IBM Library Server Print Preview

DOCNUM = SC26-8798-00
DATETIME = 10/29/96 16:07:22
BLDVERS = 1.2
TITLE = Debug Tool/VSE V1R1 Installation and Customization Guide
AUTHOR =

PATH = /home/webapps/epubs/htdocs/book

COVER Book Cover

Debug Tool for VSE/ESA

Installation and Customization Guide

Release 1

Document Number SC26-8798-00

NOTICES Notices

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" in topic FRONT_1.

EDITION Edition Notice

First Edition (December 1996)

This edition applies to the Debug Tool feature of the following compilers:

- IBM C for VSE/ESA Version 1, Release 1 (Program Number 5686-A01)
- IBM COBOL for VSE/ESA Version 1, Release 1 (Program Number 5686-068)
- IBM PL/I for VSE/ESA Version 1, Release 1 (Program Number 5686-069)

and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation, Department J58
P.O. Box 49023
San Jose, CA, 95161-9023
United States of America

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.

CONTENTS Table of Contents

Summarize

COVER **Book Cover**

NOTICES **Notices**

EDITION **Edition Notice**

CONTENTS **Table of Contents**

TABLES **Tables**

FIGURES **Figures**

FRONT_1 **Notices**
FRONT_1.1 **Programming Interface Information**
FRONT_1.2 **Trademarks**

PREFACE **About This Book**
PREFACE.1 **Debug Tool**
PREFACE.2 **IBM Language Environment for VSE/ESA**
PREFACE.3 **Using Your Documentation**
PREFACE.4 **How to Read the Syntax Diagrams**

1.0 **Chapter 1. Planning for Debug Tool**
1.1 **Planning to Install Debug Tool**
1.1.1 What You Receive with Debug Tool
1.1.2 What You Need to Install Debug Tool
1.1.3 Planning Where to Install Debug Tool
1.1.4 VTAM Considerations
1.1.5 CICS Considerations
1.1.6 Selecting National Language Support
1.1.7 Special Considerations
1.1.8 Checking Service Updates
1.1.9 Publications Useful During Installation
1.2 **Planning for Customizing Debug Tool**
1.2.1 Planning to Install in the Shared Virtual Area

1.2.2 Planning to Change the National Language for Debug Tool

2.0 **Chapter 2. Installing Debug Tool**
2.1 **Overview of Installation**
2.1.1 Checklist for Installing Debug Tool
2.2 **Step 1: Backup the Original System**
2.3 **Step 2: Allocate Space for the Library**
2.3.1 Defining a Library in VSAM Space
2.3.2 Defining a Library in Non-VSAM Space
2.4 **Step 3: Install Debug Tool**
2.4.1 Method 1: Install Debug Tool Using the Interactive Interface
2.4.2 Method 2: Install Debug Tool Using a Batch Job
2.5 **Step 4: Configure VTAM**
2.6 **Step 5: Configure CICS**

3.0 **Chapter 3. Verifying the Installation of Debug Tool**
3.1 **Installation Verification with C/VSE**
3.1.1 Batch Debug Using a Commands File
3.1.2 Batch Execution with Interactive Debug
3.1.3 Interactive Debug with CICS
3.2 **Installation Verification with COBOL/VSE**
3.2.1 Batch Debug Using a Commands File
3.2.2 Batch Execution with Interactive Debug
3.2.3 Interactive Debug with CICS
3.3 **Installation Verification with PL/I VSE**
3.3.1 Batch Debug Using a Commands File
3.3.2 Batch Execution with Interactive Debug
3.3.3 Interactive Debug with CICS

4.0 **Chapter 4. Customizing Debug Tool**
4.1 **Loading Debug Tool into the Shared Virtual Area**
4.2 **Changing the National Language for Debug Tool**

5.0 **Chapter 5. Maintaining Debug Tool**
5.1 **Re-installing Debug Tool**
5.2 **Applying Service Updates**
5.2.1 What You Receive
5.2.2 Checklist for Applying Service
5.2.3 Step 1. Check Prerequisite APARs or PTFs
5.2.4 Step 2. Backup Original System
5.2.5 Step 3. Apply Service
5.2.6 Step 4. Run the Installation Verification Programs (IVP)
5.3 **Removing Debug Tool**
5.4 **How to Report a Problem with Debug Tool**

A.0 **Appendix A. Determining the Default Userid**
A.1 Batch (non-CICS) Environment
A.2 CICS Environment

B.0 **Appendix B. SVA Space Requirements**

BIBLIOGRAPHY **Bibliography**
BIBLIOGRAPHY.1 **Debug Tool Publications**
BIBLIOGRAPHY.2 **Language Environment Publications**
BIBLIOGRAPHY.3 **LE/VSE-Conforming Language Product Publications**
BIBLIOGRAPHY.4 **Related Publications**
BIBLIOGRAPHY.5 **Softcopy Publications**

INDEX **Index**

BACK_1 **We'd Like to Hear from You**

COMMENTS **Readers' Comments**

TABLES Tables

1. How to Use Debug Tool, LE/VSE and Language Publications PREFACE.3
2. Component ID and CLC 1.1.1
3. Basic material: Debug Tool Optional Program Tape for C/VSE 1.1.1.1
4. Basic material: Debug Tool Optional Program Tape for COBOL/VSE 1.1.1.1
5. Basic material: Debug Tool Optional Program Tape for PL/I VSE 1.1.1.1

6. Distribution media: file contents 1.1.1.1
7. Basic Material: Unlicensed Publications 1.1.1.2
8. Required Licensed Programs for Debug Tool 1.1.2.1
9. Optional Licensed Programs for Debug Tool 1.1.2.1
10. Approximate Library Storage Requirements for Debug Tool 1.1.2.2
11. Summary of steps for installing Debug Tool 2.1.1
12. Installation Verification Samples 3.0
13. Summary of steps for installation verification of C/VSE with CICS. 3.1.3
14. Summary of steps for installation verification of COBOL/VSE with CICS. 3.2.3
15. Summary of steps for CICS IVP for PL/I VSE. 3.3.3
16. Summary of steps for installing service on Debug Tool 5.2.2
17. 31-bit SVA Space Requirements for Debug Tool Base. B.1
18. 24-bit SVA Space Requirements for Debug Tool Base. B.2
19. SVA Space Requirements for Debug Tool Japanese Language Component. B.3

FIGURES Figures

1. Define the Debug Tool Library Cluster in VSAM Space. 2.3.1
2. Define the LE/VSE Library. 2.3.1
3. List the Contents of Volume SYSWK1. 2.3.2
4. Define the LE/VSE Library. 2.3.2
5. Install Debug Tool 2.4.2
6. Job Control Extract for EQAWIVH1. 3.1.1
7. SYSLST Log Content after Execution of IVPCCC1. 3.1.1
8. Job Control Extract for EQAWIVH2. 3.1.2
9. Initial Debug Session Screen for IVPCCC2. 3.1.2
10. SYSLST Log Content after Execution of IVPCCC2. 3.1.2
11. Job Control Extract for EQAWIVH3. 3.1.3
12. Initial Debug Session Screen for IVPCCC3. 3.1.3
13. Log File Content after Execution of IVPCCC3. 3.1.3
14. Job Control Extract for EQAWIVC1. 3.2.1
15. SYSLST Log Content after Execution of IVPCOB1. 3.2.1
16. Job Control Extract for EQAWIVC2. 3.2.2
17. Debug Session Screen for IVPCOB2 after Commands File Execution. 3.2.2
18. SYSLST Log Content after Execution of IVPCOB2. 3.2.2
19. Job Control Extract for EQAWIVC3. 3.2.3
20. Job Control Extract for EQAWIVC4. 3.2.3
21. Initial Debug Session Screen for IVPCOB3. 3.2.3
22. Log File Content after Execution of IVPCOB3. 3.2.3
23. Job Control Extract for EQAWIVP1. 3.3.1
24. SYSLST Log Content after Execution of IVPPLI1. 3.3.1
25. Job Control Extract for EQAWIVP2. 3.3.2
26. Debug Session Screen for IVPPLI2 after Commands File Execution. 3.3.2
27. SYSLST Log Content after Execution of IVPPLI2. 3.3.2
28. Job Control Extract for EQAWIVP3. 3.3.3
29. Initial Debug Session Screen for IVPPLI3. 3.3.3
30. Log File Content after Execution of IVPPLI3. 3.3.3
31. Example of the PARM option with NATLANG (C and PL/I) 4.2
32. Example of the PARM option with NATLANG (COBOL) 4.2
33. Retrace APARs and PTFs 5.2.3
34. Apply Service 5.2.5.2
35. Delete Debug Tool from a Sublibrary 5.3
36. Remove Debug Tool from the System History File 5.3

FRONT_1 Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. 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. Subject to IBM's valid intellectual property or

other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

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 Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Subtopics:

- FRONT_1.1 Programming Interface Information
- FRONT_1.2 Trademarks

FRONT_1.1 Programming Interface Information

This book is intended to help you to plan for, install, customize, and maintain Debug Tool for VSE/ESA. This book documents the procedures required to install, customize, and maintain Debug Tool for VSE/ESA.

FRONT_1.2 Trademarks

The following terms are trademarks or service marks of the IBM Corporation in the United States or other countries or both:

BookManager	IBM	VSE/ESA
CICS	Language Environment	VTAM
CICS/VSE	QMF	
DFSORT	SQL/DS	

PREFACE About This Book

This book is intended for users of Debug Tool for VSE/ESA (Debug Tool) as implemented with IBM Language Environment for VSE/ESA (LE/VSE). It contains information on planning for, installing, and customizing Debug Tool.

This is the first release of Debug Tool for VSE/ESA. It brings to the VSE environment many of the functions of Debug Tool from the MVS and VM environments.

Subtopics:

- PREFACE.1 Debug Tool
- PREFACE.2 IBM Language Environment for VSE/ESA

- PREFACE.3 Using Your Documentation
- PREFACE.4 How to Read the Syntax Diagrams

PREFACE.1 Debug Tool

Debug Tool is distributed as part of the Full Function offering of the following IBM high-level language compilers:

- IBM C for VSE/ESA
- IBM COBOL for VSE/ESA
- IBM PL/I for VSE/ESA

Debug Tool is a powerful interactive source-level debug tool for application programs written in the above high-level languages. It allows programmers to address difficult problems at the source level where they are comfortable.

While a program is running, programmers can control and examine its execution with functions such as:

- Viewing the source listing and stepping through the source one statement at a time
- Setting dynamic break points, which can be simple or conditional based on other values in the program
- Monitoring the value of program variables
- Modifying program and variable storage
- Debugging mixed-language applications from a single debug session

The debug session can be recorded in a log file, so the programmer can replay the session. Programmers can use Debug Tool to help capture test cases for future program validation or to further isolate a problem within an application.

PREFACE.2 IBM Language Environment for VSE/ESA

Debug Tool can be used to debug application programs written in high-level languages that use the run-time environment and library of run-time callable services provided by LE/VSE.

LE/VSE establishes a common run-time environment and common run-time callable services for language products, user programs, and other products.

The common execution environment is made up of data items and services performed by library routines available to a particular application

running in the environment. The services that LE/VSE provides to your application include:

- Services that satisfy basic requirements common to most applications. These include support for the initialization and termination of applications, allocation of storage, support for *interlanguage communication (ILC)* and *condition handling*.
- Extended services often needed by applications. These functions are contained within a library of callable routines, and include interfaces to operating system functions and a variety of other commonly used functions.
- Run-time options that help the execution, performance tuning, performance, and diagnosis of your application.
- Access to language-specific library routines.

PREFACE.3 Using Your Documentation

The publications provided with Debug Tool and LE/VSE are designed to help you:

- Plan for, install, customize, and maintain Debug Tool.
- Debug problems in your LE/VSE-conforming applications.

Language programming information is provided in the high-level language programming manuals that provide language definition, library function syntax and semantics, and programming guidance information.

Each publication helps you perform a different task. For a complete list of publications you might need, see "Bibliography" in topic BIBLIOGRAPHY.

Table 1. How to Use Debug Tool, LE/VSE and Language Publications

To...	Use...	
Evaluate Debug Tool	Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Plan for, install, customize, and maintain Debug Tool	*Debug Tool for VSE/ESA Installation and Customization Guide*	SC26-8798
Use Debug Tool to debug LE/VSE-conforming applications	Debug Tool for VSE/ESA User's Guide and Reference	SC26-8797
Debug your LE/VSE-conforming application and get details on run-time messages	*LE/VSE Debugging Guide and Run-Time Messages*	SC33-6681
Diagnose problems that occur in your LE/VSE-conforming application	*LE/VSE Debugging Guide and Run-Time Messages*	SC33-6681
Understand the LE/VSE program models and concepts	*LE/VSE Concepts Guide* *LE/VSE Programming Guide*	GC33-6680 SC33-6684
Diagnose compiler problems that occur in your LE/VSE-conforming application	Your compiler Diagnosis Guide	

PREFACE.4 How to Read the Syntax Diagrams

The following rules apply to the syntax diagrams used in this book:

Arrow symbols

Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

The >>--- symbol indicates the beginning of a statement.

The ---> symbol indicates that the statement syntax is continued on the next line.

The >--- symbol indicates that a statement is continued from the previous line.

The --->< symbol indicates the end of a statement.

Diagrams of syntactical units other than complete statements start with the >--- symbol and end with the ---> symbol.

Conventions

- Keywords, their allowable synonyms, and reserved parameters, appear in uppercase. These items must be entered exactly as shown.
- Variables appear in highlighted lowercase (for example, *column-name*). They represent user-defined parameters or suboptions.
- When entering commands, separate parameters and keywords by at least one blank if there is no intervening punctuation.
- Enter punctuation marks (slashes, commas, periods, parentheses, quotation marks, equal signs) and numbers exactly as given.
- Footnotes are shown by a number in parentheses, for example, (1).

Syntax Fragments

If a syntax diagram contains too many items to fit in the diagram, the syntax is shown by a main syntax diagram and one or more syntax fragments.

A syntax fragment is referred to in the main diagram by its fragment name between two vertical bars.

Each syntax fragment appears below the main syntax diagram, and is delimited by vertical bars. A heading above the fragment indicates

the name of the fragment.

syntax_fragment:

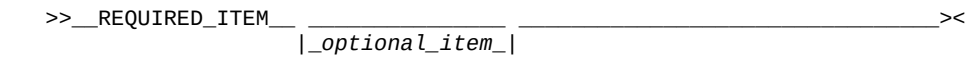
Read each syntax fragment as though it were imbedded in the main syntax diagram.

Required items

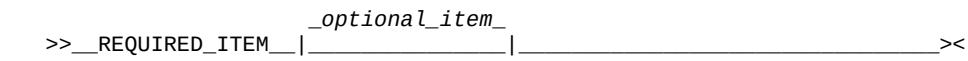
Required items appear on the horizontal line (the main path).

Optional Items

Optional items appear below the main path.

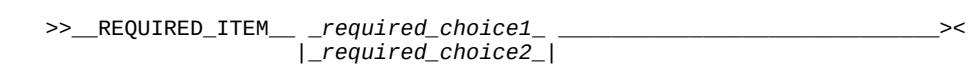


If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.

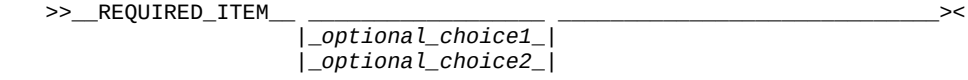


Multiple required or optional items

If you can choose from two or more items, they appear vertically in a stack. If you *must* choose one of the items, one item of the stack appears on the main path.

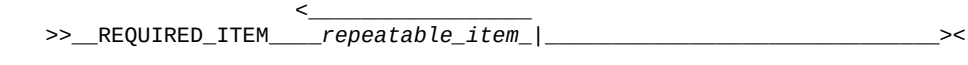


If choosing one of the items is optional, the entire stack appears below the main path.

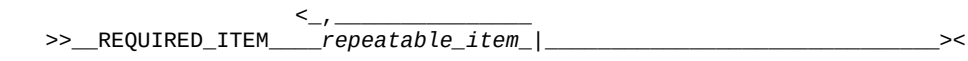


Repeatable items

An arrow returning to the left above the main line indicates that an item can be repeated.



If the repeat arrow contains a comma, you must separate repeated items with a comma.



A repeat arrow above a stack indicates that you can specify more than

one of the choices in the stack.

Default keywords

IBM-supplied default keywords appear above the main path, and the remaining choices are shown below the main path. In the parameter list following the syntax diagram, the default choices are underlined.

```
                      _default_choice__
>>__REQUIRED_ITEM__|__________________|_____________________________><
                   |_optional_choice_|
                   |_optional_choice_|
```

1.0 Chapter 1. Planning for Debug Tool

This chapter helps you plan for installing and customizing Debug Tool.

Subtopics:

- 1.1 Planning to Install Debug Tool
- 1.2 Planning for Customizing Debug Tool

1.1 Planning to Install Debug Tool

This section contains planning information to help you properly install Debug Tool:

- What you receive with Debug Tool
- What you need to install Debug Tool
- Planning where to install Debug Tool
- VTAM considerations
- CICS considerations
- Selecting national language support
- Special considerations
- Checking service updates
- Publications useful during installation

Subtopics:

- 1.1.1 What You Receive with Debug Tool
- 1.1.2 What You Need to Install Debug Tool
- 1.1.3 Planning Where to Install Debug Tool
- 1.1.4 VTAM Considerations
- 1.1.5 CICS Considerations
- 1.1.6 Selecting National Language Support
- 1.1.7 Special Considerations
- 1.1.8 Checking Service Updates
- 1.1.9 Publications Useful During Installation

1.1.1 What You Receive with Debug Tool

You receive Debug Tool when you order the full function offering of any of the following IBM compiler products:

- IBM C for VSE/ESA (C/VSE)
- IBM COBOL for VSE/ESA (COBOL/VSE)
- IBM PL/I for VSE/ESA (PL/I VSE)

When you order Debug Tool as part of the full function offering of any of the IBM compilers listed above, you receive:

- Basic machine-readable material
- Basic unlicensed publications

Table 2 lists the component identifiers (COMP IDs) and component level codes (CLCs) for Debug Tool. You can install all the Debug Tool components you received, or the Debug Tool base component and optionally one or more of the language-specific components, depending on your needs. Use this table to determine which components you want to install.

Table 2. Component ID and CLC

Component ID	CLC	Description
5686-A02-00	1G8	Debug Tool Base with uppercase and mixed-case U.S. English
5686-A02-01	1G9	Debug Tool Japanese NLF

The following sections describe these materials and publications.

Subtopics:

- 1.1.1.1 Distribution Media
- 1.1.1.2 Program Documentation

1.1.1.1 Distribution Media

You receive Debug Tool on one of the following:

- 9-track magnetic tape written at 6250 BPI
- IBM 3480 or IBM 3490 cartridge
- ¼-inch tape cartridge (Not available with C/VSE)
- 4-mm DAT cartridge

The cartridge or tape contains all the programs and data you need to install Debug Tool.

Basic Machine-Readable Material: Table 3 describes the Debug Tool optional program tape distributed with the full function C/VSE. Table 4 describes the Debug Tool optional program tape distributed with the full function COBOL/VSE. Table 5 describes the Debug Tool optional program tape distributed with the full function PL/I VSE.

Depending on the distribution medium feature you ordered, you will receive basic machine-readable material on one of the media described in these tables.

To simplify installation and distribution, the Debug Tool distribution media include uppercase U.S. English (UEN), mixed-case U.S. English (ENU),

and the Japanese (JPN) national language feature, regardless of the feature number you used to order your full function high-level language product.

The machine-readable material for Debug Tool contains the same information regardless of which full function high-level language product it is distributed with. If your installation uses multiple high-level languages, you only need to order Debug Tool with one of them. If, however, Debug Tool is ordered with multiple languages, you only need to install it once.

Table 3. Basic material: Debug Tool Optional Program Tape for C/VSE

Medium	Feature number	Physical volume	External label	Volser
6250 tape	5795 5260	1	DEBUG TOOL/VSE R1	unlabeled
IBM 3480 and IBM 3490	5976 5261	1	DEBUG TOOL/VSE R1	unlabeled
4-mm DAT cartridge	6300 5262	1	DEBUG TOOL/VSE R1	unlabeled

Table 4. Basic material: Debug Tool Optional Program Tape for COBOL/VSE

Medium	Feature number	Physical volume	External label	Volser
6250 tape	5831 5949 5962	1	DEBUG TOOL/VSE R1	unlabeled
IBM 3480 and IBM 3490	5832 5950 5963	1	DEBUG TOOL/VSE R1	unlabeled
¼-inch tape	5834 5952 5965	1	DEBUG TOOL/VSE R1	unlabeled
4-mm DAT cartridge	5701 5719 5717	1	DEBUG TOOL/VSE R1	unlabeled

Table 5. Basic material: Debug Tool Optional Program Tape for PL/I VSE

Medium	Feature number	Physical volume	External label	Volser
6250 tape	5831	1	DEBUG TOOL/VSE R1	unlabeled
IBM 3480 and IBM 3490	5832	1	DEBUG TOOL/VSE R1	unlabeled
¼-inch tape	5834	1	DEBUG TOOL/VSE R1	unlabeled
4-mm DAT cartridge	5701 5211(1)	1	DEBUG TOOL/VSE R1	unlabeled

Note:
1. Europe Middle East Africa (EMEA) only.

The file content of the distribution tape you receive depends upon the form in which you receive Debug Tool. If you receive Debug Tool on the VSE/ESA optional program tape, there might be other licensed programs on the tape. If, however, you ordered one of the languages separately, you will receive a separate distribution tape containing only Debug Tool. Table 6 describes the file contents of the separately-orderable licensed program distribution tape.

Table 6. Distribution media: file contents

File	Contents
1	Header file containing Debug Tool copyright statement
2	Backup file ID DTVSE.BASE.1.1.0 followed by an MSHP System History File
3	Debug Tool library file containing the Debug Tool with uppercase and mixed-case U.S. English
4	Header file containing Debug Tool copyright statement
5	Backup file ID DTVSE.JPN..1.1.0 followed by an MSHP System History File
6	Debug Tool library file containing the Debug Tool Japanese NLF
7	Null (Tape Mark)
8	End of backup (EOB) record
9	Null (Tape Mark)

Optional Material: There are no optional machine-readable materials for Debug Tool.

1.1.1.2 Program Documentation

This section identifies the basic and optional Debug Tool documentation you receive.

Basic Unlicensed Publications: Table 7 identifies the basic publications for Debug Tool. Unless otherwise noted, you receive one of each of these publications when you receive the basic materials for Debug Tool. For additional copies, contact your IBM representative.

Table 7. Basic Material: Unlicensed Publications

Publication title	Form number
Debug Tool for VSE/ESA Fact Sheet	GC26-8925(1)
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798(1)
Debug Tool for VSE/ESA User's Guide and	SC26-8797(1)

Reference	
Note:	
1. Add "N:" before each form number (for example N:SC26-8798) to specify publications in Japanese.	

Optional Unlicensed Program Publications: There are no optional unlicensed program publications for Debug Tool.

For a list of publications for related products, see "Bibliography" in topic BIBLIOGRAPHY.

1.1.2 What You Need to Install Debug Tool

The following sections describe system requirements for installing Debug Tool.

Subtopics:

- 1.1.2.1 Licensed Programs
- 1.1.2.2 DASD Storage

1.1.2.1 Licensed Programs

Debug Tool runs under VSE/ESA with the required licensed programs listed in Table 8 and optional licensed programs listed in Table 9. You should install all licensed programs with the minimum release listed or with any subsequent release.

The licensed programs listed in Table 8 are required to install and customize Debug Tool, or to run Debug Tool.

Table 8. Required Licensed Programs for Debug Tool

Required Licensed Program	Minimum Release	Program Number
One of: VSE/ESA VSE/ESA	Version 1 Release 4 Version 2 Release 1	5750-ACD 5690-VSE
One of: IBM Language Environment for VSE/ESA including the C-specific base component VSE C Language Run-Time Support feature of VSE/ESA Version 2	Version 1 Release 4	5686-094 5690-VSE

Release 2		
One of:		
IBM C for VSE/ESA	Release 1	5686-A01
IBM COBOL for VSE/ESA(1)	Release 1	5686-068
IBM PL/I for VSE/ESA(1)	Release 1	5686-069

Note:

1. To run Debug Tool with the output from this compiler the LE/VSE run-time library support for the compiler language is required.

The licensed programs listed in Table 9 can optionally be used with Debug Tool.

Table 9. Optional Licensed Programs for Debug Tool

Optional Licensed Program	Minimum Release	Program Number
CICS/VSE	Version 2 Release 3	5686-026
CSP/AD	Version 3 Release 3	5668-813
CSP/AE	Version 3 Release 3	5668-814
DFSORT/VSE	Version 3 Release 1	5746-SM3
BookManager/Read	Release 2 (to view softcopy documentation)	73F6-023 (Read/2)
IBM DL/I DOS/VS	Release 10	5746-XX1
IBM DOS/VS Sort/Merge	Version 2 Release 5	5746-SM2
QMF/VSE	Version 3 Release 2	5648-061
SQL/DS	Version 3 Release 4	5688-103

1.1.2.2 DASD Storage

When installing Debug Tool, you must provide DASD storage for the VSE Librarian library.

Table 10 lists the estimated minimum DASD storage required for VSE Librarian library storage. The storage requirements given in Table 10 are in addition to the storage requirements of any other licensed programs installed in the library.

You should consider allowing 10% to 15% extra library storage for any future enhancements and service updates.

Table 10. Approximate Library Storage Requirements for Debug Tool

Component	LIBR BLKS(1)	3380 TRK	3390 TRK	9345 TRK	FBA BLKS
Debug Tool Base	4600	150	120	150	9200
Debug Tool Japanese NLF	220	8	7	8	440
Total	**4820**	**158**	**127**	**158**	**9640**

Note:
1. One (1) library block equals 1024 bytes.

1.1.3 Planning Where to Install Debug Tool

The installation procedures described in Chapter 2, "Installing Debug Tool" in topic 2.0 assume that all components of Debug Tool are to be installed in the PROD sublibrary of the PRD2 library. You can choose to install Debug Tool into any other library and sublibrary.

When you have decided what library and sublibrary to use, make sure the library has sufficient free space to hold the Debug Tool feature. Table 10 in topic 1.1.2.2 lists the minimum library storage requirements.

You can check the number of free library blocks in a library using the LIBR program with the LISTDIR command. For more information about the LIBR program, see *VSE/ESA System Control Statements*.

1.1.4 VTAM Considerations

If you plan to use Debug Tool to debug your batch applications interactively you must add the Debug Tool VTAM definitions to your system by following the instructions in "Step 4: Configure VTAM" in topic 2.5.

1.1.5 CICS Considerations

If you plan to use Debug Tool to debug your CICS applications you must update your CICS system by following the instructions in "Step 5: Configure CICS" in topic 2.6.

1.1.6 Selecting National Language Support

Uppercase U.S. English is included in the base component for Debug Tool. You do not have to install the uppercase U.S. English as a separate component. The mixed-case U.S. English and Japanese national language features are included in separate components, that you need to install if you require these languages.

When installing Debug Tool, you cannot choose the default national language for Debug Tool messages. Debug Tool executes in an LE/VSE environment and therefore the national language default for Debug Tool is the same as the national language default for LE/VSE. If you need to change the national language used for a particular execution of Debug Tool, see"Changing the National Language for Debug Tool" in topic 4.2 .

Note: If your default national language for LE/VSE is mixed-case U.S. English or Japanese you **must** install the appropriate national language feature of Debug Tool.

1.1.7 Special Considerations

There are some special requirements that must be considered when using Debug Tool.

Subtopics:

- 1.1.7.1 Debug Tool Run-Time Environment
- 1.1.7.2 VSE Partition Requirements
- 1.1.7.3 PL/I VSE
- 1.1.7.4 Running CICS in 370 Mode

1.1.7.1 Debug Tool Run-Time Environment

Debug Tool requires the LE/VSE run-time library and the LE/VSE C-specific run-time library. To run Debug Tool you must install the base and C components of LE/VSE or the VSE C Language Run-Time Support feature of VSE/ESA Version 2 Release 2.

If you are writing your programs in COBOL/VSE or PL/I VSE then the LE/VSE run-time library components that match the language you are using must be installed.

1.1.7.2 VSE Partition Requirements

The minimum recommended partition size for batch programs running with Debug Tool is 6 megabytes. Batch programs can run with Debug Tool in either a static or dynamic partition.

The partition in which an interactive Debug Tool session starts is unavailable to other users for the duration of that Debug Tool session.

1.1.7.3 PL/I VSE

PL/I customers who want to use the compiler listing exit, supplied with Debug Tool, to write the compiler listing to a file or Librarian sublibrary member, must ensure that PTF UN96230 is installed on their system. This PTF introduces the new compiler list exit capability to the PL/I compiler.

1.1.7.4 Running CICS in 370 Mode

When running CICS with VSE/ESA 1.4 in 370 mode, the maximum size of a program that CICS can load in the dynamic storage area (DSA) is 512KB. The LE/VSE C event handler, CEEEV003, and the LE/VSE C I/O extensions module, EDCZ24, are larger than 512KB. Therefore, if you run Debug Tool under CICS in 370 mode, you must place the phases CEEEV003 and EDCZ24 the SVA. If you do not do so, LE/VSE will not initialize under CICS in 370 mode.

1.1.8 Checking Service Updates

Before installing Debug Tool, check with your IBM Support Center or use either Information/Access or SoftwareXcel Extended to see whether there is additional service information you need. To get this information, specify the following:

UPGRADE DTVSE110
SUBSET A021G8
A021G9

1.1.9 Publications Useful During Installation

For a list of VSE/ESA publications you might find useful during installation, see "Bibliography" in topic BIBLIOGRAPHY.

1.2 Planning for Customizing Debug Tool

This section contains planning information to help you customize Debug Tool:

- Planning to install in the Shared Virtual Area (SVA)
- Planning to change the national language for Debug Tool

Subtopics:

- 1.2.1 Planning to Install in the Shared Virtual Area
- 1.2.2 Planning to Change the National Language for Debug Tool

---

1.2.1 Planning to Install in the Shared Virtual Area

By placing your Debug Tool routines in the shared virtual area (SVA) you will:

- Reduce the overall system storage requirements by making the routines shareable
- Reduce the initialization/termination time for each invocation of Debug Tool, because phase load time decreases

Choose which routines to put in the SVA. Appendix B, "SVA Space Requirements" in topic B.0 provides an estimate of the amount of space used by each phase that can be put in the SVA. Most of the phases can be put in the 31-bit SVA. The supplied SVA loadlist $SVAEQA contains all of the phases listed in Table 17 and Table 18.

1.2.2 Planning to Change the National Language for Debug Tool

Debug Tool runs in an LE/VSE environment and therefore the default national language for Debug Tool is the same as the default national language for LE/VSE. You can change the national language used for a particular execution of Debug Tool by using the LE/VSE run-time option NATLANG. For details, see "Changing the National Language for Debug Tool" in topic 4.2.

2.0 Chapter 2. Installing Debug Tool

This chapter describes the installation method and the step-by-step procedures you use to install and activate the functions of Debug Tool.

Subtopics:

- 2.1 Overview of Installation
- 2.2 Step 1: Backup the Original System
- 2.3 Step 2: Allocate Space for the Library
- 2.4 Step 3: Install Debug Tool
- 2.5 Step 4: Configure VTAM
- 2.6 Step 5: Configure CICS

2.1 Overview of Installation

You install Debug Tool using the Maintain System History Program (MSHP), or by using the VSE/ESA Interactive Interface.

Subtopics:

- 2.1.1 Checklist for Installing Debug Tool

2.1.1 Checklist for Installing Debug Tool

Table 11 lists the steps and associated jobs for installing Debug Tool. The remaining sections in this chapter describe each step. You can use Table 11 as a checklist.

Table 11. Summary of steps for installing Debug Tool

Step	Description	Installation job	Topic
__ 1	Backup the original system	--	2.2
__ 2	Allocate space for the library (Omit if using the default sublibrary)		2.3
	In VSAM Space	EQAWVDEF EQAWLDEF	2.3.1
	In Non-VSAM Space	EQAWNDEF	2.3.2
__ 3	Install Debug Tool		2.4

	Method 1. Install Debug Tool using the Interactive Interface	--	2.4.1
	Method 2. Install Debug Tool using a batch job	EQAWINST	2.4.2
__ 4	Configure VTAM	--	2.5
__ 5	Configure CICS	EQACCSD	2.6
__ 6	Run the installation verification		3.0

2.2 Step 1: Backup the Original System

Make a backup copy of your current Debug Tool library or the library you intend to install Debug Tool into, and the system history file.

For information about backing up libraries and the system history file, see *VSE/ESA System Control Statements*.

2.3 Step 2: Allocate Space for the Library

By default, Debug Tool is installed into the PRD2.PROD sublibrary. If you want to install Debug Tool into the default sublibrary, go to "Step 3: Install Debug Tool" in topic 2.4. If you want to install Debug Tool into a sublibrary other than PRD2.PROD, then continue with this step.

Decide whether you want to install Debug Tool into a library defined in VSAM space or in non-VSAM space.

If you want to install Debug Tool into a library defined in VSAM space, you will need to:

1. Define the VSAM cluster for the library using the VSAM utility IDCAMS.
2. Define the library and sublibrary using the VSE Librarian utility LIBR.
3. If the library is not already defined in your standard labels or in your label procedures, add the necessary label information to the appropriate procedure.

See "Defining a Library in VSAM Space" in topic 2.3.1 for a sample job to define a library in VSAM space.

If you want to install Debug Tool into a library defined in non-VSAM space, you will need to:

1. Decide where to allocate space for the Debug Tool library.

2. Define the library and sublibrary using the VSE Librarian utility LIBR.

3. If the library is not already defined in your standard labels or in your label procedures, add the necessary label information to the appropriate procedure.

See "Defining a Library in Non-VSAM Space" in topic 2.3.2 for sample job control to define a library in non-VSAM space.

Subtopics:

- 2.3.1 Defining a Library in VSAM Space
- 2.3.2 Defining a Library in Non-VSAM Space

2.3.1 Defining a Library in VSAM Space

The sample job in Figure 1 defines a library in VSAM space. You will need to tailor some statements to suit your installation. The tailoring requirements for each statement are discussed in the notes following Figure 1.

```
// JOB  EQAWVDEF
*
*    Define the Debug Tool VSAM cluster
*
// OPTION LOG
*                  ________________________________________  1
*
// DLBL  IJSYSUC,'VSESP.USER.CATALOG',,VSAM
// EXEC  IDCAMS,SIZE=AUTO
    ____________________________________________________  2

   DELETE (DTVSE.LIBRARY) CLUSTER PURGE
   DEFINE CLUSTER - "
               ___|____________________________________  3

          (NAME(DTVSE.LIBRARY) -
          VOLUME(volser) - ________________________  4
          NONINDEXED -
          RECORDFORMAT(NOCIFORMAT) -
          SHR(3) -
          TRACKS(primary secondary)) - _____________  5
          DATA(NAME(DTVSE.LIBRARY.DATA))
                    "
/*                  |____________________________________  6
/&
```

Note: A hyphen at the end of a statement indicates a continuation of the VSAM parameter. The hyphen may be in any column but must be preceded by at least one blank.

Figure 1. Define the Debug Tool Library Cluster in VSAM Space.

1 Specify the catalog name.

Change the catalog name (VSESP.USER.CATALOG in the example) to the catalog in which the Debug Tool library is to be defined.

2 Delete the cluster before redefining.

An IDCAMS DELETE statement is included before the DEFINE statement so that the job can be rerun if necessary. When the job is run for the first time, the DELETE action will result in the following messages:

```
IDC3012I ENTRY DTVSE.LIBRARY NOT FOUND
IDC3009I ** VSAM CATALOG RETURN CODE IS 8 - REASON CODE IS IGG0CLCG-6
IDC0551I **ENTRY DTVSE.LIBRARY NOT DELETED
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 8
```

These messages can be ignored. IDCAMS will continue to define the cluster.

3 Specify the cluster name.

Change the cluster name (DTVSE.LIBRARY in the example) of the Debug Tool library to suit your installation.

4 Specify the volume serial number.

Change the *volser* parameter to the serial number of the volume on which the Debug Tool library is to be defined.

5 Specify the amount of space.

Change the *primary* and *secondary* parameters to specify the number of tracks to be allocated for primary and secondary space for the Debug Tool library. If you are using FBA devices, change this parameter to

```
BLOCKS(primary secondary)
```

For information about library storage requirements, see "DASD Storage" in topic 1.1.2.2.

6 Specify the data component name.

Change the data component name (DTVSE.LIBRARY.DATA in the example) of the Debug Tool library to conform with the cluster name you specified at **3** .

When you have defined the VSAM cluster for the library, use the sample job in Figure 2 to define the library and sublibrary. You will need to tailor some statements to suit your installation. The tailoring requirements for each statement are discussed in the notes following Figure 2.

```
// JOB  EQAWLDEF
*
*    Define the Debug Tool Library
*                  ________________________________________  1
*
// DLBL  IJSYSUC,'VSESP.USER.CATALOG',,VSAM
*                ________________________________________  2
*
// DLBL  DTVSE,'DTVSE.LIBRARY',,VSAM
// EXEC  LIBR ____________________________________________  3
 DELETE LIB=DTVSE
 DEFINE LIB=DTVSE
 DEFINE SUBLIB=DTVSE.BASE
/*             "
/&             |_______________________________________  4
```

Figure 2. Define the LE/VSE Library.

1 Specify the VSAM catalog.

Change the catalog name (VSESP.USER.CATALOG in the example) to match the name of the catalog you specified at **1** in Figure 1.

2 Specify the label information.

Change the file-id (DTVSE.LIBRARY in the example) of the Debug Tool library to match the name you specified at **3** in Figure 1.

3 Invoke the Librarian to define the library.

The Librarian job step includes a LIBR DELETE statement before the LIBR DEFINE statement, so the job can be rerun if necessary. This means the following messages will be issued when the job runs for the first time:

```
  L255I  LIBRARY DTVSE - OPEN FAILURE
  L027I  ABNORMAL END DURING DELETE COMMAND
         PROCESSING
  L113I  RETURN CODE OF DELETE IS 8
```

These messages are informational and can be ignored. The job continues to allocate the library.

4 Specify the sublibrary.

Change the sublibrary name to suit your installation.

Note: You can also use the Interactive Interface dialog "Resource Definition" to define a library in VSAM space.

2.3.2 Defining a Library in Non-VSAM Space

On the disk volume to be used for the library, identify suitable areas of

free space. To do this, list the volume table of contents (VTOC) of the disk to be used. The sample job in Figure 3 shows a job to list the VTOC of the volume with serial number SYSWK1. Change the volume serial number in the ASSGN statement **1** as required.

```
// JOB  EQAWVTOC
*
*    List Volume Table of Contents
*
// ASSGN SYS004,DISK,TEMP,VOL=SYSWK1,SHR
// ASSGN SYS005,SYSLST        "
// EXEC LVTOC                 |_____________________  1
/*
/&
```

Figure 3. List the Contents of Volume SYSWK1.

When you have decided where to allocate the Debug Tool library, you can use the sample job shown in Figure 4 to help you define the library. You will need to tailor some statements to suit your installation. The tailoring requirements for each statement are discussed in the notes following Figure 4.

```
// JOB  EQAWNDEF
*
*    Define a Non-VSAM Library for Debug Tool
*
// OPTION LOG    ____________________________________  1
*
// DLBL  DTVSE,'DTVSE.LIBRARY',99/365,SD
// EXTENT SYS002,volser,,,rtrk,ntrk _________________  2
// ASSGN SYS002,DISK,VOL=volser,SHR _________________  3
*
// EXEC  LIBR _______________________________________  4
 DELETE LIB=DTVSE
 DEFINE LIB=DTVSE
 DEFINE SUBLIB=DTVSE.BASE
/*             "
/&             |__________________________________  5
```

Figure 4. Define the LE/VSE Library.

1 Specify the label information.

Change the filename (DTVSE in the example) and file-id (DTVSE.LIBRARY in the example) of the Debug Tool library to suit your installation.

2 Specify the extent information:

- The *volser* parameter indicates the volume serial number of the volume on which the Debug Tool library is to be defined. Change this parameter as required.

- The *rtrk* parameter indicates the beginning track or FBA block of the disk extent. Change this parameter as required.

- The *ntrk* parameter indicates the number of tracks or FBA blocks to be allocated for the Debug Tool library. Change this parameter as required. For information about library storage requirements, see "DASD Storage" in topic 1.1.2.2.

3 Specify the volume serial number.

The *volser* parameter indicates the volume serial number of the volume on which the Debug Tool library is to be defined. Change this parameter to match the volume serial number specified on the EXTENT statement.

4 Invoke the Librarian to define the library.

The Librarian job step includes a LIBR DELETE statement before the LIBR DEFINE statement, so the job can be rerun if necessary. This means the following messages will be issued when the job runs for the first time:

```
L101I  LIBRARY DTVSE DOES NOT EXIST
L027I  ABNORMAL END DURING DELETE COMMAND
       PROCESSING
L113I  RETURN CODE OF DELETE IS 8
```

These messages are informational and can be ignored. The job continues to allocate the library.

5 Specify the sublibrary.

Change the sublibrary name to suit your installation.

2.4 Step 3: Install Debug Tool

You can install Debug Tool using either the VSE/ESA Interactive Interface or a batch installation job.

Subtopics:

- 2.4.1 Method 1: Install Debug Tool Using the Interactive Interface
- 2.4.2 Method 2: Install Debug Tool Using a Batch Job

2.4.1 Method 1: Install Debug Tool Using the Interactive Interface

The VSE/ESA Interactive Interface enables you to use dialog requests to install Debug Tool. For more information about installing licensed

programs using the Interactive Interface, see *VSE/ESA Installation*.

To install Debug Tool using the Interactive Interface, do the following:

1. Logon to the Interactive Interface as the system administrator.

2. Mount the Debug Tool distribution tape on an available tape drive.

3. In the following menus, specify the items that appear following the ===> symbol.

   a. **VSE/ESA FUNCTION SELECTION** menu:

      ===> 1 (Installation)

   b. **INSTALLATION** menu:

      ===> 1 (Install Programs - V2 Format)

      Note: You can use option 1 for both a stacked distribution tape containing one or more optional licensed programs, and a non-stacked tape containing just Debug Tool. Both are in Librarian V2 format.

   c. **INSTALL PROGRAMS - V2 FORMAT** menu:

      ===> 1 (Prepare for Installation)

   d. **PREPARE FOR INSTALLATION** menu:

      ===> cuu (address of drive with distribution tape)

   e. **JOB DISPOSITION** menu:

      Make any changes required and press ENTER to submit the job.

      The output listing from this job lists the optional programs on the distribution tape with program identifiers and recommended library sizes. The program identifier for each Debug Tool component has the format DTVSE.xxxx.1.1.0.

   f. Check the output from the batch job created by the previous steps to ensure that it was successful. Once the batch job created by these steps has successfully run, the program identifiers of the optional programs on the distribution tape are also automatically entered on the **INSTALL ADDITIONAL PROGRAM(S) FROM TAPE** menu.

   g. After the batch job created by the previous step has successfully run, return to the **VSE/ESA FUNCTION SELECTION** menu:

      ===> 1 (Installation)

h. **INSTALLATION** menu:

```
===> 1  (Install Programs - V2 Format)
```

i. **INSTALL PROGRAMS - V2 FORMAT** menu:

```
===> 2  (Install Program(s) from Tape)
```

j. **INSTALL ADDITIONAL PROGRAM(S) FROM TAPE** menu:

Enter 1 (install) in the OPT field against the required identifiers of the format DTVSE.xxxx.1.1.0, and 2 (skip installation) against any other optional licensed programs you do not intend to install at this time.

If you are not installing Debug Tool in the default sublibrary PRD2.PROD, enter the name of your library and sublibrary on this screen.

k. Press PF5 to generate the installation job.

l. **INSTALL ADDITIONAL PROGRAM(S) FROM TAPE** menu:

```
===> 1  (Save the list)
```

m. **INSTALL ADDITIONAL PROGRAM(S) FROM TAPE** menu:

```
===> cuu  (address of drive with distribution tape)
```

n. **JOB DISPOSITION** menu:

Make any changes required and press ENTER to submit the job and install Debug Tool.

o. Check the output from the batch job created by the previous steps to ensure that the install was successful. If you are installing Debug Tool from a non-stacked tape, you receive the following message:

```
  IESI0083I  TAPE IS NOT V2-STACKED
```

This message is for information only and can be ignored. The Librarian RESTORE job ends with a return code of 4. This is not an error. Installation was successful.

Subtopics:

- 2.4.1.1 Condition Code and Messages
- 2.4.1.2 List the Directories and Components

2.4.1.1 Condition Code and Messages

If you receive a condition code greater than 4:

- Check the list output for error conditions.
- See *VSE/ESA Messages and Codes* for corrective action.
- Correct the error.
- Rerun the job.
- Recheck the condition code.

2.4.1.2 List the Directories and Components

If you want to print the directory entries from the sublibrary where Debug Tool is installed, and the component records from the system history file, you can now submit a batch job to do this. The last two steps of the job shown in Figure 5 in topic 2.4.2 will produce these lists.

2.4.2 Method 2: Install Debug Tool Using a Batch Job

A sample batch job to install Debug Tool using MSHP is shown in Figure 5. This sample job uses the MSHP system history file that already exists as part of the VSE system. This system history file might already be defined in the system standard labels; if not, make sure that DLBL and EXTENT statements, with the necessary information for the system history file, are included in the job stream.

Depending on how you ordered the Debug Tool feature, you will have received a stacked tape containing one or more optional licensed programs, or a non-stacked tape containing only the Debug Tool feature. The job shown in Figure 5 will handle both stacked and non-stacked tapes.

To install Debug Tool using a batch job, create and tailor the job stream shown in Figure 5, mount the distribution tape, and run the installation job.

You will need to tailor some statements to suit your installation. The tailoring requirements for each statement are discussed in the notes following Figure 5.

```
// JOB  EQAWINST
*
*    Install Debug Tool
*
// OPTION LOG
*   ____________________________________________________  1
*
*  // DLBL  DTVSE,'DTVSE.LIBRARY'
*  // EXTENT SYS005
*  // ASSGN  SYS005,DISK,VOL=volser,SHR
*
// ASSGN  SYS006,cuu _____________________________________  2
*
// MTC    REW,SYS006
* ------------------------------------------
*  This step installs Debug Tool
*  from the distribution tape
*  using the VSE system history file
* ------------------------------------------
```

```
// EXEC  MSHP,SIZE=900K,PARM='PIDSTACKED' _____________  3
 INSTALL PROD FROMTAPE ID='DTVSE.BASE.1.1.0' -
    PROD INTO=PRD2.PROD
 INSTALL PROD FROMTAPE ID='DTVSE.JPN..1.1.0' - _______  4
    PROD INTO=PRD2.PROD
/*
* ------------------------------------------
*  List the Debug Tool Library
* ------------------------------------------
// EXEC  LIBR,PARM='MSHP' _____________________________  5
 LISTDIR SUBLIB=PRD2.PROD -
     OUTPUT=NORMAL -
     UNIT=SYSLST
/*
* ------------------------------------------
*  Retrace the Debug Tool product
* ------------------------------------------
// EXEC  MSHP,SIZE=900K _______________________________  6
RETRACE COMPONENT IDENTIFIER=5686-A02-00
RETRACE COMPONENT IDENTIFIER=5686-A02-01
/*
// MTC  RUN,SYS006
/*
/&
```

Figure 5. Install Debug Tool

1 Specify the label information.

If you are installing Debug Tool into a sublibrary other than the default (PRD2.PROD), code the DLBL job control statement (EXTENT and ASSGN statements if necessary) to match those you specified in Figure 2 in topic 2.3.1 or Figure 4 in topic 2.3.2. The library name must match the name used in the LIBR job.

Usually, you do not need to specify label information for the system history file. Your installation should have a permanent system standard label for this, with IJSYSHF as the filename. (IJSYSHF is the default filename that MSHP looks for in a label statement.)

2 Assign the distribution tape.

Replace *cuu* with the address of the tape drive on which you have mounted the distribution tape.

3 Install Debug Tool.

This job step invokes MSHP to install Debug Tool into the sublibrary identified on the INTO operand of the INSTALL statement. If you are installing Debug Tool into a sublibrary other than the default, change the name of the sublibrary on the INTO operand of each INSTALL statement to the name of the sublibrary into which you are installing the component specified in the FROMTAPE ID operand. For example, if you are installing Debug Tool into sublibrary DTVSE.BASE, change PRD2.PROD to DTVSE.BASE.

For more information about the MSHP install options, see *VSE/ESA System Control Statements*.

4 Select national language feature components.

The Debug Tool national language is the language used for the Debug

Tool messages. The base component of Debug Tool includes uppercase and mixed-case U.S. English national language support.

You can install the Japanese national language feature from the Debug Tool distribution tape. If you do not want to install the Japanese national language feature, remove this INSTALL statement.

Note: If your default national language for LE/VSE is Japanese you **must** install the Japanese national language feature of Debug Tool.

5 List the directory entries.

This job step invokes LIBR to list the directory entries of the sublibrary where Debug Tool was installed. Remove this step if the directory lists are not required. If you are installing Debug Tool into a sublibrary other than the default, change the name of the sublibrary in this job step accordingly.

The names of most modules in the Debug Tool library have a three-character prefix of EQA.

6 Retrace the Debug Tool product in the system history file.

This step prints the component records from the system history file for Debug Tool. Remove this step if an MSHP retrace listing is not required.

If you need to rerun this install job, make sure you first restore the system history file, which you should have backed up before you started the install process. You should also rerun the library allocation job, if applicable.

2.5 Step 4: Configure VTAM

This step is only required if you want to debug your batch programs interactively using Debug Tool.

1. Either:

   - Copy the EQAWAPPL.B member, supplied in the Debug Tool install sublibrary, to the SOURCE sublibrary containing your VTAM definition members. If you installed Debug Tool using the interactive interface, this member is also supplied as EQAWAPPL in ICCF library 62.

   or

   - Copy the Debug Tool minor node name definitions (APPL statements) from EQAWAPPL.B into one of your existing application major node definitions.

   Note: The supplied EQAWAPPL.B defines 20 application minor node names, DTVSE001 to DTVSE020. You can define up to 999 Debug Tool minor node names: the first 5 characters of each name must be DTVSE and the last 3 characters must be consecutive 3-digit numbers, starting at 001.

The number of minor node names you define should be sufficient to cater for the maximum number of concurrent users of Debug Tool using batch interactive debugging sessions.

2. Add the SOURCE sublibrary member name (EQAWAPPL, or the name of the updated member) to your VTAM start options configuration file, ATCCON*xx*.

3. Restart your VTAM system to activate the Debug Tool definitions.

   Until you restart your VTAM system, you can activate the Debug Tool definitions by entering the following command from your VTAM console:

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

   where *member-name* is the SOURCE sublibrary member name (for example, EQAWAPPL).

2.6 Step 5: Configure CICS

This step is only required if you want to debug your CICS programs using Debug Tool.

1. Complete the CICS installation requirements for the LE/VSE base component and LE/VSE C-specific run-time component. For details, see the *LE/VSE Installation and Customization Guide*.

2. Either:

   - Add the Debug Tool sublibrary to the CICS LIBDEF PHASE,SEARCH chain in your CICS startup job.

   or

   - Load the Debug Tool phases into the SVA. For details, see "Loading Debug Tool into the Shared Virtual Area" in topic 4.1

   Note: If you are running CICS in 370 mode, see "Running CICS in 370 Mode" in topic 1.1.7.4.

3. Either:

   - Run the EQACCSD.Z job supplied in the Debug Tool install sublibrary. This job merges the Debug Tool definitions into your CICS CSD file and adds the group EQA to the CICS startup group list VSELIST (making the group EQA available after a CICS restart). If you installed Debug Tool using the interactive interface, this job is also supplied as EQACCSD in ICCF library 62.

     Note: Your CICS system does not have to be stopped to run this job.

   or

   - If you prefer to assemble CICS tables rather than use the CSD, update your PCT and PPT with the entries in the sample members EQACPCT.Z and EQACPPT.Z, respectively. Both of these members are in the Debug Tool install sublibrary. If you installed Debug Tool using the interactive interface, these samples are also supplied as EQACPCT and EQACPPT in ICCF library 62.

4. Restart your CICS system.

3.0 Chapter 3. Verifying the Installation of Debug Tool

This chapter describes how to verify the installation of Debug Tool for each of the high-level languages that it supports.

For each high-level language supported by Debug Tool, there are three installation verification exercises:

- Batch debug using a commands file
- Batch execution with interactive debug
- Interactive debug with CICS

Table 12 lists the names of the installation verification exercises for each high-level language. You should run the installation verification exercises that correspond to the high-level languages you have installed.

Table 12. Installation Verification Samples

Verification Type	C/VSE	COBOL/VSE	PL/I VSE
Batch debug using a commands file	EQAWIVH1	EQAWIVC1	EQAWIVP1
Batch execution with interactive debug	EQAWIVH2	EQAWIVC2	EQAWIVP2
Interactive debug with CICS	EQAWIVH3	EQAWIVC3 EQAWIVC4	EQAWIVP3

Note:

1. The installation verification samples are supplied as members with a member type of Z in the Debug Tool installation sublibrary (default is PRD2.PROD).
2. If you installed Debug Tool using the VSE/ESA Interactive Interface dialogs, the installation verification samples are also available in ICCF library 62.

Subtopics:

- 3.1 Installation Verification with C/VSE
- 3.2 Installation Verification with COBOL/VSE
- 3.3 Installation Verification with PL/I VSE

3.1 Installation Verification with C/VSE

This section describes the installation verification for Debug Tool with C/VSE.

Subtopics:

- 3.1.1 Batch Debug Using a Commands File
- 3.1.2 Batch Execution with Interactive Debug
- 3.1.3 Interactive Debug with CICS

3.1.1 Batch Debug Using a Commands File

The sample job EQAWIVH1 compiles, link-edits, and runs a C/VSE program, IVPCCC1, that invokes Debug Tool. Modify the sample job control to meet your requirements before submitting the job. Figure 6 shows an extract from EQAWIVH1; tailoring requirements for each statement are discussed in the notes.

```
* $$ JOB JNM=EQAWIVH1,CLASS=W   _____________________________  1
// JOB EQAWIVH1   ____________________________________________  2
*                  ___________________________________________  3
*                 |            ________________________________  4
*
// LIBDEF *,SEARCH(lib.userlib,PRD2.PROD,PRD2.DBASE,PRD2.SCEEBASE)
*                                 "           "
// LIBDEF PHASE,CATALOG=lib.userlib       |           |_________  5
*                       "                 |___________________  6
*                       |_________________________________________  3
*       Step 1: Catalog the C/VSE source file for IVPCCC1
*
// EXEC LIBR
ACCESS SUB=lib.userlib ____________________________________  3
CATALOG IVPCCC1.C R=Y
  .
  .
  .
C/VSE program source
  .
  .
  .
/+
/*
// OPTION CATAL
*
*       Step 2: Compile IVPCCC1
*
// EXEC EDCCOMP,SIZE=EDCCOMP,PARM='/NAME("IVPCCC1")                  X
              INFILE(DD:lib.userlib(IVPCCC1.C)) SOURCE TEST'
/*                  "
*                    |_____________________________________  3
*       Step 3: Link-Edit IVPCCC1
*
// EXEC LNKEDT
/*
*       Step 4: Run the IVP program invoking Debug Tool
*                                                 _________  7
*
// EXEC IVPCCC1,SIZE=AUTO,PARM='TEST(,SYSIPT,,SYSIPT)/'
  .
  .
  .
Debug Tool preferences file
  .
  .
  .
/*
  .
  .
  .
Debug Tool commands file
  .
  .
  .
/*
/&
* $$ EOJ
```

Figure 6. Job Control Extract for EQAWIVH1.

1 Modify the VSE/POWER job control statements for your site.

2 Modify the job card as appropriate for your site.

3 Change *lib.userlib* to your temporary user sublibrary:

- The C/VSE source file cataloged in step 1 is written to this sublibrary.
- The phase IVPCCC1 created by the link-edit in step 3 is written to this sublibrary.
- Debug Tool writes a profile settings file (*member-name*.DTSAFE) to the first sublibrary in the LIBDEF SOURCE search chain. The DTSAFE member is used to save the Debug Tool session settings when your debug session terminates; these settings are used as your defaults for future debugging sessions.

  If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where Debug Tool writes the profile settings file.

  The *member-name* of the profile settings file depends on your VSE/ESA system options:

  - If your VSE/ESA system IPL activates security checking, see Appendix A, "Determining the Default Userid" in topic A.0 for a description of how *member-name* is determined.
  - If your VSE/ESA system does not use security checking then *member-name* is set to DTVSE.

4 If necessary, change the sublibrary to match the sublibrary where Debug Tool is installed.

5 If necessary, change the sublibrary to match the sublibrary where LE/VSE is installed.

6 If necessary, change the sublibrary to match the sublibrary where C/VSE is installed.

7 Any messages produced by EQAWIVH1 will be in your LE/VSE installation default national language.

If you want messages to be produced in a national language other than your LE/VSE installation default, add the NATLANG run-time option by specifying NATLANG(*xxx*) before the slash (/) at the end of the options string, where *xxx* is the three character representation of the language you wish to use (such as JPN, ENU, or UEN).

After you modify the EQAWIVH1 job, submit it. If successful, the job

finishes with return code 2, indicating unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

EQAWIVH1 creates a log on SYSLST that contains the results from the execution of the debug control statements specified in the SYSIPT preferences file and the SYSIPT commands file. Check this log against Figure 7. Except for:

- The date and time
- The address for var1
- The *lib.userlib*

these logs should match, verifying that Debug Tool installed successfully.

```
/* DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0                    */
/* 10/19/96 7:17:23 PM                                                 */
/* (C) COPYRIGHT IBM CORP. 1992, 1996                                  */
 SET MSGID ON ;
 AT 58
   LIST ( "At the breakpoint for line", %LINE ) ;
 GO ;
/* EQA1140I At the breakpoint for line                                 */
/* EQA1141I %LINE = "58\0"                                             */
 LIST var1 ;
/* EQA1141I var1 = 0                                                   */
 STEP 1 ;
 LIST var1 ;
/* EQA1141I var1 = 1                                                   */
 DESCRIBE ATTRIBUTES var1 ;
/* EQA1102I   ATTRIBUTES FOR var1                                      */
/* EQA1105I     ITS LENGTH IS 4                                        */
/* EQA1103I     ITS ADDRESS IS 006A2278                                */
/* EQA1112I     signed int                                             */
 int X ;
 X = var1 ;
 X ++ ;
 LIST X ;
/* EQA1141I X = 2                                                      */
 STEP 1 ;
 LIST var1 ;
/* EQA1141I var1 = 2                                                   */
 STEP 3 ;
/* EQA1140I At the breakpoint for line                                 */
/* EQA1141I %LINE = "58\0"                                             */
 CLEAR AT 58 ;
 STEP 9 ;
 LIST var1 ;
/* EQA1141I var1 = 13                                                  */
 STEP 22 ;
 AT EXIT *
   LIST ( "Exiting ", %CU ) ;
 GO ;
IVPCCC1 Completed
/* EQA1140I Exiting                                                    */
/* EQA1141I %CU = "DD:lib.userlib(IVPCCC1.C)\0"                        */
 SET MSGID OFF ;
/*  QUIT ;                                                             */
```

Figure 7. SYSLST Log Content after Execution of IVPCCC1.

3.1.2 Batch Execution with Interactive Debug

The sample job EQAWIVH2 compiles, link-edits, and runs a C/VSE program, IVPCCC2, that invokes Debug Tool interactively. Modify the sample job control to meet your requirements before submitting the job. Figure 8 shows an extract from EQAWIVH2; tailoring requirements for each statement are discussed in the notes.

```
* $$ JOB JNM=EQAWIVH2,CLASS=W    _____________________________  1
// JOB EQAWIVH2    _____________________________________________  2
*                    ____________________________________________  3
*                   |            ____________________________  4
*
// LIBDEF *,SEARCH=(lib.userlib,PRD2.PROD,PRD2.DBASE,PRD2.SCEEBASE)
*                                 "           "
// LIBDEF PHASE,CATALOG=lib.userlib       |           |________  5
*                        "                |___________________  6
*                       |_____________________________________  3
*       STEP 1: CATALOG the source code file for IVPCCC2
*
// EXEC LIBR
ACCESS SUB=lib.userlib _____________________________________  3
CATALOG IVPCCC2.C R=Y
  .
  .
  .
C/VSE program source
  .
  .
  .
/+
/*
// OPTION CATAL
*
*       Step 2: Compile IVPCCC2
*
// EXEC EDCCOMP,SIZE=EDCCOMP,PARM='/NAME("IVPCCC2")                X
              INFILE(DD:lib.userlib(IVPCCC2.C)) SOURCE TEST'
/*                     "
*                      |_____________________________________  3
*       Step 3: Link-Edit IVPCCC2
*
// EXEC LNKEDT
/*
*       Step 4: Run the IVP program invoking Debug Tool
*               for Interactive Debug             _____________  7
*                                                |            _  8
*
// EXEC IVPCCC2,SIZE=AUTO,PARM='TEST(,SYSIPT,,MFI%vtamluid:*)/'
  .
  .
  .
Debug Tool commands file
  .
  .
  .
/*
/&
* $$ EOJ
```

Figure 8. Job Control Extract for EQAWIVH2.

1 Modify the VSE/POWER job control statements for your site.

2 Modify the job card as appropriate for your site.

3 Change *lib.userlib* to your temporary user sublibrary:

- The C source file created in step 1 is written to this sublibrary.
- The phase IVPCCC1 created by the link-edit in step 3 is written to this sublibrary.
- Debug Tool writes a profile settings file (*member-name*.DTSAFE) to the first sublibrary in the LIBDEF SOURCE search chain. The DTSAFE member is used to save the Debug Tool session settings when your debug session terminates, these settings are used as your defaults for future debugging sessions.

  If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where Debug Tool writes the profile settings file.

  The *member-name* of the profile settings file depends on your VSE/ESA system options:

  - If your VSE/ESA system IPL activates security checking, see Appendix A, "Determining the Default Userid" in topic A.0 for a description of how *member-name* is determined.
  - If your VSE/ESA system does not use security checking then *member-name* is set to DTVSE.

4 If necessary, change the sublibrary to match the sublibrary where Debug Tool is installed.

5 If necessary, change the sublibrary to match the sublibrary where LE/VSE is installed.

6 If necessary, change the sublibrary to match the sublibrary where C/VSE is installed.

7 Change the *vtamluid* in the TEST option to the VTAM LU name of the terminal where your interactive debug session will run. For example if your VTAM LU name is D0800001, change the TEST option to TEST(,,,MFI%D0800001:*).

Note: The VTAM LU must not be in session with another application when Debug Tool attempts to acquire it.

8 Any messages produced by EQAWIVH2 will be in your LE/VSE installation default national language.

If you want messages to be produced in a national language other than your LE/VSE installation default, add the NATLANG run-time option by

specifying NATLANG(*xxx*) before the slash (/) at the end of the options string, where *xxx* is the three character representation of the language you wish to use (such as JPN, ENU, or UEN).

After you modify the EQAWIVH2 job, submit it. Debug Tool uses the C/VSE source file (cataloged in Step 1) during the interactive debug session to display the SOURCE window.

When the execution of the program IVPCCC2 starts, following successful completion of the link-edit step, an interactive debug session starts on the terminal you designated in the TEST option at **7** . The Debug Tool commands specified in the SYSIPT commands file are then executed before you enter commands interactively. After the commands in the commands file have executed, the debug screen looks like the example shown in Figure 9.

Note: The example screen is for a 32-line display and assumes the default settings are being used for Debug Tool. If your display has a different number of lines or is not using the default settings, the display you see will be slightly different to the example.

```
C        LOCATION: "DD:lib.userlib(IVPCCC2.C)" :> 58
COMMAND ===>                                                SCROLL ===> PAGE
MONITOR --+----1----+----2----+----3----+----4----+----5----+----6 LINE: 1 OF 1
****************************** TOP OF MONITOR *******************************
0001  1 i   1
***************************** BOTTOM OF MONITOR *****************************




SOURCE: DD:lib.userlib.(IVPCCC2.C) ----3----+----4----+----5----+ LINE: 58 OF 62
****************************** TOP OF SOURCE ********************************
      58     var1 = var1 + i;                                                .
      59     var1++;                                                         .
      60     }                                                               .
      61  printf("IVPCCC2 Completed\n");                                     .
      62  }                                                                  .
***************************** BOTTOM OF SOURCE ******************************


LOG 0----+----1----+----2----+----3----+----4----+----5----+----6 LINE: 8 OF 13
0008 At the breakpoint for line
0009 %LINE = "58\0"
0010  MONITOR
0011    LIST i ;
0012  LIST var1 ;
0013 var1 = 0
PF  1:?         2:STEP       3:QUIT       4:LIST       5:FIND       6:AT/CLEAR
PF  7:UP        8:DOWN       9:GO        10:ZOOM      11:ZOOM LOG  12:RETRIEVE
```

Figure 9. Initial Debug Session Screen for IVPCCC2.

You are now ready to enter commands in your interactive debug session. The script following leads you through the installation verification. At the command line in your interactive debug session, enter the commands shown following the ===> and press the ENTER key.

The initial SOURCE window is pointing at statement 58 and the value of **i** in the MONITOR window is 1.

```
===> step 1
```

The SOURCE window moves to statement 59, the MONITOR window remains unchanged, and the LOG window shows:

```
      0014  STEP 1 ;
```

```
===> list var1
===> describe attributes var1
```

The SOURCE window remains at statement 59, the MONITOR window remains unchanged, and the LOG window shows:

```
      0015  LIST var1 ;
      0016 var1 = 1
      0017  DESCRIBE ATTRIBUTES var1 ;
      0018    ATTRIBUTES FOR var1
      0019      ITS LENGTH IS 4
      0020      ITS ADDRESS IS 006A2278
      0021      signed int
```

Note: The address you see on line 0020 might vary from the sample shown.

```
===> int x
===> x=var1
===> x++
===> list x
```

The SOURCE window remains at statement 59, the MONITOR window remains unchanged, and the LOG window shows:

```
      0022  int x ;
      0023  x = var1;
      0024  x ++ ;
      0025  LIST x ;
      0026 x = 2
```

```
===> go
```

The SOURCE window moves to statement 58, the value of **i** in the MONITOR window changes to 2, and the LOG window shows:

```
      0027  GO ;
      0028 At the breakpoint for line
      0029 %LINE = "58\0"
```

```
===> clear at 58
===> step 9
```

The SOURCE window moves to statement 59, the value of **i** in the MONITOR window changes to 4, and the LOG window shows:

```
      0030  CLEAR AT 58 ;
      0031  STEP 9 ;



===> list var1
===> step 22
```

The SOURCE window moves to statement 61, the value of **i** in the MONITOR window changes to 10, and the LOG window shows:

```
      0032  LIST var1 ;
      0033 var1 = 13
      0034  STEP 22 ;



===> at exit * list ("Exiting ",%cu)
===> go
```

The SOURCE window moves to statement 62, the MONITOR window remains unchanged, and the LOG window shows:

```
      0035  AT EXIT *
      0036    LIST ( "Exiting ", %CU ) ;
      0037  GO ;
      0038 Exiting
      0039 %CU = "DD:lib.userlib(IVPCCC2.C)\0"



===> quit
```

The SOURCE window remains at statement 62, and the LOG window shows:

```
      0040  QUIT ;
```

and the MONITOR window is replaced with:

```
 ____________________________________________________________________________
|                                                                            |
| C        LOCATION: "DD:lib.userlib(IVPCCC2.C)" :> main                     |
| COMMAND ===>                                              SCROLL ===> PAGE |
| ************************************************************************** |
|                                                                            |
| DO YOU REALLY WANT TO TERMINATE THIS SESSION? N                            |
| ENTER     Y FOR YES   AND   N FOR NO                                       |
|                                                                            |
| ************************************************************************** |
|                                                                            |
```

Overtype the N with Y and press the ENTER key. Your interactive debug session will now terminate and the terminal will return to your default VTAM display.

If successful, the job finishes with return code 2 indicating there were unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

Your interactive debug session creates a log on SYSLST that contains the results of your interactive debug session. Check this log against Figure 10. Except for:

- The date and time
- The address for var1
- The *lib.userlib*

these logs should match, verifying that Debug Tool installed successfully.

```
/* DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0                    */
/* 10/19/96 11:25:58 AM                                                */
/* (C) COPYRIGHT IBM CORP. 1992, 1996                                  */
 SET PACE 1 ;
 AT 58
   LIST ( "At the breakpoint for line", %LINE ) ;
 GO ;
/* At the breakpoint for line                                          */
/* %LINE = "58\0"                                                      */
 MONITOR
   LIST i ;
 LIST var1 ;
/* var1 = 0                                                            */
 STEP 1 ;
 LIST var1 ;
/* var1 = 1                                                            */
 DESCRIBE ATTRIBUTES var1 ;
/*   ATTRIBUTES FOR var1                                               */
/*     ITS LENGTH IS 4                                                 */
/*     ITS ADDRESS IS 006A2278                                         */
/*     signed int                                                      */
 int X ;
 X = var1 ;
 X ++ ;
 LIST X ;
/* X = 2                                                               */
 GO ;
/* At the breakpoint for line                                          */
/* %LINE = "58\0"                                                      */
 CLEAR AT 58 ;
 STEP 9 ;
 LIST var1 ;
/* var1 = 13                                                           */
 STEP 22 ;
 AT EXIT *
   LIST ( "EXITING ", %CU ) ;
 GO ;
IVPCCC2 Completed
/* EXITING                                                             */
/* %CU = "DD:lib.userlib(IVPCCC2.C)\0"                                 */
/*  QUIT ;                                                             */
```

Figure 10. SYSLST Log Content after Execution of IVPCCC2.

3.1.3 Interactive Debug with CICS

Table 13 lists the steps that must be taken to complete the installation

verification for Debug Tool with C/VSE in a CICS environment.

Table 13. Summary of steps for installation verification of C/VSE with CICS.

Step	Description
__ 1	Compile and link-edit the CICS installation verification program
__ 2	Add your user sublibrary to the CICS search chain
__ 3	Define the installation verification program and transaction code to your CICS system
__ 4	Invoke the installation verification program and follow the script

Step 1. Compile and link-edit the CICS installation verification program: The sample job EQAWIVH3 compiles and link-edits a C/VSE program, IVPCCC3, that invokes Debug Tool when the program is started under CICS. Modify the sample job control to meet your requirements before submitting the job. Figure 11 shows an extract from EQAWIVH3; tailoring requirements for each statement are discussed in the notes.

```
* $$ JOB JNM=EQAWIVH3,CLASS=W    _____________________________  1
* $$ PUN DISP=I,PRI=6,CLASS=W
// JOB EQAWIVH3    ____________________________________________  2
// ASSGN SYSIPT,SYSRDR
// EXEC IESINSRT
// JOB EQAWIVH3
*
*       STEP 2: Catalog the source output from the CICS translate
*                          ____________________________________  3
*
// EXEC LIBR,PARM='ACC S=lib.userlib;CATALOG IVPCCC3.C R=Y;'
* $$ END
*       Step 1: CICS Translate IVPCCC3
*                      ________________________________________  4
*
// LIBDEF *,SEARCH=(PRD2.PROD)
// OPTION DECK
// EXEC DFHEDP1$,PARM='CICS'
  .
  .
  .
C/VSE program source
  .
  .
  .
/*
// EXEC IESINSRT
/+
/*
*       Step 3: Compile IVPCCC3
*                      ________________________________________  4
*
// LIBDEF *,SEARCH=(PRD2.PROD,PRD2.DBASE,PRD2.SCEEBASE)
*                               "          "
*                               |          |_____________________  5
*                               |________________________________  6
// LIBDEF PHASE,CATALOG=lib.userlib
*                         "
// OPTION CATAL            |_____________________________________  3
   PHASE IVPCCC3,*
```

```
   INCLUDE DFHELII
// EXEC EDCCOMP,SIZE=EDCCOMP,PARM='RENT                                  X
              INFILE(DD:lib.userlib(IVPCCC3.C)) SOURCE TEST'
/*                        "
*                         |_____________________________________  3
*       Step 4: Pre-Link IVPCCC3
*
// EXEC EDCPRLK,SIZE=EDCPRLK
/*
*       Step 5: Link-Edit IVPCCC3
*
// EXEC LNKEDT
/*
#&
$ $$ EOJ
* $$ END
/&
* $$ EOJ
```

Figure 11. Job Control Extract for EQAWIVH3.

1 Modify the VSE/POWER job control statements for your site.

2 Modify the job card as appropriate for your site.

3 Change *lib.userlib* to your temporary user sublibrary:

- The C/VSE translated source file is cataloged to this sublibrary in step 2.
- The phase IVPCCC3 created by the link-edit in step 3 is written to this sublibrary.

Notes:

Sublibrary *lib.userlib* must be available to CICS in:

- The SOURCE search chain. IVPCCC3.C member must be available to enable Debug Tool to display the source code in the SOURCE window. This member contains the translated source required by Debug Tool.
- The PHASE search chain. IVPCCC3.PHASE must be available to CICS to enable it to be executed.

Alternatively, IVPCCC3.C and IVPCCC3.PHASE members can be copied from *lib.userlib* into a sublibrary available to CICS.

4 If necessary, change the sublibrary to match the sublibrary where Debug Tool is are installed.

5 If necessary, change the sublibrary to match the sublibrary where C/VSE is installed.

6 If necessary, change the sublibrary to match the sublibrary where LE/VSE is installed.

After you modify the EQAWIVH3 job, submit it. Step 2 writes the translated source to member IVPCCC3.C in you user sublibrary. In the sample job this is *lib.userlib*. This member is used during the interactive debug session to display the SOURCE window.

EQAWIVH3 creates two jobs and returns two output listings. The first job is the CICS translation of the source for IVPCCC3. If successful, this job completes with return code zero. The second job catalogs the translated source member, compiles, pre-links, and link-edits IVPCCC3. If successful, the job finishes with return code 2, indicating unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

Step 2. Add your user sublibrary to the CICS search chain: Add *lib.userlib* to the LIBDEF SOURCE and LIBDEF PHASE job control statements in your CICS start up job control. If you do not want to add this sublibrary to your CICS system you can:

° Copy the listing member IVPCCC3.C from *lib.userlib* to a sublibrary that exists in your CICS LIBDEF SOURCE search chain.

° Copy the phase IVPCCC3.PHASE from *lib.userlib* to a sublibrary that exists in your CICS LIBDEF PHASE search chain.

Step 3. Define the installation verification program and transaction code to your CICS system: At your CICS terminal, logon to a user ID with authority to use the CEDA and CEMT transaction codes. Enter the commands:

° CEDA DEFINE PROG(IVPCCC3) LANG(C) GROUP(IVPDTCCC)

This defines the installation verification program for C/VSE.

° CEDA DEFINE TRAN(IVPH) PROG(IVPCCC3) GROUP(IVPDTCCC)

This defines the transaction code for the C/VSE installation verification program.

° CEDA INSTALL GROUP(IVPDTCCC)

This installs the installation verification group just created.

° CEMT SET PROG(IVPCCC3) NEW

This command might be required if CICS was already running and you copied the IVPCCC3 phase to an existing CICS sublibrary.

Step 4. Invoke the installation verification program and follow the script: Enter the transaction code IVPH to start the execution of the program IVPCCC3. An interactive debug session starts immediately on the same terminal as the IVPH transaction. The initial debug screen looks like the example shown in Figure 12.

Notes:

1. Debug Tool writes a profile settings file (*member-name*.DTSAFE) to the first sublibrary in the CICS LIBDEF SOURCE search chain. The DTSAFE member is used to save the Debug Tool session settings when your debug session terminates, these settings are used as your defaults for future debugging sessions.

   If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where Debug Tool writes the profile settings file.

   The *member-name* of the profile settings file depends on how you use your CICS system:

   - If you logged on to CICS with a user ID and password, then *member-name* is set to your user ID.

   - If you do not have to log on to use CICS applications then *member-name* is set to DTVSE.

2. The example screen is for a 24-line display and assumes the default settings are being used for Debug Tool. If your display has a different number of lines or is not using the default settings, the display you see will be slightly different to the example.

```
 ____________________________________________________________________________
|                                                                            |
| C        LOCATION: "DD:lib.userlib(IVPCCC3.C)" INITIALIZATION              |
| COMMAND ===>                                              SCROLL ===> PAGE |
| MONITOR --+----1----+----2----+----3----+----4----+----5----+----6 LINE: 0 OF 0 |
| ****************************** TOP OF MONITOR ******************************* |
| ***************************** BOTTOM OF MONITOR ******************************* |
|                                                                            |
|                                                                            |
|                                                                            |
| SOURCE: DD:lib.userlib(IVPCCC3.C) ----3----+----4----+----5----+ LINE: 1 OF 179 |
| ******************************* TOP OF SOURCE ******************************** |
|       1                                                                  .  |
|       2   #pragma runopts (test(all,*,prompt,*))                         .  |
|       3  #ifndef __dfheitab                                              .  |
|       4    #define __dfheitab 1                                          .  |
| LOG 0----+----1----+----2----+----3----+----4----+----5----+----6- LINE: 1 OF 3 |
| ******************************** TOP OF LOG ********************************* |
| 0001 DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0                      |
| 0002 10/19/96 1:07:08 PM                                                   |
| 0003 (C) COPYRIGHT IBM CORP. 1992, 1996                                    |
| ******************************** BOTTOM OF LOG ******************************* |
|                                                                            |
|                                                                            |
| PF  1:?          2:STEP       3:QUIT       4:LIST       5:FIND       6:AT/CLEAR |
| PF  7:UP         8:DOWN       9:GO        10:ZOOM      11:ZOOM LOG  12:RETRIEVE |
|                                                                            |
|____________________________________________________________________________|
```

Note: The date and time will vary from the sample shown.

Figure 12. Initial Debug Session Screen for IVPCCC3.

You are now ready to enter commands in your CICS debug session. The script following leads you through the installation verification. At the command line in your interactive debug session enter the commands shown following the ===> and press the ENTER key.

```
===> set log on file lib.userlib(IVPCCC3.LOG)
```

Note: Replace *lib.userlib* with your user library or specify a different log file definition if required.

The LOG window shows:

```
0004  SET LOG ON FILE lib.userlib(IVPCCC3.LOG) ;
0005 DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0
0006 10/19/96 1:07:08 PM
0007 (C) COPYRIGHT IBM CORP. 1992, 1996
```

Note: The date and time will vary from the sample shown.

```
===> at 137 list("Break point 1",countx)
===> at 141 list("Break point 2",countx)
```

Note: After entering the "at 137" command, you can use PF12 to recall the command. You can then modify it to look like the "at 141" command.

The SOURCE window remains unchanged, and the LOG window shows:

```
0008  AT 137
0009    LIST ( "Break point 1", countx ) ;
0010  AT 141
0011    LIST ( "Break point 2", countx ) ;
```

```
===> go
```

The screen is cleared and the message IVPCCC3 STARTED is displayed at the top left hand corner of the screen.

Press the ENTER key

The Debug Tool window is re-displayed. The SOURCE window moves to statement 137, and the LOG window shows:

```
0012  GO ;
0013 Break point 1
0014 countx = 0
```

```
===> go
```

The SOURCE window moves to statement 141, and the LOG window shows:

```
0015  GO ;
0016 Break point 2
0017 countx = 1
```

```
===> go
```

The SOURCE window loops back to statement 141, and the LOG window shows:

```
      0018  GO ;
      0019 Break point 2
      0020 countx = 2



===> go
```

The screen is cleared and the message IVPCCC3 MESSAGE AT COUNTX = 3 is displayed at the top left hand corner of the screen.

Press the ENTER key

The Debug Tool window is re-displayed. The SOURCE window loops back to statement 141, and the LOG window shows:

```
      0021  GO ;
      0022 Break point 2
      0023 countx = 3



===> clear at 141
===> go
```

The screen is cleared and the message IVPCCC3 COMPLETED is displayed at the top left hand corner of the screen.

Press the ENTER key

The Debug Tool window is re-displayed. The SOURCE window moves to statement 176, and the LOG window shows:

```
      0024  CLEAR AT 141 ;
      0025  GO ;
      0026  YOU WERE PROMPTED BECAUSE THE CEE067  CONDITION WAS RAISED IN YOUR PROGRAM
      0027  CEE067 IS A SEVERITY OR CLASS 1 CONDITION.
      0028  THE OPERATING SYSTEM HAS GENERATED THE FOLLOWING MESSAGE:
      0029     CEE0199W THE TERMINATION OF A THREAD WAS SIGNALED DUE TO A STOP STATEME
      0030  THE CURRENT LOCATION IS IVPCCC3 ::> "DD:lib.userlib(IVPCCC3.C)" :> 176.



===> quit
```

The SOURCE window remains at statement 176, and the LOG window shows:

```
      0031   QUIT ;
```

and the MONITOR window is replaced with:

```
 ____________________________________________________________________________
|                                                                            |
| C        LOCATION: "DD:lib.userlib(IVPCCC3.C)" :> 176                      |
| COMMAND ===>                                                SCROLL ===> PAGE  |
| ************************************************************************** |
|                                                                            |
| DO YOU REALLY WANT TO TERMINATE THIS SESSION? N                            |
| ENTER     Y FOR YES   AND   N FOR NO                                       |
|                                                                            |
| ************************************************************************** |
|                                                                            |
```

Overtype the N with Y and press the ENTER key. Your CICS debug session now terminates and the terminal returns to CICS with your transaction code IVPH in the top left hand corner.

Your CICS debug session creates a log in member IVPCCC3.LOG of sublibrary *lib.userlib* that contains the results of your CICS debug session. List this log and check it against Figure 13. Except for the date and time in line two, and *lib.userlib* in the second last line, these logs should match, verifying that Debug Tool installed successfully.

```
/* DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0                    */
/* 10/19/96 1:07:08 PM                                                 */
/* (C) COPYRIGHT IBM CORP. 1992, 1996                                  */
 AT 137
   LIST ( "Break point 1", countx ) ;
 AT 141
   LIST ( "Break point 2", countx ) ;
 GO ;
/* Break point 1                                                       */
/* countx = 0                                                          */
 GO ;
/* Break point 2                                                       */
/* countx = 1                                                          */
 GO ;
/* Break point 2                                                       */
/* countx = 2                                                          */
 GO ;
/* Break point 2                                                       */
/* countx = 3                                                          */
 CLEAR AT 141 ;
 GO ;
/* YOU WERE PROMPTED BECAUSE THE CEE067  CONDITION WAS RAISED IN YOUR  */
/* PROGRAM.                                                            */
/* CEE067 IS A SEVERITY OR CLASS 1 CONDITION.                          */
/* THE OPERATING SYSTEM HAS GENERATED THE FOLLOWING MESSAGE:           */
/*    CEE0199W THE TERMINATION OF A THREAD WAS SIGNALED DUE TO A STOP  */
/* STATEMENT.                                                          */
/* THE CURRENT LOCATION IS IVPCCC3 ::> "DD:lib.userlib(IVPCCC3.C)" :> 176. */
/*  QUIT ;                                                             */
```

Figure 13. Log File Content after Execution of IVPCCC3.

3.2 Installation Verification with COBOL/VSE

This section describes the installation verification for Debug Tool with COBOL/VSE.

Subtopics:

- 3.2.1 Batch Debug Using a Commands File
- 3.2.2 Batch Execution with Interactive Debug
- 3.2.3 Interactive Debug with CICS

3.2.1 Batch Debug Using a Commands File

The sample job EQAWIVC1 compiles, link-edits, and runs a COBOL/VSE program, IVPCOB1, that invokes Debug Tool. Modify the sample job control to meet your requirements before submitting the job. Figure 14 shows an extract from EQAWIVC1; tailoring requirements for each statement are discussed in the notes.

```
* $$ JOB JNM=EQAWIVC1,CLASS=W    _____________________________  1
// JOB EQAWIVC1    ____________________________________________  2
*                   ___________________________________________  3
*                  |            ______________________________  4
*
// LIBDEF *,SEARCH=(lib.userlib,PRD2.PROD,PRD2.SCEEBASE)
// LIBDEF PHASE,CATALOG=lib.userlib       "
// OPTION CATAL          "                |_____________________  5
*                        |_____________________________________  3
*       Step 1: Compile IVPCOB1
*
// EXEC IGYCRCTL,SIZE=IGYCRCTL
 CBL QUOTE TEST NAME
  .
  .
  .
COBOL/VSE program source
  .
  .
  .
/*
*       Step 2: Link-Edit IVPCOB1
*
// EXEC LNKEDT
/*
*       Step 3: Run the IVP program invoking Debug Tool
*                                  _____________________________  6
*
// EXEC IVPCOB1,SIZE=AUTO,PARM='/TEST(,SYSIPT,,SYSIPT)'
  .
  .
  .
Debug Tool preferences file
  .
  .
  .
/*
  .
  .
  .
Debug Tool commands file
  .
  .
  .
/*
/&
* $$ EOJ
```

Figure 14. Job Control Extract for EQAWIVC1.

1 Modify the VSE/POWER job control statements for your site.

2 Modify the job card as appropriate for your site.

3 Change *lib.userlib* to your temporary user sublibrary:

- The phase IVPCOB1 created by the link-edit in step 2 is written to this sublibrary.
- Debug Tool writes a profile settings file (*member-name*.DTSAFE) to the first sublibrary in the LIBDEF SOURCE search chain. The DTSAFE member is used to save the Debug Tool session settings when your debug session terminates; these settings are used as your defaults for future debugging sessions.

  If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where Debug Tool writes the profile settings file.

  The *member-name* of the profile settings file depends on your VSE/ESA system options:

  - If your VSE/ESA system IPL activates security checking, see Appendix A, "Determining the Default Userid" in topic A.0 for a description of how *member-name* is determined.
  - If your VSE/ESA system does not use security checking then *member-name* is set to DTVSE.

4 If necessary, change the sublibrary to match the sublibrary where Debug Tool and COBOL/VSE are installed (Debug Tool and COBOL/VSE might not be installed in the same sublibrary).

5 If necessary, change the sublibrary to match the sublibrary where LE/VSE is installed.

6 Any messages produced by EQAWIVC1 will be in your LE/VSE installation default national language.

If you want messages to be produced in a national language other than your LE/VSE installation default, add the NATLANG run-time option by specifying NATLANG(*xxx*) after the slash (/) at the start of the options string, where *xxx* is the three character representation of the language you wish to use (such as JPN, ENU, or UEN).

After you modify the EQAWIVC1 job, submit it. If successful, the job finishes with return code 2, indicating unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

EQAWIVC1 creates a log on SYSLST that contains the results from the execution of the debug control statements specified in the SYSIPT preferences file and the SYSIPT commands file. Check this log against Figure 15. Except for:

- The date and time
- The address for STR1
- The *lib.userlib*

these logs should match, verifying that Debug Tool installed successfully.

```
* DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0
* 10/19/96 11:48:11 AM
* (C) COPYRIGHT IBM CORP. 1992, 1996
  SET MSGID ON ;
  AT 86
    LIST ( "At the breakpoint for line", %LINE ) ;
  GO ;
* EQA1140I At the breakpoint for line
* EQA1141I %LINE = 86.1
  LIST STR1 ;
* EQA1141I STR1 = 'ONE  '
  STEP 6 ;
  LIST STR1 ;
* EQA1141I STR1 = 'TOP  '
  DESCRIBE ATTRIBUTES STR1 ;
* EQA1102I   ATTRIBUTES FOR STR1
* EQA1105I     ITS LENGTH IS 5
* EQA1103I     ITS ADDRESS IS 00521DE8
* EQA1112I     77 IVPCOB1:>STR1   X(5) DISP
  STEP 1 ;
  77 TEMP PIC X(5) ;
  MOVE STR1 TO TEMP ;
  LIST TEMP ;
* EQA1141I TEMP = 'BOT  '
  GO ;
* EQA1140I At the breakpoint for line
* EQA1141I %LINE = 86.1
  CLEAR AT 86 ;
  AT EXIT *
    LIST ( "Exiting ", %CU ) ;
  GO ;
* EQA1140I Exiting
* EQA1141I %CU = IVPCOB1
  SET MSGID OFF ;
*  QUIT ;
```

Figure 15. SYSLST Log Content after Execution of IVPCOB1.

3.2.2 Batch Execution with Interactive Debug

The sample job EQAWIVC2 compiles, link-edits, and runs a COBOL/VSE program, IVPCOB2, that invokes Debug Tool interactively. Modify the sample job control to meet your requirements before submitting the job. Figure 16 shows an extract from EQAWIVC2; tailoring requirements for each statement are discussed in the notes.

```
* $$ JOB JNM=EQAWIVC2,CLASS=W    _____________________________  1
// JOB EQAWIVC2    ____________________________________________  2
*                  ____________________________________________  3
```

```
*                  |           _____________________________  4
*
// LIBDEF *,SEARCH=(lib.userlib,PRD2.PROD,PRD2.SCEEBASE)
// LIBDEF PHASE,CATALOG=lib.userlib       "
// OPTION CATAL          "                 |_____________________  5
*                        |_____________________________________  3
*       Step 1: Compile IVPCOB2
*
// EXEC IGYCRCTL,SIZE=IGYCRCTL,PARM='EXIT(PRTEXIT(EQALIST))'
 CBL QUOTE TEST NAME
  .
  .
  .
COBOL/VSE program source
  .
  .
  .
/*
*       Step 2: Link-Edit IVPCOB2
*
// EXEC LNKEDT
/*
*       Step 3: Run the IVP program invoking Debug Tool
*               for Interactive Debug
*                                  _____________________________  6
*                                 |                 ____________  7
*
// EXEC IVPCOB2,SIZE=AUTO,PARM='/TEST(,SYSIPT,,MFI%vtamluid:*)'
  .
  .
  .
Debug Tool commands file
  .
  .
  .
/*
/&
* $$ EOJ
```

Figure 16. Job Control Extract for EQAWIVC2.

1 Modify the VSE/POWER job control statements for your site.

2 Modify the job card as appropriate for your site.

3 Change *lib.userlib* to your temporary user sublibrary:

- The compiler listing exit EQALIST, supplied with Debug Tool, writes the compiler listing to the first sublibrary in the LIBDEF SOURCE search chain. If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where the compiler listing exit writes the compiler listing file.

- The phase IVPCOB2 created by the link-edit in step 2 is written to this sublibrary.

- Debug Tool writes a profile settings file (*member-name*.DTSAFE) to the first sublibrary in the LIBDEF SOURCE search chain. The DTSAFE member is used to save the Debug Tool session settings when your debug session terminates, these settings are used as your defaults for future debugging sessions.

  If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where Debug Tool writes the profile settings file.

The *member-name* of the profile settings file depends on your VSE/ESA system options:

- If your VSE/ESA system IPL activates security checking, see Appendix A, "Determining the Default Userid" in topic A.0 for a description of how *member-name* is determined.

- If your VSE/ESA system does not use security checking then *member-name* is set to DTVSE.

4 If necessary, change the sublibrary to match the sublibrary where Debug Tool and COBOL/VSE are installed (Debug Tool and COBOL/VSE may not be installed in the same sublibrary).

5 If necessary, change the sublibrary to match the sublibrary where LE/VSE is installed.

6 Any messages produced by EQAWIVC2 will be in your LE/VSE installation default national language.

If you want messages to be produced in a national language other than your LE/VSE installation default, add the NATLANG run-time option by specifying NATLANG(*xxx*) after the slash (/) at the start of the options string, where *xxx* is the three character representation of the language you wish to use (such as JPN, ENU, or UEN).

7 Change the *vtamluid* in the TEST option to the VTAM LU name of the terminal where your interactive debug session will run. For example if your VTAM LU name is D0800001, change the TEST option to TEST(,SYSIPT,,MFI%D0800001:*).

Note: The VTAM LU must not be in session with another application when Debug Tool attempts to acquire it.

After you modify the EQAWIVC2 job, submit it. If successful, the job finishes with return code 2, indicating unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem. Step 1 uses the print exit EQALIST supplied with Debug Tool to write a blank-compressed copy of the compiler listing to member IVPCOB2.LIST in the first sublibrary of the LIBDEF SOURCE search chain (see **3** in the notes for Figure 16). In the sample job this is *lib.userlib*. This member is used during the interactive debug session to display the SOURCE window.

When the execution of the program IVPCOB2 starts, following successful completion of the link-edit step, an interactive debug session starts on the terminal you designated in the TEST option at **7** . The Debug Tool commands specified in the SYSIPT commands file are then executed before you enter commands interactively. After the commands in the commands file have executed, the debug screen looks like the example shown in Figure 17.

Note: The example screen is for a 32-line display and assumes the default settings are being used for Debug Tool. If your display has a different number of lines or is not using the default settings, the display you see will be slightly different to the example.

```
| COBOL    LOCATION: IVPCOB2 :> 92.1                                          |
| COMMAND ===>                                              SCROLL ===> PAGE  |
| MONITOR --+----1----+----2----+----3----+----4----+----5----+----6 LINE: 1 OF 1 |
| ****************************** TOP OF MONITOR ******************************* |
| 0001  1 VARBL1   11                                                         |
| ***************************** BOTTOM OF MONITOR ****************************** |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
| SOURCE: IVPCOB2 --1----+----2----+----3----+----4----+----5----+ LINE: 87 OF 96 |
|       87              MOVE "BEG" TO STR2                                  .  |
|       88              MOVE "UP" TO STR3                                   .  |
|       89              ADD 1 TO VARBL1                                     .  |
|       90              SUBTRACT 2 FROM VARBL2                              .  |
|       91              ADD 1 TO R                                          .  |
|       92              MOVE "BOT" TO STR1                                  .  |
|       93              MOVE "END" TO STR2 MOVE "DOW" TO STR3               .  |
|       94            END-PERFORM.                                          .  |
|       95            MOVE "DONE" TO STR1. MOVE "END" TO STR2. MOVE "FIN" TO ST .  |
| LOG 0----+----1----+----2----+----3----+----4----+----5----+----6 LINE: 9 OF 14 |
| 0009 %LINE = 86.1                                                           |
| 0010  MONITOR                                                               |
| 0011    LIST VARBL1 ;                                                       |
| 0012  LIST STR1 ;                                                           |
| 0013 STR1 = 'ONE  '                                                         |
| 0014  STEP 6 ;                                                              |
| PF  1:?         2:STEP       3:QUIT       4:LIST       5:FIND       6:AT/CLEAR  |
| PF  7:UP        8:DOWN       9:GO        10:ZOOM      11:ZOOM LOG  12:RETRIEVE  |
|                                                                             |
|_____________________________________________________________________________|
```

Figure 17. Debug Session Screen for IVPCOB2 after Commands File Execution.

You are now ready to enter commands in your interactive debug session. The script following leads you through the installation verification. At the command line in your interactive debug session, enter the commands shown following the ===> and press the ENTER key.

The initial SOURCE window is pointing at statement 92 and the value of VARBL1 in the MONITOR window is 11.

```
===> list str1
```

The SOURCE window remains at statement 92, the MONITOR window remains unchanged, and the LOG window shows:

```
      0015  LIST STR1 ;
      0016 STR1 = 'TOP  '
```

```
===> describe attributes str1
```

The SOURCE window remains at statement 92, the MONITOR window remains unchanged, and the LOG window shows:

```
      0017  DESCRIBE ATTRIBUTES STR1 ;
      0018    ATTRIBUTES FOR STR1
      0019      ITS LENGTH IS 5
      0020      ITS ADDRESS IS 00521DE8
      0021      77 IVPCOB2:>STR1   X(5) DISP
```

Note: The address you see on line 0020 might vary from the sample shown.

```
===> step 1
```

The SOURCE window moves to statement 93, the MONITOR window remains unchanged, and the LOG window shows:

```
      0022  STEP 1 ;
```

```
===> 77 temp pic x(5)
===> move str1 to temp
===> list temp
```

The SOURCE window remains at statement 93, the MONITOR window remains unchanged, and the LOG window shows:

```
      0023  77 TEMP PIC X(5) ;
      0024  MOVE STR1 TO TEMP ;
      0025  LIST TEMP ;
      0026 TEMP = 'BOT  '
```

```
===> go
```

The SOURCE window moves to statement 86, the MONITOR window remains unchanged, and the LOG window shows:

```
      0027  GO ;
      0028 At the breakpoint for line
      0029 %LINE = 86.1
```

```
===> clear at 86
===> at exit * list ("Exiting ",%cu)
===> go
```

The SOURCE window moves to statement 1, the value of VARBL1 in the MONITOR window changes to 18, and the LOG window shows:

```
      0030  CLEAR AT 86 ;
      0031  AT EXIT *
      0032    LIST ( "Exiting ", %CU ) ;
      0033  GO ;
      0034 Exiting
      0035 %CU = IVPCOB2
```

```
===> quit
```

The SOURCE window remains at statement 1, and the LOG window shows:

```
      0036  QUIT ;
```

and the MONITOR window is replaced with:

```
 _____________________________________________________________________________
|                                                                             |
| COBOL    LOCATION: IVPCOB2 EXIT                                             |
| COMMAND ===>                                                 SCROLL ===> PAGE  |
| ***************************************************************************  |
|                                                                             |
| DO YOU REALLY WANT TO TERMINATE THIS SESSION? N                             |
| ENTER     Y FOR YES   AND   N FOR NO                                        |
|                                                                             |
| ***************************************************************************  |
|                                                                             |
```

Overtype the N with Y and press the ENTER key. Your interactive debug session now terminates and the terminal returns to your default VTAM display.

If successful, the job finishes with return code 2 indicating there were unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

Your interactive debug session creates a log on SYSLST that contains the results of your interactive debug session. Check this log against Figure 18. Except for:

- The date and time
- The address for STR1
- The *lib.userlib*

these logs should match, verifying that Debug Tool installed successfully.

```
* DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0
* 10/19/96 11:30:29 AM
* (C) COPYRIGHT IBM CORP. 1992, 1996
  SET PACE 1 ;
  AT 86
    LIST ( "At the breakpoint for line", %LINE ) ;
  GO ;
* At the breakpoint for line
* %LINE = 86.1
  MONITOR
    LIST VARBL1 ;
  LIST STR1 ;
* STR1 = 'ONE  '
  STEP 6 ;
  LIST STR1 ;
* STR1 = 'TOP  '
  DESCRIBE ATTRIBUTES STR1 ;
*   ATTRIBUTES FOR STR1
*     ITS LENGTH IS 5
*     ITS ADDRESS IS 00521DE8
*     77 IVPCOB2:>STR1   X(5) DISP
  STEP 1 ;
  77 TEMP PIC X(5) ;
  MOVE STR1 TO TEMP ;
  LIST TEMP ;
* TEMP = 'BOT  '
  GO ;
* At the breakpoint for line
* %LINE = 86.1
  CLEAR AT 86 ;
  AT EXIT *
```

```
            LIST ( "Exiting ", %CU ) ;
          GO ;
        * Exiting
        * %CU = IVPCOB2
        *  QUIT ;
```

Figure 18. SYSLST Log Content after Execution of IVPCOB2.

3.2.3 Interactive Debug with CICS

Table 14 lists the steps that must be taken to complete the installation verification for Debug Tool with COBOL/VSE in a CICS environment.

Table 14. Summary of steps for installation verification of COBOL/VSE with CICS.

Step	Description
1	Assemble the LE/VSE default options module CEEUOPT
2	Compile and link-edit the CICS installation verification program
3	Add your user sublibrary to the CICS search chain
4	Define the installation verification program and transaction code to your CICS system
5	Invoke the installation verification program and follow the script

Step 1. Assemble the LE/VSE default options module CEEUOPT: The sample job EQAWIVC3 assembles and catalogs the LE/VSE default options module CEEUOPT. This default options module must be link-edited into your CICS installation verification phase. The TEST option in the CEEXOPT macro sets the Debug Tool options for your CICS program. Modify the sample job control to meet your requirements before submitting the job. Figure 19 shows an extract from EQAWIVC3; tailoring requirements for each statement are discussed in the notes.

```
* $$ JOB JNM=EQAWIVC3,CLASS=W    _____________________________  1
* $$ PUN DISP=I,PRI=6,CLASS=W
// JOB EQAWIVC3    ____________________________________________  2
*                   ___________________________________________  4
*
// LIBDEF *,SEARCH=(PRD2.SCEEBASE)
// ASSGN SYSIPT,SYSRDR
// EXEC IESINSRT
// JOB EQAWIVC3
*
*       Step 2: Catalog module CEEUOPT.OBJ
*
// EXEC LIBR,PARM='ACCESS SUBLIB=lib.userlib'
* $$ END                          "
```

```
*                               |_____________________________  3
*       Step 1: Assemble the LE/VSE default options
*
// OPTION DECK
// EXEC ASMA90,SIZE=ASMA90
   PUNCH ' CATALOG CEEUOPT.OBJ,REPLACE=YES'
   PRINT ON,NOGEN
CEEUOPT  CSECT
CEEUOPT  AMODE ANY
CEEUOPT  RMODE ANY
       CEEXOPT TEST=(ALL,*,PROMPT,*)
       END
/*
// ASSGN SYSIPT,SYSRDR
// EXEC IESINSRT
/*
#&
$ $$ EOJ
* $$ END
/&
* $$ EOJ
```

Figure 19. Job Control Extract for EQAWIVC3.

1 Modify the VSE/POWER job control statements for your site.

2 Modify the job card as appropriate for your site.

3 Change *lib.userlib* to your temporary user sublibrary:

This is where the options module CEEUOPT will be cataloged.

4 If necessary, change the sublibrary to match the sublibrary where LE/VSE is installed.

EQAWIVC3 creates two jobs and returns two output listings. The first job is the assembly of the default options module CEEUOPT; if successful, this job completes with return code zero. The second job catalogs the object module CEEUOPT; if successful, this job completes with return code zero.

Step 2. Compile and link-edit the CICS installation verification program: The sample job EQAWIVC4 compiles and link-edits a COBOL/VSE program, IVPCOB3, that invokes Debug Tool when the program is started under CICS. Modify the sample job control to meet your requirements before submitting the job. Figure 20 shows an extract from EQAWIVC4; tailoring requirements for each statement are discussed in the notes.

```
* $$ JOB JNM=EQAWIVC4,CLASS=W    _____________________________  1
* $$ PUN DISP=I,PRI=6,CLASS=W
// JOB EQAWIVC4    ____________________________________________  2
// ASSGN SYSIPT,SYSRDR
// EXEC IESINSRT     ___________________________________________  3
// JOB EQAWIVC4     |            ____________________________  4
*
// LIBDEF *,SEARCH=(lib.userlib,PRD2.PROD,PRD2.SCEEBASE)
// LIBDEF PHASE,CATALOG=lib.userlib       "
// OPTION CATAL         "                |_____________________  5
```

```
*                      |_____________________________________  3
*       Step 2: Compile IVPCOB3
*
// OPTION CATAL
   PHASE IVPCOB3,*
   INCLUDE DFHELII
   INCLUDE CEEUOPT
// EXEC IGYCRCTL,SIZE=IGYCRCTL,PARM='EXIT(PRTEXIT(EQALIST))'
* $$ END
*       Step 1: Translate IVPCOB3 for CICS
*
// LIBDEF *,SEARCH=(PRD2.PROD)
// OPTION DECK      "
// EXEC DFHECP1$    |_____________________________________  4
 CBL XOPTS(COBOL2) APOST TEST(ALL)
  .
  .
  .
COBOL/VSE program source
  .
  .
  .
/*
// EXEC IESINSRT
/*
*       Step 3: Link-Edit IVPCOB3
*
// EXEC LNKEDT
#&
$ $$ EOJ
* $$ END
/&
* $$ EOJ
```

Figure 20. Job Control Extract for EQAWIVC4.

1 Modify the VSE/POWER job control statements for your site.

2 Modify the job card as appropriate for your site.

3 Change *lib.userlib* to your temporary user sublibrary:

- The compiler listing exit EQALIST, supplied with Debug Tool, writes the compiler listing to the first sublibrary in the LIBDEF SOURCE search chain. If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where the compiler listing exit writes the compiler listing file.

- The phase IVPCOB3 created by the link-edit in step 3 is written to this sublibrary.

Notes:

Sublibrary *lib.userlib* must be available to CICS in:

- The SOURCE search chain. IVPCOB3.LIST member must be available to enable Debug Tool to display the source code in the SOURCE window.

- The PHASE search chain. IVPCOB3.PHASE must be available to CICS to enable it to be executed.

Alternatively, IVPCOB3.LIST and IVPCOB3.PHASE members can be copied from *lib.userlib* into a sublibrary available to CICS.

4 If necessary, change the sublibrary to match the sublibrary where Debug Tool and COBOL/VSE are installed (Debug Tool and COBOL/VSE may not be installed in the same sublibrary).

5 If necessary, change the sublibrary to match the sublibrary where LE/VSE is installed.

After you modify the EQAWIVC4 job, submit it. Step 2 uses the print exit EQALIST supplied with Debug Tool to write a blank-compressed copy of the compiler listing to member IVPCOB3.LIST in the first sublibrary of the LIBDEF SOURCE search chain (see **3** in the notes for Figure 20). In the sample job this is *lib.userlib*. This member is used during the interactive debug session to display the SOURCE window.

EQAWIVC4 creates two jobs and returns two output listings. The first job is the CICS translation of the source in IVPCOB3. If successful, this job completes with return code zero. The second job compiles and link-edits IVPCOB3. If successful, the job finishes with return code 2, indicating unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

Step 3. Add your user sublibrary to the CICS search chain: Add *lib.userlib* to the LIBDEF SOURCE and LIBDEF PHASE job control statements in your CICS start up job control. If you do not want to add this sublibrary to your CICS system you can:

° Copy the listing member IVPCOB3.LIST from *lib.userlib* to a sublibrary that exists in your CICS LIBDEF SOURCE search chain.

° Copy the phase IVPCOB3.PHASE from *lib.userlib* to a sublibrary that exists in your CICS LIBDEF PHASE search chain.

Step 4. Define the installation verification program and transaction code to your CICS system: At your CICS terminal, logon to a user ID with authority to use the CEDA and CEMT transaction codes. Enter the commands:

° CEDA DEFINE PROG(IVPCOB3) LANG(COBOL) GROUP(IVPDTCOB)

This defines the installation verification program for COBOL/VSE.

° CEDA DEFINE TRANS(IVPC) PROGRAM(IVPCOB3) GROUP(IVPDTCOB)

This defines the transaction code for the COBOL/VSE installation verification program.

° CEDA INSTALL GROUP(IVPDTCOB)

This installs the installation verification program group just created.

- CEMT SET PROG(IVPCOB3) NEW

  This command might be required if CICS was already running and you copied the IVPCOB3 phase to an existing CICS sublibrary.

Step 5. Invoke the installation verification program and follow the script: Enter the transaction code IVPC to start the execution of the program IVPCOB3. An interactive debug session starts immediately on the same terminal as the IVPC transaction. The initial debug screen looks like the example shown in Figure 21.

Notes:

1. Debug Tool writes a profile settings file (*member-name*.DTSAFE) to the first sublibrary in the CICS LIBDEF SOURCE search chain. The DTSAFE member is used to save the Debug Tool session settings when your debug session terminates, these settings are used as your defaults for future debugging sessions.

   If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where Debug Tool writes the profile settings file.

   The *member-name* of the profile settings file depends on how you use your CICS system:

   - If you logged on to CICS with a user ID and password, then *member-name* is set to your user ID.
   - If you do not have to log on to use CICS applications then *member-name* is set to DTVSE.

2. The example screen is for a 24-line display and assumes the default settings are being used for Debug Tool. If your display has a different number of lines or is not using the default settings, the display you see will be slightly different to the example.

```
 ____________________________________________________________________________
|                                                                            |
| COBOL    LOCATION: IVPCOB3 INITIALIZATION                                  |
| COMMAND ===>                                          SCROLL ===> PAGE     |
| MONITOR --+----1----+----2----+----3----+----4----+----5----+----6 LINE: 0 OF 0  |
| ****************************** TOP OF MONITOR ******************************  |
| ***************************** BOTTOM OF MONITOR ******************************  |
|                                                                            |
|                                                                            |
|                                                                            |
| SOURCE: IVPCOB3 --1----+----2----+----3----+----4----+----5----+ LINE: 1 OF 201  |
| ******************************* TOP OF SOURCE ********************************  |
|        1        IDENTIFICATION DIVISION.                                .  |
|        2        PROGRAM-ID. IVPCOB3.                                    .  |
|        3        ENVIRONMENT DIVISION.                                   .  |
|        4        DATA DIVISION.                                          .  |
| LOG 0----+----1----+----2----+----3----+----4----+----5----+----6- LINE: 1 OF 3  |
| ******************************** TOP OF LOG *********************************  |
| 0001 DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0                      |
| 0002 10/19/96 11:14:20 AM                                                  |
| 0003 (C) COPYRIGHT IBM CORP. 1992, 1996                                    |
```

```
| ****************************** BOTTOM OF LOG ******************************  |
|                                                                               |
|                                                                               |
| PF  1:?          2:STEP       3:QUIT       4:LIST       5:FIND       6:AT/CLEAR  |
| PF  7:UP         8:DOWN       9:GO        10:ZOOM      11:ZOOM LOG  12:RETRIEVE  |
|                                                                               |
|_______________________________________________________________________________|
```

Note: The date and time will vary from the sample shown.

Figure 21. Initial Debug Session Screen for IVPCOB3.

You are now ready to enter commands in your CICS debug session. The script following leads you through the installation verification. At the command line in your interactive debug session enter the commands shown following the ===> and press the ENTER key.

```
===> set log on file lib.userlib(IVPCOB3.LOG)
```

Note: Replace *lib.userlib* with your user library or specify a different log file definition if required.

The LOG window shows:

```
0004  SET LOG ON FILE lib.userlib(IVPCOB3.LOG) ;
0005 DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0
0006 10/19/96 11:14:20 AM
0007 (C) COPYRIGHT IBM CORP. 1992, 1996
```

Note: The date and time will vary from the sample shown.

```
===> at 168 list("Break point 1",countx)
===> at 170 list("Break point 2",countx)
```

Note: After entering the "at 168" command, you can use PF12 to recall the command. You can then modify it to look like the "at 170" command.

The SOURCE window remains unchanged, and the LOG window shows:

```
0008  AT 168
0009    LIST ( "Break point 1", COUNTX ) ;
0010  AT 170
0011    LIST ( "Break point 2", COUNTX ) ;
```

```
===> go
```

The screen is cleared and the message IVPCOB3 STARTED is displayed at the top left hand corner of the screen.

Press the ENTER key

The Debug Tool window is re-displayed. The SOURCE window moves to statement 168, and the LOG window shows:

```
0012  GO ;
0013 Break point 1
```

```
      0014 COUNTX = 01

===> go
```

The SOURCE window moves to statement 170, and the LOG window shows:

```
      0015  GO ;
      0016 Break point 2
      0017 COUNTX = 01



===> go
```

The SOURCE window loops back to statement 170, and the LOG window shows:

```
      0018  GO ;
      0019 Break point 2
      0020 COUNTX = 02



===> go
```

The screen is cleared and the message IVPCOB3 MESSAGE AT COUNTX = 3 is displayed at the top left hand corner of the screen.

Press the ENTER key

The Debug Tool window is re-displayed. The SOURCE window loops back to statement 170, and the LOG window shows:

```
      0021  GO ;
      0022 Break point 2
      0023 COUNTX = 03



===> clear at 170
===> go
```

The screen is cleared and the message IVPCOB3 COMPLETED is displayed at the top left hand corner of the screen.

Press the ENTER key

The Debug Tool window is re-displayed. The SOURCE window moves to statement 199, and the LOG window shows:

```
      0024  CLEAR AT 170 ;
      0025  GO ;
      0026  YOU WERE PROMPTED BECAUSE THE CEE067  CONDITION WAS RAISED IN YOUR PROGRAM
      0027  CEE067 IS A SEVERITY OR CLASS 1 CONDITION.
      0028  THE OPERATING SYSTEM HAS GENERATED THE FOLLOWING MESSAGE:
      0029     CEE0199W THE TERMINATION OF A THREAD WAS SIGNALED DUE TO A STOP STATEME
      0030  THE CURRENT LOCATION IS IVPCOB3 ::> IVPCOB3 :> 199.1.



===> quit
```

The SOURCE window remains at statement 199, and the LOG window shows:

```
      0031   QUIT ;
```

and the MONITOR window is replaced with:

```
 ____________________________________________________________________________
|                                                                            |
| COBOL    LOCATION: IVPCOB3 :> 199.1                                        |
| COMMAND ===>                                                 SCROLL ===> PAGE  |
| ************************************************************************** |
|                                                                            |
| DO YOU REALLY WANT TO TERMINATE THIS SESSION? N                            |
| ENTER     Y FOR YES   AND   N FOR NO                                       |
|                                                                            |
| ************************************************************************** |
|                                                                            |
```

Overtype the N with Y and press the ENTER key. Your CICS debug session now terminates and the terminal returns to CICS with your transaction code IVPC in the top left hand corner.

Your CICS debug session creates a log in member IVPCOB3.LOG of sublibrary *lib.userlib* that contains the results of your CICS debug session. List this log and check it against Figure 22. Except for the date and time in line two, these logs should match, verifying that Debug Tool installed successfully.

```
* DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0
* 10/19/96 11:14:20 AM
* (C) COPYRIGHT IBM CORP. 1992, 1996
  AT 168
    LIST ( "Break point 1", COUNTX ) ;
  AT 170
    LIST ( "Break point 2", COUNTX ) ;
  GO ;
* Break point 1
* COUNTX = 01
  GO ;
* Break point 2
* COUNTX = 01
  GO ;
* Break point 2
* COUNTX = 02
  GO ;
* Break point 2
* COUNTX = 03
  CLEAR AT 170 ;
  GO ;
* YOU WERE PROMPTED BECAUSE THE CEE067  CONDITION WAS RAISED IN YOUR
* PROGRAM.
* CEE067 IS A SEVERITY OR CLASS 1 CONDITION.
* THE OPERATING SYSTEM HAS GENERATED THE FOLLOWING MESSAGE:
*    CEE0199W THE TERMINATION OF A THREAD WAS SIGNALED DUE TO A STOP
* STATEMENT.
* THE CURRENT LOCATION IS IVPCOB3 ::> IVPCOB3 :> 199.1.
*  QUIT ;
```

Figure 22. Log File Content after Execution of IVPCOB3.

3.3 Installation Verification with PL/I VSE

This section describes the installation verification for Debug Tool with PL/I VSE.

Subtopics:

- 3.3.1 Batch Debug Using a Commands File
- 3.3.2 Batch Execution with Interactive Debug
- 3.3.3 Interactive Debug with CICS

3.3.1 Batch Debug Using a Commands File

The sample job EQAWIVP1 compiles, link-edits, and runs a PL/I VSE program, IVPPLI1, that invokes Debug Tool. Modify the sample job control to meet your requirements before submitting the job. Figure 23 shows an extract from EQAWIVP1; tailoring requirements for each statement are discussed in the notes.

```
* $$ JOB JNM=EQAWIVP1,CLASS=W    _____________________________  1
// JOB EQAWIVP1    ____________________________________________  2
*                    ___________________________________________  3
*                   |            ____________________________  4
*
// LIBDEF *,SEARCH=(lib.userlib,PRD2.PROD,PRD2.SCEEBASE)
// LIBDEF PHASE,CATALOG=lib.userlib       "
// OPTION CATAL          "                |_____________________  5
*                        |_____________________________________  3
*       Step 1: Compile IVPPLI1
*
// EXEC IEL1AA,SIZE=IEL1AA
*PROCESS TEST(ALL) NAME('IVPPLI1') INCLUDE;
  .
  .
  .
PL/I VSE program source
  .
  .
  .
/*
*       Step 2: Link-Edit IVPPLI1
*
// EXEC LNKEDT
/*
*       Step 3: Run the IVP program invoking Debug Tool
*                                                  _________  6
*
// EXEC IVPPLI1,SIZE=AUTO,PARM='TEST(,SYSIPT,,SYSIPT)/'
  .
  .
  .
Debug Tool preferences file
  .
  .
  .
/*
  .
  .
  .
```

```
Debug Tool commands file
   .
   .
   .
/*
/&
* $$ EOJ
```

Figure 23. Job Control Extract for EQAWIVP1.

1 Modify the VSE/POWER job control statements for your site.

2 Modify the job card as appropriate for your site.

3 Change *lib.userlib* to your temporary user sublibrary:

- The phase IVPPLI1 created by the link-edit in step 2 is written to this sublibrary.

- Debug Tool writes a profile settings file (*member-name*.DTSAFE) to the first sublibrary in the LIBDEF SOURCE search chain. The DTSAFE member is used to save the Debug Tool session settings when your debug session terminates; these settings are used as your defaults for future debugging sessions.

  If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where Debug Tool writes the profile settings file.

  The *member-name* of the profile settings file depends on your VSE/ESA system options:

  - If your VSE/ESA system IPL activates security checking, see Appendix A, "Determining the Default Userid" in topic A.0 for a description of how *member-name* is determined.

  - If your VSE/ESA system does not use security checking then *member-name* is set to DTVSE.

4 If necessary, change the sublibrary to match the sublibrary where Debug Tool and PL/I VSE are installed (Debug Tool and PL/I VSE may not be installed in the same sublibrary).

5 If necessary, change the sublibrary to match the sublibrary where LE/VSE is installed.

6 Any messages produced by EQAWIVP1 will be in your LE/VSE installation default national language.

If you want messages to be produced in a national language other than your LE/VSE installation default, add the NATLANG run-time option by specifying NATLANG(*xxx*) before the slash (/) at the end of the options string, where *xxx* is the three character representation of the language you wish to use (such as JPN, ENU, or UEN).

After you modify the EQAWIVP1 job, submit it. If successful, the job finishes with return code 2, indicating unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

EQAWIVP1 creates a log on SYSLST that contains the results from the execution of the debug control statements specified in the SYSIPT preferences file and the SYSIPT commands file. Check this log against Figure 24. Except for:

- The date and time
- The address for MYSTRING
- The *lib.userlib*

these logs should match, verifying that Debug Tool installed successfully.

```
/* DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0                                */
/* 10/19/96 1:02:56 PM                                                             */
/* (C) COPYRIGHT IBM CORP. 1992, 1996                                              */
 SET MSGID ON ;
 LIST ( 'IVPPLI1 started' ) ;
/* EQA1140I 'IVPPLI1 started'                                                      */
 AT 9
   LIST ( 'At the breakpoint for line ', %LINE ) ;
 GO ;
/* EQA1140I 'At the breakpoint for line '                                          */
/* EQA1141I %LINE = '9.1'                                                          */
 STEP 10 ;
 LIST ( 'MyString equal to:  ', MYSTRING ) ;
/* EQA1140I 'MyString equal to:  '                                                 */
/* EQA1141I MYSTRING = 'ABC          '                                             */
 STEP 1 ;
 LIST ( 'MyString equal to:  ', MYSTRING ) ;
/* EQA1140I 'MyString equal to:  '                                                 */
/* EQA1141I MYSTRING = 'ABCD         '                                             */
 STEP 5 ;
 DESCRIBE ATTRIBUTES MYSTRING ;
/* EQA1342I   ATTRIBUTES FOR MYSTRING CHARACTER(13) AUTOMATIC                      */
/* EQA1337I     ITS ADDRESS IS  005C1335 AND ITS LENGTH IS 13                      */
 DECLARE MYTEMP CHARACTER ( 13 ) ;
 MYTEMP = MYSTRING ;
 LIST MYTEMP ;
/* EQA1141I MYTEMP = 'ABCDE        '                                               */
 GO ;
/* EQA1306I YOU WERE PROMPTED BECAUSE THE FINISH  CONDITION WAS RAISED IN YOUR PROGRAM. */
/* EQA1309I FINISH IS A SEVERITY OR CLASS 1 CONDITION.                             */
/* EQA1238I THE CURRENT LOCATION IS IVPPLI1 :> 13.1.                               */
 LIST ( 'Exiting ', %CU ) ;
/* EQA1140I 'Exiting '                                                             */
/* EQA1141I %CU = 'IVPPLI1'                                                        */
 GO ;
 LIST ( 'IVPPLI1 Ended' ) ;
/* EQA1140I 'IVPPLI1 Ended'                                                        */
 SET MSGID OFF ;
/*  QUIT ;                                                                         */
```

Figure 24. SYSLST Log Content after Execution of IVPPLI1.

3.3.2 Batch Execution with Interactive Debug

The sample job EQAWIVP2 compiles, link-edits, and runs a PL/I VSE program, IVPPLI2, that invokes Debug Tool interactively. Modify the sample job control to meet your requirements before submitting the job. Figure 25 shows an extract from EQAWIVP2; tailoring requirements for each statement are discussed in the notes.

```
* $$ JOB JNM=EQAWIVP2,CLASS=W    _____________________________  1
// JOB EQAWIVP2    ____________________________________________  2
*                   ___________________________________________  3
*                  |            ________________________________  4
*
// LIBDEF *,SEARCH=(lib.userlib,PRD2.PROD,PRD2.SCEEBASE)
// LIBDEF PHASE,CATALOG=lib.userlib       "
// OPTION CATAL          "                |____________________  5
*                        |_____________________________________  3
*       Step 1: Compile IVPPLI2
*
// EXEC IEL1AA,SIZE=IEL1AA,PARM='EXIT(PRTEXIT(EQALIST))'
*PROCESS TEST(ALL) NAME('IVPPLI2') INCLUDE;
  .
  .
  .
PL/I VSE program source
  .
  .
  .
/*
*       Step 2: Link-Edit IVPPLI2
*
// EXEC LNKEDT
/*
*       Step 3: Run the IVP program invoking Debug Tool
*               for Interactive Debug                ___________  6
*                                                   |          _  7
*
// EXEC IVPPLI2,SIZE=AUTO,PARM='TEST(,SYSIPT,,MFI%vtamluid:*)/'
  .
  .
  .
Debug Tool commands file
  .
  .
  .
/*
/&
* $$ EOJ
```

Figure 25. Job Control Extract for EQAWIVP2.

1 Modify the VSE/POWER job control statements for your site.

2 Modify the job card as appropriate for your site.

3 Change *lib.userlib* to your temporary user sublibrary:

- The compiler listing exit EQALIST, supplied with Debug Tool, writes the compiler listing to the first sublibrary in the LIBDEF SOURCE search chain. If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where the compiler listing exit writes the compiler listing file.

- The phase IVPPLI2 created by the link-edit in step 2 is written to this sublibrary.

- Debug Tool writes a profile settings file (*member-name*.DTSAFE) to the first sublibrary in the LIBDEF SOURCE search chain. The DTSAFE member is used to save the Debug Tool session settings when your debug session terminates, these settings are used as your defaults for future debugging sessions.

  If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where Debug Tool writes the profile settings file.

  The *member-name* of the profile settings file depends on your VSE/ESA system options:

  - If your VSE/ESA system IPL activates security checking, see Appendix A, "Determining the Default Userid" in topic A.0 for a description of how *member-name* is determined.

  - If your VSE/ESA system does not use security checking then *member-name* is set to DTVSE.

4 If necessary, change the sublibrary to match the sublibrary where Debug Tool and PL/I VSE are installed (Debug Tool and COBOL/VSE may not be installed in the same sublibrary).

5 If necessary, change the sublibrary to match the sublibrary where LE/VSE is installed.

6 Change the *vtamluid* in the TEST option to the VTAM LU name of the terminal where your interactive debug session will run. For example if your VTAM LU name is D0800001, change the TEST option to TEST(,SYSIPT,,MFI%D0800001:*).

Note: The VTAM LU must not be in session with another application when Debug Tool attempts to acquire it.

7 Any messages produced by EQAWIVP2 will be in your LE/VSE installation default national language.

If you want messages to be produced in a national language other than your LE/VSE installation default, add the NATLANG run-time option by specifying NATLANG(*xxx*) before the slash (/) at the end of the options string, where *xxx* is the three character representation of the language you wish to use (such as JPN, ENU, or UEN).

After you modify the EQAWIVP2 job, submit it. If successful, the job finishes with return code 2, indicating unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem. Step 1 uses the print exit EQALIST supplied with Debug Tool to write a blank-compressed copy of the compiler listing to

member IVPPLI2.LIST in the first sublibrary of the LIBDEF SOURCE search chain (see **3** in the notes for Figure 25). In the sample job this is *lib.userlib*. This member is used during the interactive debug session to display the SOURCE window.

When the execution of the program IVPPLI2 starts, following successful completion of the link-edit step, an interactive debug session starts on the terminal you designated in the TEST option at **6** . The Debug Tool commands specified in the SYSIPT commands file are then executed before you enter commands interactively. After the commands in the commands file have executed, the debug screen looks like the example shown in Figure 26.

Note: The example screen is for a 32-line display and assumes the default settings are being used for Debug Tool. If your display has a different number of lines or is not using the default settings, the display you see will be slightly different to the example.

```
PL/I     LOCATION: IVPPLI2 :> 10.1
COMMAND ===>                                                  SCROLL ===> PAGE
MONITOR --+----1----+----2----+----3----+----4----+----5----+----6 LINE: 1 OF 1
****************************** TOP OF MONITOR *******************************
0001  1 COUNTER       5
***************************** BOTTOM OF MONITOR *****************************
SOURCE: IVPPLI2 --1----+----2----+----3----+----4----+----5----+ LINE: 63 OF 67
      10       Substr(MyString,Counter,1) = Substr(B_cont,Counter,1);
      11       Substr(B_cont,Counter,1) = Substr(Fills,Counter,1);
      12     End;
      13   End IVPPLI2;
***************************** BOTTOM OF SOURCE ******************************
LOG 0----+----1----+----2----+----3----+----4----+----5----+---- LINE: 12 OF 17
0012  MONITOR
0013    LIST COUNTER ;
0014  STEP 13 ;
0015  LIST ( 'MyString equal to:  ', MYSTRING ) ;
0016 'MyString equal to:  '
0017 MYSTRING = 'ABCD         '
PF  1:?          2:STEP       3:QUIT       4:LIST       5:FIND       6:AT/CLEAR
PF  7:UP         8:DOWN       9:GO        10:ZOOM      11:ZOOM LOG  12:RETRIEVE
```

Figure 26. Debug Session Screen for IVPPLI2 after Commands File Execution.

You are now ready to enter commands in your interactive debug session. The script following leads you through the installation verification. At the command line in your interactive debug session, enter the commands shown following the ===> and press the ENTER key.

The initial SOURCE window is pointing at statement 10 and the value of COUNTER in the MONITOR window is 5.

```
===> step 1
```

The SOURCE window moves to statement 11, the MONITOR window remains unchanged, and the LOG window shows:

```
      0018  STEP 1 ;
```

```
===> list ('MyString is equal to ',mystring)
===> step 5
```

The SOURCE window moves to statement 10, the value of COUNTER in the MONITOR window changes to 7, and the LOG window shows:

```
      0019  LIST ( 'MyString is equal to ',MYSTRING ) ;
      0020  'MyString is equal to '
      0021 MYSTRING = 'ABCDE        '
      0022  STEP 5 ;
```

```
===> describe attributes mystring
===> declare mytemp character (13)
===> mytemp=mystring
===> list mytemp
```

The SOURCE window remains at statement 10, the MONITOR window remains unchanged, and the LOG window shows:

```
      0023  DESCRIBE ATTRIBUTES MYSTRING ;
      0024    ATTRIBUTES FOR MYSTRING CHARACTER (13) AUTOMATIC
      0025      ITS ADDRESS IS 005C1335 AND ITS LENGTH IS 13
      0026  DECLARE MYTEMP CHARACTER ( 13 ) ;
      0027  MYTEMP = MYSTRING ;
      0028  LIST MYTEMP ;
      0029 MYTEMP = 'ABCDEF       '
```

Note: The address you see on line 0025 might vary from the sample shown.

```
===> go
```

The SOURCE window moves to statement 13, the value of COUNTER in the MONITOR window changes to 14, and the LOG window shows:

```
      0030  GO ;
      0031 YOU WERE PROMPTED BECAUSE THE FINISH  CONDITION WAS RAISED IN YOUR PROGRAM
      0032 FINISH IS A SEVERITY OR CLASS 1 CONDITION.
      0033 THE CURRENT LOCATION IS IVPPLI2 :> 13.1.
```

```
===> monitor 1
```

The SOURCE window remains at statement 13, in the MONITOR window the MONITOR 1 LIST command is replaced with a null command, and the LOG window shows:

```
      0034  MONITOR 1 ;
      0035 THE MONITOR 1 COMMAND IS REPLACED.
```

```
===> list ('Exiting ',%cu)
```

The SOURCE window remains at statement 13, the MONITOR window remains unchanged, and the LOG window shows:

```
      0036  LIST ( 'Exiting ', %CU ) ;
      0037 'Exiting '
      0038 %CU = 'IVPPLI2'
```

```
===> go
===> list ('IVPPLI2 Ended')
```

The SOURCE window clears, the MONITOR window remains unchanged, and the LOG window shows:

```
      0039  GO ;
      0040  LIST ( 'IVPPLI2 Ended' ) ;
      0041 'IVPPLI2 Ended'
```

```
===> quit
```

The LOG window shows:

```
      0042  QUIT ;
```

and the MONITOR window is replaced with:

```
 ____________________________________________________________________________
|                                                                            |
| PL/I     LOCATION: UNKNOWN (APPLICATION PROGRAM HAS TERMINATED)            |
| COMMAND ===>                                              SCROLL ===> PAGE |
| ************************************************************************** |
|                                                                            |
| DO YOU REALLY WANT TO TERMINATE THIS SESSION? N                            |
| ENTER     Y FOR YES   AND   N FOR NO                                       |
|                                                                            |
| ************************************************************************** |
|                                                                            |
```

Overtype the N with Y and press the ENTER key. Your interactive debug session now terminates and the terminal returns to your default VTAM display.

If successful, the job finishes with return code 2 indicating there were unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

Your interactive debug session creates a log on SYSLST that contains the results of your interactive debug session. Check this log against Figure 27. Except for:

- The date and time
- The address for MYSTRING
- The *lib.userlib*

these logs should match, verifying that Debug Tool installed successfully.

```
/* DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0                                */
/* 10/19/96 1:39:33 PM                                                          */
/* (C) COPYRIGHT IBM CORP. 1992, 1996                                           */
 SET PACE 1 ;
 LIST ( 'IVPPLI2 Started' ) ;
/* 'IVPPLI2 Started'                                                            */
 AT 9
   LIST ( 'At the breakpoint for line ', %LINE ) ;
 GO ;
/* 'At the breakpoint for line '                                                */
/* %LINE = '9.1'                                                                */
 MONITOR
   LIST COUNTER ;
 STEP 13 ;
 LIST ( 'MyString equal to:  ', MYSTRING ) ;
/* 'MyString equal to:  '                                                       */
/* MYSTRING = 'ABCD         '                                                   */
 STEP 1 ;
 LIST ( 'MyString is equal to ', MYSTRING ) ;
/* 'MyString is equal to '                                                      */
/* MYSTRING = 'ABCDE        '                                                   */
 STEP 5 ;
 DESCRIBE ATTRIBUTES MYSTRING ;
/*   ATTRIBUTES FOR MYSTRING CHARACTER(13) AUTOMATIC                            */
/*     ITS ADDRESS IS  005C1335 AND ITS LENGTH IS 13                            */
 DECLARE MYTEMP CHARACTER ( 13 ) ;
 MYTEMP = MYSTRING ;
 LIST MYTEMP ;
/* MYTEMP = 'ABCDEF       '                                                     */
 GO ;
/* YOU WERE PROMPTED BECAUSE THE FINISH  CONDITION WAS RAISED IN YOUR PROGRAM.  */
/* FINISH IS A SEVERITY OR CLASS 1 CONDITION.                                   */
/* THE CURRENT LOCATION IS IVPPLI2 :> 13.1.                                     */
 MONITOR 1 ;
/* THE MONITOR 1 COMMAND IS REPLACED.                                           */
 LIST ( 'Exiting ', %CU ) ;
/* 'Exiting '                                                                   */
/* %CU = 'IVPPLI2'                                                              */
 GO ;
 LIST ( 'IVPPLI2 Ended' ) ;
/* 'IVPPLI2 Ended'                                                              */
/*  QUIT ;                                                                      */
```

Figure 27. SYSLST Log Content after Execution of IVPPLI2.

3.3.3 Interactive Debug with CICS

Table 15 lists the steps that must be taken to complete the installation verification for Debug Tool with PL/I VSE in a CICS environment.

Table 15. Summary of steps for CICS IVP for PL/I VSE.

Step	Description
__ 1	Compile and link-edit the CICS installation verification

	program
__ 2	Add your user sublibrary to the CICS search chain
__ 3	Define the installation verification program and transaction code to your CICS system
__ 4	Invoke the installation verification program and follow the script

Step 1. Compile and link-edit the CICS installation verification program: The sample job EQAWIVP3 compiles and link-edits a PL/I VSE program, IVPPLI3, that invokes Debug Tool when the program is started under CICS. Modify the sample job control to meet your requirements before submitting the job. Figure 28 shows an extract from EQAWIVP3; tailoring requirements for each statement are discussed in the notes.

```
* $$ JOB JNM=EQAWIVP3,CLASS=W    _____________________________  1
* $$ PUN DISP=I,PRI=6,CLASS=W
// JOB EQAWIVP3    _____________________________________________  2
// ASSGN SYSIPT,SYSRDR
// EXEC IESINSRT     ____________________________________________  3
// JOB EQAWIVP3     |            ____________________________  4
*
// LIBDEF *,SEARCH=(lib.userlib,PRD2.PROD,PRD2.SCEEBASE)
// LIBDEF PHASE,CATALOG=lib.userlib       "
*                       "                |_____________________  5
*                       |_____________________________________  3
*       Step 2: Compile IVPPLI3
*
// OPTION CATAL
   PHASE IVPPLI3,*
   INCLUDE DFHELII
// EXEC IEL1AA,SIZE=IEL1AA,PARM='EXIT(PRTEXIT(EQALIST))'
* $$ END
*       Step 1: Translate IVPPLI3 for CICS
*                    ____________________________________________  4
*
// LIBDEF *,SEARCH=(PRD2.PROD)
// OPTION DECK
// EXEC DFHEPP1$,PARM='CICS'
*PROCESS TEST(ALL) INCLUDE;
  .
  .
  .
PL/I VSE program source
  .
  .
  .
/*
// EXEC IESINSRT
/*
*       Step 3: Link-Edit IVPPLI3
*
// EXEC LNKEDT
#&
$ $$ EOJ
* $$ END
/&
* $$ EOJ
```

Figure 28. Job Control Extract for EQAWIVP3.

1 Modify the VSE/POWER job control statements for your site.

2 Modify the job card as appropriate for your site.

3 Change *lib.userlib* to your temporary user sublibrary:

- The compiler listing exit EQALIST, supplied with Debug Tool, writes the compiler listing to the first sublibrary in the LIBDEF SOURCE search chain. If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where the compiler listing exit writes the compiler listing file.

- The phase IVPPLI3 created by the link-edit in step 3 is written to this sublibrary.

Notes:

Sublibrary *lib.userlib* must be available to CICS in:

- The SOURCE search chain. IVPPLI3.LIST member must be available to enable Debug Tool to display the source code in the SOURCE window.

- The PHASE search chain. IVPPLI3.PHASE must be available to CICS to enable it to be executed.

Alternatively, IVPPLI3.LIST and IVPPLI3.PHASE members can be copied from *lib.userlib* into a sublibrary available to CICS.

4 If necessary, change the sublibrary to match the sublibrary where Debug Tool and PL/I VSE are installed (Debug Tool and PL/I VSE may not be installed in the same sublibrary).

5 If necessary, change the sublibrary to match the sublibrary where LE/VSE is installed.

After you modify the EQAWIVP3 job, submit it. Step 2 uses the print exit EQALIST supplied with Debug Tool to write a blank-compressed copy of the compiler listing to member IVPPLI3.LIST in the first sublibrary of the LIBDEF SOURCE search chain (see **3** in the notes for Figure 28). In the sample job this is *lib.userlib*. This member is used during the interactive debug session to display the SOURCE window.

EQAWIVP3 creates two jobs and returns two output listings. The first job is the CICS translation of the source in IVPPLI3. If successful, this job completes with return code zero. The second job compiles and link-edits IVPPLI3. If successful, the job finishes with return code 2, indicating unresolved weak external references during the link-edit step. This return code is normal and does not indicate a problem.

Step 2. Add your user sublibrary to the CICS search chain: Add *lib.userlib* to the LIBDEF SOURCE and LIBDEF PHASE job control statements in your CICS start up job control. If you do not want to add this

sublibrary to your CICS system you can:

- Copy the listing member IVPPLI3.LIST from *lib.userlib* to a sublibrary that exists in your CICS LIBDEF SOURCE search chain.

- Copy the phase IVPPLI3.PHASE from *lib.userlib* to a sublibrary that exists in your CICS LIBDEF PHASE search chain.

Step 3. Define the installation verification program and transaction code to your CICS system: At your CICS terminal, logon to a user ID with authority to use the CEDA and CEMT transaction codes. Enter the commands:

- CEDA DEFINE PROG(IVPPLI3) LANG(PLI) GROUP(IVPDTPLI)

  This defines the installation verification program for PL/I VSE.

- CEDA DEFINE TRAN(IVPP) PROG(IVPPLI3) GROUP(IVPDTPLI)

  This defines the transaction code for the PL/I VSE installation verification program.

- CEDA INSTALL GROUP(IVPDTPLI)

  This installs the new IVP group just created.

- CEMT SET PROG(IVPPLI3) NEW

  This command might be required if CICS was already running and you copied the IVPPLI3 phase to an existing CICS sublibrary.

Step 4. Invoke the installation verification program and follow the script: Enter the transaction code IVPP to start the execution of the program IVPPLI3. An interactive debug session starts immediately on the same terminal as the IVPP transaction. The initial debug screen looks like the example shown in Figure 29.

Notes:

1. Debug Tool writes a profile settings file (*member-name*.DTSAFE) to the first sublibrary in the CICS LIBDEF SOURCE search chain. The DTSAFE member is used to save the Debug Tool session settings when your debug session terminates, these settings are used as your defaults for future debugging sessions.

   If you put a sublibrary other than *lib.userlib* as the first sublibrary in the SOURCE search chain, then this is where Debug Tool writes the profile settings file.

   The *member-name* of the profile settings file depends on how you use your CICS system:

   - If you logged on to CICS with a user ID and password, then *member-name* is set to your user ID.

- If you do not have to log on to use CICS applications then *member-name* is set to DTVSE.

2. The example screen is for a 24-line display and assumes the default settings are being used for Debug Tool. If your display has a different number of lines or is not using the default settings, the display you see will be slightly different to the example.

```
|
| PL/I     LOCATION: IVPPLI3 INITIALIZATION                                  |
| COMMAND ===>                                              SCROLL ===> PAGE |
| MONITOR --+----1----+----2----+----3----+----4----+----5----+----6 LINE: 0 OF 0 |
| ****************************** TOP OF MONITOR ******************************* |
| ****************************** BOTTOM OF MONITOR **************************** |
|                                                                              |
|                                                                              |
|                                                                              |
| SOURCE: IVPPLI3 --1----+----2----+----3----+----4----+----5----+ LINE: 1 OF 176 |
| ******************************** TOP OF SOURCE ******************************** |
|       1   IVPPLI3: PROC (DFHEIPTR)OPTIONS(MAIN REENTRANT);                  .  |
|       2   DCL 1 DFHCNSTS  STATIC,  /* CONSTANTS USED BY TRANSLATOR */       .  |
|                 2  DFHLDVER CHAR(22) INIT('LD TABLE DFHEITAB 230.'),        .  |
|                 2  DFHEIB0 FIXED BIN(15) INIT(0),                          .  |
| LOG 0----+----1----+----2----+----3----+----4----+----5----+----6- LINE: 1 OF 3 |
| ********************************* TOP OF LOG ********************************* |
| 0001 DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0                        |
| 0002 10/19/96 4:09:56 PM                                                     |
| 0003 (C) COPYRIGHT IBM CORP. 1992, 1996                                      |
| ******************************** BOTTOM OF LOG ******************************* |
|                                                                              |
|                                                                              |
| PF  1:?          2:STEP       3:QUIT       4:LIST       5:FIND       6:AT/CLEAR |
| PF  7:UP         8:DOWN       9:GO        10:ZOOM      11:ZOOM LOG  12:RETRIEVE |
|                                                                              |
|
```

Note: The date and time will vary from the sample shown.

Figure 29. Initial Debug Session Screen for IVPPLI3.

You are now ready to enter commands in your CICS debug session. The script following leads you through the installation verification. At the command line in your interactive debug session enter the commands shown following the ===> and press the ENTER key.

```
===> set log on file lib.userlib(IVPPLI3.LOG)
```

Note: Replace *lib.userlib* with your user library or specify a different log file definition if required.

The LOG window shows:

```
0004  SET LOG ON FILE lib.userlib(IVPPLI3.LOG) ;
0005 DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0
0006 10/19/96 4:09:56 PM
0007 (C) COPYRIGHT IBM CORP. 1992, 1996
```

Note: The date and time will vary from the sample shown.

```
===> at 27 list('Break point 1',countx)
===> at 28 list('Break point 2',countx)
```

Note: After entering the "at 27" command, you can use PF12 to recall the command. You can then modify it to look like the "at 28" command.

The SOURCE window remains unchanged, and the LOG window shows:

```
      0008  AT 27
      0009    LIST ( 'Break point 1', COUNTX ) ;
      0010  AT 28
      0011    LIST ( 'Break point 2', COUNTX ) ;
```

```
===> go
```

The screen is cleared and the message IVPPLI3 STARTED is displayed at the top left hand corner of the screen.

Press the ENTER key

The Debug Tool window is re-displayed. The SOURCE window moves to statement 27, and the LOG window shows:

```
      0012  GO ;
      0013 'Break point 1'
      0014 COUNTX =         1
```

```
===> go
```

The SOURCE window moves to statement 28, and the LOG window shows:

```
      0015  GO ;
      0016 'Break point 2'
      0017 COUNTX =         1
```

```
===> go
```

The SOURCE window loops back to statement 28, and the LOG window shows:

```
      0018  GO ;
      0019 'Break point 2'
      0020 COUNTX =         2
```

```
===> go
```

The screen is cleared and the message IVPPLI3 MESSAGE AT COUNTX = 3 is displayed at the top left hand corner of the screen.

Press the ENTER key

The Debug Tool window is re-displayed. The SOURCE window loops back to statement 28, and the LOG window shows:

```
      0021  GO ;
```

```
      0022 'Break point 2'
      0023 COUNTX =         3



===> clear at 28
===> go
```

The screen is cleared and the message IVPPLI3 COMPLETED is displayed at the top left hand corner of the screen.

Press the ENTER key

The Debug Tool window is re-displayed. The SOURCE window moves to statement 50, and the LOG window shows:

```
      0024  CLEAR AT 28 ;
      0025  GO ;
      0026  YOU WERE PROMPTED BECAUSE THE CEE067  CONDITION WAS RAISED IN YOUR PROGRAM
      0027  CEE067 IS A SEVERITY OR CLASS 1 CONDITION.
      0028  THE OPERATING SYSTEM HAS GENERATED THE FOLLOWING MESSAGE:
      0029     CEE0199W THE TERMINATION OF A THREAD WAS SIGNALED DUE TO A STOP STATEME
      0030  THE CURRENT LOCATION IS IVPPLI3 ::> IVPPLI3 :> 50.1.



===> quit
```

The SOURCE window remains at statement 50, and the LOG window shows:

```
      0031   QUIT ;
```

and the MONITOR window is replaced with:

```
 ____________________________________________________________________________
|                                                                            |
| PL/I     LOCATION: IVPPLI3 :> 50.1                                         |
| COMMAND ===>                                                 SCROLL ===> PAGE  |
| ************************************************************************** |
|                                                                            |
| DO YOU REALLY WANT TO TERMINATE THIS SESSION? N                            |
| ENTER     Y FOR YES   AND   N FOR NO                                       |
|                                                                            |
| ************************************************************************** |
|                                                                            |
```

Over type the N with Y and press the ENTER key. Your CICS debug session now terminates and the terminal returns to CICS with your transaction code IVPP in the top left hand corner.

Your CICS debug session creates a log in member IVPPLI3.LOG of sublibrary *lib.userlib* that contains the results of your CICS debug session. List this log and check it against Figure 30. Except for the date and time in line two, these logs should match, verifying that Debug Tool installed successfully.

```
  /* DEBUG TOOL FOR VSE/ESA VERSION 1 RELEASE 1 MOD 0                              */
  /* 10/19/96 4:09:56 PM                                                  */
  /* (C) COPYRIGHT IBM CORP. 1992, 1996                                   */
   AT 27
     LIST ( 'Break point 1', COUNTX ) ;
```

```
 AT 28
   LIST ( 'Break point 2', COUNTX ) ;
 GO ;
/* 'Break point 1'                                                    */
/* COUNTX =         1                                                 */
 GO ;
/* 'Break point 2'                                                    */
/* COUNTX =         1                                                 */
 GO ;
/* 'Break point 2'                                                    */
/* COUNTX =         2                                                 */
 GO ;
/* 'Break point 2'                                                    */
/* COUNTX =         3                                                 */
 CLEAR AT 28 ;
 GO ;
/* YOU WERE PROMPTED BECAUSE THE CEE067  CONDITION WAS RAISED IN YOUR */
/* PROGRAM.                                                           */
/* CEE067 IS A SEVERITY OR CLASS 1 CONDITION.                         */
/* THE OPERATING SYSTEM HAS GENERATED THE FOLLOWING MESSAGE:          */
/*    CEE0199W THE TERMINATION OF A THREAD WAS SIGNALED DUE TO A STOP */
/* STATEMENT.                                                         */
/* THE CURRENT LOCATION IS IVPPLI3 ::> IVPPLI3 :> 50.1.               */
/*  QUIT ;                                                            */
```

Figure 30. Log File Content after Execution of IVPPLI3.

4.0 Chapter 4. Customizing Debug Tool

This chapter describes how to customize Debug Tool after installation.

Subtopics:

- 4.1 Loading Debug Tool into the Shared Virtual Area
- 4.2 Changing the National Language for Debug Tool

4.1 Loading Debug Tool into the Shared Virtual Area

Placing routines in the SVA reduces overall system storage requirements. Also, initialization and termination time is reduced for each application, since load time decreases.

All the phases listed in Table 17 in topic B.1 and Table 18 in topic B.2 can be included in the SVA. To include them:

1. Modify the SVA statement of the VSE IPL ASI (Automated System Initialization) procedure to allow space for the phases:
   - Increase the SDL parameter by the number of new phases being added to the SVA.
   - Increase the PSIZE parameter by the amount of storage required to contain the new phases being added to the 24-bit and 31-bit SVA.

2. Modify the VSE background (BG) ASI procedure to automatically load the selected phases into the SVA:

   - Modify the ALLOC statements for the partitions to ensure that the remaining SVA storage is large enough to contain the selected phases.

   - Modify the LIBDEF PHASE SEARCH job control statement preceding the SET SDL statement to include the name of the sublibrary containing the Debug Tool phases.

   - After the SET SDL statement, add the statement:

     ```
     LIST=$SVAEQA
     ```

     If not all phases are required to be loaded into the SVA, add the statement:

     ```
     phasename,SVA
     ```

     for each phase required to be loaded into the SVA.

3. Shut down and re-IPL your VSE system.

The supplied SVA loadlist member $SVAEQA contains all the Debug Tool phases (except the Japanese NLS phases) that are eligible for inclusion in the SVA.

The phases listed in Table 19 in topic B.3 are the Japanese NLS phases that can be loaded in the SVA, after the Japanese NLS component is installed. To include Japanese NLS phases in the SVA, follow steps 1 to 3 above using the *phasename*,SVA method in step 2.

For more information on loading phases in the SVA, see *VSE/ESA System Control Statements*.

4.2 Changing the National Language for Debug Tool

Debug Tool executes in a LE/VSE environment, therefore the Debug Tool default national language is the same as the LE/VSE default for the national language. To change the LE/VSE default for the national language, see *LE/VSE Installation and Customization Guide*.

To change the national language used for individual executions of a program that invokes Debug Tool, use the PARM operand of the job control EXEC statement, as shown in Figure 31 and Figure 32.

```
                                                 _________  1

// EXEC  USRPROG,SIZE=USRPROG,PARM='NATLANG(xxx)/'
         "             "
         |_____________|_____________________________________  2
```

Figure 31. Example of the PARM option with NATLANG (C and PL/I)

```
                                                 _________  1

// EXEC  COBPROG,SIZE=COBPROG,PARM='/NATLANG(xxx)'
         "             "
         |_____________|_____________________________________  2
```

Figure 32. Example of the PARM option with NATLANG (COBOL)

1 Specify the national language required, where *xxx* may be:

ENU Mixed-case U.S. English
UEN Uppercase U.S. English
JPN Japanese

2 Change the name of the program to your application program name.

5.0 Chapter 5. Maintaining Debug Tool

This chapter describes how to replace or re-install Debug Tool, and how to apply service updates to Debug Tool. To effectively use the maintenance procedures, you must have already installed Debug Tool and any required products.

In addition, this chapter describes how to remove Debug Tool.

Subtopics:

- 5.1 Re-installing Debug Tool
- 5.2 Applying Service Updates
- 5.3 Removing Debug Tool
- 5.4 How to Report a Problem with Debug Tool

5.1 Re-installing Debug Tool

You do not need to remove Debug Tool from your system before re-installing Debug Tool. However, if you intend to re-install the product in a different sublibrary from the previous installation you must remove Debug Tool from the system history file before you can re-install it. If you are re-installing in the same sublibrary, you might need to delete Debug Tool from the sublibrary and release the space to ensure there is sufficient library space available for the re-installation.

If you do need to remove Debug Tool from your system, the section "Removing Debug Tool" in topic 5.3 describes how to do this.

To re-install Debug Tool, you follow the same steps as for installing Debug Tool. See Chapter 2, "Installing Debug Tool" in topic 2.0.

5.2 Applying Service Updates

You might need to apply maintenance or service updates to Debug Tool periodically. There are two types of formally supported software fixes. One is the program temporary fix (PTF) applied as corrective service or as preventive maintenance. The other is the authorized program analysis report (APAR) fix applied as a code replacement in a corrective maintenance mode.

Subtopics:

- 5.2.1 What You Receive
- 5.2.2 Checklist for Applying Service
- 5.2.3 Step 1. Check Prerequisite APARs or PTFs
- 5.2.4 Step 2. Backup Original System
- 5.2.5 Step 3. Apply Service
- 5.2.6 Step 4. Run the Installation Verification Programs (IVP)

5.2.1 What You Receive

If you report a problem with Debug Tool to your IBM Support Center, you will receive a tape containing one or more APARs or PTFs that have been created to solve your problem.

You might also receive a list of prerequisite APARs or PTFs that should have been applied to your system before applying the current service. These prerequisite APARs or PTFs might relate to Debug Tool or any other licensed product you have installed, including VSE/ESA.

You apply service to Debug Tool using either the VSE/ESA Interactive Interface or a batch job.

The following checklist provides a summary of steps you should use to apply service to Debug Tool.

5.2.2 Checklist for Applying Service

Table 16 lists the steps for installing corrective service on Debug Tool. You can use Table 16 as a checklist.

Table 16. Summary of steps for installing service on Debug Tool

Step	Description	Jobname	Topic
__ 1	Ensure prerequisite APARs or PTFs are applied	EQAWRETR	5.2.3
__ 2	Backup existing system	___	5.2.4
__ 3	Apply service	EQAWSERV	5.2.5
__ 4	Run the installation verification		2.5, 5.2.6

5.2.3 Step 1. Check Prerequisite APARs or PTFs

Prerequisite APARs or PTFs are APARs or PTFs that need to be applied to your system before you can apply the current maintenance. These APARs or PTFs might apply to Debug Tool or any licensed program you have installed at your installation.

Your IBM Support Center will have given you a list of any relevant prerequisite APARs or PTFs. Most probably they will already be applied to your system. You can verify this by retracing the APARs and PTFs in your system history file. The job shown in Figure 33 shows how to retrace APARs and PTFs in the system history file.

```
// JOB  EQAWRETR
*
*    Retrace APARs and PTFs
*
// EXEC  MSHP,SIZE=900K
RETRACE APARS
RETRACE PTFS
/*
/&
```

Figure 33. Retrace APARs and PTFs

Use the listing produced by this job to check whether you have applied the prerequisite APARs or PTFs. If you have not, your IBM Support Center will arrange to send them to you and you should apply them before applying other service.

5.2.4 Step 2. Backup Original System

Make a backup copy of your current Debug Tool library and the system history file. For information about backing up libraries and the system history file, see *VSE/ESA System Control Statements*.

5.2.5 Step 3. Apply Service

You can apply service to Debug Tool from the provided service tape using either the Interactive Interface or a batch job.

You will receive detailed instructions for applying service with the service tape.

Subtopics:

- 5.2.5.1 Method 1: Apply Service Using the Interactive Interface
- 5.2.5.2 Method 2: Apply Service Using a Batch Job

5.2.5.1 Method 1: Apply Service Using the Interactive Interface

The VSE/ESA Interactive Interface enables you to use dialog requests to apply service to Debug Tool. For more information about the functions of the Interactive Interface, see *VSE/ESA Administration*. For more information on how to use the dialogs to apply service, see *VSE/ESA System Upgrade and Service*.

5.2.5.2 Method 2: Apply Service Using a Batch Job

The batch job to apply service to Debug Tool uses the MSHP system history file where Debug Tool was installed.

A sample job to apply service using the Maintain System History Program (MSHP) is shown in Figure 34. For more information on MSHP see *VSE/ESA System Control Statements.*

```
// JOB  EQAWSERV
*
*    Apply Service
*
// ASSGN SYS006,cuu _____________________________  1
// EXEC  MSHP,SIZE=900K
INSTALL SERVICE FROMTAPE _____________________  2
/*
/&
```

Figure 34. Apply Service

1 Specify the tape address.

Change *cuu* to the address of the tape drive where you have mounted

the service tape.

2 This shows the MSHP statement to install service from a tape. The information in the system history file will direct MSHP to apply the service to the sublibrary in which Debug Tool is installed. You do not need to supply this information.

5.2.6 Step 4. Run the Installation Verification Programs (IVP)

After you have applied all the files on the service tape, run the installation verification exercises that correspond to the high-level languages you have installed at your site to ensure that Debug Tool functions properly. See "Step 4: Configure VTAM" in topic 2.5.

5.3 Removing Debug Tool

You do not have to remove Debug Tool from your system before installing a new version or release.

If you do have to remove Debug Tool from your system for any reason, you must delete all the Debug Tool entries from your sublibrary and remove Debug Tool from the system history file. Figure 36 shows the job control needed to remove Debug Tool from the system history file.

To delete all Debug Tool entries from your sublibrary, use the DELETE command of the LIBR program. The job shown in Figure 35 shows the job control needed to delete Debug Tool from the default sublibrary PRD2.PROD. You will need to tailor some statements to suit your installation. The tailoring requirements for each statement are discussed in the notes following Figure 35.

```
// JOB  EQAWLDEL
*
*    Delete Debug Tool
*
* Label for the Debug Tool library _____________________  1
*
// EXEC  LIBR,SIZE=200K
 ACCESS S=PRD2.PROD ___________________________________  2
 DELETE CEEEVDBG.PHASE
 DELETE EQA*.*
 DELETE HDA02*.Z
 DELETE $SVAEQA.*
 DELETE $SV$EQA.*
/*
/&
```

Figure 35. Delete Debug Tool from a Sublibrary

1 Specify the Debug Tool library.

If you have installed Debug Tool into a sublibrary other than the default (PRD2.PROD) then insert the required DLBL, EXTENT and ASSGN information for the Debug Tool library.

2 Specify the Debug Tool library

If you have installed Debug Tool in a sublibrary other than the default, change this statement.

To remove Debug Tool from the system history file, use the REMOVE command of the Maintain System History Program (MSHP). The sample job shown in Figure 36 shows the job control needed to remove Debug Tool from the system history file.

```
// JOB  EQAWREM
*
*    Remove the Debug Tool product
*
// EXEC  MSHP,SIZE=900K
REMOVE 5686-A02-00
REMOVE 5686-A02-01       ____________________  1
/*
/&
```

Figure 36. Remove Debug Tool from the System History File

1 Japanese national language feature

Remove this control statement if you did not install the Japanese national language feature.

5.4 How to Report a Problem with Debug Tool

For information on how to report a problem with Debug Tool, see your high-level language compiler Diagnosis Guide. Use the Debug Tool component IDs and CLCs in Table 2 in topic 1.1.1.

A.0 Appendix A. Determining the Default Userid

Unless otherwise specified, Debug Tool uses default names for preference and profile settings files, where the default is based on a userid. How the userid is determined depends upon whether you are running your debugging session in the batch environment or the CICS environment. Debug Tool determines the userid for a debugging session as described below.

Subtopics:

- A.1 Batch (non-CICS) Environment
- A.2 CICS Environment

A.1 Batch (non-CICS) Environment

In the batch environment, Debug Tool determines the userid as follows:

° For a VSE/ESA system IPLed with security checking active (SEC=YES specified in the IPL SYS command):

- If a userid is associated with the job, Debug Tool uses that userid. The userid associated with a job can be provided by the following sources:
  - The userid specified in the // ID job control statement
  - The userid specified in the SEC parameter of the VSE/POWER * $$ JOB statement
  - The logon userid, if the job was submitted by a user logged on to the VSE/ESA Interactive Interface
  - The logon userid, if the job was submitted by a user logged on to VSE/ICCF
  - The logon userid, if the job was submitted by a user logged on to a workstation with the SEND/RECEIVE command interface
- If a userid is not associated with the job, Debug Tool sets the userid to one of the following:
  - The name of the VTAM logical unit when debugging in full-screen mode session
  - The Debug Tool default value, DTVSE, when debugging in batch mode

° For a VSE/ESA system IPLed without security checking active (SEC=NO specified in the IPL SYS command), Debug Tool sets the userid to one of the following:

- The name of the VTAM logical unit when debugging in full-screen mode session
- The Debug Tool default value, DTVSE, when debugging in batch mode

---

A.2 CICS Environment

In the CICS environment, Debug Tool sets the userid to one of the following:

- The CICS logon userid, if available
- The Debug Tool default value, DTVSE, if the CICS logon userid is not available

---

B.0 Appendix B. SVA Space Requirements

This appendix lists the SVA space requirements for the Debug Tool phases.

Subtopics:

- B.1 Debug Tool Base Phases Loaded Above the 16-Megbyte Line
- B.2 Debug Tool Base Phases Loaded Below the 16-Megbyte Line
- B.3 Debug Tool Japanese Language Component Phases

B.1 Debug Tool Base Phases Loaded Above the 16-Megbyte Line

Table 17. 31-bit SVA Space Requirements for Debug Tool Base.

Phase	Space Needed for 31-bit SVA (bytes decimal)
CEEEVDBG	429504
EQADCCNM	21856
EQADCER0	1096
EQADCER1	1096
EQALISTC	7528
EQAMSGT	424
EQAXSIO	776
EQA00ER0	1720
EQA00ER1	1720
EQA00OSX	13872
EQA10ANL	1304
EQA10ER0	29736
EQA10ER1	29736
EQA10FND	4856
EQA10GL0	1560
EQA10GL1	1560
EQA10LX0	6544

EQA10LX1	6544
EQA10MG0	8896
EQA10MG1	8896
EQA10VC0	22864
EQA10VC1	22864
EQA10XSC	2072
EQA12CED	8128
EQA12CEI	1384
EQA12CES	4032
EQA12CEX	952
EQA12CUR	3552
EQA12DCL	5288
EQA12DSP	1496
EQA12LXB	18144
EQA12QNS	2240
EQA12SCN	3808
EQA12SYA	20152
EQA12SYC	4720
EQA12SYE	4376
EQA13CEX	3192
EQA13CUR	4496
EQA13DCL	7040
EQA13DSP	2088
EQA13LXB	22912
EQA13MDV	1856
EQA13QNS	2232
EQA13SCN	6176
EQA13SWC	4312
EQA13SYA	23032
EQA13SYC	4336
EQA13SYE	6224
EQA14CAL	3136
EQA14CEX	1880
EQA14CMS	1912
EQA14CUR	2408
EQA14DCL	2968
EQA14DSP	1360
EQA14EVL	4512

EQA14LXB	14848
EQA14MDV	2904
EQA14PRF	3536
EQA14QNS	2728
EQA14SCN	4360
EQA14SYA	19792
EQA14SYC	4640
EQA14SYE	3024
EQA30TKI	489176
EQA50CTL	168712
EQA50VIO	220
EQA50XIO	9096
Total	**1534404**

B.2 Debug Tool Base Phases Loaded Below the 16-Megbyte Line

Table 18. 24-bit SVA Space Requirements for Debug Tool Base.

Phase	Space Needed for 24-bit SVA (bytes decimal)
EQADCHU	8008
EQADCMU	1024
EQALIST	736
EQA00E23	2280
EQA10LN0	2632
EQA10LN1	2632
EQA10LTB	264
EQA10OSV	11408
EQA50MIO	3712
Total	**32696**

B.3 Debug Tool Japanese Language Component Phases

Table 19. SVA Space Requirements for Debug Tool Japanese Language Component.

Phase	Space Needed (bytes decimal)
EQA00ER2	1880
EQA10ER2	31288
EQA10GL2	1560
EQA10LN2(1)	2632
EQA10LX2	6592
EQA10MG2	10080
EQA10VC2	22864
EQADCER2	1096
Total	**77992**

Note:

1. This module is loaded in the 24-bit SVA

BIBLIOGRAPHY Bibliography

Subtopics:

- BIBLIOGRAPHY.1 Debug Tool Publications
- BIBLIOGRAPHY.2 Language Environment Publications
- BIBLIOGRAPHY.3 LE/VSE-Conforming Language Product Publications
- BIBLIOGRAPHY.4 Related Publications
- BIBLIOGRAPHY.5 Softcopy Publications

BIBLIOGRAPHY.1 Debug Tool Publications

Debug Tool for VSE/ESA

Fact Sheet, GC26-8925

User's Guide and Reference, SC26-8797

Installation and Customization Guide, SC26-8798

BIBLIOGRAPHY.2 Language Environment Publications

IBM Language Environment for VSE/ESA

Fact Sheet, GC33-6679

Concepts Guide, GC33-6680

Debugging Guide and Run-Time Messages, SC33-6681

Installation and Customization Guide, SC33-6682

Licensed Program Specifications, GC33-6683

Programming Guide, SC33-6684

Programming Reference, SC33-6685

Run-Time Migration Guide, SC33-6687

Writing Interlanguage Communication Applications, SC33-6686

C Run-Time Programming Guide, SC33-6688

C Run-Time Library Reference, SC33-6689

BIBLIOGRAPHY.3 LE/VSE-Conforming Language Product Publications

IBM C for VSE/ESA

Licensed Program Specifications, GC09-2421

Installation and Customization Guide, GC09-2422

Migration Guide, SC09-2423

User's Guide, SC09-2424

Language Reference, SC09-2425

Diagnosis Guide, GC09-2426

IBM COBOL for VSE/ESA

General Information, GC26-8068

Licensed Program Specifications, GC26-8069

Migration Guide, GC26-8070

Installation and Customization Guide, SC26-8071

Programming Guide, SC26-8072

Language Reference, SC26-8073

Diagnosis Guide, SC26-8528

Reference Summary, SX26-3834

IBM PL/I for VSE/ESA

Fact Sheet, GC26-8052

Programming Guide, SC26-8053

Language Reference, SC26-8054

Licensed Program Specifications, GC26-8055

Migration Guide, SC26-8056

Installation and Customization Guide, SC26-8057

Diagnosis Guide, SC26-8058

Compile-Time Messages and Codes, SC26-8059

Reference Summary, SX26-3836

BIBLIOGRAPHY.4 Related Publications

CICS/VSE

System Definition and Operations Guide, SC33-0706

Customization Guide, SC33-0707

Resource Definition (Macro), SC33-0709

Application Programming Guide, SC33-0712

Application Programming Reference, SC33-0713

Problem Determination Guide, SC33-0716

VSE/ESA Version 1 Release 4

Planning, SC33-6503

Administration, SC33-6505

Messages and Codes, SC33-6507

Guide to System Functions, SC33-6511

System Control Statements, SC33-6513

System Macros User's Guide, SC33-6515

System Macros Reference, SC33-6516

VSE/VSAM Commands and Macros, SC33-6532

VSE/VSAM User's Guide, SC33-6535

Release Information Guide, SC33-6536

VSE/ESA Version 2

Planning, SC33-6603

Administration, SC33-6605

Messages and Codes, SC33-6607

Guide to System Functions, SC33-6611

System Control Statements, SC33-6613

System Macros User's Guide, SC33-6615

System Macros Reference, SC33-6616

VSE/VSAM Commands and Macros, SC33-6532

VSE/VSAM User's Guide, SC33-6535

BIBLIOGRAPHY.5 Softcopy Publications

The following collection kit contains the LE/VSE and LE/VSE-conforming language product publications:

VSE Collection, SK2T-0060

You can order these publications from Mechanicsburg through your IBM representative.

INDEX Index

A

allocating Debug Tool library space, 2.3
APARs, 5.2.3
applying service updates
  backup original system, 5.2.4
  retracing APARs and PTFs, 5.2.3
  system history file, 5.2.3
  using a batch job, 5.2.5.2
  using the interactive interface, 5.2.5.1

B

batch interactive verification
  C/VSE, 3.1.2
  COBOL/VSE, 3.2.2
  PL/I VSE, 3.3.2
batch verification
  C/VSE, 3.1.1
  COBOL/VSE, 3.2.1
  PL/I VSE, 3.3.1
BookManager/Read, minimum release, 1.1.2.1

C

C for VSE/ESA, minimum release, 1.1.2.1
CICS
  C/VSE verification, 3.1.3
  COBOL/VSE verification, 3.2.3
  PL/I VSE verification, 3.3.3
  planning, 1.1.5
  running in 370 mode, 1.1.7.4
CICS/VSE, minimum release, 1.1.2.1
CLC (component level code), 1.1.1
COBOL for VSE/ESA, minimum release, 1.1.2.1
component identifier, 1.1.1
component level code (CLC), 1.1.1
components, 1.1.1.1
CSP/AD, minimum release, 1.1.2.1
CSP/AE, minimum release, 1.1.2.1
customizing Debug Tool, 4.0

D

DASD
  storage requirements, 1.1.2.2
Debug Tool
  default library, 1.1.3
  library requirements, 1.1.2.2
  publications, 1.1.1.2
default library, 1.1.3
default userid, A.0
DFSORT/VSE, minimum release, 1.1.2.1
distribution media, 1.1.1.1
distribution tape, 2.4.2
DL/I DOS/VS, minimum release, 1.1.2.1

DOS/VS Sort/Merge, minimum release, 1.1.2.1

E

EQAWINST sample job, 2.4.2
EQAWIVC1 sample job, 3.2.1
EQAWIVC2 sample job, 3.2.2
EQAWIVC3 sample job, 3.2.3
EQAWIVC4 sample job, 3.2.3
EQAWIVH1 sample job, 3.1.1
EQAWIVH2 sample job, 3.1.2
EQAWIVH3 sample job, 3.1.3
EQAWIVP1 sample job, 3.3.1
EQAWIVP2 sample job, 3.3.2
EQAWIVP3 sample job, 3.3.3
EQAWLDEF sample job, 2.3.1
EQAWLDEL sample job, 5.3
EQAWNDEF sample job, 2.3.2
EQAWREM sample job, 5.3
EQAWSERV sample job, 5.2.5.2
EQAWVDEF sample job, 2.3.1
EQAWVTOC sample job, 2.3.2

F

feature numbers, 1.1.1.1

I

IBM Support Center, 1.1.8
identifier, component, 1.1.1
installation
  allocate library space, 2.3
  backup original system, 2.2
  messages, 2.4.1.1
  summary of steps, 2.1.1
  using a batch job, 2.4.2
  using the interactive interface, 2.4.1
installation jobstream
  EQAWINST, 2.4.2
installation steps
  condition codes and messages, 2.4.1.1
  create library, 2.3.1
                  2.3.2
  list directory entries, 2.4.2
  retrace Debug Tool, 2.4.2
  using batch job, 2.4.2
  using the interactive interface, 2.4.1
interactive interface
  using to install Debug Tool, 2.4.1

J

job control
  for allocating Debug Tool library, 2.3.1
                                     2.3.2

for installing Debug Tool, 2.4.2

L

language support
  installing languages, 2.4.2
  selecting languages, 2.4.2
LE/VSE, minimum release, 1.1.2.1
LIBR
  delete, 5.3
  LISTDIR, 1.1.3
LIBR librarian program
  using to define Debug Tool library space, 2.3.1
                                            2.3.2
library
  allocating space for Debug Tool, 2.3
library requirements, 1.1.2.2

M

machine-readable material, 1.1.1.1
maintaining Debug Tool, 5.0
MSHP
  INSTALL product, 2.4.2
  REMOVE Debug Tool, 5.3
  RETRACE component, 2.4.2

N

national language support (NLS)
  installing, 2.4.2
  planning for, 1.2.2
  selecting, 1.1.6
             2.4.2
             4.2

O

optional licensed programs for Debug Tool, 1.1.2.1

P

partition requirements, 1.1.7.2
PL/I for VSE/ESA, minimum release, 1.1.2.1
PL/I listing exit, 1.1.7.3
PL/I PTF, 1.1.7.3
PL/I special considerations, 1.1.7.3
PTFs, 5.2.3
publications
  Debug Tool, 1.1.1.2
  VSE/ESA, 1.1.9

Q

QMF/VSE, minimum release, 1.1.2.1

R

re-installing Debug Tool, 5.1
removing Debug Tool, 5.3

S

sample jobs
  EQAWINST, 2.4.2
  EQAWIVC1, 3.2.1
  EQAWIVC2, 3.2.2
  EQAWIVC3, 3.2.3
  EQAWIVC4, 3.2.3
  EQAWIVH1, 3.1.1
  EQAWIVH2, 3.1.2
  EQAWIVH3, 3.1.3
  EQAWIVP1, 3.3.1
  EQAWIVP2, 3.3.2
  EQAWIVP3, 3.3.3
  EQAWLDEF, 2.3.1
  EQAWLDEL, 5.3
  EQAWNDEF, 2.3.2
  EQAWREM, 5.3
  EQAWSERV, 5.2.5.2
  EQAWVDEF, 2.3.1
  EQAWVTOC, 2.3.2
Shared Virtual Area
  loading phases, 4.1
  planning for, 1.2.1
  space requirements, B.0
special considerations, 1.1.7.1
  CICS in 370 mode, 1.1.7.4
  partition requirements, 1.1.7.2
  PL/I, 1.1.7.3
  run-time environment, 1.1.7.1
SQL/DS, minimum release, 1.1.2.1
standard labels
  IJSYSHF--system history file, 2.4.2
storage
  DASD requirements, 1.1.2.2
SUBSET value, 1.1.8
SVA
  See Shared Virtual Area
system history file, 2.4.2
                     5.2.3

T

tapes, 2.4.2

U

UPGRADE value, 1.1.8
userid, determining default, A.0

V

verification for C/VSE
  EQAWIVH1, 3.1.1
  EQAWIVH2, 3.1.2
  EQAWIVH3, 3.1.3
verification for COBOL/VSE
  EQAWIVC1, 3.2.1
  EQAWIVC2, 3.2.2
  EQAWIVC3, 3.2.3
  EQAWIVC4, 3.2.3
verification for PL/I VSE
  EQAWIVP1, 3.3.1
  EQAWIVP2, 3.3.2
  EQAWIVP3, 3.3.3
verify applied service, 5.2.6
verify installation
  with C/VSE, 3.1
  with COBOL/VSE, 3.2
  with PL/I VSE, 3.3
verifying Debug Tool, 3.0
VSE C Language Run-Time Support feature of VSE/ESA Version 2 Release 2, 1.1.2.1
VSE/ESA
  minimum release, 1.1.2.1
  publications, 1.1.9
  system requirements, 1.1.2
VTAM
  planning, 1.1.4

BACK_1 We'd Like to Hear from You

Debug Tool for VSE/ESA
Installation and Customization Guide
Release 1

Publication No. SC26-8798-00

Please use one of the following ways to send us your comments about this book:

° Mail--Print and use the Readers' Comments form on the next page. To print the form, select **Print** or **Copy** from the **Services** pull-down menu. Enter *COMMENTS* as the topic to be printed or copied. Mail the completed form to:

IBM Corporation, Department W92/H3
P. O. Box 49023
San Jose, California 95161-9945
U.S.A.

If you are sending the form from a country other than the United States, give it to your local IBM branch office or IBM representative for mailing.

° Fax--Print and use the Readers' Comments form on the next page and fax it to this U.S. number: 800-426-7773. To print the form, follow the instructions under "Mail."

Be sure to include the following with your comments:

- Title and publication number of this book
- Your name, address, and telephone number if you would like a reply

Your comments should pertain only to the information in this book and the way the information is presented. To request additional publications, or to comment on other IBM information or the function of IBM products, please give your comments to your IBM representative or to your IBM authorized remarketer.

IBM may use or distribute your comments without obligation.

COMMENTS Readers' Comments

Debug Tool for VSE/ESA
Installation and Customization Guide
Release 1

Publication No. SC26-8798-00

How satisfied are you with the information in this book?

Legend:
1 Very satisfied
2 Satisfied
3 Neutral
4 Dissatisfied
5 Very dissatisfied

Please circle the number that corresponds to the level of your satisfaction.

Technically accurate	1	2	3	4	5
Complete	1	2	3	4	5
Easy to find	1	2	3	4	5
Easy to understand	1	2	3	4	5
Well organized	1	2	3	4	5
Applicable to your tasks	1	2	3	4	5
Grammatically correct and consistent	1	2	3	4	5
Graphically well designed	1	2	3	4	5
Overall satisfaction	1	2	3	4	5

Please tell us how we can improve this book:

May we contact you to discuss your comments? Yes No

Name . . . . . _____
Company or Organization _____
Address . . . . . _____
_____
_____
Phone No. . . . . _____

IBM Library Server Print Preview

DOCNUM = SC26-8798-00
DATETIME = 10/29/96 16:07:22
BLDVERS = 1.2
TITLE = Debug Tool/VSE V1R1 Installation and Customization Guide
AUTHOR =

PATH = /home/webapps/epubs/htdocs/book
