CA Application Quality and Testing Tools

Symbolic Guide

r8.5
CA Product References

This document references the following CA products:
- CA Endevor® Software Change Manager (CA Endevor SCM)
- CA InterTest™ Batch
- CA InterTest™ for CICS
- CA Librarian®
- CA Optimizer®
- CA Optimizer®/II
- CA Panvalet®
- CA Realia® II
- CA SymDump® Batch
- CA SymDump® for CICS

Contact CA

Contact Technical Support

For your convenience, CA provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA products. At [http://ca.com/support](http://ca.com/support), you can access the following:
- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA product documentation, complete our short customer survey, which is also available on the CA Support website, found at [http://ca.com/docs](http://ca.com/docs).
# Contents

## Chapter 1: Introduction to Symbolic Support
- What is Symbolic Support? ................................................................. 7
- How Does Symbolic Support Work? ................................................ 8
- Symbolic Support for Optimized Applications .................................... 8
- Supported Compilers and Assemblers .............................................. 9
  - Considerations for Using the Integrated Preprocessors ..................... 10
- The PROTSYM File ........................................................................ 10
- Sharing PROTSYM Files .................................................................. 11
- Loading Symbolic Information ....................................................... 11

## Chapter 2: Creating a PROTSYM File
- CAVHPROT .................................................................................. 13

## Chapter 3: Adding Symbolic Information
- IN25SYMC ................................................................................... 18
  - IN25SYMC JCL ........................................................................... 18
  - IN25SYMC Options ..................................................................... 18
  - Required OS/VS COBOL Options ................................................. 21
  - Executing IN25SYMC as a Standalone Program ......................... 22
  - Adding IN25SYMC to Your OS/VS COBOL Procedure ............... 23
- IN25COB2 ..................................................................................... 25
  - IN25COB2 JCL ........................................................................... 25
  - IN25COB2 Options ..................................................................... 26
  - Required COBOL II Options ...................................................... 28
  - Executing IN25COB2 as a Standalone Program ......................... 29
  - Adding IN25COB2 to Your COBOL II Procedure ....................... 30
- IN25SYMP ..................................................................................... 32
  - IN25SYMP JCL ........................................................................... 32
  - IN25SYMP Options ..................................................................... 33
  - Required PL/I Options ................................................................ 35
  - Executing IN25SYMP as a Standalone Program ......................... 36
  - Adding IN25SYMP to Your PL/I Procedure ................................. 37
- IN25SYMA ..................................................................................... 39
  - IN25SYMA JCL ........................................................................... 39
  - IN25SYMA Options ..................................................................... 39
  - Required Assembler Options .................................................... 42
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executing IN25SYMA as a Standalone Program</td>
<td>43</td>
</tr>
<tr>
<td>Adding IN25SYMA to Your Assembler Procedure</td>
<td>44</td>
</tr>
<tr>
<td>IN25LINK</td>
<td>46</td>
</tr>
<tr>
<td>IN25LINK JCL</td>
<td>46</td>
</tr>
<tr>
<td>IN25LINK Options</td>
<td>47</td>
</tr>
<tr>
<td>Required Linkage Editor Options</td>
<td>50</td>
</tr>
<tr>
<td>Executing IN25LINK as a Standalone Program</td>
<td>50</td>
</tr>
<tr>
<td>Adding IN25LINK to Your Link-Edit Procedure</td>
<td>50</td>
</tr>
<tr>
<td>IN25SYMD</td>
<td>53</td>
</tr>
<tr>
<td>IN25SYMD Options</td>
<td>53</td>
</tr>
<tr>
<td>Examples</td>
<td>54</td>
</tr>
<tr>
<td>Chapter 4: Maintaining a PROTSYM File</td>
<td>57</td>
</tr>
<tr>
<td>IN25UTIL</td>
<td>57</td>
</tr>
<tr>
<td>IN25UTIL JCL</td>
<td>57</td>
</tr>
<tr>
<td>IN25UTIL Functions</td>
<td>58</td>
</tr>
<tr>
<td>Examples</td>
<td>59</td>
</tr>
<tr>
<td>Chapter 5: Messages</td>
<td>63</td>
</tr>
<tr>
<td>SYM Messages</td>
<td>63</td>
</tr>
<tr>
<td>UTIL Messages</td>
<td>105</td>
</tr>
<tr>
<td>Index</td>
<td>121</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction to Symbolic Support

This guide is intended as a reference for programmers using the symbolic support features of CA Application Quality and Testing Tools.

What is Symbolic Support?

The term *symbolic support* refers to the use of source code information from application programs to enhance and simplify the use of CA Application Quality and Testing Tools products for z/OS.

Some of these products include:
- CA InterTest Batch
- CA InterTest for CICS
- CA Optimizer/II
- CA SymDump Batch
- CA SymDump for CICS

These products provide application programmers with the critical tools needed to improve productivity throughout the application life cycle. Symbolic support makes these products easier to learn and use by speaking to programmers using terms that they recognize and understand from their own source code.

For example, using symbolic support with an interactive debugger like CA InterTest Batch lets programmers do the following:
- Enter breakpoint commands right on the source listing display.
- Stop execution at every label in a program.
- Automatically display the values of referenced variables at each statement.
- Easily display the value of any program variable.
- Set conditional breakpoints based on variable values.
- View a trace of all previously executed source statements.
Symbolic support eliminates the need to manually locate variables in storage, compute program offsets for source statements, or determine which statements were executing. You do not need to keep program listings open while debugging. All of this is done for you automatically when you use symbolic support with the CA Application Quality and Testing Tools products for z/OS.

How Does Symbolic Support Work?

When your application programs are compiled or assembled, symbolic information about the program is written to various reports in the output listing. A program called a postprocessor reads the output listing, collects the symbolic information, and stores it in a symbolic repository called a PROTSYM.

Using the listing postprocessors to collect symbolic information does not alter your program in any way. The listing produced by your compiler or assembler is used only as input. Your object module is not altered. Only the PROTSYM is updated.

After the symbolic information has been stored in the PROTSYM, you can access the information by any of the CA Application Quality and Testing Tools products to provide symbolic support for your application.

Symbolic Support for Optimized Applications

CA Application Quality and Testing Tools supports programs that have been optimized, either by the COBOL compiler's OPTIMIZE option or by CA Optimizer or CA Optimizer/II. However, debugging and post mortem analysis of these programs can sometimes result in unexpected behavior.

**Note:** The PL/I compilers are optimizing compilers.

Often as part of the optimization process, a compiler will relocate individual instructions, statements, or even entire paragraphs so that the optimized program will run more efficiently. This means that some or all of the instructions generated for a given statement may be moved to another statement, or that some or all of the statements in a paragraph may be moved to another paragraph. When this type of optimization occurs, the resulting object program and corresponding listing may not accurately represent the relationship between the source statements and their generated object code, or even between a paragraph label and the statements contained within the paragraph. As a result, there may be times when the breakpoint intercept does not occur, or when the wrong sequence of statements appears to be executed while single-stepping, or when the abending object code does not correspond to the correct source statement. There may also be times when the debugger appears to highlight the wrong statement at a breakpoint intercept or the dump analysis identifies the wrong statement as the abending source statement.
These unexpected displays do not indicate that a program is being executed incorrectly or that an abend is being incorrectly analyzed. They simply indicate that the debugger or dump analyzer sometimes cannot accurately identify exactly which object code corresponds to which source statement, or which statement is contained within which paragraph.

The CA Application Quality and Testing Tools products use the information in the compiler-generated procedure map or offset report to establish the program offset for each statement and label in the program. During execution or abend processing, the debugger or abend analyzer recognizes the start of the new statement or label by matching the program offset of the currently executing instruction with the PROTSYM information obtained from the compiler listing. Therefore, the accuracy with which the debugger or abend analyzer can represent a breakpoint or other intercept or the abending statement is only as good as the information in the compiler listing.

Inaccuracies may include, but will not be limited to:

- Incorrect execution when using the SKIP, GO stmt# or CS stmt# commands
- Failure to stop at a breakpoint at a paragraph label or statement
- Unexpected or out of sequence highlighting of statements when single-stepping
- Incorrect identification of the statement which contains the abending object code

Additionally, application abends may result from the use of the SKIP, GO stmt# or CS stmt# commands because the optimized object code may have register requirements that do not support changes to the flow of control. These commands should be avoided when debugging an optimized program.

For the best debugging results, avoid using optimization whenever possible in your testing environment. Production applications may be compiled with optimization, and debugging these applications as they exist without recompiling is supported. However, be aware that you may experience some of the inaccuracies listed previously under these circumstances.

**Supported Compilers and Assemblers**

Symbolic information is currently supported for programs compiled or assembled by the following IBM products:

- OS/VS COBOL
- OS PL/I
- VS COBOL II
Considerations for Using the Integrated Preprocessors

The integrated CICS translator and integrated SQL coprocessor of Enterprise COBOL for z/OS are fully supported by the postprocessor. It should be noted, however, that duplicate statement numbers for those statements generated by the integrated preprocessors are not saved in the PROTSYM. The CA Application Quality and Testing Tools products required this modification to the saved listing. In addition, the compiler's LIST option is required to correctly load the symbolic information into the PROTSYM file.

The postprocessor also supports the integrated CICS and SQL preprocessors of PL/I for z/OS. However, programs that contain EXEC SQL INCLUDE statements for user-defined members still require a separate precompile step. (EXEC SQL INCLUDE statements for SQLCA and SQLDA are supported when using the integrated SQL preprocessor.) It should also be noted that duplicate statement numbers for those statements generated by the integrated preprocessors are not saved in the PROTSYM. The CA Application Quality and Testing Tools products required this modification to the saved listing.

The integrated INCLUDE and MACRO preprocessors of PL/I for z/OS are not supported. A separate precompile step is required when incorporating external files into your program.

The PROTSYM File

The PROTSYM file is a VSAM relative record data set (RRDS) with an upper limit of approximately four million 2 KB data records and capable of storing symbolic information for up to 147,000 application programs at one time.

The PROTSYM file is defined by IDCAMS and must be initialized by program IN25UTIL before you can add symbolic information.
Member CAVHPROT in CAI.CAVHJCL contains sample JCL that you can use to allocate and initialize a PROTSYM file.

**Note:** The PROTSYM file cannot reside in the LSR pool.

## Sharing PROTSYM Files

Your PROTSYM files can be shared between CA products and across multiple systems and environments. A single PROTSYM file contains symbolic information for both CICS and batch programs.

Use RESERVE and DEQ macros when updating the PROTSYM file to allow sharing of the file between regions and systems. The resource major name used in the RESERVE and DEQ macros is INTERTST. If your installation uses a service that converts RESERVEs into cross-system ENQs, define the major name INTERTST to the service.

Depending on your needs, you can maintain more than one PROTSYM file at your installation. All of the CA Application Quality and Testing Tools products support the use of multiple PROTSYM files.

## Loading Symbolic Information

By modifying the JCL procedures used to compile or assemble your applications, you can automatically update the symbolic information in your PROTSYM file every time a program is rebuilt. This is the easiest way to help ensure that the symbolic information in your PROTSYM file matches the executable for every program. This is also the method that CA recommends for maintaining symbolic information.

Alternatively, you can save the listings from your compiles or assemblies and load the symbolic information later as needed. If you choose this method, you can load symbolic information into your PROTSYM file using a separate batch job. CA provides batch utilities that let you load one or more program listings residing in partitioned data sets (PDS or PDSE), CA Librarian, CA Panvalet, or CA Endevor SCM format.

Some of the CA Application Quality and Testing Tools products provide additional online functionality for viewing and maintaining PROTSYM files. For more information about the online utilities for any CA product, see the *User Guide* for that product.
Chapter 2: Creating a PROTSYM File

This chapter describes how to create a PROTSYM file.

CAVHPROT

Member CAVHPROT in CAI.CAVHJCL contains sample JCL for defining and initializing a PROTSYM file.

Note: If you are installing one of the CA Testing and Fault Management products, be sure to follow the instructions for creating the PROTSYM file in the Installation Guide for the product you are installing. Some products may provide custom JCL members that have been tailored for use with the product.

CAVHPROT contains the following two steps:

- Step 1 (DEFSYM)—Invokes IDCAMS to define the PROTSYM file.
- Step 2 (LOAD)—Invokes IN25UTIL to initialize each of the PROTSYM records.
The following JCL for member CAVHPROT shows these two steps:

```plaintext
//CAVHPROT JOB
//DEFSYM EXEC PGM=IDCAMS,REGION=1024K
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
  DELETE $PROTSYM$ CLUSTER PURGE  
  SET MAXCC=0  
  DEFINE CLUSTER (NAME($PROTSYM$)  
    REC($RECS$)  
    CISZ(2048) /* DO NOT CHANGE */  
    VOLUME($SYMVOL$)  
    RECSZ(2040 2040)  
    SHR(4 4) /* DO NOT CHANGE */  
    NUMBERED)  
  DATA (NAME($PROTSYM$.DATA))  
  /*  
//LOAD EXEC PGM=IN25UTIL,REGION=2048K  
//STEPLIB DD DSN=$LOADLIB$,DISP=SHR  
//MESSAGE DD SYSOUT=*  
//PROTSYM DD DSN=$PROTSYM$,DISP=SHR  
//CARDS DD *  
PASSWORD=$PASSWORD$  
INITIALIZE  
REPORT  
/*  
//
```

Make the following substitutions in member CAVHPROT:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PROTSYM$</td>
<td>Is the fully-qualified name of your new PROTSYM library.</td>
</tr>
<tr>
<td>$SYMVOL$</td>
<td>The volume on which the PROTSYM resides.</td>
</tr>
<tr>
<td>$RECS$</td>
<td>Is the primary space allocation in records. (See Notes 1 and 2.)</td>
</tr>
<tr>
<td>$LOADLIB$</td>
<td>Is the one- to eight-character PROTSYM update password for your installation, from IN25SOPT. (See Note 3.)</td>
</tr>
</tbody>
</table>
Submit the JCL to allocate and initialize a new PROTSYM file.

**Notes:**

1. Do not allocate any secondary space.
2. The space required depends on many factors including the size of your programs, the number of variables and labels, the average length of their names, and the LISTER options used for loading symbolic information. We recommend an initial allocation of 10,000 records. You can allocate new PROTSYM files as needed, and expand and reorganize existing files.
3. If you have not altered the installation default, specify PASSWORD=12345678
# Chapter 3: Adding Symbolic Information

This chapter describes how to add symbolic information to your PROTSYM files using the following postprocessors:

<table>
<thead>
<tr>
<th>Postprocessor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN25SYMC</td>
<td>Loads symbolic information for programs compiled using:</td>
</tr>
<tr>
<td></td>
<td>■ OS/VS COBOL version 2.3 plus PTF8 or higher</td>
</tr>
<tr>
<td></td>
<td>■ CA Optimizer</td>
</tr>
<tr>
<td>IN25COB2</td>
<td>Loads symbolic information for programs compiled using:</td>
</tr>
<tr>
<td></td>
<td>■ Enterprise COBOL for z/OS</td>
</tr>
<tr>
<td></td>
<td>■ IBM COBOL for VM</td>
</tr>
<tr>
<td></td>
<td>■ IBM COBOL for MVS and VM</td>
</tr>
<tr>
<td></td>
<td>■ AD/CYCLE COBOL/370</td>
</tr>
<tr>
<td></td>
<td>■ VS COBOL II</td>
</tr>
<tr>
<td></td>
<td>■ CA Optimizer/II</td>
</tr>
<tr>
<td>IN25SYMP</td>
<td>Loads symbolic information for programs compiled using:</td>
</tr>
<tr>
<td></td>
<td>■ Enterprise PL/I for z/OS</td>
</tr>
<tr>
<td></td>
<td>■ IBM PL/I for MVS and VM</td>
</tr>
<tr>
<td></td>
<td>■ Visual Age PL/I</td>
</tr>
<tr>
<td></td>
<td>■ OS PL/I</td>
</tr>
<tr>
<td>IN25SYMA</td>
<td>Loads symbolic information for programs compiled using:</td>
</tr>
<tr>
<td></td>
<td>■ High level Assembler for MVS and VM and VSE</td>
</tr>
<tr>
<td></td>
<td>■ Assembler H</td>
</tr>
<tr>
<td>IN25LINK</td>
<td>Reads IBM linkage editor output to collect and load subroutine mapping information for composite load modules.</td>
</tr>
<tr>
<td>IN25SYMD</td>
<td>Loads multiple COBOL, PL/I, and Assembler listings residing in PDS, PDSE, CA Librarian, CA Panvalet, or CA Endevor SCM format.</td>
</tr>
</tbody>
</table>
Use program IN25SYMC to load symbolic information for programs compiled using OS/VS COBOL or CA Optimizer.

You can execute IN25SYMC as a standalone batch job to load a single COBOL listing that has been previously saved to a permanent file, or add it to your existing OS/VS COBOL or CA Optimizer JCL procedure. The method you select depends entirely on the procedures at your own installation. Both methods are described in this section.

### IN25SYMC JCL

The following table describes the DD statements used by IN25SYMC:

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEPLIB</td>
<td>The load library containing IN25SYMC.</td>
</tr>
<tr>
<td>INPUT</td>
<td>The listing that was written to SYSPRINT by the OS/VS COBOL compiler, or by CA Optimizer, during compilation.</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>All or part of the original compiler listing is written to this file, depending on your request.</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>Any messages produced by IN25SYMC during postprocessing are written here.</td>
</tr>
<tr>
<td>PROTSYM</td>
<td>The file to which the symbolic information is written.</td>
</tr>
<tr>
<td>CARDS</td>
<td>The input control statements that define the request.</td>
</tr>
</tbody>
</table>

**Note:** If you are adding a new step for IN25SYMC to a JCL procedure, use program IN25PARM to write your input control statements to the CARDS file.

### IN25SYMC Options

Options are passed to IN25SYMC using a parameter statement in the CARDS DD. Specify the parameter statement as an in-stream control card, or when using a JCL procedure, generate it using program IN25PARM.
The following JCL shows these options:

```
//IN25PARM EXEC PGM=IN25PARM,PARM='parameter statement'
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//CARDS DD DISP=(,PASS),DSN=&&CARDS,UNIT=SYSDA,SPACE=(TRK,(1,1))
```

Parameter statements in the CARDS DD must begin in column 1.

The program name is the only required field on the parameter statement. This positional parameter defines the name that is used to store the symbolic information in the PROTSYM file. This name is used by the CA Application Quality and Testing Tools product to locate the symbolic information and is displayed when listing the contents of your PROTSYM.

In most cases, this name should be the same as the PROGRAM-ID. However, when loading symbolic information for use with CA InterTest for CICS, you must specify the name of the CICS program definition, or when using composite support, specify the monitor name.

The following example shows an in-stream parameter statement that you can use to save symbolic information using the name ORDEDIT:

```
//CARDS DD *
ORDEDIT */
```

### Controlling Printed Output with the CUTPRINT Option

Because you can load symbolic information from a permanent data set or a temporary listing file, you can also print all or part of the listing generated by the compiler.

Append the CUTPRINT option to your parameter statement to control printing of the compiler listing as follows:

- **,CUTPRINT=ALL**
  - Do not print any of the compiler listing.
- **,CUTPRINT=MAP**
  - Print the listing up to, but not including, the Data Division Map report.
- **,CUTPRINT=REF**
  - Print the listing up to, but not including, the cross reference of data names.
The following sample parameter statement saves symbolic information for program ORDEDIT and prints only the source code section of the compiler listing:

```
//CARDS DD *
ORDEDIT,CUTPRINT=MAP
/*
```

**Note:** Specify the CUTPRINT parameter only when you do not want all or part of your listing printed. The entire listing is printed if this parameter is omitted.

**Saving Your Listing for Online Display with the LISTER Option**

Append the LISTER option to your parameter statement to control which portion of your source listing is saved to the PROTSYM file as follows:

```
,LISTER=ALL
   Saves the entire OS/VS COBOL listing.

,LISTER=MAP
   Saves the OS/VS COBOL listing up to, but not including, the Data Division Map report.

,LISTER=REF
   Saves the OS/VS COBOL listing up to, but not including, the cross reference of data names.
```

The following sample parameter statement saves symbolic information for program ORDEDIT, does not print any of the listing, and saves the listing up to, but not including, the Data Division Map report to the PROTSYM file:

```
//CARDS DD *
ORDEDIT,CUTPRINT=ALL,LISTER=MAP
/*
```

**Notes:**

- If the LISTER parameter is omitted, no listing is saved in the symbolic file.
- The LISTER parameter is required for use with CA Optimizer, CA SymDump Batch, and CA InterTest Batch.
- To reduce overhead and save space in your PROTSYM file, we recommend that you specify LISTER=MAP when executing IN25SYMC.
Setting Data as Nonpurgeable

You can mark any saved symbolic data for this program as nonpurgeable. If a program's data is marked as nonpurgeable, the data is not removed from the PROTSYM when deleting programs using a purge interval batch run. However, you can delete the data by program name. See the chapter "Maintaining a PROTSYM File" for instructions on deleting data from the symbolic file.

To mark data as nonpurgeable, add the NOPURGE option to your parameter statement as the last option.

The following sample parameter statement saves symbolic information for program ORDEDIT, prints the entire listing, saves the entire listing in the PROTSYM file, and does not let symbolic data be removed from the symbolic file by a purge interval batch run.

```
//CARDS DD  *
ORDEDIT,LIST=ALL,NOPURGE
/*
```

Required OS/VS COBOL Options

The following compiler options are required to load symbolic information for OS/VS COBOL programs into the PROTSYM file:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIST or PMAP</td>
<td>Produces a condensed Procedure Division map or full Assembler Procedure Division map.</td>
</tr>
<tr>
<td>DMAP</td>
<td>Produces a Data Division map.</td>
</tr>
<tr>
<td>NONUM</td>
<td>Suppresses compiler-generated line numbers.</td>
</tr>
<tr>
<td>SXREF</td>
<td>Produces a cross-reference of data and paragraph names.</td>
</tr>
<tr>
<td>VERB</td>
<td>Produces a report of verb names.</td>
</tr>
</tbody>
</table>

The following compiler options are required to load symbolic information for a program compiled using CA Optimizer into the PROTSYM file:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAP or MDMAP</td>
<td>Produces a Data Division map or merged Data Division map.</td>
</tr>
<tr>
<td>MLIST</td>
<td>Produces a merged Procedure Division map.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>NONUM</td>
<td>Suppresses compiler-generated line numbers.</td>
</tr>
<tr>
<td>XREF</td>
<td>Produces a cross-reference of data and paragraph names.</td>
</tr>
</tbody>
</table>

To use symbolic references in OS/VS COBOL, you must declare at least one data item in working storage.

**Executing IN25SYMC as a Standalone Program**

Member CAVHSYMC in CAI.CAVHPROC contains sample JCL for executing postprocessor IN25SYMC as a standalone batch job. Use this member to load symbolic information from previously saved OS/VS COBOL listings.

```plaintext
//CAVHSYMC PROC PROTSYM=CAI.PROTSYM,
//    NAME=XXXXXXXX,
//    LISTLIB=USER.LISTLIB,
//    MEMBER=XXXXXXXX,
//    LISTER=ALL,
//    CUTPRINT=ALL
//*
//IN25PARM EXEC PGM=IN25PARM,REGION=512K,
//    PARM='&MEMBER,LISTER=&LISTER,CUTPRINT=&CUTPRINT'
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//CARDS   DD DSN=&&CARDS,DISP=(,PASS),
//        UNIT=SYSDA,SPACE=(TRK,(1,1))
//*
//IN25SYMC EXEC PGM=IN25SYMC,REGION=2M
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//PROTSYM DD DISP=SHR,DSN=PROTSYM
//INPUT   DD DISP=SHR,DSN=&LISTLIB(&MEMBER)
//CARDS   DD DSN=&&CARDS,DISP=(OLD,DELETE)
//OUTPUT  DD SYOUT=*,DCB=(LRECL=121,BLKSIZE=2440,RECFM=FBA)
//MESSAGE DD SYOUT=*  
//*
```

You can override the following procedure variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROTSYM</td>
<td>Specifies the name of the symbolic file being updated.</td>
</tr>
<tr>
<td>NAME</td>
<td>Specifies the name that is used to store the symbolic information in the PROTSYM file. This name is used by the CA Application Quality and Testing Tools products to locate the symbolic information and is displayed when listing the contents of your PROTSYM.</td>
</tr>
</tbody>
</table>
### Variable Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>In most cases, this name should be the same as the PROGRAM-ID. However, when loading symbolic information for use with CA InterTest for CICS, specify the name of the CICS program definition, or for composite support, specify the monitor name.</td>
<td></td>
</tr>
<tr>
<td>LISTLIB</td>
<td>Specifies the name of the partitioned data set containing the listing from the OS/VS COBOL compiler or CA Optimizer.</td>
</tr>
<tr>
<td>MEMBER</td>
<td>Specifies the name of the member in the listing library that contains the compiler listing for the program being added.</td>
</tr>
<tr>
<td>LISTER</td>
<td>Specifies how much of the listing to write to the OUTPUT file.</td>
</tr>
<tr>
<td>CUTPRINT</td>
<td>Specifies how much of the listing to write to the OUTPUT file.</td>
</tr>
</tbody>
</table>

### Adding IN25SYMC to Your OS/VS COBOL Procedure

To automatically update the symbolic information in your PROTSYM file whenever an OS/VS COBOL program is compiled, you can add a postprocessor step directly to the JCL procedure that you use to compile your programs.

**Note:** These same steps also apply to your CA Optimizer procedure.

Follow these steps to update your existing compile procedure:

1. Ensure that your compile step specifies all of the required OS/VS COBOL options.
2. Change the DD statement so that a temporary disk file is created for your listing, if the SYSPRINT output from your compile step is written to SYSOUT.
3. Add a new IN25SPARM step following your compile step to generate the parameter statement for the postprocessor.
4. Add a new IN25SYMC step to postprocess the listing from the compile step. The INPUT DD on this step refers to the same file as the SYSPRINT DD from the compile step.
5. Add a new IEBGENER step to print the compiler listing only if the compiler detects errors.
The following example shows modifications to a compile procedure:

```cobol
//COB  EXEC PGM=IKFCBL00,REGION=4M,
     PARM='SOURCE,DMAP,SXREF,PMAP,VERB,NONUM,&OPTIONS'  <= 1
     (Your existing DD statements for OS/VS COBOL)

//SYSPRINT DD DSN=&LST,DISP=(NEW,PASS),  <= 2
     UNIT=SYSDA,SPACE=(CYL,(1,2))
//*
//*   GENERATE THE PARAMETER STATEMENT FOR IN25SYMC
//*  //CARDS  EXEC PGM=IN25PARM,REGION=1M,COND=(4,LT),  <= 3
     PARM='&MEMBER,LISTER=ALL'
//STEPLIB DD DSN=CAI.CAVHLOAD,DISP=SHR
//CARDS  DD DSN=&CARDS,DISP=(NEW,PASS),
     UNIT=SYSDA,SPACE=(TRK,(1,1))
//*
//*   POST-PROCESS THE COMPILER LISTING
//*
//SYM  EXEC PGM=IN25SYMC,REGION=4M,COND=(4,LT)  <= 4
//STEPLIB DD DSN=CAI.CAVHLOAD,DISP=SHR
//PROTSYM DD DSN=USER.PROTSYM,DISP=SHR
//OUTPUT DD SYSOUT=*,
     // DCB=(LRECL=121,BLKSIZE=2420,RECFM=FBA)
//INPUT DD DSN=&LST,DISP=(OLD,PASS)  (See Note 1)
//CARDS  DD DSN=&CARDS,DISP=(OLD,DELETE)  (See Note 2)
//MESSAGE DD SYSOUT=*  
//*
//PRINT  EXEC PGM=IEBGENER,COND=(5,GT,COB)  <= 5
//SYSUT1 DD DSN=&LST,DISP=(OLD,DELETE)
//SYSUT2 DD SYSOUT=*  
//SYSPRINT DD DUMMY
//SYSIN  DD DUMMY
```

**Notes:**

1. If the SYSPRINT DD on your compile step refers to a permanent data set, the INPUT DD for IN25SYMC must point to the same data set.
2. If you prefer to pass your parameter statement as an override in the invoking JCL, delete the CARDS step, delete this DD statement, and add SYM.CARDS DD to your invoking JCL member.
IN25COB2

Use program IN25COB2 to load symbolic information for programs compiled using any of the following products:

- Enterprise COBOL for z/OS
- IBM COBOL for VM
- IBM COBOL for MVS and VM
- AD/CYCLE COBOL/370
- VS COBOL II
- CA Optimizer/II

**Note:** In this section, the term COBOL II refers to any of the COBOL dialects supported by the IBM compilers listed previously.

Execute IN25COB2 as a standalone batch job to load a single COBOL II listing that has been previously saved to a permanent file, or add it to your existing COBOL II or CA Optimizer/II JCL procedure. The method you select depends entirely on the procedures at your own installation. Both methods are described in this section.

IN25COB2 JCL

The following table describes the DD statements used by IN25COB2:

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEPLIB</td>
<td>The load library containing IN25COB2.</td>
</tr>
<tr>
<td>INPUT</td>
<td>The listing that was written to SYSPRINT by the COBOL II compiler, or by CA Optimizer/II, during compilation.</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>All or part of the original compiler listing is written to this file, depending on your request.</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>All messages produced by IN25COB2 during postprocessing are written to this file.</td>
</tr>
<tr>
<td>PROTSYM</td>
<td>The file to which the symbolic information is written.</td>
</tr>
<tr>
<td>CARDS</td>
<td>The input control statements that define the request.</td>
</tr>
</tbody>
</table>

**Note:** If you are adding a new step for IN25COB2 to a JCL procedure, use program IN25PARM to write your input control statements to the CARDS file.
IN25COB2 Options

Options are passed to IN25COB2 using a parameter statement in the CARDS DD. Specify the parameter statement as an in-stream control card, or when using a JCL procedure, generate it using program IN25PARM as follows:

```
//IN25PARM EXEC PGM=IN25PARM,PARM='parameter statement'
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//CARDS DD DISP=(,PASS),DSN=&&CARDS,UNIT=SYSDA,SPACE=(TRK,(1,1))
```

Parameter statements in the CARDS DD must begin in column 1.

The program name is the only required field on the parameter statement. This positional parameter defines the name that is used to store the symbolic information in the PROTSYM file. This name is used by the CA Application Quality and Testing Tools products to locate the symbolic information and is displayed when listing the contents of your PROTSYM.

In most cases, this name should be the same as the PROGRAM-ID. However, when loading symbolic information for use with CA InterTest for CICS, you must specify the name of the CICS program definition, or when using composite support, specify the monitor name.

The following example shows an in-stream parameter statement that can be used to save symbolic information using the name ORDEDIT:

```
//CARDS DD *
ORDEDIT
/*/CUTPRINT Option

Because you can load symbolic information from a permanent data set or a temporary listing file, you can also print all or part of the listing generated by the compiler.

Append the CUTPRINT option to your parameter statement to control printing of the compiler listing as follows:

```
,CUTPRINT=ALL
  Do not print any of the compiler listing.

,CUTPRINT=MAP
  Print the listing up to, but not including, the Data Division Map report.

,CUTPRINT=REF
  Print the listing up to, but not including, the cross reference of data names.
```
The following sample parameter statement saves symbolic information for program ORDEDIT and prints only the source code section of the compiler listing:

```cobol
//CARDS DD *
ORDEDIT,CUTPRINT=REF
/*

Note: Specify the CUTPRINT parameter only when you do not want all or part of your listing printed. The entire listing is printed if this parameter is omitted.
```

### Saving Your Listing for Online Display with the LISTER Option

Append the LISTER option to your parameter statement to control which portion of your source listing is saved to the PROTSYM file, as follows:

- **,LISTER=ALL**
  - Saves the entire COBOL II listing.
- **,LISTER=MAP**
  - Saves the COBOL II listing up to, but not including, the Data Division map report.
- **,LISTER=REF**
  - Saves the COBOL II listing up to, but not including, the cross reference of data names.

The following sample parameter statement saves symbolic information for program ORDEDIT, does not print any of the listing, and saves the listing up to, but not including, the Data Division map report to the PROTSYM file:

```cobol
//CARDS DD *
ORDEDIT,CUTPRINT=ALL,LISTER=MAP
/*
```

**Notes:**

- If the LISTER parameter is omitted, no listing is saved in the symbolic file.
- The LISTER parameter is required for use with CA Optimizer/II, CA SymDump Batch, and CA InterTest Batch.
- To reduce overhead and save space in your PROTSYM file, we recommend that you specify LISTER=MAP when executing IN25COB2 unless compiling with Optimizer/II, which requires LISTER=MMAP.
Setting Data as Nonpurgeable

You can mark any saved symbolic data for this program as nonpurgeable. If a program's data is marked as nonpurgeable, the data is not removed from the PROTSYM when deleting programs using a purge interval batch run. However, you can delete the data by program name. See the chapter "Maintaining a PROTSYM File" for instructions on deleting data from the symbolic file.

To mark data as nonpurgeable, add the NOPURGE option to your parameter statement as the last option.

The following sample parameter statement saves symbolic information for program ORDEdit, prints the entire listing, saves the entire listing in the PROTSYM file, and does not let symbolic data be removed from the symbolic file by a purge interval batch run.

```//CARDS DD * ORDEdit,LISTER=ALL,NOPURGE /*
```

Required COBOL II Options

The following compiler options are required to load symbolic information for COBOL II programs into the PROTSYM file:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Produces a Data Division map.</td>
</tr>
<tr>
<td>NONUMBER</td>
<td>Suppresses compiler-generated line numbers.</td>
</tr>
<tr>
<td>OFFSET or LIST</td>
<td>Produces a condensed Procedure map or full Assembler Procedure map.</td>
</tr>
<tr>
<td>XREF</td>
<td>Produces a cross-reference of data and procedure names.</td>
</tr>
</tbody>
</table>

**Note:** The LIST option is required when using the integrated CICS translator or integrated SQL coprocessor of COBOL for z/OS.

The following compiler options are required to load symbolic information for a program compiled using CA Optimizer/II into the PROTSYM file:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERTST</td>
<td>Required only when optimizing programs that are monitored using CA InterTest for CICS.</td>
</tr>
<tr>
<td>MAP or MMAP</td>
<td>Produces a Data Division map or merged Data Division map.</td>
</tr>
</tbody>
</table>
### Adding Symbolic Information

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMAP</td>
<td>Required when optimizing programs that are monitored using CA InterTest for CICS.</td>
</tr>
<tr>
<td>MOFFSET</td>
<td>Produces a merged Procedure map.</td>
</tr>
<tr>
<td>NONUM</td>
<td>Suppresses compiler-generated line numbers.</td>
</tr>
<tr>
<td>XREF</td>
<td>Produces a cross-reference of data and paragraph names.</td>
</tr>
</tbody>
</table>

**Note:** If you are using CA Optimizer/II r7 or higher, you can use the SYM compile-time option to automatically load symbolic information into your PROTSYM file during optimization. When using the SYM option, add a PROTSYM DD statement to your compile/optimize step. No additional option requirements exist when using this method.

To use symbolic references in COBOL II, you must declare at least one data item in working storage.

### Executing IN25COB2 as a Standalone Program

Member CAVHCOB2 in CAI.CAVHPROC contains sample JCL for executing postprocessor IN25COB2 as a standalone batch job. Use this member to load symbolic information from previously saved COBOL II listings.

```plaintext
//CAVHCOB2 PROC PROTSYM=CAI.PROTSYM,
  //   NAME=XXXXXXXX,
  //   LISTLIB=USER.LISTLIB,
  //   MEMBER=XXXXXXXX,
  //   LISTER=ALL,
  //   CUTPRINT=ALL
  //*
//IN25PARM EXEC PGM=IN25PARM,REGION=512K,
//   PARM='&MEMBER,LISTER=&LISTER,CUTPRINT=&CUTPRINT'
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//CARDS DD DSN=&&CARDS,DISP=(,PASS),
  // UNIT=SYSDA,SPACE=(TRK,(1,1))
  //*
//IN25COB2 EXEC PGM=IN25COB2,REGION=2M
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//PROTSYM DD DISP=SHR,DSN=6PROTSYM
//INPUT DD DISP=SHR,DSN=6LISTLIB(&MEMBER)
//CARDS DD DSN=6&CARDS,DISP=(OLD,DELETE)
//OUTPUT DD SYSOUT=*,DCB=(LRECL=133,BLKSIZE=3990,RECFM=FBA)
//MESSAGE DD SYSOUT=*  
//```
You can override the following procedure variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROTSYM</td>
<td>Specifies the name of the symbolic file being updated.</td>
</tr>
<tr>
<td>NAME</td>
<td>Specifies the name that is used to store the symbolic information in the PROTSYM file. This name is used by the CA Application Quality and Testing Tools products to locate the symbolic information and is displayed when listing the contents of your PROTSYM. In most cases, this name should be the same as the PROGRAM-ID. However, when loading symbolic information for use with CA InterTest for CICS, specify the name of the CICS program definition, or for composite support, specify the monitor name.</td>
</tr>
<tr>
<td>LISTLIB</td>
<td>Specifies the name of the partitioned data set containing the listing from the COBOL II compiler or CA Optimizer/II.</td>
</tr>
<tr>
<td>MEMBER</td>
<td>Specifies the name of the member in the listing library that contains the compiler listing for the program being added.</td>
</tr>
<tr>
<td>LISTER</td>
<td>Specifies how much of the listing to save in the PROTSYM file.</td>
</tr>
<tr>
<td>CUTPRINT</td>
<td>Specifies how much of the listing to write to the OUTPUT file.</td>
</tr>
</tbody>
</table>

**Adding IN25COB2 to Your COBOL II Procedure**

To automatically update the symbolic information in your PROTSYM file whenever a COBOL II program is compiled, you can add a postprocessor step directly to the JCL procedure that you use to compile your programs.

**Note:** These same steps also apply to the CA Optimizer/II procedure.

Follow these steps to update your existing compile procedure:

1. Ensure that your compile step specifies all of the required COBOL II options.
2. Change the DD statement so that a temporary disk file is created for your listing, if the SYSPRINT output from your compile step is written to SYSOUT.
3. Add a new IN25PARM step following your compile step to generate the parameter statement for the postprocessor.
4. Add a new IN25COB2 step to postprocess the listing from the compile step. The INPUT DD on this step refers to the same file as the SYSPRINT DD from the compile step.

5. Add a new IEBGENER step to print the compiler listing only if the compiler detects errors.

The following example shows modifications to a compile procedure:

```
//COB EXEC PGM=IGYCRCTL,REGION=4M, // PARM='S,MAP,X,LIST,NONUM,&OPTIONS'  <= 1
    (Your existing DD statements for COBOL II)

//SYSPRINT DD DSN=&&LST,DISP=(NEW,PASS),    <= 2
    //UNIT=SYSDA,SPACE=(CYL,(1,2))
/*
  GENERATE THE PARAMETER STATEMENT FOR IN25COB2
*/
//CARDS EXEC PGM=IN25PARM,REGION=1M,COND=(4,LT),  <= 3
    //PARM='&MEMBER,LISTER=ALL'
//STEPLIB DD DSN=CAI.CAVHLOAD,DISP=SHR
//CARDS DD DSN=&&CARDS,DISP=(NEW,PASS),
    //UNIT=SYSDA,SPACE=(TRK,(1,1))
/*
  POST-PROCESS THE COMPILER LISTING
*/
//SYM EXEC PGM=IN25COB2,REGION=4M,COND=(4,LT)  <= 4
//STEPLIB DD DSN=CAI.CAVHLOAD,DISP=SHR
//PROTSYM DD DSN=USER.PROTSYM,DISP=SHR
//OUTPUT DD SYSUT1,*,
    //DCB=(LRECL=133,BLKSIZE=3990,RECFM=FBA)
//INPUT DD DSN=&&LST,DISP=(OLD,PASS)              (See Note 1)
//CARDS DD DSN=&&CARDS,DISP=(OLD,DELETE)         (See Note 2)
//MESSAGE DD SYSUT1,*
/*
  PRINT EXEC PGM=IEBGENER,COND=(5,GT,COB) <= 5
//SYSUT1 DD DSN=&&LST,DISP=(OLD,DELETE)
//SYSUT2 DD SYSUT1,*
//SYSPRINT DD DUMMY
//SYSIN DD DUMMY
```

**Notes:**

1. If the SYSPRINT DD on your compile step refers to a permanent data set, the INPUT DD for IN25COB2 must point to the same data set.

2. If you prefer to pass your parameter statement as an override in the invoking JCL, delete the CARDS step, delete this DD statement, and add SYM.CARDS DD to your invoking JCL member.
IN25SYMP

Use program IN25SYMP to load symbolic information for programs compiled using any of the following products:

- Enterprise PL/I for z/OS
- IBM PL/I for MVS and VM
- Visual Age PL/I
- OS PL/I

Note: In this section, the term PL/I refers to any of the PL/I dialects supported by the IBM compilers previously listed.

Execute IN25SYMP as a standalone batch job to load a single PL/I listing that has been previously saved to a permanent file, or add it to your existing PL/I JCL procedure. The method you select depends entirely on the procedures at your own installation. Both methods are described in this section.

IN25SYMP JCL

The following table describes the DD statements used by IN25SYMP:

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEPLIB</td>
<td>The load library containing IN25SYMP.</td>
</tr>
<tr>
<td>INPUTT</td>
<td>The listing that was written to SYSPRINT by the PL/I compiler during compilation.</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>All or part of the original compiler listing is written to this file, depending on your request.</td>
</tr>
<tr>
<td>MESSAGE and MSGS</td>
<td>Messages produced by IN25SYMP during postprocessing are written to these files.</td>
</tr>
<tr>
<td>PROTSYM</td>
<td>The file to which the symbolic information is written.</td>
</tr>
<tr>
<td>CARDS</td>
<td>The input control statements that define the request.</td>
</tr>
</tbody>
</table>

Note: If you are adding a new step for IN25SYMP to a JCL procedure, use program IN25PARM to write your input control statements to the CARDS file.
IN25SYMP Options

Options are passed to IN25SYMP using a parameter statement in the CARDS DD. Specify the parameter statement as an in-stream control card, or when using a JCL procedure, generate it using program IN25PARM as follows:

```
//IN25PARM EXEC PGM=IN25PARM,PARM='parameter statement'
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//CARDS DD DISP=(,PASS),DSN=&&CARDS,UNIT=SYSDA,SPACE=(TRK,(1,1))
```

Parameter statements in the CARDS DD must begin in column 1.

The program name is the only required field on the parameter statement. This positional parameter defines the name that is used to store the symbolic information in the PROTSYM file. This name is used by the CA Application Quality and Testing Tools product to locate the symbolic information and is displayed when listing the contents of your PROTSYM.

When loading symbolic information for use with CA InterTest for CICS, you must specify the name of the CICS program definition, or when using composite support, specify the monitor name.

The following example shows an in-stream parameter statement that you can use to save symbolic information using the name ORDEDIT:

```
//CARDS DD *
ORDEDIT
/*
```

Controlling Printed Output with the CUTPRINT Option

Because you can load symbolic information from a permanent data set or a temporary listing file, you can also print all or part of the listing generated by the compiler.

Append the CUTPRINT option to your parameter statement to control printing of the compiler listing as follows:

- `\,CUTPRINT=ALL`  
  Prints none of the compile listing.
- `\,CUTPRINT=REF`  
  Terminates printing after the XREF table.
- `\,CUTPRINT=OFFSET`  
  Terminates printing after the table of offsets.
The following sample parameter statement saves symbolic information for program ORDEDIT and terminates printing after the XREF table:

```plaintext
//CARDS   DD *
ORDEDIT,CUTPRINT=REF
/*
```

**Note:** Specify the CUTPRINT parameter only when you do not want all or part of your listing printed. The entire listing is printed if you omit this parameter.

**Saving Your Listing for Online Display with the LISTER Option**

Append the LISTER option to your parameter statement to control which portion of your source listing is saved to the PROTSYM file, as follows:

- `.LISTER=ALL`
  - Saves the entire PL/I listing.
- `.LISTER=REF`
  - Saves only the source and XREF sections.
- `.LISTER=OFFSET`
  - Saves the listing up to, and including, the table of offsets.

The following sample parameter statement saves symbolic information for program ORDEDIT, does not print any of the listing, and saves only the source and XREF sections of the listing:

```plaintext
//CARDS DD *
ORDEDIT,CUTPRINT=ALL,LISTER=REF
/*
```

**Notes:**
- If the LISTER parameter is omitted, no listing is saved in the symbolic file.
- The LISTER parameter is required for use with CA Optimizer/II, CA SymDump Batch, and CA InterTest Batch.

**Setting Data as Nonpurgeable**

You can mark any saved symbolic data for this program as nonpurgeable. If a program's data is marked as nonpurgeable, the data is not removed from the PROTSYM when deleting programs using a purge interval batch run. However, you can delete the data by program name. See the chapter "Maintaining a PROTSYM File" for instructions about deleting data from the symbolic file.

To mark data as nonpurgeable, add the NOPURGE option to your parameter statement as the last option.
The following sample parameter statement saves symbolic information for program ORDEdit, prints the entire listing, saves the entire listing in the PROTSYM file, and does not let symbolic data be removed from the symbolic file by a purge interval batch run.

```
//CARDS DD *
ORDEdit, LISTER=ALL, NOPURGE
/*
```

### Required PL/I Options

The following compiler options are required to load symbolic information for PL/I programs when using the OS PL/I or IBM PL/I for MVS and VM compiler:

- AGGREGATE
- ATTRIBUTES(FULL)
- MAP
- NEST
- NOGONUM
- NONUMBER
- OPTIONS
- SOURCE
- STMT or GOSTMT
- STORAGE
- XREF(FULL)

The following compiler options are required to load symbolic information for PL/I programs when using Enterprise PL/I for z/OS or Visual Age PL/I:

- AGGREGATE
- ATTRIBUTES(FULL)
- LIMITS(NAME(31))
- LIST
- MAP
- NATLANG(ENU)
- NEST
- NOGONUM
- NOSTMT
- NUMBER
- OPTIONS
- SOURCE
- STORAGE
- XREF(FULL)

### Notes:

- Because of special considerations, if you must use the %NOPRINT compiler option, contact CA Technical Support.

- For the CA Application Quality and Testing Tools products to support date/time stamp comparison between your symbolic information and your executables, you must select TSTAMP=YES when installing your PL/I compiler.

- CA InterTest Batch, CA SymDump Batch, and CA Optimizer/II display only controlled variables.

- When using the IBM PL/I for MVS and VM compiler, the ESD option is required for programs that have controlled variables.
Executing IN25SYMP as a Standalone Program

Member CAVHSYMP in CAI.CAVHPROC contains sample JCL for executing postprocessor IN25SYMP as a standalone batch job. Use this member to load symbolic information from previously saved PL/I listings as follows:

```
//CAVHSYMP PROC PROTSYM=CAI.PROTSYM,
  // NAME=XXXXXXXX,
  // LISTLIB=USER.LISTLIB,
  // MEMBER=XXXXXXXX,
  // LISTER=ALL,
  // CUTPRINT=ALL
  //*
//IN25PARM EXEC PGM=IN25PARM,REGION=512K,
  // PARM='&MEMBER,LISTER=&LISTER,CUTPRINT=&CUTPRINT'
// STEPLIB DD DISP=SHR, DSN=CAI.CAVHLOAD
// CARDS DD DSN=&&CARDS, DISP=(PASS),
  // UNIT=SYSDA, SPACE=(TRK,(1,1))
  //*
//IN25SYMP EXEC PGM=IN25SYMP, REGION=2M
// STEPLIB DD DISP=SHR, DSN=CAI.CAVHLOAD
// PROTSYM DD DISP=SHR, DSN=PROTSYM
// INPUTT DD DISP=SHR, DSN=LISTLIB(&MEMBER)
// CARDS DD DSN=&&CARDS, DISP=(OLD,DELETE)
// SYSPRINT DD SYSOUT=*  
// MESSAGE DD SYSOUT=* 
// MSGS  DD SYSOUT=* 

/*

You can override the following procedure variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROTSYM</td>
<td>Specifies the name of the symbolic file being updated.</td>
</tr>
<tr>
<td>NAME</td>
<td>Specifies the name that is used to store the symbolic information in the PROTSYM file. This name is used by the CA Application Quality and Testing Tools products to locate the symbolic information and is displayed when listing the contents of your PROTSYM. When loading symbolic information for use with CA InterTest for CICS, specify the name of the CICS program definition, or for composite support, specify the monitor name.</td>
</tr>
<tr>
<td>LISTLIB</td>
<td>Specifies the name of the partitioned data set containing the listing from the PL/I compiler.</td>
</tr>
<tr>
<td>MEMBER</td>
<td>Specifies the name of the member in the listing library that contains the compiler listing for the program being added.</td>
</tr>
</tbody>
</table>
```
### Adding IN25SYMP to Your PL/I Procedure

To automatically update the symbolic information in your PROTSYM file whenever a PL/I program is compiled, add a postprocessor step directly to the JCL procedure you use to compile your programs.

Follow these steps to update your existing compile procedure:

1. Ensure that your compile step specifies all of the required PL/I options.
2. Change the DD statement so that a temporary disk file is created for your listing, if the SYSPRINT output from your compile step is written to SYSOUT.
3. Add a new IN2SPARM step following your compile step to generate the parameter statement for the postprocessor.
4. Add a new IN25SYMP step to postprocess the listing from the compile step. The INPUTT DD on this step refers to the same file as the SYSPRINT DD from the compile step.
5. Add a new IEBGENER step to print the compiler listing only if the compiler detects errors.

#### Variable Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISTER</td>
<td>Specifies how much of the listing to save in the PROTSYM file.</td>
</tr>
<tr>
<td>CUTPRINT</td>
<td>Specifies how much of the listing to write to the OUTPUT file.</td>
</tr>
</tbody>
</table>
The following example shows modifications to a compile procedure:

```
//PLI      EXEC PGM=IEL0AA,REGION=4M,
   // PARM='S,X(F),A(F),OP,M,LIST,STG,AG,NEST,GS,NIS'      <= 1
   // (Your existing DD statements for PL/I)

//SYSPRINT DD DSN=&&LST,DISP=(NEW,PASS),  <= 2
   // UNIT=SYSDA,SPACE=(CYL,(1,2))
   //*
/*   GENERATE THE PARAMETER STATEMENT FOR IN25SYMP
   //*
//CARDS   EXEC PGM=IN25PARM,REGION=1M,COND=(4,LT),   <= 3
   // PARM='&MEMBER,LISTER=ALL'
//STEPLIB  DD DSN=CAI.CAVHLOAD,DISP=SHR
//CARDS    DD DSN=&&CARDS,DISP=(NEW,PASS),
   // UNIT=SYSDA,SPACE=(TRK,(1,1))
   //*
/*   POST-PROCESS THE COMPILER LISTING
   //*
//SYM      EXEC PGM=IN25SYMP,REGION=4M,COND=(4,LT)       <= 4
//STEPLIB  DD DSN=CAI.CAVHLOAD,DISP=SHR
//PROTSYM  DD DSN=USER.PROTSYM,DISP=SHR
//SYSPRINT DD SYSOUT=*   <= 4
//INPUTT   DD DSN=&&LST,DISP=(OLD,PASS)                      (See Note 1)
//CARD5    DD DSN=&&CARDS,DISP=(OLD,DELETE)                  (See Note 2)
//MESSAGE  DD SYSOUT=*                                                
//MSGS     DD SYSOUT=*                                                
//*
//PRINT    EXEC PGM=IEBGENER,COND=(5,G,T,PLI)                 <= 5
//SYSUT1   DD DSN=&&LST,DISP=(OLD,DELETE)
//SYSUT2   DD SYSOUT=*                                                
//SYSPRINT DD DUMMY
//SYSPRINT DD DUMMY
/*
```

Notes:

1. If the SYSPRINT DD on your compile step refers to a permanent data set, the INPUTT DD for IN25SYMP must point to the same data set.

2. If you prefer to pass your parameter statement as an override in the invoking JCL, delete the CARDS step, delete this DD statement, and add SYM.CARDS DD to your invoking JCL member.
IN25SYMA

Use program IN25SYMA to load symbolic information for Assembler programs into your PROTSYM file.

Execute IN25SYMA as a standalone batch job to load a single Assembler listing that has been previously saved to a permanent file, or add it to your existing Assembler JCL procedure. The method you select depends entirely on the procedures at your own installation. Both methods are described in this section.

IN25SYMA JCL

The following table describes the DD statements used by IN25SYMA:

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEPLIB</td>
<td>The load library containing IN25SYMA.</td>
</tr>
<tr>
<td>INPUT</td>
<td>The listing that was written to SYSPRINT by the Assembler.</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>All or part of the original Assembler listing is written to this file, depending on your request.</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>Messages produced by IN25SYMA during postprocessing are written to this file.</td>
</tr>
<tr>
<td>PROTSYM</td>
<td>The file to which the symbolic information is written.</td>
</tr>
<tr>
<td>CARDS</td>
<td>The input control statements that define the request.</td>
</tr>
</tbody>
</table>

Note: If you are adding a new step for IN25SYMA to a JCL procedure, use program IN25PARM to write your input control statements to the CARDS file.

IN25SYMA Options

Options are passed to IN25SYMA using a parameter statement in the CARDS DD. Specify the parameter statement as an in-stream control card, or when using a JCL procedure, generate it using program IN25PARM as follows:

```plaintext
//IN25PARM EXEC PGM=IN25PARM,PARM='parameter statement'
//STEPLIB   DD DISP=SHR,DSN=CAI.CAVLOAD
//CARDS     DD DISP=(,PASS),DSN=$CARDS,UNIT=SYSDA,SPACE=(TRK,(1,1))
```

Parameter statements in the CARDS DD must begin in column 1.
The program name is the only required field on the parameter statement. This positional parameter defines the name that is used to store the symbolic information in the PROTSYM file. This name is used by the CA Application Quality and Testing Tools products to locate the symbolic information and is displayed when listing the contents of your PROTSYM.

In most cases, this name should be the same as the first CSECT in the Assembler listing. When loading symbolic information for use with CA InterTest for CICS, you must specify the name of the CICS program definition, or when using composite support, specify the monitor name.

The following example shows an in-stream parameter statement that can be used to save symbolic information using the name ORDEDITION:

```
//CARDS DD *
ORDEDITION
/*
```

## Controlling Printed Output with the CUTPRINT Option

Because you can load symbolic information from a permanent data set or a temporary listing file, you can also print all or part of the listing generated by the assembler.

Append the CUTPRINT option to your parameter statement to control printing of the assembler listing as follows:

- **,CUTPRINT=ALL**: Do not print any of the assembler listing.
- **,CUTPRINT=REF**: Stops printing the listing after the Cross Reference report.

The following sample parameter statement saves symbolic information for program ORDEDITION without printing any of the listing:

```
//CARDS DD *
ORDEDITION,CUTPRINT=ALL
/*
```

**Note:** Specify the CUTPRINT parameter only when you do not want all or part of your listing printed. The entire listing is printed if this parameter is omitted.
Saving Your Listing for Online Display with the LISTER Option

Append the LISTER option to your parameter statement to control which portion of your source listing is saved to the PROTSYM file as follows:

,LISTER=ALL
   Saves the entire assembler listing.

,LISTER=REF
   Saves the listing up to, but not including, the Cross Reference report.

The following sample parameter statement saves symbolic information for program ORDEDIT while printing and saving the listing up to, but not including, the Cross Reference report:

//CARDS DD  *
ORDEDIT,CUTPRINT=REF,LISTER=REF
/*

Notes:
- If the LISTER parameter is omitted, no listing is saved in the symbolic file.
- The LISTER parameter is required for use with CA Optimizer/II, CA SymDump Batch, and CA InterTest Batch.

Setting Data as Nonpurgeable

You can mark any saved symbolic data for this program as nonpurgeable. If a program's data is marked as nonpurgeable, the data is not removed from the PROTSYM when deleting programs using a purge interval batch run. However, you can delete the data by program name. See the chapter "Maintaining a PROTSYM File" for instructions on deleting data from the symbolic file.

To mark data as nonpurgeable, add the NOPURGE option to your parameter statement as the last option.

The following sample parameter statement saves symbolic information for program ORDEDIT, prints the entire listing, saves the entire listing in the PROTSYM file, and does not let symbolic data be removed from the symbolic file by a purge interval batch run.

//CARDS DD  *
ORDEDIT,LISTER=ALL,NOPURGE
/*
**Required Assembler Options**

The following listing options are required to load symbolic information for assembler programs:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DXREF</td>
<td>DSECT Cross-Reference</td>
</tr>
<tr>
<td>ESD</td>
<td>External Symbol Dictionary</td>
</tr>
<tr>
<td>USING</td>
<td>Using Map report</td>
</tr>
<tr>
<td>XREF(FULL) or XREF(SHORT)</td>
<td>Full cross-reference or cross-reference of referenced names</td>
</tr>
</tbody>
</table>

**Notes:**

- Do not suppress statements that define the start of a DSECT in the listing by PRINT OFF or PRINT NOGEN.
- High Level Assembler r2.0 users must also specify the LIST(121) option.
- High Level Assembler r4.0 users must specify the THREAD option. The NOTHREAD option is not supported.
Executing IN25SYMA as a Standalone Program

Member CAVHSYMA in CAI.CAVHPROC contains sample JCL for executing postprocessor IN25SYMA as a standalone batch job. Use this member to load symbolic information from previously saved Assembler listings.

```
//CAVHSYMA PROC PROTSYM=CAI.PROTSYM,
//   NAME=XXXXXXXX,
//   LISTLIB=USER.LISTLIB,
//   MEMBER=XXXXXXXX,
//   LISTER=ALL,
//   CUTPRINT=ALL
//*
//IN25PARM EXEC PGM=IN25PARM,REGION=512K,
//   PARM='&MEMBER,LISTER=&LISTER,CUTPRINT=&CUTPRINT'
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//CARDS DD DSN=&&CARDS,DISP=(,PASS),
//   UNIT=SYSDA,SPACE=(TRK,(1,1))
//*
//IN25SYMA EXEC PGM=IN25SYMA,REGION=2M
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//PROTSYM DD DISP=SHR,DSN=&PROTSYM
//INPUT DD DISP=SHR,DSN=&LISTLIB(&MEMBER)
//CARDS DD DSN=&&CARDS,DISP=(OLD,DELETE)
//OUTPUT DD SYSPOUT=*, DCB=(RECFM=FBM,LRECL=121,BLKSIZE=2420)
//MESSAGE DD SYSPOUT=* 
//*
```

You can override the following procedure variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROTSYM</td>
<td>Specifies the name of the symbolic file being updated.</td>
</tr>
<tr>
<td>NAME</td>
<td>Specifies the name that is used to store the symbolic information in the PROTSYM file. This name is used by the CA Application Quality and Testing Tools products to locate the symbolic information and is displayed when listing the contents of your PROTSYM. In most cases, this is the name of the first CSECT in the assembler listing. When loading symbolic information for use with CA InterTest for CICS, specify the name of the CICS program definition, or for composite support, specify the monitor name.</td>
</tr>
<tr>
<td>LISTLIB</td>
<td>Specifies the name of the partitioned data set containing the Assembler listing.</td>
</tr>
<tr>
<td>MEMBER</td>
<td>Specifies the name of the member in the listing library that contains the Assembler listing for the program being added.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LISTER</td>
<td>Specifies how much of the listing to save in the PROTSYM file.</td>
</tr>
<tr>
<td>CUTPRINT</td>
<td>Specifies how much of the listing to write to the OUTPUT file.</td>
</tr>
</tbody>
</table>

## Adding IN25SYMA to Your Assembler Procedure

To automatically update the symbolic information in your PROTSYM file whenever a program is assembled, add a postprocessor step directly to the JCL procedure that you use to assemble your programs.

Follow these steps to update your existing assembly procedure:

1. Be sure that your assemble step specifies all of the required Assembler options.
2. If the SYSPRINT output from your assemble step is written to SYSOUT, change the DD statement so that a temporary disk file is created for your listing.
3. Add a new IN25PARM step following your assemble step to generate the parameter statement for the postprocessor.
4. Add a new IN25SYMA step to postprocess the listing from the assemble step. The INPUT DD on this step refers to the same file as the SYSPRINT DD from the assemble step.
5. Add a new IEBGENER step to print the Assembler listing only if errors were detected during the assembly.
The following example shows modifications to an assembly procedure:

```plaintext
//ASM EXEC PGM=ASMA90,REGION=4M, // PARM='LIST,OBJECT,XREF(FULL),ESD' <= 1
(Your existing DD statements for the Assembler)

//SYSPRINT DD DSN=&&LST,DISP=(NEW,PASS), // UNIT=SYSDA,SPACE=(CYL,(1,2)) /*
/* GENERATE THE PARAMETER STATEMENT FOR IN25SYMA */
/*
//CARDS EXEC PGM=IN25PARM,REGION=1M,COND=(4,LT), // PARM='&MEMBER,LISTER=ALL'
//STEPLIB DD DSN=CAI.CAVHLOAD,DISP=SHR //CARDS DD DSN=&&CARDS,DISP=(NEW,PASS), // UNIT=SYSDA,SPACE=(TRK,(1,1)) /*
/* POST-PROCESS THE COMPILER LISTING */
/*
//SYM EXEC PGM=IN25SYMA,REGION=4M,COND=(4,LT) <= 4
//STEPLIB DD DSN=CAI.CAVHLOAD,DISP=SHR //PROTSYM DD DSN=USER.PROTSYM,DISP=SHR //OUTPUT DD SYSOUT=*,DCB=(RECFM=FBM,LRECL=121,BLKSIZE=2420) //INPUT DD DSN=&&LST,DISP=(OLD,PASS) (See Note 1) //CARDS DD DSN=&&CARDS,DISP=(OLD,DELETE) (See Note 2) //MESSAGE DD SYSOUT=* /*
/* //PRINT EXEC PGM=IEBGENER,COND=(5,GT,ASM) <= 5 //SYSUT1 DD DSN=&&LST,DISP=(OLD,DELETE) //SYSUT2 DD SYSOUT=* //SYSPRINT DD DUMMY //SYSIN DD DUMMY
```

**Notes:**

1. If the SYSPRINT DD on your compile step refers to a permanent data set, the INPUT DD for IN25SYMA must point to the same data set.

2. If you prefer to pass your parameter statement as an override in the invoking JCL, delete the CARDS step, delete this DD statement, and add SYM.CARDS DD to your invoking JCL member.
IN25LINK

Use program IN25LINK to define the additional symbolic information required for testing composite modules using CA InterTest for CICS.

A composite module consists of separately compiled or assembled parts that are brought together under a single module name by the IBM Linkage Editor. In CICS, a composite module has only one CICS program definition. You can write the main program and the called subroutines in the same or different languages.

The Composite Support feature of CA InterTest for CICS lets you test the subroutines of a composite module as if they were separate programs with separate CICS program definitions. For more information about this feature, see the CA InterTest for CICS User Guide.

IN25LINK uses output from the IBM Linkage Editor and your own additional input cards to associate the subroutines of a composite module with the monitor names you define. Typically, this program is executed as a standalone batch job to load output from a single link step.

**Note:** While it is more efficient to use IN25LINK for composite support, you can provide the same information online using the CA InterTest for CICS CNTL menu.

IN25LINK JCL

The following table describes the DD statements used by IN25LINK:

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEPLIB</td>
<td>The load library containing IN25LINK.</td>
</tr>
<tr>
<td>INPUT</td>
<td>The listing that was written to SYSPRINT by the IBM Linkage Editor.</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>All or part of the original listing is written to this file, depending on your request.</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>Messages produced by IN25LINK during postprocessing are written to this file.</td>
</tr>
<tr>
<td>PROTSYM</td>
<td>The file to which the symbolic information is written.</td>
</tr>
<tr>
<td>CARDS</td>
<td>The input control statements that define the relationships between your monitor names and the subroutines of your composite module.</td>
</tr>
</tbody>
</table>
IN25LINK Options

Options are passed to IN25LINK using parameter statements in the CARDS DD. Parameter statements in the CARDS DD must begin in column 1.

Identifying the Composite Module

The first parameter card is required. It specifies the CICS program definition name of the composite module, beginning in column 1, as follows:

composite-name[,NOPURGE]

The program name specified in this parameter statement must be the same as the CICS program definition name of the composite module.

You can also specify the NOPURGE option on this statement. This option specifies that symbolic information for the composite module cannot be purged from the PROTSYM file during a purge interval batch job.

Identifying the Main Program and Subroutines

Subsequent parameter cards optionally identify the main program and each subroutine that you want to test separately. These cards can be entered in any order and must specify two names, separated by commas, as follows:

link-name, monitor-name

link-name identifies the name of the control section as listed in the link-edit map, and must begin in column 1. For a PL/I program, specify the name of the compiler-generated control section that ends with '1'.

monitor-name specifies the name under which the program is monitored. Follow these rules in selecting a monitor-name:

- Each monitor-name must be unique.
- The monitor-name of the main program must be the same as the CICS program definition name for the composite module.
- The monitor-name of a subroutine cannot be the same as a CICS program definition name.
- The monitor-name can be identical to the link-name.
- If you have used a postprocessor to add symbolic information for the main program or a subroutine, the monitor-name is the name that you specified on the parameter card to that postprocessor.
Only the first parameter card is required; omit all subsequent cards. If you omit the subsequent parameter cards, only the name of the composite module is stored in the PROTSYM file. When you attempt to monitor the composite module with CA InterTest for CICS, you are prompted for additional composite information.

Excluding Subroutines

By default, IN25LINK excludes subroutines with CEE, DFH, DLZ, IBM, IGZ, and ILB prefixes when it reads the link-edit map. Usually, you will not want to test these programs.

You can change the default exclusion rules using a parameter card, positioned anywhere in the CARDS file after the first card, beginning in column 1, as follows:

```
EXCLUDE= (or)
EXCLUDE=name[,name,...,name]
```

Specifying `EXCLUDE` without any names instructs IN25LINK not to exclude any subroutines.

Specifying `EXCLUDE=xxxxxxxx` instructs IN25LINK to exclude subroutines whose names are represented by `xxxxxxxx`. Specify the entire `link-name` to exclude a specific subroutine, or specify a prefix to exclude a group of subroutines.

For example, to exclude all subroutines with the prefix ACA1, specify:

```
EXCLUDE=ACA1
```

Example

A composite module with the CICS program definition name BIGMOD consists of several separately compiled programs and library modules. Its main program is named MAINMOD and is written in COBOL.

BIGMOD also has three Assembler subroutines named SUBMODA, SUBMODB, and SUBMODC that you want to test separately. None of the subroutines has its own CICS program definition entry.

After assembling SUBMODA, SUBMODB, and SUBMODC, you loaded their symbolic information by executing postprocessor IN25SYMA. You selected monitor names of ASMMODA, ASMMODB, and ASMMODC for the subroutines.
Next, you compiled MAINMOD and loaded its symbolic information by executing postprocessor IN25COB2. You specified BIGMOD as the monitor name, matching the CICS program definition as required.

The linkage editor combines the main program and its three subroutines, creating the composite load module named BIGMOD. The link-edit map output for BIGMOD shows a main entry address of 160, that BIGMOD has been replaced in the load library, and that it contains the following modules:

<table>
<thead>
<tr>
<th>Control Section</th>
<th>Origin</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHECI</td>
<td>00</td>
<td>160</td>
<td>Command level COBOL stub</td>
</tr>
<tr>
<td>MAINMOD</td>
<td>160</td>
<td>78A8</td>
<td>Main program-COBOL</td>
</tr>
<tr>
<td>ILB0ATE</td>
<td>7A08</td>
<td>11A</td>
<td>COBOL library module</td>
</tr>
<tr>
<td>ILB0COM07B28</td>
<td>73</td>
<td></td>
<td>COBOL library module</td>
</tr>
<tr>
<td>SUBMODA</td>
<td>7CA0</td>
<td>1200</td>
<td>Subprogram-Assembler</td>
</tr>
<tr>
<td>SUBMODB</td>
<td>8EA0</td>
<td>100</td>
<td>Subprogram-Assembler</td>
</tr>
<tr>
<td>SUBMODC</td>
<td>9EA1</td>
<td>93</td>
<td>Subprogram-Assembler</td>
</tr>
</tbody>
</table>

Sample JCL for the link-edit step and IN25LINK is shown as follows:

```bash
//
/** Link Edit Step */
//
//LKED EXEC PGM=IEWL,......
//SYSLIB DD.....
//SYSLMOD DD.....
//SYSUT1 DD.....
//SYSLIN DD.....
//SYSPRINT DD DSN=&&INPUT,DISP=(,PASS),
  // UNIT=SYSDA,SPACE=(TRK,(2,5)),
// DCB=(DSORG=PS,LRECL=121,BLKSIZE=2420,RECFM=FB)
/**
/** IN25LINK Step */
/**
//POSTLINK EXEC PGM=IN25LINK,REGION=512K
//STEPLIB DD DSN=CAI.CAVHLOAD,DISP=SHR
//INPUT DD DSN=&&INPUT,DISP=(OLD,DELETE)
//MESSAGE DD SYSOUT=* 
//OUTPUT DD SYSOUT=*,
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=2420)
//PROTSYM DD DSN=INTRTST.PROTSYM,DISP=SHR
//CARDS DD *
BIGMOD           CICS program definition name of composite module
MAINMOD,BIGMOD   link-name and monitor-name of main program
SUBMODA,ASMMODA  link-name and monitor-name of subprogram
SUBMODB,ASMMODB  link-name and monitor-name of subprogram
SUBMODC,ASMMODC  link-name and monitor-name of subprogram
/*
**Required Linkage Editor Options**

If you are using DFSMS 1.1, you must specify the MAP parameter on the link-edit step.

**Executing IN25LINK as a Standalone Program**

Member CAVHLINK in CAI.CAVHPROC contains sample JCL for executing postprocessor IN25LINK as a standalone batch job. Use this member to load composite information from a previously saved Linkage Editor listing.

```plaintext
//CAVHLINK PROC PROTSYM=CAI.PROTSYM,
//                 LISTLIB=USER.LISTLIB
//                 MEMBER=XXXXXXXX
//*
//IN25LINK EXEC PGM=IN25LINK,REGION=2M
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//PROTSYM DD DISP=SHR,DSN=6PROTSYM
//INPUT DD DISP=SHR,DSN=6LISTLIB(&MEMBER)
//OUTPUT DD SYSOUT=*,DCB=(LRECL=121,BLKSIZE=2440,RECFM=FBA)
//CARDS DD DDNAME=CARDS
//MESSAGE DD SYSOUT=*  
```

You can override the following procedure variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROTSYM</td>
<td>Specifies the name of the symbolic file being updated.</td>
</tr>
<tr>
<td>LISTLIB</td>
<td>Specifies the name of the partitioned data set containing your saved IBM Linkage Editor output listings.</td>
</tr>
<tr>
<td>MEMBER</td>
<td>Specifies the name of the member in the listing library that contains the listing for the composite module.</td>
</tr>
</tbody>
</table>

When invoking this JCL procedure, you must include a CARDS DD statement in the invoking JCL.

**Adding IN25LINK to Your Link-Edit Procedure**

To automatically update the symbolic information in your PROTSYM file whenever a program is link-edited, you can add a postprocessor step directly to the JCL procedure that you use to link-edit your programs.
Follow these steps to update your existing link-edit procedure:

1. Be sure that your link-edit step specifies all of the required link-edit options.

2. If the SYSPRINT output from your link-edit step is written to SYSOUT, change the DD statement so that a temporary disk file is created for your listing.

3. Add a new IN25PARM step following your link-edit step to generate the parameter statement for the postprocessor.

4. Add a new IN25LINK step to postprocess the listing from the link-edit step. The INPUT DD on this step refers to the same file as the SYSPRINT DD from the link-edit step.

5. Add a new IEBGENER step to print the linkage editor listing only if errors were detected during the link.
The following example shows modifications to a link-edit procedure:

```plaintext
//LINK      EXEC PGM=IEWL,REGION=2M, PARM='LIST,LET,XREF,MAP' <= 1
  (Your existing DD statements for the link edit)

//SYSPRINT DD DSN=&&LST,DISP=(NEW,PASS), UNIT=SYSDA,SPACE=(CYL,(1,2)) /*
  /*   GENERATE THE PARAMETER STATEMENT FOR IN25LINK /*
  /*
  //CARDS   EXEC PGM=IN25PARM,REGION=1M,COND=(4,LT), PARM='&MEMBER' <= 3
  //STEPLIB  DD DSN=CAI.CAVHLOAD,DISP=SHR
  //CARDS    DD DSN=&&CARDS,DISP=(NEW,PASS), UNIT=SYSDA,SPACE=(TRK,(1,1)) /*
  /*    POST-PROCESS THE LINK EDIT OUTPUT /*
  /*
  //SYM      EXEC PGM=IN25LINK,REGION=4M,COND=(4,LT) <= 4
  //STEPLIB  DD DSN=CAI.CAVHLOAD,DISP=SHR
  //PROTSYM  DD DSN=USER.PROTSYM,DISP=SHR
  //OUTPUT   DD SYSOUT=*,DCB=(RECFM=FBM,LRECL=121,BLKSIZE=2420)
  //INPUT    DD DSN=&&LST,DISP=(OLD,PASS)                        (See Note 1)
  //CARDS    DD DSN=&&CARDS,DISP=(OLD,DELETE)                   (See Note 2)
  //MESSAGE  DD SYSOUT=*                                       /*
  /*
  //PRINT    EXEC PGM=IEBGENER,COND=(5,GT,LINK) <= 5
  //SYSUT1   DD DSN=&&LST,DISP=(OLD,DELETE)
  //SYSUT2   DD SYSOUT=*                                       //SYSPRINT DD DUMMY
  //SYSIN    DD DUMMY
```

**Notes:**

1. If the SYSPRINT DD on your link-edit step refers to a permanent data set, the INPUT DD for IN25LINK must point to the same data set.

2. If you prefer to pass your parameter statement as an override in the invoking JCL, delete the CARDS step, delete this DD statement, and add SYM.CARDS DD to your invoking JCL member.

**Note:** Using this method, IN25PARM only generates the first parameter card identifying the name of the composite module. No subsequent cards are generated. This serves only to identify the name of the composite module, which is stored in the PROTSYM file. When you attempt to monitor the composite module with CA InterTest for CICS, you may be prompted for additional composite information.
Use program IN25SYMD to load symbolic information from multiple COBOL, Assembler, or PL/I listings into your PROTSYM in a single run.

The following table describes the DD statements used by IN25SYMD:

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEPLIB</td>
<td>The load library containing IN25SYMD.</td>
</tr>
<tr>
<td>PROTSYM</td>
<td>The file to which the symbolic information is written.</td>
</tr>
<tr>
<td>LISTLIB</td>
<td>The data set name of the PDS, PDSE, CA Librarian library, CA Panvalet library, or CA Endevor SCM library containing the listings to be added.</td>
</tr>
<tr>
<td>REPORT</td>
<td>An execution summary is written to this file.</td>
</tr>
<tr>
<td>OPTIN</td>
<td>The input control statements that define the request.</td>
</tr>
</tbody>
</table>

If loading symbolic information from CA Endevor SCM, the CA Endevor SCM AUTHLIB and CONLIB must be either in LINKLIST or in the STEPLIB concatenation. When loading symbolic information from CA Librarian or CA Panvalet, the CA Librarian or CA Panvalet CAILIB must be either in Linklist or in the STEPLIB concatenation.

**IN25SYMD Options**

Options are passed to the IN25SYMD using parameter statements in the OPTIN DD. Specify the parameter statements as an in-stream file.

Parameter statements contain one or more control statements, separated by commas, and each having the following syntax:

\[ \text{keyword=value} \]

Specify the following option keywords to IN25SYMD:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTYP</td>
<td>Identifies the library type of the listing library specified by the LISTLIB DD statement. This keyword is required. Valid values are:</td>
</tr>
</tbody>
</table>

- PDS—Partitioned data set (including PDSE)
- SEQ—Sequential data set (see Note)
- LIB—CA Librarian library
IN25SYMD

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAN</td>
<td>CA Panvalet library</td>
</tr>
<tr>
<td>NDV</td>
<td>CA Endevor SCM library</td>
</tr>
</tbody>
</table>

**FROM** Identifies the member name for single listings, the starting member name for a range of members, or a name prefix with trailing asterisk. This keyword is required.

**TO** Identifies the last member name in a range of members.

**MSG** Identifies the message reporting level. Valid values are:
- ALL—displays all messages.
- RC—displays a one-line return code message for each program.
- NONE—suppresses all messages.

**Note:** When LTYP=SEQ is specified, the sequential file contains only one program listing. Specify the symbolic name using the FROM keyword, omit the TO keyword, and change the LISTLIB DD name to INPUT.

When using LTYP=NDV, you must also change the EXEC card to the following JCL:

```jcl
//STEP1 EXEC PGM=NDVRC1,PARM='IN25SYMD',REGION=4M
```

**Examples**

This section contains postprocessor IN25SYMD examples.

**Example 1**

All of the programs with the prefix PAY are loaded into the PROTSYM file from a CA Librarian library, with all of the messages displayed in the REPORT file.

```jcl
//STEP1 EXEC PGM=IN25SYMD,REGION=4M
//STELIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//PROTSYM DD DISP=SHR,DSN=USER.PROTSYM
//LISTLIB DD DISP=SHR,DSN=USER.LIBRARIAN.LIBRARY
//REPORT DD SYOUT=* 
//OPTIN DD *
LTYP=LIB,FROM=PAY*,MSG=ALL */
```
Example 2

Program COBDEMO is loaded into the PROTSYM file from a sequential listing file, with messages suppressed. Note that the DD statement for the LISTLIB has been renamed to INPUT for LTYP=SEQ.

```
//STEP1   EXEC PGM=IN25SYMD,REGION=4M
//STEPLIB  DD DISP=SHR,DSN=CAI.CAVLOAD
//PROTSYM  DD DISP=SHR,DSN=USER.PROTSYM
//INPUT    DD DISP=SHR,DSN=USER.SEQ.LISTING
//REPORT   DD SYSOUT=*  
//OPTIN    DD *
  LTYP=SEQ, FROM=COBDEMO, MSG=NONE
/*
```

Example 3

Programs whose names begin with C, D, or E are loaded into the PROTSYM file from a partitioned data set. A one-line return code message is written to the REPORT file for each program.

```
//STEP1   EXEC PGM=IN25SYMD,REGION=4M
//STEPLIB  DD DISP=SHR,DSN=CAI.CAVLOAD
//PROTSYM  DD DISP=SHR,DSN=USER.PROTSYM
//LISTLIB  DD DISP=SHR,DSN=USER.PDS.LIBRARY
//REPORT   DD SYSOUT=*  
//OPTIN    DD *
  LTYP=PDS, FROM=C, TO=E9999999, MSG=RC
/*
```
Chapter 4: Maintaining a PROTSYM File

This chapter describes how to maintain the PROTSYM file.

IN25UTIL

The IN25UTIL batch utility program maintains and reports on the symbolic file.

This program runs in batch, separate from the postprocessors that are used to load symbolic information into the symbolic file.

IN25UTIL JCL

Member CAVHUTIL in CAI.CAVHPROC contains the following sample JCL for executing IN25UTIL:

```
// JOB
//IN25UTIL EXEC PGM=IN25UTIL,REGION=2M
//STEPLIB DD DISP=SHR,DSN=CAI.CAVHLOAD
//PROTSYM DD DISP=SHR,DSN=USER.PROTSYM
//OUTPUT DD SYSOUT=*,DCB=(LRECL=133,BLKSIZE=3990)
//MESSAGE DD SYSOUT=* 
//CARDS DD *
(input cards go here)
/*
```

The following table describes the DD statements used by IN25UTIL:

<table>
<thead>
<tr>
<th>DDname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEPLIB</td>
<td>The load library containing IN25UTIL.</td>
</tr>
<tr>
<td>PROTSYM</td>
<td>The file on which maintenance or reporting is being performed.</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>Output and messages from IN25UTIL are written to this file.</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>Output from the PRINT function is written to this file.</td>
</tr>
<tr>
<td>CARDS</td>
<td>Contains the function request statements.</td>
</tr>
</tbody>
</table>
IN25UTIL Functions

IN25UTIL functions are requested using control statements in the CARDS file. All control statements in the CARDS file must begin in column 1.

The following table describes the maintenance functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASSWORD=pw</td>
<td>Specifies a password that is required when maintaining the symbolic file. You need to specify the password only once per execution, but it must precede the first update request. The value specified must match the value of the SYMPSWD keyword in the IN25SOPT macro. The default installation password is 12345678.</td>
</tr>
<tr>
<td>INITIALIZE</td>
<td>Initializes the symbolic file. This function must always be run after a symbolic file is created using VSAM Access Method Services. For a newly defined file, the IN25UTIL program preformats all records. If you perform this function for an existing file, all symbolic data is removed. The PASSWORD control statement must precede the INITIALIZE control statement.</td>
</tr>
<tr>
<td>PURGE=nnn</td>
<td>Removes symbolic data for any program that has not been compiled or assembled within the number of days specified by nnn, where nnn is a decimal number from 1 to 365. The PASSWORD control statement must precede the PURGE control statement. Data for programs loaded using the NOPURGE postprocessor option are not affected by this function.</td>
</tr>
<tr>
<td>DELETE=name</td>
<td>Deletes all symbolic data for the program specified by name. The PASSWORD control statement must precede the DELETE control statement.</td>
</tr>
<tr>
<td>REPORT</td>
<td>Produces a Symbolic File report that contains statistics and a detailed report on each program.</td>
</tr>
<tr>
<td>PRINT=name</td>
<td>Prints the saved source listing for the program specified by name.</td>
</tr>
<tr>
<td>UNLOAD=name</td>
<td>Copies the symbolic data for the program specified by name to the unload device. Specify UNLOAD=ALL to unload symbolic data for every program.</td>
</tr>
</tbody>
</table>
**Function**  
**Description**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| RELOAD=name | Reloads the symbolic data for the program specified by name from the device specified by the RELOAD DD statement. Specify RELOAD=ALL to reload the symbolic data for every program.  
Programs are reloaded only if they do not exist in the PROTSYM.  
The PASSWORD control statement must precede the RELOAD control statement. |

**Examples**

The following examples contain sample JCL for performing common symbolic file maintenance tasks.

**Initializing a Symbolic File**

The following example initializes a symbolic file:

```clike
//UTILITY  JOB
//STEP1    EXEC PGM=IN25UTIL
//STEPLIB  DD DSN=CAI.CAVHLOAD,DISP=SHR
//MESSAGE  DD SYSOUT=*  
//PROTSYM  DD DSN=CAI.PROTSYM,DISP=SHR
//CARDS    DD * 
PASSWORD=12345678
INITIALIZE
/*
```

**Purging Symbolic Information by Age**

The following example purges all programs that have not been compiled or assembled within the last 20 days:

```clike
//UTILITY  JOB
//STEP1    EXEC PGM=IN25UTIL
//STEPLIB  DD DSN=CAI.CAVHLOAD,DISP=SHR
//MESSAGE  DD SYSOUT=*  
//PROTSYM  DD DSN=CAI.PROTSYM,DISP=SHR
//CARDS    DD * 
PASSWORD=12345678
PURGE=20
/*
```
Deleting Symbolic Information by Program

The following example deletes all symbolic data for programs ORDEDIT and TEST1:

```
//UTILITY  JOB
//STEP1    EXEC PGM=IN25UTIL
//STEPLIB  DD DSN=CAI.CAVHLOAD,DISP=SHR
//MESSAGE  DD SYSOUT=*  
//PROTSYM  DD DSN=CAI.PROTSYM,DISP=SHR
//CARDS    DD *
PASSWORD=12345678
DELETE=ORDEDIT
DELETE=TEST1
/*
```

Generating Reports and Purging Programs

The following example generates a system report, purges all programs that have not been compiled or assembled within 90 days, and generates another system report:

```
//UTILITY  JOB
//STEP1    EXEC PGM=IN25UTIL
//STEPLIB  DD DSN=CAI.CAVHLOAD,DISP=SHR
//MESSAGE  DD SYSOUT=*  
//PROTSYM  DD DSN=CAI.PROTSYM,DISP=SHR
//CARDS    DD *
PASSWORD=12345678
REPORT
PURGE=90
REPORT
/*
```

Unloading Programs

The following example unloads all programs on the symbolic file to tape:

```
//UTILITY  JOB
//STEP1    EXEC PGM=IN25UTIL
//STEPLIB  DD DSN=CAI.CAVHLOAD,DISP=SHR
//MESSAGE  DD SYSOUT=*  
//PROTSYM  DD DSN=CAI.PROTSYM,DISP=SHR
//UNLOAD   DD UNIT=TAPE, VOL=SER=UNLDTP,
  // LABEL=(,NL), DISP=(OLD,KEEP),
  // DCB=(RECFM=FB, LRECL=2042, BLKSIZE=20420)
//CARDS    DD *
UNLOAD=ALL
/*
```
Reloading Programs

The following example reloads all programs on tape to the symbolic file:

```plaintext
//UTILITY JOB
//STEP1 EXEC PGM=IN25UTIL
//STEPLIB DD DSN=CAI.CAVHLOAD,DISP=SHR
//MESSAGE DD SYSOUT=* 
//RELOAD DD UNIT=TAPE,
// VOL=SER=UNLDTP,
// LABEL=(,NL),
// DISP=(OLD,KEEP),
// DCB=(RECFM=FB,LRECL=2042,BKSIZE=20420)
//PROTSYM DD DSN=CAI.PROTSYM,DISP=SHR
//CARDS DD *
PASSWORD=12345678
RELOAD=ALL
/*
```

Printing a Program Listing

The following example prints the saved listing for program ORDEDIT:

```plaintext
//UTILITY JOB
//STEP1 EXEC PGM=IN25UTIL
//STEPLIB DD DSN=CAI.CAVHLOAD,DISP=SHR
//MESSAGE DD SYSOUT=* 
//OUTPUT DD SYSOUT=A,DCB=(LRECL=133,BKSIZE=3990)
//PROTSYM DD DSN=CAI.PROTSYM,DISP=SHR
//CARDS DD *
PRINT=ORDEDIT
/*
```
Reorganizing the Symbolic File

The following example reorganizes or changes the size of the symbolic file. This job unloads all programs, deletes and defines the symbolic file, initializes the symbolic file, reloads all programs, and generates a system report.

//UTILITY JOB
//UNLOAD EXEC PGM=IN25UTIL
//STEPLIB DD DSN=CAI.CAVHLOAD,DISP=SHR
//SYSUDUMP DD SYSOUT=* 
//MESSAGE DD SYSOUT=* 
//UNLOAD DD DISP=(,PASS),
//UNIT=TAPE,
//VOL=SER=RELDTAP,
//LABEL=(,NL),
//DCB=(RECFM=FB,LRECL=2042,BLKSIZE=20420)
//PROTSYM DD DSN=CAI.PROTSYM,DISP=SHR
//CARDS DD *
UNLOAD=ALL
/*
//IDCAMS EXEC PGM=IDCAMS,COND=(0,NE,UNLOAD)
//SYSUT1 DD UNIT=SYSDA,VOL=SER=SYMVOL,DISP=SHR
DELETE 'CAI.PROTSYM'
DEFINE CLUSTER (NAME(CAI.PROTSYM)-
    VOLUME(SYMVOL) -
    FILE(SYSUT1) -
    CYLINDERS(20) -
    CISZ(2048) -
    RECSZ(2040 2040) -
    SHR(4 4) -
    NUMBERED) -
DATA (NAME(CAI.PROTSYM.DATA))
/*
//RELOAD EXEC PGM=IN25UTIL,COND=(0,NE,UNLOAD)
//STEPLIB DD DSN=CAI.CAVHLOAD,DISP=SHR
//SYSUDUMP DD SYSOUT=* 
//MESSAGE DD SYSOUT=* 
//RELOAD DD DISP=(OLD,KEEP),
//UNIT=TAPE,
//VOL=SER=RELDTAP,
//LABEL=(,NL),
//DCB=(RECFM=FB,LRECL=2042,BLKSIZE=20420)
//PROTSYM DD DSN=CAI.PROTSYM,DISP=SHR
//CARDS DD *
PASSWORD=12345678
INITIALIZE 
RELOAD=ALL 
REPORT 
/*
Chapter 5: Messages

This chapter contains messages produced by the symbolic postprocessors and the maintenance program IN25UTIL.

SYM Messages

SYM002

LISTER= OPTION INVALID, NO LISTING SAVED

Reason:
The LISTER option was invalid.

Action:
Correct the LISTER option and resubmit your job stream.

SYM003

CUTPRINT= OPTION INVALID, LISTING WILL BE PRINTED

Reason:
The CUTPRINT option was invalid.

Action:
Correct the CUTPRINT option and resubmit your job stream.

SYM004

SYMBOLIC INFORMATION HAS BEEN SET AS NON-PURGABLE

Reason:
The NOPURGE option was specified on the parameter statement.

Action:
None.
SYM005

PARAMETER OPTIONS INVALID, SYMBOLIC STEP NOT RUN

Reason:
An invalid option was specified on the parameter statement.

Action:
Correct the parameter statement and resubmit your job stream.

SYM006

LISTER=XXX REQUESTED

Reason:
This identifies the LISTER option specified.

Action:
None.

SYM007

CUTPRINT=XXX REQUESTED

Reason:
This identifies the CUTPRINT option specified.

Action:
None.

SYM008

CONTROL STATEMENT IS MISSING - JOB TERMINATED

Reason:
No parameter statement was found.

Action:
Correct the parameter statement and resubmit your job stream.
**SYM009**

**PROTSYM HAS REACHED MAXIMUM NUMBER OF PROGRAMS**

**Reason:**
The symbolic file has exceeded the maximum number of program entries.

**Action:**
Delete some programs for your file and resubmit your job stream, or define a new PROTSYM file and then resubmit your job stream using the new file.

**SYM010**

**PROCESSING HAS BEGUN FOR PROGRAM NAME - XXXXXXXX**

**Reason:**
Program xxxxxxxx will be processed and its output will reside on the symbolic file.

**Action:**
None.

**SYM011**

**SEQUENCE NUMBERS ARE NOT IN ASCENDING SEQUENCE. JOB WILL BE TERMINATED - CONTACT CA**

**Reason:**
This is an internal verification check error.

**Action:**
Contact CA Technical Support.

**SYM012**

**RECORD COUNT ERROR AT XXXXXXX**

**Reason:**
This is an internal verification check error.

**Action:**
Contact CA Technical Support.
SYM013

QQQQ ERROR R15 = X'YY' ERROR CODE = X'ZZ'

Reason:
A VSAM request, qqqq, has resulted in an error. The maximum number of program entries has been reached. This usually indicates a corrupted symbolic file.

Action:
Contact CA Technical Support.

SYM014

PROTSYM OUT OF SPACE

Reason:
The symbolic file has exceeded the maximum number of records.

Action:
Delete some programs for your file or increase the size of your symbolic file. Then resubmit your job stream. Or, define a new PROTSYM file and then resubmit your job stream using the new file.

SYM015

DD STATEMENT MISSING FOR PROTSYM

Reason:
The DD statement for the symbolic file could not be found.

Action:
Correct the JCL and resubmit the job stream.

SYM016

ENQ ERROR - CODE = X'YY'

Reason:
In VSE, an error occurred when an SVC 63 (ENQ) was issued. This is normally caused by a NOT FOUND condition for the symbolic file.

Action:
Check the JCL and correct. If you still cannot determine the cause of this error, contact CA Technical Support.
SYM017

UTILITY IN PROGRESS ON PROTSYM FILE - JOB TERMINATED

Reason:
The CA InterTest batch utility program was processing the symbolic file at the same time that the postprocessor step was running.

Action:
First ensure that the last UTILITY run was successful. To turn off this indicator, run a UTILITY job with a REPORT function.

SYM018

MAIN STORAGE NOT AVAILABLE - INCREASE REGION SIZE

Reason:
A GETMAIN (MVS) or GETVIS (VSE) request has failed due to insufficient storage.

Action:
Resubmit the job stream with a larger region size.

SYM019

PROGRAM ABENDED 111 AT DISPLACEMENT XXXXX

Reason:
A condition has occurred that required a termination with a dump.

Action:
Contact CA Technical Support.

SYM020

SYMBOLIC FILE UPDATED SUCCESSFULLY

Reason:
The postprocessor has ended successfully and the program has been added to the symbolic file.

Action:
None.
SYM021

**XXXXXX SOURCE STATEMENTS SAVED**

**Reason:**
On a successful completion, the number of source statements, xxxxxxxx, displays.

**Action:**
None.

SYM022

**YYYY TOTAL RECORDS INSERTED INTO SYMBOLIC FILE**

**Reason:**
On a successful completion, the total number of records added to the symbolic file, yyyy, displays.

**Action:**
None.

SYM023

**POST-PROCESSOR TERMINATED**

**Reason:**
The CA InterTest postprocessor program has ended.

**Action:**
None.

SYM024

**INPUT FILE PROCESSED**

**Reason:**
The CA InterTest postprocessor program has read all of the data passed to it in the INPUT data set.

**Action:**
None.
SYM025

PROCEDURE NAMES CROSS-REFERENCE NOT FOUND

Reason:
The procedure names cross-reference output area was not found.

Action:
Without this area, CA InterTest is unable to process any requests using a PARAGRAPH NAME. If PARAGRAPH NAMES are needed, correct the COBOL options and resubmit the job stream.

SYM026

INPUT FILE IS EMPTY

Reason:
The data set specified by the INPUT JCL statement was empty.

Action:
Correct and resubmit the job stream.

SYM027

CAPEX OPTION ERROR - MLIST MUST BE SPECIFIED

Reason:
The required CA Optimizer (CAPEX) option MLIST was not specified.

Action:
Correct and resubmit the job stream.

SYM028

COBOL COMPILER OPTIONS ARE INCORRECT - XXXX - NOT FOUND

Reason:
Area xxxxx could not be found in the COBOL listing. For working storage, at least one data item must be declared.

Action:
Correct COBOL compiler options or add a working storage data item to produce the required area and resubmit the job stream.
SYM029

**XXXXX NOT REQUESTED - PROCESSING TERMINATED**

**Reason:**
The required COBOL II option, xxxx, was not specified.

**Action:**
Correct and resubmit the job stream.

SYM030

**NO CSECT FOUND - PLEASE REMOVE ANY PRINT NOGEN STATEMENTS THAT MIGHT PREVENT THE CSECT STATEMENT FROM BEING PRINTED**

**Reason:**
After examining the entire Assembler listing, no labeled CSECT could be found.

**Action:**
Correct and resubmit the job stream.

SYM031

**INPUT IS FROM COMMAND TRANSLATOR**

**Reason:**
The input passed to the CA InterTest postprocessor program was generated by the CICS command translator and not by the compiler or Assembler.

**Action:**
Correct the procedure so that the passed output is from the compiler/Assembler.

SYM032

**OPEN FOR INPUT FAILED**

**Reason:**
The INPUT file could not be opened. Check the printed output for additional error messages that may have been produced by the operating system.

**Action:**
Correct the JCL and resubmit the job stream.
SYM033

OPEN FOR OUTPUT FAILED

Reason:
The OUTPUT file could not be opened. Check the printed output for additional error messages that may have been produced by the operating system.

Action:
Correct the JCL and resubmit the job stream.

SYM034

OPEN FOR CARDS FAILED

Reason:
The CARDS file could not be opened. Check the printed output for additional error messages that may have been produced by the operating system.

Action:
Correct the JCL and resubmit the job stream.

SYM035

NO CROSS-REFERENCE FOUND - SYMBOLIC NAMES CANNOT BE USED

Reason:
The Cross-Reference area could not be found in the Assembler listing. Without this area, symbolic names cannot be resolved.

Action:
Resubmit the job stream with the XREF (SHORT) or XREF (FULL) Assembler parameter.

SYM036

PASSED PARAMETER STATEMENTS

Reason:
The next group of printed lines will be an echo of all the parameter statements read by the postprocessor.

Action:
None.
**SYM038**

**XXX LINK-EDITOR MAP RECORD(S) HAVE BEEN ADDED TO ENTRY**

**Reason:**
The IN25LINK program has added xxx linkage editor records to an existing symbolic entry.

**Action:**
None.

**SYM039**

**NO LINK-EDITOR ENTRY FOUND FOR XXXXXXXXX**

**Reason:**
The link-name, xxxxxxxxx, specified on an IN25LINK postprocessor control card could not be found in the output produced by the linkage editor.

**Action:**
You can do one of the following:
- Ignore the error.
- Correct the parameter card and resubmit the job stream.
- Use the CNTL Composite Support menu to add the information online. If the LISTER parameter is omitted, no listing is saved in the symbolic file.

**SYM040**

**THE FOLLOWING OPTIONS WERE NOT SPECIFIED**

**Reason:**
The list of options following this message was not specified. Without these options, some CA InterTest facilities may not function.

**Action:**
Correct the JCL and resubmit the job stream.
SYM042

NO STATEMENT INFORMATION WAS FOUND IN THE PASSED LISTING

Reason:
The postprocessor program could not find the data needed to process the statement information.
For a COBOL program, this data is produced by the CLIST or PMAP compiler option.
For a COBOL II program, this data is produced by the OFFSET or LIST compiler option.
For a CA Optimizer program, this data is produced by the MLIST option.
For a CA Optimizer/II program, this data is produced by the MOFFSET option.
The following are some of the conditions that may cause this error:
- Errors in the compile which cause the suppression of the data.
- Incorrect compiler options.
- Compiler control statements that suppress the needed data.

Action:
Determine why the required area was suppressed. Correct and resubmit the job stream.

SYM043

THE CSECT NAMED xxxxxxxx WAS NOT FOUND IN THE EXTERNAL SYMBOL DICTIONARY

Reason:
In a VSE Assembler, a CSECT statement named xxxxxxxx was found, but that name could not be found in the ESD list.

Action:
Correct the JCL and resubmit the job stream.
SYM044

**INPUT DATA WAS NOT PRODUCED BY THE LINKAGE-EDITOR**

**Reason:**
The data passed to the IN25LINK postprocessor using the INPUT JCL statement did not contain the output produced by the linkage editor step, or the linkage editor PARM=MAP is not in effect.

**Action:**
Correct the JCL and resubmit the job stream.

SYM045

**INVALID POWER PARAMETER FORMAT**

**Reason:**
The POWER parameter was specified incorrectly.

**Action:**
Correct the parameter card and resubmit the job stream.

SYM046

**INVALID POWER CLASS SPECIFIED (NOT A - Z)**

**Reason:**
The POWER LST class specified on the parameter card was not an alphabetic character.

**Action:**
Correct the parameter card and resubmit the job stream.

SYM047

**INVALID POWER JOB NAME**

**Reason:**
The name specified in the POWER parameter is invalid.

**Action:**
Correct the parameter card and resubmit the job stream.
SYM048

IN25PWRI, POWER INTERFACE MODULE, WAS NOT FOUND

Reason:
The CA InterTest module, IN25PWRI, was not found in the load library specified in the JCL. This module is required when using the VSE/POWER LST queue.

Action:
You can do one of the following:
- Change the JCL to point to a load library that contains the IN25PWRI module.
- Change the JCL to use SYSLST instead of POWER.

SYM049

IN25OPTS MODULE NOT FOUND - DEFAULTS ARE USED

Reason:
Module IN25OPTS was not found in the load library specified in the JCL.

Action:
If the default POWER options can be used, no action is required. However, we recommend that the correct IN25OPTS module be added to the load library.

SYM050

POWER INTERFACE IS NOT ACTIVE

Reason:
The POWER parameter was specified incorrectly.

Action:
Correct the parameter card and resubmit the job stream.

SYM051

SYNTAX ERROR IN WH= OPTION

Reason:
A syntax error was found in the CA Realia/II Workbench Host Option parameter.

Action:
Correct the parameter card and resubmit the job stream.
SYM052

WH= SOURCE FILE OPTION IS INCORRECT

Reason:
An error was found in the CA Realia/II Workbench Host Option parameter.

Action:
Correct the parameter card and resubmit the job stream.

SYM053

WH= MEMBER NAME IS MISSING OR INCORRECT

Reason:
An error was found in the CA Realia/II Workbench Host Option parameter.

Action:
Correct the parameter and resubmit the job stream.

SYM054

REALIA WORKBENCH HOST OPTION WAS REQUESTED WITHOUT HAVING A LISTER OPTION SPECIFIED

Reason:
The LISTER option was not specified in the CA Realia/II Workbench Host Option parameter.

Action:
The LISTER=ALL option is set by default.

SYM055

REALIA WORKBENCH HOST OPTION INTERFACE MODULE WAS NOT LINKED WITH THIS MODULE

Reason:
The CA Realia/II Workbench Host Option Interface Module was not linked.

Action:
Correct and resubmit the job.
SYM056

A CA LMP RIMSTAT ERROR HAS BEEN DETECTED

Reason:
The CA License Management Program is not properly installed.

Action:
Ensure that you have the proper level of CA Common Services for z/OS installed.

SYM057

CSECT NAME 'XXXXXXXX' IS GT 8 BYTES. CSECT IGNORED

Reason:
CSECT name is greater than 8 bytes.

Action:
Correct and resubmit the job.

SYM058

HLASM OPTION "LIST(133|MAX)" IS NOT SUPPORTED. USE OPTION "LIST(121)".

Reason:
HLASM option LIST(133|MAX) is not supported.

Action:
Correct the option by using LIST(121).

SYM059

HLASM OPTION "NOTHREAD" IS NOT SUPPORTED. USE OPTION "THREAD".

Reason:
The NOTHREAD option is not supported.

Action:
Correct the option by using the THREAD option (IBM default).

SYM090

ATTEMPT TO UPDATE SAM RECORD VIA DATA RPL
SYM Messages

SYM091

ATTEMPT TO UPDATE DATA RECORD VIA SAM RPL

SYM092

ATTEMPT TO UPDATE DIRECTORY RECORD VIA DATA RPL

SYM093

ATTEMPT TO UPDATE DATA RECORD VIA DIRECTORY RPL

SYM094

NO ENQ IS SET FOR A XXXXXX

SYM095

DIRECTORY RECORD IS CORRUPTED

SYM097

MISMATCH ON AVAILABLE FREE RECORDS IN ONE SAM

SYM Messages 2-11

Symbolic Post-Processor Program Messages

SYM960

MAX SAM RECS EXCEEDED. PROTSYM FILE MAY BE CORRUPTED

SYM961

MAX DIR RECS EXCEEDED. PROTSYM FILE MAY BE CORRUPTED

Reason:
Error messages SYM090 to SYM097, and SYM960 to SYM961 are produced by a special checkout procedure.

This procedure looks for conditions that corrupt the symbolic file. If any of the conditions are found, one of the above messages is issued, and the batch processor program abends.

Action:
Contact CA Technical Support for help in handling these errors.
**SYMP001I**

** **** INTERTEST VERSION ID: x.x COMPILED: dd mm yy hh:mm:ss **

**Reason:**
This reason is informative. It identifies the version and compile date and time for the PL/I postprocessor program, IN25SYMP.

**Action:**
None.

**SYMP002I**

**INTERTEST - PL/I POSTCOMPILER RUN FOR PROGRAM: proglame**

**Reason:**
This reason is informative during initialization of IN25SYMP. Processing for the requested program has begun.

**Action:**
None.

**SYMP003I**

**LISTER RECORDS ADDED TO PROTSYM FILE: nn**

**Reason:**
This reason is informative during termination of IN25SYMP. The number of records used on the PROTSYM file for listing information displays.

**Action:**
None.

**SYMP004I**

**SYMBOLIC RECORDS ADDED TO PROTSYM FILE: nn**

**Reason:**
This reason is informative during termination of IN25SYMP. The number of records used on the PROTSYM file for symbolic information displays.

**Action:**
None.
SYMP005I

IN25ASMP RETURN CODE ON FINAL PROCESSING: nn

Reason:
This reason is informative. The final return code from routine IN25ASMP displays.

Action:
None.

SYMP006I

IN25SYMP RETURN CODE ON FINAL PROCESSING: nn

Reason:
This reason is informative. The final return code from routine IN25SYMP displays.

Action:
None.

SYMP101W

BLOCK NAME CANNOT BE RESOLVED. VARIABLE AND PARAMETER RESOLUTION MAY BE AFFECTED. DRC=n BLOCK NAME= blockname

Reason:
A variable was found in the variable storage map section of the compiler output whose associated block cannot be identified.

Action:
Contact CA Technical Support with program listings.

SYMP102W

VARIABLE DISPLACEMENT CANNOT BE DETERMINED. DRC=n, VARIABLE NAME= variable name

Reason:
A variable was found in the variable storage map section of the compiler output whose associated attributes could not be identified.

Action:
Make sure all data names are unique within each block.
SYMP103W

PARAMETER DISPLACEMENT CANNOT BE DETERMINED. DRC=n, PARAMETER NAME= parameter name

Reason:
A parameter passed to a procedure could not be resolved and will not be available to the CORE transaction, or an associated block could not be determined.

Action:
Make sure all data names are unique within each block and that no procedure or entry statements are suppressed by %NOPRINT. Check for other error messages.

SYMP104W

BASED/DEFINED VAR PTR/BASE CANNOT BE FOUND. DRC=n, VARIABLE NAME= variable name POINTER NAME = pointer name

Reason:
The pointer associated with a BASED variable or the base variable of a redefined variable (see message SYMP115W) could not be resolved. The pointer name can be the name of the associated pointer or the base area for the redefinition.

Action:
Try simplifying the expression of the base area. For example, use the following syntax to declare a redefined variable:

```c
...BASED( ADDR( variable) )
```

or

```c
...DEFINED base_variable...
```

Also, make sure the variable does not refer to a subscripted array element or qualified name.
SYMP105W

ADDRESSES OF THE FOLLOWING BASED VARIABLES CANNOT BE DETERMINED FROM THE PL/I CROSS REFERENCE. IN ORDER TO ACCESS THESE VARIABLES ON-LINE, YOU MUST SPECIFY A QUALIFIED NAME IN THE INTERTEST CORE COMMAND.

Reason:
One or more BASED (*) variables were found in the PL/I cross reference. IN25SYMP was unable to determine the pointer to the variables specified.

Action:
This reason is informative only. When attempting to view the variable online with CA InterTest for CICS, you must use the POINTER option in the CORE command.

SYMP106W

ADDRESSES OF THE BASED STRUCTURES CONTAINING THE FOLLOWING VARIABLES COULD NOT BE DETERMINED. IN ORDER TO ACCESS THESE VARIABLES ON-LINE, YOU MUST SPECIFY A QUALIFIED NAME IN THE INTERTEST CORE COMMAND.

Reason:
A SYMP105W message was issued for the major structure containing the variable names.

Action:
This reason is informative only. When attempting to view the variable online with CA InterTest for CICS, you must use the POINTER option in the CORE command.

SYMP107W

THE FOLLOWING PL/I COMPILER OPTION IS REQUIRED IN ORDER FOR IN25SYMP TO PROCESS CORRECTLY compiler option.

Reason:
The listed PL/I compiler option was not specified.

Action:
Be sure that the following options were specified to the PL/I compiler:

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGGREGATE</td>
<td>OPTIONS</td>
</tr>
<tr>
<td>ATTRIBUTES(FULL)</td>
<td>SOURCE</td>
</tr>
<tr>
<td>MAP</td>
<td>STMT or GOSTMT</td>
</tr>
<tr>
<td>NEST</td>
<td>STORAGE</td>
</tr>
<tr>
<td>OFFSET</td>
<td>XREF(FULL)</td>
</tr>
</tbody>
</table>
If all of the previous options were not specified, code the appropriate options and rerun the job.

If all of the previous options were specified, ensure that the options section is included in the PL/I compiler listing and that the output of the compiler was directed to the INPUTT DD statement, or that the compiler terminated with a return code of 8 or less.

**SYMP108W**

*** WARNING: "%NOPRINT" OPTION SPECIFIED. ***

**Reason:**
The NOPRINT option was specified on the control card. This option suppresses checking for the occurrence of the preprocessor control statement %NOPRINT. If the use of this facility suppresses the listing of PROCEDURE or ENTRY statements that contain parameter lists, or suppresses the listing of the final END statement of the program, the results may be unpredictable. Possible effects include the inability to resolve parameter variables properly, as well as possible abnormal termination of SYMP.

**Action:**
Remove the NOPRINT option or ensure required information is not suppressed.

**Note:** Indiscriminate use of this option could cause program failure or incorrect results.

**SYMP110W**

**STRUCTURE NAME "IN" ENCOUNTERED WHILE SCANNING CROSS REFERENCE/ATTRIBUTE LIST VARIABLE: variable name STATEMENT NO: statement number**

**Reason:**
While resolving structure nesting information, the identified variable in the specified statement number was found in a major or minor structure named IN. If this is not true, the variable may not be properly resolved.

**Action:**
None, if the previous condition described is correct. Otherwise, contact CA Technical Support.
SYMP111W

UNABLE TO DETERMINE LENGTH OF DATA ITEM. "REFER" OPTION NOT SUPPORTED FOR BASED VARIABLES. VARIABLE: variable name
STATEMENT NO: statement number

Reason:
Dynamically sized items are not supported. Examples include string parameters that inherit their size from the calling block, automatic strings whose length is an expression, and based strings whose length is specified by the REFER option. Such variables may not be available to the CORE transaction when debugging.

Action:
None.

SYMP112W

UNEXPECTED CONVERSION ERROR WHILE PROCESSING THE CROSS REFERENCE/ATTRIBUTE LIST FOR: VARIABLE: variable name
STATEMENT NO: statement number

Reason:
An unanticipated error occurred during processing of the attribute list.

Action:
Contact CA Technical Support.

SYMP113W

LABEL DISPLACEMENT CANNOT BE DETERMINED FOR: xxxxxxxxx

Reason:
The indicated label could not be associated with its block.

Action:
Make sure label names are unique. Check the statement/offset table for missing or out-of-order entries. Do not use END block-name statements to close multiple blocks.
SYMP114W

NO STORAGE WILL BE ALLOCATED FOR THE FOLLOWING VARIABLES. THE VARIABLES CANNOT BE ACCESSED.

List of variables

**Reason**

The PL/I compiler has not allocated storage for these variables. This is normally the result of the variables not being referenced or their values never being used.

**Action**

Informational only. No action is necessary.

SYMP115W

THE FOLLOWING LINE CONTAINS "%NOPRINT" WHICH COULD PREVENT PROPER RESOLUTION OF SOME VARIABLES.

input compiler record

**Reason:**

Use of the compiler option %NOPRINT may cause the suppression of information needed to resolve variables properly.

**Action:**

Make sure %NOPRINT does not suppress the printing of:

- PROCEDURE or ENTRY statements containing parameter lists.
- Declarations of redefined variables using expressions, such as:
  
  ```pli
  ...BASED( ADDR( variable ) )
  ...DEFINED base_variable...
  ```

SYMP116W

THE FOLLOWING "DEFINED" STRUCTURE IS NESTED TOO DEEPLY TO PROPERLY RESOLVE ITS LENGTH: structure

**Reason:**

This error occurs only with a structure nesting depth that exceeds 16, which is not currently allowed by any version of the PL/I compiler.

**Action:**

None.
SYMP117W

LENGTH OF DEFINED STRUCTURE "structure" CANNOT BE RESOLVED DUE TO CONTAINED ARRAY "array"

Reason:
When a CORE command displays the specified DEFINED structure containing an array or a substructure that contains an array, the area displayed is correct, but it is limited to the correct length. This error occurs only with a structure nesting depth that exceeds 16, which is not currently allowed by any version of the PL/I compiler.

Action:
None.

SYMP118W

VARIABLE "element name" IS A DUPLICATE NAME WITHIN A STRUCTURE. STRUCTURE, "structure name", IS DECLARED IN STATEMENT statement number DRC = n.

Reason:
A duplicate name was found in a structure. All names must be unique within a major structure. If not, the following problems can occur:

- Selection of an incorrect variable
- Incorrect attributes for a variable
- Incorrect resolution of parameters
- System errors not apparently related to these variables

Action:
Assign unique names.
SYMP119W

NO AUTOMATIC VARIABLES FOUND FOR PROCEDURE procname
PARAMETER parameter CANNOT BE RESOLVED DRC = n.

Reason:
No AUTOMATIC variables were declared for the specified procedure. Parameters
cannot be resolved for any PL/I procedure unless AUTOMATIC variables are
declared in it.

Action:
You can circumvent this restriction by declaring an AUTOMATIC variable. This
variable does not have to be referenced.

SYMP120W

STRUCTURE INFORMATION FOR: variable-name IS INVALID. SPECIFY
CONNECT ATTRIBUTE, OTHERWISE ONLINE STRUCTURE REQUEST WILL
SHOW INVALID INFORMATION. STATEMENT #: nnn

Reason:
The CONNECT attribute is missing from the named variable, found in the
specified statement number. When the CONNECT attribute is missing, CA
InterTest and CA SymDump cannot properly format the variable.

Action:
Redefine the variable with the CONNECT attribute.

SYMP121W

COMPILER EXTERNAL SYMBOL DICTIONARY NOT AVAILABLE.
ADDRESSES OF THE FOLLOWING CONTROLLED VARIABLES CANNOT BE
DETERMINED.

List of variable names

Reason:
The External Symbol Dictionary is not in the compiler listings. As a result it is not
possible to determine the address of controlled variables. Therefore they cannot
be displayed.

Action:
Informational only. If you desire to display controlled variables, recompile with
the ESD option.
SYMP122W

**THE FOLLOWING VARIABLES CAN ONLY BE ACCESSED THROUGH CA INTERTEST BATCH.**

List of variables

**Reason:**
The information required to locate these variables is not available in a CICS environment.

**Action:**
Informational only. These variables cannot be displayed by CA InterTest.

SYMP123W

**THE FOLLOWING EXTERNAL VARIABLES CANNOT BE ACCESSED THROUGH CA INTERTEST BATCH DUE TO THE RENT OPTION.**

**Reason:**
The information necessary to locate external variables for reentrant compiled programs is not available and therefore cannot be displayed.

**Action:**
Informational only. No action is necessary.

SYMP124W

**THE POINTER FOR A BASED ADDR VARIABLE CAN NOT HAVE MORE THAN 3 SUBSCRIPTS**

**VARIABLE NAME = variable_name**

**Reason:**
CA InterTest for CICS and CA InterTest Batch do not support variables with more than three subscripts.

**Action:**
Informational only. These variables cannot be displayed.
SYMP125W

CURRENTLY, THE FOLLOWING VARIABLES CAN NOT BE PROPERLY DISPLAYED BY INTERTEST/CICS and INTERTEST BATCH.

List of variables

Reason:
CA InterTest for CICS and CA InterTest Batch cannot support variables with BASED ADDRESS variables with subscripts.

Action:
Informational only. These variables cannot be displayed.

SYMP126W

THE FOLLOWING VARIABLES RESIDE WITHIN AN UNNAMED STRUCTURE AND MAY NOT BE QUALIFIED

List of variables

Reason:
The indicated variables are defined within a structure that was defined with a name of '*'. Since the variable name '*' cannot be referenced, the structure name cannot be used as qualifier.

Action:
Informational only. If the variable names indicated are not unique, the variables may not be displayable by the CA Application Quality and Testing Tools.
**SYMP182W**

**UNABLE TO OBTAIN TIMESTAMP INFORMATION FROM LISTING, CURRENT DATE/TIME WILL BE USED. PARAMETER parameter CANNOT BE RESOLVED DRC = n.**

**Reason:**
The compiler generated timestamp cannot be found in the SYSPRINT listing used as input. The date and time of the postprocessor's execution is used in place of the compiler timestamp.

**Action:**
Contact CA Technical Support.

**Note:** For messages SYMP500 to SYMP599, you are advised to contact CA Technical Support. Provide the message number, text, and DRC code (if available). Also provide the IN25SYMP version ID and compile date and time (see message SYMP001I) and the release number of the PL/I compiler you are using.

**SYMP500E**

**ERROR SCANNING SOURCE FOR COMMENTS/LITERALS STATEMENT NUMBER: statement number COMMENT/LITERAL SCAN FLAGS: xxxxxxxxx**

**Reason:**
An error occurred while scanning for comments and literals. If comments or literals contain text that resembles a PROCEDURE or ENTRY statement with parameters, some parameter variables may not be properly resolved.

**Action:**
Make sure comments and literals do not contain text as previously described. Check message SYMP103W. Contact CA Technical Support if this error occurs.
SYMP501E

UNABLE TO DETERMINE BEGINNING STATEMENT NUMBER FOR BLOCK. ANY VARIABLES DECLARED IN THIS BLOCK PRIOR TO THE INDICATED STATEMENT NUMBER WILL NOT BE RESOLVED. xxxxxxxx AT STATEMENT:statement number

Reason:
An error occurred while identifying the beginning statement number for a block. Variables declared between the beginning of the block and the statement number may not be properly resolved.

Action:
Place an executable statement before the first variable declaration in this block. Contact CA Technical Support if this error occurs.

SYMP502E

EXPECTED "IN..." NOT FOUND WHILE SCANNING CROSS REFERENCE/ATTRIBUTE LIST VARIABLE variable STATEMENT NO: statement number

Reason:
An error occurred while resolving structure nesting. Structure elements may not be properly resolved.

Action:
Contact CA Technical Support.

SYMP503E

WARNING - THE FOLLOWING LINE ASSUMED TO BE PART OF COMPILER INFORMATORY MESSAGE AND IGNORED:

input compiler record DRC = n

Reason:
The specified compiler input record was ignored.

Action:
Contact CA Technical Support.
SYM Messages

SYMP503E

WARNING - THE FOLLOWING LINE ASSUMED TO BE A REPETITION OF VARIABLE MAP COLUMN HEADINGS:

input compiler record DRC = n

Reason:
The specified compiler input record was assumed to repeat variable map column headings.

Action:
Contact CA Technical Support.

SYMP503E

WARNING - THE FOLLOWING LINE ASSUMED TO IDENTIFY COMPILER DIAGNOSTIC MESSAGES SECTION:

input compiler record DRC = n

Reason:
The specified compiler input record was assumed to identify the compiler's diagnostic messages section.

Action:
Contact CA Technical Support.

SYMP503E

WARNING - THE FOLLOWING LINE CONTAINS TITLE IN UNEXPECTED POSITION - ASSUMED TO LIST STATEMENT OFFSETS:

input compiler record DRC = n

Reason:
The specified compiler input record contains a title in an unexpected position. This line is assumed to list statement offsets.

Action:
Contact CA Technical Support.
SYMP503E

**WARNING - THE FOLLOWING LINE CONTAINS UNEXPECTED DECIMAL OFFSET - HEX VALUE WILL BE USED:**

input compiler record DRC = n

**Reason:**
The specified compiler input record contains a decimal offset. The hexadecimal value will be used instead.

**Action:**
Contact CA Technical Support.

SYMP504E

**UNEXPECTED STRUCTURE NESTING DEPTH OF n EXCEEDS PREVIOUS COMPILER LIMIT IN STATEMENT statement number**

**Reason:**
The nesting depth exceeds the limit in the specified compiler statement.

**Action:**
Contact CA Technical Support.

SYMP505E

**BLOCK NOT FOUND FOR VARIABLE variable, DECLARED IN STATEMENT NO. nn**

**Reason:**
The block for the specified variable could not be found.

**Action:**
Contact CA Technical Support.

SYMP507E

**STATEMENT OUT OF EXPECTED RANGE FOR BLOCK block name**

**Reason:**
The statement was out of the expected range for the specified block.

**Action:**
Contact CA Technical Support.
SYMP508E

**SEARCH FAILED FOR:** variable DRC = n

**Reason:**
The specified variable could not be found.

**Action:**
Contact CA Technical Support.

SYMP509E

**INCLUDE OF USER FILES THROUGH INTEGRATED PREPROCESSOR NOT SUPPORTED. USE SEPARATE PREPROCESSOR STEP**

**Reason:**
The integrated INCLUDE and MACRO preprocessors of PL/I for z/OS are not supported. If you are using the integrated SQL preprocessor, programs that contain EXEC SQL INCLUDE statements for user-defined members still require a separate preprocessor step. (EXEC SQL INCLUDE statements for SQLCA and SQLDA are supported when using the integrated SQL preprocessor.)

**Action:**
Use a separate preprocessor step to incorporate external files into your program.

SYMP599E

**UNEXPECTED CHAR IN MARGIN DELIMITER POSITION M, L, MACRO, MRGCHR: aaa, bbb, ccc, ddd DRC = n.**

**Reason:**
An unexpected character was found in the margin delimiter position.

**Action:**
Contact CA Technical Support.
SYMP599E

"aaaa" POSITIONED AT BLANK STRING: "portion of listing line" DRC = n.

Reason:
aaaa was positioned at a blank.

Action:
Contact CA Technical Support.

SYMP599E

"IN" FOUND AT INVALID LOCATION STRING: "portion of listing line" DRC = n.

Reason:
An IN structure was found at an invalid location.

Action:
Contact CA Technical Support.

SYMP599E

VARIABLE "variable", DECLARED IN STATEMENT statement number, IS ALREADY RESOLVED DRC = n.

Reason:
The specified variable is already resolved.

Action:
Contact CA Technical Support.

SYMP599E

AGGREGATE NAME NOT MATCHED aggregate name DRC = n.

Reason:
The specified aggregate name was not found.

Action:
Contact CA Technical Support.
**SYMP599E**

**FORWARD SCAN FAILED TO RESOLVE BLOCK OUTERMOST BLOCK ASSUMED DRC = n.**

**Reason:**
The block could not be resolved. The outermost block is assumed.

**Action:**
Contact CA Technical Support.

**SYMP599E**

**STRUCTURE PROCESSING ERROR variable name**

**Reason:**
An error has occurred in recording a variables structure position while processing the aggregate table.

**Action:**
This can occur if the PL/I compile abnormally terminates or terminates with a return code greater than eight. Check these conditions and if they occur, correct them and rerun the job. If neither has occurred, contact CA Technical Support with the complete job output.

**SYMP599E**

**STORAGE DESCRIPTOR NAME NOT MATCHED variable name**

**Reason:**
A variable name in the storage offset table could not be matched.

**Action:**
This can occur if the PL/I compile abnormally terminates or terminates with a return code greater than eight. Check these conditions and if they occur, correct them and rerun the job. If neither has occurred, contact CA Technical Support with the complete job output.
**SYMP801E**

nn STATEMENTS ENCOUNTERED; PROGRAMS WITH MORE THAN 10,000 STATEMENTS NOT SUPPORTED. RESULTS WILL BE UNPREDICTABLE, WITH SEVERE ERRORS LIKELY.

**Reason:**
Because of a bug in the PL/I compiler that truncates high order digits from statement numbers in the statement/offset table, programs with over 9,999 statements cannot be supported completely. This message is usually accompanied by other error messages. Online debugging will be affected significantly.

**Action:**
None.

**SYMP901S**

THE PL/1 COMPILER OPTIONS LISTED PREVIOUSLY WERE NOT SPECIFIED AND HAVE CAUSED IN25SYMP TO TERMINATE

**Reason:**
One or more SYMP107W messages were issued.

**Action:**
Ensure that the following options were specified to the PL/I compiler:

- AGGREGATE
- ATTRIBUTES(FULL)
- MAP
- NEST
- OFFSET
- OPTIONS
- SOURCE
- STMT or GOSTMT
- STORAGE
- XREF(FULL)

See the SYMP107W messages for the specific options missing.

If all options were not specified, code the appropriate options and rerun the job.

If all of the previous options have been specified, ensure that the options section is included in the PL/I compiler listing and that the output of the compiler was directed to the INPUTT DD statement, or that the compiler terminated with a return code of 8 or less.
**SYMP902S**

**UNEXPECTED TERMINATION OF COMPILER OUTPUT HAS OCCURRED**

**DRC = n.**

**Reason:**
While scanning the PL/I compiler output, an end-of-file condition was raised for the INPUTT DD statement.

**Action:**
Check that all the PL/I options required by IN25SYMP were set, or that the compiler terminated with a return code of 8 or less. Correct the problem and rerun the compile and IN25SYMP.

**SYMP903S**

**UNEXPECTED ERROR DETECTED. ERROR CORE=nnnn DRC = n.**

**Reason:**
A PL/I error condition occurred that IN25SYMP was not designed to handle.

**Action:**
Check the PL/I ONCODEs and take the appropriate suggested action. If unsuccessful, contact CA Technical Support with the dump.

**SYMP904S**

**IN25ASMP FUNCTION FAILED. FUNCTION CODE= n DRC = n.**

**Reason:**
The routine that does I/O to the PROTSYM file returned with a non-zero return code.

**Action:**
Check the error messages from IN25ASMP and take appropriate action, and then rerun the job. This error usually indicates a problem with the PROTSYM file. If this is the first use of the PROTSYM file, check the job that initialized the file for normal completion. If this is not the first use of the file, check for a physical or logical error on the file. The MESSAGE DD statement contains more information.

**Note:** Function code 8 usually means the file is full.
SYMP905S

**ERROR ENCOUNTERED ON FILE INPUTT, ERROR CODE = nnnn DRC = n.**

**Reason:**
An error has occurred trying to open the INPUTT file.

**Action:**
Ensure that the ddname DLBL INPUTT points to a file that contains the output listing of the PL/I compiler.

SYMP906S

**ERROR ENCOUNTERED ON FILE SYSPRINT, ERROR CODE = nnnn DRC = n.**

**Reason:**
An error has occurred attempting to open the SYSPRINT file.

**Action:**
Ensure that the ddname DLBL SYSPRINT is coded in your JCL.

SYMP908S

**SUBSCRIPT OVERFLOW IN TABLE T4 DRC = n.**

**Reason:**
The internal table used to keep track of variable information has exceeded its limits. This error is usually caused when the Attribute and Cross Reference section of the compile output could not be found by IN25SYMP.

**Action:**
This can occur if the compiler options ATTRIBUTE(FULL) and XREF(FULL) were not specified, or if the PL/I compile abnormally terminates or terminates with a return code greater than 8. Check for these conditions and if they occur, correct and rerun the job. If neither has occurred, contact CA Technical Support with the dump.
SYMP909S

SUBSCRIPT OVERFLOW IN TABLE T5 DRC = n.

Reason:
The internal table used to keep track of PROC BLOCKS has exceeded its limits. This error is usually caused when the Storage Requirements section of the compile output could not be found by IN25SYMP. This can occur if the compiler option STORAGE was not specified, or if the PL/I compile has abnormally terminated or terminated with a return code greater than 8.

Action:
Check for these conditions and if they occur, correct and rerun the job. If neither has occurred, contact CA Technical Support with the dump.

SYMP910S

SUBSCRIPT OVERFLOW IN TABLE T6 DRC = n.

Reason:
An internal table used to keep track of information about statements has overflowed.

Action:
This occurs under several conditions. If reorder has been specified in the procedure block, remove the reorder option. This can also occur if many source statement lines contain multiple PL/I statements (for example, A=B; C=D; E=F;). In this case, separate the PL/I statements into multiple source lines. This can also occur if the PL/I preprocessor command %NOPRINT was specified. In this case, remove the %NOPRINT command.
SYMP911S

**SUBSCRIPT OVERFLOW IN TABLE T6A DRC = n.**

**Reason:**
An internal table used to keep track of information about statements has overflowed.

**Action:**
This occurs under several conditions. If reorder has been specified in the procedure block, remove the reorder option.

This can also occur if a large number of source statement lines contain multiple PL/I statements (for example, \( A=B; \ C=D; \ E=F; \)). In this case, separate the PL/I statements into multiple source lines.

This can also occur if the PL/I preprocessor command \( %\text{NOPRINT} \) was specified. In this case, remove the \( %\text{NOPRINT} \) command.

SYMP912S

**INPUT FROM THE PL/I COMPILER CONTAINS AN INVALID STATEMENT NUMBER IN COLUMNS 1-8 OF THE RECORD.**

**Reason:**
An error was detected while attempting to find the largest PL/I statement number.

**Action:**
Check compiler output and the line listed. Correct the problem and rerun the job.

SYMP916S

**UNEXPECTED ERROR ENCOUNTERED DURING PHASE 1 DRC = n.**

**Reason:**
An error has occurred during phase 1 of IN25SYMP processing.

**Action:**
Check the job log for operating system or PL/I error indicators. Correct and rerun the job.
**SYMP917S**

**DOS INITIALIZATION FAILED IN "ASMP"; POST-PROCESSOR SYMP WILL BE TERMINATED DRC = n.**

**Reason:**
For VSE only. Usually, a control card is missing or invalid. Or, there may be a problem with the symbolic file or with the JCL for the MESSAGE file.

**Action:**
Correct the control card, symbolic file, or JCL.

**SYMP919S**

**END OF FILE ENCOUNTERED FOR "INPUTT" DURING PHASE 1 OF PRE-PROCESSOR DRC = n.**

**Reason:**
An end-of-file condition occurred during the phase 1 scan of the PL/I compiler output.
IN25SYMP prints the compiler output and terminates processing. This condition can occur when the PL/I compiler detects program errors that would prohibit successful compilation.

**Action:**
Refer to the PL/I compiler output for error messages.

**SYMP920S**

"CDLOAD" FAILED FOR "IN25ASMP". RETURN CODE = nnnn

**Reason:**
VSE only. The VSAM I/O routine, IN25ASMP, could not be dynamically loaded. Possible causes include incorrect installation or partition JCL that does not allow sufficient storage for loading this phase.

**Action:**
Check that the specified phase is in the execution phase library and that there is adequate storage.
POWER SPOOL RETRIEVAL FAILURE: ATTEMPT NUMBER: request

Reason:
For VSE/POWER only. A POWER spool retrieval failed. This message identifies the request. If only one job gets this error, the JCL for the job is probably incorrect. If all jobs get this error, the product may not have been properly installed and customized.

Action:
Check the JCL and, if necessary, the installation and customization.

INCOMPLETE COMPILER OUTPUT DUE TO ERRORS HAS CAUSED TERMINATION OF IN25SYMP OPTION CAUSING PL/1 COMPILER TERMINATION WAS: option

Reason:
The PL/I compiler options NOSYNTAX and NOCOMPILE can prevent the normal completion of the compilation process, either conditionally, depending on the severity of the errors, or unconditionally. IN25SYMP terminates immediately when it recognizes an aborted compilation.

Action:
Usually, correction of compilation errors resolves this problem.

TABLE SIZE EXCEEDS HALFWORD LIMIT FOR: name

Reason:
An internal table has exceeded the limit of the halfword index. The postprocessor program terminated to prevent subsequent errors. This situation should not occur for programs with significantly less than 32,767 statements. Because support is limited to programs with less than 10,000 statements, this situation is unlikely to occur.

Action:
Contact CA Technical Support.
SYMP924S

SUBSCRIPT OVERFLOW IN TABLE T7

Reason:
The internal table used to keep track of variable displacements has exceeded its limits.

Action:
This can occur if the PL/I compile abnormally terminates or terminates with a return code greater than eight. Check these conditions and if they occur, correct them and rerun the job. If neither has occurred, contact CA Technical Support with the complete job output.

SYMP925S

SUBSCRIPT OVERFLOW IN TABLE T10

Reason:
The internal table used to keep track of the pseudo register information for controlled variables has exceeded its limits.

Action:
This error is usually caused when the Attribute and Cross Reference section of the compile output could not be found by IN25SYMP. This can occur if the compiler options ATTRIBUTE(FULL) and XREF(FULL) were not specified, or the PL/I compiler abnormally terminated or terminates with a return code greater than eight. Check these conditions and if they occur, correct and rerun the job. If neither has occurred, contact CA Technical Support with the complete job output.

SYMP926S

SUBSCRIPT OVERFLOW IN TABLE T11

Reason:
The internal table used to keep track of external symbol dictionary entries has exceeded its limits.

Action:
This can occur if the compiler option ESD was not specified, or the PL/I compiler abnormally terminated or terminates with a return code greater than eight. Check these conditions and if they occur, correct and rerun. If neither has occurred, contact CA Technical Support with the complete job output.
UTIL Messages

SYMP999S

UNEXPECTED ERROR CODE. nnn PROCESSING TERMINATED

Reason:
An undocumented error code has forced termination of IN25SYMP.

Action:
Contact CA Technical Support.

UTIL Messages

UTIL001

X....X

Reason:
This message is an echo of the input request. The request is indicated by x....x.

Action:
None.

UTIL002

INITIALIZATION COMPLETED

Reason:
The symbolic file has been initialized.

Action:
None.

UTIL003

XXXXXXXXX DELETED FROM SYMBOLIC FILE

Reason:
Program xxxxxxxx was deleted from the symbolic file.

Action:
None.
**UTIL004**

**XXXXXXX UNLOADED FROM SYMBOLIC FILE**

**Reason:**
Program xxxxxxxxx has been unloaded from the symbolic file.

**Action:**
None.

**UTIL005**

**XXXXXXX RELOADED TO SYMBOLIC FILE**

**Reason:**
Program xxxxxxxxx was reloaded to the symbolic file.

**Action:**
None.

**UTIL006**

**XXXXXXX RELOADED TO SYMBOLIC FILE AND HAS BEEN RENAMED TO YYYYYYYY**

**Reason:**
Program xxxxxxxxx was reloaded to the symbolic file with the name yyyyyyyy.

**Action:**
None.

**UTIL007**

**UNLOAD PROCESSING COMPLETED**

**Reason:**
The UNLOAD function completed.

**Action:**
None.
UTIL008

**RELOAD PROCESSING COMPLETED**

**Reason:**
The RELOAD function completed.

**Action:**
None.

UTIL009

**PURGE PROCESSING COMPLETED - XXXXX RECORDS HAVE BEEN FREED**

**Reason:**
The PURGE function completed and freed up the number of records indicated by XXXXX.

**Action:**
None.

UTIL010

**DEVICE NOW CLOSED**

**Reason:**
The device, specified by a CLOSE function, closed. Any subsequent requests for this device cause that device to open at load point.

**Action:**
None.

UTIL011

**INTERTEST BATCH UTILITY RUN COMPLETED SUCCESSFULLY**

**Reason:**
All requested functions have been performed and the symbolic file was updated successfully.

**Action:**
None.
UTIL012

RELOAD PROCESSING STARTED FOR PROGRAM XXXXXXXX

Reason:
The RELOAD function began for program xxxxxxxx.

Action:
None.

UTIL047

XXXXXXXX MUST BE DELETED

Reason:
While processing program xxxxxxxx, a condition occurred that indicated that data for the program was corrupted.

Action:
Delete this program.

UTIL048

XXXXXXXX CANNOT BE UNLOADED

Reason:
Program xxxxxxxx was found to be corrupted and could not be unloaded.

Action:
None.
XXX has zero records - Entry bypassed

Reason:
Program xxxxx was found to be corrupted.

Action:
Complete the following steps:
1. Delete the program from the symbolic file.
2. Using the UTILITY job, run an UNLOAD=ALL request.
3. Using the UTILITY job, run an INITIALIZE request.
4. Using the UTILITY job, run a RELOAD=ALL request, using the file created in step 2 as input.
5. Resubmit the original job.

Password missing - Request ignored

Reason:
The function requested requires a PASSWORD parameter card. Only this request is ignored.

Action:
Add a PASSWORD parameter card as the first parameter card in the job stream and resubmit the job.

Password incorrect

Reason:
The password provided on the PASSWORD parameter card does not match the password that was generated by the SYMPSWD parameter in IN2SOPT. The job is terminated.

Action:
Change the password specified by the PASSWORD parameter card to match the generated password and resubmit the entire job stream.
UTIL052

INVALID REQUEST

Reason:
The function requested is invalid. This condition may be caused by a misspelled option or invalid input. The job is terminated.

Action:
Correct the requested function and resubmit the entire job stream.

UTIL053

XXXXXXXXX CANNOT FIT ON FILE - PROGRAM IS BYPASSED

Reason:
The symbolic file did not contain enough free space to handle a RELOAD function for program xxxxxxxx. The program is bypassed.

Action:
Complete the following steps:
1. Using the UTILITY job, run an UNLOAD=ALL request.
2. Delete the symbolic file using IDCAMS.
3. Run an IDCAMS DEFINE for the symbolic file with a larger space allocation. Remember that a secondary space allocation is not permitted.
4. Using the UTILITY job, run an INITIALIZE request.
5. Using the UTILITY job, run a RELOAD=ALL request, using the file created in step 1 as input.
6. Resubmit the original job.

Or:
1. Using the UTILITY job, run a PURGE request to free up space.
2. Resubmit the original job.
UTIL054

PURGE INTERVAL INVALID OR MISSING - REQUEST NOT PROCESSED

Reason:
The PURGE request did not specify a number of days or the number of days
specified was not within the range of 1 through 365. This request is ignored.

Action:
Correct the PURGE request and resubmit the job.

UTIL055

PROGRAM NOT FOUND IN FILE OR WAS NOT USABLE

Reason:
The program specified for a requested function was not found in the symbolic file
or was unusable. This request is ignored.

Action:
Run a REPORT function. If the program is found, delete it and resubmit the
original job.

UTIL056

PROGRAM NAME IS GREATER THAN 8 CHARACTERS LONG - REQUEST IGNORED

Reason:
The program specified for a requested function contained more than eight
characters. This request is ignored.

Action:
Correct the requested function and resubmit the job.
STDOUT Messages

UTIL057

SYMBOLIC FILE IS EMPTY - PLEASE RUN INITIALIZATION AS FIRST STEP

Reason:
The symbolic file did not contain the required control records. The job is terminated.

Action:
Using the UTILITY job, run an INITIALIZE request.

UTIL058

VSAM RECORD LENGTH NOT = 2040

Reason:
The symbolic file was created with a wrong record size. The job is terminated.

Action:
Follow the instructions in the chapter "Creating a PROTSYM File" for creating a PROTSYM file.

UTIL059

NO LISTER INFORMATION FOR THIS PROGRAM - PRINT= REQUEST IGNORED

Reason:
The program specified by the PRINT request did not contain any saved source. This request is ignored.

Action:
None.

UTIL060

REQUEST FOR VIRTUAL STORAGE FAILED

Reason:
The request for GETMAIN or GETVIS storage failed. The job is terminated.

Action:
Resubmit the entire job stream, using a bigger region size or run the job in a larger partition.
UTIL061

**OPEN FAILURE FOR UNLOAD DEVICE**

**Reason:**
The open request for the device to be used for an UNLOAD function failed.

**Action:**
Correct the JCL and resubmit the job.

UTIL062

**OPEN FAILURE FOR RELOAD DEVICE**

**Reason:**
The open request for the device to be used for a RELOAD function failed.

**Action:**
Correct the JCL and resubmit the job.

UTIL063

**UNLOAD=ALL HAS BEEN RUN - ALL OTHER UNLOAD REQUESTS ARE IGNORED**

**Reason:**
An UNLOAD request has been made after an UNLOAD=ALL request. This request is ignored.

**Action:**
Resubmit the UNLOAD request which was rejected in a separate job stream.

UTIL064

**INPUT FROM RELOAD FILE IS INVALID**

**Reason:**
Data being retrieved for a RELOAD request was not in the correct format. This condition may be caused by invalid JCL or by data that was overlaid since its creation by the UNLOAD request. This request is ignored.

**Action:**
If invalid JCL caused the error, correct the JCL and resubmit the request. If invalid data caused the error, the problem is not correctable.
UTIL065

RELOAD FILE IS EMPTY

Reason:
An end-of-file condition occurred on the first read from the RELOAD device. This condition occurs when the file is empty. This request is ignored.

Action:
If invalid JCL caused the error, correct the JCL and resubmit the request; otherwise, the problem is not correctable.

UTIL066

XXXXXXX ALREADY EXISTS IN FILE: RELOAD FOR THIS PROGRAM IGNORED

Reason:
The program to be reloaded already exists on the file. This request is ignored.

Action:
Delete the program, using a DELETE request, and resubmit the RELOAD request.

UTIL067

xxxxxxx yyyy ERROR R15 = X’rr’ ERROR CODE = X’ee’

Reason:
A VSAM error occurred while running the UTILITY program. The message contains the following information:
- xxxxxxx is the name of the symbolic file.
- yyyy is the type of request (OPEN, GET, or PUT).
- rr is the return code, in hexadecimal.
- ee is the error code, in hexadecimal.

The job may or may not be terminated, depending on the function requested and when the error occurred.

Action:
Using the information from the message, find the explanation of the error in the VSAM manual that contains the error messages. Handle the error as described in your manual.
UTIL068

SEQUENCE NUMBER NOT FOUND

Reason:
While processing a request, an internal record key was not found. This condition indicates a corrupted file.

Action:
Delete the program that caused the problem.

UTIL070

RECORD COUNT ERROR AT ENDREQ

Reason:
An internal check of the file has failed. This condition indicates a corrupted file. The job is terminated with a dump.

Action:
Contact CA Technical Support for assistance.

UTIL071

UNLOAD OR RELOAD NOT SPECIFIED

Reason:
A CLOSE request was made but did not specify UNLOAD or RELOAD. The job is terminated.

Action:
Correct the CLOSE request and resubmit the entire job stream.

UTIL072

ERROR OCCURRED WHILE READING RELOAD DEVICE (DOS)

Reason:
The system detected an error condition while reading a record from the device pointed to by SYS005. The job is terminated.

Action:
Determine the cause of the error, correct the problem, and resubmit the entire job stream.
UTIL073

ERROR OCCURRED WHILE WRITING TO UNLOAD DEVICE (DOS)

Reason:
The system detected an error condition while writing a record to the device pointed to by SYS005. The job is terminated.

Action:
Determine the cause of the error, correct the problem, and resubmit the entire job stream.

UTIL074

ERROR OCCURRED WHILE READING PARAMETER CARDS (DOS)

Reason:
The system detected an error condition while reading a control card. The job is terminated.

Action:
Determine the cause of the error, correct the problem, and resubmit the entire job stream.

UTIL075

ENQ ERROR: CODE = X’yy’ (DOS)

Reason:
An error occurred while issuing a SVC 63 (lock) request. The job is terminated.

Action:
Contact CA Technical Support for assistance.

UTIL076

REQUESTED DATA SPACE EXCEEDS MAXIMUM FOR FILE

Reason:
The symbolic file has been defined with a size that exceeds its capacity. The maximum size of this file is about 4,000,000 2 KB records. The job is terminated.

Action:
This error is usually caused by defining the file with a secondary space allocation. If this is the case, redefine the file without a secondary allocation and initialize it.
**UTIL077**

**INTERTEST BATCH UTILITY RUN UNSUCCESSFUL**

**ALL UPDATES HAVE BEEN BACKED-OUT**

**Reason:**
This message is produced on any error condition that terminates the job. If this message is produced, all requested functions, even if they were correct, are backed out.

**Action:**
After the error condition is corrected, you must resubmit the entire job stream.

**UTIL078**

**INTERTEST BATCH UTILITY RUN UNSUCCESSFUL**

**UPDATES HAVE NOT BEEN BACKED-OUT**

**Reason:**
This message is produced when an error condition, which is not serious, has occurred. If this message is produced, only the requests that were in error are ignored and all valid requests were performed.

**Action:**
Review the output for any previous error messages, resolve those errors, and try again.

**UTIL079**

**PROGRAM NOT FOUND IN RELOAD FILE - REQUEST IGNORED**

**Reason:**
The program specified by a RELOAD request was not found in the reload file.

**Action:**
None.
**UTIL Messages**

**UTIL080**

**VSAM CI SIZE NOT = 2048**

**Reason:**
The symbolic file was created with the wrong CI size.

**Action:**
Recreate the symbolic file with the correct CI size.

**UTIL095**

**CHAIN TO NEXT DIRECTORY IS CORRUPTED**

**UTIL096**

**MISMATCH ON TOTAL AVAILABLE FREE RECORDS**

**UTIL097**

**MISMATCH ON AVAILABLE FREE RECORDS IN ONE SAM**

**UTIL098**

**INCORRECT NUMBER OF DIRECTORIES ON INPUT**

**UTIL099**

**INCORRECT NUMBER OF DIRECTORIES ON OUTPUT**

**Reason:**
Messages UTIL095 through UTIL099 are produced by the special checkout procedure. This procedure checks for a corrupted symbolic file.

If a corrupted file is found at the start of the job, one of the above messages displays. If a corrupted file is found at the end of the job, one of the above messages displays, an attempt is made to back out all updates, and the IN25UTIL program abends.

**Action:**
In response to these messages, try the following procedures before contacting CA Technical Support:

1. Run the IN25UTIL program to unload the existing data.
2. Recreate the symbolic file from scratch, using IDCAMS.
3. Run the IN25UTIL program to initialize the symbolic file.

4. Run the IN25UTIL program to reload the symbolic file with the saved data. At this point, the symbolic file should be repaired. If the problem still exists, contact CA Technical Support.

**UTIL100**

**WARNING** PROTSYM IS NOT COMPATIBLE WITH INTERTEST RELEASES BELOW 5.4

**Reason:**
You are attempting to use a r5.4 symbolic file with an earlier CA InterTest release. The new, larger symbolic files can only be used with r5.4 and higher; they are not downward compatible with earlier CA InterTest releases.

**Action:**
For CA InterTest releases prior to r5.4, only use symbolic files created with a pre-5.4 release.

**UTIL101**

INVALID RECORD ON FILE. UNLOAD AND RELOAD OF FILE RECOMMENDED

**Reason:**
An invalid record has been encountered on the PROTSYM during a utility function.

**Action:**
Reorganize your PROTSYM by doing an UNLOAD=ALL followed by a RELOAD=ALL to remove the bad record.
Index

A
Assembler • 39, 53
   IN2SSYMA postprocessor program • 39
   IN2SSYM postprocessor program • 53
assemblers, supported • 9

B
batch utility program, IN25UTIL • 57

C
CA Endevor SCM, IN2SSYM postprocessor program • 53
CA InterTest for CICS, IN25LINK postprocessor program • 46
CA Librarian, IN2SSYM postprocessor program • 53
CA Optimizer, IN2SSYM postprocessor program • 18
CA Optimizer/II, IN25COB2 postprocessor program • 25
CA Panvalet, IN2SSYM postprocessor program • 53
CAVICOB • 29
CAVHLINK • 50
CAVHPROT • 13
CAVHSYMA • 43
CAVHSYC • 22
CAVHSYMP • 36
CAVHUTIL • 57
COBOL • 25, 53
   IN25COB2 postprocessor program • 25
   IN2SSYM postprocessor program • 53
compiler options • 21, 28, 35
   CA Optimizer programs • 21
   CA Optimizer/II programs • 28
   COBOL II programs • 28
   OS/VS COBOL programs • 21
   PL/I programs • 35
compilers, supported • 9
composite module • 46, 47
   NOPURGE • 47
   testing subroutines • 46
composite name • 47
Composite Support feature • 46
CUTPRINT option • 19, 26, 33, 40
   IN25COB2 • 26
   IN2SSYMA • 40
   IN2SSYM • 19
   IN2SSYMP • 33

D
DELETE • 58
deleting symbolic data • 60
DEQ macro • 11

E
ENQ macro • 11
EXCLUDE • 48

I
IBM Linkage Editor • 46
IN25COB2 • 25, 26, 29, 30
   adding to COBOL procedure • 30
   CAVICOB JCL member • 29
   options • 26
   using • 25
IN25LINK • 46, 47, 48, 50
   adding to link-edit procedure • 50
   CAVLINK JCL member • 50
   example • 48
   excluding subroutines • 48
   options • 47
   using • 46
IN25SPARM program • 18, 26, 33, 39
   IN25COB2 • 26
   IN2SSYMA • 39
   IN2SSYM • 18
   IN2SSYMP • 33
IN25SOPT macro, PASSWORD • 58
IN2SSYMA • 39, 43, 44
   adding to Assembler procedure • 44
   CAVSYMA JCL member • 43
   options • 39
   using • 39
IN2SSYM • 18, 22, 23
   adding to COBOL procedure • 23
   CAVSYMC JCL member • 22
   options • 18
   using • 18
IN2SSYM postprocessor program • 53, 54
assemblers • 9
compilers • 9
SYM messages • 63
symbolic file • 58, 59, 60, 61, 62
  initializing • 59
  password • 58
  reloading programs • 61
  reorganizing • 62
  unloading programs to tape • 60
symbolic information • 11, 17, 59, 60
  adding to PROTSYM files • 17
  deleting • 60
  loading • 11
  postprocessor programs • 17
  purging • 59
symbolic support • 8
  optimized applications • 8
symbolic support, description • 7
T
testing composite modules • 46
U
UNLOAD • 58
unloading programs to tape • 60
UTIL messages • 105