Problem Determination Tools for z/OS

Common Component Customization Guide and User Guide

Version 1 Release 7

SC19-4159-05

Problem Determination Tools for z/OS

IBM

Common Component Customization Guide and User Guide

Version 1 Release 7

SC19-4159-05

Note

Before you read this document, look at the general information under "Notices" on page 109.

Edition notice

| This edition, which is published in December 2016, applies to Version 1 Release 7 Modification Level 0 of IBM Problem Determination Tools for z/OS Common Component (program number 5655-IPV), and to Version 13 Release 1 Modification Level 0 of IBM File Manager for z/OS (program number 5655-Q12), IBM Fault Analyzer for
| z/OS (program number 5655-Q11), IBM Debug for z Systems (program number 5655-Q10), IBM Application Performance Analyzer for z/OS (program number 5697-Q49), and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. For information on how to send comments, see "How to send your comments to IBM" on page vi.

This publication is available on the Web at:

www.ibm.com/software/awdtools/filemanager/

Contents

Preface

This document provides information for installing, configuring, and using the IBM® Problem Determination Tools Common Component server.

Who should use this document

This document is intended for those persons responsible for installing and using the IBM Problem Determination Tools Common Component server, and assumes a working knowledge of:

- z/OS® operating system
- system programming
- configuration of servers

Terminology used in this document

In this document, the IBM Problem Determination Tools Common Component server is referred to as the "PD Tools Common Component" server, or "PDTCC" server.

The following names are used in this book:

IBM Application Delivery Foundation for z Systems (ADFz) family of products
Previously known as IBM Problem Determination Tools (PD Tools) products.

IBM Debug for z Systems (zDebug)
Previously known as IBM Debug Tool for z/OS (Debug Tool).

Summary of changes

Sixth Edition (SC19-4159-05)

This edition of the document provides information applicable to IBM Problem Determination Tools for z/OS Common Component Version 1 Release 7. Here are the major changes to this document from the previous edition.

- Updated product names to reflect current offerings. For more information, see "Terminology used in this document."
- Added File Manager Remote Services to the list of products that use the Common Server. For more information, see "1. Common Server" on page 1.
- The updated chapter Chapter 8, "IPVLANGO Automatic Binary Optimizer LANGX file update utility," on page 83.

Fifth Edition (SC19-4159-04)

- A new feature PDTCC event processing is added. With the PDTCC event processing feature, any products or systems including ADFz family of products can send data to an asynchronous installation-written back-end for processing. For more information, see Chapter 10, "PDTCC event processing," on page 87.
- A new option EventProcessingExit is added, which is used to define an exit for the PDTCC event processing feature. For more information, see "EventProcessingExit" on page 13.

Fourth Edition (SC19-4159-03)

This edition provides information about how to maintain PD Tools Common Component. For more information, see Chapter 9, “Maintaining PD Tools Common Component,” on page 85.

Third Edition (SC19-4159-02)

This version of the book contains minor clarifications and corrections, and also the following significant changes relative to the previous edition:

These new chapters:
- Chapter 6, “IPVLANGX compiler listing to side file conversion utility,” on page 69
- Chapter 7, “IPVLANGP side file formatting utility,” on page 75
- Chapter 8, “IPVLANGO Automatic Binary Optimizer LANGX file update utility,” on page 83

These, and other minor changes, are indicated by a “|” changebar in the left margin of the page.

Second Edition (SC19-4159-01)

This edition incorporates the changes that were introduced by PI14699 and PI15084. It also acknowledges the integration of the common server, such that one server is referenced, instead of individual servers for each product.

As well, “z/OS XL C and C++ programs” on page 55 has been reworked, to provide more accurate information about C and C++ parameters.

First Edition (SC19-4159-00)

This edition of the document provides information applicable to Problem Determination Tools Common Component Version 1 Release 7. Here are the major changes to this document from the previous edition, for Version 1 Release 6, SC19-3690.

- New sections “Update sample IPVCONFG” on page 9, “Create matching WORKDIR by running job IPVMKDIR” on page 10, and “Check address space timeout” on page 10.
- The new major topics Chapter 4, “Options,” on page 13 and Chapter 5, “Quick start guide for compiling and assembling programs for use with IBM Application Delivery Foundation for z Systems family of products,” on page 17.

The source files for this document were migrated from one source type to another. As a result of this migration, there are minor formatting adjustments and other changes. These changes, and other small changes such as minor editorial clarifications, are not listed.

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of the following methods to send us your comments:

1. Send an email to comments@us.ibm.com
2. Use the form on the Web at:

   www.ibm.com/software/ad/rcf/
3. Mail the comments to the following address:

   IBM Corporation
   DTX/E269
   555 Bailey Avenue
   San Jose, CA
   95141-1003
   U.S.A.

Include the following information:
- Your name and address
- Your email address
- Your telephone or fax number
- The publication title and order number:
  PDTCC V1R7 Customization Guide and User Guide
  SC19-4159-05
- The topic and page number that is related to your comment
- The text of your comment.

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

IBM or any other organizations use the personal information that you supply only to contact you about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed in the previous topic. Instead, do one of the following actions:
- Contact your IBM service representative
- Call IBM technical support
- Visit the IBM support portal at http://www.ibm.com/systems/z/support/

Chapter 1. Introduction to IBM Problem Determination Tools Common Component

The IBM Problem Determination Tools Common Component has three major features, which are shared by the IBM Application Delivery Foundation for z Systems family of products:

1. Common Server
2. IPVLANGX, IPVLANGP, and IPVLANGO
3. Interactive Panel Viewer

From now on, the IBM Problem Determination Tools Common Component is referred to as "PD Tools Common Component" or "PDTCC".

1. Common Server

The Common Server is an extensible server program that runs on a z/OS system to serve clients. Multiple clients can connect to a single instance of the server program and request a service by invoking a specific extension of the server. The server needs to be customized to install various extensions. Without installing the extensions, the Common Server program alone does not serve any purpose.

The following products use the Common Server:

zDebug DTSP plug-in for Eclipse
See *IBM z/OS Debugger Customization Guide V14.0 (5724-T07)* for details on customization.

Fault Analyzer plug-in for Eclipse
See *Fault Analyzer for z/OS User's Guide and Reference V13.1 (SC19-4116)* for details on customization.

File Manager plug-in for Eclipse
See *File Manager for z/OS Customization Guide V13.1 (SC19-4118)* for details on customization.

File Manager for CICS® V13.1.
See *File Manager for z/OS Customization Guide V13.1 (SC19-4118)* and *File Manager for z/OS User's Guide and Reference for CICS V13.1 (SC19-4122)* for details on customization.

File Manager Remote Services
See *File Manager for z/OS Customization Guide V13.1 (SC19-4118)* "Preparing for File Manager Remote Services" for details on customization.

Application Performance Analyzer plug-in for Eclipse
See *Application Performance Analyzer for z/OS Customization Guide V13.1 (SC14-7598)* for details on customization.

For more information about configuring the product-specific extensions to the Common Server, see the product-specific customization guide.

2. IPVLANGX, IPVLANGP and IPVLANGO

IPVLANGX, IPVLANGP and IPVLANGO provide utility programs that undertake various functions.

Currently, the following products use one or more of these utilities:
- Fault Analyzer for z/OS
- IBM Debug for z Systems
- Application Performance Analyzer for z/OS

Here is more information about each utility:

IPVLANGX
A utility program that converts a compiler listing, or SYSADATA file, to a special format ADFz side file, in the remainder of this document referred to as a "LANGX side file", or simply a "LANGX file". A LANGX side file is typically a lot smaller in size than a compiler listing. (See Chapter 6, "IPVLANGX compiler listing to side file conversion utility," on page 69 for details.)

IPVLANGP
A utility program that creates a readable listing from a LANGX side file, a SYSDEBUG side file generated by using the COBOL or PL/I TEST(SEPARATE) option, or a COBOL program object containing DWARF debugging information generated by the TEST(SOURCE) option (see Chapter 7, "IPVLANGP side file formatting utility," on page 75 for details).

This listing might be useful if side files, rather than compiler listings, are kept in order to conserve DASD space. The utility program is able to format the side file in a way that resembles the original compiler listing.

IPVLANGP also supports the setting of zDebug Deferred Breakpoints.

IPVLANGO
A utility program used to create new LANGX side files to support the Automatic Binary Optimizer (see Chapter 8, "IPVLANGO Automatic Binary Optimizer LANGX file update utility," on page 83 for details).

3. Interactive Panel Viewer

The Interactive Panel Viewer feature enables ISPF-based applications to display panels under CICS. The following products use the Interactive Panel Viewer feature:

Fault Analyzer for z/OS
See *Fault Analyzer for z/OS User's Guide and Reference V13.1 (SC19-4116)* for details on customization.

File Manager for CICS V13.1
See *File Manager for z/OS Customization Guide V13.1 (SC19-4122)* for details on customization.

Chapter 2. Server overview

The PDTCC server runs a process that identifies a connection request on a specific port. The PDTCC server can be started manually, or during an IPL, by running a customized procedure. A sample procedure, IPVSRV1, is supplied in the sample library hlq.SIPVSAM1.

Multiple servers might be simultaneously run, provided different port numbers are used for each server.

For participating products that use the PDTCC server, the server negotiates SSL encrypted communications if configured to do so, then verifies the client user ID, password, or passphrase. If valid, the server creates a new process for that user.

The PDTCC server consists of a main program module, IPVSRV, and supporting message and API-related modules: IPVSEND, IPVRECV, IPVRXRCV, IPVRXSND, IPVSRVTE, IPVSRVSL, IPVSRVRF, IPVSRVSF, IPVMSGT, IPVCMENU, IPVVRFY, IPVCMJPN, IPVCTRC, IPVTRACE and IPVCMKOR.

IPVSRV requires a parameter string **'port family trace'** where:

port Describes the port number that is used to bind and accept incoming connections.

family The addressing family to bind to. For example, `AF_INET`, or `AF_INET6`.

trace N, T, D, U, or omitted. This parameter specifies the level of tracing to be performed by the server, and is intended only for diagnostic purposes. N is for no tracing, while T or D produce IPVTRACE, or STDOUT, outputs of undocumented messages that show flow and processing details for diagnostic purposes. U produces trace entries showing user connections to participating ADFz products.

Sample server procedure

The PDTCC server is recommended to run as a started task, although it might be run as a job.

A sample procedure, IPVSRV1, is supplied in the hlq.IPVSAM1 data set. Copy the procedure to your procedure library.

```
//IPVSRV1  PROC PORT=2800,FAMILY='AF_INET',TRACE=N
//*******************************************************************
//*   IBM Problem Determination Tools Common Components             *
//*   Release 7                                                     *
//*                                                                 *
//*   Licensed Materials - Property of IBM                          *
//*                                                                 *
//*   5655-W68                                                      *
//*                                                                 *
//*       Copyright IBM Corp. 2006, 2012.                           *
//*   All Rights Reserved.                                          *
//*                                                                 *
//*   US Government Users Restricted Rights - Use,                  *
//*   duplication or disclosure restricted by GSA ADP               *
//*   Schedule Contract with IBM Corp.                              *
//*                                                                 *
//*******************************************************************
```

```
//* FAMILY=AF_INET|AF_INET6   for TCP/IP V4 or V6 socket and bind
//* TRACE=N|D|U   No server trace, detailed trace or
//* user connection trace
//*
//* This is not a complete JCL procedure. It requires customisation
//* steps before running. To customise,
//* 1. Customise the IPVCONFG member
//* 2. Customise and run the IPVMKDIR sample job to match
//* 3. replace IPV with your high level qualifier for the PDTCC product
//* 4. Uncomment and replace CEE for your hlq for the LE C runtime
//*    if SCEERUN is not in the site linklist
//*
//RUN      EXEC PGM=IPVSRV,REGION=40M,
//            PARM=('&PORT &FAMILY &TRACE')
// SET IPV=IPV                                  >== Update HLQ
//* Common component authorised library
//STEPLIB  DD DISP=SHR,DSN=&IPV.SIPVMODA        >== PDTCC APF LIBRARY
//*        DD DISP=SHR,DSN=CEE.SCEERUN          >== LE C RUNTIME
//SYSPRINT DD SYSOUT=*
//IPVTRACE DD SYSOUT=*                          >== OUTPUT if Tracing
//STDOUT   DD SYSOUT=*
//* Server wide, then participating product configurations
//CONFIG   DD DISP=SHR,DSN=&IPV.SIPVSAM1(IPVCONFG)
```

Startup, shutdown, and activity tracing

The START procname operator command can be used to start the server.

To stop the server, the P procname operator command can be used.

To enable activity tracing, usually as an IBM support request, the following modify command can be used:

```
F procname,APPL=TRACEON
```

To disable activity tracing, the following modify command can be used:

```
F procname,APPL=TRACEOFF
```

To display the release and PTF level of the running server, the following modify command can be used:

```
F procname,APPL=VER
```

Configuration file keyword descriptions

The configuration data might contain line comments. Line comments begin with an * or a #, and continue to the end of the line.

CONFIG=*name*
: *name* is the name of the configuration as specified by the client. At least one configuration is expected with a name of DEFAULT. Other configuration keywords apply to the current CONFIG name, in top-down order.

WORKDIR=/path
: The CONFIG=DEFAULT set of parameters needs the WORKDIR=path keyword. This keyword specifies where the server can write semi-permanent (existing at least while the server task is running) files. A sample job, IPVMKDIR is supplied in the sample library to create this path.

SSL_REQUIRED=YES|TLSV1|TLSV1.1|TLSV1.2|NO (Optional, default is NO)
: Determines whether SSL encrypted communications are mandatory for the

server and the desired protocol level. SSL communications are achieved by utilising the System SSL APIs. The default protocol level is TLS V1.1 when YES is specified. Older clients (prior to common component client 13.1.0.16) require TLS V1.

To use TLS V1.2, clients must be at level 13.1.0.17 or later.

If SSL encryption is used, then the server uses a certificate stored in either a RACF® keystore, when specified via the SSL_KEYRING keyword, or a GSKKYMAN managed key database and certificate for this server as specified in the SSL_CERT keyword or, if that keyword is omitted, at the WORKDIR specified location.

SSL_CERT=/path/keyringfile (optional, for use of user created certificate)
The path and name of a key database that contains a stored certificate that is used by the server. This parameter is passed to the gsk toolkit as the GSK_KEYRING_FILE setting. If this parameter is omitted, the server attempts to create a key database and self-signed certificate as it starts up.

SSL_CERTPW=keyringpw (optional, for use of user created certificate)
The password to be used to access the certificate repository. If omitted, the server uses a default password.

SSL_KEYRING=userid/keyring
If SSL is being used for the server, this configuration option provides the userid and keyring name for a certificate being held in a SAF keyring. The userid should match the ID used when creating the keyring.

SSL_LABEL=labelstring (optional, for use of user created certificate)
The label of the certificate from the key database to be used.

SPAWN_ACCT=accountdata
Allows specification of the account data used for the spawned address space. This is as per the _BPX_ACCT_DATA environment variable discussed in the z/OS UNIX System Services Planning manual.

SPAWN_TIME=nn
Allows specification of the CPU time limit, in seconds, used for the spawned address space.

SPAWN_PROGRAM=PROGRAM
Specification of the program that is launched for the client connection. The server checks the existence of the named program. If you want to specify the name of a z/OS UNIX executable file, rather than a load module in a STEPLIB data set, include the path. Otherwise, the server creates a sticky bit file in the WORKDIR specified location. Sticky bit is the mechanism in the z/OS UNIX file system of indicating that this file is a load library member. The program is launched as a USS process, but can be a traditional z/OS program.

SPAWN_STEPLIB=steplib1:steplib2 (optional)
Allows specification of the run libraries that are used for the spawned address space. Support for continuing library specifications is provided by ending a line with the colon character.

SPAWN_PARMS_SECTION
This entry marks the beginning of extra parameters that are passed to the spawned process. The contents of this area are determined by the products that use the server.

Launching a TSO environment is provided for by the common server when the SPAWN_PROGRAM is set to IPVSRVTE. In such a configuration, the launched process deals with these extra keywords that follow the SPAWN_PARMS_SECTION:

SPAWN_DD=ddname=datasetname1:datasetname2
: Specification of a data set or data sets that are allocated with DISP=SHR to the supplied DD name.

SPAWN_DD=ddname=SYSOUT=c
: Specification of a sysout allocation that is allocated with the specified class c, to the supplied DD name.

SOCKETFIONBIO
: Specification that the socket communications run in nonblocking mode.

  Specify this keyword only when the application for the particular CONFIG allows or expects it.

TSO_CMD=command;
: Specification of a command that is run in the TSO environment. This command typically instigates the launch of the participating products main serving function. This parameter can be repeated as needed for multiple TSO commands.

MIXEDCASEPASS=YES|NO (optional, default is NO)
: Determines whether uppercase translation is performed for incoming passwords for this system. If this system supports mixed case passwords, set this to YES and specify this keyword in the CONFIG=DEFAULT section.

SPAWN_REGIONSZ=nnn (optional, default is to inherit the region size of the server)
: Determines the region size (in MB) for the launched process. Participating products being launched have their own recommendations for this sizing.

Chapter 3. Customizing the PDTCC Server

This chapter provides you with instructions on how to customize the PDTCC Server. In brief, this consists of the following general checklist:

- APF authorize the SIPVMODA library
- Add programs in SIPVMODA to program control
- Add user for server started task
- Add task to STARTED class
- Add sample IPVSRV1 to system procedure library
- Permit server user/group to BPX.SERVER facility
- Permit server user/group to CSF* profiles (if used)
- Update sample IPVCONFG
- Create matching WORKDIR by running job IPVMKDIR
- Review address space timeout settings

Required Authorizations

The STEPLIB hlq.SIPVMODA must be APF-authorized.

Associate the started task that is used to run the Common Server with a user ID with an OMVS segment. Give the task READ access to the BPX.SERVER facility. Make sure write access to the z/OS UNIX directory is available, as specified by the WORKDIR= configuration parameter. Edit and run the job IPVMKDIR in the sample library (IPV.SIPVSAM1) to create this directory. Furthermore, any users logging in to the Common Server require read access to this location. Similarly, if you configure the Common Server to a key database of your own creation, the Common Server, and any users logging in to the Common Server, require read access to the specified key database.

Products that make use of the SPAWN_JOBNAME configuration keyword, require the user ID of the common server to be permitted to the BPX.SUPERUSER facility and to the BPX.JOBNAME facility if that is defined.

If enhanced program security is enabled, at a minimum the following programs must be defined to program control, unless BPX.DAEMON.HFSCTL was set up:

- IPVSRV
- IPVMSGT
- IPVCMENU
- IPVCMJPN
- IPVCMKOR
- UIPVMSGT
- IPV0LVL

To eliminate incorrect notifications about program control apply the fix for z/OS APAR OA39888 (or equivalent for your z/OS level).

Alternatively, define all common server programs in the library IPV.SIPVMODA to program control, rather than specifying individual programs.

If enhanced program security is enabled, IPVSRV must be defined with the MAIN attribute, using the APPLDATA operand on the PROGRAM profile.

Example commands for RACF

To activate program control if not already active, use the following command:

```
SETROPTS WHEN(PROGRAM)
```

To add all common server programs in a library to program control, use the following command:

```
RDEFINE PROGRAM IPV* ADDMEM('IPV.SIPVMODA'//NOPADCHK) UACC(READ)
```

In addition, the following command is required for alias member UIPVMSGT:

```
RDEFINE PROGRAM UIPVMSGT ADDMEM('IPV.SIPVMODA'//NOPADCHK) UACC(READ)
```

To add individual programs, use the following command:

```
RDEFINE PROGRAM IPVSRV ADDMEM('IPV.SIPVMODA'//NOPADCHK) UACC(READ)
```

To refresh, use the following command:

```
SETROPTS WHEN(PROGRAM) REFRESH
```

Note:

1. If you are using Japanese, then include the module IPVCMJPN in program control.
2. If you are using Korean, then include the module IPVCMKOR in program control.

If RACF, or an equivalent security product is implemented, the PDTCC Server (IPVSRV1) started task must also be defined to the STARTED class. For example, to add IPVSRV1 as an STC, the RACF commands in the example that is shown here could be used, where `ISPSRV1` is the name of your PDTCC Server procedure and *userid* is the userid that the started task runs under:

```
RDEFINE STARTED IPVSRV1.* STDATA(USER(userid))

SETROPTS RACLIST(STARTED) REFRESH
```

For more information about started tasks and security, see the z/OS Security Server RACF Security Administrator's Guide, or equivalent documentation for your security product.

Setting SSL encrypted communications

The sample IPVCONFG configuration file member has SSL encrypted communications active with the following line under the CONFIG=DEFAULT section:

```
SSL_REQUIRED=YES
```

If SSL encryption is not required in your environment, comment out this line and uncomment the next line (or alter your existing line to `SSL_REQUIRED=NO`).

If using a SAF keyring, uncomment and modify the SSL_KEYRING line. The SSL_LABEL line should also be uncommented and modified if the certificate you generate does not have a label of 'PDTCC Server Certificate'.

For use of a certificate in a keyring, the userid of the server task or job, as well as the userids connecting to the server need to be permitted UPDATE access to the

IRR.DIGTCERT.LISTING facility and CONTROL access to the IRR.DIGCERT.GENCERT facility in order to share the certificate amongst users of the common server.

For RACF users, a keyring and certificate could be created by the following example commands:

```
RACDCERT ID(IPVSRV) ADDRING(RINGA)
RACDCERT GENCERT SITE SIZE(1024)          -
         SUBJECTSDN(                      -
           CN('Common Server')            -
           OU('ADL')                      -
           O('ADL')                       -
           C('AU'))                       -
 WITHLABEL('PDTCC Server Certificate')
RACDCERT ID(IPVSRV)                                      -
         CONNECT(SITE LABEL('PDTCC Server Certificate')  -
         RING(RINGA) USAGE(PERSONAL)                     -
         DEFAULT)
SETR RACL REFR(DIGTCERT)
```

Note in the above that the userid IPVSRV is used for the userid of the common server task.

Updating the server config to include SSL_KEYRING=IPVSRV/RINGA would use the above generated certificate. These commands serve as a working example only and should be updated as desired to match your needs. RACDCERT commands are documented in the z/OS Security Server RACF Command Language Reference.

If you are using ICSF and have protected resources through the CSFSERV facility class, the server user or group id needs to be permitted to the resource, for example:

```
PERMIT  CSF*  CLASS(CSFSERV)
            ID(groupid)  ACCESS(READ)
```

For more details see the *Cryptographic Services ICSF Administrator's Guide*.

If you wish to specify a cipher string for the System SSL component to use, you can do this by modifying the server JCL to specify an `ENVAR(GSK_V3_CIPHER_SPECS=xx)` or `ENVAR(GSK_V3_CIPHER_SPECS_EXPANDED=xx)` as required. The sample server JCL member IPVSRV1 includes an example format of the above.

Update sample IPVCONFG

Update the sample configuration member to suit your site, according to the comments in that member. In general terms, review the following items in the config file:

- Alter SPAWN_DD=ddname=SYSOUT=C to suitable classes for your site. For example, for tracing activity, the CONFIG=DEFAULT section contains a SPAWN_DD=IPVTRACE=SYSOUT=H card that other configurations inherit and write trace output (if activated) to. Adjust this class to a class suitable for your site.
- Alter SPAWN_STEPLIB data set names to the installation high-level qualifiers for the relevant libraries. The SPAWN_STEPLIB statement is not required if all of the libraries are already in the linklist for your site.

- If a configuration makes use of the SPAWN_JOBNAME statement, then all address spaces that are launched for that connection type run with that specified jobname (the owner of each job reflects the user that is logged in).
- Do not alter CONFIG=name and SPAWN_PROGRAM=name values unless otherwise detailed in the participating product's documentation.

The configuration file supports the setting and reference of substitution variables in the following form:

```
$VAR=value
```

For setting these variables, specify the above form before the first CONFIG statement, or between the CONFIG and SPAWN_PARMS_SECTION statements. If using concatenations for the CONFIG DD, the first CONFIG refers to the statements in the first of the concatenations.

In following statements in the configuration, occurrences of '$VAR' are replaced by the 'value' specified. This could be used to represent high level qualifiers that are repeated in the configuration file. For example, set the value:

```
$IPVHLQ=SYS1.IPV
```

Then allow a reference in a following statement, such as:

```
SPAWN_STEPLIB=$IPVHLQ.SIPVMODA
```

The sample IPVCONFG makes use of this for high level qualifiers but it could also be used for other substitutions as desired.

Create matching WORKDIR by running job IPVMKDIR

This job can be found in the sample library and creates a directory to be used with the server. As can be seen in the supplied sample, the job creates a directory hierarchy in the form of

```
/etc/ipv/v17/ipvsrv1
```

You can alter this to suit your site, and the WORKDIR statement in the server configuration needs updating to the created directory. Do not use `/tmp` as a directory location.

As the files in the workdir need to be owned by the servers userid, and the IPVMKDIR job issues the chown command, the file system they are mounted at needs to allow the changing of userid via the SETUID attribute.

Check address space timeout

When an address space is launched for a client, and it has completed its current function, the address space is waiting for TCP/IP communications from the peer. In line with this, the client address space might be subject to an s522 abend if waiting longer than the active site settings for job wait time. The job wait time is controlled by the `JWT` parameter of the `SMFPRMxx` member, but might also be set to never time out by the site settings for `MAXCPUTIME` in the site's `BPXPRMxx` member. Set these parameters as needed by the site.

Add ports to TCPIP reservation list

Add the ports for the server, or servers, you want to run to the reserved port list in your TCPIP configuration data.

Configuration considerations for IBM Explorer for z/OS (z/OS Explorer)

The port number that is used by the ADFz server must be specified in the `rse.env` directive `PD_SERVER_PORT` statement as follows:

```
PD_SERVER_PORT=nnnn
```

where *nnnn* is the port number.

`rse.env` is located in `/etc/zexpl/`. For more details, see /etc/zexpl/rse.env.

Chapter 4. Options

For the IPVLANG utilities, you can specify installation-wide default options in the IPVCNF00 parmlib configuration member.

You can create a member IPVCNF00 in SYS1.PARMLIB, or any other data set that is part of the logical parmlib concatenation.

Note: If not providing general READ access to data sets in the logical parmlib concatenation, then an IPVOPTLM configuration-options module can be used to specify an alternative data set, as explained in "Using an IPVOPTLM configuration-options module" on page 14.

If you do not specify an option, then it takes either the product default (as indicated on the syntax diagram for each option), or has no value at all.

Options that are specified in the IPVCNF00 parmlib configuration member are subject to these syntax rules:
- Only columns 1 - 71 are processed.
- Options can be specified anywhere in a line. They do not have to start in column 1.
- You can use a blank or a comma as a delimiter.
- Options can be continued across any number of lines
- Options specifications are not case-sensitive—all options are converted to uppercase.
- Comments are permitted anywhere and can be nested. The characters "/*" identify the beginning of a comment, and "*/" identify the end.

Option descriptions

The following explains each option in detail.

EventProcessingExit

Use the EventProcessingExit option to define an exit that is to be invoked to perform asynchronous event processing. For more information, see Chapter 10, "PDTCC event processing," on page 87.

Syntax

```
►►──EventProcessingExit(exit-name)──────────────────────────────────►◄
```

exit-name
The name of an Event Processing user exit that contains an LE fetchable function of the same name. The maximum length of the name is 8 characters.

| If this option is changed, ADFz family of products that use the Event Processing user exit are affected.

Locale

Syntax

```
   ┌─NOLOCALE───────────┐
►►─┴─LOCALE(locale-name)─┴──────────────────────────────────────────►◄
```

The Locale option specifies the locale to be used for cultural environment-dependent presentation.

The locale name that is specified as *locale-name* can be one of those supplied with z/OS C/C++ for the setlocale() runtime function. A list of locale names can be found in the *z/OS C/C++ Programming Guide*, "Appendix D. Locales Supplied with z/OS C/C++".

Specifying the NoLocale option is the equivalent to specifying Locale(C).

The following are affected by the Locale option:

IPVLANGP source code comments
: Characters in source code comments which are considered non-printable given the current locale are shown as periods.

Fault Analyzer for z/OS
: Affected are things like date and time formatting, collating sequences of sorted information, and determination of non-printable characters which are shown as periods.

  Note: If used, the equivalent Fault Analyzer for z/OS Locale option overrides the IPVCNF00 Locale option specification.

Using an IPVOPTLM configuration-options module

An optional IPVOPTLM configuration-options module can be used to provide settings which are required before reading the IPVCNF00 parmlib member.

The name of the configuration-options load module must be IPVOPTLM, and it must be placed in an APF-authorized library in order to be used. Place the library in LNKLST so that the IPVOPTLM load module can be found. IBM recommends the use of IPV.SIPVMODA for this library.

A sample job to create an IPVOPTLM configuration-options load module is provided as member IPVOPTLM in data set IPV.SIPVSAM1.

Individual settings in the IPVOPTLM configuration-options module are specified using the IPVOPT macro, as explained in the sample job.

The following describes the currently only available setting.

IPVCNFDS
: To accommodate installations that do not provide general READ access to SYS1.PARMLIB (or any one of the data sets in the logical parmlib concatenation), an alternative data set can be specified as follows:

```
IPVOPT IPVCNFDS,dsname
```

where *dsname* is the name of the alternative parmlib data set.

The following example shows the specification of data set PDTOOLS.PARMLIB as an alternative parmlib data set name:

```
IPVOPT IPVCNFDS,PDTOOLS.PARMLIB
```

Using an IPVOPTLM configuration-options module

Chapter 5. Quick start guide for compiling and assembling programs for use with IBM Application Delivery Foundation for z Systems family of products

This chapter describes the minimal steps that are required to prepare your programs for use with IBM Application Delivery Foundation for z Systems family of products. For more detailed information, refer to 'Part 2. Preparing your program for debugging' of the *IBM z/OS Debugger User's Guide*, 'Part 2. Fault Analyzer Installation and Administration' of the *Fault Analyzer for z/OS User's Guide*, or Appendix B. of the *Application Performance Analyzer for z/OS User's Guide*.

The purpose of this chapter is to provide instructions for a single compile method for organizations that are using some combination of IBM Debug for z Systems, Fault Analyzer for z/OS, and Application Performance Analyzer for z/OS. If your enterprise is only using IBM Debug for z Systems, you can alternatively refer to 'Part 2. Preparing your program for debugging' of the *IBM z/OS Debugger User's Guide*. If your enterprise is only using Fault Analyzer for z/OS, alternatively refer to 'Part 2. Fault Analyzer Installation and Administration' of the *Fault Analyzer for z/OS User's Guide*. If your enterprise is only using Application Performance Analyzer for z/OS, alternatively refer to Appendix B of the *Application Performance Analyzer for z/OS User's Guide*.

IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS are designed to use load modules and other files that are produced by IBM compilers. You must compile your programs with certain compiler options so that they produce load modules and files that these products can use.

This chapter uses the term 'source information files' to refer to the types of files that are used by IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS. The different kinds of source information files that are the subject of discussion in this chapter include:

- SYSDEBUG files
- LANGX files
- Compiler listings
- DWARF files
- Expanded source files

Different compilers generate different kinds of source information files. If you use more than one compiler, you might have more than one type of source information library.

When you compile your programs with the compiler options described in this chapter, you can use the load modules and source information files that are created by the compilers as follows:

- Prepare the module for debugging (if you are using IBM Debug for z Systems). IBM Debug for z Systems lets you work with program statements and variables.

  When a program is compiled with the right options, the module that is produced by the compiler can be debugged and a source information file, which contains program statements, can be produced. When you use IBM Debug for z

Systems to debug a program, IBM Debug for z Systems uses the source information file to display the program source statements in the source window.

Depending on the source language and compiler that are used, the load module, the source information file, or the DWARF file contains information about statements and variables, such as offsets and lengths, and contains information that allows the debugger to locate statements and variables in storage. If you do not compile with the correct compile options, debugging is limited to something called 'disassembly' mode, where machine code is displayed, but no source statements or variables.

- Use Fault Analyzer for z/OS to automatically pinpoint the source statement that caused an abend, and can show you the values of variables in your programs at the time of an abend.
- Use Application Performance Analyzer for z/OS to show you precisely which program statements are using the most CPU time and wait time. Use this information to tune your applications.

Updating your build process

If someone recently installed one or more of the IBM Application Delivery Foundation for z Systems family of products on your system, the program build processes might not have been updated yet. Updating the build processes is an important and necessary part of implementing the IBM Application Delivery Foundation for z Systems family of products.

In many organizations, there is clear ownership of these build processes. In other organizations, it might not be obvious who makes the changes. Many organizations use standard compile processes or PROCs that your system administrators maintain and have updated to prepare programs for the IBM Application Delivery Foundation for z Systems family of products. In this case, find out what processes have been made available and how to use them. In other organizations, each developer maintains their own compile JCL or PROCs to compile programs. In this case, update your own compile JCL to prepare your programs for the IBM Application Delivery Foundation for z Systems family of products as described below.

Start by researching what is required for each compiler individually. For example, the changes that are required for Enterprise COBOL for z/OS, Enterprise PL/I for z/OS, C/C++ and Assembler are all slightly different.

In general, there are three changes that might be needed to compiler JCL to produce programs that can be used by the IBM Application Delivery Foundation for z Systems family of products:

1. Specify compiler options that are required by the IBM Application Delivery Foundation for z Systems family of products.
2. Code the JCL to produce and save the source information files that the IBM Application Delivery Foundation for z Systems family of products need. Newer compilers can generate the required source information files directly. Some older compilers require an extra step in the compile job to run a special utility program that produces the needed file.
3. In certain environments, it is advantageous to include a special IBM Debug for z Systems module into the application load module during the link-edit step. In most cases this special module is optional, but it can simplify starting IBM

Debug for z Systems for certain types of programs. For certain older compilers running in certain environments, you must include a special module to enable IBM Debug for z Systems.

Updating your promotion process

Typically, when a program is tested, program load modules are promoted through different stages before reaching production. For example, when a new program is compiled for the first time, it might be placed into a test load library. After unit testing is completed, perhaps the compiled program is promoted to a quality assurance environment. And eventually, it is promoted into production. On your system, you might know these stages by different names, such as:

- Unit test
- System test
- Model office

Consider whether you want the ability to use IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS throughout your programs' life cycle. Even if you do not plan to use IBM Debug for z Systems with production programs, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS are useful in those stages. To enable the IBM Application Delivery Foundation for z Systems family of products at each stage, update your promotion processes to retain the source information files. Promotions can be accomplished by performing a recompile, a copy, or a move. Perform the same steps with your source information files that you perform with your load modules or object modules. For each load library or object library, have a corresponding set of source information libraries. Whenever you promote a load module or object module, promote the source information file as well. This ensures that the source information file is available for Fault Analyzer and Application Performance Analyzer, and you can continue to take advantage of the IBM Application Delivery Foundation for z Systems family of products at all stages of your program's life cycle.

Preparing your programs

Each compiler produces different kinds of source information files, and each of the IBM Application Delivery Foundation for z Systems family of products reads different kinds of files. It can be time-consuming to research all the different combinations, but for each compiler, there is a suggested method that is described below. If you use the suggested methods, then your programs are ready to take full advantage of the IBM Application Delivery Foundation for z Systems family of products.

- "Enterprise COBOL for z/OS Version 5 programs" on page 20
- "Enterprise COBOL for z/OS Version 4 programs" on page 20
- "Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM programs" on page 24
- "COBOL for MVS and VM programs" on page 28
- "VS COBOL II programs" on page 31
- "OS/VS COBOL programs" on page 35
- "Enterprise PL/I Version 3.7 and later programs" on page 37
- "Enterprise PL/I Version 3.5 and Version 3.6 programs" on page 42
- "Enterprise PL/I Version 3.4 and earlier programs" on page 48
- "PL/I for MVS and VM and OS PL/I programs" on page 52

- "z/OS XL C and C++ programs" on page 55
- "Assembler programs" on page 64

Enterprise COBOL for z/OS Version 5 programs

The following table shows various compiler options that can be used to prepare Enterprise COBOL for z/OS Version 5 programs for use with theIBM Application Delivery Foundation for z Systems family of products (IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate whether the program object produced is suitable for a production environment. Program objects suitable for a production environment have no significant runtime overhead.

The table shows what is required for **full** function.

Table 1. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for Enterprise COBOL for z/OS Version 5

Compiler options	Source information file type produced	Is the program object production ready?	Options supported and suggested for IBM Debug for z Systems	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
TEST(SOURCE)	NOLOAD class in the object	Yes	Supported	Supported	Supported
NOTEST, LIST, MAP, SOURCE, NONUMBER, XREF(SHORT)	Listing	Yes	Not Supported	Not Supported	Supported

Enterprise COBOL for z/OS Version 4 programs

The following table shows various compiler options that can be used to prepare Enterprise COBOL for z/OS Version 4 programs for use with the IBM Application Delivery Foundation for z Systems family of products (IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate whether the load module produced is suitable for a production environment. Load modules suitable for a production environment have no significant runtime overhead.

Table 2. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for Enterprise COBOL for z/OS Version 4.

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for IBM Debug for z Systems	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
TEST (NOHOOK, SEPARATE, EJPD), LIST, MAP, SOURCE, NONUMBER, XREF(SHORT)	SYSDEBUG	Yes	Suggested for production and test		
NOTEST, LIST, MAP, SOURCE, NONUMBER, XREF(SHORT)	Compiler listing	Yes	N/A	Supported	Supported
NOTEST, LIST, MAP, SOURCE, NUMBER, XREF(SHORT)		Yes	N/A	Supported	N/A
LIST, MAP, SOURCE, NONUMBER, XREF(SHORT)	LANGX file	Yes	Not supported	Supported	Supported
LIST, MAP, SOURCE, NONUMBER, NOTEST, NOOPT, XREF	LANGX file	Yes	Supported	Supported	Supported

Preparing Enterprise COBOL for z/OS Version 4 programs

Perform the following steps for compiling your Enterprise COBOL for z/OS Version 4 programs using the compiler options suggested in Table 2:

1. Create libraries (PDSE is suggested unless PDS is required in your organization) for SYSDEBUG files. Create one or more SYSDEBUG libraries for each environment, such as test, and production.
2. Create a corresponding SYSDEBUG library for each load library. Specify `LRECL=(80 to 1024),RECFM=FB,BLKSIZE=(multiple of lrecl < 32K)`.
3. For all programs in both test and production environments, specify the following compiler options: `TEST(NOHOOK,SEPARATE,EJPD),LIST,MAP,SOURCE,NONUMBER,XREF(SHORT)`.

   The `TEST` compiler option is required if you plan to use IBM Debug for z Systems to debug a program. The `TEST` option is optional if you plan to use Fault Analyzer for z/OS or Application Performance Analyzer for z/OS.

   The `SEPARATE` suboption produces a SYSDEBUG file.

   `NOHOOK` and `SEPARATE` produce a production-ready module that can still be debugged.

If the OPT option is also used, EJPD might reduce optimization but enables the debugger's JUMPTO and GOTO commands. These commands are disabled when OPT and NOEJPD are both used.

4. When the TEST option is not used, save the compiler listing in a file, or use the IPVLANGX utility program to create a LANGX file. To minimize JCL changes, IPVLANGX has aliases to match names. These are:

   IBM Debug for z Systems
   EQALANGX

   Fault Analyzer for z/OS
   IDILANGX

   Application Performance Analyzer for z/OS
   CAZLANGX

   Fault Analyzer for z/OS and Application Performance Analyzer for z/OS can use compiler listings and LANGX files to provide source-level support.
5. The LIST, MAP, SOURCE, and XREF options are needed only if a compiler listing or a LANGX file is used to provide source information to Fault Analyzer for z/OS or Application Performance Analyzer for z/OS. If a SYSDEBUG file is used with these products or if you are not using Fault Analyzer for z/OS or Application Performance Analyzer for z/OS, the LIST, MAP, SOURCE, and XREF options are optional.
6. The NONUMBER compiler option is needed only if a compiler listing file is used to provide source information to Application Performance Analyzer for z/OS. If a SYSDEBUG file is used with Application Performance Analyzer for z/OS, or if you are not using Application Performance Analyzer for z/OS, the NONUMBER option is optional.
7. Code a SYSDEBUG DD in the JCL of the compiler step:
   ```
   //SYSDEBUG DD DSN= SYSDEBUG.pds(pgmname),DISP=SHR
   ```
   Save the SYSDEBUG file that is produced by the compiler in the SYSDEBUG library and specify a member name that is equal to the program name of your application program. This file is the source information file for IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.
8. Modify the promotion process to promote SYSDEBUG files. When a load module is promoted, for example from test to production, promote the corresponding SYSDEBUG file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the SYSDEBUG file that you perform with the module during promotion.
9. Optionally, include a zDebug Language Environment® (LE) exit module into the load module during the linkage editor step. This inclusion is one way to enable zDebug's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS™ batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2® stored procedures. Do not include the exit module for CICS programs.

   You can also use module EQAD3CXT for batch programs, IMS/TM, IMS BTS programs, and DB2 type MAIN stored procedures.

Sample JCL for compiling Enterprise COBOL for z/OS Version 4 programs

Here is a JCL example for compiling an Enterprise COBOL for z/OS Version 4 program for use with the IBM Application Delivery Foundation for z Systems family of products. This sample is a generic sample, and might not meet all your requirements to generate your modules.

Notice that the TEST compiler option is specified. Code the correct suboptions of the TEST compiler option for the version of the compiler that you are using. You can also code any other compatible compiler options that are required by your programs.

Also. notice that a SYSDEBUG DD statement was coded. This statement indicates the source information file that the compiler produces. It refers to a SYSDEBUG library that is a PDS or PDSE. The member name must be the same as the program name.

For Enterprise COBOL for z/OS, these are the only required changes.

However, there is an optional change in the linkage editor step. The example shows that a special Language Environment exit module is included in the application load module. Although this is exit module not required, it enables the use of zDebug panel 6, which makes the debugger easier to start in some environments. If you prefer to use panel 6 to start zDebug, this method is one way to enable it. If you do not plan to use zDebug panel 6, then do not include an exit module.

```
//*     - - -  ADD A JOB CARD ABOVE THIS LINE  - - -
//*
//*  SAMPLE JCL TO PREPARE AN ENTERPRISE COBOL PROGRAM
//*  FOR THE IBM ZSERIES PD TOOLS PRODUCTS:
//*     FAULT ANALYZER, DEBUG TOOL, AND APPLICATION PERF. ANALYZER
//*
//*  NOTES:
//*
//*   COMPILER:
//*    1. A TEST COMPILER PARM IS REQUIRED FOR DEBUG TOOL
//*    2. COMPILER PARM TEST(NOHOOK,SEPARATE,EJPD) HAS ADVANTAGES:
//*         - THE MODULE IS READY FOR DEBUG TOOL
//*         - THE MODULE IS PRODUCTION-READY (NO RUN-TIME OVERHEAD)
//*         - A SYSDEBUG FILE IS CREATED THAT CAN BE USED BY DT,FA,APA
//*    3. COMPILER PARMS LIST,MAP,SOURCE,XREF ARE REQUIRED IF YOU PLAN
//*       TO USE THE COMPILER LISTING WITH FA OR APA, OR IPVLANGX
//*
//*   BINDER (LINKAGE EDITOR):
//*    4. THE INCLUDE FOR MODULE EQAD?CXT IS *OPTIONAL*.  IT IS AN
//*       LE EXIT MODULE THAT CAN BE USED TO START DEBUG TOOL.
//*       UNDERSTAND THE METHODS AVAILABLE FOR STARTING DEBUG TOOL,
//*       AND CHOOSE WHETHER YOU WANT TO USE THE LE EXITS.
//*         IF YOU USE THIS METHOD, LOAD THE CORRECT EXIT MODULE:
//*            EQADBCXT: FOR BATCH PROGRAMS
//*            EQADICXT: FOR ONLINE IMS PROGRAMS
//*            EQADDCXT: FOR DB2 STORED PROCEDURES (OF TYPE MAIN AND SUB)
//*           (for SUB this is supported only for invocations through call_sub)
//*            (DO NOT INCLUDE AN EXIT FOR CICS PROGRAMS)
//*            YOU CAN ALSO USE MODULE EQAD3CXT FOR BATCH PROGRAMS, ONLINE IMS
//*            PROGRAMS, AND DB2 TYPE MAIN STORED PROCEDURES.
//*
//*  SET PARMS FOR THIS COMPILE:
//*  ---------------------------
//   SET MEM=SAM1                             PROGRAM NAME
//   SET COBOLLIB='IGY.V4R1.SIGYCOMP'         COBOL COMPILER LOADLIB
//   SET DTLIB='EQAW.SEQAMOD'                 DEBUG TOOL LOADLIB
//   SET LELIB='CEE.SCEELKED'                 LE LINKEDIT LIBRARY
//   SET UNITDEV=SYSALLDA                     UNIT FOR TEMP FILES
//*
//*  ****************************
//*        COMPILE STEP
//*  ****************************
//COMPILE  EXEC PGM=IGYCRCTL,REGION=0M,
//   PARM=('TEST(NOHOOK,SEPARATE,EJPD),LIST,MAP,XREF(SHORT),NONUMBER,SOURCE')
//STEPLIB  DD DISP=SHR,DSN=&COBOLLIB
//SYSIN    DD DISP=SHR,DSN=&SYSUID..ADLAB.SOURCE(&MEM)
//SYSLIB   DD DISP=SHR,DSN=&SYSUID..ADLAB.COPYLIB
//SYSPRINT DD DISP=SHR,DSN=&SYSUID..ADLAB.LISTING(&MEM)
//SYSDEBUG DD DISP=SHR,DSN=&SYSUID..ADLAB.SYSDEBUG(&MEM)
//SYSLIN   DD DISP=(MOD,PASS),DSN=&&LOADSET,UNIT=&UNITDEV,
//            SPACE=(80,(10,10))
//SYSUT1   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT2   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT3   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT4   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT5   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT6   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT7   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//*
//CBLPRINT  EXEC PGM=IEBGENER,REGION=0M
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSN=&SYSUID..ADLAB.LISTING(&MEM),DISP=SHR
//SYSUT2    DD SYSOUT=*
//SYSIN     DD DUMMY
//*  *********************************
//*        LINK-EDIT (BINDER) STEP
//*  *********************************
//LKED EXEC PGM=IEWL,REGION=0M,COND=(5,LT,COMPILE),PARM='LIST,XREF'
//SYSLIB   DD DISP=SHR,DSN=&LELIB
//DTLIB    DD DISP=SHR,DSN=&DTLIB
//SYSLMOD  DD DSN=&SYSUID..ADLAB.LOAD(&MEM),DISP=SHR
//SYSLIN   DD DISP=(OLD,DELETE),DSN=&&LOADSET
//*  INCLUDING A DEBUG TOOL LE EXIT (EQADBCXT, EQADDCXT, EQADICXT, OR EQAD3CXT)
//*  IS OPTIONAL.
//*  AN EXIT ENABLES STARTING DEBUG TOOL USING THE USER EXIT DATA SET UTILITY
//*  (ONE OF THE DEBUG TOOL ISPF UTILITIES)
//*  //        DD *
//*   INCLUDE DTLIB(EQADBCXT)
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=&UNITDEV,DCB=BLKSIZE=1024,SPACE=(1024,(200,20))
```

Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM programs

The following table shows various compiler options that can be used to prepare Enterprise COBOL for z/OS Version 3 and COBOL for OS/390® and VM programs for use with the IBM Application Delivery Foundation for z Systems family of products (IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate whether the load module produced is suitable for a production environment. Load modules suitable for a production environment have no significant runtime overhead.

Table 3. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM.

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for IBM Debug for z Systems	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
TEST(NONE, SYM, SEPARATE), LIST, MAP, SOURCE, NONUMBER, XREF(SHORT)	SYSDEBUG	Yes	Suggested for production and test		
NOTEST, LIST, MAP, SOURCE, NONUMBER, NOOPT, XREF(SHORT)	Compiler listing	Yes	N/A	Supported	Supported
NOTEST, LIST, MAP, SOURCE, XREF(SHORT), NUMBER		Yes	N/A	Supported	N/A
LIST, MAP, SOURCE, NONUMBER, XREF(SHORT)	LANGX file	Yes	Not supported	Supported	Supported
LIST, MAP, SOURCE, NONUMBER, NOTEST, NOOPT, XREF	LANGX file	Yes	Supported	Supported	Supported

Preparing Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM programs

Perform the following steps for compiling your Enterprise COBOL for z/OS Version 3 and COBOL for OS/390 and VM programs using the compiler options suggested in Table 3:

1. Create libraries (PDSE is suggested unless PDS is required in your organization) for SYSDEBUG files. Allocate one or more SYSDEBUG libraries for each environment, such as test, and production.
2. Create a corresponding SYSDEBUG library for each load library. Specify `LRECL=(80 to 1024),RECFM=FB,BLKSIZE=(multiple of lrecl < 32K)`.
3. For all programs in both test and production environments, use `TEST(NONE,SYM,SEPARATE),LIST,MAP,SOURCE,NONUMBER,XREF(SHORT)`.

   `TEST` is required by IBM Debug for z Systems.

   The `SEPARATE` suboption produces a SYSDEBUG file. Specifying `NONE` with `SEPARATE` produces a production-ready module that can still be debugged.

   If `OPTIMIZE` is specified, the debugger `JUMPTO` and `GOTO` commands are disabled. These commands are enabled when `NOOPTIMIZE` is specified.

4. The LIST, MAP, SOURCE, and XREF options are needed only if a compiler listing or a LANGX file is used to provide source information to Fault Analyzer for z/OS or Application Performance Analyzer for z/OS. If a SYSDEBUG file is used with these products, or if you are not using Fault Analyzer for z/OS or Application Performance Analyzer for z/OS, the LIST, MAP, SOURCE, and XREF options are optional.
5. The NONUMBER compiler option is needed only if a compiler listing file is used to provide source information to Application Performance Analyzer for z/OS. If a SYSDEBUG file is used with Application Performance Analyzer for z/OS, or if you are not using Application Performance Analyzer for z/OS, the NONUMBER option is optional.
6. Code a SYSDEBUG DD in the JCL of the compiler step.

   ```
   //SYSDEBUG DD DSN= SYSDEBUG.pds(pgmname),DISP=SHR
   ```

   Save the SYSDEBUG file that is produced by the compiler in the SYSDEBUG library and specify a member name that is equal to the program name of your application program. This file is the source information file for IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.
7. Modify the promotion process to promote SYSDEBUG files. When a load module is promoted, for example from test to production, promote the corresponding SYSDEBUG file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the SYSDEBUG file that you perform with the module during promotion.
8. Optionally, include a zDebug Language Environment exit module into the load module during the linkage editor step. This inclusion is one way to enable zDebug's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

   You can also use module EQAD3CXT for batch programs, IMS/TM, IMS BTS programs, and DB2 type MAIN stored procedures.

Sample JCL for compiling Enterprise COBOL for z/OS Version 3 programs

Here is a JCL example for compiling an Enterprise COBOL for z/OS Version 3 program for use with the IBM Application Delivery Foundation for z Systems family of products. This example is a generic sample, and might not meet all your requirements.

Notice that a TEST option is specified. Code the correct suboption of the TEST compiler option for the version of the compiler that you are using. You can also code any other compatible compiler options that are required by your programs.

Also, notice that a SYSDEBUG DD statement was coded. This statement indicates the source information file that the compiler produces. It refers to a SYSDEBUG library that is a PDS or PDSE. The member name must be the same as the program name.

For Enterprise COBOL for z/OS, these are the only required changes.

However, there is an optional change in the linkage editor step. The example shows that a special Language Environment exit module is included in the application load module. Although this exit module is not required, it enables the

| use of zDebug panel 6, which makes the debugger easier to start in some
| environments. If you prefer to use panel 6 to start zDebug, this method is one way
| to enable it. If you do not plan to use zDebug panel 6, then do not include an exit
module.

```
//*     - - -  ADD A JOB CARD ABOVE THIS LINE  - - -
//*
//*  SAMPLE JCL TO PREPARE AN ENTERPRISE COBOL PROGRAM
//*  FOR THE IBM ZSERIES PD TOOLS PRODUCTS:
//*     FAULT ANALYZER, DEBUG TOOL, AND APPLICATION PERF. ANALYZER
//*
//*  NOTES:
//*
//*   COMPILER:
//*    1. A TEST COMPILER PARM IS REQUIRED FOR DEBUG TOOL
//*    2. COMPILER PARM TEST(NONE,SYM,SEP) HAS THREE ADVANTAGES:
//*         - THE MODULE IS READY FOR DEBUG TOOL
//*         - THE MODULE IS PRODUCTION-READY (NO RUN-TIME OVERHEAD)
//*         - A SYSDEBUG FILE IS CREATED THAT CAN BE USED BY DT,FA,APA
//*    3. COMPILER PARMS LIST,MAP,SOURCE,XREF ARE REQUIRED IF YOU PLAN
//*       TO USE THE COMPILER LISTING WITH FA OR APA, OR IPVLANGX
//*    4. COMPILER PARM NOOPT IS OPTIONAL.  HOWEVER, THE DEBUG TOOL
//*       COMMANDS JUMPTO AND GOTO WILL NOT BE AVAILABLE IF
//*       THE OPT PARM IS USED
//*
//*   BINDER (LINKAGE EDITOR):
//*    5. THE INCLUDE FOR MODULE EQAD?CXT IS *OPTIONAL*.  IT IS AN
//*       LE EXIT MODULE THAT CAN BE USED TO START DEBUG TOOL.
//*       UNDERSTAND THE METHODS AVAILABLE FOR STARTING DEBUG TOOL,
//*       AND CHOOSE WHETHER YOU WANT TO USE THE LE EXITS.
//*         IF YOU USE THIS METHOD, LOAD THE CORRECT EXIT MODULE:
//*            EQADBCXT: FOR BATCH PROGRAMS
//*            EQADICXT: FOR ONLINE IMS PROGRAMS
//*            EQADDCXT: FOR DB2 STORED PROCEDURES (OF TYPE MAIN AND SUB)
//*           (for SUB this is supported only for invocations through call_sub)
//*            (DO NOT INCLUDE AN EXIT FOR CICS PROGRAMS)
//*            YOU CAN ALSO USE MODULE EQAD3CXT FOR BATCH PROGRAMS, ONLINE IMS
//*            PROGRAMS, DB2 TYPE MAIN STORED PROCEDURES.
//*
//*  SET PARMS FOR THIS COMPILE:
//*  ---------------------------
//   SET MEM=SAM1                             PROGRAM NAME
//   SET COBOLLIB='IGY.V3R4.SIGYCOMP'         COBOL COMPILER LOADLIB
//   SET DTLIB='EQAW.SEQAMOD'                 DEBUG TOOL LOADLIB
//   SET LELIB='CEE.SCEELKED'                 LE LINKEDIT LIBRARY
//   SET UNITDEV=SYSALLDA                     UNIT FOR TEMP FILES
//*
//*  ****************************
//*        COMPILE STEP
//*  ****************************
//COMPILE  EXEC PGM=IGYCRCTL,REGION=0M,
//   PARM=('TEST(NONE,SYM,SEPARATE),LIST,MAP,SOURCE,NONUMBER,XREF(SHORT)')
//STEPLIB  DD DISP=SHR,DSN=&COBOLLIB
//SYSIN    DD DISP=SHR,DSN=&SYSUID..ADLAB.SOURCE(&MEM)
//SYSLIB   DD DISP=SHR,DSN=&SYSUID..ADLAB.COPYLIB
//SYSPRINT DD DISP=SHR,DSN=&SYSUID..ADLAB.LISTING(&MEM)
//SYSDEBUG DD DISP=SHR,DSN=&SYSUID..ADLAB.SYSDEBUG(&MEM)
//SYSLIN   DD DISP=(MOD,PASS),DSN=&&LOADSET,UNIT=&UNITDEV,
//            SPACE=(80,(10,10))
//SYSUT1   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT2   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT3   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT4   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT5   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT6   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT7   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//*
```

```
//CBLPRINT  EXEC PGM=IEBGENER,REGION=0M
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSN=&SYSUID..ADLAB.LISTING(&MEM),DISP=SHR
//SYSUT2    DD SYSOUT=*
//SYSIN     DD DUMMY
//*  *********************************
//*        LINK-EDIT (BINDER) STEP
//*  *********************************
//LKED EXEC PGM=IEWL,REGION=0M,COND=(5,LT,COMPILE),PARM='LIST,XREF'
//SYSLIB   DD DISP=SHR,DSN=&LELIB
//DTLIB    DD DISP=SHR,DSN=&DTLIB
//SYSLMOD  DD DSN=&SYSUID..ADLAB.LOAD(&MEM),DISP=SHR
//SYSLIN   DD DISP=(OLD,DELETE),DSN=&&LOADSET
//*  INCLUDING A DEBUG TOOL LE EXIT (EQADBCXT, EQADDCXT, EQADICXT OR EQAD3CXT)
//*  IS OPTIONAL.
//*  AN EXIT ENABLES STARTING DEBUG TOOL USING THE USER EXIT DATA SET UTILITY
//*  (ONE OF THE DEBUG TOOL ISPF UTILITIES)
//*  //        DD *
//*   INCLUDE DTLIB(EQADBCXT)
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=&UNITDEV,DCB=BLKSIZE=1024,SPACE=(1024,(200,20))
```

COBOL for MVS and VM programs

The following table shows various compiler options that can be used to prepare COBOL for MVS™ and VM programs for use with the IBM Application Delivery Foundation for z Systems family of products (IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate whether the load module produced is suitable for a production environment. Load modules suitable for a production environment have no significant runtime overhead.

Table 4. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for COBOL for MVS and VM.

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for IBM Debug for z Systems	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
TEST(ALL, SYM), LIST, MAP, SOURCE, NOOPT, NONUMBER, XREF(SHORT)	Compiler listing	No	Suggested for test. (Using zDebug in production for this compiler is not suggested.)		
NOTEST, LIST, MAP, SOURCE, NONUMBER, XREF(SHORT)		Yes	N/A	Suggested for production	
NOTEST, LIST, MAP, SOURCE, NONUMBER, XREF(SHORT)	LANGX file	Yes	N/A	Supported	Supported

Preparing COBOL for MVS and VM programs

Perform the following steps for compiling your COBOL for MVS and VM programs:

1. Create libraries (PDSE is suggested unless PDS is required in your organization) for compiler listing files. Allocate one or more compiler listing libraries for each environment, such as test and production.
2. Create a corresponding listing library for each load library. Specify `LRECL=133,RECFM=FBA,BLKSIZE=(multiple of lrecl < 32K)`.
3. For all programs, such as batch, CICS, and IMS:
   - In test environments, specify compiler options `TEST(ALL,SYM),NOOPT,LIST,MAP,SOURCE,NONUMBER,XREF(SHORT)` to create a module that can be used with IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.

     `TEST` is required for IBM Debug for z Systems.

     The `ALL` suboption adds debug hooks, which add some runtime overhead.

     `SYM` stores symbolics data that is required by IBM Debug for z Systems into the module, which can make it significantly larger.

     The other options format the compiler listing as required by IBM Debug for z Systems, Fault Analyzer for z/OS, and Application Performance Analyzer for z/OS.
   - In production environments, specify compiler options `NOTEST,LIST,MAP,SOURCE,NONUMBER,XREF(SHORT)` to create a production-ready module that can be used with Fault Analyzer for z/OS and Application Performance Analyzer for z/OS (but not IBM Debug for z Systems). Specify `OPTIMIZE` if preferred.

     `NOTEST` disables source level debugging with zDebug, but can provide better performance and smaller module size.

     The other options (except `OPTIMIZE`) format the compiler listing as required by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.
4. Modify the SYSPRINT DD in the JCL of the compiler step to refer to a file.

   ```
   //SYSPRINT DD DSN= compiler.listing.pds(pgmname),DISP=SHR
   ```

   Save the compiler listing in a file in the compiler listing library and specify a member name that is equal to the program name of your application program. This file is the source information file for IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.
5. Modify the promotion process to promote compiler listing files. When a load module is promoted, for example, from test to production, promote the corresponding compiler listing file or files. A promotion can be a recompile, a copy, or a move. Perform the same steps with the compiler listing file that you perform with the module during promotion.
6. Optionally, include a zDebug Language Environment exit module into the load module during the linkage editor step. This inclusion is one way to enable zDebug's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, that is based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

   You can also use module EQAD3CXT for batch programs, IMS/TM, IMS BTS programs, and DB2 type MAIN stored procedures.

Sample JCL for compiling COBOL for MVS and VM programs

Here is a JCL example for compiling a COBOL for MVS and VM program for use with the IBM Application Delivery Foundation for z Systems family of products. This sample is a generic sample, and might not meet all your requirements.

Notice that a TEST option is specified. Code the correct suboptions of the TEST compiler option for the version of the compiler that you are using. You can also code any other compatible compiler options that are required by your programs.

Also, notice that the SYSPRINT DD refers to a permanent file. This file is the source information file that the compiler produces. It refers to a listing library that is a PDS or PDSE. The member name must be the same as the program name. For COBOL for MVS and VM, these are the only required changes.

However, there is an optional change in the linkage editor step. The example shows that a special Language Environment exit module is included in the application load module. Although this exit module is not required, it enables the use of zDebug panel 6, which makes the debugger easier to start in some environments. If you prefer to use panel 6 to start zDebug, this method is one way to enable it. If you do not plan to use zDebug panel 6, then do not include an exit module.

```
//*     - - -  ADD A JOB CARD ABOVE THIS LINE  - - -
//*
//*  SAMPLE JCL TO PREPARE A COBOL FOR MVS AND VM PROGRAM
//*  FOR THE IBM ZSERIES PD TOOLS PRODUCTS:
//*     FAULT ANALYZER, DEBUG TOOL, AND APPLICATION PERF. ANALYZER
//*
//*  NOTES:
//*
//*   COMPILER:
//*    1. A TEST COMPILER PARM IS REQUIRED FOR DEBUG TOOL
//*    2. COMPILER PARMS LIST,MAP,SOURCE,XREF ARE REQUIRED IF YOU PLAN
//*       TO USE THE COMPILER LISTING WITH FA OR APA, OR IPVLANGX
//*    3. COMPILER PARM NOOPT IS OPTIONAL.  HOWEVER, THE DEBUG TOOL
//*       COMMANDS JUMPTO AND GOTO WILL NOT BE AVAILABLE IF
//*       THE OPT PARM IS USED
//*
//*   BINDER (LINKAGE EDITOR):
//*    4. THE INCLUDE FOR MODULE EQAD?CXT IS *OPTIONAL*.  IT IS AN
//*       LE EXIT MODULE THAT CAN BE USED TO START DEBUG TOOL.
//*       UNDERSTAND THE METHODS AVAILABLE FOR STARTING DEBUG TOOL,
//*       AND CHOOSE WHETHER YOU WANT TO USE THE LE EXITS.
//*         IF YOU USE THIS METHOD, LOAD THE CORRECT EXIT MODULE:
//*            EQADBCXT: FOR BATCH PROGRAMS
//*            EQADICXT: FOR ONLINE IMS PROGRAMS
//*            EQADDCXT: FOR DB2 STORED PROCEDURES (OF TYPE MAIN AND SUB)
//*           (for SUB this is supported only for invocations through call_sub)
//*            (DO NOT INCLUDE AN EXIT FOR CICS PROGRAMS)
//*           YOU CAN ALSO USE MODULE EQAD3CXT FOR BATCH PROGRAMS, ONLINE IMS
//*            PROGRAMS, DB2 TYPE MAIN STORED PROCEDURES.
//*
//*  SET PARMS FOR THIS COMPILE:
//*  ---------------------------
//   SET MEM=SAM1                             PROGRAM NAME
//   SET COBOLLIB='IGY.SIGYCOMP'              COBOL COMPILER LOADLIB
//   SET DTLIB='EQAW.SEQAMOD'                 DEBUG TOOL LOADLIB
//   SET LELIB='CEE.SCEELKED'                 LE LINKEDIT LIBRARY
//   SET UNITDEV=SYSALLDA                     UNIT FOR TEMP FILES
//*
//*  ****************************
//*        COMPILE STEP
//*  ****************************
```

```
////COMPILE  EXEC PGM=IGYCRCTL,REGION=0M,
//   PARM=(NOTEST,LIST,MAP,SOURCE,NONUMBER,XREF(SHORT)')
//STEPLIB  DD DISP=SHR,DSN=&COBOLLIB
//SYSIN    DD DISP=SHR,DSN=&SYSUID..ADLAB.SOURCE(&MEM)
//SYSLIB   DD DISP=SHR,DSN=&SYSUID..ADLAB.COPYLIB
//SYSPRINT DD DISP=SHR,DSN=&SYSUID..ADLAB.LISTING(&MEM)
//SYSDEBUG DD DISP=SHR,DSN=&SYSUID..ADLAB.SYSDEBUG(&MEM)
//SYSLIN   DD DISP=(MOD,PASS),DSN=&&LOADSET,UNIT=&UNITDEV,
//            SPACE=(80,(10,10))
//SYSUT1   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT2   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT3   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT4   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT5   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT6   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT7   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//*
//CBLPRINT  EXEC PGM=IEBGENER,REGION=0M
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSN=&SYSUID..ADLAB.LISTING(&MEM),DISP=SHR
//SYSUT2    DD SYSOUT=*
//SYSIN     DD DUMMY
//*  *********************************
//*        LINK-EDIT (BINDER) STEP
//*  *********************************
//LKED EXEC PGM=IEWL,REGION=0M,COND=(5,LT,COMPILE),PARM='LIST,XREF'
//SYSLIB   DD DISP=SHR,DSN=&LELIB
//*** DTLIB    DD DISP=SHR,DSN=&DTLIB
//SYSLMOD  DD DSN=&SYSUID..ADLAB.LOAD(&MEM),DISP=SHR
//SYSLIN   DD DISP=(OLD,DELETE),DSN=&&LOADSET
//*  INCLUDING A DEBUG TOOL LE EXIT (EQADBCXT, EQADDCXT, EQADICXT OR EQAD3CXT)
//*  IS OPTIONAL.
//*  AN EXIT ENABLES STARTING DEBUG TOOL USING THE USER EXIT DATA SET UTILITY
//*  (ONE OF THE DEBUG TOOL ISPF UTILITIES)
//*  //        DD *
//*   INCLUDE DTLIB(EQADBCXT)
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=&UNITDEV,DCB=BLKSIZE=1024,SPACE=(1024,(200,20))
```

VS COBOL II programs

If you are currently using the TEST option to compile your programs, consider using NOTEST. Using NOTEST allows you to take advantage of IBM Debug for z Systems functionality that is not available when compiling with the TEST option. Examples of IBM Debug for z Systems functions that are available when compiling with the NOTEST option include the automonitor feature and using AT ENTRY *program name* breakpoints. Compiling with NOTEST also allows you to generate a module that can be debugged but does not incur extra overhead when running without the debugger.

The following table shows various compiler options that can be used to prepare VS COBOL II programs for use with the IBM Application Delivery Foundation for z Systems family of products (IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate whether the load module produced is suitable for a production environment. Load modules suitable for a production environment have no significant runtime overhead.

Table 5. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for VS COBOL II.

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for IBM Debug for z Systems	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
NOTEST, LIST, MAP, SOURCE, XREF, NONUMBER, NOOFFSET	Compiler listing	Yes	N/A	Supported	Supported
NOTEST, LIST, MAP, SOURCE, XREF, NUMBER		Yes	N/A	Supported	N/A
NOTEST, LIST, MAP, NOOPT, SOURCE, XREF, NONUMBER	LANGX file	Yes	Suggested for production and test		

Preparing VS COBOL II programs

Perform the following steps for compiling your VS COBOL II programs using the compiler options suggested in Table 5:

1. Allocate libraries (PDSE is suggested unless PDS is required for your organization) for LANGX files. Allocate one or more LANGX libraries for each environment, such as test and production.
2. Create a corresponding LANGX library for each load library. Specify `LRECL=1562 or greater,RECFM=VB,BLKSIZE= lrecl+4 to 32k`.
3. For all programs, such as batch, CICS, and IMS, in both test and production environments, compile with `NOTEST,LIST,MAP,NOOPT,SOURCE,XREF,NONUMBER` compiler options.
4. Modify the SYSPRINT DD in the compiler step to refer to a file. It can be either a permanent or temporary file. This file is the input to the IPVLANGX utility.
5. Add a step after the compiler step to run the ADFz IPVLANGX utility. This utility program reads the compiler listing and generates a LANGX file. This file is the source information file for IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS. Save the LANGX file in the LANGX file library and specify a member name that is equal to the program name of your application program.
6. If the module is linked with Language Environment services, optionally include a zDebug Language Environment exit module into the load module during the linkage editor step. This approach is one way to enable the zDebug panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs or if the module is not linked with Language Environment services (it is linked with COBOL II runtime services).

You can also use module EQAD3CXT for batch programs, IMS/TM, IMS BTS programs, and DB2 type MAIN stored procedures.

7. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.

Sample JCL for compiling VS COBOL II programs

Here is an example of JCL for compiling a VS COBOL II program for use with IBM Application Delivery Foundation for z Systems family of products. This sample is a generic sample, and might not meet all your requirements.

Notice the compiler options that are used and notice that the compiler listing is passed to an added step that generates a LANGX file. The compiler listing can be stored in a permanent file or can be passed in a temporary file. For VS COBOL II, these are the only required changes.

However, there is an optional change in the linkage editor step. The example includes a special Language Environment exit module in the application load module. Although this exit module is not required, it enables the use of zDebug panel 6, which makes the debugger easier to start in some environments. If you prefer to use panel 6 to start zDebug, this inclusion is one way to enable it. If you do not plan to use zDebug panel 6, then do not include an exit module. Do not include the exit module for CICS programs or if the module is not linked with Language Environment services (it is linked with COBOL II runtime services).

```
//*     - - -  ADD A JOB CARD ABOVE THIS LINE  - - -
//*
//*  SAMPLE JCL TO PREPARE A VS COBOL II PROGRAM
//*  FOR THE IBM ZSERIES PD TOOLS PRODUCTS:
//*     FAULT ANALYZER, DEBUG TOOL, AND APPLICATION PERF. ANALYZER
//*
//*  NOTES:
//*
//*   COMPILER:
//*    1. COMPILER OPTIONS LIST,MAP,SOURCE,XREF ARE REQUIRED IF YOU
//*       PLAN TO USE THE LISTING WITH A PD TOOLS PRODUCT, OR TO
//*       PROCESS THE LISTING WITH THE IPVLANGX UTILITY
//*    2. COMPILER OPTION NOTEST IS SUGGESTED FOR ALL COBOL II
//*       PROGRAMS, EVEN IF IBM DEBUG TOOL FOR Z/OS WILL BE USED
//*
//*   BINDER (LINKAGE EDITOR):
//*    3. IN THIS EXAMPLE, THE MODULE IS LINKED WITH LANGUAGE
//*       ENVIRONMENT RUNTIME SERVICES. THIS IS CONTROLLED BY THE
//*       LIBRARY OR LIBRARIES SPECIFIED IN THE SYSLIB DD IN THE
//*       BINDER STEP.
//*    4. THE INCLUDE FOR MODULE EQAD?CXT IS *OPTIONAL*.  IT IS AN
//*       LE EXIT MODULE THAT CAN BE USED TO START DEBUG TOOL.
//*       UNDERSTAND THE METHODS AVAILABLE FOR STARTING DEBUG TOOL,
//*       AND CHOOSE WHETHER YOU WANT TO USE THE LE EXITS.
//*         IF YOU USE THIS METHOD, LOAD THE CORRECT EXIT MODULE:
//*            EQADBCXT: FOR BATCH PROGRAMS
//*            EQADICXT: FOR ONLINE IMS PROGRAMS
//*            EQADDCXT: FOR DB2 STORED PROCEDURES (OF TYPE MAIN AND SUB)
//*           (for SUB this is supported only for invocations through call_sub)
//*         (DO NOT INCLUDE AN EXIT FOR CICS PROGRAMS, OR FOR
//*          PROGRAMS LINKED WITH THE COBOL II RUNTIME SERVICES
//*          INSTEAD OF LANGUAGE ENVIRONMENT RUNTIME SERVICES)
//*          YOU CAN ALSO USE MODULE EQAD3CXT FOR BATCH PROGRAMS, ONLINE IMS
//*          PROGRAMS, DB2 TYPE MAIN STORED PROCEDURES.
//*
```

```
//*  SET OPTIONS FOR THIS COMPILE:
//*  ---------------------------
//   SET MEM=SAMII1                          PROGRAM NAME
//   SET COB2COMP='IGY.V1R4M0.COB2COMP'      COBOL II COMPILER LIB
//   SET DTLIB='EQAW.SEQAMOD'                DEBUG TOOL LOADLIB
//   SET LELKED='CEE.SCEELKED'               LE LINK LIBRARY
//   SET LELIB='CEE.SCEERUN'                 LE RUNTIME LIBRARY
//   SET UNITDEV=SYSALLDA                    TEMP data set UNIT
//   SET LANGX='IPVLANGX'                    IPVLANGX UTILITY PROGRAM
//   SET LANGXLIB='IPV.SIPVMODA'             LIB FOR IPVLANGX UTILITY
//*    NOTE: USE THE IPVLANGX FACILITY SHIPPED WITH THE COMMON COMPONENT.
//*
//*  ****************************
//*        COMPILE STEP
//*  ****************************
//COMPILE  EXEC PGM=IGYCRCTL,REGION=4M,
//   PARM=('NOTEST,LIST,MAP,NOOPT,SOURCE,XREF,NONUMBER',
//         'RES,APOST,LIB,DYNAM,NORENT,NOSSRANGE')
//STEPLIB  DD DISP=SHR,DSN=&COB2COMP
//SYSIN    DD DISP=SHR,DSN=&SYSUID..ADLAB.SOURCE(&MEM)
//SYSLIB   DD DISP=SHR,DSN=&SYSUID..ADLAB.COPYLIB
//SYSPRINT DD DISP=SHR,DSN=&SYSUID..ADLAB.LISTING(&MEM)
//SYSLIN   DD DISP=(MOD,PASS),DSN=&&LOADSET,UNIT=&UNITDEV,
//            SPACE=(80,(10,10))
//SYSUT1   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT2   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT3   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT4   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT5   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT6   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT7   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//*
//CBLPRINT  EXEC PGM=IEBGENER,REGION=0M
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSN=&SYSUID..ADLAB.LISTING(&MEM),DISP=SHR
//SYSUT2    DD SYSOUT=*
//SYSIN     DD DUMMY
//*
//* *********************************
//* STEP TO GENERATE A LANGX FILE
//* *********************************
//LANGX EXEC PGM=&LANGX,REGION=32M,
//   PARM='(COBOL ERROR 64K CREF'
//STEPLIB  DD DISP=SHR,DSN=&LANGXLIB
//         DD DISP=SHR,DSN=&LELIB
//LISTING  DD DSN=&SYSUID..ADLAB.LISTING(&MEM),DISP=SHR
//IDILANGX DD DISP=SHR,DSN=&SYSUID..ADLAB.EQALANGX(&MEM)
//*
//*  *********************************
//*        LINK-EDIT (BINDER) STEP
//*  *********************************
//LKED EXEC PGM=IEWL,REGION=0M,COND=(5,LT,COMPILE),PARM='LIST,XREF'
//SYSLIB   DD DISP=SHR,DSN=&LELKED
//DTLIB    DD DISP=SHR,DSN=&DTLIB
//SYSLMOD  DD DSN=&SYSUID..ADLAB.LOAD(&MEM),DISP=SHR
//SYSLIN   DD DISP=(OLD,DELETE),DSN=&&LOADSET
//*  INCLUDING A DEBUG TOOL LE EXIT (EQADBCXT, EQADDCXT, EQADICXT OR EQAD3CXT)
//* IS OPTIONAL
//*  AN EXIT ENABLES STARTING DEBUG TOOL USING THE USER EXIT DATA SET UTILITY
//*  (ONE OF THE DEBUG TOOL ISPF UTILITIES)
//*  //        DD *
//*   INCLUDE DTLIB(EQADBCXT)
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=&UNITDEV,DCB=BLKSIZE=1024,SPACE=(1024,(200,20))
```

OS/VS COBOL programs

The following table shows various compiler options that can be used to prepare OS/VS COBOL programs for use with the IBM Application Delivery Foundation for z Systems family of products (IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate whether the load module produced is suitable for a production environment. Load modules suitable for a production environment have no significant runtime overhead.

Table 6. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for OS/VS COBOL.

Compiler options	**Source information file type produced**	**Is the load module production ready?**	**Options supported and suggested for IBM Debug for z Systems**	**Options supported and suggested for Fault Analyzer for z/OS**	**Options supported and suggested for Application Performance Analyzer for z/OS**
DMAP, NOCLIST, NOLST, PMAP, SOURCE, VERB, XREF(SHORT)	Compiler listing	Yes	N/A	Supported	Supported
(LIST, NOPMAP) or (CLIST, NOPMAP) or (CLIST, PMAP)		Yes	N/A	Supported	N/A
NOBATCH, NOCLIST, NOCOUNT, DMAP, NOLST, PMAP, SOURCE, NOSYMDMP, NOTEST, NOOPT, VERB, XREF(SHORT)	LANGX file	Yes	Suggested for production and test		

Preparing OS/VS COBOL programs

Perform the following steps for compiling your OS/VS COBOL programs:

1. Allocate libraries (PDSE is suggested unless PDS is required for your organization) for LANGX files. Allocate one or more LANGX libraries for each environment, such as test and production.
2. Create a corresponding LANGX library for each load library. Specify `LRECL=1562 or greater,RECFM=VB,BLKSIZE= lrecl+4 to 32k`.
3. For all programs, such as batch, CICS, and IMS, in both test and production environments, compile with the `NOBATCH`, `NOCLIST`, `NOCOUNT`, `DMAP`, `NOLST`, `PMAP`, `SOURCE`, `NOSYMDMP`, `NOTEST`, `NOOPT`, `VERB`, `XREF(SHORT)` compiler options. The module is production-ready and can be debugged using IBM Debug for z Systems.
4. Modify the SYSPRINT DD in the compiler step to refer to a file. It can be either a permanent or temporary file. This is the input to the IPVLANGX utility.

5. Add a step after the compiler step to run the ADFz IPVLANGX utility. This utility program reads the compiler listing and generates a LANGX file, which is the source information file for IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS. Save the LANGX file in the LANGX file library, and specify a member name that is equal to the program name of your application program.
6. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.

Sample JCL for compiling OS/VS COBOL programs

Here is a JCL example for compiling an OS/VS program for use with the IBM Application Delivery Foundation for z Systems family of products:

```
//*     - - -  ADD A JOB CARD ABOVE THIS LINE  - - -
//*
//*  SAMPLE JCL TO PREPARE AN OS VS COBOL PROGRAM
//*  FOR THE IBM ZSERIES PD TOOLS PRODUCTS:
//*     FAULT ANALYZER, DEBUG TOOL, AND APPLICATION PERF. ANALYZER
//*
//*  NOTES:
//*
//*   COMPILER:
//*    -  COMPILER PARMS DMAP,NOCLIST,NOLST,PMAP,SOURCE,VERB,XREF
//*       ARE REQUIRED IF YOU PLAN TO USE THE COMPILER LISTING WITH
//*       PD TOOLS AND/OR PROCESS IT WITH IPVLANGX
//*
//*   A STEP THAT PROCESSES THE SYSADATA FILE,
//*   AND CREATES A LANGX FILE IS NEEDED.
//*
//*  SET PARMS FOR THIS COMPILE:
//*  ---------------------------
// SET MEM=SAMOS1                       PROGRAM NAME
// SET OSVSCOMP='IGY.VSCOLIB'           OS VS COBOL COMPILER LIBRARY
// SET LELIB='CEE.SCEELKED'             LE LINKEDIT LIBRARY
// SET UNITDEV=SYSALLDA                 UNIT FOR TEMP FILES
// SET SCEERUN='CEE.SCEERUN'            LANGUAGE ENVIRON SCEERUN LIB
// SET LANGX='IPVLANGX'                 IPVLANGX UTILITY PROGRAM
// SET LANGXLIB='IPV.SIPVMODA'          LIBRARY FOR IPVLANGX UTILITY
//*    NOTE: USE THE IPVLANGX FACILITY SHIPPED WITH THE COMMON COMPONENT.
//*
//*  ****************************
//*        COMPILE STEP
//*  ****************************
//COMPILE  EXEC PGM=IKFCBL00,REGION=4M,
//   PARM=('DMAP,NOCLIST,NOLST,NOOPT,SOURCE,VERB,XREF(SHORT)')
//*   FOR DT (CHECK DEFAULTS): NOBATCH,NOCOUNT,PMAP,NOSYMDMP,NOTEST
//STEPLIB  DD DISP=SHR,DSN=&OSVSCOMP
//SYSIN    DD DISP=SHR,DSN=&SYSUID..ADLAB.SOURCE(&MEM)
//SYSLIB   DD DISP=SHR,DSN=&SYSUID..ADLAB.COPYLIB
//SYSPRINT DD DISP=SHR,DSN=&SYSUID..ADLAB.OSVSCOB.LISTING(&MEM)
//SYSLIN   DD DISP=(MOD,PASS),DSN=&&LOADSET,UNIT=&UNITDEV,
//            SPACE=(80,(10,10))
//SYSUT1   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT2   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT3   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT4   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT5   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT6   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//SYSUT7   DD SPACE=(80,(10,10),,,ROUND),UNIT=&UNITDEV
//*
//CBLPRINT  EXEC PGM=IEBGENER,REGION=0M
//SYSPRINT  DD SYSOUT=*
```

```
//SYSUT1    DD DSN=&SYSUID..ADLAB.OSVSCOB.LISTING(&MEM),DISP=SHR
//SYSUT2    DD SYSOUT=*
//SYSIN     DD DUMMY
//*
//*  *********************************
//*     STEP TO GENERATE LANGX FILE
//*  *********************************
//LANGX    EXEC PGM=&LANGX,REGION=32M,
//  PARM='(COBOL ERROR 64K CREF'
//STEPLIB  DD DISP=SHR,DSN=&LANGXLIB
//         DD DISP=SHR,DSN=&SCEERUN
//LISTING  DD DSN=&SYSUID..ADLAB.OSVSCOB.LISTING(&MEM),DISP=SHR
//IDILANGX DD DISP=SHR,DSN=&SYSUID..ADLAB.EQALANGX(&MEM)
//*
//*  *********************************
//*        LINK-EDIT (BINDER) STEP
//*  *********************************
//LKED EXEC PGM=IEWL,REGION=0M,COND=(5,LT,COMPILE),PARM='LIST,XREF'
//SYSLIB   DD DISP=SHR,DSN=&LELIB
//SYSLMOD  DD DSN=&SYSUID..ADLAB.LOAD(&MEM),DISP=SHR
//SYSLIN   DD DISP=(OLD,DELETE),DSN=&&LOADSET
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=&UNITDEV,DCB=BLKSIZE=1024,SPACE=(1024,(200,20))
```

Enterprise PL/I Version 3.7 and later programs

The following table shows various compiler options that can be used to prepare Enterprise PL/I Version 3.7 and later programs for use with the IBM Application Delivery Foundation for z Systems family of products (IBM Debug for z Systems, IBM Fault Analyzer for z/OS and IBM Application Performance Analyzer for z/OS). The methods suggested in the following table indicate whether the load module produced is suitable for a production environment. Load modules suitable for a production environment have no significant runtime overhead.

Table 7. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for Enterprise PL/I Version 3.7 and later.

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for IBM Debug for z Systems	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
For Enterprise PL/I Version 3.7: TEST(ALL, SYM, NOHOOK, SEPARATE, SEPNAME, AALL), NOPT, AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL) For Enterprise PL/I Version 3.8 and later: TEST(ALL, SYM, NOHOOK, SEPARATE, SEPNAME), LISTVIEW (AALL), NOPT, AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)	SYSDEBUG file used by IBM Debug for z Systems and Fault Analyzer for z/OS. LANGX file used by Application Performance Analyzer for z/OS	Although the module is larger than a module compiled with the NOTEST option, you can use the module in production if needed.	Suggested for test. You can also use these options in a production environment if the increased load module size is not an issue.		

Table 7. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for Enterprise PL/I Version 3.7 and later (continued).

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for IBM Debug for z Systems	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NOTEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)	Compiler listing	Yes	N/A	Supported	N/A
	LANGX file	Yes	N/A	Suggested for production and test	

Preparing Enterprise PL/I Version 3.7 and later programs

Perform the following steps for compiling your Enterprise PL/I Version 3.7 and later programs:

1. Create a library (PDSE is suggested unless PDS is required for your organization) for SYSDEBUG files. This library is only needed in test environments where debugging is performed using `LRECL=(80 to 1024),RECFM=FB,BLKSIZE=(multiple of lrecl < 32K)`.
2. Allocate one or more LANGX libraries for each environment, such as test and production.
3. Create a corresponding LANGX library for each load library. Specify `LRECL=1562 or greater,RECFM=VB,BLKSIZE= lrecl+4 to 32k`.
4. For all programs, such as batch, CICS, and IMS:
   - In test environments:
     - When using the Enterprise PL/I Version 3.7 compiler:

       For all programs, specify the following compiler options: `TEST(ALL,SYM,NOHOOK,SEPARATE,SEPNAME,AALL), NOPT, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)`.
     - When using the Enterprise PL/I Version 3.8 and later compilers:

       For all programs, specify the following compiler options: `TEST(ALL,SYM,NOHOOK,SEPARATE,SEPNAME), LISTVIEW(AALL), NOPT, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)`.

     `TEST(...)` and `NOPT` are required by zDebug.

     The `SEPARATE` suboption produces a SYSDEBUG file. Save the SYSDEBUG file that is created by the compiler for IBM Debug for z Systems and optionally, IBM Fault Analyzer for z/OS.

     The `AALL (AFTERALL)` suboption of `TEST` or `LISTVIEW` stores program source information in the SYSDEBUG file that contains information after the last preprocessor, such as macros and INCLUDEs. This expanded source information is available in the source window of IBM Debug for z Systems while debugging.

The other options format the compiler listing as required for the IPVLANGX utility.

Consider using the `TEST(ALL,NOHOOK,SEPARATE)` options for best performance and to produce a module that can be debugged. Depending on the policies in your organization, the module can be considered for production.

- In production environments:
  - When using the Enterprise PL/I Version 3.7 or later compiler:

    For all programs, specify `NOTEST, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)`.

  `NOTEST` disables zDebug, but produces a smaller load module.

  The other options format the compiler listing as required for the IPVLANGX utility to produce a production-ready module that can be used with IBM Fault Analyzer for z/OS and IBM Application Performance Analyzer for z/OS (but not IBM Debug for z Systems).

5. When a `TEST(...SEPARATE)` option is used, code a SYSDEBUG DD in the second compiler step as follows:

```
//SYSDEBUG DD DSN= sysdebug.pds(pgmname),DISP=SHR
```

   This is the source information file for IBM Debug for z Systems, and optionally, IBM Fault Analyzer for z/OS. Save it in the SYSDEBUG library, and specify a member name that is equal to the primary entry point name or CSECT name of your application program.
6. Modify the SYSPRINT DD in the compiler step. This file is the compiler listing. Write the listing to either a permanent or temporary file. This file is the input to the IPVLANGX utility.

   Note: This compiler typically renames CSECTs according to an internal compiler algorithm. Therefore, it is not recommended to store PL/I compiler listings or side files using CSECT names as they might not be found by IBM Application Performance Analyzer for z/OS or IBM Fault Analyzer for z/OS. Instead, use the primary entry point name.
7. Add a step after the compile step to run the IPVLANGX utility. This utility reads the compiler listing and generates a LANGX file. This file is the source information file for IBM Fault Analyzer for z/OS and IBM Application Performance Analyzer for z/OS. Save the LANGX file in the LANGX file library, and specify a member name that is equal to the primary entry point name of your application program.
8. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.
9. If you compile with the TEST option and promote these modules into production, promote the SYSDEBUG file for your production environment.
10. Optionally, include a zDebug Language Environment exit module into the load module during the linkage editor step. This approach is one way to

| enable zDebug's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

You can also use module EQAD3CXT for batch programs, IMS/TM, IMS BTS programs, and DB2 type MAIN stored procedures

Sample JCL for compiling Enterprise PL/I for z/OS Version 3.7 or later programs

Here is a JCL example for compiling an Enterprise PL/I for z/OS Version 3.7 or
| later program for use with the IBM Application Delivery Foundation for z Systems
| family of products.

```
//* - - - ADD A JOB CARD ABOVE THIS LINE - - -
//*
//* SAMPLE JCL TO PREPARE AN ENTERPRISE PL/I V3.7 OR LATER
//* PROGRAM FOR THE IBM ZSERIES PD TOOLS PRODUCTS:
//* FAULT ANALYZER, DEBUG TOOL, AND APPLICATION PERF. ANALYZER
//*
//* NOTES:
//*
//* COMPILER:
//* 1. COMPILER PARMS TEST IS REQUIRED FOR DEBUG TOOL
//* 2. COMPILER PARM NOPT IS RECOMMENDED FOR DEBUG TOOL
//* 3. COMPILER PARM:
//*     TEST(ALL,SYM,NOHOOK,SEPARATE,SEPNAME,AALL) (V3.7)
//*     TEST(ALL,SYM,NOHOOK,SEPARATE,SEPNAME),LISTVIEW(AALL), (V3.8+)
//*    IS USED BECAUSE:
//*    - THE MODULE IS READY FOR DEBUG TOOL
//*    - NOHOOK DOES NOT HAVE RUN-TIME CPU OVERHEAD. HOWEVER, THE
//*      MODULE IS LARGER BECAUSE OF STATEMENT TABLE
//*    - A SYSDEBUG FILE IS CREATED THAT CAN BE USED BY DT,FA,APA
//* 4. COMPILER PARMS AGGREGATE,ATTRIBUTES(FULL),NOBLKOFF,LIST,
//*    MAP,NEST,NONUMBER,OPTIONS,SOURCE,STMT,XREF(FULL) ARE NEEDED
//*    TO PROCESS THE COMPILER LISTING WITH IPVLANGX
//*
//* BINDER (LINKAGE EDITOR):
//* 5. THE INCLUDE FOR MODULE EQAD?CXT IS OPTIONAL. IT IS AN
//*    LE EXIT MODULE THAT CAN BE USED TO START DEBUG TOOL.
//*    UNDERSTAND THE METHODS AVAILABLE FOR STARTING DEBUG TOOL,
//*    AND CHOOSE WHETHER YOU WANT TO USE THE LE EXITS.
//*    IF YOU USE THIS METHOD, LOAD THE CORRECT EXIT MODULE:
//*      EQADBCXT: FOR BATCH PROGRAMS
//*      EQADICXT: FOR ONLINE IMS PROGRAMS
//*      EQADDCXT: FOR DB2 STORED PROCEDURES (OF TYPE MAIN AND SUB)
//*           (for SUB this is supported only for invocations through call_sub)
//*    (DO NOT INCLUDE AN EXIT FOR CICS PROGRAMS)
//*     YOU CAN ALSO USE MODULE EQAD3CXT FOR BATCH PROGRAMS, ONLINE IMS
//*          PROGRAMS, DB2 TYPE MAIN STORED PROCEDURES.
//*
//* SET PARMS FOR THIS COMPILE:
//* ---------------------------
//  SET MEM=PADSTAT                    PROGRAM NAME
//  SET PLICOMP='IBMZ.V3R7.SIBMZCMP'  PLI COMPILER LOADLIB
//  SET DTLIB='EQAW.SEQAMOD'          DEBUG TOOL LOADLIB
//  SET LEHLQ='CEE'                   LE HIGH LVL QUALIFIER
//  SET UNITDEV=SYSALLDA              UNIT FOR TEMP FILES
//  SET LANGX='IPVLANGX'              IPVLANGX UTILITY PROGRAM
//  SET LANGXLIB='IPV.SIPVMODA'       LIBRARY FOR IPVLANGX UTILITY
//*    NOTE: USE THE IPVLANGX FACILITY SHIPPED WITH THE COMMON COMPONENT.
//*
//ALLOCOBJ EXEC PGM=IEFBR14 ALLOC OBJ LIB IF NEEDED
//OBJ DD DSN=&SYSUID..ADLAB.OBJ,SPACE=(CYL,(3,1,15)),
// DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=8000,DISP=(MOD,CATLG)
//*
//* ***************************************
//* COMPILE STEP
//* ***************************************
//COMPILE EXEC PGM=IBMZPLI,REGION=0M,
// PARM=('+DD:OPTIONS')
//* THE +DD:OPTIONS PARAMETER IS USED TO DIRECT THE COMPILER TO
//* GET THE COMPILATION OPTIONS FROM THE OPTIONS DD STATEMENT
//OPTIONS  DD *
TEST(ALL,SYM,NOHOOK,SEPARATE,SEPNAME,AALL),LIST,MAP,SOURCE,
XREF(FULL),NOBLKOFF,AGGREGATE,ATTRIBUTES(FULL),NEST,OPTIONS,NOPT,
STMT,NONUMBER,OFFSET
/*
//*  Note: The above options are for Enterprise PL/I Version 3.7
//*        For Enterprise PL/I Version 3.8+, change the TEST option
//*        to  TEST(ALL,SYM,NOHOOK,SEPARATE,SEPNAME), and add the
//*        LISTVIEW(AALL) option
//STEPLIB  DD DSN=&PLICOMP,DISP=SHR
//         DD DSN=&LEHLQ..SCEERUN,DISP=SHR
//SYSIN    DD DISP=SHR,DSN=&SYSUID..ADLAB.SOURCE(&MEM)
//SYSLIB   DD DISP=SHR,DSN=&SYSUID..ADLAB.COPYLIB
//SYSPRINT DD DISP=SHR,DSN=&SYSUID..ADLAB.ENTPLI.LISTING(&MEM)
//SYSDEBUG DD DISP=SHR,DSN=&SYSUID..ADLAB.SYSDEBUG(&MEM)
//SYSUT1   DD SPACE=(CYL,(5,2),,CONTIG),DCB=BLKSIZE=1024,UNIT=&UNITDEV
//SYSLIN   DD DSN=&SYSUID..ADLAB.OBJ(&MEM),DISP=SHR
//*
//PLIPRINT EXEC PGM=IEBGENER,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=&SYSUID..ADLAB.ENTPLI.LISTING(&MEM),DISP=SHR
//SYSUT2   DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//* *********************************
//* STEP TO GENERATE LANGX FILE
//* *********************************
//LANGX EXEC PGM=&LANGX,REGION=32M,
// PARM='(PLI ERROR 64K CREF'
//STEPLIB  DD DISP=SHR,DSN=&LANGXLIB
//         DD DISP=SHR,DSN=&LEHLQ..SCEERUN
//LISTING  DD DSN=&SYSUID..ADLAB.ENTPLI.LISTING(&MEM),DISP=SHR
//IDILANGX DD DISP=SHR,DSN=&SYSUID..ADLAB.EQALANGX(&MEM)
//*
//* *********************************
//* LINK-EDIT (BINDER) STEP
//* *********************************
//LINK EXEC PGM=IEWL,PARM=(LET,MAP,LIST),REGION=0M
//SYSLIB   DD DSN=&LEHLQ..SCEELKED,DISP=SHR
//DTLIB    DD DSN=&DTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLMOD  DD DISP=SHR,DSN=&SYSUID..ADLAB.LOAD(&MEM)
//SYSUT1   DD UNIT=SYSDA,SPACE=(TRK,(10,10))
//SYSLIN   DD DSN=&SYSUID..ADLAB.OBJ(&MEM),DISP=(OLD,PASS)
//*  INCLUDING A DEBUG TOOL LE EXIT (EQADBCXT, EQADDCXT, EQADICXT OR EQAD3CXT)
//* IS OPTIONAL. THE EXIT ENABLES STARTING DEBUG TOOL WITH THE
//* USER EXIT DATA SET UTILITY (ONE OF THE DEBUG TOOL ISPF UTILITIES)
//*
//* //  DD *
//* INCLUDE DTLIB(EQADBCXT)
```

Enterprise PL/I Version 3.5 and Version 3.6 programs

The following table shows various compiler options that can be used to prepare
| Enterprise PL/I Version 3.5 and Version 3.6 programs for use with the IBM
| Application Delivery Foundation for z Systems family of products (IBM Debug for
| z Systems, IBM Fault Analyzer for z/OS and IBM Application Performance

Analyzer for z/OS). The methods suggested in the following table indicate whether the load module produced is suitable for a production environment. Load modules suitable for a production environment have no significant runtime overhead.

Table 8. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for Enterprise PL/I Version 3.5 and Version 3.6.

Compiler options	**Source information file type produced**	**Is the load module production ready?**	**Options supported and suggested for IBM Debug for z Systems**	**Options supported and suggested for Fault Analyzer for z/OS**	**Options supported and suggested for Application Performance Analyzer for z/OS**
Preprocess (1st stage) to expand source, In compile (2nd stage): For Enterprise PL/I Version 3.5: TEST(ALL, SYM, NOHOOK, SEPARATE), NOPT, AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL) For Enterprise PL/I Version 3.6: TEST(ALL, SYM, NOHOOK, SEPARATE, SEPNAME), NOPT, AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)	SYSDEBUG file used by IBM Debug for z Systems and Fault Analyzer for z/OS. LANGX file used by Application Performance Analyzer for z/OS	Although the module is larger than a module compiled with the NOTEST option, you can use the module in production if needed.	Suggested for test. You can also use these options in a production environment if the increased load module size is not an issue.		

Table 8. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for Enterprise PL/I Version 3.5 and Version 3.6 (continued).

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for IBM Debug for z Systems	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NOTEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)	Compiler listing	Yes	N/A	Supported	N/A
	LANGX file	Yes	N/A	Suggested for production and test	

Preparing Enterprise PL/I Version 3.5 and Version 3.6 programs

Perform the following steps for compiling your Enterprise PL/I Version 3.5 and Version 3.6 programs:

1. Create a library (PDSE is suggested unless PDS is required for your organization) for SYSDEBUG files. This library is only needed in test environments where debugging is performed using `LRECL=(80 to 1024),RECFM=FB,BLKSIZE=(multiple of lrecl < 32K)`.
2. Allocate one or more LANGX libraries for each environment, such as test and production.
3. Create a corresponding LANGX library for each load library. Specify `LRECL=1562 or greater,RECFM=VB,BLKSIZE= lrecl+4 to 32k`.
4. Run a two-stage compile. The first stage preprocesses the program, so the IBM Application Delivery Foundation for z Systems family of products have access to fully expanded source code with INCLUDEs and macros. The second stage compiles the program. For all programs, such as batch, CICS, and IMS:
   - In the first compile stage, in both test and production environments, specify compiler options `MACRO,MDECK,NOCOMPILE,NOSYNTAX,INSOURCE` to expand INCLUDEs and macros. The output SYSPUNCH DD is the input SYSIN DD to the second compile stage.
   - In the second compile stage, in test environments,
     - When using the Enterprise PL/I Version 3.5 compiler:

       For all programs, specify the following compiler options: `TEST(ALL,SYM,NOHOOK,SEPARATE), NOPT, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL).`
     - When using the Enterprise PL/I Version 3.6 compiler:

       For all programs, specify the following compiler options: `TEST(ALL,SYM,NOHOOK,SEPARATE,SEPNAME), NOPT, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL).`

TEST(...) and NOPT are required by zDebug.

The SEPARATE suboption produces a SYSDEBUG file. Save the SYSDEBUG file that is created by the compiler for zDebug (and optionally, Fault Analyzer).

The other options format the compiler listing as required for the IPVLANGX utility.

Consider using TEST(ALL,SYM,NOHOOK,SEPARATE) for best performance and to produce a module that can be debugged. Depending on the policies in your organization, the module can be considered for production.

- In the second compile stage, in production environments, specify compiler options NOTEST, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL).

  Note: The above options can be used with both the Enterprise PL/I Version 3.5 and Version 3.6 compilers.

NOTEST disables zDebug, but produces a smaller load module.

The other options format the compiler listing as required for the IPVLANGX utility to produce a production-ready module that can be used with Fault Analyzer for z/OS and Application Performance Analyzer for z/OS (but not IBM Debug for z Systems).

5. When a TEST(...SEPARATE) parm is used, code a SYSDEBUG DD in the second compiler step as follows:

```
//SYSDEBUG DD DSN= sysdebug.pds(pgmname),DISP=SHR
```

   This is the source information file for IBM Debug for z Systems, IBM Application Performance Analyzer for z/OS and optionally, IBM Fault Analyzer for z/OS. Save it in the SYSDEBUG library, and specify a member name that is equal to the primary entry point name or CSECT name of your application program.
6. Modify the SYSPRINT DD in the second compiler stage. This file is the compiler listing. Write the listing to either a permanent or temporary file. This file is the input to the IPVLANGX utility.

   Note: This compiler typically renames CSECTs according to an internal compiler algorithm. Therefore, it is not recommended to store PL/I compiler listings or side files using CSECT names as they might not be found by IBM Application Performance Analyzer for z/OS or IBM Fault Analyzer for z/OS. Instead, use the primary entry point name.
7. Add a step after the compile step to run the IPVLANGX utility. This utility reads the compiler listing and generates a LANGX file. This file is the source information file for IBM Fault Analyzer for z/OS and IBM Application Performance Analyzer for z/OS. Save the LANGX file in the LANGX file library, and specify a member name that is equal to the primary entry point name of your application program.
8. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.
9. If you compile with the TEST option and promote these modules into production, promote the SYSDEBUG file for your production environment.

10. Optionally, include a zDebug Language Environment exit module into the load module during the linkage editor step. This approach is one way to enable zDebug's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

    You can also use module EQAD3CXT for batch programs, IMS/TM, IMS BTS programs, and DB2 type MAIN stored procedures

Sample JCL for compiling Enterprise PL/I Version 3.5 or Version 3.6 programs

Here is a JCL example for compiling an Enterprise PL/I for z/OS Version 3.5 or Version 3.6 program for use with the IBM Problem Determination Tools products.

```
//*     - - -  ADD A JOB CARD ABOVE THIS LINE  - - -
//*
//* SAMPLE JCL TO PREPARE AN ENTERPRISE PL/I V3.5 OR
//* ENTERPRISE PL/I V3.6 PROGRAM FOR THE IBM ZSERIES
//* FOR THE IBM ZSERIES PD TOOLS PRODUCTS:
//* FAULT ANALYZER, DEBUG TOOL, AND APPLICATION PERF. ANALYZER
//*
//* NOTES:
//*
//* COMPILER:
//* 1. A 2-STAGE COMPILE IS PERFORMED. STAGE 1 (PREPROCESS) IS
//*    DONE TO EXPAND INCLUDES AND MACROS IN THE PROGRAM, SO THAT
//*    THE SYSDEBUG FILE CREATED IN STAGE 2 (COMPILE) HAS ALL STMTS.
//* 2. COMPILER PARMS TEST AND NOPT ARE REQUIRED FOR DEBUG TOOL
//* 3. COMPILER PARM TEST(ALL,SYM,NOHOOK,SEP) (V3.5) OR
//*    TEST(ALL,SYM,NOHOOK,SEP,SEPNAME) (V3.6) IS USED BECAUSE:
//*    - THE MODULE IS READY FOR DEBUG TOOL
//*         - NOHOOK DOES NOT HAVE RUN-TIME CPU OVERHEAD. HOWEVER, THE
//*           MODULE IS LARGER BECAUSE OF STATEMENT TABLE
//*         - A SYSDEBUG FILE IS CREATED THAT CAN BE USED BY DT,FA,APA
//*    4. COMPILER PARMS AGGREGATE,ATTRIBUTES(FULL),NOBLKOFF,LIST,
//*       MAP,NEST,NONUMBER,OPTIONS,SOURCE,STMT,XREF(FULL) ARE NEEDED
//*       TO PROCESS THE COMPILER LISTING WITH IPVLANGX
//*
//*   BINDER (LINKAGE EDITOR):
//*    5. THE INCLUDE FOR MODULE EQAD?CXT IS OPTIONAL.  IT IS AN
//*       LE EXIT MODULE THAT CAN BE USED TO START DEBUG TOOL.
//*       UNDERSTAND THE METHODS AVAILABLE FOR STARTING DEBUG TOOL,
//*       AND CHOOSE WHETHER YOU WANT TO USE THE LE EXITS.
//*         IF YOU USE THIS METHOD, LOAD THE CORRECT EXIT MODULE:
//*            EQADBCXT: FOR BATCH PROGRAMS
//*            EQADICXT: FOR ONLINE IMS PROGRAMS
//*            EQADDCXT: FOR DB2 STORED PROCEDURES (OF TYPE MAIN AND SUB)
//*           (for SUB this is supported only for invocations through call_sub)
//*            (DO NOT INCLUDE AN EXIT FOR CICS PROGRAMS)
//*     YOU CAN ALSO USE MODULE EQAD3CXT FOR BATCH PROGRAMS, ONLINE IMS
//*          PROGRAMS, DB2 TYPE MAIN STORED PROCEDURES.
//*  SET PARMS FOR THIS COMPILE:
//*  ---------------------------
//   SET MEM=PADSTAT                        PROGRAM NAME
//   SET PLICOMP='IBMZ.V3R5.SIBMZCMP'       PLI COMPILER LOADLIB
//   SET DTLIB='EQAW.SEQAMOD'               DEBUG TOOL LOADLIB
//   SET LEHLQ='CEE'                        LE HIGH LVL QUALIFIER
//   SET UNITDEV=SYSALLDA                   UNIT FOR TEMP FILES
//   SET LANGX='IPVLANGX'                   IPVLANGX UTILITY PROGRAM
//   SET LANGXLIB='IPV.SIPVMODA'            LIBRARY FOR IPVLANGX UTILITY
//*    NOTE: USE THE IPVLANGX FACILITY SHIPPED WITH THE COMMON COMPONENT.
//*
//ALLOCOBJ EXEC PGM=IEFBR14                ALLOC OBJ LIB IF NEEDED
```

```
//OBJ   DD DSN=&SYSUID..ADLAB.OBJ,SPACE=(CYL,(3,1,15)),
//  DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=8000,DISP=(MOD,CATLG)
//*  ***************************************
//*     PREPROCESS STEP (COMPILE STAGE 1)
//*  ***************************************
//PRECOMP  EXEC PGM=IBMZPLI,REGION=0M,
//   PARM=('MACRO,MDECK,NOCOMPILE,NOSYNTAX,INSOURCE')
//STEPLIB  DD   DSN=&PLICOMP,DISP=SHR
//         DD   DSN=&LEHLQ..SCEERUN,DISP=SHR
//SYSIN    DD   DISP=SHR,DSN=&SYSUID..ADLAB.SOURCE(&MEM)
//SYSLIB   DD   DISP=SHR,DSN=&SYSUID..ADLAB.COPYLIB
//SYSPRINT DD   SYSOUT=*
//SYSUT1   DD   SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024,
//             UNIT=&UNITDEV
//SYSPUNCH DD DISP=(MOD,PASS),DSN=&&SRC1,UNIT=&UNITDEV,
//            SPACE=(80,(10,10))
//*
//*  ***************************************
//* COMPILE STEP (COMPILE STAGE 2)
//* ***************************************
//COMPILE EXEC PGM=IBMZPLI,REGION=0M,
// PARM=('+DD:OPTIONS')
//* THE +DD:OPTIONS PARAMETER IS USED TO DIRECT THE COMPILER TO
//* GET THE COMPILATION OPTIONS FROM THE OPTIONS DD STATEMENT
//OPTIONS  DD *
TEST(ALL,SYM,NOHOOK,SEPARATE),LIST,MAP,SOURCE,XREF(FULL),
NOBLKOFF,AGGREGATE,ATTRIBUTES(FULL),NEST,OPTIONS,NOPT,
STMT,NONUMBER,OFFSET
/*
//*  Note: The above options are for Enterprise PL/I Version 3.5
//*        For Enterprise PL/I Version 3.6, change the TEST option
//*        to:  TEST(ALL,SYM,NOHOOK,SEPARATE,SEPNAME)
//STEPLIB  DD DSN=&PLICOMP,DISP=SHR
//         DD DSN=&LEHLQ..SCEERUN,DISP=SHR
//SYSIN    DD DSN=&&SRC1,DISP=(OLD,PASS)
//SYSLIB   DD DISP=SHR,DSN=&SYSUID..ADLAB.COPYLIB
//SYSPRINT DD DISP=SHR,DSN=&SYSUID..ADLAB.ENTPLI.LISTING(&MEM)
//SYSDEBUG DD DISP=SHR,DSN=&SYSUID..ADLAB.SYSDEBUG(&MEM)
//SYSUT1   DD SPACE=(CYL,(5,2),,CONTIG),DCB=BLKSIZE=1024,UNIT=&UNITDEV
//SYSLIN   DD DSN=&SYSUID..ADLAB.OBJ(&MEM),DISP=SHR
//*
//PLIPRINT  EXEC PGM=IEBGENER,REGION=0M
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSN=&SYSUID..ADLAB.ENTPLI.LISTING(&MEM),DISP=SHR
//SYSUT2    DD SYSOUT=*
//SYSIN     DD DUMMY
//*
//*  *********************************
//*     STEP TO GENERATE LANGX FILE
//*  *********************************
//LANGX    EXEC PGM=&LANGX,REGION=32M,
//  PARM='(PLI ERROR 64K CREF'
//STEPLIB  DD DISP=SHR,DSN=&LANGXLIB
//         DD DISP=SHR,DSN=&LEHLQ..SCEERUN
//LISTING  DD DSN=&SYSUID..ADLAB.ENTPLI.LISTING(&MEM),DISP=SHR
//IDILANGX DD DISP=SHR,DSN=&SYSUID..ADLAB.EQALANGX(&MEM)
//*
//* *********************************
//* LINK-EDIT (BINDER) STEP
//* *********************************
//LINK EXEC PGM=IEWL,PARM=(LET,MAP,LIST),REGION=0M
//SYSLIB DD DSN=&LEHLQ..SCEELKED,DISP=SHR
//DTLIB DD DSN=&DTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLMOD DD DISP=SHR,DSN=&SYSUID..ADLAB.LOAD(&MEM)
//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(10,10))
//SYSLIN DD DSN=&SYSUID..ADLAB.OBJ(&MEM),DISP=(OLD,PASS)
```

```
//*  INCLUDING A DEBUG TOOL LE EXIT (EQADBCXT, EQADDCXT, EQADICXT OR EQAD3CXT)
//*  IS OPTIONAL.  THE EXIT ENABLES STARTING DEBUG TOOL WITH THE
//*  USER EXIT DATA SET UTILITY (ONE OF THE DEBUG TOOL ISPF UTILITIES)
//*  //       DD *
//*   INCLUDE DTLIB(EQADBCXT)
```

Enterprise PL/I Version 3.4 and earlier programs

The following table shows various compiler options that can be used to prepare Enterprise PL/I Version 3.4 and earlier programs for use with the IBM Application Delivery Foundation for z Systems family of products (IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate whether the load module produced is suitable for a production environment. Load modules suitable for a production environment have no significant runtime overhead.

Table 9. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for Enterprise PL/I Version 3.4 and earlier.

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for IBM Debug for z Systems	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
Preprocess (1st stage) to expand source, In compile (2nd stage): TEST(ALL), NOPT, AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL))	Expanded source file used by IBM Debug for z Systems, LANGX file used by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS	No	Suggested for test. (Using zDebug in production for this compiler is not recommended.)		
AGGREGATE, ATTRIBUTES (FULL), NOBLKOFF, LIST, MAP, NEST, NOTEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL))	Compiler listing	Yes	N/A	Supported	N/A
	LANGX file	Yes	N/A	Suggested for production and test	

Preparing Enterprise PL/I Version 3.4 and earlier programs

Perform the following steps for compiling your Enterprise PL/I Version 3.4 and earlier programs:

1. Create a library (PDSE is suggested unless PDS is required for your organization) for expanded source files. This library is only needed in test environments where debugging is performed. The library can be any RECFM / LRECL / BLKSIZE supported as input by the compiler.
2. Allocate libraries (PDSE is suggested unless PDS is required for your organization) for LANGX files. Allocate one or more LANGX libraries for each environment, such as test or production.
3. Create a corresponding LANGX library for each load library. Specify `LRECL=1562 or greater,RECFM=VB,BLKSIZE= lrecl+4 to 32k`.
4. Run a 2–stage compile. The first stage preprocesses the program, so the IBM Application Delivery Foundation for z Systems family of products have access to fully expanded source code with INCLUDEs and macros. The second stage compiles the program.
   - In the first compile stage, in both test and production environments:
     - Specify compiler options `MACRO,MDECK,NOCOMPILE,NOSYNTAX,INSOURCE` to expand INCLUDEs and macros.
     - Save the output, the expanded source file, in a permanent file in the expanded source file library and specify *member name = program name*. This file is the source information file for IBM Debug for z Systems. The output SYSPUNCH DD is the input SYSIN DD to the second compiler stage.
   - In the second compile stage, for all programs, such as batch, CICS, and IMS:
     - In test environments, specify compiler options `TEST(ALL), NOPT, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL)`.

       `TEST(ALL)` and `NOPT` are required by zDebug. Debug hooks are inserted, which add some runtime overhead. Symbolic data that is required by zDebug is also stored in the module, which can make it larger.

       The other options format the compiler listing as required for the IPVLANGX utility.
     - In production environments, specify compiler options `NOTEST, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OFFSET, OPTIONS, SOURCE, STMT, XREF(FULL))`.

       `NOTEST` disables DzDebug, but provides the best performance. This produces a production-ready module that can be used with Fault Analyzer for z/OS and Application Performance Analyzer for z/OS (but not zDebug).

       The other options format the compiler listing as required for the IPVLANGX utility.
5. Modify the SYSPRINT DD in the second compiler stage. This file is the compiler listing. Save the compiler listing to either a permanent or temporary file. This file is the input to the IPVLANGX utility.

   Note: This compiler typically renames CSECTs according to an internal compiler algorithm. Therefore, it is not recommended to store PL/I compiler listings or side files using CSECT names as they might not be found by Application Performance Analyzer for z/OS or Fault Analyzer for z/OS. Instead, use the primary entry point name.
6. Add a step after the compiler step to run the IPVLANGX utility. The IPVLANGX utility reads the compiler listing and generates a LANGX file,

which is the source information file for Fault Analyzer for z/OS and Application Performance Analyzer for z/OS. Save the LANGX file in the LANGX file library, and specify a member name that is equal to the primary entry point name or CSECT name of your application program.

7. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.
8. Optionally, include a zDebug Language Environment exit module into the load module during the linkage editor step. This approach is one way to enable zDebug's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

   You can also use module EQAD3CXT for batch programs, IMS/TM, IMS BTS programs, and DB2 type MAIN stored procedures
9. For CICS applications only, if the zDebug DTCN transaction is used to start zDebug, link edit the zDebug CICS startup exit module EQADCCXT into the application load module to enable zDebug in CICS. This link edit is not needed if using the CADP transaction instead of DTCN.

Sample JCL for compiling Enterprise PL/I for z/OS Version 3.4 or earlier programs

Here is a JCL example for compiling an Enterprise PL/I for z/OS Version 3.4 or earlier program for use with the IBM Application Delivery Foundation for z Systems family of products.

```
//*     - - -  ADD A JOB CARD ABOVE THIS LINE  - - -
//*
//*  SAMPLE JCL TO COMPILE WITH ENTERPRISE PLI V3.4 AND PREVIOUS
//*  FOR THE IBM ZSERIES PD TOOLS PRODUCTS:
//*     FAULT ANALYZER, DEBUG TOOL, AND APPLICATION PERF. ANALYZER
//*
//*  NOTES:
//*
//*   COMPILER:
//*    1. A 2-STAGE COMPILE IS PERFORMED.  STAGE 1 (PREPROCESS) IS
//*       DONE TO EXPAND INCLUDES AND MACROS IN THE PROGRAM, SO THAT
//*       A SOURCE FILE IS CREATED FOR DEBUG TOOL THAT HAS ALL STMTS.
//*    2. COMPILER PARM TEST AND NOPT ARE REQUIRED FOR DEBUG TOOL
//*    3. COMPILER PARMS AGGREGATE,ATTRIBUTES(FULL),NOBLKOFF,LIST,
//*       MAP,NEST,NONUMBER,OPTIONS,SOURCE,STMT,XREF(FULL) ARE NEEDED
//*       TO PROCESS THE COMPILER LISTING WITH IPVLANGX
//*
//*   BINDER (LINKAGE EDITOR):
//*    4. THE INCLUDE FOR MODULE EQAD?CXT IS OPTIONAL.  IT IS AN
//*       LE EXIT MODULE THAT CAN BE USED TO START DEBUG TOOL.
//*       UNDERSTAND THE METHODS AVAILABLE FOR STARTING DEBUG TOOL,
//*       AND CHOOSE WHETHER YOU WANT TO USE THE LE EXITS.
//*         IF YOU USE THIS METHOD, LOAD THE CORRECT EXIT MODULE:
//*            EQADBCXT: FOR BATCH PROGRAMS
//*            EQADICXT: FOR ONLINE IMS PROGRAMS
//*            EQADDCXT: FOR DB2 STORED PROCEDURES (OF TYPE MAIN AND SUB)
//*           (for SUB this is supported only for invocations through call_sub)
//*            (DO NOT INCLUDE AN EXIT FOR CICS PROGRAMS)
//*     YOU CAN ALSO USE MODULE EQAD3CXT FOR BATCH PROGRAMS, ONLINE IMS
//*          PROGRAMS, DB2 TYPE MAIN STORED PROCEDURES.
//*
//*  SET PARMS FOR THIS COMPILE:
```

```
//*  --------------------------
//   SET MEM=PTEST                        PROGRAM NAME
//   SET PLICOMP='IBMZ.V3R4.SIBMZCMP'     PLI COMPILER LOADLIB
//   SET DTLIB='EQAW.SEQAMOD'             DEBUG TOOL LOADLIB
//   SET LEHLQ='CEE'                      LE HIGH LVL QUALIFIER
//   SET UNITDEV=SYSALLDA                 UNIT FOR TEMP FILES
//   SET LANGX='IPVLANGX'                 IPVLANGX UTILITY PROGRAM
//   SET LANGXLIB='IPV.SIPVMODA'          LIBRARY FOR IPVLANGX UTILITY
//*    NOTE: USE THE IPVLANGX FACILITY SHIPPED WITH THE COMMON COMPONENT.
//*
//ALLOCOBJ EXEC PGM=IEFBR14              ALLOC OBJ LIB IF NEEDED
//XSOURCE DD DSN=&SYSUID..ADLAB.EXPANDED.SOURCE,SPACE=(CYL,(3,1,15)),
//  DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=8000,DISP=(MOD,CATLG)
//OBJ   DD DSN=&SYSUID..ADLAB.OBJ,SPACE=(CYL,(3,1,15)),
//  DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=8000,DISP=(MOD,CATLG)
//*  ***************************************
//*     PREPROCESS STEP (COMPILE STAGE 1)
//*  ***************************************
//PRECOMP  EXEC PGM=IBMZPLI,REGION=0M,
//   PARM=('MACRO,MDECK,NOCOMPILE,NOSYNTAX,INSOURCE')
//STEPLIB  DD   DSN=&PLICOMP,DISP=SHR
//         DD   DSN=&LEHLQ..SCEERUN,DISP=SHR
//SYSIN    DD   DISP=SHR,DSN=&SYSUID..ADLAB.SOURCE(&MEM)
//SYSLIB   DD   DISP=SHR,DSN=&SYSUID..ADLAB.COPYLIB
//SYSPRINT DD   SYSOUT=*
//SYSUT1   DD   SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024,
//             UNIT=&UNITDEV
//SYSPUNCH DD DISP=SHR,DSN=&SYSUID..ADLAB.EXPANDED.SOURCE(&MEM)
//*
//*  ***************************************
//*     COMPILE STEP (COMPILE STAGE 2)
//*  ***************************************
//COMPILE EXEC PGM=IBMZPLI,REGION=0M,
// PARM=('+DD:OPTIONS')
//* THE +DD:OPTIONS PARAMETER IS USED TO DIRECT THE COMPILER TO
//* GET THE COMPILATION OPTIONS FROM THE OPTIONS DD STATEMENT
//OPTIONS  DD *
TEST(ALL),LIST,MAP,SOURCE,XREF(FULL),
NOBLKOFF,AGGREGATE,ATTRIBUTES(FULL),NEST,OPTIONS,NOPT,
STMT,NONUMBER,OFFSET
/*
//STEPLIB  DD DSN=&PLICOMP,DISP=SHR
//         DD DSN=&LEHLQ..SCEERUN,DISP=SHR
//SYSIN    DD DISP=SHR,DSN=&SYSUID..ADLAB.EXPANDED.SOURCE(&MEM)
//SYSLIB   DD DISP=SHR,DSN=&SYSUID..ADLAB.COPYLIB
//SYSPRINT DD DISP=SHR,DSN=&SYSUID..ADLAB.ENTPLI.LISTING(&MEM)
//SYSUT1   DD SPACE=(CYL,(5,2),,CONTIG),DCB=BLKSIZE=1024,UNIT=&UNITDEV
//SYSLIN   DD DSN=&SYSUID..ADLAB.OBJ(&MEM),DISP=SHR
//*
//PLIPRINT  EXEC PGM=IEBGENER,REGION=0M
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSN=&SYSUID..ADLAB.ENTPLI.LISTING(&MEM),DISP=SHR
//SYSUT2    DD SYSOUT=*
//SYSIN     DD DUMMY
//*
//*  *********************************
//*     STEP TO GENERATE LANGX FILE
//*  *********************************
//LANGX    EXEC PGM=&LANGX,REGION=32M,
//  PARM='(PLI ERROR 64K CREF'
//STEPLIB  DD DISP=SHR,DSN=&LANGXLIB
//         DD DISP=SHR,DSN=&LEHLQ..SCEERUN
//LISTING  DD DSN=&SYSUID..ADLAB.ENTPLI.LISTING(&MEM),DISP=SHR
//IDILANGX DD DISP=SHR,DSN=&SYSUID..ADLAB.EQALANGX(&MEM)
//*
//* *********************************
//* LINK-EDIT (BINDER) STEP
//* *********************************
//LINK EXEC PGM=IEWL,PARM=(LET,MAP,LIST),REGION=0M
//SYSLIB DD DSN=&LEHLQ..SCEELKED,DISP=SHR
//DTLIB DD DSN=&DTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLMOD DD DISP=SHR,DSN=&SYSUID..ADLAB.LOAD(&MEM)
//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(10,10))
//SYSLIN DD DSN=&SYSUID..ADLAB.OBJ(&MEM),DISP=(OLD,PASS)
//*  INCLUDING A DEBUG TOOL LE EXIT (EQADBCXT, EQADDCXT, EQADICXT OR EQAD3CXT)
//*  IS OPTIONAL.  THE EXIT ENABLES STARTING DEBUG TOOL WITH THE
//*  USER EXIT DATA SET UTILITY (ONE OF THE DEBUG TOOL ISPF UTILITIES)
//*  //       DD *
//*   INCLUDE DTLIB(EQADBCXT)
```

PL/I for MVS and VM and OS PL/I programs

The following table shows various compiler options that can be used to prepare PL/I for MVS and VM and OS PL/I programs for use with the IBM Application Delivery Foundation for z Systems family of products (IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate whether the load module produced is suitable for a production environment. Load modules suitable for a production environment have no significant runtime overhead.

For the test environment, you need both the listing and the LANGX file (for Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). In production, only the LANGX file is suggested.

Table 10. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for PL/I for MVS and VM and OS PLI.

Compiler options	Source information file type produced	Is the load module production ready?	Options supported and suggested for IBM Debug for z Systems	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
TEST(ALL), AGGREGATE, ATTRIBUTES (FULL), ESD, LIST, MAP, NEST, NOPT, OPTIONS, SOURCE, STMT, XREF(FULL)	Compiler listing	No	Suggested for test. Using z/OS Debugger in production for this compiler is not recommended.	Supported	Supported
	LANGX file	No	N/A	Supported	N/A
NOTEST, AGGREGATE, ATTRIBUTES (FULL), ESD, LIST, MAP, NEST, OPTIONS, SOURCE, STMT, XREF(FULL)	Compiler listing	Yes	N/A	Supported	Suggested for production and test
	LANGX file	Yes	N/A	Suggested for production and test	N/A

Perform the following steps for compiling your PL/I for MVS and VM and OS PL/I programs:

1. Create a library (PDSE is suggested unless PDS is required for your organization) for compiler listing files. This library is only needed in test environments where debugging is performed. Specify `LRECL=125 minimum,RECFM=VBA,BLKSIZE= lrecl+4 to 32k`.
2. Allocate libraries (PDSE is suggested unless PDS is required for your organization) for LANGX files. Allocate one or more LANGX libraries for each environment, such as test and production.
3. Create a corresponding LANGX library for each load library. Specify `LRECL=1562 or greater,RECFM=VB,BLKSIZE= lrecl+4 to 32k`.
4. For all programs, such as batch, CICS, and IMS:
   - In test environments, specify compiler options `TEST(ALL)`, `NOPT`, `AGGREGATE`, `ATTRIBUTES(FULL)`, `ESD`, `LIST`, `MAP`, `NEST`, `OPTIONS`, `SOURCE`, `STMT`, `XREF(FULL)`.

     `TEST(ALL)` and `NOOPT` are required by zDebug. `TEST` adds debug hooks, which add some runtime overhead. Symbolic data that is required by zDebug is stored in the module, which can make it larger.

     The other options format the compiler listing as required by zDebug and by the IPVLANGX utility.
   - In production environments, specify compiler options `NOTEST`, `AGGREGATE`, `ATTRIBUTES(FULL)`, `ESD`, `LIST`, `MAP`, `NEST`, `OPTIONS`, `SOURCE`, `STMT`, `XREF(FULL)`.

     `NOTEST` disables zDebug, but provides the best performance.

     The other options format the compiler listing as required for the IPVLANGX utility.

     This produces a production-ready module that can be used with Fault Analyzer for z/OS and Application Performance Analyzer for z/OS but not IBM Debug for z Systems.
5. Modify the SYSPRINT DD in the compiler step. This parameter is the compiler listing. Save this to a permanent file. The compiler listing is the input to the IPVLANGX utility and is the source information file for IBM Debug for z Systems.

   Note: This compiler typically renames CSECTs according to an internal compiler algorithm. Therefore, it is not recommended to store PL/I compiler listings or side files using CSECT names as they might not be found by Application Performance Analyzer for z/OS or Fault Analyzer for z/OS. Instead, use the primary entry point name.
6. Add a step after the compiler step to run the IPVLANGX utility. This utility reads the compiler listing and saves a LANGX file. This file is the source information file for Fault Analyzer for z/OS and Application Performance Analyzer for z/OS. Save it in the LANGX file library and specify a member name that is equal to the primary entry point name of your application program.
7. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.
8. Optionally, include a zDebug Language Environment exit module into the load module during the linkage editor step. This approach is one way to enable

zDebug's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

You can also use module EQAD3CXT for batch programs, IMS/TM, IMS BTS programs, and DB2 type MAIN stored procedures

9. For CICS applications only, if the zDebug DTCN transaction is used to start zDebug, link edit the zDebug CICS startup exit module EQADCCXT into the application load module to enable zDebug in CICS. This link edit is not needed if using the CADP transaction instead of DTCN.

Preparing PL/I for MVS and VM and OS PL/I programs

Sample JCL for compiling PL/I for MVS and VM programs

Here is a JCL example for compiling an PL/I for MVS and VM program for use with the IBM Application Delivery Foundation for z Systems family of products.

```
//*     - - -  ADD A JOB CARD ABOVE THIS LINE  - - -
//*
//*  SAMPLE JCL TO PREPARE A PLI FOR MVS AND VM PROGRAM
//*  FOR THE IBM ZSERIES PD TOOLS PRODUCTS:
//*     FAULT ANALYZER, DEBUG TOOL, AND APPLICATION PERF. ANALYZER
//*
//*  NOTES:
//*
//*   COMPILER:
//*    1. COMPILER PARM TEST IS REQUIRED FOR DEBUG TOOL
//*    2. COMPILER PARMS AGGREGATE,ATTRIBUTES(FULL),ESD,LIST,
//*       MAP,NEST,OPTIONS,SOURCE,STMT,XREF(FULL) ARE NEEDED
//*       FOR PD TOOLS TO PROCESS THE COMPILER LISTING
//*
//*   BINDER (LINKAGE EDITOR):
//*    3. THE INCLUDE FOR MODULE EQAD?CXT IS OPTIONAL.  IT IS AN
//*       LE EXIT MODULE THAT CAN BE USED TO START DEBUG TOOL.
//*       UNDERSTAND THE METHODS AVAILABLE FOR STARTING DEBUG TOOL,
//*       AND CHOOSE WHETHER YOU WANT TO USE THE LE EXITS.
//*         IF YOU USE THIS METHOD, LOAD THE CORRECT EXIT MODULE:
//*            EQADBCXT: FOR BATCH PROGRAMS
//*            EQADICXT: FOR ONLINE IMS PROGRAMS
//*            EQADDCXT: FOR DB2 STORED PROCEDURES (OF TYPE MAIN AND SUB)
//*           (for SUB this is supported only for invocations through call_sub)
//*            (DO NOT INCLUDE AN EXIT FOR CICS PROGRAMS)
//*     YOU CAN ALSO USE MODULE EQAD3CXT FOR BATCH PROGRAMS, ONLINE IMS
//*          PROGRAMS, DB2 TYPE MAIN STORED PROCEDURES.
//*  SET PARMS FOR THIS COMPILE:
//*  ---------------------------
//   SET MEM=PADSTAT                      PROGRAM NAME
//   SET PLICOMP='IEL.V1R1M1.SIELCOMP'    PLI COMPILER LOADLIB
//   SET DTLIB='EQAW.SEQAMOD'             DEBUG TOOL LOADLIB
//   SET LEHLQ='CEE'                      LE HIGH LVL QUALIFIER
//   SET UNITDEV=SYSALLDA                 UNIT FOR TEMP FILES
//   SET LANGX='IPVLANGX'                 IPVLANGX UTILITY PROGRAM
//   SET LANGXLIB='IPV.SIPVMODA'          LIBRARY FOR IPVLANGX UTILITY
//*    NOTE: USE THE IPVLANGX FACILITY SHIPPED WITH THE COMMON COMPONENT.
//*
//ALLOCOBJ EXEC PGM=IEFBR14              ALLOC OBJ LIB IF NEEDED
//OBJ   DD DSN=&SYSUID..ADLAB.OBJ,SPACE=(CYL,(3,1,15)),
//  DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=8000,DISP=(MOD,CATLG)
//*
//*  **************************************
//*     COMPILE STEP
//*  **************************************
//*
```

```
//COMPILE  EXEC PGM=IEL1AA,REGION=6M,
// PARM=('TEST(ALL),NOPT,AGGREGATE,ATTRIBUTES(FULL),ESD,LIST,MAP,',
//       'NEST,OPTIONS,SOURCE,STMT,XREF(FULL)')
//STEPLIB  DD   DSN=&PLICOMP,DISP=SHR
//SYSIN    DD   DISP=SHR,DSN=&SYSUID..ADLAB.SOURCE(&MEM)
//SYSLIB   DD   DISP=SHR,DSN=&SYSUID..ADLAB.COPYLIB
//SYSPRINT DD   DISP=SHR,DSN=&SYSUID..ADLAB.PLIMVS.LISTING(&MEM)
//SYSUT1   DD   SPACE=(CYL,(1,1)),UNIT=SYSDA
//SYSLIN   DD DSN=&SYSUID..ADLAB.OBJ(&MEM),DISP=SHR
//*
//PLIPRINT  EXEC PGM=IEBGENER,REGION=0M
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSN=&SYSUID..ADLAB.PLIMVS.LISTING(&MEM),DISP=SHR
//SYSUT2    DD SYSOUT=*
//SYSIN     DD DUMMY
//*
//*  *********************************
//*     STEP TO GENERATE LANGX FILE
//*  *********************************
//LANGX    EXEC PGM=&LANGX,REGION=32M,
//  PARM='(PLI ERROR 64K CREF'
//STEPLIB  DD DISP=SHR,DSN=&LANGXLIB
//         DD DISP=SHR,DSN=&LEHLQ..SCEERUN
//LISTING  DD DSN=&SYSUID..ADLAB.PLIMVS.LISTING(&MEM),DISP=SHR
//IDILANGX DD DISP=SHR,DSN=&SYSUID..ADLAB.EQALANGX(&MEM)
//*
//* *********************************
//* LINK-EDIT (BINDER) STEP
//* *********************************
//LINK EXEC PGM=IEWL,PARM=(LET,MAP,LIST),REGION=0M
//SYSLIB DD DSN=&LEHLQ..SCEELKED,DISP=SHR
//DTLIB DD DSN=&DTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLMOD DD DISP=SHR,DSN=&SYSUID..ADLAB.LOAD(&MEM)
//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(10,10))
//SYSLIN DD DSN=&SYSUID..ADLAB.OBJ(&MEM),DISP=(OLD,PASS)
//*  INCLUDING A DEBUG TOOL LE EXIT (EQADBCXT, EQADDCXT, EQADICXT OR EQAD3CXT)
//*  IS OPTIONAL.  THE EXIT ENABLES STARTING DEBUG TOOL WITH THE
//*  USER EXIT DATA SET UTILITY (ONE OF THE DEBUG TOOL ISPF UTILITIES)
//*  //       DD *
//*   INCLUDE DTLIB(EQADBCXT)
```

z/OS XL C and C++ programs

The following table shows various compiler options that can be used to prepare z/OS XL C and C++ programs for use with the IBM Application Delivery Foundation for z Systems family of products (IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate whether the load module produced is suitable for a production environment. Load modules suitable for a production environment have no significant runtime overhead.

Table 11. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for C and C++.

Compiler options	Output produced	Options supported and suggested for IBM Debug for z Systems	Options supported and suggested for Fault Analyzer for z/OS	Options supported and suggested for Application Performance Analyzer for z/OS
Preprocess (1st stage) to expand source: PP(COMMENTS, NOLINES) Compile (2nd stage): DEBUG (FORMAT (DWARF), NOHOOK, SYMBOL, FILE (location)), LIST, LONGNAME, NOOFFSET [1]	• Expanded source file • DWARF file (used by IBM Debug for z Systems and Fault Analyzer for z/OS) • Compiler listing (used by Application Performance Analyzer for z/OS)	Supported. You can use it for production if the OPT compile option is not used. Full functionality available.	Supported. Use of the OPT compile option might result in incorrect source line being reported. Full functionality available.	Supported.
	• .mdbg file[2]	Recommended. You can use it for production if the OPT compile option is not used. Full functionality available.	Supported. Use of the OPT compile option might result in incorrect source line being reported. Full functionality available.	Not supported.
Preprocess (1st stage) to expand source: PP(COMMENTS, NOLINES) Compile (2nd stage): DEBUG (FORMAT (DWARF), HOOK (LINE, NOBLOCK, PATH), SYMBOL, FILE (location)), LIST, LONGNAME, NOOPT, NOOFFSET [1]	• Expanded source file • DWARF file (used by IBM Debug for z Systems and Fault Analyzer for z/OS) • Compiler listing (used by Application Performance Analyzer for z/OS)	Supported. Full functionality available. Use in production not recommended.	Supported. Full functionality available.	Supported.
	• .mdbg file[2]	Supported. Full functionality available. Use in production not recommended.	Supported. Full functionality available.	Not supported.

Table 11. Examples of compiler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for C and C++ (continued).

Compiler options	**Output produced**	**Options supported and suggested for IBM Debug for z Systems**	**Options supported and suggested for Fault Analyzer for z/OS**	**Options supported and suggested for Application Performance Analyzer for z/OS**
Preprocess (1st stage) to expand source: PP(COMMENTS, NOLINES) Compile (2nd stage): TEST, AGGREGATE[3], NOIPA, LIST, NESTINC (255), NOOFFSET, NOOPT, SOURCE, XREF, LONGNAME	• Expanded source file (used by IBM Debug for z Systems) • Compiler listing (used by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS)	Supported. Use in production not recommended.	Supported. Use in production not recommended. Variables not reported.	Supported. Use in production not recommended.
	• Expanded source file (used by IBM Debug for z Systems) • LANGX file (used by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS)	Supported. Use in production not recommended.	Supported. Use in production not recommended. Variables not reported.	Supported. Use in production not recommended.
NOTEST, AGGREGATE[3], NOIPA, LIST, NESTINC (255), NOOFFSET, NOOPT, SOURCE, XREF, LONGNAME	• Compiler listing (used by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS)	Not supported.	Supported. Suggested for production and test. Variables not reported.	Supported. Suggested for production and test.
	• LANGX file (used by Fault Analyzer for z/OS and Application Performance Analyzer for z/OS)	Not supported.	Supported. Suggested for production and test. Variables not reported.	Supported. Suggested for production and test.
UNIX System Services compile -g	• DWARF file (used by IBM Debug for z Systems, Fault Analyzer for z/OS, and Application Performance Analyzer for z/OS)	Supported.	Supported.	Supported.
	• .mdbg file[2]	Supported.	Supported.	Not supported.

Note:
1. The FORMAT(DWARF) option is supported for z/OS Version 1.6 and higher.
2. For C and C++ programs that are compiled with z/OS XL C/C++, Version 1.10 or later, if you specify the FORMAT(DWARF) suboption of the DEBUG compiler option, the load modules are smaller and you can create .mdbg files with captured source using the CDADBGLD utility.

   zDebug needs only the .mdbg file to debug your program.
3. For C++, do not use the AGGREGATE keyword. Use ATTRIBUTES instead.

Preparing z/OS XL C and C++ programs

Perform the following steps for compiling your z/OS XL C and C++ programs:

1. Create a library (PDSE is suggested unless PDS is required for your organization) for expanded source files. This library is only needed in test environments where debugging is performed. This can be any RECFM / LRECL / BLKSIZE supported as input by the compiler.
2. Allocate libraries (PDSE is suggested unless PDS is required for your organization) for compiler listing files. Allocate one or more compiler listing libraries for each environment, such as test and production.
3. Create a corresponding listing library for each load library. Specify LRECL=133,RECFM=FBA,BLKSIZE=(multiple of lrecl up to 32k) or LRECL=137 or greater, RECFM=VBA,BLKSIZE= lrecl+4 to 32k.
4. Run a 2-stage compile. The first stage preprocesses the program, so the IBM Application Delivery Foundation for z Systems family of products have access to fully expanded source code. The second stage compiles the program.
   - In the first compile stage, in both test and production environments:
     - Specify compiler options `PP(COMMENTS,NOLINES)` to expand INCLUDEs and macros. The output is SYSUT10 DD, which is the expanded source file and is the input for the second compiler stage. Note that DB2 programs containing SQL statements cannot use the PP option directly. If used, the behavior is undefined. Follow z/OS Debugger instructions to Processing SQL Statements first. See *Debug Tool for z/OS V13.1 User's Guide*, Chapter 7. "Preparing a DB2 program", section "Processing SQL statements".

     Modify the SYSUT10 DD to enable zDebug, by saving it in an expanded source library and specify a member name that is equal to the primary entry point name or CSECT name of your application program.
   - You can prepare your program with a one stage compile, skipping the expanding source preprocessing step recommended above. If you do this, you need to be aware of the following:
     - Case 1: If there are no executable statements in the header file, the header file is not included in the captured source that is saved in the mdbg file and is not available for browsing during a zDebug session. All other z/OS Debugger functionality is still available.
     - Case 2: If there are executable statements in the header file, the header file is included in the captured source that is saved in the mdbg file and is available for browsing during a z/OS Debugger session.
   - For all programs, such as batch, CICS, and IMS, for the second compiler stage, refer to Table 11 on page 56 for the appropriate options.
5. Modify the SYSCPRT DD in the second compiler stage to refer to a file. This file is the compiler listing and is the source information file for Application Performance Analyzer for z/OS. Save it in the compiler listing library and specify a member that is equal to the CSECT name of your application program.

   ```
   //SYSCPRT DD DSN=compiler.listing.pds(csect-name),DISP=SHR
   ```

   Note: To enable source support in Fault Analyzer, it is a requirement that CSECTs in C programs are named using:

   ```
   #pragma csect(code, "csect_name")
   ```

   where, if using a PDS(E), *csect_name* matches the compiler listing or LANGX file member name.

6. Modify the promotion process to promote compiler listing files. When a load module is promoted, for example, from test to production, promote the corresponding compiler listing file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the compiler listing file that you perform with the module during promotion. You also need to promote any file that is related to the compile, not just the listing. So you need to promote, for example, dbg and mdbg files.
7. Optionally, include a zDebug Language Environment exit module into the load module during the linkage editor step. This approach is one way to enable zDebug's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID.

   Use module EQAD3CXT for batch programs, IMS/TM, IMS BTS programs, and DB2 type MAIN stored procedures.
8. For CICS applications only: if the zDebug DTCN transaction is used to start zDebug, link edit the zDebug CICS startup exit module EQADCCXT into the application load module to enable zDebug in CICS. This link edit is not needed if using the CADP transaction instead of DTCN.

Sample JCL for compiling z/OS C programs with TEST

Here is a JCL example for compiling a z/OS C program for use with the IBM Application Delivery Foundation for z Systems family of products.

```
//* ADD A JOB CARD HERE
//*
//*
//*  SAMPLE JCL TO PREPARE A Z/OS C PROGRAM USING TEST WITH HOOKS
//*  FOR THE IBM ZSERIES PD TOOLS PRODUCTS:
//*     FAULT ANALYZER, DEBUG TOOL, AND APPLICATION PERF. ANALYZER
//*
//*  NOTES:
//*
//*   COMPILER:
//*    1. A 2-STAGE COMPILE IS PERFORMED.  STAGE 1 (PREPROCESS) IS
//*       DONE TO EXPAND INCLUDES AND MACROS IN THE PROGRAM AND TO
//*       PRODUCE AN EXPANDED SOURCE FILE.
//*    2. THE EXPANDED SOURCE FILE IS RETAINED.  IT IS USED BY
//*       DEBUG TOOL.
//*    2. COMPILER PARMS TEST AND NOOPT ARE REQUIRED FOR DEBUG TOOL.
//*    3. COMPILER PARMS AGGREGATE, NOIPA, LIST, NOOFFSET, SOURCE,
//*       AND XREF(FULL) ARE NEEDED TO FORMAT THE COMPILER LISTING
//*       SO THAT IT CAN BE PROCESSED WITH IPVLANGX
//*
//*   A STEP RUNS TO PRODUCE A LANGX FILE FOR FAULT ANALYZER AND APA.
//*    NOTE: USE THE IPVLANGX FACILITY SHIPPED WITH THE COMMON COMPONENT.
//*
//*   BINDER (LINKAGE EDITOR):
//*    1. AN INCLUDE FOR MODULE EQAD?CXT IS OPTIONAL.  IT IS AN
//*       LE EXIT MODULE THAT CAN BE USED TO START DEBUG TOOL.
//*       UNDERSTAND THE METHODS AVAILABLE FOR STARTING DEBUG TOOL,
//*       AND CHOOSE WHETHER YOU WANT TO USE THE LE EXITS.
//*         IF YOU USE THIS METHOD, INCLUDE THE CORRECT EXIT MODULE:
//*            EQADBCXT: FOR BATCH PROGRAMS
//*            EQADICXT: FOR ONLINE IMS PROGRAMS
//*            EQADDCXT: FOR DB2 STORED PROCEDURES (OF TYPE MAIN AND SUB)
//*           (for SUB this is supported only for invocations through call_sub)
//*            (DO NOT INCLUDE AN EXIT FOR CICS PROGRAMS)
//*     YOU CAN ALSO USE MODULE EQAD3CXT FOR BATCH PROGRAMS, ONLINE IMS
//*          PROGRAMS, DB2 TYPE MAIN STORED PROCEDURES.
//*
//*  SET PARMS FOR THIS COMPILE:
//*  ---------------------------
//*       CPRFX: THE PREFIX THE C COMPILE IS INSTALLED UNDER
//*       LEPRFX: THE PREFIX FOR THE LE RUNTIME AND LINK LIBS
//*       DTPRFX: THE PREFIX OF THE DEBUG TOOL SEQAMOD LIBRARY
//*       LANGXLIB: THE PROGRAM OBJECT LIBRARY FOR THE COMMON COMPONENT
//*
//    SET CPRFX=CBC
//    SET LEPRFX=CEE
//    SET DTPRFX=EQAW
//    SET LANGXLIB=IPV.SIPVMODA
//*
//*****************************************************************/
//* CREATE C COMPILER LISTING SYSPRINT, EXPANDED SOURCE DEBUG,      */
//* AND EQALANGX FILES                                              */
//*****************************************************************/
//ALLOC EXEC PGM=IEFBR14
//LISTING DD DSN=&SYSUID..ADLAB.CLST,
//           DISP=(MOD,CATLG),
//           DCB=(DSORG=PO,RECFM=VBA,LRECL=137,BLKSIZE=0),
//           SPACE=(TRK,(20,20,50)),UNIT=SYSDA
//DBGSRC  DD DSN=&SYSUID..ADLAB.CDBG,
//           DISP=(MOD,CATLG),
//           DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=0),
//           SPACE=(TRK,(20,20,50)),UNIT=SYSDA
//LANGX   DD DSN=&SYSUID..ADLAB.EQALANGX,
//           DISP=(MOD,CATLG),
//           DCB=(DSORG=PO,RECFM=VB,LRECL=1562,BLKSIZE=0),
//           SPACE=(TRK,(40,40,50)),UNIT=SYSDA
//*                                                               *
//***************************************************************
//*-----------------------------------------------------------------
//*  COMPILE STEP1: GENERATE EXPANDED C SOURCE FILE IN THE DD
//*                SYSUT10
//*-----------------------------------------------------------------
//COMP1   EXEC PGM=CCNDRVR,REGION=0M,
// PARM=('PP(COMMENTS,NOLINES)')
//STEPLIB  DD  DSNAME=&LEPRFX..SCEERUN2,DISP=SHR
//         DD  DSNAME=&CPRFX..SCCNCMP,DISP=SHR
//SYSMSGS  DD  DUMMY,DSN=&CPRFX..SCBC3MSG(EDCMSGE),DISP=SHR
//SYSLIB   DD  DSNAME=&LEPRFX..SCEEH.H,DISP=SHR
//         DD  DSNAME=&LEPRFX..SCEEH.SYS.H,DISP=SHR
//         DD  DSNAME=&SYSUID..ADLAB.COPYLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSCPRT  DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT5   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT6   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT7   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT8   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT9   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=VB,LRECL=137,BLKSIZE=882)
//SYSUT10  DD  DISP=SHR,DSN=&SYSUID..ADLAB.CDBG(TMC01A)
//SYSUT14  DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT16  DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT17  DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSLIN   DD  DUMMY
//SYSIN    DD  DSNAME=&SYSUID..ADLAB.SOURCE(TMC01A),DISP=SHR
//*
//*-----------------------------------------------------------------
//*  COMPILE STEP2: COMPILE THE EXPANDED SOURCE FILE WITH THE DEBUG
//*                  COMPILER OPTION TEST
//*-------------------------------------------------------------------
//COMP2   EXEC PGM=CCNDRVR,REGION=0M,
//  PARM=('TEST, AGGREGATE, NOIPA, LIST, NESTINC(255),',
//    ' NOOFFSET, NOOPT, SOURCE, XREF, LONGNAME')
//STEPLIB  DD  DSNAME=&LEPRFX..SCEERUN2,DISP=SHR
//         DD  DSNAME=&CPRFX..SCCNCMP,DISP=SHR
//         DD  DSNAME=&LEPRFX..SCEERUN,DISP=SHR
//SYSMSGS  DD  DUMMY,DSN=&CPRFX..SCBC3MSG(EDCMSGE),DISP=SHR
//SYSLIB   DD  DSNAME=&LEPRFX..SCEEH.H,DISP=SHR
//         DD  DSNAME=&LEPRFX..SCEEH.SYS.H,DISP=SHR
//SYSCPRT  DD  DISP=SHR,DSN=&SYSUID..ADLAB.CLST(TMC01A)
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT5   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT6   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT7   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT8   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT9   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=VB,LRECL=137,BLKSIZE=882)
//SYSUT10  DD  SYSOUT=*
//SYSUT14  DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT16  DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT17  DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSLIN   DD DSN=&&TEMOBJ1(TMC01A),DISP=(,PASS),UNIT=SYSDA,
// SPACE=(TRK,(20,20,20)),DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80,DSORG=PO)
//SYSIN    DD  DSNAME=&SYSUID..ADLAB.CDBG(TMC01A),DISP=SHR
//*
//*-------------------------------------------------------------------
//* LINK STEP: LINK THE COMPILED OBJECT DECK
//*-------------------------------------------------------------------
//LKED EXEC PGM=IEWL,PARM=(LET,MAP,LIST)
//SYSLIB DD DSN=&LEPRFX..SCEELKED,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLMOD DD DISP=SHR,DSN=&SYSUID..ADLAB.LOAD
//SYSUT1 DD SPACE=(TRK,(10,10)),UNIT=SYSDA
//OBJECT DD DISP=(OLD,PASS),DSN=&&TEMOBJ1
//* DTLIB DD DSN=&DTPRFX..SEQAMOD,DISP=SHR
//SYSLIN DD *
 INCLUDE OBJECT(TMC01A)
 ENTRY CEESTART
 NAME TMC01(R)
/*
//*  INCLUDING A DEBUG TOOL LE EXIT (EQADBCXT, EQADDCXT, EQADICXT OR EQAD3CXT)
//*  IS OPTIONAL.  THE EXIT ENABLES STARTING DEBUG TOOL WITH THE
//*  USER EXIT DATA SET UTILITY (ONE OF THE DEBUG TOOL ISPF UTILITIES).
//*  AN INCLUDE CAN BE ADDED TO SYSLIN IN THE APPRORIATE SEQUENCE:
//*   INCLUDE DTLIB(EQADBCXT)
//*************************************************************
//* GENERATE THE TMC01A EQALANGX FILE
//*************************************************************
//LANGX1   EXEC PGM=IPVLANGX,REGION=32M,
//  PARM='(C ERROR'
//STEPLIB  DD DISP=SHR,DSN=&LANGXLIB
//         DD DISP=SHR,DSN=&LEPRFX..SCEERUN
//LISTING  DD DSN=&SYSUID..ADLAB.CLST(TMC01A),DISP=SHR
//IDILANGX DD DSN=&SYSUID..ADLAB.EQALANGX(TMC01A),DISP=(OLD)
```

Sample JCL for compiling z/OS C++ programs

Here is a JCL example for compiling a z/OS C++ program for use with the IBM Application Delivery Foundation for z Systems family of products.

```
//* ADD A JOB CARD HERE
//*
//*
//*  SAMPLE JCL TO PREPARE A Z/OS C++ PROGRAM USING DWARF WITHOUT HOOKS
//*  FOR THE IBM ZSERIES PD TOOLS PRODUCTS:
//*     FAULT ANALYZER, DEBUG TOOL, AND APPLICATION PERF. ANALYZER
//*
//*  NOTES:
//*
//*   COMPILER:
//*    1. A 2-STAGE COMPILE IS PERFORMED.  STAGE 1 (PREPROCESS) IS
//*       DONE TO EXPAND INCLUDES AND MACROS IN THE PROGRAM AND TO
//*       PRODUCE AN EXPANDED SOURCE FILE.
//*    2. THE EXPANDED SOURCE FILE IS RETAINED.  IT IS USED BY
//*       THE MDBG CREATE ROUTINE TO CAPTURE THE SOURCE.
//*    2. COMPILER PARMS ARE SPECIFIED TO GENERATE A DWARF FILE WITH
//*       NOHOOKS.  OTHER OPTIONS ARE SPECIFIED TO FULFILL FA, DT AND
//*       APA REQUIREMENTS.
//*
//*   BIND:
//*    1. AN INCLUDE FOR MODULE EQAD?CXT IS OPTIONAL.  IT IS AN
//*       LE EXIT MODULE THAT CAN BE USED TO START DEBUG TOOL.
//*       UNDERSTAND THE METHODS AVAILABLE FOR STARTING DEBUG TOOL,
//*       AND CHOOSE WHETHER YOU WANT TO USE THE LE EXITS.
//*         IF YOU USE THIS METHOD, INCLUDE THE CORRECT EXIT MODULE:
//*            EQADBCXT: FOR BATCH PROGRAMS
//*            EQADICXT: FOR ONLINE IMS PROGRAMS
//*            EQADDCXT: FOR DB2 STORED PROCEDURES (OF TYPE MAIN AND SUB)
//*           (FOR SUB THIS IS SUPPORTED ONLY FOR INVOCATIONS THROUGH CALL_SUB)
//*            (DO NOT INCLUDE AN EXIT FOR CICS PROGRAMS)
//*          YOU CAN ALSO USE MODULE EQAD3CXT FOR BATCH PROGRAMS, ONLINE IMS
//*          PROGRAMS, DB2 TYPE MAIN STORED PROCEDURES.
//*
//*   MDBG:
//*    AN MDBG FILE IS CREATED FOR DEBUG TOOL. IT WILL CONTAIN ALL THE
//*    ROUTINES IN THE PROGRAM OBJECT WITH DBG FILES AND THE CAPTURED
//*    SOURCE. IN ORDER TO USE THIS FILE IN DEBUG TOOL, THE DEBUG TOOL
//*    SESSION NEEDS TO HAVE THE EQAOPTS MDBG COMMAND SET TO YES.
//*
//*  SET PARMS FOR THIS COMPILE:
//*  ---------------------------
//*       CPRFX: THE PREFIX THE C++ COMPILE IS INSTALLED UNDER
//*       LEPRFX: THE PREFIX FOR THE LE RUNTIME AND LINK LIBS
//*       DTPRFX: THE PREFIX OF THE DEBUG TOOL SEQAMOD LIBRARY
//*
//    SET CPRFX=CBC
//    SET LEPRFX=CEE
//    SET DTPRFX=EQAW
//*
//******************************************************************/
//* CREATE C++ COMPILER LISTING SYSPRINT, EXPANDED SOURCE DEBUG,    */
//* DBG AND MDBG files.                                             */
//******************************************************************/
//ALLOC EXEC PGM=IEFBR14
//LISTING DD DSN=&SYSUID..ADLAB.CLST,
//           DISP=(MOD,CATLG),
//           DCB=(DSORG=PO,RECFM=VBA,LRECL=137,BLKSIZE=0),
//           SPACE=(TRK,(20,20,50)),UNIT=SYSDA
//DBGSRC  DD DSN=&SYSUID..ADLAB.CDBG,
//           DISP=(MOD,CATLG),
//           DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=0),
//           SPACE=(TRK,(20,20,50)),UNIT=SYSDA
//DBG     DD DSN=&SYSUID..ADLAB.DBG,
//           DISP=(MOD,CATLG),
//           DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=0),
//           SPACE=(TRK,(40,40,50)),UNIT=SYSDA
//MDBG    DD DSN=&SYSUID..ADLAB.MDBG,
//           DISP=(MOD,CATLG),
//           DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=0),
//           SPACE=(TRK,(40,40,50)),UNIT=SYSDA
//*                                                                *
//****************************************************************
//*-----------------------------------------------------------------
//*  COMPILE STEP1: GENERATE EXPANDED C++ SOURCE FILE IN THE DD
//*                 SYSUT10
//*-----------------------------------------------------------------
//COMP1   EXEC PGM=CCNDRVR,REGION=0M,
// PARM=('PP(COMMENTS,NOLINES)')
//STEPLIB  DD  DSNAME=&LEPRFX..SCEERUN2,DISP=SHR
//         DD  DSNAME=&CPRFX..SCCNCMP,DISP=SHR
//SYSMSGS  DD  DUMMY,DSN=&CPRFX..SCBC3MSG(EDCMSGE),DISP=SHR
//SYSLIB   DD  DSNAME=&LEPRFX..SCEEH.H,DISP=SHR
//         DD  DSNAME=&LEPRFX..SCEEH.SYS.H,DISP=SHR
//         DD  DSNAME=&SYSUID..ADLAB.COPYLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSCPRT  DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT5   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT6   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT7   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT8   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT9   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=VB,LRECL=137,BLKSIZE=882)
//SYSUT10  DD  DISP=SHR,DSN=&SYSUID..ADLAB.CDBG(TMC01A)
//SYSUT14  DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT16  DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT17  DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSLIN   DD  DUMMY
//SYSIN    DD  DSNAME=&SYSUID..ADLAB.SOURCE(TMC01A),DISP=SHR
//*
//*-----------------------------------------------------------------
//*  COMPILE STEP2: COMPILE THE EXPANDED SOURCE FILE WITH THE DEBUG
//*                 COMPILER DEBUG(FORMAT(DWARF, NOHOOK))
//*-----------------------------------------------------------------
//COMP2   EXEC PGM=CCNDRVR,REGION=0M,
//  PARM=('/CXX DEBUG(FORMAT(DWARF), NOHOOK, SYMBOL),',
//    ' LIST, LONGNAME, NOOFFSET')
//STEPLIB  DD  DSNAME=&LEPRFX..SCEERUN2,DISP=SHR
//         DD  DSNAME=&CPRFX..SCCNCMP,DISP=SHR
//         DD  DSNAME=&LEPRFX..SCEERUN,DISP=SHR
//SYSMSGS  DD  DUMMY,DSN=&CPRFX..SCBC3MSG(EDCMSGE),DISP=SHR
//SYSLIB   DD  DSNAME=&LEPRFX..SCEEH.H,DISP=SHR
//         DD  DSNAME=&LEPRFX..SCEEH.SYS.H,DISP=SHR
//SYSCPRT  DD  DISP=SHR,DSN=&SYSUID..ADLAB.CLST(TMC01A)
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT5   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT6   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT7   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT8   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT9   DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//   DCB=(RECFM=VB,LRECL=137,BLKSIZE=882)
//SYSUT10  DD  SYSOUT=*
//SYSUT14  DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT16  DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSUT17  DD  UNIT=SYSDA,SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=3200,BLKSIZE=12800)
//SYSLIN   DD DSN=&&TEMOBJ1(TMC01A),DISP=(,PASS),UNIT=SYSDA,
// SPACE=(TRK,(20,20,20)),DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80,DSORG=PO)
//SYSIN    DD  DSNAME=&SYSUID..ADLAB.CDBG(TMC01A),DISP=SHR
//*
//*-------------------------------------------------------------------
//* BIND STEP: BIND THE COMPILED OBJECT DECK INTO A PDSE
//*-------------------------------------------------------------------
//BIND EXEC PGM=IEWL,PARM=(LET,MAP,LIST)
//SYSLIB DD DSN=&LEPRFX..SCEELKED,DISP=SHR
//       DD DSN=&LEPRFX..SCEECPP,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLMOD DD DISP=SHR,DSN=&SYSUID..ADLAB.LOADPDSE
//SYSUT1 DD SPACE=(TRK,(10,10)),UNIT=SYSDA
//OBJECT DD DISP=(OLD,PASS),DSN=&&TEMOBJ1
//* DTLIB DD DSN=&DTPRFX..SEQAMOD,DISP=SHR
//SYSLIN DD *
 INCLUDE OBJECT(TMC01A)
 ENTRY CEESTART
 NAME TMC01(R)
/*
//*
//*-------------------------------------------------------------------
//* BUILD MDBG STEP
//*-------------------------------------------------------------------
//DBGLD EXEC PGM=CDADBGLD,REGION=1500K,
//      PARM=('ENVAR("LIBPATH=/usr/lib")/VERSION CAPSRC')
//STEPLIB  DD DISP=SHR,DSN=&LEPRFX..SCEERUN2
//         DD DISP=SHR,DSN=&LEPRFX..SCEERUN
//SYSIN    DD DISP=SHR,DSN=&SYSUID..ADLAB.LOADPDSE(TMC01)
//SYSMDBG  DD DISP=SHR,DSN=&SYSUID..ADLAB.MDBG(TMC01)
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
/*
```

Assembler programs

The following table shows various assembler options that can be used to prepare programs for use with the IBM Application Delivery Foundation for z Systems family of products (IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS). The methods suggested in the following table indicate whether the load module produced is suitable for a production environment. Load modules suitable for a production environment have no significant runtime overhead.

Table 12. Examples of assembler options and source information files that are supported by IBM Application Delivery Foundation for z Systems family of products for Assembler.

Assembler options	**Source information file type produced**	**Is the load module production ready?**	**Options supported and suggested for IBM Debug for z Systems**	**Options supported and suggested for Fault Analyzer for z/OS**	**Options supported and suggested for Application Performance Analyzer for z/OS**
ADATA	SYSADATA file	Yes	N/A	Supported	Supported
ADATA	LANGX file	Yes	Suggested for production and test		

Preparing Assembler programs

Perform the following steps for assembling your programs:

1. Allocate libraries (PDSE is suggested unless PDS is required for your organization) for LANGX files. Allocate one or more LANGX libraries for each environment, such as test and production.
2. Create a corresponding LANGX library for each load library. Specify LRECL=1562 or greater,RECFM=VB,BLKSIZE= lrecl+4 to 32k.
3. For all programs, such as batch, CICS, and IMS, in both test and production environments, specify `ADATA`.

   `ADATA` instructs the assembler to produce a SYSADATA file, which contains source and symbolic data about the program. This produces a production-ready module that can be debugged using IBM Debug for z Systems. `ADATA` does not affect the contents of the assembled module.
4. Add a SYSADATA DD in the assembler step. This file is created by the assembler and it can be a permanent or temporary file. Specify `LRECL=8188 or greater,RECFM=VB,BLKSIZE= lrecl+4 to 32k`. This file is the input to the IPVLANGX utility.
5. Add a step after the assembler step to run the IPVLANGX utility. The IPVLANGX utility reads the SYSADATA file and creates a LANGX file. The LANGX file is the source information file for IBM Debug for z Systems, Fault Analyzer for z/OS and Application Performance Analyzer for z/OS.
6. Save the LANGX file in the LANGX file library, and specify a member name that is equal to the CSECT name.
7. Modify the promotion process to promote LANGX files. When a load module is promoted, for example, from test to production, promote the corresponding LANGX file or files. A promotion can be a recompile, copy, or move. Perform the same steps with the LANGX file that you perform with the module during promotion.
8. If the assembler program is Language Environment-enabled, optionally include a zDebug Language Environment exit module into the load module during the linkage editor step. This approach is one way to enable zDebug's panel 6 in ISPF, a simple panel-driven method to start the debugger automatically when a program runs, without JCL changes, based on the program name and user ID. Use module EQADBCXT for batch programs (including IMS batch), EQADICXT for IMS/TM programs and EQADDCXT for DB2 stored procedures. Do not include the exit module for CICS programs.

   You can also use module EQAD3CXT for batch programs, IMS/TM, IMS BTS programs, and DB2 type MAIN stored procedures

9. For CICS programs only: If the program is a CICS main program, is enabled for Language Environment, and the zDebug DTCN transaction is used to start zDebug, then supplied module EQADCCXT must be included in the load module during the linkage editor step.

Sample JCL for assembling a program

Here is a JCL example for assembling a program for use with the IBM Application Delivery Foundation for z Systems family of products.

```
//*     - - -  ADD A JOB CARD ABOVE THIS LINE  - - -
//*
//*  SAMPLE JCL TO PREPARE AN ASSEMBLER PROGRAM
//*  FOR THE IBM ZSERIES PD TOOLS PRODUCTS:
//*     FAULT ANALYZER, DEBUG TOOL, AND APPLICATION PERF. ANALYZER
//*
//*  NOTES:
//*
//*   ASSEMBLER:
//*    1. AN ADATA PARM IS REQUIRED TO PRODUCE A SYSADATA FILE
//*
//*   A STEP THAT PROCESSES THE SYSADATA FILE,
//*   AND CREATES A LANGX FILE IS NEEDED.
//*
//*   BINDER (LINKAGE EDITOR):
//*    1. AMODE / RMODE CAN BE CODED AS NEEDED BY THE PROGRAM.  THEY ARE
//*       NOT REQUIRED FOR PD TOOLS.
//*
//*  SET PARMS FOR THIS COMPILE:
//*  ---------------------------
//   SET MEM=ASAM1                        PROGRAM NAME
//   SET Language EnvironmentHLQ='CEE'    Language Environment HIGH LVL QUALIFIER
//   SET UNITDEV=SYSALLDA                 UNIT FOR TEMP FILES
//   SET LANGX='IPVLANGX'                 IPVLANGX UTILITY PROGRAM
//   SET LANGXLIB='IPV.SIPVMODA'          LIBRARY FOR IPVLANGX UTILITY
//*    NOTE: USE THE IPVLANGX FACILITY SHIPPED WITH THE COMMON COMPONENT.
//*
//*  *********************************
//*      ASSEMBLER STEP
//*  *********************************
//ASM1 EXEC  PGM=ASMA90,COND=(4,LT),REGION=32M,
//      PARM='ADATA,OBJECT'
//SYSIN    DD DISP=SHR,DSN=&SYSUID..ADLAB.SOURCE(&MEM)
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DISP=SHR,DSN=&SYSUID..ADLAB.OBJ(&MEM)
//SYSADATA DD DISP=SHR,DSN=&SYSUID..ADLAB.SYSADATA(&MEM)
//SYSLIB  DD  DSN=SYS1.MODGEN,DISP=SHR
//        DD  DSN=SYS1.MACLIB,DISP=SHR
//        DD  DSN=&LEHLQ..SCEEMAC,DISP=SHR
//SYSUT1  DD  DISP=(NEW,DELETE),DSN=&&SYSUT1,SPACE=(1700,(900,450)),
//     UNIT=&UNITDEV
//SYSUT2  DD  DISP=(NEW,DELETE),DSN=&&SYSUT2,SPACE=(1700,(600,300)),
//     UNIT=&UNITDEV
//SYSUT3  DD  DISP=(NEW,DELETE),DSN=&&SYSUT3,SPACE=(1700,(600,300)),
//     UNIT=&UNITDEV
//*
//*  *********************************
//*     STEP TO GENERATE LANGX FILE
//*  *********************************
//LANGX    EXEC PGM=&LANGX,REGION=32M,
//  PARM='(ASM ERROR'
//STEPLIB  DD DISP=SHR,DSN=&LANGXLIB
//         DD DISP=SHR,DSN=&LEHLQ..SCEERUN
//SYSADATA DD DSN=&SYSUID..ADLAB.SYSADATA(&MEM),DISP=SHR
//IDILANGX DD DSN=&SYSUID..ADLAB.EQALANGX(&MEM),DISP=SHR
//*
//*  *********************************
//*        LINK-EDIT (BINDER) STEP
//*  *********************************
//LINK     EXEC PGM=IEWL,PARM='MAP',REGION=0M
//SYSLIB   DD DSN=&LEHLQ..SCEELKED,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLMOD  DD DISP=SHR,DSN=&SYSUID..ADLAB.LOAD(&MEM)
//SYSUT1   DD UNIT=SYSDA,SPACE=(TRK,(10,10))
//SYSLIN   DD DSN=&SYSUID..ADLAB.OBJ(&MEM),DISP=SHR
//         DD *
     MODE AMODE(31),RMODE(24)
     ENTRY ASAM1
//*
```

Chapter 6. IPVLANGX compiler listing to side file conversion utility

IPVLANGX is a utility program that converts a compiler listing, or SYSADATA file, to a LANGX side file.

Creating side files using IPVLANGX

To create a side file from a compiler listing, a program named IPVLANGX is used.

The sample JCL in Figure 1 on page 70:

- Compiles a COBOL program.

  Note: You can only compile one program per compile step in order to name the compiler listing PDS(E) member (if using a partitioned data set), and to ensure that only one compiler listing is written to the output file.
- Executes IPVLANGX to process the listing and store it as a side file where the ADFz family of products can access it.

  For return codes issued by IPVLANGX, see "IPVLANGX return codes" on page 100.
- Writes the listing as part of the job output.

The sample file is provided as member IPVSCMPS in the IPV.SIPVSAM1 data set.

```
//IPVSCMPS JOB (GSF),'GENERATE.SIDE.FILE',NOTIFY=&SYSUID.,
//         MSGCLASS=X,CLASS=A,MSGLEVEL=(1,1)
//         JCLLIB ORDER=(IGY.V2R1M0.SIGYPROC) <== INSTALLATION
//*                                             IGYWCLG PROC
//*
//*********************************************************/
//* THIS JOB RUNS A COBOL COMPILE PLUS PRODUCES A SIDE FILE  */
//* FROM A PROGRAM LISTING THAT THE PD TOOLS PRODUCTS CAN    */
//* USE FOR OBTAINING SOURCE INFORMATION.                    */
//* THE COMPILE OUTPUT IS THEN WRITTEN TO SYSUT2 IN THE      */
//* IEBGENER STEP.                                           */
//*********************************************************/
//*
//CBLRUN   EXEC IGYWC,PARM.COBOL='LIST,MAP,Source,XREF'
//COBOL.SYSIN DD DATA,DLM='##'
⋮
(Program source not shown)
⋮
##
//COBOL.SYSPRINT DD DSN=&&COBLIST(IPVSCBL1),
//         DISP=(,PASS),SPACE=(TRK,(10,5,5),RLSE),
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=0)
//*
//IPVLANGX EXEC PGM=IPVLANGX,REGION=4096K,
// PARM='IDISCBL1 (COBOL ERROR'
//LISTING  DD  DISP=(OLD,PASS),DSN=&&COBLIST  1
//IDILANGX DD  DISP=SHR,DSN=IPV.IPVLANGX      2
//SYSUDUMP DD  SYSOUT=*
//*
//IEBGENER EXEC PGM=IEBGENER,REGION=4096K
//SYSUT1   DD  DISP=OLD,DSN=&&COBLIST(IPVSCBL1)
//SYSUT2   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
/*
```

Figure 1. Sample JCL to compile a COBOL program and store the side file

| After you have created and stored a side file, there is no benefit to the ADFz
| family of products in retaining the listing.

If you already have listings, you can turn them into side files. Here is sample JCL to do this conversion (it is provided as member IPVSFILE in the IPV.SIPVSAM1 data set):

```
//IPVLANGX JOB (C97),'IPVLANGX',MSGCLASS=X,
//         CLASS=A,NOTIFY=&SYSUID
//************************************************************
//* This job produces a side file from a program listing that
//* the PD Tools products can use for obtaining source information.
//* This particular example is set up for a COBOL extraction
//* from IPV.LISTING.COBOL(COBOLA) to IPV.IPVLANGX
//************************************************************
//IPVLANGX EXEC PGM=IPVLANGX,REGION=4096K,
// PARM='COBOLA (COBOL ERROR'
//LISTING  DD  DISP=SHR,DSN=IDI.LISTING.COBOL  1
//IDILANGX DD  DISP=SHR,DSN=IDI.IPVLANGX       2
//SYSUDUMP DD  SYSOUT=*
```

Figure 2. Sample JCL to create a side file from a COBOL listing

Notes®:

1 DDname must be LISTING for all types of compiler listings, or SYSADATA for an assembler ADATA file.

2 DDname must be IDILANGX for the output LANGX side file. The data set must be sequential or PDS(E), RECFM=VB, LRECL≥1562.

Refer to the documentation for the individual ADFz family of products for information about how to provide the LANGX side file for processing.

A compiler listing is the only data format that IPVLANGX accepts as input, with the exception of SYSADATA for assembler.

IPVLANGX parameters

The PARM string passed to IPVLANGX should contain:

Syntax

```
►►──PARM='─┬──────────┬─(language─┬────────────────────────────────────┬──────►◄
           └─mbr_name─┘            │ ┌────────────────────────────────┐ │
                                   └─▼─┬─ERROR──────────────────────┬─┴─┘
                                       │              ┌─,───┐      │
                                       │              ▼ (1) │      │
                                       └─PermitLangx(───────msgid─┴─)─┘
```

Notes:

1 Either comma or blank character is permitted as delimiter.

where:

mbr_name
: The compiler listing or ADATA file member name in the input data set identified by the LISTING DD name (for a compiler listing) or the SYSADATA DD name (if an ADATA file). This parameter is optional. If omitted, the JCL must specify for the compiler listing or ADATA file, either a sequential data set, or a PDS(E) data set with member name. Also, the output IPVLANGX member is named according to the input program name. In the case of COBOL, for example, this name is the name found on the PROGRAM-ID source line.

language
The language of the compiler listing or ADATA file, as:
- COBOL
- PLI
- C
- ASM

ERROR
An optional parameter that provides more diagnostics on variables for which information is incomplete.

PermitLangx(*msgid***, ...)**
An optional parameter that specifies message IDs for compiler error messages which should be ignored. For details, see *IBM Fault Analyzer for z/OS User's Guide and Reference*. Chapter 33 "Options", section "PermitLangx".

Including an IPVLANGX step in your SCLM translator

If you use the ISPF/PDF Software Configuration and Library Manager (SCLM) to manage your application software, then you might want to include an IPVLANGX step in your SCLM translator, since LANGX side files generally take up less disk space than compiler listings. Shown in the following, and included in the PD Tools Common Component softcopy samples data set, are examples of an IPVLANGX step inserted into a High Level Assembler and a COBOL SCLM translator.

High Level Assembler SCLM example

This example is included in data set IPV.SIPVSAM1 as member IPVSCLMA.

```
*          SYSADATA DDNAME used in HLASM step.
*          (* SYSADATA *)
           FLMALLOC  IOTYPE=W,DDNAME=SYSADATA,RECFM=VB,RECNUM=9000,    C
                LRECL=8188,BLKSIZE=8192,PRINT=Y
*
*
* IPVLANGX BUILD TRANSLATOR
*
           FLMTRNSL  CALLNAM='IPVLANGX',                               C
                FUNCTN=BUILD,                                          C
                COMPILE=IPVLANGX,                                      C
                DSNAME=IPV.SIPVMODA,                                   C
                VERSION=3.5.2,                                         C
                GOODRC=0,                                              C
                PORDER=1,                                              C
                OPTIONS='@@FLMMBR(ASM ERROR'
*
*          (* SYSADATA *)
           FLMALLOC  IOTYPE=U,DDNAME=SYSADATA
*
*          (* IDILANGX *)
           FLMALLOC  IOTYPE=P,DDNAME=IDILANGX,DFLTTYP=IDILANGX,        C
                KEYREF=OUT2,BLKSIZE=27998,LRECL=1562,RECFM=VB,         C
                RECNUM=10000,DIRBLKS=50,DFLTMEM=*
```

COBOL SCLM example

This example is included in data set IPV.SIPVSAM1 as member IPVSCLMC.

```
************************************************************************
*        --COPY SYSPRINT FILE TO LISTING
* The COPYFILE EXEC, in dataset PDFTDEV.PROJDEFS.EXEC contains the
* following:
*
```

```
* /* REXX */
* /*******************************************************************/
* /* Copy file I to file O.  Both are assumed to be pre-allocated.     */
* /*******************************************************************/
* PARSE UPPER ARG I","O .
* "EXECIO * DISKR "I" (STEM R. FINIS "
* "EXECIO * DISKW "O" (STEM R. FINIS "
* RETURN
*
*********************************************************************
*
         FLMTRNSL   CALLNAM='COPY FILES      ',                      C
               FUNCTN=BUILD,                                         C
               COMPILE=COPYFILE,                                     C
               DSNAME=PDFTDEV.PROJDEFS.EXEC,                         C
               CALLMETH=TSOLNK,                                      C
               VERSION=1.0,                                          C
               PORDER=1,                                             C
               OPTIONS=(SYSPRINT,LISTING),                           C
               GOODRC=0

           FLMALLOC  IOTYPE=W,RECFM=VBA,LRECL=133,                   C
               RECNUM=90000,DDNAME=LISTING
*
           FLMTRNSL  CALLNAM='IPVLANGX',                             C
               FUNCTN=BUILD,                                         C
               COMPILE=IPVLANGX,                                     C
               DSNAME=IPV.SIPVMODA,                                  C
               VERSION=3.5.2,                                        C
               GOODRC=0,                                             C
               PORDER=1,                                             C
               OPTIONS='@@FLMMBR(COBOL ERROR'
*
*          (* LISTING *)
           FLMALLOC  IOTYPE=U,DDNAME=LISTING
*
*          (* IDILANGX *)
           FLMALLOC  IOTYPE=P,DDNAME=IDILANGX,DFLTTYP=IDILANGX,      C
               KEYREF=OUT2,BLKSIZE=27998,LRECL=1562,RECFM=VB,        C
               RECNUM=10000,DIRBLKS=50,DFLTMEM=*
```

Chapter 7. IPVLANGP side file formatting utility

A utility program, IPVLANGP, is provided, which can be used to create a readable listing from one of the following:

- A LANGX side file.
- For Enterprise PL/I, a SYSDEBUG side file that is generated by using the PL/I TEST(SEPARATE) compiler option.
- For Enterprise COBOL prior to Version 5, a SYSDEBUG side file that is generated by using the COBOL TEST(SEPARATE) option.
- For Enterprise COBOL Version 5, a program object containing DWARF debugging information generated by using the TEST(SOURCE) option.

This approach might be useful if side files, rather than compiler listings, are kept in order to conserve DASD space. The utility program is able to format the side file or program object DWARF debugging information in a way that resembles the original compiler listing.

IPVLANGP can be executed in a number of different ways:

- As an ISPF option 3.4 (Data Set List Utility) line command against a sequential LANGX side file or COBOL or PL/I SYSDEBUG side file data set, or if the data set is partitioned, as a line command against a member of the data set. For Enterprise COBOL Version 5, IPVLANGP can also be issued against a program object member of a PDSE load library.

  If a sequential COBOL or PL/I SYSDEBUG side file data set is used, or if the member of a partitioned COBOL or PL/I SYSDEBUG side file data set does not match any PROGRAM-ID named program contained within it, then a prompt is displayed which permits a program name to be specified.

  When a COBOL Version 5 program object contains more than one compile unit, a prompt is displayed to select the desired program.

  All LANGX side files that are contained in a sequential data set, or a partitioned data set member, are displayed, regardless of whether these match the member name or not.

  The output is written to a temporary data set and displayed using ISPF EDIT.
- From a Fault Analyzer ISPF interface display, using the Services action-bar pull-down menu, and selecting the "IPVLANGP Side File Formatting Utility" option. A prompt is displayed, from which you specify the data set to be formatted.

  The output is presented in an ISPF display, but may be copied to a data set using the COPY command.
- As a batch job, like the following:

  ```
  //PRTLANGX JOB ...
  //STEP1    EXEC PGM=IPVLANGP,PARM='parms'
  //SYSPRINT DD   SYSOUT=*
  ```

  The PARM string passed to IPVLANGP should contain:

Syntax

```
►►──PARM='data_set_name──┬──────────────────┬──────────────────────────►◄
                         └─PROG:program_name─┘
```

where:

data_set_name
: The name of a sequential LANGX side file or COBOL or PL/I SYSDEBUG side file data set, or if the data set is partitioned (as is the case for COBOL Version 5 program objects), the data set name with member name included in parentheses.

  Examples:

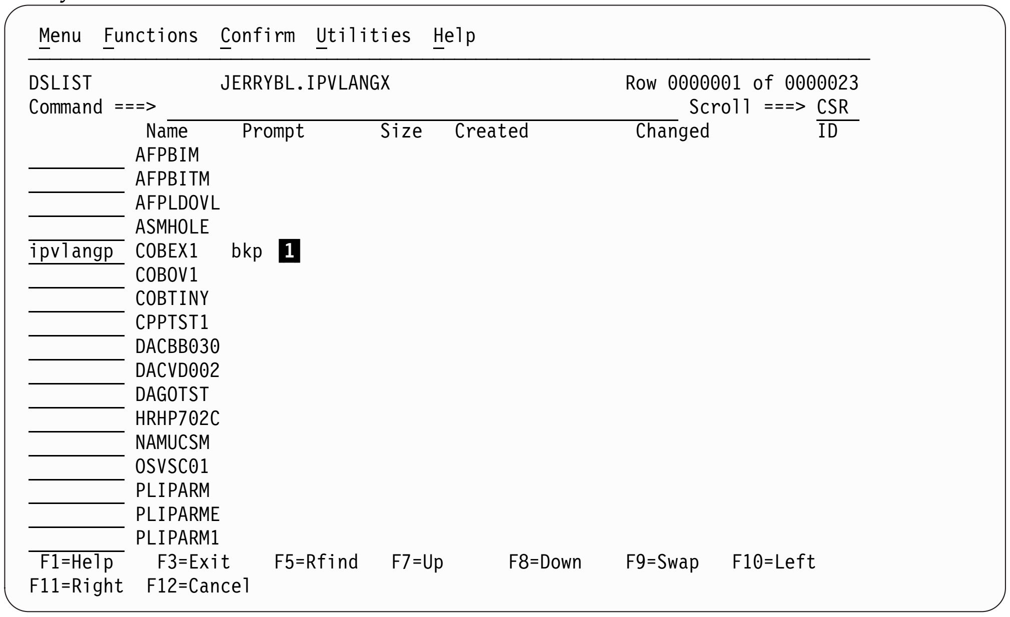
  ```
  MY.SYSDEBUG.SEQ.DS
  MY.IPVLANGX.PDS.DS(MYPROG)
  ```

program_name
: The name of a PROGRAM-ID named program contained within a COBOL or PL/I SYSDEBUG side file or COBOL Version 5 program object. This name must be specified if the COBOL or PL/I SYSDEBUG side file data set is sequential, or if the member name of a partitioned data set does not match the name of any program contained within it.

The formatted listing is written to the SYSPRINT DD. Normal listing file attributes, such as variable-blocked record format and logical record length of 137, are generally appropriate.

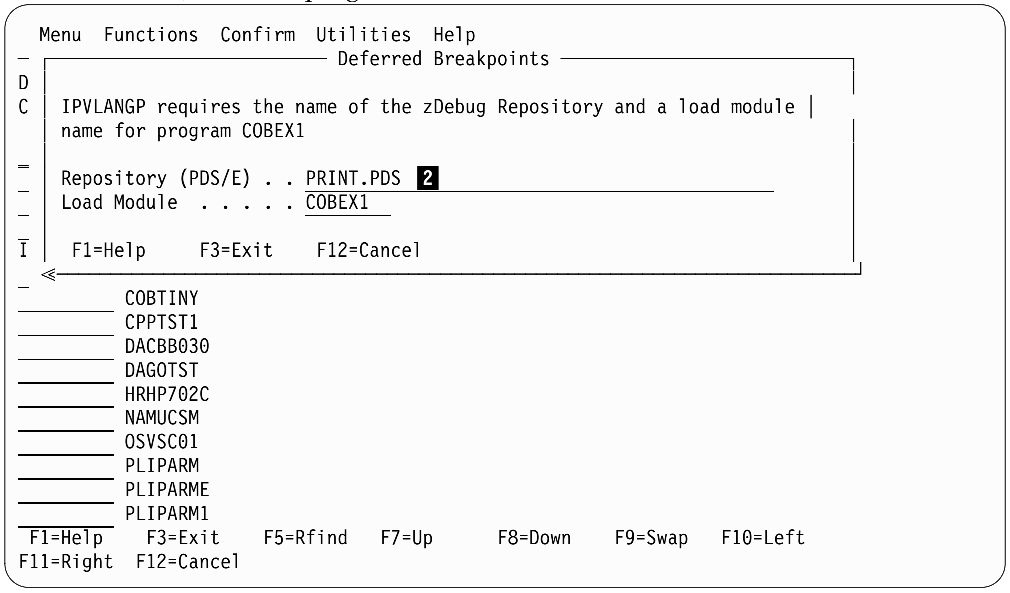
Deferred Breakpoints Feature

Start IPVLANGP with the "bkp" parameter for a COBOL or PL/I SYSDEBUG file, a COBOL LANGX file, or a COBOL Version 5 program object containing DWARF debugging information. You can also start it from zDebug Utilities or Fault Analyzer Services menus. Here it is started from an ISPF member list:


```
 Menu  Functions  Confirm  Utilities  Help
──────────────────────────────────────────────────────────────────────────────
DSLIST           JERRYBL.IPVLANGX                        Row 0000001 of 0000023
Command ===> ____________________________________________ Scroll ===> CSR
           Name     Prompt       Size   Created          Changed          ID
__________ AFPBIM
__________ AFPBITM
__________ AFPLDOVL
__________ ASMHOLE
ipvlangp   COBEX1   bkp  1
__________ COBOV1
__________ COBTINY
__________ CPPTST1
__________ DACBB030
__________ DACVD002
__________ DAGOTST
__________ HRHP702C
__________ NAMUCSM
__________ OSVSC01
__________ PLIPARM
__________ PLIPARME
__________ PLIPARM1
 F1=Help    F3=Exit    F5=Rfind   F7=Up      F8=Down    F9=Swap   F10=Left
F11=Right  F12=Cancel
```

1 The "bkp" parameter enables setting of deferred breakpoints.

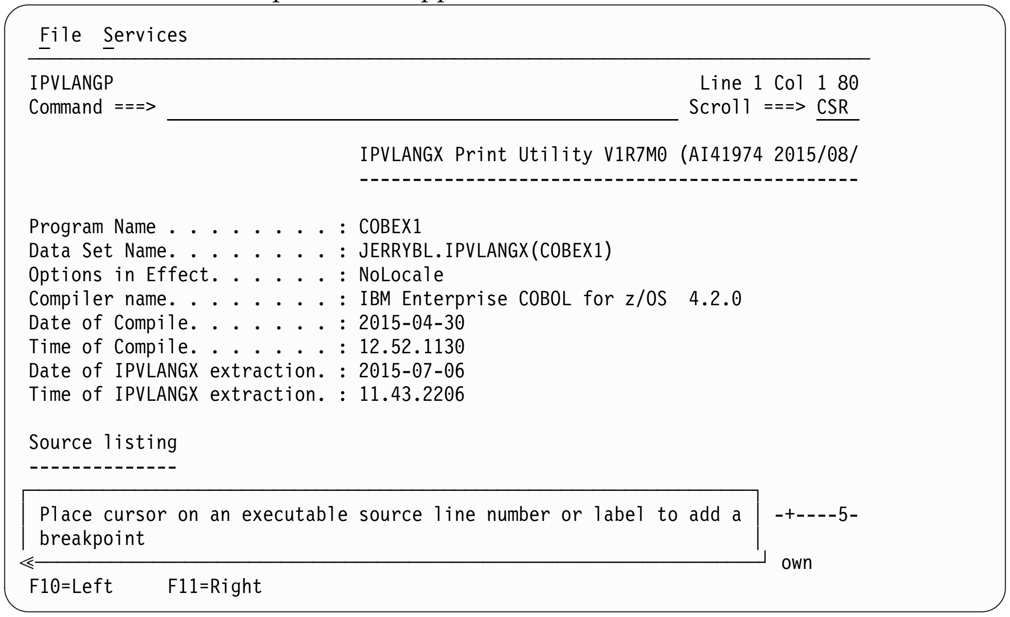
IPVLANGP then prompts for the zDebug Repository and the program's load module name (default is program name):



```
  Menu  Functions  Confirm  Utilities  Help
_ ┌──────────────────────── Deferred Breakpoints ──────────────────────────┐
D │                                                                        │
C │ IPVLANGP requires the name of the zDebug Repository and a load module  │
  │ name for program COBEX1                                                │
_ │                                                                        │
_ │ Repository (PDS/E) . . PRINT.PDS  2                                    │
_ │ Load Module  . . . . . COBEX1                                          │
_ │                                                                        │
I │  F1=Help     F3=Exit    F12=Cancel                                     │
_ └────────────────────────────────────────────────────────────────────────┘
_________ COBTINY
_________ CPPTST1
_________ DACBB030
_________ DAGOTST
_________ HRHP702C
_________ NAMUCSM
_________ OSVSC01
_________ PLIPARM
_________ PLIPARME
_________ PLIPARM1
 F1=Help    F3=Exit    F5=Rfind   F7=Up      F8=Down    F9=Swap   F10=Left
F11=Right  F12=Cancel
```

2 The Repository data set name is saved in an ISPF variable and is automatically initialized to the last used data set name on subsequent invocations.

The main IPVLANGP panel now appears:



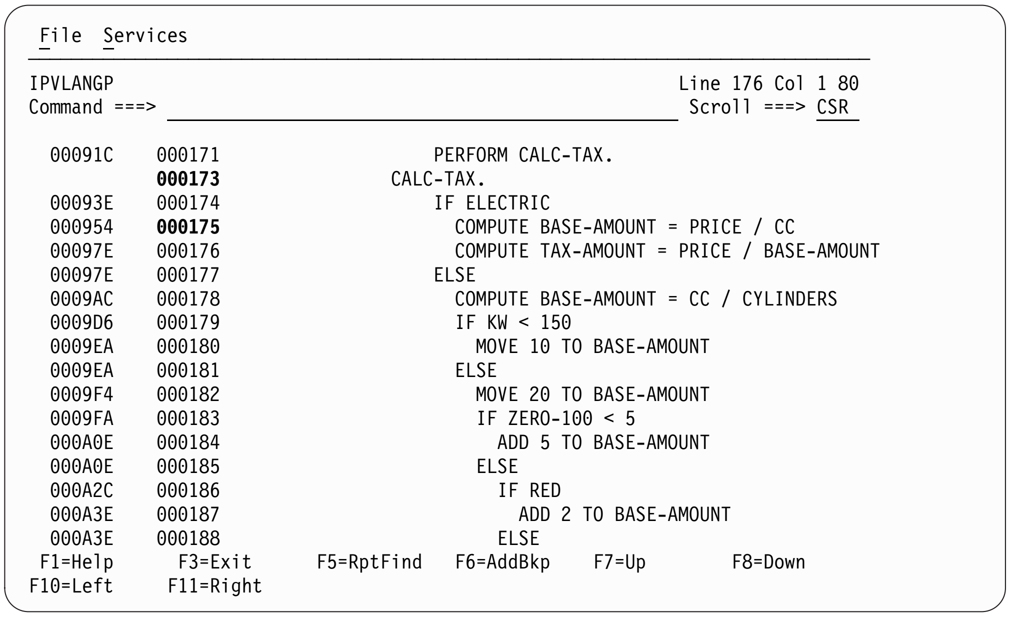
```
 File  Services
───────────────────────────────────────────────────────────────────────────
IPVLANGP                                                    Line 1 Col 1 80
Command ===> ______________________________________________ Scroll ===> CSR

                              IPVLANGX Print Utility V1R7M0 (AI41974 2015/08/
                              ---------------------------------------------

Program Name . . . . . . . . : COBEX1
Data Set Name. . . . . . . . : JERRYBL.IPVLANGX(COBEX1)
Options in Effect. . . . . . : NoLocale
Compiler name. . . . . . . . : IBM Enterprise COBOL for z/OS  4.2.0
Date of Compile. . . . . . . : 2015-04-30
Time of Compile. . . . . . . : 12.52.1130
Date of IPVLANGX extraction. : 2015-07-06
Time of IPVLANGX extraction. : 11.43.2206

Source listing
--------------
┌─────────────────────────────────────────────────────────────────┐
│ Place cursor on an executable source line number or label to add a │  -+----5-
│ breakpoint                                                      │
└─────────────────────────────────────────────────────────────────┘  own
F10=Left     F11=Right
```

Any existing breakpoints for the program are retrieved from the repository. Line (Stmt) numbers where breakpoints are set will be highlighted, for example lines 173 and 175 in the following example:



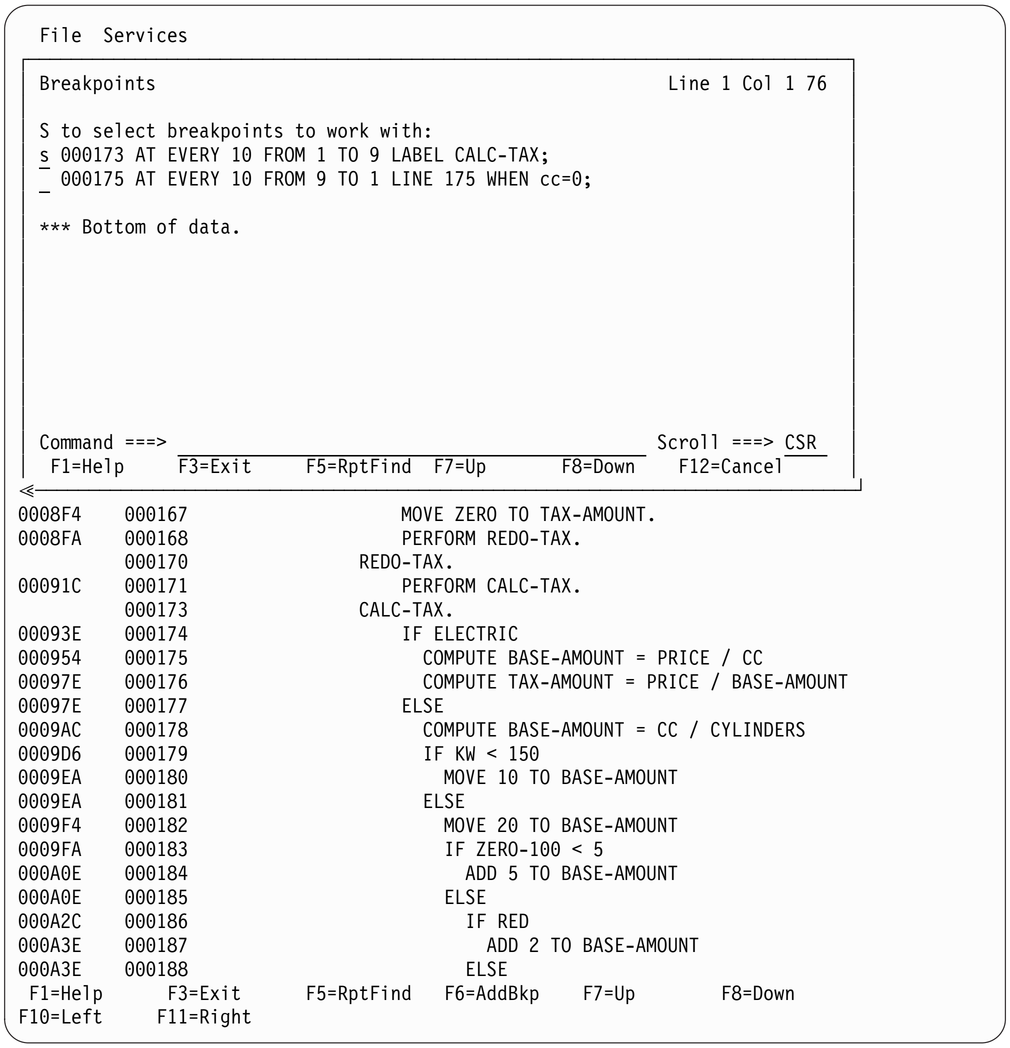
```
 File  Services
________________________________________________________________________________
IPVLANGP                                                      Line 176 Col 1 80
Command ===> ________________________________________________ Scroll ===> CSR

  00091C    000171                    PERFORM CALC-TAX.
            000173                CALC-TAX.
  00093E    000174                    IF ELECTRIC
  000954    000175                      COMPUTE BASE-AMOUNT = PRICE / CC
  00097E    000176                      COMPUTE TAX-AMOUNT = PRICE / BASE-AMOUNT
  00097E    000177                    ELSE
  0009AC    000178                      COMPUTE BASE-AMOUNT = CC / CYLINDERS
  0009D6    000179                      IF KW < 150
  0009EA    000180                        MOVE 10 TO BASE-AMOUNT
  0009EA    000181                      ELSE
  0009F4    000182                        MOVE 20 TO BASE-AMOUNT
  0009FA    000183                        IF ZERO-100 < 5
  000A0E    000184                          ADD 5 TO BASE-AMOUNT
  000A0E    000185                        ELSE
  000A2C    000186                          IF RED
  000A3E    000187                            ADD 2 TO BASE-AMOUNT
  000A3E    000188                          ELSE
 F1=Help      F3=Exit      F5=RptFind   F6=AddBkp    F7=Up        F8=Down
F10=Left     F11=Right
```

Breakpoints can be added or viewed by pressing PF6, which is sensitive to the cursor position:

- With the cursor on a Line (Stmt) number, a popup appears allowing a new line or label breakpoint to be added, or an existing breakpoint to be modified (or cleared).
- If the cursor is outside of the source area (for example, on the command line), a list of existing breakpoints is shown allowing one or more to be worked with.

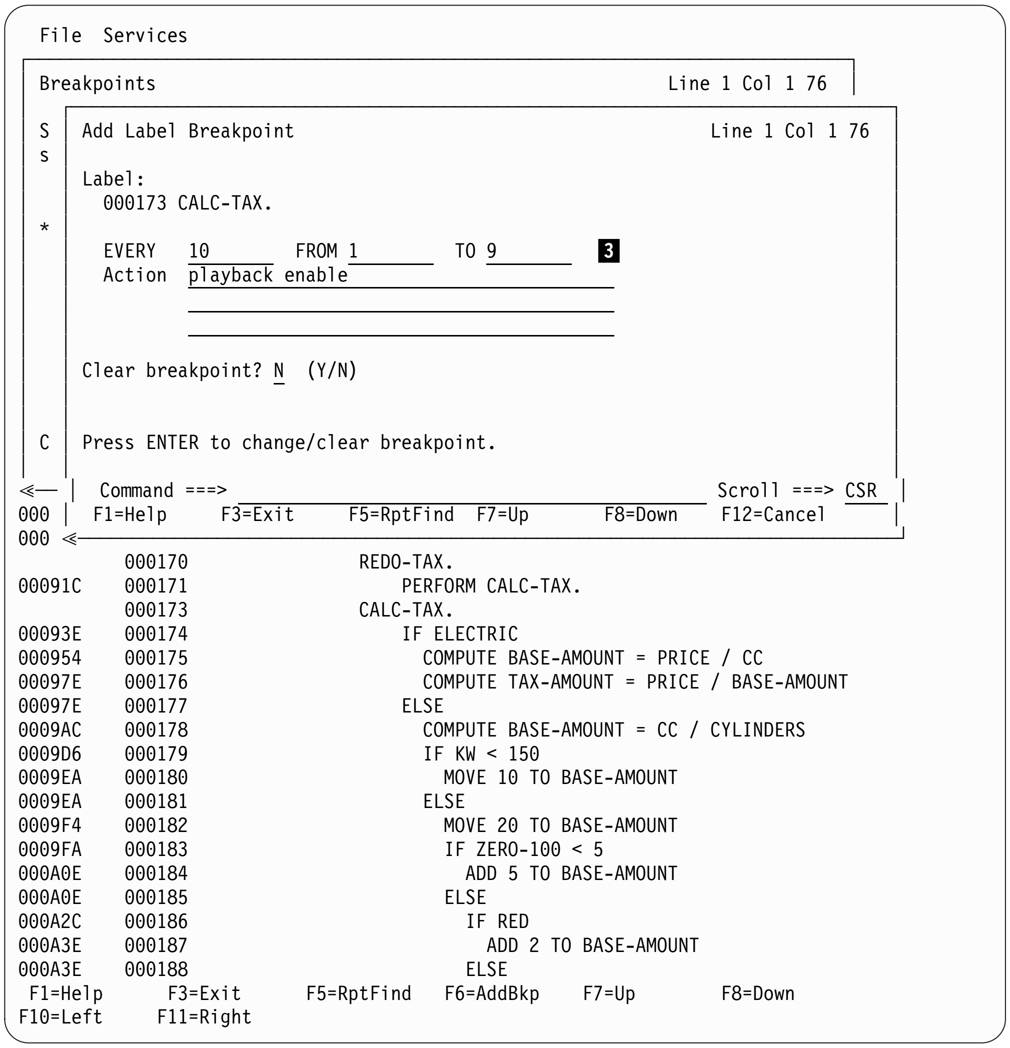
In the following example, PF6 is pressed with the cursor on the command line:



```
  File  Services
 ┌─────────────────────────────────────────────────────────────────────────────┐
 │ Breakpoints                                                Line 1 Col 1 76  │
 │                                                                             │
 │ S to select breakpoints to work with:                                       │
 │ s 000173 AT EVERY 10 FROM 1 TO 9 LABEL CALC-TAX;                            │
 │ _ 000175 AT EVERY 10 FROM 9 TO 1 LINE 175 WHEN cc=0;                        │
 │                                                                             │
 │ *** Bottom of data.                                                         │
 │                                                                             │
 │                                                                             │
 │                                                                             │
 │                                                                             │
 │                                                                             │
 │                                                                             │
 │                                                                             │
 │                                                                             │
 │ Command ===> ____________________________________________ Scroll ===> CSR   │
 │  F1=Help     F3=Exit     F5=RptFind  F7=Up       F8=Down    F12=Cancel      │
 └─────────────────────────────────────────────────────────────────────────────┘
0008F4    000167                 MOVE ZERO TO TAX-AMOUNT.
0008FA    000168                 PERFORM REDO-TAX.
          000170             REDO-TAX.
00091C    000171                 PERFORM CALC-TAX.
          000173             CALC-TAX.
00093E    000174                 IF ELECTRIC
000954    000175                   COMPUTE BASE-AMOUNT = PRICE / CC
00097E    000176                   COMPUTE TAX-AMOUNT = PRICE / BASE-AMOUNT
00097E    000177                 ELSE
0009AC    000178                   COMPUTE BASE-AMOUNT = CC / CYLINDERS
0009D6    000179                   IF KW < 150
0009EA    000180                     MOVE 10 TO BASE-AMOUNT
0009EA    000181                   ELSE
0009F4    000182                     MOVE 20 TO BASE-AMOUNT
0009FA    000183                     IF ZERO-100 < 5
000A0E    000184                       ADD 5 TO BASE-AMOUNT
000A0E    000185                     ELSE
000A2C    000186                       IF RED
000A3E    000187                         ADD 2 TO BASE-AMOUNT
000A3E    000188                       ELSE
 F1=Help      F3=Exit      F5=RptFind   F6=AddBkp    F7=Up        F8=Down
F10=Left     F11=Right
```

As shown, breakpoints are listed in zDebug command format.

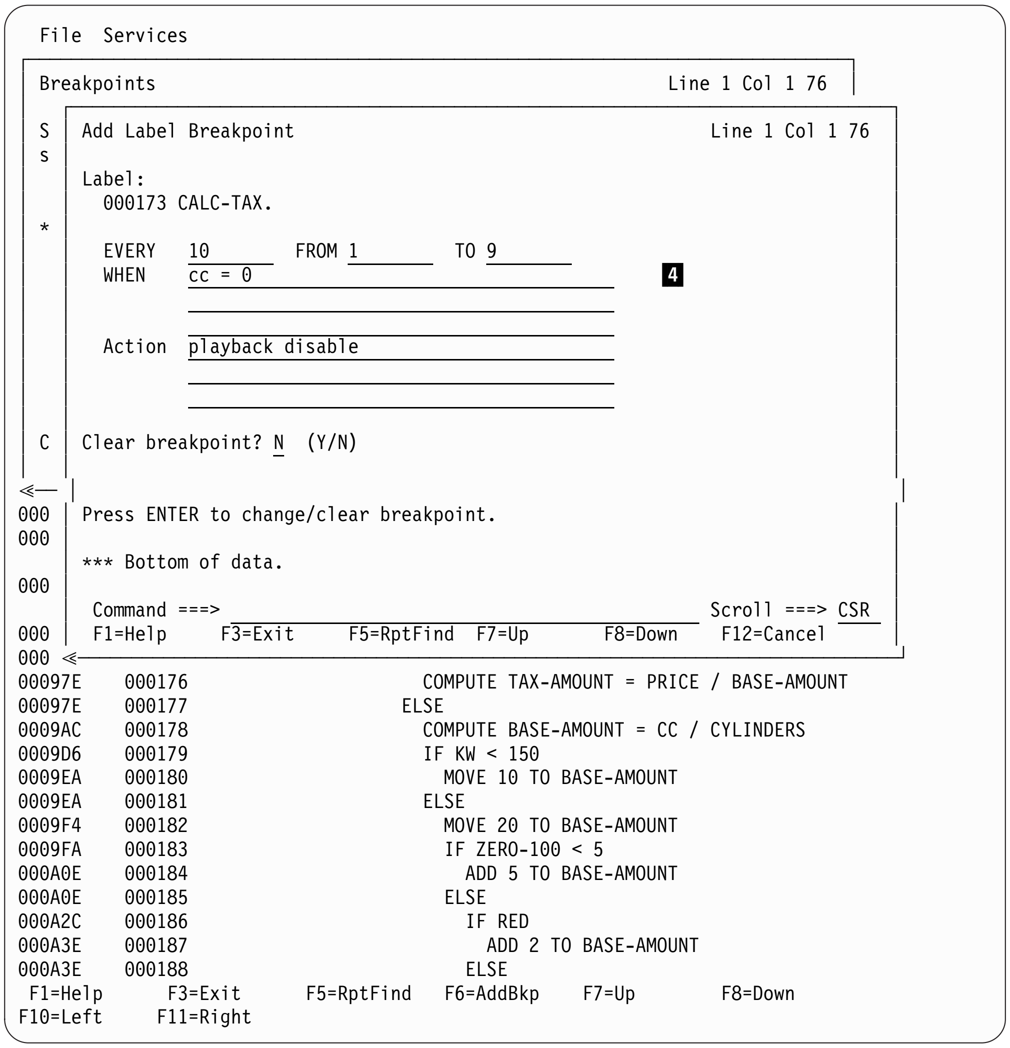
Enter the 'S' line command to work with one or more existing line or label breakpoints. Here, the line 173 breakpoint is selected:



```
 File  Services
┌──────────────────────────────────────────────────────────────────────┐
│Breakpoints                                            Line 1 Col 1 76│
│ ┌──────────────────────────────────────────────────────────────────────┐
│S│ Add Label Breakpoint                                   Line 1 Col 1 76│
│s│                                                                       │
│ │ Label:                                                                │
│ │   000173 CALC-TAX.                                                    │
│*│                                                                       │
│ │   EVERY   10______  FROM 1_______  TO 9_______  3                     │
│ │   Action  playback enable_____________________                        │
│ │           ____________________________________                        │
│ │           ____________________________________                        │
│ │                                                                       │
│ │ Clear breakpoint? N  (Y/N)                                            │
│ │                                                                       │
│ │                                                                       │
│C│ Press ENTER to change/clear breakpoint.                               │
│ │                                                                       │
<── │  Command ===> ____________________________________  Scroll ===> CSR │
000 │  F1=Help     F3=Exit     F5=RptFind  F7=Up       F8=Down    F12=Cancel│
000 <──────────────────────────────────────────────────────────────────────┘
          000170               REDO-TAX.
00091C    000171                   PERFORM CALC-TAX.
          000173               CALC-TAX.
00093E    000174                   IF ELECTRIC
000954    000175                     COMPUTE BASE-AMOUNT = PRICE / CC
00097E    000176                     COMPUTE TAX-AMOUNT = PRICE / BASE-AMOUNT
00097E    000177                   ELSE
0009AC    000178                     COMPUTE BASE-AMOUNT = CC / CYLINDERS
0009D6    000179                     IF KW < 150
0009EA    000180                       MOVE 10 TO BASE-AMOUNT
0009EA    000181                     ELSE
0009F4    000182                       MOVE 20 TO BASE-AMOUNT
0009FA    000183                       IF ZERO-100 < 5
000A0E    000184                         ADD 5 TO BASE-AMOUNT
000A0E    000185                       ELSE
000A2C    000186                         IF RED
000A3E    000187                           ADD 2 TO BASE-AMOUNT
000A3E    000188                         ELSE
 F1=Help      F3=Exit      F5=RptFind   F6=AddBkp    F7=Up        F8=Down
F10=Left     F11=Right
```

The Add Label Breakpoint popup is used to add, change or clear a label breakpoint.

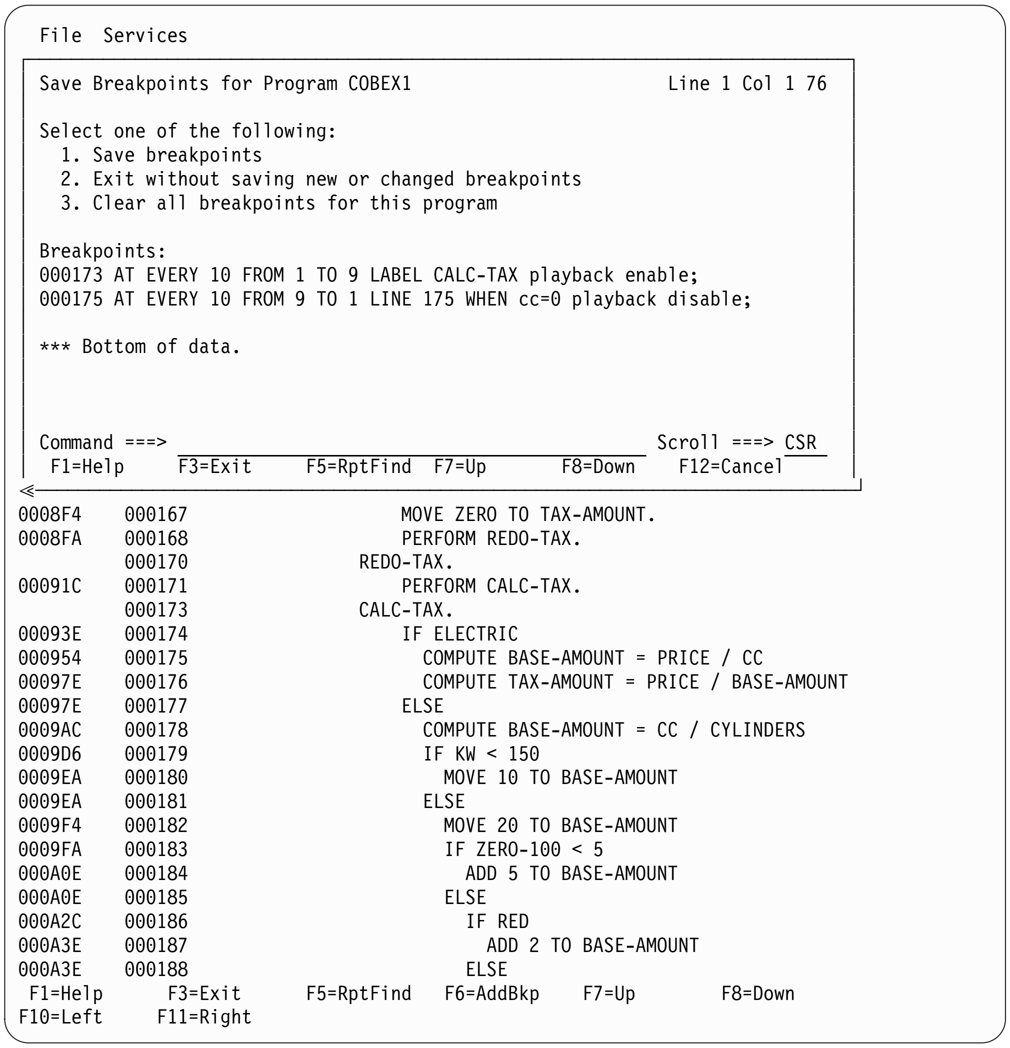
3 Line and label breakpoints may specify an EVERY clause and an Action.


```
 File  Services
┌───────────────────────────────────────────────────────────┐
│Breakpoints                                      Line 1 Col 1 76 │
│   ┌───────────────────────────────────────────────────────────┐
│ S │ Add Label Breakpoint                              Line 1 Col 1 76 │
│ s │                                                              │
│   │ Label:                                                       │
│   │   000173 CALC-TAX.                                           │
│ * │                                                              │
│   │   EVERY   10        FROM 1        TO 9                       │
│   │   WHEN    cc = 0                             4               │
│   │                                                              │
│   │                                                              │
│   │   Action  playback disable                                   │
│   │                                                              │
│   │                                                              │
│   │                                                              │
│ C │ Clear breakpoint? N  (Y/N)                                   │
│   │                                                              │
<── │                                                              │
000 │ Press ENTER to change/clear breakpoint.                      │
000 │                                                              │
    │ *** Bottom of data.                                          │
000 │                                                              │
    │  Command ===>                                  Scroll ===> CSR │
000 │  F1=Help     F3=Exit     F5=RptFind  F7=Up       F8=Down    F12=Cancel │
000 <──────────────────────────────────────────────────────────────┘
00097E    000176                    COMPUTE TAX-AMOUNT = PRICE / BASE-AMOUNT
00097E    000177                  ELSE
0009AC    000178                    COMPUTE BASE-AMOUNT = CC / CYLINDERS
0009D6    000179                    IF KW < 150
0009EA    000180                      MOVE 10 TO BASE-AMOUNT
0009EA    000181                    ELSE
0009F4    000182                      MOVE 20 TO BASE-AMOUNT
0009FA    000183                      IF ZERO-100 < 5
000A0E    000184                        ADD 5 TO BASE-AMOUNT
000A0E    000185                      ELSE
000A2C    000186                        IF RED
000A3E    000187                          ADD 2 TO BASE-AMOUNT
000A3E    000188                        ELSE
 F1=Help      F3=Exit      F5=RptFind   F6=AddBkp    F7=Up        F8=Down
F10=Left     F11=Right
```

Similarly, the Add Line Breakpoint popup is used to add, change or clear (delete) a line breakpoint.

4 Line breakpoints may also specify a WHEN condition.

On exiting IPVLANGP, the following popup appears:



```
 File  Services
┌─────────────────────────────────────────────────────────────────────────┐
│ Save Breakpoints for Program COBEX1                      Line 1 Col 1 76 │
│                                                                          │
│ Select one of the following:                                             │
│   1. Save breakpoints                                                    │
│   2. Exit without saving new or changed breakpoints                      │
│   3. Clear all breakpoints for this program                              │
│                                                                          │
│ Breakpoints:                                                             │
│ 000173 AT EVERY 10 FROM 1 TO 9 LABEL CALC-TAX playback enable;           │
│ 000175 AT EVERY 10 FROM 9 TO 1 LINE 175 WHEN cc=0 playback disable;      │
│                                                                          │
│ *** Bottom of data.                                                      │
│                                                                          │
│                                                                          │
│                                                                          │
│ Command ===> ____________________________________________ Scroll ===> CSR │
│  F1=Help     F3=Exit     F5=RptFind  F7=Up       F8=Down    F12=Cancel   │
≪─────────────────────────────────────────────────────────────────────────┘
0008F4    000167                   MOVE ZERO TO TAX-AMOUNT.
0008FA    000168                   PERFORM REDO-TAX.
          000170               REDO-TAX.
00091C    000171                   PERFORM CALC-TAX.
          000173               CALC-TAX.
00093E    000174                   IF ELECTRIC
000954    000175                     COMPUTE BASE-AMOUNT = PRICE / CC
00097E    000176                     COMPUTE TAX-AMOUNT = PRICE / BASE-AMOUNT
00097E    000177                   ELSE
0009AC    000178                     COMPUTE BASE-AMOUNT = CC / CYLINDERS
0009D6    000179                     IF KW < 150
0009EA    000180                       MOVE 10 TO BASE-AMOUNT
0009EA    000181                     ELSE
0009F4    000182                       MOVE 20 TO BASE-AMOUNT
0009FA    000183                       IF ZERO-100 < 5
000A0E    000184                         ADD 5 TO BASE-AMOUNT
000A0E    000185                       ELSE
000A2C    000186                         IF RED
000A3E    000187                           ADD 2 TO BASE-AMOUNT
000A3E    000188                         ELSE
 F1=Help      F3=Exit      F5=RptFind   F6=AddBkp    F7=Up        F8=Down
F10=Left     F11=Right
```

Breakpoints are saved back to the Repository in XML format. Use DTU to convert the breakpoint XML definitions to a z/OS Debugger commands file ready for use with the next debug session.

Deferred Breakpoints also feature in Fault Analyzer's COBOL Explorer.

Chapter 8. IPVLANGO Automatic Binary Optimizer LANGX file update utility

The Automatic Binary Optimizer for z/OS (ABO) product optimizes COBOL object code produced by the following compilers:

- Enterprise COBOL for z/OS Version 4
- Enterprise COBOL for z/OS Version 3
- COBOL for OS/390 & VM V2R2
- COBOL for OS/390 & VM V2R1
- COBOL for MVS & VM V1R2
- COBOL/370 1.1
- VS COBOL II V1.4.0 (LE enabled modules only)
- VS COBOL II V1.3.x (LE enabled modules only)

ABO optimization results in code changes that render any existing compiler listing or side file unusable with the optimized program. The IPVLANGO utility creates LANGX file members that can be used to provide source-level debugging of the optimized program with ADFz family of products such as Fault Analyzer, zDebug, and Application Performance Analyzer. New LANGX file members can be created from compiler listings, SYSDEBUG side files, or existing LANGX files.

The sample job step in Figure 3 on page 84 takes the listing transforms file from a previous ABO step and merges it with one or more LANGX file members to create 'optimized' LANGX members. (Refer to ABO documentation for the complete optimization JCL, which this sample job step can be appended to.) If the listing transforms file is a PDS(E), it must specify a member name. In Figure 3, the input is a LANGX data set (DD:IPVLANGX); alternatively, it could be a compiler listing (DD:IPVLCOB) or a SYSDEBUG data set (DD:IPVSYSDB).

As ABO can process multiple programs in a single invocation, the listing transforms file has a PROC section for each optimized program. To accommodate this, compiler listing, SYSDEBUG, and LANGX data set DDs should specify a PDS(E) data set without a member name. The input PDS(E) should contain a member for each PROC in the listing transforms file. Likewise, the output LANGX PDS(E) contains a member for each PROC in the listing transforms file.

The IPVLANGO utility uses the following DDs:

LISTING
: The (input) ABO listing transforms file that was written in the ABO step to SYSPRINT. This can be a sequential data set or a PDS(E) member.

IPVLANGX | IPVLCOB | IPVSYSDB
: The original (input) side file that represents one or more unoptimized programs. This must be a PDS(E) that contains a member for each program. Do not specify a member name. Use one of these DDs depending on the input side file format.

IPVLANGO
: The new (output) LANGX side file that represents one or more optimized programs. This must be a VB PDS(E) with LRECL>=1562 and must not be the same data set as the one specified for IPVLANGX. A member is written

for each PROC in the listing transforms file. Do not specify a member name. Note that this data set cannot subsequently be used as input to the IPVLANGO utility.

```
//LANGO EXEC PGM=IPVLANGO
//LISTING DD DISP=SHR,DSN=*.OPT.SYSPRINT   <--- Input ABO transforms file
//IPVLANGX DD DISP=SHR,DSN=JERRYBL.BINOPT.LANGX
//IPVLANGO DD DISP=SHR,DSN=JERRYBL.BINOPT.LANGX.ABO
```

Figure 3. Sample job step to create an 'optimized' LANGX side file

Chapter 9. Maintaining PD Tools Common Component

Take the following steps to apply maintenance to PDTCC:

1. If PDTCC SMP/E target libraries are in LINKLIST, remove them from LLA and VLF control before you perform the SMP/E APPLY. The removal is to avoid errors when modules are loaded from LINKLIST because SMP/E compressed or added extents to the libraries.
2. Perform SMP/E APPLY.
3. If the updated PDTCC modules are in LPA, do one of the following actions:
   - IPL with CLPA
   - Perform dynamic updates as follows:
     - If the PDTCC module IPVLANGX is placed in LPA by using the command **SETPROG**, which is opposed to placing IPV.SIPVLPA1 in LPA, take the following actions:
       a. Issue the following command:
          ```
          SETPROG LPA,DELETE,MOD=IPVLANGX),FORCE=YES
          ```
          For complete information about the command **SETPROG**, see MVS System Commands.
       b. Issue the following command:
          ```
          F LLA,REFRESH
          ```
       c. Optional: To add the IPVLANGX module to LPA and regain the region size space advantage, issue the following command:
          ```
          SETPROG LPA,ADD,MOD=(IPVLANGX),DSN=LNKLST
          ```
     - If IPV.SIPVLPA1 is included in your LPALIST, issue the following command:
       ```
       SETPROG LPA,ADD,MOD=(IPVLANGX),DSN=LNKLST
       ```

Chapter 10. PDTCC event processing

PDTCC event processing is a feature that allows any products or systems including ADFz family of products to send data to an asynchronous installation-written back-end for processing. Do not send sensitive information by using this feature. The validity of all data is the responsibility of the users.

The PDTCC event processing feature includes the following items:
- Sender load module IPVEPSND
- Receiver load module IPVEPRCV
- IPVCNF00 option EVENTPROCESSINGEXIT
- The Event Processing user exit

Sender load module IPVEPSND

The IPVEPSND load module contains the fetchable LE function IPVEPSND(). It spawns an extra BPX batch address space in which the module IPVEPRCV runs asynchronously.

This module transfers data to IPVEPRCV via stdin, and any debug information is passed back to it via stdout if the debug mode is active.

If this feature is enabled via the EventProcessingExit option, ADFz family of products transparently call IPVEPSND to perform event processing.

Note: If the debug mode is activated, IPVEPRCV does not run asynchronously to IPVEPSND.

Usage

IPVEPSND, the fetchable LE function IPVEPSND(), is defined as:

```
int IPVEPSND(char *ProdID, char *UsrPgm, char *BufPtr, char *DbgDDn);
```

The parameters are defined as follows:

ProdID The product ID. For example: the Fault Analyzer product ID is “IDI”; the File Manager product ID is “FMN”.

UsrPgm
The user program name. For ADFz family of products, the user program name is obtained via the `EVENTPROCESSINGEXIT` option.

BufPtr The buffer pointer. The 31-bit address of a storage buffer to be passed to the Event Processing exit in the following format:

Byte 0-3
Total data length

Byte 4 Segment data

Byte 0-1
Segment length

Byte 2 Segment data

Repeat segment length and segment data for all segments

DbgDDn
: The debug DDname.

The returned int value includes the following descriptive bytes:

Byte 0 IPVEPSND return code (0 = successful).

Byte 1 IPVEPRCV return code (0 = successful). This byte is valid only if the debug DDname is specified.

Byte 2-3
: User exit return code. This byte is valid only if the debug DDname is specified.

Example

```
void *fetch_ptr;
typedef void exit_U(char *ProdID, char *UsrPgm, void *BufPtr, char *DbgDDn);
#pragma linkage(exit_U, OS)
exit_U *exit_ep;
char *exit_name;
char event_data[1024];
char *data_item1 = "Fred=Yes";
char *data_item2 = "Barney=No";
int i;

i = 4;
*(short *)&event_data[i] = strlen(data_item1);
memcpy(data_buffer + i + 2, data_item1, strlen(data_item1));
i += (2 + strlen(data_item1));

*(short *)&event_data[i] = strlen(data_item2);
memcpy(data_buffer + i + 2, data_item2, strlen(data_item2));
i += (2 + strlen(data_item2));

*(int *)&event_data[0] = i;

// Set exit_name to the current EVENTPROCESSINGEXIT option value...

fetch_ptr = (void *)fetch("IPVEPSND");
if (fetch_ptr) {
  exit_ep = (exit_U *)fetch_ptr;
  exit_ep("XYZ", exit_name, event_data, 0);
}
```

Receiver load module IPVEPRCV

The IPVEPRCV load module is internal. It is the load module that is executed in the BPX batch address space that is spawned by IPVEPSND().

IPVEPRCV provides an interface to the user exit that is specified via the EVENTPROCESSINGEXIT option.

IPVCNF00 option EVENTPROCESSINGEXIT

Use the EventProcessingExit option to specify the name of the exit to be invoked to process an event. For more information, see "EventProcessingExit" on page 13.

The Event Processing user exit

The exit name that is specified in the IPVEPSND *UsrPgm* parameter must be a fetchable LE function with the following format:

```
int exit_name(char *ProdID, char *BufPtr);
```

The *ProdID* and *BufPtr* are the same as specified for IPVEPSND.

The user exit load module must be available via the normal search path, that is, JOBLIB, STEPLIB, LINKLIST, and so on.

The user exit is invoked in problem state, that is, key 8.

| For ADFz family of products, the exit name is specified via the EventProcessingExit option.

Example customer event processing user exit

The following code is a stub example of an exit written in C that can be specified via the EVENTPROCESSINGEXIT option, named as IPVEPXIT.

```
#pragma linkage(IPVEPXIT, fetchable)

#include <string.h>

int IPVEPXIT(char *ProdID, char *BufPtr) {

  if (!BufPtr) {
    printf("BufPtr is null! Exiting...\n");
    return;
  }

  if (strcmp(ProdID, "IDI")) {

    //Event from Fault Analyzer has been detected.
    //processFAEvent(BufPtr);  ...

  } else if (strcmp(ProdID, "FMN")) {

    //Event from File Manager has been detected.
    //processFMEvent(BufPtr);  ...

  } else if (strcmp(ProdID, "ABC")) {

    //Event from application ABC has been detected.
    //processABCEvent(BufPtr); ...

  }
}
```

Appendix A. Messages

Common Server messages

For the Common Server messages, selected batch messages are listed in alphanumeric order. For each message, the information that is provided comprises:
- The message identifier.
- The text of the message.
- An explanation of the message.
- The required user response.

The messages that are issued have a unique alphanumeric identifier with the format:

`IPV`*`nnnns`*

where:

nnnn	Is a 4-digit number.
s	Is a severity level indicator with the following meanings: • I - Informational • W - Warning • S - Severe

IPV0001I Server on port %i exiting

Explanation: The server is finished processing. Either errors occurred during startup, running, or the server is responding to a shutdown command.

System action: The server finishes processing.

User response: If the shutdown was unexpected, examine previous messages for the cause.

IPV0002I Error establishing SSL environment: %i

Explanation: An error occurred while establishing the SSL environment.

System action: The PD TOOLS Common Component Server attempts to continue.

User response: Examine previous messages for reasons for environment failure. If previous messages do not help, contact IBM support.

IPV0003S Console modify/stop interface failed rc=%i, errno=%i error= %s

Explanation: An error occurred while establishing the console interface.

System action: The PD TOOLS Common Component Server exits.

User response: Examine the provided error for reasons for failure. If previous messages do not help, contact IBM support.

IPV0004I Number of configurations %i

Explanation: During start or configuration refresh, the CONFIG data was read and the specified number of configurations were recognized.

System action: None.

User response: If the number of configurations is unexpected, check the CONFIG concatenations and contents.

IPV0005I Config number %i startup %s

Explanation: During start or configuration refresh, the configuration specified an initial program to run.

System action: None.

User response: None.

IPV0006W System call rc=%i error=%s

Explanation: A call to run a program according to a configuration failed.

System action: None.

User response: None.

IPV0007W Expected a portnumber integer. Received %s

Explanation: The server expects an integer portnumber as the first parameter.

System action: The server attempts to continue starting, using port 2800.

User response: Check the invocation parameter for the server.

IPV0008W Expected AF_INET or AF_INET6. Received %s

Explanation: The server expects the address family type as the second parameter.

System action: The server attempts to continue starting, using the AF_INET family.

User response: Check the invocation parameter for the server.

IPV0009I Using address family %s.

Explanation: The server is using the specified address family.

System action: None.

User response: None.

IPV0010I Using port %i.

Explanation: The server is using the specified port number.

System action: None.

User response: None.

IPV0011S listen() error: %s

Explanation: The listen call failed with the specified error.

System action: The server is shut down.

User response: Correct the listed error if possible and restart the server.

IPV0012W Spawn failure for %s. Error: %s __errno2 = %08x

Explanation: The attempt to spawn the specified program failed with the listed error and error code.

System action: The server continues to run.

User response: Examine the error and possibly examine the CONFIG file ensuring that customization occurred correctly.

IPV0013W Missing value for keyword '%s'

Explanation: While reading the CONFIG file, an expected value for a keyword was missing.

System action: The server continues to run.

User response: Check the CONFIG file for the specified keyword and specify an appropriate value.

IPV0014W Failure to acquire storage for configuration instance %i

Explanation: While preparing configurations, a failure to acquire storage occurred.

System action: The server attempts to continue to run.

User response: Check the REGION specification for the server. Increase and restart the server.

IPV0015I PDTCC Server Running on port %i.

Explanation: Console message to indicate that the server is now accepting connections.

System action: None.

User response: None.

IPV0016I Established SSL environment.

Explanation: The call to System SSL to initialize an environment was successful.

System action: None.

User response: None.

IPV0017W Unable to create temporary file %s. %s

Explanation: The call to create a temporary file for a configuration failed.

System action: The server attempts to continue, however the configuration might be unusable.

User response: Examine the file path and error condition as shown. Correct the configuration file or update the directory permissions and restart or refresh the server.

IPV0018W Unable to verify dsn %s

Explanation: The existence of data set %s in a STEPLIB= value could not be verified.

System action: The server attempts to continue, however the configuration might be unusable.

User response: Examine the named data set and ensure that it is the correct name. If necessary, update the configuration file and restart or refresh the server.

IPV0019W Unable to open CONFIG %s

Explanation: During startup, or a refresh command, the DD CONFIG was unable to be opened.

System action: If this occurs during initial start of the server, the server terminates. During a refresh, no new configurations are loaded.

User response: Examine the error and the CONFIG data sets to ensure that they exist. If necessary, update the configuration file and restart or refresh the server.

IPV0020I REFRESH completed, %i configs processed.

Explanation: A REFRESH console command has now completed. The server has re-read the configurations as specified in the CONFIG DD.

System action: None.

User response: None.

IPV0021W REFRESH found errors in new configs, not activated.

Explanation: A REFRESH console command was issued, but during reading of the CONFIG DD, some errors occurred.

System action: The server continues with its prior configuration.

User response: Check the server output for possible further information on the problems that are found in the CONFIG file(s)

IPV0022S Creation of key database at %s failed, error %s

Explanation: The configuration specifies that the server create a certificate to be used, however an error as described occurred when attempting to create the key database.

System action: The server terminates.

User response: f the error is an IO error, check the specified location for enough space (65KB). Otherwise, check that the location is writeable. To specify an alternate location, set the configuration keyword WORKDIR to the directory to be used.

IPV0023S Creation of self-signed certificate failed, error %s

Explanation: The configuration specifies that the server create a certificate to be used, however an error as described occurred when attempting to create the self-signed certificate in the key database.

System action: The server terminates.

User response: Check the listed error and check documentation for the gsk_create_self_signed_certificate API.

IPV0024I Traceon received, trace already active.

Explanation: The Server received a modify command to turn on tracing, but it is already on.

System action: None.

User response: None.

IPV0025I Traceon received, trace turned on.

Explanation: The Server received a modify command to turn on tracing and has done so. Trace output goes to the IPVTRACE file(DD) if present, or to the STDOUT file if not.

System action: None.

User response: None.

IPV0026I Traceoff received, trace already off.

Explanation: The Server received a modify command to turn off tracing but it is already off.

System action: None.

User response: None.

IPV0027I Traceoff received, trace turned off.

Explanation: The Server received a modify command to turn off tracing and has done so.

System action: None.

User response: None.

IPV0028I Unrecognized modify command.

Explanation: The Server received a modify command, but did not recognize it.

System action: None.

User response: Check that modify contained one of the valid requests; TRACEON, TRACEOFF, VER or REFRESH.

IPV0029W Client config name %s not found in CONFIG DD content.

Explanation: The Server received a client connection request for the named config, but no matching CONFIG=name statement was found in the data that was contained in the CONFIG DD concatenation.

System action: The client connection request is refused.

User response: Check that the configurations referenced by the CONFIG DD for the server, contain a CONFIG=name section.

IPV0030I API start PID=processid

Explanation: A process (processid) launched by the common server has invoked the common server subordinate API to start the environment setup and handshake with client.

System action: None.

User response: None.

IPV0031I API closure PID=processid

Explanation: A process has invoked the common server subordinate API to close the environment setup and client connection.

System action: None.

User response: None.

IPV0032I PDTCC Server Release=%s PTF=%s

Explanation: In response to the VER modify command, the server lists its release and PTF level information.

System action: None.

User response: None.

IPV0033W Unknown token %s with value %s for CONFIG=%s

Explanation: While processing the configuration file, an unrecognized token/value pair was found.

System action: The invalid token is ignored and processing attempts to continue.

User response: Review the configuration file for the named token. Look for a misspelling or incorrect token or value.

IPV0041W Maximum user variables (500) reached when processing token %s, value %s in configuration %s

Explanation: The limit of substitution values has been reached.

System action: The server attempts to continue, however the configurations might be unusable.

User response: Examine the number of $token=value pairs present in the configuration file and reduce to less than 500.

IPV0042W Unable to stat file %s.

Explanation: The server is unable to check the configuration launch file entry.

System action: The server attempts to continue, however this launch configuration is unusable.

User response: Examine the file path and ensure that the setup was completed correctly. Most likely the file or directory path is not owned or correctly permitted in order for this server instance to access the named file. The WORKDIR configuration step of installation needs to be checked and rerun.

IPV0043W Not owner of launch file %s.

Explanation: The server is not the owner of a configuration launch file entry.

System action: The server attempts to continue, however this launch configuration is unusable.

User response: Examine the file path and ensure that the setup was completed correctly. Correct the condition by ensuring that the file owner is updated to the userid of the server. The file system that the file is mounted on needs to allow SETUID for the owner to be changed with the chmod command.

IPV0044W Launch file %s is not marked as sticky.

Explanation: A configuration launch file has not been created correctly.

System action: The server attempts to continue, however this launch configuration is unusable.

User response: Examine the file path and WORKDIR location. If the WORKDIR is correct, the installation configuration step for the WORKDIR might need to be rerun.

IPVLANGX messages

These messages are issued by the IPVLANGX program, which is used internally by Fault Analyzer or invoked by the user when creating side files.

IDISF8001I IPVLANGX Version *version* **(Release** *release***)**

Explanation: This message shows the IPVLANGX program identification, version, and release date.

System action: Processing continues.

User response: None

IDISF8002I Output file: *member_name DDname*

Explanation: This message identifies the file to which the extract data information is written by IPVLANGX.

The *member_name* field is not included in the message if using a sequential file.

System action: Processing continues.

User response: None

IDISF8003I ... scanning *txt1*

Explanation: This message indicates that the information that was specified in *txt1* is being read from the associated file and processed.

System action: Processing continues.

User response: None

IDISF8004I ... checking *txt1*

Explanation: This message indicates that the information t at was specified in *txt1* is checked for consistency.

System action: Processing continues.

User response: None

IDISF8005I *txt1* **Pass** *dec2* **processing begins**

Explanation: This message indicates pass *dec2* of the multi-pass processing task that is specified in *txt1* is now being performed.

System action: Processing continues.

User response: None

IDISF8006I Post-processing begins

Explanation: This message indicates that all necessary information was read from the associated files, and post-processing of this information is being performed.

System action: Processing continues.

User response: None

IDISF8007I ... matching *txt1*

Explanation: This message indicates that the information that was specified in *txt1* is now being correlated.

System action: Processing continues.

User response: None

IDISF8008I ... performing *txt1*

Explanation: This message indicates that the processing step specified in *txt1* is now being performed.

System action: Processing continues.

User response: None

IDISF8010I *txt1* **records scanned:** *dec2*

Explanation: This message indicates that *dec2* records were read from the *txt1* file when the current compile unit was processed by IPVLANGX.

System action: Processing continues.

User response: None

IDISF8011I ...Symbols *txt1***..** *dec2*

Explanation: This message indicates that the current compile unit contained *dec2* symbols with characteristics of type *txt1*

System action: Processing continues.

User response: None

IDISF8012I ...Long Name Resolution IDs: *dec1*

Explanation: This message indicates that the current compile unit contained *dec1* Long Name Resolution Identifiers.

System action: Processing continues.

User response: None

IDISF8013I ...Total symbols: *dec1*

Explanation: This message indicates that the current compile unit contained *dec1* symbols.

System action: Processing continues.

User response: None

IDISF8014I Records written to output file: *dec1*

Explanation: This message shows the number of records of extract data information which were written to the output file.

System action: Processing continues.

User response: None

IDISF8015I Operation completed for this compile unit

Explanation: Processing was completed for the current compile unit.

System action: Processing continues if extra compile units are present.

User response: None

IDISF8016I *txt1 member_name DDname*

Explanation: This message identifies the input file(s) which were processed by IPVLANGX

The *txt1* field is normally "Input file:" or "Input files:".

The *member_name* field is not included in the message if using a sequential file.

System action: Processing continues.

User response: None

IDISF8017I Operation completed for this extract file

Explanation: This message is the last message to be displayed by IPVLANGX, and indicates that processing is completed for this IPVLANGX extract data file.

System action: Processing has completed.

User response: None

IDISF8018I *txt1* **bytes scanned:** *dec2*

Explanation: This message indicates that *dec2* bytes of data were read from the *txt1* file when the current compile unit was processed by IPVLANGX.

System action: Processing continues.

User response: None

IDISF8020I ...Blocks of dead code eliminated....... *dec1*

Explanation: This message indicates that *dec1* blocks of code which had been removed by optimization by the compiler have been identified. The source code and variables that are associated with these blocks have been eliminated from the extract data.

System action: Processing continues.

User response: None

IDISF8050W Argument missing for *txt1* **option.** *txt2*

Explanation: The argument for IPVLANGX option *txt1* was not found during processing of the IPVLANGX invocation parameters.

System action: The default argument for the *txt1* option is assumed.

User response: If you are using IPVLANGX directly to create a side file, specify the invocation options as explained in Chapter 6, "IPVLANGX compiler listing to side file conversion utility," on page 69. If the message is issued during fault analysis, contact your IBM service representative.

IDISF8051S Argument/Option too long, "*txt1***"**

Explanation: The invocation parameter *txt1* is not recognized as a valid IPVLANGX argument (or option). It exceeds the maximum length of a valid argument (or option), and might be spelled incorrectly.

System action: Processing is terminated.

User response: If you are using IPVLANGX directly to create a side file, specify the invocation options as explained in Chapter 6, "IPVLANGX compiler listing to side file conversion utility," on page 69. If the message is issued during fault analysis, contact your IBM service representative.

IDISF8052S Argument/Option not recognized, "*txt1***"**

Explanation: The invocation parameter *txt1* is not recognized as a valid IPVLANGX argument (or option).

System action: Processing is terminated.

User response: If you are using IPVLANGX directly to create a side file, specify the invocation options as explained in Chapter 6, "IPVLANGX compiler listing to side file conversion utility," on page 69. If the message is issued during fault analysis, contact your IBM service representative.

IDISF8055S A left parenthesis was found inside options

Explanation: An extra left parenthesis (after the initial left parenthesis which signals the start of the IPVLANGX options) was encountered during processing of the IPVLANGX invocation parameters.

System action: Processing is terminated.

User response: If you are using IPVLANGX directly to create a side file, specify the invocation options as explained in Chapter 6, "IPVLANGX compiler listing to side file conversion utility," on page 69. If the message is issued during fault analysis, contact your IBM service representative.

IDISF8056S No file name was specified

Explanation: The PDS or PDSE data set member name of the primary program information file from which source and variable data is to be extracted was not found during processing of the IPVLANGX invocation parameters.

System action: Processing is terminated.

User response: If you are using IPVLANGX directly to create a side file, specify the invocation options as explained in Chapter 6, “IPVLANGX compiler listing to side file conversion utility,” on page 69. If the message is issued during fault analysis, contact your IBM service representative.

IDISF8057S Argument/Option already specified, "*txt1*"

Explanation: The argument (or option) *txt1* was encountered more than once during processing of the IPVLANGX invocation parameters.

System action: Processing is terminated.

User response: If you are using IPVLANGX directly to create a side file, specify the invocation options as explained in Chapter 6, “IPVLANGX compiler listing to side file conversion utility,” on page 69. If the message is issued during fault analysis, contact your IBM service representative.

IDISF8058S Argument/Option "*txt1*" conflicts with previous Argument/Option

Explanation: A conflict between the argument (or option) *txt1* and another previously specified argument (or option) was detected during processing of the IPVLANGX invocation parameters.

System action: Processing is terminated.

User response: If you are using IPVLANGX directly to create a side file, specify the invocation options as explained in Chapter 6, “IPVLANGX compiler listing to side file conversion utility,” on page 69. If the message is issued during fault analysis, contact your IBM service representative.

IDISF8059I Application language not specified, option "*txt1*" assumed

Explanation: In the absence of an IPVLANGX option which explicitly specifies the application programming language, the IPVLANGX option *txt1* was assumed.

System action: Processing continues.

User response: If you are using IPVLANGX directly to create a side file, specify the invocation options as explained in Chapter 6, “IPVLANGX compiler listing to side file conversion utility,” on page 69. If the message is issued during fault analysis, contact your IBM service representative.

IDISF8100S *txt1* contains NO recognized records

Explanation: The input file that was identified in *txt1* did not contain the expected records. This could happen if, for example, the IDIADATA DDname was accidentally directed at a compiler listing data set.

System action: Processing is terminated.

User response: Ensure that the input file that was used is correct.

IDISF8103S *txt1* has unrecognized records following last valid section

Explanation: The input file for IPVLANGX (the compiler listing) contains more output than just from the compile step. That is, there might be precompiler or postcompiler output, such as from a DB2 precompiler step or a link-edit step. Once this information is removed, the message is no longer be produced and the side file should be created as expected.

System action: Processing is terminated.

User response: Ensure that the input file that was used is correct.

IDISF8110W Compiler option(s) incorrectly specified

Explanation: The format of the input file is insufficient for IPVLANGX processing because one or more of the required compiler options have not been specified.

System action: Processing is terminated.

User response: Recompile the module with the required compiler options as specified in “Preparing your programs” on page 19.

IDISF8114S *txt1* required for source support - fatal

Explanation: *txt1* specifies the compiler options that are required for a successful IPVLANGX execution. Source code information cannot be complete without these options. This message might be preceded by message IDISF8110W.

System action: Processing is terminated.

User response: Recompile the module with the required compiler options as specified in “Preparing your programs” on page 19.

IDISF8115W *txt1* required for symbol support

Explanation: *txt1* specifies the compiler options that are required for a successful IPVLANGX execution. Source code information cannot be complete without these options. This message is preceded by message IDISF8110W.

System action: Processing continues, however, the analysis report might be missing information.

User response: Recompile the module with the required compiler options as specified in “Preparing your programs” on page 19.

IDISF8116W *txt1* **required for structure/union support**

Explanation: *txt1* specifies the compiler options that are required for a successful IPVLANGX execution. Source code information cannot be complete without these options. This message is preceded by message IDISF8110W.

System action: Processing continues, however, the analysis report might be missing information.

User response: Recompile the module with the required compiler options as specified in “Preparing your programs” on page 19.

IDISF8120W *txt1* **detected.** *txt2* **option assumed**

Explanation: The format of the input file indicates that the specified option is no longer in effect.

System action: Processing continues, assuming an appropriate option to match the format of the input file.

User response: Use the correct compiler option, or make the compiler directive which adjusted the compiler option visible to IPVLANGX, as appropriate. If the problem persists, contact your IBM service representative.

IDISF8130S File not found "*txt1***"**

Explanation: The IPVLANGX input compiler listing or SYSADATA file *txt1* could not be found to allow IPVLANGX processing to begin.

System action: Processing is terminated.

User response: Correct the file specification, or make the file available to IPVLANGX, as appropriate. If the problem persists, contact your IBM service representative.

IDISF8131S Files not found "*txt1***", and "***txt2***"**

Explanation: The IPVLANGX extract data file could not be found using either the primary file identifier *txt1*, or the alternative file identifier *txt2* to allow IPVLANGX processing to begin.

System action: Processing is terminated.

User response: Correct the file specification, or make the file available to IPVLANGX, as appropriate. If the problem persists, contact your IBM service representative.

IDISF8132S Input or Output file format invalid

Explanation: The attributes or contents of a file have been found to be inappropriate, during IPVLANGX processing.

One or more preceding messages identify the file which was being processed when the error occurred or the reason for the failure. Reasons for this message might be error messages in the input compiler listing or missing required compiler options. For details on required compiler options, refer to “Preparing your programs” on page 19.

System action: Processing is terminated.

User response: Correct the problem that was identified in the preceding message. If the problem persists, contact your IBM service representative.

IDISF8133S File DD not allocated "*txt1***"**

Explanation: The Data Definition (DD) for the *txt1* file was found to be unallocated.

System action: Processing is terminated.

User response: Allocate the file, using a JCL DD statement, or TSO ALLOCATE statement, as appropriate. If the problem persists, contact your IBM service representative.

IDISF8134S File DDs not allocated "*txt1***", and "***txt2***"**

Explanation: The Data Definitions (DDs) for both the primary *txt1* file and the alternative *txt2* file were found to be unallocated.

System action: Processing is terminated.

User response: Allocate the file, using a JCL DD statement, or TSO ALLOCATE statement, as appropriate. If the problem persists, contact your IBM service representative.

IDISF8135S *txt1* **file incorrectly defined**

Explanation: The attributes of the *txt1* file have been examined, and found to be inappropriate.

System action: Processing is terminated.

User response: Ensure that the correct data set was specified in the *txt1* file allocation. If the correct data set was specified, the data set was allocated with incorrect attributes, in which case it must be reallocated. If the problem persists, contact your IBM service representative.

IDISF8136S Premature *txt1* **End-of-File encountered**

Explanation: IPVLANGX had begun scanning the *txt1* file data, but the file ended before all expected data records had been scanned.

System action: Processing is terminated.

User response: Ensure that the correct data set was specified in the *txt1* file allocation. If the correct data set was specified, the file might have been truncated and must be replaced with the complete data. If the problem persists, contact your IBM service representative.

IDISF8137S ***txt1*** **disk/directory is full**

Explanation: There is insufficient space to write further records to the *txt1* file.

This might be caused by :

- PDS directory has no free entries
- data set has maximum number of extents
- insufficient free space on the DASD volume for another extent

System action: Processing is terminated.

User response: Determine the resource which is exhausted, and correct as appropriate. If the problem persists, contact your IBM service representative.

IDISF8138T **Insufficient virtual memory available**

Explanation: There is insufficient free storage for IPVLANGX to continue processing.

System action: Processing is terminated.

User response: Free up virtual storage which is in use, or make more virtual storage available, as appropriate. If the problem persists, contact your IBM service representative.

Note: IPVLANGX uses storage above the 16MB line, if it is available.

IDISF8139S **File is TERSEd or PACKed "*txt1*"**

Explanation: The specified file was found to have a Fixed record format, and 1024-byte record length. It was likely compressed using TERSE or COPYFILE.

System action: Processing is terminated.

User response: Restore the file to its original format, using the appropriate utility program. If the problem persists, contact your IBM service representative.

IDISF8150T **Maximum number of symbols exceeded**

Explanation: The maximum number of symbols that a single compile unit can contain is 65534. This limit is exceeded by the current compile unit.

System action: Processing is terminated.

User response: Reduce the number of symbols below the limit. If the problem persists, contact your IBM service representative.

IDISF8152W **Incomplete info for symbol "*txt1*" (ident: *dec2*)**

Explanation: During the extraction process, complete information was not available for the symbol shown. The extract data for unrelated symbols and program source is not affected.

System action: Processing continues.

User response: Use IPVLANGX to format the extract data, and determine the missing information. Given this, examine the IPVLANGX input file(s) and determine the cause of the problem. If the problem persists, contact your IBM service representative.

IDISF8158T **Invalid COBOL source column indicators - fatal**

Explanation: Expected source column indicators were not found in the COBOL listing.

System action: Processing is terminated.

User response: Check if the attributes of file read have changed from those of the original compiler listing file.

IDISF8231S **Missing *txt1* ESD information**

Explanation: The name of a CSECT could not be determined.

System action: Processing continues but analysis might be incomplete.

User response: Ensure that CSECTs are named in accordance with the requirements in “Preparing your programs” on page 19.

IDISF8233S **Unable to determine identity of unnamed PC Section**

Explanation: The name of a CSECT could not be determined.

System action: Processing continues but analysis might be incomplete.

User response: Ensure that CSECTs are named in accordance with the requirements in “Preparing your programs” on page 19.

IDISF8250A **SYSADATA input record *record-number*, invalid ESDID ignored**

Explanation: An invalid ESDID was encountered on a SYSADATA record read. The ESDID was ignored.

System action: Processing continues.

User response: None.

IPVLANGX return codes

The following return codes are issued by IPVLANGX:

RC	Meaning
0	Operation successful, output file was written.
0*xxx*	Error was discovered while parsing arguments/options. *xxx* can have these values:

1	Token too long
2	Left parenthesis found inside options
3	Unknown option

1*xyy* **or 2***xyy*	Error occurred during scan of compiler listing or SYSADATA file.
3*xyy*	Error occurred while writing output file.

For return codes 1*xyy*, 2*xyy*, and 3*xyy*, the values for *xyy* are:

0*yy*	*yy* is the return code from the file WRITE routine
1*yy*	*yy* is the return code from the file OPEN routine
2*yy*	*yy* is the return code from the file READ routine
3*yy*	*yy* is the return code from the file WRITE routine
4*yy*	*yy* is the return code from the file POINT routine
5*yy*	*yy* is the return code from the memory ALLOCATE routine
6*yy*	*yy* is the return code from the memory FREE routine
7*yy*	*yy* is the return code from the file CLOSE routine
8*yy*	*yy* is the return code from the file NOTE routine

Examples of IPVLANGX return codes

0310 Compiler listing file is not in the expected format. A possible reason is that the required compiler options have not been used.

1128 Input compiler listing file could not be found. A possible reason is that a member name of a PDS(E) data set has not been specified, either in the parameters for IPVLANGX, or added to the data set name of the PDS(E).

3128 Output IPVLANGX file could not be found, or the attributes of an existing file do not match those required by IPVLANGX (RECFM=VB and LRECL≥1562).

3315 One or more records that were written to IPVLANGX were truncated due to insufficient logical record length. The minimum required logical record length for the IPVLANGX data set is 1562 bytes. Unpredictable results might occur if attempting to use the truncated side file as input.

Appendix B. Troubleshooting

Error scenarios and tracing

If the installed library has not been added to program control, this message appears in the JESMSGLG for the server task:

```
ICH420I PROGRAM IPVSRV FROM LIBRARY IPV.V1R6M0.SIPVMODA CAUSED
THE ENVIRONMENT TO BECOME UNCONTROLLED.  BPXP014I ENVIRONMENT MUST
BE CONTROLLED FOR SERVER (BPX.SERVER) PROCESSING.
```

Messages similar to the following might be generated if the user connecting to the server does not have read access to the SIPVMODA library:

```
ICH408I USER(VIKRAMM ) GROUP(USERCOD ) NAME(MANCHALA, VIKRAM ) 218
IPV.V160.SIPVMODA CL(DATASET ) VOL(COD035)
INSUFFICIENT ACCESS AUTHORITY
FROM IPV.V160.* (G)
ACCESS INTENT(READ ) ACCESS ALLOWED(NONE )
IEC150I 913-38,IFG0194E,VIKRAMM,OS390,ISP19502,8E10,COD035,IPV.V160.SIPVMODA
```

Messages on SYSLOG at the time of attempted connection, like the ones that are shown here, are generated when the relevant CONFIG contains an invalid library, or is missing a library from the SPAWN_STEPLIB statement:

```
IEA995I SYMPTOM DUMP OUTPUT
SYSTEM COMPLETION CODE=EC6  REASON CODE=0B26C032
 TIME=11.37.04  SEQ=38113  CPU=0000  ASID=00ED
 PSW AT TIME OF ERROR  070C3000   82C44CE8  ILC 2  INTC 0D
   NO ACTIVE MODULE FOUND
   NAME=UNKNOWN
   DATA AT PSW  02C44CE2 - C06C18F2  0A0D41B0  D4D0180B
   AR/GR 0: 00000000/00000026_00000648   1: 00000000/00000000_04EC6000
         2: 01FF000C/00000000_0B26C032   3: 00000000/00000000_8286F5B8
         4: 00000000/00000000_00000000   5: 00000000/00000000_00000000
         6: 01FF000C/00000000_00000700   7: 01FF000C/00000000_09BFC3F8
         8: 00000000/00000000_11F4B610   9: 00000000/00000000_163031FF
         A: 00000000/00000000_11F4B610   B: 01FF000C/00000000_7FFC3A00
         C: 00000000/00000000_02C47AC0   D: 00000000/00000000_16302200
         E: 00000000/00000000_82C44CB0   F: 00000000/00000000_0B26C032
 END OF SYMPTOM DUMP
```

If the above are not occurring, but connections are still not successful, shutdown the server and start it again with tracing active. If using the supplied sample, this can be done on the start command. For example, S IPVSRV,TRACE=D. This produces trace entries in the server task on the IPVTRACE DD.

A typical trace, with SSL active, before connections are made, looks similar to the one shown here. The main entries of interest confirming startup was successful are highlighted:

```
2012-04-10-10:54:39.442 [IPVSRV:266] Server built at: Apr 10 2012 10:54:03
2012-04-10-10:54:39.601 [IPVSRV:952] Record in length:1903
2012-04-10-10:54:39.601 [IPVSRV:969] Token: CONFIG Value: DEFAULT
2012-04-10-10:54:39.601 [IPVSRV:989] Config DEFAULT allocated.
2012-04-10-10:54:39.601 [IPVSRV:969] Token: SSL_REQUIRED Value: YES
2012-04-10-10:54:39.601 [IPVSRV:969] Token: WORKDIR Value: /etc/ipv/v17/ipvsrv1
2012-04-10-10:54:39.602 [IPVSRV:1070] Confirmed temporary write access ok dir=/etc/ipv/v17/ipvsrv1.
2012-04-10-10:54:39.602 [IPVSRV:969] Token: SPAWN_STEPLIB Value: IPV16SVC.SOPERW.LOAD ...
2012-04-10-10:54:39.602 [IPVSRV:969] Token: CONFIG Value: FM
2012-04-10-10:54:39.602 [IPVSRV:989] Config FM allocated.
2012-04-10-10:54:39.602 [IPVSRV:969] Token: SPAWN_PROGRAM Value: FMNCSEP
2012-04-10-10:54:39.602 [IPVSRV:1089] Creating temp filename.
2012-04-10-10:54:39.602 [IPVSRV:1106] Created temporary spawn image file ok.
2012-04-10-10:54:39.602 [IPVSRV:1116] spawn_program /etc/ipv/v17/ipvsrv1/FMNCSEP
2012-04-10-10:54:39.602 [IPVSRV:1117] spawn_fn FMNCSEP
```

```
2012-04-10-10:54:39.602 [IPVSRV:969] Token: SPAWN_JOBNAME Value: FMCLIENT
2012-04-10-10:54:39.602 [IPVSRV:969] Token: SPAWN_STEPLIB Value: FMN.V12R1M0.OPTIONS...
2012-04-10-10:54:39.602 [IPVSRV:969] Token: SPAWN_PARMS_SECTION Value:
2012-04-10-10:54:39.602 [IPVSRV:969] Token: CONFIG Value: UTPLX
2012-04-10-10:54:39.602 [IPVSRV:989] Config UTPLX allocated.
2012-04-10-10:54:39.602 [IPVSRV:969] Token: SPAWN_PROGRAM Value: IPVPSAMP
2012-04-10-10:54:39.602 [IPVSRV:1089] Creating temp filename.
2012-04-10-10:54:39.602 [IPVSRV:1106] Created temporary spawn image file ok.
2012-04-10-10:54:39.602 [IPVSRV:1116] spawn_program /etc/ipv/v17/ipvsrv1/IPVPSAMP
2012-04-10-10:54:39.602 [IPVSRV:1117] spawn_fn IPVPSAMP
2012-04-10-10:54:39.602 [IPVSRV:969] Token: SPAWN_STEPLIB Value: FMN12SVC...
2012-04-10-10:54:39.602 [IPVSRV:969] Token: SPAWN_PARMS_SECTION Value:
2012-04-10-10:54:39.602 [IPVSRV:969] Token: CONFIG Value: UTCAPI
2012-04-10-10:54:39.602 [IPVSRV:989] Config UTCAPI allocated.
2012-04-10-10:54:39.602 [IPVSRV:969] Token: SPAWN_PROGRAM Value: IDIGMAIN
2012-04-10-10:54:39.602 [IPVSRV:1089] Creating temp filename.
2012-04-10-10:54:39.602 [IPVSRV:1106] Created temporary spawn image file ok.
2012-04-10-10:54:39.602 [IPVSRV:1116] spawn_program /etc/ipv/v17/ipvsrv1/IDIGMAIN
2012-04-10-10:54:39.602 [IPVSRV:1117] spawn_fn IDIGMAIN
2012-04-10-10:54:39.602 [IPVSRV:969] Token: SPAWN_STEPLIB Value: FMN12SVC...
2012-04-10-10:54:39.602 [IPVSRV:969] Token: SPAWN_PARMS_SECTION Value:
2012-04-10-10:54:40.495 [IPVSRV:1956] Environment open rc=0 Handle=16AB09A8 Ha=16AA6490
2012-04-10-10:54:40.495 [IPVSRV:1965] Set SSLV2 off rc=0
2012-04-10-10:54:40.495 [IPVSRV:1973] Set SSLV3 off rc=0
2012-04-10-10:54:40.495 [IPVSRV:1982] Set TLSV1 on rc=0
2012-04-10-10:54:40.495 [IPVSRV:1997] Certfile=/etc/ipv/v17/ipvsrv1/IPVSRVC3-IPVCERT.kdb
2012-04-10-10:54:40.495 [IPVSRV:1998] Set keyring rc=0
2012-04-10-10:54:40.495 [IPVSRV:2006] Set pw rc=0
2012-04-10-10:54:40.511 [IPVSRV:2014] Environment init rc=0 Handle=16AB09A8
2012-04-10-10:54:40.511 [IPVSRV:281] Mixed case password support is off
2012-04-10-10:54:40.512 [IPVSRV:1902] Set socket linger rc=0
2012-04-10-10:54:40.512 [IPVSRV:1906] Set socket reuseaddr rc=0
2012-04-10-10:54:40.512 [IPVSRV:1910] Set socket keepalive rc=0
2012-04-10-10:54:40.512 [IPVSRV:301] Launching accept thread socket 0, listen code 0
2012-04-10-10:54:40.512 [IPVSRV:513] Acceptor thread running.
2012-04-10-10:54:40.512 [IPVSRV:527] About to accept.
```

If the highlighted statements are similar to the example that is shown here, all rc=0, then try to connect.

Several trace entries that are created by the server are similar to the ones that are shown here. Again, those that are of interest are highlighted.

```
2012-04-10-10:55:02.943 [IPVSRV:543] Connect received.
2012-04-10-10:55:02.943 [IPVSRV:549] Set client socket linger rc=0
2012-04-10-10:55:02.944 [IPVSRV:570] Thread launch
2012-04-10-10:55:02.944 [IPVSRV:527] About to accept.
2012-04-10-10:55:02.944 [IPVSRV:428] Conversation thread started.
2012-04-10-10:55:02.944 [IPVSRV:451] Server and peer on different hosts.
2012-04-10-10:55:02.944 [IPVSRV:1461] Outgoing message length=111, message=SSL=Y,
SERVERVERSION=01.01,SERVERNAME=IPVSRVC3,SYSNAME=z/OS,NODENAME=FMD2,
RELEASE=11.00,VERSION=01,MACHINE=2094
2012-04-10-10:55:02.944 [IPVSRV:1524] Sent 115 bytes
2012-04-10-10:55:02.945 [IPVSRV:2028] gsk_secure_socket_open rc=0
2012-04-10-10:55:02.945 [IPVSRV:2040] Set native socket rc=0
2012-04-10-10:55:02.945 [IPVSRV:2049] Set keyring label PDTCC Server Certificate rc=0
2012-04-10-10:55:02.945 [IPVSRV:2058] Set session type rc=0
2012-04-10-10:55:03.980 [IPVSRV:2081] Secure socket init rc=0
2012-04-10-10:55:03.980 [IPVSRV:1328] RecvSSL
2012-04-10-10:55:04.191 [IPVSRV:1360] Header indicates length 50
2012-04-10-10:55:04.722 [IPVSRV:1423] Incoming message: >>user=SOPERW3 pass=AXXXXXXX
config=UTCAPI DEBUG=YES<<
2012-04-10-10:55:04.723 [IPVSRV:1640] Uppercasing password 8 chars
2012-04-10-10:55:04.723 [IPVSRV:588] process_launch trying to match config UTCAPI.
2012-04-10-10:55:04.723 [IPVSRV:657] Parms: SOCKETH=00000001 ....
2012-04-10-10:55:04.723 [IPVSRV:658] Steplib: STEPLIB=IPV16SVC ...
2012-04-10-10:55:04.723 [IPVSRV:1828] Authenticated ok for user SOPERW3.
2012-04-10-10:55:04.724 [IPVSRV:1461] Outgoing message length=7, message=AUTH=Y
2012-04-10-10:55:04.724 [IPVSRV:1524] Sent 11 bytes
2012-04-10-10:55:05.282 [IPVSRV:702] gsk_secure_socket_close okay
2012-04-10-10:55:05.285 [IPVSRV:739] Spawned /etc/ipv/v17/ipvsrv1/IDIGMAIN Process 83886421
2012-04-10-10:55:05.285 [IPVSRV:745] Close client sock rc=0
```

If the Spawned trace line is present, check the SYSLOG at the time of the spawn for any messages that are issued by a started task. If there are no log messages, then look for output that is produced by the spawned user. For instance, in the example that is shown here, the user SOPERW3 has generated some output. Once you have this information, and the servers IPVTRACE output, contact IBM support.

Support resources and problem-solving information

This section shows you how to quickly locate information to help answer your questions and solve your problems. If you have to call IBM support, this section provides information that you need to provide to the IBM service representative to help diagnose and resolve the problem.

For a comprehensive multimedia overview of IBM software support resources, see the IBM Education Assistant presentation "IBM Software Support Resources for System z Enterprise Development Tools and Compilers products" at http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp?topic=/com.ibm.iea.debugt/debugt/6.1z/TrainingEducation/SupportInfoADTools/player.html.

- "Searching IBM support Web sites for a solution"
- "Obtaining fixes" on page 104
- "My Notifications" on page 105
- "Receiving support updates through RSS feeds" on page 106
- "If you need to contact IBM Software Support" on page 106

Searching IBM support Web sites for a solution

You can search the available knowledge bases to determine whether your problem was already encountered and is already documented.

- "Searching the information center"
- "Searching product support documents"
- "IBM Support Assistant" on page 104

Searching the information center

You can find this publication and documentation for many other products in the IBM System z Enterprise Development Tools & Compilers information center at http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp. Using the information center, you can search product documentation many ways. You can search across the documentation for multiple products, search across a subset of the product documentation that you specify, or search a specific set of topics that you specify within a document. Search terms can include exact words or phrases, wild cards, and Boolean operators.

To learn more about how to use the search facility that is provided in the IBM System z Enterprise Development Tools & Compilers information center, you can view the multimedia presentation at http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp?topic=/com.ibm.help.doc/InfoCenterTour800600.htm.

Searching product support documents

Use the System z Enterprise Development Tools & Compilers information center or the IBM support site at www.ibm.com/software/support to search for the latest, most complete information that might help you resolve your problem.

When you access the IBM support site, you can specify any of the following products for which you want information to be displayed:

- Application Performance Analyzer for z/OS
- Debug Tool for z/OS
- Enterprise COBOL for z/OS
- Enterprise PL/I for z/OS
- Fault Analyzer for z/OS
- File Manager for z/OS
- Optim Move for DB2
- WebSphere Developer Debugger for System z
- Workload Simulator for z/OS and OS/390 Support

When you access the IBM support site, you can also use the IBM Support Portal to customize the support information to be displayed and save product names that you specify. There is also a search facility provided with the IBM Support Portal that allows you to narrow the search scope and search only product support documents for the products that you specify. The IBM Support Portal can be accessed through the IBM support site at www.ibm.com/software/support or directly at www.ibm.com/support/entry/portal. For information about customizing your IBM support site experience using the IBM Support Portal, refer to https://www.ibm.com/blogs/SPNA/entry/the_ibm_support_portal_videos?lang=en_us.

IBM Support Assistant

The IBM Support Assistant (also referred to as ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. It provides quick access to support-related information. You can use the IBM Support Assistant to help you in the following ways:
- Search through IBM and non-IBM knowledge and information sources across multiple IBM products to answer a question or solve a problem.
- Find more information through product and support pages, customer news groups and forums, skills and training resources and information about troubleshooting and commonly asked questions.

In addition, you can use the built-in Updater facility in IBM Support Assistant to obtain IBM Support Assistant upgrades and new features to add support for software products and capabilities as they become available.

For more information, and to download and start using the IBM Support Assistant for IBM System z Enterprise Development Tools & Compilers products, visit http://www.ibm.com/support/docview.wss?rs=2300&context=SSFMHB&dc=D600&uid=swg21242707&loc=en_US&cs=UTF-8&lang=en.

General information about the IBM Support Assistant can be found on the IBM Support Assistant home page at http://www.ibm.com/software/support/isa.

Obtaining fixes

A product fix might be available to resolve your problem. To determine what fixes and other updates are available, the following information is available from the IBM support site. You can also view the following information from the IBM Support Portal when you specify the applicable products.
- Latest PTFs for Application Performance Analyzer for z/OS
- Latest PTFs for Debug Tool for z/OS

- Latest PTFs for Fault Analyzer for z/OS
- Latest PTFs for File Export for z/OS
- Latest PTFs for File Manager for z/OS
- Latest fixes for Optim Move for DB2
- Latest PTFs for WebSphere Studio Asset Analyzer for Multiplatforms
- Latest PTFs for Workload Simulator for z/OS and OS/390

When you find a fix that you are interested in, click the name of the fix to read its description and to optionally download the fix.

The IBM Support Portal is a way for you to specify specific products for which you want to display support information. The Support Portal can be accessed through the IBM support site at www.ibm.com/software/support or directly at www.ibm.com/support/entry/portal. For information about how to customize your IBM support site experience using the IBM Support Portal, refer to https://www.ibm.com/blogs/SPNA/entry/the_ibm_support_portal_videos?lang=en_us.

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

My Notifications

With My Notifications, you can subscribe to Support updates for any IBM product. You can specify that you want to receive daily or weekly email announcements. You can specify what type of information you want to receive (such as publications, hints and tips, product flashes (also known as alerts), downloads, and drivers). My Notifications enables you to customize and categorize the products about which you want to be informed and the delivery methods that best suit your needs.

To subscribe to Support updates, follow the steps below.

1. Click My notifications to get started. Click **Subscribe now!** on the page.
2. Sign in My notifications with your IBM ID. If you do not have an IBM ID, create one ID by following the instructions.
3. After you sign in My notifications, enter the name of the product that you want to subscribe in the **Product lookup** field. The look-ahead feature lists products matching what you typed. If the product does not appear, use the **Browse for a product** link.
4. Next to the product, click the **Subscribe** link. A green check mark is shown to indicate the subscription is created. The subscription is listed under Product subscriptions.
5. To indicate the type of notices for which you want to receive notifications, click the **Edit** link. To save your changes, click the **Submit** at the bottom of the page.
6. To indicate the frequency and format of the email message you receive, click **Delivery preferences**. Then, click **Submit**.
7. Optionally, you can click the RSS/Atom feed by clicking **Links**. Then, copy and paste the link into your feeder.
8. To see any notifications that were sent to you, click **View**.

Receiving support updates through RSS feeds

To receive RSS feeds about fixes and other software support news, go to the following web site and select the products in which you are interested:

- http://www.ibm.com/software/support/rss/other/index.html.

If you need to contact IBM Software Support

IBM Software Support provides assistance with product defects.

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

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

  Online

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

  By phone

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

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

To contact IBM Software support, follow these steps:

1. "Determining the business impact" on page 107
2. "Describing problems and gathering information" on page 107
3. "Submitting problems" on page 108

Determining the business impact

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

Severity 1
The problem has a **critical** business impact. You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.

Severity 2
The problem has a **significant** business impact. The program is usable, but it is severely limited.

Severity 3
The problem has **some** business impact. The program is usable, but less significant features (not critical to operations) are unavailable.

Severity 4
The problem has **minimal** business impact. The problem causes little impact on operations, or a reasonable circumvention to the problem was implemented.

Describing problems and gathering information

When describing a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Software Support specialists can help you solve the problem efficiently.

To save time, if there is a Mustgather document available for the product, refer to the Mustgather document and gather the information specified. Mustgather documents contain specific instructions for submitting your problem to IBM and gathering information needed by the IBM support team to resolve your problem. To determine if there is a Mustgather document for this product, go to the product support page and search on the term Mustgather. At the time of this publication, the following Mustgather documents are available:

- Mustgather: Read first for problems that are encountered with Application Performance Analyzer for z/OS: http://www.ibm.com/support/docview.wss?rs=2300&context=SSFMHB&q1=mustgather&uid=swg21265542&loc=en_US&cs=utf-8&lang;=en
- Mustgather: Read first for problems that are encountered with Debug Tool for z/OS: http://www.ibm.com/support/docview.wss?rs=615&context=SSGTSD&q1=mustgather&uid=swg21254711&loc=en_US&cs=utf-8&lang=en
- Mustgather: Read first for problems that are encountered with Fault Analyzer for z/OS:http://www.ibm.com/support/docview.wss?rs=273&context=SSXJAJ&q1=mustgather&uid=swg21255056&loc=en_US&cs=utf-8&lang=en
- Mustgather: Read first for problems that are encountered with File Manager for z/OS: http://www.ibm.com/support/docview.wss?rs=274&context=SSXJAV&q1=mustgather&uid=swg21255514&loc=en_US&cs=utf-8&lang=en
- Mustgather: Read first for problems that are encountered with Enterprise COBOL for z/OS: http://www.ibm.com/support/docview.wss?rs=2231&context=SS6SG3&q1=mustgather&uid=swg21249990&loc=en_US&cs=utf-8&lang=en
- Mustgather: Read first for problems that are encountered with Enterprise PL/I for z/OS: http://www.ibm.com/support/docview.wss?rs=619&context=SSY2V3&q1=mustgather&uid=swg21260496&loc=en_US&cs=utf-8&lang=en

If the product does not have a Mustgather document, provide answers to the following questions:

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

Submitting problems

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

Online
Click **Open service request** on the IBM Software Support site at http://www.ibm.com/software/support/probsub.html. In the Other support tools section, select IBMLink to open an Electronic Technical Response (ETR). Enter your information into the appropriate problem submission form.

By phone
Call 1-800-IBMSERV (1-800-426-7378) in the United States, or from other countries go to the contacts page of the *IBM Software Support Handbook* at http://techsupport.services.ibm.com/guides/contacts.html and click the name of your geographic region.

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

After a Problem Management Record (PMR) is open, you can submit diagnostic MustGather data to IBM by using one of the following methods:

- FTP diagnostic data to IBM
- If FTP is not possible, email diagnostic data to techsupport@mainz.ibm.com. You must add PMR xxxxx bbb ccc in the subject line of your email. xxxxx is your PMR number, bbb is your branch office, and ccc is your IBM country code. Click here http://itcenter.mainz.de.ibm.com/ecurep/mail/subject.html for more details.

Always update your PMR to indicate that data was sent. You can update your PMR online or by phone as described above.

Notices

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Trademarks

Bibliography

File Manager publications

File Manager Customization Guide, SC19-4118
File Manager User's Guide and Reference, SC19-4119
File Manager User's Guide and Reference for DB2 Data, SC19-4120
File Manager User's Guide and Reference for IMS Data, SC19-4121
File Manager User's Guide and Reference for CICS, SC19-4122
File Manager Fact Sheet, G325-2429
File Manager License Information, GC19-4117
File Manager Program Directory, GI10-8968

Related publications for Problem Determination Tools

IBM Problem Determination Tools for z/OS Common Component Customization Guide and User Guide, SC19-4159

Related publications for COBOL

IBM COBOL Language Reference, SC26-9046
IBM COBOL Programming Guide for OS/390 & VM, SC26-9049

Related publications for PL/I

IBM VisualAge PL/I Language Reference, SC26-9476
IBM VisualAge PL/I for OS/390 Programming Guide, SC26-9473

Related publications for z/OS

z/OS DFSMS Access Method Services for Catalogs, SC26-7394
z/OS DFSMS Object Access Method Application Programmer's Reference , SC35-0425
z/OS DFSMS: Using Data Sets, SC26-7410
z/OS DFSMS: Using Magnetic Tapes, SC26-7412
z/OS ISPF User's Guide Vol 1, SC34-4822
z/OS ISPF User's Guide Vol II, SC34-4823
z/OS MVS JCL Reference, SA22-7597
z/OS MVS System Messages, Vol 5, SA22-7635
z/OS Support for Unicode Using Conversion Services, SA22-7649
z/OS TSO/E Command Reference, SA22-7782
z/OS TSO/E Programming Services, SA22-7789
z/OS TSO/E REXX Reference, SA22-7790
z/OS TSO/E REXX User's Guide, SA22-7791

Index

Printed in USA

SC19-4159-05