOPT macro:



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

ON
Indicates that you want Debug Tool to use SVC screening in order to support MVS LOAD, DELETE, and LINK SVCs.

OFF
Indicates that you want Debug Tool to not use SVC screening. Debug Tool will not know about programs started through MVS LOAD, DELETE, and LINK SVCs.

CONFLICT=
Specifies what you want Debug Tool to do when ON is specified or defaulted and SVC screening is already used by another program.

OVERRIDE
Indicates that you want Debug Tool to override the current SVC screening and take control of SVC screening.

NOOVERRIDE
Indicates that if SVC screening is already in use, Debug Tool does not initiate SVC screening and proceeds as if OFF were specified.

NOMERGE
Indicates that SVC screening is not to be merged with SVC screening used by any other product. `NOMERGE` is the default.

MERGE
Indicates that when SVC screening is already being used by another program when Debug Tool starts, Debug Tool saves the current SVC screening environment, then enables SVC screening for both Debug Tool and the other program. When Debug Tool terminates, it restores the original SVC screening environment.

Currently, Debug Tool supports the `MERGE` option with only one other program: COPE.

If you specify the `MERGE` option and Debug Tool does not recognize the program that is using the SVC screening, the `MERGE` option is ignored and Debug Tool starts based on the value of the `CONFLICT` option.

MERGE=(COPE)
If COPE is active, Debug Tool saves the current SVC screening environment, then enables SVC screening for both Debug Tool and COPE. When Debug Tool terminates, it restores COPE's SVC screening environment.

If COPE is not active, Debug Tool starts based on the value of the `CONFLICT` option.

The default parameters for the EQAXOPT SVCSCREEN macro is one of the following situations:

- If Debug Tool is started by using the EQANMDBG program:
  `SVCSCREEN,ON,CONFLICT=NOOVERRIDE,NOMERGE`
- If Debug Tool is started by any other method:
  `SVCSCREEN,OFF,CONFLICT=NOOVERRIDE,NOMERGE`

If Debug Tool is started by using the EQANMDBG program, the `OFF` setting is ignored.

Specifying the SVC screening option

The following table shows examples of combinations of EQAXOPT SVCSCREEN parameters:

Table 2. Combination of SVSCREEN options and their effects

SVCSCREEN options	Type of Debug Tool session	Action
`OFF,CONFLICT=NOOVERRIDE` (default)	Debug Tool started by using EQANMDBG	Same as for `ON,CONFLICT=NOOVERRIDE`.
	Debug Tool started by any other method	• Debug Tool does not enable its SVC screening. • You cannot debug programs that do not run in Language Environment which were started by programs that do run in Language Environment. • Debug Tool does not detect the MVS services LINK, LOAD and DELETE. • The CONFLICT setting is ignored when the OFF setting is specified.
`OFF,CONFLICT=OVERRIDE`	Debug Tool started by using EQANMDBG	Same as for `ON,CONFLICT=OVERRIDE`.
	Debug Tool started by any other method	Same as for `OFF,CONFLICT=NOOVERRIDE`. The CONFLICT setting is ignored when the OFF setting is specified.
`ON,CONFLICT=NOOVERRIDE`	Debug Tool started by using EQANMDBG	If SVC screening is active, Debug Tool terminates. If SVC screening is not active, Debug Tool enables its SVC screening, runs the debugging session, and disables its SVC screening after the debugging session ends.
	Debug Tool started by any other method	If SVC screening is active, Debug Tool does not enable its SVC screening. You cannot debug programs that do not run in Language Environment which were started by programs that do run in Language Environment. Debug Tool does not detect the MVS services LINK, LOAD and DELETE. If SVC screening is not active, Debug Tool enables its SVC screening, runs the debugging session, and disables its SVC screening after the debugging session ends.

Table 2. Combination of SVSCREEN options and their effects (continued)

SVCSCREEN options	Type of Debug Tool session	Action
`ON,CONFLICT=OVERRIDE`	Debug Tool started by using EQANMDBG	If any SVC screening is active and the `NOMERGE` option is in effect, Debug Tool overrides the existing SVC screening. This is also the default behavior. Debug Tool enables its SVC screening, runs the debugging session, and disables its SVC screening after the debugging session ends. If any SVC screening was active, Debug Tool restores the previous SVC screening. If you specify the `MERGE` option, see the following information on `MERGE`.
	Debug Tool started by any other method	

Each user or group can control this behavior by creating their own copy of EQAOPTS with their desired options and placing it in the load module search path before *hlq*.SEQAMOD.

To set the SVC screening option, do the following steps:

1. Review "Syntax of the invocation of the EQAXOPT SVCSCREEN macro" on page 10.
2. Use Table 2 on page 12 as a guide to select the appropriate suboptions.
3. Specify the SVSCREEN option and selection suboptions in the EQAOPTS option file. There are several options that you can specify in the EQAOPTS option file. To help you organize all the options you are selecting , check EQAXOPT SVCSCREEN on the checklist in step 2 on page 75 of the instructions in topic Appendix A, "Defining EQAOPTS options," on page 75. When you are done selecting all the options you want to use, follow the instructions in Appendix A, "Defining EQAOPTS options," on page 75.

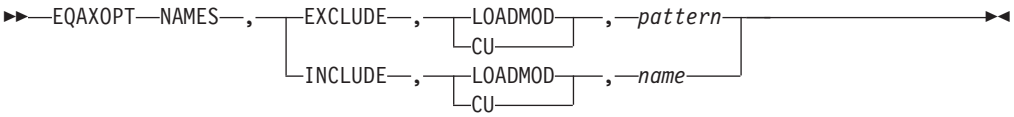
When you are done deciding on all the options and values you want to use for EQAOPTS, follow the instructions in Appendix A, "Defining EQAOPTS options," on page 75.

Supplying NAMES commands for the initial load module

The *Debug Tool User's Guide* describes how the NAMES command can be used to perform several specific functions dealing with load module and compile unit names recognized by Debug Tool. However, the NAMES command cannot be used to alter the behavior of load module or compile unit names that have already been seen by Debug Tool at the time the NAMES command is processed.

If it becomes necessary to perform these functions on the initial load module processed by Debug Tool or on any of the compile unit's contained in that load module, you must provide the information (that would otherwise have been specified using the NAMES command) in the EQAOPTS Debug Tool customization module.

One or more invocations of the EQAXOPT macro with the NAMES operand can be used for this purpose. The syntax of this macro is shown in the following diagram:



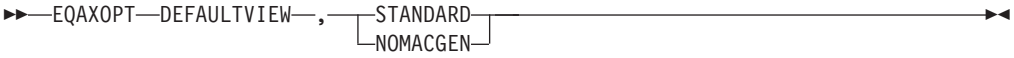
Each of these fields corresponds to the similar field in the NAMES command.

Specify the EQAXOPT NAMES option, with the names or naming patterns for load modules or compile units, in the EQAOPTS option file. There are several options that you can specify in the EQAOPTS option file. To help you organize all the options you are selecting , check EQAXOPT NAMES on the checklist in step 2 on page 75 of the instructions in topic Appendix A, "Defining EQAOPTS options," on page 75. When you are done selecting all the options you want to use, follow the instructions in Appendix A, "Defining EQAOPTS options," on page 75.

Setting the initial value for SET DEFAULT VIEW

The default view used when a LOADDEBUGDATA command is issued for an assembler CU can be set using the SET DEFAULT VIEW command. It is possible to use the EQAXOPT Debug Tool customization module to specify the initial value to be used for SET DEFAULT VIEW.

The following invocation of the EQAXOPT macro with the DEFAULTVIEW operand can be used for this purpose. The syntax of this macro is shown in the following diagram:



Each of these fields corresponds to the similar field in the SET DEFAULT VIEW command. If EQAXOPT DEFAULTVIEW is not coded, the initial setting for DEFAULTVIEW is STANDARD.

Specify the EQAXOPT DEFAULTVIEW option in the EQAOPTS option file, indicating which view you want used as the default view. There are several options that you can specify in the EQAOPTS option file. To help you organize all the options you are selecting , check EQAXOPT NAMES on the checklist in step 2 on page 75 of the instructions in topic Appendix A, "Defining EQAOPTS options," on page 75. When you are done selecting all the options you want to use, follow the instructions in Appendix A, "Defining EQAOPTS options," on page 75.

Modifying Debug Tool behavior when requested user interface is not available

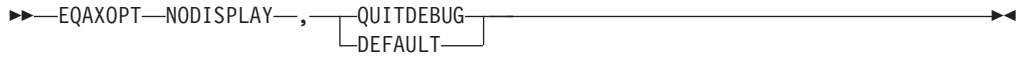
In the following two situations, in which a user can request a specific user interface, that interface may not be available:

- full-screen mode through a VTAM terminal (with or without the Terminal Interface Monitor). If the terminal is not available, the program being debugged terminates with a U4038 abend.
- remote debugger. If the remote debugger is not available, Debug Tool will use full-screen mode if the user is running under TSO. If the user is not using TSO, Debug Tool will use batch mode.

In both cases, Write To Operator (WTO) messages also appear.

You can modify these behaviors by specifying the EQAXOPT NODISPLAY option so that Debug Tool continues processing as if the user immediately entered a `QUIT DEBUG` command. This modification prevents any forced abend or the debugger from starting, which is often preferable.

The following invocation of the EQAXOPT macro with the `NODISPLAY` operand can be used for this purpose. The syntax of this macro is shown in the following diagram:



DEFAULT
Debug Tool follows the default behavior.

QUITDEBUG
Debug Tool displays a message that indicates that Debug Tool will quit, and that the user interface could not be used. Debug Tool processing continues as if the user entered a `QUIT DEBUG` command.

Specify the EQAXOPT NODISPLAY option in the EQAOPTS option file, indicating which behavior you want Debug Tool to display. There are several options that you can specify in the EQAOPTS option file. To help you organize all the options you are selecting , check EQAXOPT NODISPLAY on the checklist in step 2 on page 75 of the instructions in topic Appendix A, “Defining EQAOPTS options,” on page 75. When you are done selecting all the options you want to use, follow the instructions in Appendix A, “Defining EQAOPTS options,” on page 75.

Specifying SUBSYS to access source code in a library system

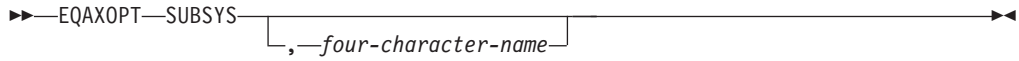
This topic describes when and how to specify the SUBSYS allocation parameter in the EQAOPTS option file.

If the following conditions apply at your site, you need specify the SUBSYS=*library_subsystem_name* allocation parameter in the EQAOPTS option file:

- The source code is managed by a library system that requires that you specify the SUBSYS=*library_subsystem_name* allocation parameter when you allocate a data set.
- Your users are debugging C, C++, or Enterprise PL/I programs compiled without the `SEPARATE` suboption of the `TEST` compiler option.

You must run Debug Tool and the specified subsystem on the same system. You cannot use this feature to debug programs that run under CICS.

The following diagram describes the EQAXOPT SUBSYS option:



Specify the EQAXOPT SUBSYS option in the EQAOPTS option file, specifying the four character name of the subsystem library. There are several options that you can specify in the EQAOPTS option file. To help you organize all the options you

are selecting , check EQAXOPT SUBSYS on the checklist in step 2 on page 75 of the instructions in topic Appendix A, “Defining EQAOPTS options,” on page 75. When you are done selecting all the options you want to use, follow the instructions in Appendix A, “Defining EQAOPTS options,” on page 75.

Suppressing the prompt Debug Tool displays for FINISH, CEE066, or CEE067 conditions

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.

The following diagram describes the syntax of the THREADTERMCOND option:

Specify the EQAXOPT THREADTERMCOND option in the EQAOPTS option file, specifying NOPROMPT to suppress the prompt when these conditions are raised. There are several options that you can specify in the EQAOPTS option file. To help you organize all the options you are selecting , check EQAXOPT THREADTERMCOND on the checklist in step 2 on page 75 of the instructions in topic Appendix A, “Defining EQAOPTS options,” on page 75. When you are done selecting all the options you want to use, follow the instructions in Appendix A, “Defining EQAOPTS options,” on page 75.

Specifying a code page

This topic describes the steps you need to take to do the following tasks:

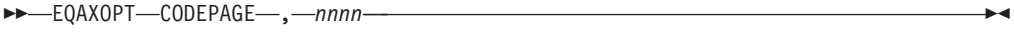
- Debug programs in remote debug mode that contain NLS characters.
- Use the STORAGE command to update COBOL NATIONAL variables.
- Properly display C/C++ variables that contain NLS characters

The default code page used by Debug Tool and the remote debuggers is 037. If you need to use a different code page, you can do one of the following tasks:

- For full-screen mode or remote debug mode, you can specify a different code page with the CODEPAGE option of the EQAOPTS option file.
- For remote debug mode, users can use the VADSCP*nnnnn* suboption of the TEST runtime option to specify a different code page.

In remote debug mode, if you do both tasks, the code page specified in the CODEPAGE option overrides the CCSID specified in the VADSCP*nnnnn* suboption.

The following diagram describes the syntax of the EQAXOPT CODEPAGE option:



Specify the EQAXOPT CODEPAGE option in the EQAOPTS option file with the code page you want used. There are several options that you can specify in the EQAOPTS option file. To help you organize all the options you are selecting , check EQAXOPT CODEPAGE on the checklist in step 2 on page 75 of the instructions in topic Appendix A, “Defining EQAOPTS options,” on page 75. When

you are done selecting all the options you want to use, follow the instructions in Appendix A, “Defining EQAOPTS options,” on page 75.

Chapter 3. Customizing Debug Tool Utilities functions shipped with Debug Tool

Debug Tool Utilities is a utility that brings together tools provided by Debug Tool and Debug Tool Utilities and Advanced Functions. If you order only Debug Tool, you receive Debug Tool Setup Utility, which manages setup files. Setup files help application programmers prepare programs to debug them interactively or in batch mode. If you also order Debug Tool Utilities and Advanced Functions, you receive the following tools and functions:

- Program Preparation Utilities to help application programmers precompile, compile, and link their programs and then start Debug Tool.
- Coverage Utility to help application programmers conduct coverage tests on their programs.
- COBOL and CICS Command Level Conversion Aid (CCCA) to help application programmers convert older COBOL programs to Enterprise COBOL programs. See *COBOL and CICS Command Level Conversion Aid for OS/390 & MVS & VM: User's Guide* for a list of older COBOL compilers supported by CCCA.
- Load Module Analyzer helps you analyze load modules to determine the language translator that was used to compile or assemble each CSECT in the load module.
- Manage TEST Run-time Option Data Set helps you edit a TEST runtime option data set that the Debug Tool Language Environment user exit routines uses to start a debug session.
- Manage IMS Programs can help you create and maintain setup files and submit batch jobs to create a private message region or to run a BMP program. If you are running IMS Version 8 (or later), you can also manage your Language Environment run-time options.

If you have only Debug Tool, do the instructions in this section and you can skip Chapter 4, "Customizing Debug Tool Utilities functions shipped with Debug Tool Utilities and Advanced Functions," on page 25. If you have Debug Tool Utilities and Advanced Functions, do the instructions in Chapter 4, "Customizing Debug Tool Utilities functions shipped with Debug Tool Utilities and Advanced Functions," on page 25 after you have completed the instructions in this section.

The instructions in this section describe the following customization tasks:

- Modify the TSO logon procedure so that your users can start Debug Tool Utilities by using the `EQASTART` command.
- 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 Problem Determination Tools interface.

Choosing a method to start Debug Tool Utilities

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

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

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

1. Include or copy the Debug Tool Utilities data sets to your system's TSO logon data sets. To add the data sets, do one of the following alternatives:
   - Include the data sets listed in Table 3, Table 4, or Table 5 on page 21 into the DD concatenations specified in the tables.
   - Copy[3] the members of the data sets listed in Table 3, Table 4, or Table 5 on page 21 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 7.
2. Edit the EQASTART 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 7.
3. Inform your users how to specify a language other than the one selected in step 2. If your users need to start Debug Tool in a language other than the default, they need to add the NATLANG(*xxx*) parameter to the EQASTART command.

Table 3. For English, data sets that need to be included or copied into the specified DD concatenations

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

Table 4. For uppercase English, data sets that need to be included or copied into the specified DD concatenations

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

Table 5. For Japanese, data sets that need to be included or copied into the specified DD concatenations

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

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

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

Customizing the data set names in EQASTART

You must modify member EQASTART of the *hlq*.SEQAEXEC data set to specify the data set names that you chose at installation time. Edit the EQASTART 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 and OS/390
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 DDnames (as described in Method 2 in "Choosing a method to start Debug Tool Utilities" on page 19), then change line **1** to the following:

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

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

Customizing Debug Tool Setup Utility

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

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

1. Edit the EQAZPROC 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 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 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 for Problem Determination Tools for multiple systems

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

- Modify `EQASTART` 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)'`

Chapter 4. Customizing Debug Tool Utilities functions shipped with Debug Tool Utilities and Advanced Functions

Debug Tool Utilities and Advanced Functions adds the following tools and functions to Debug Tool Utilities:

- Program Preparation Utilities to help application programmers precompile, compile, and link their programs and then start Debug Tool.
- Coverage Utility to help application programmers conduct coverage tests on their programs.
- COBOL and CICS Command Level Conversion Aid (CCCA) to help application programmers convert older COBOL programs to Enterprise COBOL programs. See *COBOL and CICS Command Level Conversion Aid for OS/390 & MVS & VM: User's Guide* for a list of older COBOL compilers supported by CCCA.
- Load Module Analyzer helps you analyze load modules to determine the language translator that was used to compile or assemble each CSECT in the load module.
- Manage TEST Run-time Option Data Set helps you edit a TEST runtime option data set that the Debug Tool Language Environment user exit routine uses to start a debug session.
- Manage IMS Programs can help you create and maintain setup files and submit batch jobs to create a private message region or to run a BMP program. If you are running IMS Version 8 (or later), you can also manage your Language Environment run-time options.

Do the steps described in this section only if you installed Debug Tool Utilities and Advanced Functions, and after you have completed the instructions in Chapter 3, "Customizing Debug Tool Utilities functions shipped with Debug Tool," on page 19.

Customizing the Program Preparation Utilities

The Program Preparation Utilities help 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` 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
- yccchga1
- yccwrka1
- ycctkna1

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

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

Parameters you can set

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

1 COBOL compiler parameters

3 PL/I compiler parameters

4 C and C++ compiler parameters

5 Assembler parameters

L Link Edit parameters

c CCCA parameters

F Fault Analyzer parameters

G Fault Analyzer listing create parameters

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

The last five characters correspond to the parameter:

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

cicmd Load module name for CICS translator.

cicps CICS translator options.

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

cmod Load module name for the compiler or utility.

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

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

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

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

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

cttl Title for the compiler.

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

db2md
Load module name for DB2 precompiler.

db2ps DB2 precompiler options.

flg Enable or disable the compiler or development utility.

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

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

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

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

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

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

/U Current TSO user ID.

/P Current TSO profile prefix.

sds1 Shared data set prefix for CCCA.

svs1 Shared VSAM data set prefix for CCCA.

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

Customizing Preparation Utilities for multiple systems

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

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

Customizing Coverage Utility

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

Setting up the APF-authorized non-link list data sets

This section describes where to place certain Coverage Utility load modules so that the correct people have access to them.

Placing Coverage Utility load modules in an APF-authorized data set accessible to all users

Certain Coverage Utility load modules must be placed in an APF-authorized data set that is accessible to all your users. The APF-authorized data set does not need to be in the link list.

1. Make the load modules in *hlq*.SEQAMOD accessible to all users by using one of the following alternatives:
   - Mark the *hlq*.SEQAMOD data set as APF-authorized[1] and make it accessible to all users by creating a Resource Access Control Facility (RACF) profile.
   - Do not mark the *hlq*.SEQAMOD data set as APF-authorized. Copy[3] the following load modules to an APF-authorized data set that all users can access.
     - EQACUOCM (monitor interface)
     - EQACU9M0 (monitor messages)
     - EQACU9M1 (monitor messages)
     - EQACU9M2 (monitor messages and only if the Japanese feature is installed)
     - EQACU9M3 (monitor messages and only if the Korean feature is installed)
2. Add the EQACUOCM program to the AUTHPGM entry in the member IKJTSOxx of the SYS1.PARMLIB data set.
3. Issue the PARMLIB UPDATE(xx) command from TSO or IPL your system.
4. Edit the EQASTART member of the *hlq*.SEQAEXEC data set and set the *INST_Auth_SEQAMOD* variable to the name of the data set from step 1 that contains EQACUOCM.

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

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

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

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

Creating RACF profiles

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

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

SETROPTS WHEN(PROGRAM) REFRESH

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

SETROPTS WHEN(PROGRAM) REFRESH
```

The above code restricts 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 Users Guide* for more information on the amount of ECSA storage used by each user session.

CSA 13248 bytes

SQA 25496 bytes

Perform the following steps to:

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

1. Reserve two free *user* SVC numbers. User SVC numbers must be in the range 200 to 255 (X'C8' to X'FF'). Verify that these SVC numbers are not being used on your system. SYS1.PARMLIB(IEASVC*xx*) does not need to be updated since these user SVCs can only be installed dynamically. However, for future reference, add a comment to IEASVC*xx* to indicate that these SVCs are used.
2. Copy *hlq*.SEQASAMP(EQACUOPS) to your SYS1.PROCLIB data set as member EQACUOIN. Make the following edits to the new EQACUOIN member:
   a. Change the STEPLIB data set name to the name of the APF-authorized data set that contains the EQACUOIN and EQACUOSV modules.
   b. Change the PARM operands to contain the two user SVC numbers (in hexadecimal notation) that you reserved for Coverage Utility. Verify that you typed these numbers correctly.
3. Use the PERMIT commands, as described in "Creating RACF profiles" on page 29, to give the process started by EQACUOIN access to the EQACUOIN and EQACUOSV load modules. The process started by EQACUOIN is assigned an ID by the RACF started procedures table or STARTED class. Use this ID as the value for the *id* variable of the ID parameter of the PERMIT command.
4. The SYS1.PARMLIB(COMMND*xx*) data set contains the names of programs to start at IPL time. Add the following line to the COMMND*xx* member of the SYS1.PARMLIB data set:

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

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

   ```
   S EQACUOIN
   ```

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

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

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

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

   ```
   BROWSE    SMITH.MSGS.FILE                              Line 00000000 Col
   Command ===>                                                 Scroll ==
   ****************************** Top of Data ******************************
   Monitor Release: V8R1M0 Date: 2002.245
   MAST: 00F9B8E8 PSA: 00F87000 CPU: 00F87000 SEST: 00F9BCE8 UNID: 00000000
   ```

Customizing the Coverage Utility defaults

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

1. Change all occurrences of EQAW to *hlq*. For example, to use the high-level qualifier EQAW.V8R1M0, change all occurrences of EQAW to EQAW.V8R1M0.

2. In the execute step data entry for EXEJOBLB, change the name *hlq*.SEQAMOD to the name of the APF-authorized data set you used for the EQACUOCM program identified in "Placing Coverage Utility load modules in an APF-authorized data set accessible to all users" on page 28.
3. Enter the Coverage Utility Monitor SVC numbers (in hexadecimal notation) in the CUSVC2B and CUSVC4B entries.
4. 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.
5. 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.
6. 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)'
7. If you do not want Coverage Utility to generate or build any data set names automatically, set the USEPRGNM variable to N.

Configuring for IMSplex users

To determine if you need to do the steps described in this topic, read "Preparing IMS programs" in *Debug Tool User's Guide*. If your users use the Manage IMS Programs - 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 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

Configuring for debugging Q++ programs

This topic describes one of the tasks you need to do to enable Debug Tool to debug MasterCraft Q++ programs, provided by Tata Consultancy Services Ltd. For more information on how to enable Debug Tool to support MasterCraft Q++, contact Tata Consultancy Services Ltd.

The following diagram describes the syntax of the EQAXOPT EQAQPP option:

Specify the EQAXOPT EQAQPP option in the EQAOPTS option file with either ON or OFF. There are several options that you can specify in the EQAOPTS option file. To help you organize all the options you are selecting , check EQAXOPT EQAQPP on the checklist in step 2 on page 75 of the instructions in topic Appendix A, “Defining EQAOPTS options,” on page 75. When you are done selecting all the options you want to use, follow the instructions in Appendix A, “Defining EQAOPTS options,” on page 75.

Chapter 5. Enabling debugging in full-screen mode through a VTAM terminal

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

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

How Debug Tool uses VTAM in full-screen mode through a VTAM terminal

The following steps describe how a user would start a debugging session for a batch job using full-screen mode through a VTAM terminal. Study these steps to understand how Debug Tool uses VTAM in full-screen mode through a VTAM terminal and to understand why you need to do the configuration steps described in "The steps for enablement" on page 34.

1. Start two terminal emulator sessions. Connect the second session to a terminal LU that can handle a full-screen mode debugging session through a VTAM terminal.
2. On the first terminal emulator session, log on to TSO.
3. Note the LU name to which the second terminal emulator session is connected. If the second session displays a session manager screen, exit from the session manager.
4. Edit the `PARM` string of your batch job so that you specify the `TEST` run time parameter in the following format:

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

   VTAM_LU_id is the VTAM LU name to which the second terminal emulator session is connected.

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

   ```
   TEST(,,MFI%network_identifier.VTAM_LU_id:*)
   ```

   network_identifer is optional and identifies the network in which the second terminal emulator resides.

   Place a slash (/) before or after the parameter, depending on your programming language.
5. Submit the batch job. The following tasks are completed:
   a. Debug Tool allocates a VTAM minor node `ACB (EQAMVnnn)` for its end of a VTAM conversation.
   b. Debug Tool uses VTAM to initiate a conversation with the terminal LU to which the second terminal emulator session is connected. In particular it will acquire the terminal LU and do a `SIMLOGON` from it.
   c. A VTAM conversation is then conducted between the Debug Tool minor node and the terminal LU.

The user does not logon to any host application through the second terminal emulator session. Debug Tool initiates the connection between itself and the terminal LU to which the second session is connected.

6. On the second terminal emulator session, a full-screen mode debugging session is displayed. 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 VTAM for Debug Tool, in VTAM for the terminal definitions, and in TCP/IP (if the TN3270 server is used to manage the terminal). Section "The steps for enablement" describes these definitions and configuration.

The steps for enablement

To enable full-screen mode through a VTAM terminal, do the following steps:

1. Define the VTAM minor nodes that Debug Tool uses for its end of the conversation with the terminal LU.
2. Define the terminals used by Debug Tool.
3. Configure the TN3270 Telnet Server.
4. Verify the installation of the facility to debug programs in full-screen mode through a VTAM terminal.

Defining the VTAM EQAMVnnn minor nodes

You must define the minor nodes that Debug Tool uses for its end of the VTAM conversation with the terminal LU. You can define up to 999 minor nodes for Debug Tool. You can define a minor node by using one of the following naming conventions:

- Define each minor node with the following naming convention: the first five characters of the minor node name must be `EQAMV` and the last three characters must be consecutive three digit numbers, starting with 001.
- Define each minor node 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`* minor node names are used internally by Debug Tool. Do not confuse these node names with the terminal minor node (LU) names that define the display terminal. The user needs to know only the terminal LU name of the display terminal, which he specifies with the `MFI%` sub-option of the `TEST` run time option.

The number of minor node names you define must be sufficient to allow for the maximum number of concurrent Debug Tool full-screen mode through a VTAM terminal sessions. (Debug Tool uses one of these minor node 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 minor node names by using the `EQAMV`*`nnn`* naming convention. Debug Tool uses the `EQAMV`*`nnn`* minor node names for internal processing.

The `EQAWAPPL` member in the *`hlq`*`.SEQASAMP` data set predefines 20 minor node names, `EQAMV001` to `EQAMV020`. You can copy `EQAWAPPL` into a new member or into an existing member in the VTAM definitions library (`VTAMLST`).

- To 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*.
- To copy EQAWAPPL into an existing member:
  1. Select a member in the VTAM definitions library (VTAMLST) that contains the major node definitions.
  2. Copy the minor node name definitions (APPL statements) for Debug Tool from the EQAWAPPL member into the selected member.

If you are running VTAM in a sysplex or a VTAM multi-domain environment and you require the ability to debug full-screen mode through a VTAM terminal on more than one host in the sysplex, 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 20 concurrent users debugging programs in full-screen mode through a VTAM 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
...
EQAMV020 APPL  AUTH=(PASS,ACQ),PARSESS=NO
```

- SYSB VTAMLST EQAWAPPL entry:

```
EQAAPPLB VBUILD TYPE=APPL
EQAMV021 APPL  AUTH=(PASS,ACQ),PARSESS=NO
EQAMV022 APPL  AUTH=(PASS,ACQ),PARSESS=NO
...
EQAMV040 APPL  AUTH=(PASS,ACQ),PARSESS=NO
```

- SYSC VTAMLST EQAWAPPL entry:

```
EQAAPPLC VBUILD TYPE=APPL
EQAMV041 APPL  AUTH=(PASS,ACQ),PARSESS=NO
EQAMV042 APPL  AUTH=(PASS,ACQ),PARSESS=NO
...
EQAMV060 APPL  AUTH=(PASS,ACQ),PARSESS=NO
```

You can have up to 999 unique minor node names for full-screen mode through a VTAM terminal spread across the sysplex.

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 minor nodes" on page 34):

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

Activating the VTAM EQAMVnnn minor nodes

Activate the VTAM minor nodes by entering the following command from the console, where *member-name* is the member name in the VTAM library:

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

If you used the Model Application Names function, enter the one of the following commands from the console:

- `DISPLAY NET,E,ID=EQAMV001`
- `DISPLAY NET,E,ID=ABCDE001`, where `ABCDE001` is the first application node name. Use this command if you used something other than `EQAMV`*`nnn`* for the application node name.

Defining terminal LUs used by Debug Tool

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

"Terminal LU specifications"

"Terminal LU state requirements"

Terminal LU specifications

All terminal LUs that are used to debug programs in full-screen mode through a VTAM terminal must have a default `LOGMODE` specified in the corresponding VTAM definitions. This `LOGMODE` 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 `LOGMODE`.

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 logmode that matches your terminal emulator session characteristics. For example, if you have a TN3270E emulator that does not respond to a query, select one of the following logmodes 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 conversation is *`hlq`*`.SEQASAMP(EQAWTRML)`. See the logon mode definitions in the *IBM Communications Server SNA Resource Definition Reference (SC31-8778)* for further information about log modes. The `MODETAB` logon mode table load module that contains the `DLOGMOD` default log on 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 minor nodes" on page 35.

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 system on which the application 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 session, 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

This section applies if you use the IBM Communications Server for z/OS TN3270 Telnet Server.

If you use this TN3270 Telnet Servers to manage your terminals, you can set up a new TN3270 telnet port that will allow the TN3270 Telnet Server to support the requirements stated previously. The particular requirements are as follows:

- 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.

The following changes guide you though configuring such a port. For reference you might also want to refer ahead to working examples of this new port shown in the next section.

1. Select an unused port, such as 2023. If you have a firewall installed, ensure that this port is allowed through the firewall.
2. Do one of the following steps:
   - If you are running the TN3270 Telnet Server in the TCP/IP address space, specify a PORT *num* TCP INTCLIEN statement to reserve the new port for the TN3270 Telnet Server.
   - If you are running the TN3270 Telnet Server in a separate address space, specify a PORT *num* TCP *jobname* NOAUTOLOG statement to reserve the new port for the TN3270 Telnet Server.
3. Create a new set of TELNETPARMS and BEGINVTAM blocks for the new port by copying the existing TELNETPARMS and BEGINVTAM blocks for port 23.
4. Customize the new TELNETPARMS and BEGINVTAM blocks to use this new port number. Ensure that the previous TELNETPARMS and BEGINVTAM blocks also specify a port number (typically 23).
5. Make the following changes to your new BEGINVTAM block:

a. If you intend to use this new port for only Debug Tool in full screen-mode through a VTAM terminal, you can remove all the statements from the BEGINVTAM block that you created in step 3 on page 37, 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 VTAM terminals for debugging in full-screen mode through a VTAM 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 do a SIMLOGON of an application that is named `EQAMV`*`nnn`* from the terminal LU.

   If you defined the minor node name that Debug Tool uses for its side of the VTAM conversation with a node name other than `EQAMV`*`nnn`*, then you should specify that node 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 through a VTAM 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 on 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 conversation.
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.

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 (SC31-8775)* for more information on 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 through a VTAM terminal.

Example: Activating full-screen mode through a VTAM 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
...
EQAMV020 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:

```
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
...
TRMLU020 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 41, and "Example 3" on page 41.

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*%VTAM_LU_id* sub-option 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 37.

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

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

BEGINVTAM
 4    PORT 2023


   LUGROUP DBGTOOL
 5c       TRMLU001..TRMLU020
   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%*VTAM_LU_id* suboption of the TEST run-time parameter. 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 37.

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

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

BEGINVTAM
4    PORT 2023


   LUGROUP DBGTOOL
5c       TRMLU001..TRMLU020
   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%*VTAM_LU_id* suboptions of the TEST run-time parameter.

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 37.

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

;
```

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

BEGINVTAM
 4    PORT 2023


   LUGROUP DBGTOOL
 5c       TRMLU001..TRMLU020
   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 through a VTAM terminal

Connect a terminal emulator session to the new telnet port to one of the terminal LUs setup as described above. Issue the DISPLAY command from your system console as shown in "Terminal LU state requirements" on page 36. Verify that the output of the DISPLAY command is correct. Then run one of the install verification jobs described below.

To help you verify the installation of the facility to debug full-screen mode through a VTAM 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)
- If you have installed Debug Tool Utilities and Advanced Functions:
  - EQAZIVPB (Language Environment assembler)
  - EQAZIVPD (non-Language Environment assembler)
  - EQAZIVPW (OS/VS COBOL)
  - EQAZIVPY (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 use full-screen mode through a VTAM terminal without having to know the LU name of the VTAM terminal. The Terminal Interface Manager provides a method to correlate a user ID to the terminal. This is useful in situations where it is cumbersome to edit the TEST run time parameter with the LU name of the VTAM terminal.

Complete the steps in "The steps for enablement" on page 34 before you do the instructions in this section to ensure that the basic full-screen mode through a VTAM 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 VTAM in full-screen mode through a VTAM terminal" on page 33 to understand how using the Terminal Interface Manager affects the flow of work.

1. Start two terminal emulator sessions. Connect the second session to a terminal that starts the 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  PF12=LOGOFF
```

   The terminal is now ready to receive a Debug Tool full-screen mode through a VTAM terminal session.
4. Edit the PARM string of your batch job so that you specify the TEST run time parameter as follows:

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

   Place a slash (/) before or after the parameter, depending on your programming language. *userid* is the TSO user ID that you provided to the Terminal Interface Manager.
5. Submit the batch job.

   The tasks completed are similar to the tasks described in step 5 on page 33 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 33.

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 43. 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 on page 43. 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.

The steps for enablement

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

1. Define the VTAM minor node as described in "Defining the Terminal Interface Manager VTAM minor node."
2. Start the Debug Tool Terminal Interface Manager as described in "Starting the Debug Tool Terminal Interface Manager" on page 45.
3. Configure the Telnet Server as described in "Configuring the TN3270 Telnet Server to access the Terminal Interface Manager" on page 45.
4. Verify that the customizations are completed correctly by following the steps in "Verifying the customization of the Terminal Interface Manager" on page 47.

Defining the Terminal Interface Manager VTAM minor node

You must define the VTAM minor node that the Terminal Interface Manager will use for its conversations. To define the VTAM minor node, do the following steps:

1. Define the minor node as shown in the EQAWSESS member in the *hlq*.SEQASAMP data set by doing one of the following:
   - 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 minor node name definition (APPL statement) 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 Inteface manager load module, EQAYSESM, resides in an APF authorized library (this module can be found in the *hlq*.SEQAAUTH data set). This is required to allow access to functions to validate users by TSO user ID and password.
3. Start the Debug Tool Terminal Interface Manager using the START command from the console. The START command can be added to the COMMND*xx* member of SYS1.PARMLIB to start the Debug Tool Terminal Interface Manager when the system is IPLed.

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 40 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 40, with the changes highlighted with an asterisk (*).

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

; Add a TELNETPARMS block for the new port

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

; Add a BEGINVTAM block for the new port

BEGINVTAM
 *   PORT 2024

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

  LUGROUP DBGTOOL
      TRMLU001..TRMLU020
  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 39.

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 through a VTAM 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. A Telnet Solicitor Logon panel is displayed. The terminal is now ready to receive a Debug Tool full-screen mode through a VTAM 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 through a VTAM 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 through a VTAM terminal debugging session.
- VTAM on LPAR1 defines the terminal LU applids and the EQASESSM applid for the Terminal Interface Manager.
- VTAM on LPAR 1 needs visibility to the EQAMVnnn applids 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 applids. For example, LPAR 1 has applids 001-050, LPAR 2 has applids 051-100, LPAR 3 has applids 101-150, and LPAR 4 has applids 151-200.
- Each VTAM on LPAR 2, 3 and 4 needs visibility to the EQASESSM applid 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 applids on LPAR 1.

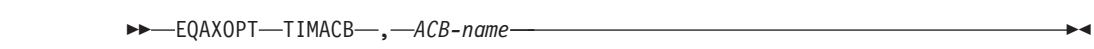
Running the Terminal Interface Manager on more than one LPAR on the same VTAM network

This topic describes the modifications you need to make to the steps described in "Defining the Terminal Interface Manager VTAM minor node" on page 44, "Starting the Debug Tool Terminal Interface Manager" on page 45, and "Configuring the TN3270 Telnet Server to access the Terminal Interface Manager" on page 45 in order to make full-screen mode through a VTAM terminal with Terminal Interface Manager work in an environment where you want to run the Terminal Interface Manager on more than on LPAR in the same VTAM network. It also describes the modification you must make to the EQAOPTS option file to complete this modification.

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

1. In "Defining the Terminal Interface Manager VTAM minor node" on page 44, 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 45, 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 45, when you modify the TCP/IP TN3270 server's configuration file, modify the DEFAULTAPPL statement to specify the ACB name you created in step 1, instead of EQASESSM.
4. Specify the EQAXOPT TIMACB option in the EQAOPTS option file. The following diagram describes the syntax of this option, where *ACB-name* is the new ACB name you created in step 1:

   Place this customized EQAOPTS module in the load module search path in front of *hlq*.SEQAMOD for the Debug Tool users who are using this new instance of the Terminal Interface Manager.

   There are several options that you can specify in the EQAOPTS option file. To help you organize all the options you are selecting , check EQAXOPT TIMACB on the checklist in step 2 on page 75 of the instructions in topic Appendix A, "Defining EQAOPTS options," on page 75. Specify an instance of this option for eveyr ACB name you created in step 1. When you are done selecting all the options you want to use, follow the instructions in Appendix A, "Defining EQAOPTS options," on page 75.

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.

Chapter 6. Enabling the EQAUEDAT user exit

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

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

To enable this user exit, do the following steps:

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

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

Chapter 7. Preparing your environment to debug a DB2 stored procedures

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

- `LOADLIB` for the stored procedure
- `SEQAMOD` for Debug Tool
- `SCEERUN` 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

Chapter 8. Adding support for debugging under CICS

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

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

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

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

To add Debug Tool support for CICS applications:

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

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

      Review the instructions in both members and run the batch job to add the definitions to your CICS CSD.
2. Update the JCL that starts CICS:
   a. Include Debug Tool's *`hlq`*`.SEQAMOD` data set and the Language Environment run-time libraries (`SCEECICS`, `SCEERUN`, and, if required by your applications, `SCEERUN2`) in the `DFHRPL` concatenation. The `DFHRPL` concatenation is in the CICS region start-up JCL.
   b. Remove any data sets from the concatenation that refer to old releases of Debug Tool.
   c. Include `EQA00DYN` and `EQA00HFS` from Debug Tool's *`hlq`*`.SEQAMOD` data set in the `STEPLIB` concatenation by either of the following ways:
      - Use the Authorized Program Facility (APF) to authorize[1] the *`hlq`*`.SEQAMOD` data set and add the data set to the STEPLIB concatenation.
      - Copy[3] the `EQA00DYN` and `EQA00HFS` modules from the *`hlq`*`.SEQAMOD` data set to a library that is already in the `STEPLIB` concatenation.
   d. Ensure that no DD statements exist for `CINSPIN`, `CINSPLS`, `CINSPOT`, `IBMDBGIN`, or `IGZDBGIN`.
3. For any terminal that Debug Tool uses to display a debugging session, verify that the CICS TYPETERM definition for that terminal specifies a minimum value of 4096 for the `RECEIVESIZE` parameter or sets the `BUILDCHAIN` parameter to YES.
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 in conjunction with the CICS start up parameter INITPARM=(EQA0CPLT='NWP').
   c. If you are using DTCN, ensure that the region shares the EQADTCN2 temporary storage queue (TSQ). See "Sharing DTCN repository profile items among CICS systems" on page 57 for more information on defining the region that owns EQADTCN2. The most common multi-region debugging scenario is where the EQADTCN2 temporary storage queue 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 55 for information about EQA0CPLT), or use the Debug Tool DTCP transaction to start or stop this support as needed. In the AOR, enter DTCPO on a clear CICS screen to activate this support and enter DTCPF to deactivate it. You can activate and deactivate this support multiple times.
   d. If you are using CADP for debugging profiles, set the startup parameter DEBUGTOOL=YES for any region where a Debug Tool session might start. This parameter activates the Debug Tool new program support.
6. If users need to debug Enterprise PL/I for z/OS, Version 3 Release 4 (or later), applications under CICS:
   a. Install the following co-requisites:
      - The PTF for the Language Environment run time APAR PK03093, which is available for z/OS Version 1 Release 3 through Version 1 Release 6.
      - The PTF for the Enterprise PL/I for z/OS, Version 3 Release 4, compiler 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 in conjunction with the CICS start up parameter INITPARM=(EQA0CPLT='NWP'). See step 9 on page 55 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 56.
8. If your CICS region is started with the SEC parameter set to YES and the XCMD parameter is set to YES to activate command security, review the access settings for the following resources:

EXITPROGRAM
Do one of the following options:

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

  See step 9 for instructions on using EQA0CPLT.

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

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

For more information about the CICS security features, see *CICS RACF Security Guide*.

9. (Optional) Set up the CICS PLTPI program called EQA0CPLT:
   a. Add the program EQA0CPLT to your Program List Table (PLTPI). EQA0CPLT initializes parts of Debug Tool during CICS startup as indicated by a CICS INITPARM system initialization parameter. Run EQA0CPLT as a Stage 2 or Stage 3 PLTPI program. The following sample PLT includes EQA0CPLT:

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

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

      NLE
      Non-Language Environment support. See "Activating CICS non-Language Environment exits" on page 56.

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

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

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

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

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

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

    Important: Users must allocate files with the correct attributes to optimize the performance of Debug Tool.
11. (Optional) Increase the DSALIM and EDSALIM sizes in your CICS region so that Debug Tool functions properly with multiple concurrent users. The amount of increase is based on the current workload in the CICS region.

    Recommendation: Increase the sizes of SALIM 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 supercedes the function provided earlier by PLTPI program EQANCPLT. See step 9 on page 55 for instructions on using EQA0CPLT. For more information on PLT processing, see the *CICS Resource Definition Guide*.

Sharing DTCN repository profile items among CICS systems

The DTCN debug profile repository is a CICS temporary storage queue called EQADTCN2. If you want to share the repository among CICS systems (for example, MRO), designate a single CICS region as the *queue-owning region* and define the queue as REMOTE in a TSMODEL resource definition (or temporary storage table) on regions that need to access it remotely. This makes the queue profile items owned by one CICS system accessible to other CICS systems.

The following sample resource definition shows how to define the Debug Tool EQADTCN2 temporary storage queue in a region that will use it remotely. To optimize the performance of Debug Tool, it is important that you define this queue as Location 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        ==>
```

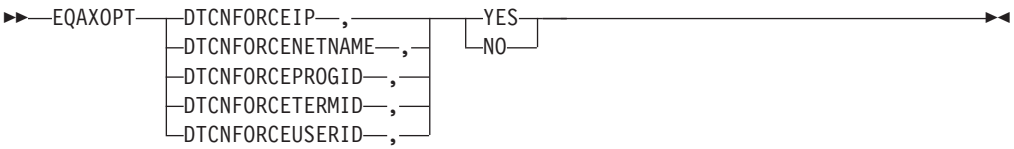
For details on defining a TSMODEL resource, see *CICS Resource Definition Guide*.

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. The following list describes each resource type you can requires your users to specify:

- DTCNFORCEIP, which requires the user to specify the IP name or address.
- DTCNFORCENETNAME, which requires the user to specify the four character name of a CICS terminal or a CICS system.
- DTCNFORCEPROGID, which requires the user to specify the name of a program or programs he wants to debug.
- DTCNFORCETERMID, which requires the user to specify the CICS terminal.
- DTCNFORCEUSERID, which requires the user to specify a user ID.

The syntax of this option is described in the following diagram:



Specify which of these EQAXOPT options in the EQAOPTS option file you want to require your users to specify. There are several options that you can specify in the EQAOPTS option file. To help you organize all the options you are selecting, check each EQAXOPT option you wan to use on the checklist in step 2 on page 75 of the instructions in topic Appendix A, “Defining EQAOPTS options,” on page 75. When you are done selecting all the options you want to use, follow the instructions in Appendix A, “Defining EQAOPTS options,” on page 75.

Overriding the default number of program elements held in cache

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

The following diagram describes the syntax of this option:

To make this customization, check EQAXOPT CACHENUM on the checklist in step 2 on page 75 of the instructions in topic Appendix A, “Defining EQAOPTS options,” on page 75. Write in the new value in the blank provided. When you are done deciding on all the options and values you want to use for EQAOPTS, follow the instructions in Appendix A, “Defining EQAOPTS options,” on page 75.

Enabling communication between Debug Tool and a remote debugger

If you want to use the IPv6 protocol to communicate with Debug Tool or if you are using CICS Transaction Server for z/OS, Version 2 Release 2 or earlier, you must activate the TCP/IP Socket Interface for CICS.

If you have CICS Transaction Server for z/OS Version 2 Release 3 or later and you do not want to use the IPv6 protocol to communicate with Debug Tool, you can use either the TCP/IP Socket Interface for CICS or the CICS Socket Domain.

To use 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 on 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.

To use the TCP/IP Socket Interface for CICS, see the z/OS Communications Server *IP CICS Sockets Guide*, SC31–8807.

Enabling the CADP transaction

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

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

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

Running multiple debuggers in a CICS region

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

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

Running the installation verification programs

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

IVP	Task
EQAWIVCI	Dynamic Debug facility and Enterprise PL/I TEST(ALL,SYM,NOHOOK,SEPARATE)
EQAWIVCP	Dynamic Debug facility and COBOL TEST(NONE,SYM,SEPARATE) or TEST(NOHOOK,SEPARATE)
EQAWIVC2	C TEST(ALL)
EQAWIVCG	C DEBUG(FORMAT(DWARF),HOOK(LINE,NOBLOCK,PATH),SYMBOL)
EQAWIVC8	Enterprise PL/I TEST(ALL)
EQAZIVCC[1]	Non-Language Environment Assembler

[1]This IVP is available only if you installed Debug Tool Utilities and Advanced Functions.

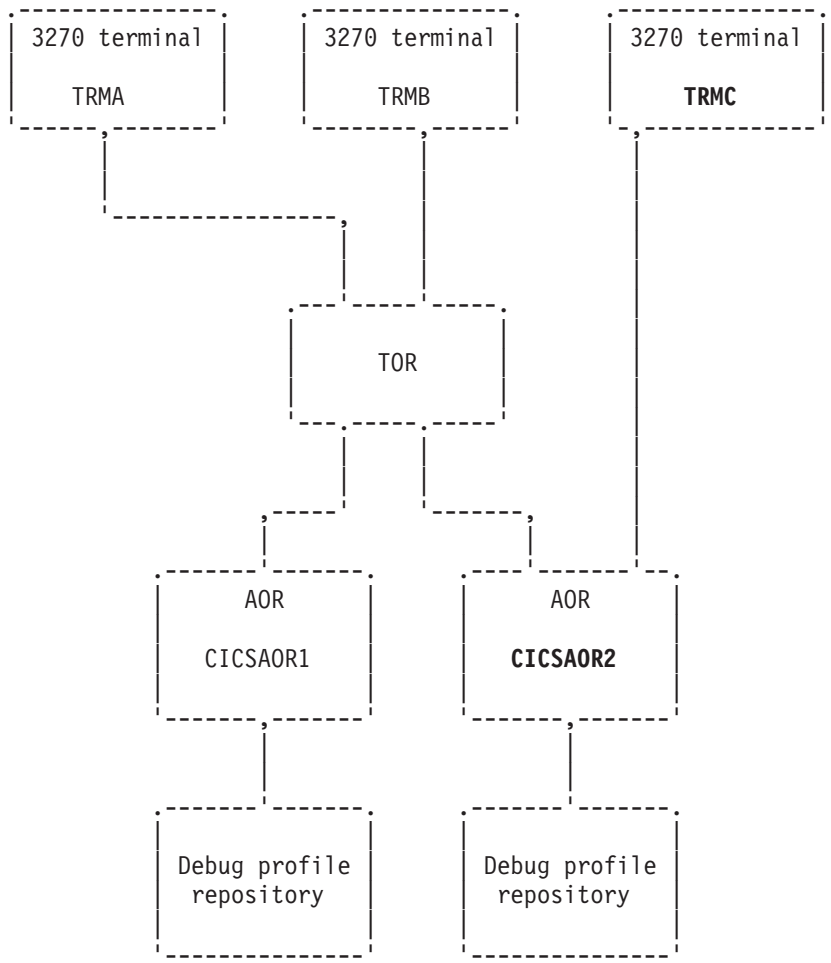
Configuring Debug Tool to run in a CICSplex environment

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 61
- "Terminal connects to a TOR which routes the application to an AOR; debugging profiles managed by DTCN" on page 62
- "Terminal connects to an AOR that runs an application that does not use a terminal" on page 64
- "Terminal connects to a TOR that runs an application that does not use a terminal" on page 64

Terminal connects to an AOR that runs the application

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



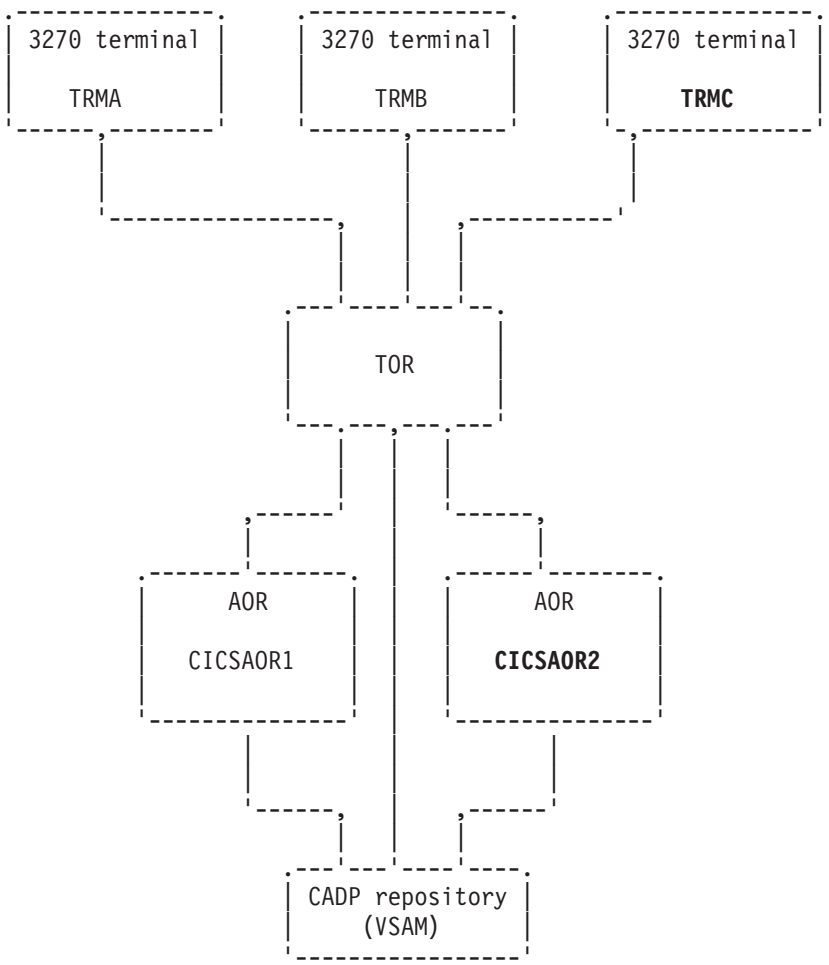
For this scenario to work, the CICS system administrator must complete the following tasks for the region CICSAOR2:

- Define Debug Tool resources in the CICS CSD and install them in the CICS region, as described in Chapter 8, "Adding support for debugging under CICS," on page 53, step 1.
- Provide access to these resources, as described in Chapter 8, "Adding support for debugging under CICS," on page 53, step 2a on page 53.

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

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

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



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

- Define Debug Tool resources in the CICS CSD and install them in the CICS region, as described in Chapter 8, "Adding support for debugging under CICS," on page 53, step 1.
- Provide access to these resources, as described in Chapter 8, "Adding support for debugging under CICS," on page 53, step 2a on page 53.
- Run the correct programs and use the correct CICS start up parameters for each type of profile, as described in the following steps:

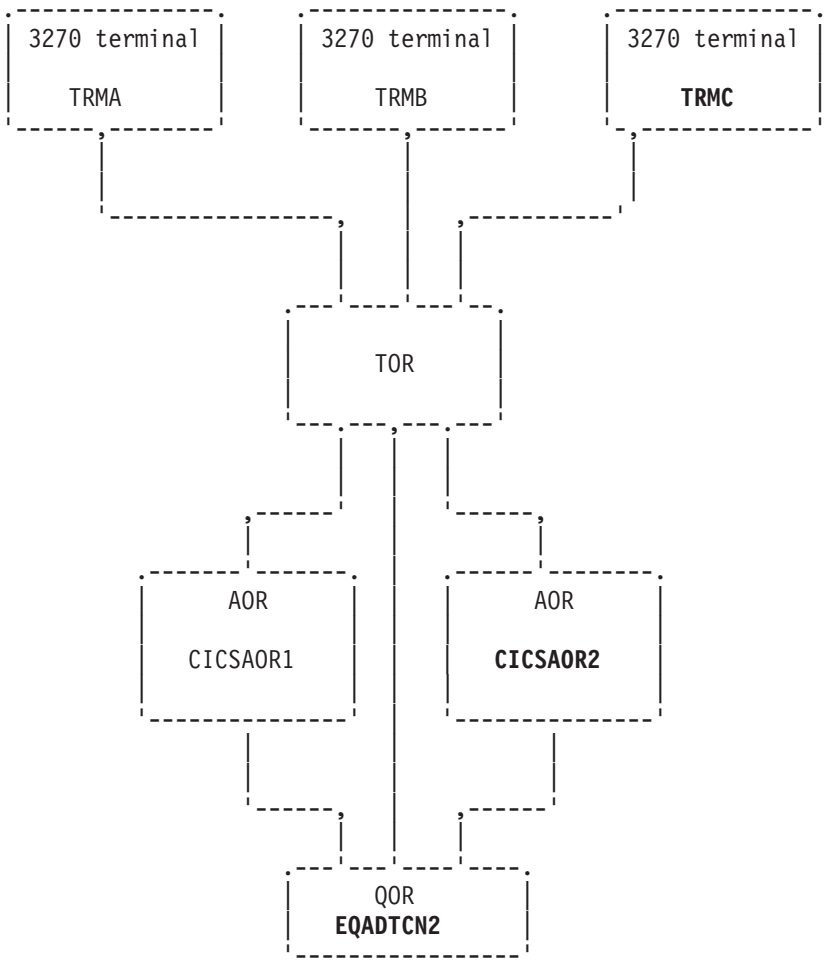
  CADP Chapter 8, "Adding support for debugging under CICS," on page 53, step 5d on page 54, 6b on page 54, 9b on page 55, and "Enabling the CADP transaction" on page 59.

  DTCN
  Chapter 8, "Adding support for debugging under CICS," on page 53, step 5a on page 54 and 9b on page 55.

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

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

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



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

- Define Debug Tool resources in the CICS CSD and install them in the CICS region, as described in Chapter 8, "Adding support for debugging under CICS," on page 53, step 1.
- Provide access to these resources, as described in Chapter 8, "Adding support for debugging under CICS," on page 53, step 2a on page 53.
- Designate a single CICS region as the QOR and define the queue accessible remotely, as described in "Sharing DTCN repository profile items among CICS systems" on page 57.

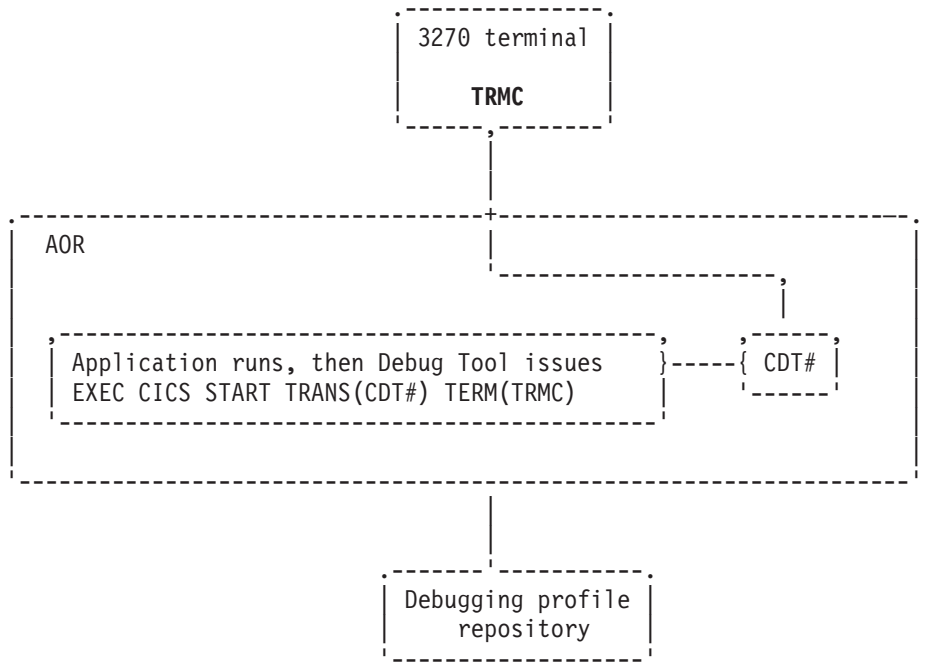
Variation on this scenario: The temporary storage queue (EQADTCN2) does not need to be located in a QOR. It can be located in the TOR, any of the AORs, or in the coupling facility. Wherever you put the temporary storage queue, keep the following considerations in mind:

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

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

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

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



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

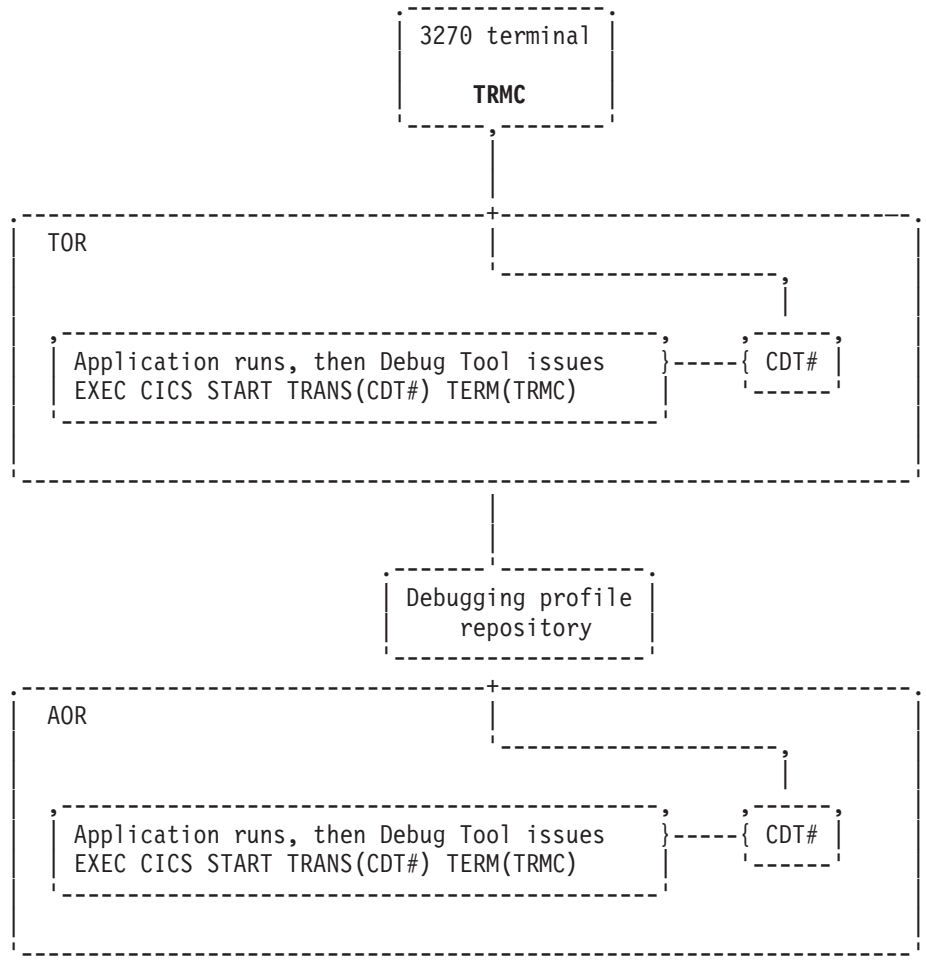
- Define Debug Tool resources in the CICS CSD and install them in the CICS region, as described in Chapter 8, "Adding support for debugging under CICS," on page 53, step 1 on page 53.
- Provide access to these resources, as described in Chapter 8, "Adding support for debugging under CICS," on page 53, step 2a on page 53.
- 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 8, "Adding support for debugging under CICS," on page 53, Chapter 8, "Adding support for debugging under CICS," on page 53, step 5d on page 54, 6b on page 54, 9b on page 55, and "Enabling the CADP transaction" on page 59.

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

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

1. You store a debugging profile into a repository using either DTCN or CADP.

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



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

- If you are using DTCN to manage debugging profiles, do the following tasks:
  –
  – Define Debug Tool resources in the CICS CSD and install them in the CICS region, as described in Chapter 8, “Adding support for debugging under CICS,” on page 53, step 1 on page 53.
  – Provide access to these resources, as described in Chapter 8, “Adding support for debugging under CICS,” on page 53, step 2a on page 53.
- 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 8, “Adding support for debugging under CICS,” on page 53, Chapter 8,

"Adding support for debugging under CICS," on page 53, step 5d on page 54, 6b on page 54, 9b on page 55, and "Enabling the CADP transaction" on page 59.
- 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 8, "Adding support for debugging under CICS," on page 53, step 1 on page 53.
  - Provide access to these resources, as described in Chapter 8, "Adding support for debugging under CICS," on page 53, step 2a on page 53.
- 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 8, "Adding support for debugging under CICS," on page 53, Chapter 8, "Adding support for debugging under CICS," on page 53, step 5d on page 54, 6b on page 54, 9b on page 55, and "Enabling the CADP transaction" on page 59.
- 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 Customization Guide* describes these exits.
  - Run a PLT program that enables the CICS exits XICTENF and XALTENF. The *CICS 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.

To authorize DTST users so they can modify CICS-key and USER-key storage, do the following steps:

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
   ```

Chapter 9. Adding support for debugging under IMS

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

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

   Scenario A
   You run programs in IMS Transaction Manager, BTS, or DB and are managing TEST run time options with a user exit. Do the steps described in “Scenario A: Running IMS and managing TEST run time options with a user exit” on page 70 to enable this scenario.

   Scenario B
   You run programs in IMS Transaction Manager, BTS, or DB and are managing TEST run time options with CEEUOPT or CEEROPT. Do the steps described in“Scenario B: Running IMS and managing TEST run time options with CEEUOPT or CEEROPT” on page 70 to enable this scenario.

   Scenario C
   You run assembler programs without Language Environment in IMS Transaction Manager and you specify some TEST run time options with the EQASET transaction. Do the steps described in “Scenario C: Running assembler program without Language Environment in IMS TM and managing TEST run time options with EQASET” on page 70 to enable this scenario.

   Scenario D
   You run programs in an IMSplex environment and are managing TEST run time options with either a user exit, CEEUOPT, or CEEROPT. Do the steps described in “Scenario D: Running IMSplex environment” on page 70 to enable this scenario.

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

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

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

Do the following steps to enable this scenario:

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

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

Do the following steps to enable this scenario:

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

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

Do the following steps to enable this scenario:

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

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

3. Add the application front end parameter APPLFE=EQANIAFE to the MPR start up job.
4. Assign the EQASET transaction to a class served by the MPR that is started with the APPLFE=EQANIAFE parameter.
5. Include the Debug Tool *hlq*.SEQAMOD data set and the Language Environment run-time library in the STEPLIB concatenation of your IMS region.
6. To give IMS users enough time to run and debug their applications, increase the time-out limit in the message-processing region (MPR) region to 1440.

Scenario D: Running IMSplex environment

Do the following steps to enable this scenario:

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

Chapter 10. Enabling additional languages for some Debug Tool components via EQACUIDF

The EQACUIDF member of *hlq*.SEQABMOD contains the default and allowable values for the parameters NATLANG, LOCALE and LINECOUNT. These parameters are used by the following Debug Tool and Debug Tool Utilities and Advanced Functions components:

- Debug Tool Utilities ISPF dialog: NATLANG
- EQANMDBG (non-CICS non-LE support: NATLANG
- Debug Tool Coverage Utility: NATLANG, LOCALE, LINECOUNT

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, EQACUIDM 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 7.

Chapter 11. Enabling support for display of NLS characters and modification of COBOL NATIONAL variables

If your users do either of the following tasks, you must create a conversion image for Debug Tool:

- Debug programs in remote debug mode that contain NLS characters.
- Use the STORAGE command to update COBOL NATIONAL variables.
- Properly display C/C++ variables that contain NLS characters

Creating a conversion image for Debug Tool

You need to create conversion image so that Debug Tool can properly communicate NLS characters between the remote debugger and the host. A conversion image contains the following information:

- The conversion table that specifies the source CCSID (Coded Character Set Identifiers) and target CCSID. For Debug Tool, specify a pair of conversion images between the host code page and Unicode code page (UTF-8). The host code page is specified in the VADSCP*nnnnn* suboption of TEST run-time option or in the CODEPAGE option in the EQAOPTS file. If both the VADSCP*nnnnn* suboption and the CODEPAGE option are specified, only the CODEPAGE option is used. The following table shows the images required for CCSIDs 930, 939 (Japanese EBCDIC), 933 (Korean EBCDIC), and 1141 (Germany EBCDIC). See *Debug Tool Reference and Messages* for a detailed description of the suboption VADSCP*nnnnn*.

  Table 7. Source and target CCSID to specify, depending on the code page option used

VADSCP*nnnn* suboption or CODEPAGE option	**Source CCSID**	**Target CCSID**
VADSCP930 or CODEPAGE,930	1390[1]	1208 (UTF-8)
	1208	1390[1]
VADSCP939 or CODEPAGE,939	1399[1]	1208 (UTF-8)
	1208	1399[1]
VADSCP933 or CODEPAGE,933	933	1208 (UTF-8)
	1208	933
VADSCP1141 or CODEPAGE,1141	1141	1208 (UTF-8)
	1208	1141
Notes: 1. For backward compatibility, 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 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 NLS characters. 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 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.

Appendix A. Defining EQAOPTS options

This topic describes how to define the EQAOPTS options so that they become effective at your site. Use these instructions in conjunction with the instructions in the corresponding topic.

1. Print out this topic.
2. Read the following topics:
   - "Overriding the default number of program elements held in cache" on page 58
   - "Specifying a code page" on page 16
   - "Setting the initial value for SET DEFAULT VIEW" on page 14
   - "Requiring users to specify resource types" on page 58
   - "Configuring for debugging Q++ programs" on page 31
   - "Specifying global preferences" on page 8
   - "Supplying NAMES commands for the initial load module" on page 13
   - "Modifying Debug Tool behavior when requested user interface is not available" on page 14
   - "Modifying the name of the default data sets that store settings, breakpoints, and monitor values" on page 9
   - "Specifying SUBSYS to access source code in a library system" on page 15
   - "SVC screening option" on page 10
   - "Suppressing the prompt Debug Tool displays for FINISH, CEE066, or CEE067 conditions" on page 16
   - "Running the Terminal Interface Manager on more than one LPAR on the same VTAM network" on page 47

   As you encounter a topic that describes an EQAOPTS option you might want to use, record the options and values you want to define for that option in the following checklist:

   __ EQAXOPT CACHENUM,*number:*_____

   __ EQAXOPT CODEPAGE,*code_page_number:*_____

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

      __ STANDARD

      __ NOMACGEN

   __ EQAXOPT DTCNFORCEIP, then select one of the following options:

      __ YES

      __ NO

   __ EQAXOPT DTCNFORCENETNAME, then select one of the following options:

      __ YES

      __ NO

   __ EQAXOPT DTCNFORCEPROGID, then select one of the following options:

      __ YES

      __ NO

   __ EQAXOPT DTCNFORCETERMID, then select one of the following options:

      __ YES

__ NO

__ EQAXOPT DTCNFORCEUSERID, then select one of the following options:

__ YES

__ NO

__ EQAXOPT EQAQPP, then select one of the following options:

__ ON

__ OFF

__ EQAXOPT GPFDSN,'*file_name:*_____'

__ EQAXOPT 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 SAVESETDSN,'*file_name:*_____'

__ EQAXOPT SAVEBPDSN,'*file_name:*_____'

__ EQAXOPT SUBSYS,*subsystem library name:*_____

__ EQAXOPT SVCSCREEN, then select one of the following options:

__ ON

__ OFF

Select one of the following options:

__ CONFLICT=OVERRIDE

__ CONFLICT=NOOVERIDE

If you want to do something about COPE, select one of the following options:

__ NOMERGE

__ MERGE=(COPE)

__ EQAXOPT THREADTERMCOND, then select one of the following options:

__ PROMPT

__ NOPROMPT

__ EQAXOPT TIMACB,*ACB-name:*_____

3. Copy the EQAOPTS member from the *hlq*.SEQASAMP library to a private library.
4. Edit this copy of EQAOPTS and code the EQAOPTS option or options you want. The following diagram describes the minimum assembler source required to generate the EQAOPTS load module:

```
EQAOPTS  CSECT ,
EQAOPTS  AMODE 31
EQAOPTS  RMODE ANY
         EQAXOPT  END
         END ,
```

To this minimum source you add each EQAXOPT option you selected in step 2 on page 75. See the topics listed in step 2 on page 75 or the EQAXOPT member of the *hlq*.SEQASAMP data set for the complete syntax of the macro invocation.

5. Follow the directions in EQAOPTS 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.
6. Place the EQAOPTS load module in a private data set that is in the load module search path and appears before *hlq*.SEQAMOD.

Appendix B. Applying maintenance

Appendix C, "Support information," on page 81 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 and (if purchased) Debug Tool Utilities and Advanced Functions. 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 or OS/390.

Checklist for applying an APAR or PTF

The following checklist describes the steps and associated SMP/E commands to install the APAR or PTF:

__ Step 1. Prepare to install the APAR or PTF.
__ Step 2. Receive the APAR or PTF. (SMP/E RECEIVE)
__ Step 3. Review the HOLDDATA.
__ Step 4. Accept previously applied APARs or PTFs (optional). (SMP/E ACCEPT)
__ Step 5. Apply APAR or PTF. (SMP/E APPLY)
__ Step 6. Run REPORT CROSSZONE and apply any missing requisites.
__ Step 7. Test APAR or PTF.
__ Step 8. Accept APAR or PTF. (SMP/E ACCEPT)

Step 1. Prepare to install APAR or PTF

Before you start to install an APAR or PTF, do the following:

1. Create a backup copy of the current Debug Tool libraries. Save this copy of Debug Tool until you have completed installing the APAR or PTF, and you are confident that the service runs correctly.
2. Research each service tape through the IBM Support Center for any errors or additional information. Note all errors on the tape that were reported by APARs or PTFs and apply the relevant fixes. You should also review the current Preventive Service Planning (PSP) information.

Step 2. Receive the APAR or PTF

Receive the service using the SMP/E RECEIVE command from either the SMP/E dialogs in ISPF, or using a batch job similar to EQAWRECV or EQAZRECV 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 or EQAZACPT 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 or EQAZAPLY 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 or EQAZRPXZ 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 or EQAZACPT in *hlq*.SEQASAMP.

Appendix C. Support information

If you have a problem with your IBM software, you want to resolve it quickly. This section describes the following options for obtaining support for IBM software products:

- "Searching knowledge bases"
- "Obtaining fixes"
- "Receiving weekly support updates" on page 82
- "Contacting IBM Software Support" on page 82

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

IBM provides this documentation in an information center. You can use the search function of the information center to query conceptual information, instructions for completing tasks, and reference information.

Searching the Internet

If you cannot find an answer to your question in the information center, search the Internet for the latest, most complete information that might help you resolve your problem.

To search multiple Internet resources for your product, use the **Web search** topic in your information center. In the navigation frame, click **Troubleshooting and support** ▸ **Searching knowledge bases** and select **Web search**. From this topic, you can search a variety of resources, including the following:

- IBM technotes
- IBM downloads
- IBM Redbooks®
- IBM developerWorks®
- Forums and newsgroups
- Google

Obtaining fixes

A product fix might be available to resolve your problem. To determine what fixes are available for your IBM software product, follow these steps:

1. Go to the IBM Software Support Web site at http://www.ibm.com/software/support.
2. Click **Downloads and drivers** in the **Support topics** section.
3. Select the **Software** category.
4. Select a product in the **Sub-category** list.
5. In the **Find downloads and drivers by product** section, select one software category from the **Category** list.
6. Select one product from the **Sub-category** list.

7. Type more search terms in the **Search within results** if you want to refine your search.
8. Click **Search**.
9. From the list of downloads returned by your search, click the name of a fix to read the description of the fix and to optionally download the fix.

For more information about the types of fixes that are available, see the *IBM Software Support Handbook* at http://techsupport.services.ibm.com/guides/handbook.html.

Receiving weekly support updates

To receive weekly e-mail notifications about fixes and other software support news, follow these steps:

1. Go to the IBM Software Support Web site at http://www.ibm.com/software/support.
2. Click **My support** in the upper right corner of the page.
3. If you have already registered for **My support**, sign in and skip to the next step. If you have not registered, click **register now**. Complete the registration form using your e-mail address as your IBM ID and click **Submit**.
4. Click **Edit profile**.
5. In the **Products** list, select **Software**. A second list is displayed.
6. In the second list, select a product segment, for example, **Application servers**. A third list is displayed.
7. In the third list, select a product sub-segment, for example, **Distributed Application & Web Servers**. A list of applicable products is displayed.
8. Select the products for which you want to receive updates, for example, **IBM HTTP Server** and **WebSphere Application Server**.
9. Click **Add products**.
10. After selecting all products that are of interest to you, click **Subscribe to email** on the **Edit profile** tab.
11. Select **Please send these documents by weekly email**.
12. Update your e-mail address as needed.
13. In the **Documents** list, select **Software**.
14. Select the types of documents that you want to receive information about.
15. Click **Update**.

If you experience problems with the **My support** feature, you can obtain help in one of the following ways:

Online
Send an e-mail message to erchelp@ca.ibm.com, describing your problem.

By phone
Call 1-800-IBM-4You (1-800-426-4968).

Contacting IBM Software Support

IBM Software Support provides assistance with product defects.

Before contacting IBM Software Support, your company must have an active IBM software maintenance contract, and you must be authorized to submit problems to IBM. The type of software maintenance contract that you need depends on the type of product you have:

- For IBM distributed software products (including, but not limited to, Tivoli®, Lotus®, and Rational® products, as well as DB2 and WebSphere products that run on Windows, or UNIX operating systems), enroll in Passport Advantage® in one of the following ways:

  Online

  Go to the Passport Advantage Web site at http://www.lotus.com/services/passport.nsf/ WebDocs/Passport_Advantage_Home and click **How to Enroll**.

  By phone

  For the phone number to call in your country, go to the IBM Software Support Web site at http://techsupport.services.ibm.com/guides/contacts.html and click the name of your geographic region.

- For customers with Subscription and Support (S & S) contracts, go to the Software Service Request Web site at https://techsupport.services.ibm.com/ssr/login.
- For customers with IBMLink™, CATIA, Linux, S/390®, iSeries®, pSeries®, zSeries®, and other support agreements, go to the IBM Support Line Web site at http://www.ibm.com/services/us/index.wss/so/its/a1000030/dt006.
- For IBM eServer™ software products (including, but not limited to, DB2 and WebSphere products that run in zSeries, pSeries, and iSeries environments), you can purchase a software maintenance agreement by working directly with an IBM sales representative or an IBM Business Partner. For more information about support for eServer software products, go to the IBM Technical Support Advantage Web site at http://www.ibm.com/servers/eserver/techsupport.html.

If you are not sure what type of software maintenance contract you need, call 1-800-IBMSERV (1-800-426-7378) in the United States. From other countries, go to the contacts page of the *IBM Software Support Handbook* on the Web at http://techsupport.services.ibm.com/guides/contacts.html and click the name of your geographic region for phone numbers of people who provide support for your location.

To contact IBM Software support, follow these steps:

1. "Determining the business impact"
2. "Describing problems and gathering information" on page 84
3. "Submitting problems" on page 84

Determining the business impact

When you report a problem to IBM, you are asked to supply a severity level. Therefore, you need to understand and assess the business impact of the problem that you are reporting. Use the following criteria:

Severity 1

The problem has a *critical* business impact. You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.

Severity 2

The problem has a *significant* business impact. The program is usable, but it is severely limited.

Severity 3

The problem has *some* business impact. The program is usable, but less significant features (not critical to operations) are unavailable.

Severity 4

The problem has *minimal* business impact. The problem causes little impact on operations, or a reasonable circumvention to the problem was implemented.

Describing problems and gathering information

When describing a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Software Support specialists can help you solve the problem efficiently. To save time, know the answers to these questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, and messages that are related to the problem symptoms? IBM Software Support is likely to ask for this information.
- Can you re-create the problem? If so, what steps were performed to re-create the problem?
- Did you make any changes to the system? For example, did you make changes to the hardware, operating system, networking software, and so on.
- Are you currently using a workaround for the problem? If so, be prepared to explain the workaround when you report the problem.

Submitting problems

You can submit your problem to IBM Software Support in one of two ways:

Online

Click **Submit and track problems** on the IBM Software Support site athttp://www.ibm.com/software/support/probsub.html. Type your information into the appropriate problem submission form.

By phone

For the phone number to call in your country, go to the contacts page of the *IBM Software Support Handbook* at http://techsupport.services.ibm.com/guides/contacts.html and click the name of your geographic region.

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Software Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the Software Support Web site daily, so that other users who experience the same problem can benefit from the same resolution.

Appendix D. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The accessibility features in z/OS provide accessibility for Debug Tool.

The major accessibility features in z/OS enable users to:

- Use assistive technology products such as screen readers and screen magnifier software
- Operate specific or equivalent features by using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products work with the user interfaces that are found in z/OS. For specific guidance information, consult the documentation for the assistive technology product that you use to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces by using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Volume 1* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Accessibility of this document

Information in the following formats of this document is accessible to visually impaired individuals who use a screen reader:

- PDF format when viewed with Adobe® Acrobat® Reader 5.0 or later
- BookManager® format when viewed with IBM BookManager BookServer (except for syntax diagrams)

Syntax diagrams start with the word Format or the word Fragments. Each diagram is preceded by two images. For the first image, the screen reader will say "Read syntax diagram". The associated link leads to an accessible text diagram. When you return to the document at the second image, the screen reader will say "Skip visual syntax diagram" and has a link to skip around the visible diagram.

For BookManager users only: A screen reader might say the lines, symbols, and words in a diagram, but not in a meaningful way. For example, you might hear "question question dash dash MOVE dash dash plus dash dash literal-1 dash dash plus" for part of the MOVE statement. You can enter **Say Next Paragraph** to move quickly through syntax diagrams if your screen reader has that capability.

Notices

This information was developed for products and services offered in the U.S.A. IBM might not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with the local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Trademarks and service marks

The following terms, denoted by an asterisk (*) on the first occurrence in this publication, are trademarks or service marks of International Business Machines Corporation in the United States or other countries:

IBM
The IBM logo
ibm.com
AD/Cycle
Advanced Peer-to-Peer Networking
AIX
BookManager
C/370
CICS
COBOL/370
DB2
DB2 Universal Database
developerWorks
eServer
IBM
IBMLink
IMS
iSeries
Language Environment
Lotus
MVS
MVS/ESA
OS/390
Passport Advantage
pSeries
RACF
Rational
Redbooks
Resource Link
S/390
Tivoli
VisualAge
VSE/ESA
VTAM
WebSphere
z/OS
z/VM
zSeries

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

Acrobat, Adobe, the Adobe logo, PostScript®, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other contries.

Java™ and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

LINUX is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT®, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

MasterCraft is a trademark of Tata Consultancy Services Ltd.

Glossary

This glossary defines technical terms and abbreviations used in *Debug Tool Customization Guide* documentation. If you do not find the term you are looking for, refer to the *IBM Glossary of Computing Terms*, located at the IBM Terminology web site:

`http://www.ibm.com/ibm/terminology`

B

batch. Pertaining to a predefined series of actions performed with little or no interaction between the user and the system. Contrast with *interactive*.

batch job. A job submitted for batch processing. See *batch*. Contrast with *interactive*.

C

CADP. A CICS-supplied transaction used for managing debugging profiles from a 3270 terminal.

compile. To translate a program written in a high level language into a machine-language program.

compile unit. A sequence of HLL statements that make a portion of a program complete enough to compile correctly. Each HLL product has different rules for what comprises a compile unit.

compiler. A program that translates instructions written in a high level programming language into machine language.

D

data set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

debug. To detect, diagnose, and eliminate errors in programs.

DTCN. Debug Tool Control utility, a CICS transaction that enables the user to identify which CICS programs to debug.

debugging profile. Data that specifies a set of application programs which are to be debugged together.

F

full-screen mode. An interface mode for use with a nonprogrammable terminal that displays a variety of information about the program you are debugging.

H

hook. An instruction inserted into a program by a compiler when you specify the `TEST` compile option. Using a hook, you can set breakpoints to instruct Debug Tool to gain control of the program at selected points during its execution.

L

link-edit. To create a loadable computer program using a linkage editor.

load module. A program in a form suitable for loading into main storage for execution. In this document this term is also used to refer to a Dynamic Load Library (DLL).

logical window. A group of related debugging information (for example, variables) that is formatted so that it can be displayed in a physical window.

LU. See "logical unit."

logical unit. (1) A type of network accessible unit that enables users to gain access to network resources and communicate with each other. (2) A name used by VTAM to identify a terminal or other resource.

N

network identifier. In TCP/IP, that part of the IP address that defines a network. The length of the network ID depends on the type of network class (A, B, or C).

node name. The name assigned to a node during network definition. The format for the node name is *netid.cpname*.

P

parameter. Data passed between programs or procedures.

partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

PDS. See *partitioned data set*.

physical window. A section of the screen dedicated to the display of one of the four logical windows: Monitor window, Source window, Log window, or Memory window.

PLU. See *primary logical unit*.

primary logical unit. (1) In SNA, the logical unit that contains the primary half-session for a particular logical unit-to-logical unit (LU-to-LU) session. (2) In SNA, the logical unit (LU) that sends the BIND to activate a session with its partner LU.

profile. A group of customizable settings that govern how the user's session appears and operates.

program. A sequence of instructions suitable for processing by a computer. Processing can include the use of an assembler, a compiler, an interpreter, or a translator to prepare the program for execution, as well as to execute it.

S

secondary logical unit. (1) In SNA, the logical unit (LU) that contains the secondary half-session for a particular LU-LU session. An LU may contain secondary and primary half-sessions for different active LU-LU sessions. (2) A VTAM Secondary Logical Unit (i.e., terminal).

session. The events that take place between the time the user starts an application and the time the user exits the application.

SIMLOGON. A VTAM macro instruction that initiates a session in which the application program acts as the PLU.

Single Point of Control. The control interface that sends commands to one or more members of an IMSplex and receives command responses.

SLU. See "secondary logical unit."

SPOC. See "Single Point of Control."

U

utility. A computer program in general support of computer processes; for example, a diagnostic program, a trace program, or a sort program.

V

VTAM. See "Virtual Telecommunications Access Method."

Virtual Telecommunications Access Method (VTAM). (1) IBM software that controls communication and the flow of data in an SNA network by providing the SNA application programming interfaces and SNA networking functions. An SNA network includes subarea networking, Advanced Peer-to-Peer Networking® (APPN), and High-Performance Routing (HPR). Beginning with Release 5 of the OS/390 operating system, the VTAM for MVS/ESA™ function was included in Communications Server for OS/390; this function is called Communications Server for OS/390 - SNA Services. (2) An access method commonly used by MVS to communicate with terminals and other communications devices.

Bibliography

Debug Tool publications

Using CODE/370 with VS COBOL II and OS PL/I, SC09-1862

Debug Tool for z/OS

Debug Tool Reference Summary, SC19-1199

Debug Tool Customization Guide, SC19-1200

Program Directory for Debug Tool for z/OS, GI10-8761

Debug Tool Reference and Messages, GC19-1198

Debug Tool User's Guide, SC19-1196

Debug Tool Utilities and Advanced Functions for z/OS

Debug Tool Coverage Utility Users Guide, SC19-1197

Program Directory for Debug Tool Utilities and Advanced Functions for z/OS, GI10-8762

High level language publications

z/OS C and C++

Compiler and Run-Time Migration Guide, GC09-4913

Curses, SA22-7820,

Language Reference, SC09-4815

Programming Guide, SC09-4765

Run-Time Library Reference, SA22-7821

User's Guide, SC09-4767

Enterprise COBOL for z/OS, Version 4

Enterprise COBOL for z/OS Compiler and Runtime Migration Guide, GC27-1409

Enterprise COBOL for z/OS Customization Guide, SC23-8526

Enterprise COBOL for z/OS Licensed Program Specifications, GI11-7871

Enterprise COBOL for z/OS Language Reference, SC23-8528

Enterprise COBOL for z/OS Programming Guide, SC23-8529

Enterprise COBOL for z/OS and OS/390, Version 3

Migration Guide, GC27-1409

Customization, GC27-1410

Licensed Program Specifications, GC27-1411

Language Reference, SC27-1408

Programming Guide, SC27-1412

COBOL for OS/390 & VM

Compiler and Run-Time Migration Guide, GC26-4764

Customization under OS/390, GC26-9045

Language Reference, SC26-9046

Programming Guide, SC26-9049

Enterprise PL/I for z/OS and OS/390

Diagnosis Guide, SC27-1459

Language Reference, SC27-1460

Licensed Program Specifications, GC27-1456

Messages and Codes, SC27-1461

Migration Guide, GC27-1458

Programming Guide, SC27-1457

VisualAge PL/I for OS/390

Compiler and Run-Time Migration Guide, SC26-9474

Diagnosis Guide, SC26-9475

Language Reference, SC26-9476

Licensed Program Specifications, GC26-9471

Messages and Codes, SC26-9478

Programming Guide, SC26-9473

PL/I for MVS & VM

Compile-Time Messages and Codes, SC26-3229

Compiler and Run-Time Migration Guide, SC26-3118

Diagnosis Guide, SC26-3149

Installation and Customization under MVS, SC26-3119

Language Reference, SC26-3114

Licensed Program Specifications, GC26-3116

Programming Guide, SC26-3113

Reference Summary, SX26-3821

Related publications

CICS

Application Programming Guide, SC34-6231

Application Programming Primer, SC34-0674

Application Programming Reference, SC34-6232

DB2 Universal Database™ for z/OS

Administration Guide, SC18-7413

Application Programming and SQL Guide, SC18-7415

Command Reference, SC18-7416

Data Sharing: Planning and Administration, SC18-7417

Installation Guide, GC18-7418

Messages and Codes, GC18-7422

Reference for Remote DRDA Requesters and Servers*, SC18-7424

Release Planning Guide, SC18-7425

SQL Reference, SC18-7426

Utility Guide and Reference, SC18-7427

IMS

IMS Application Programming: Database Manager, SC27-1286

IMS Application Programming: EXEC DLI Commands for CICS & IMS, SC27-1288

IMS Application Programming: Transaction Manager, SC27-1289

TSO/E

Command Reference, SA22-7782

Programming Guide, SA22-7788

System Programming Command Reference, SA22-7793

User's Guide, SA22-7794

z/OS

MVS JCL Reference, SA22-7597

MVS JCL User's Guide, SA22-7598

MVS System Commands, SA22-7627

z/OS Language Environment

Concepts Guide, SA22-7567

Customization, SA22-7564

Debugging Guide, GA22-7560

Programming Guide, SA22-7561

Programming Reference, SA22-7562

Run-Time Migration Guide, GA22-7565

Vendor Interfaces, SA22-7568

Writing Interlanguage Communication Applications, SA22-7563

Softcopy publications

Online publications are distributed on CD-ROMs and can be ordered through your IBM representative. *Debug Tool User's Guide*, *Debug Tool Customization Guide*, and *Debug Tool Reference and Messages* are distributed on the following collection kit:

SK3T-4269

Online publications can also be downloaded from the IBM Web site. Visit the IBM Web site for each product to find online publications for that product.

Index

J

K

L

M

N

O

P

R

S

T

U

V

W

Readers' Comments — We'd Like to Hear from You

Debug Tool for z/OS
Debug Tool Utilities and Advanced Functions for z/OS
Customization Guide
Version 8.1

Publication No. SC19-1200-02

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: COMMENTS@US.IBM.COM

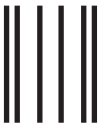
If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
SC19-1200-02

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Silicon Valley Laboratory
Department J87/D325
555 Bailey Ave.
San Jose, CA
95141-9989

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Program Number: 5655-S17

Printed in USA

SC19-1200-02