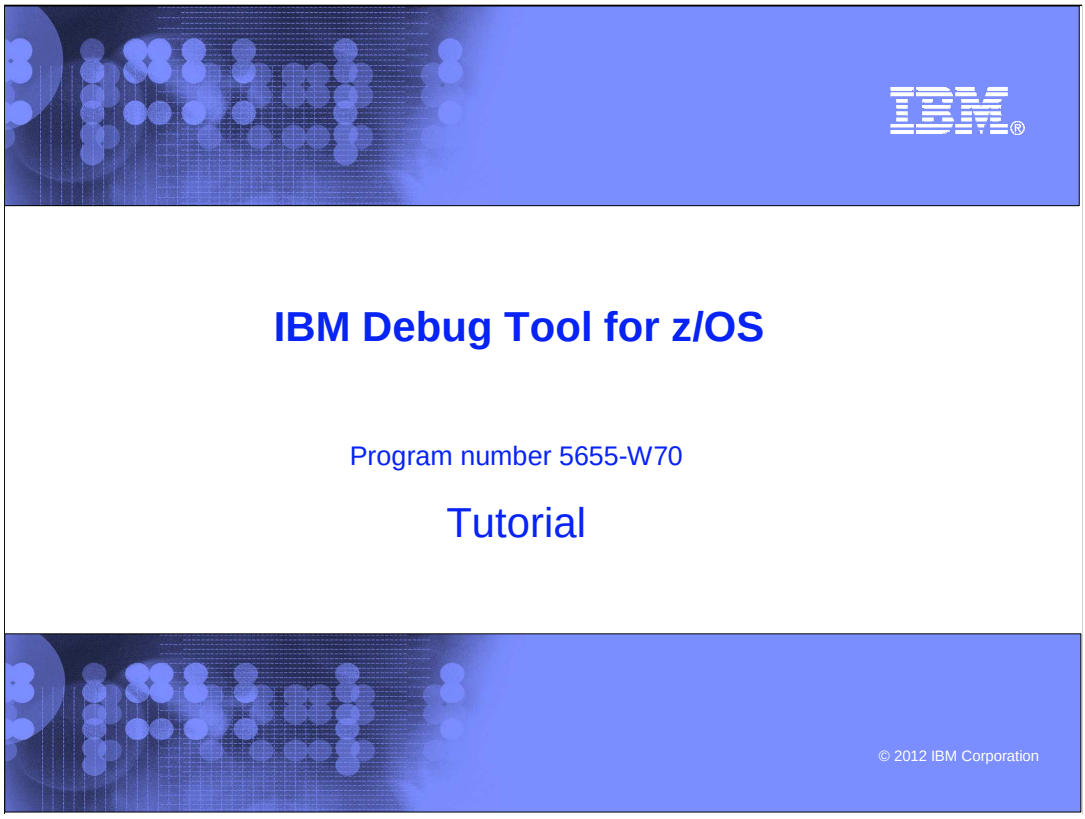
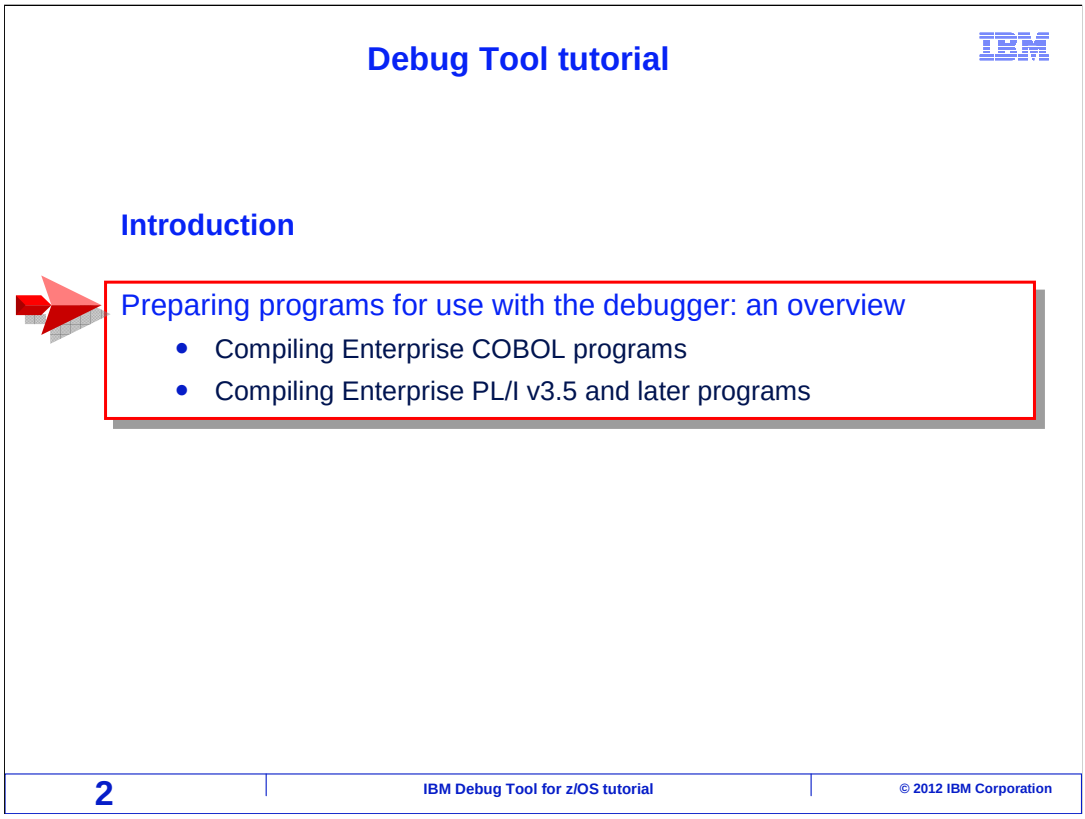


This is the tutorial for IBM Debug Tool for z/OS®, one of the IBM zSeries® problem determination tools.



In this section, you will see an overview of how programs compiled with the IBM Enterprise COBOL or Enterprise PL/I compilers can be prepared so they can be used with the IBM problem determination tools, including Debug Tool.

Compiling a program for use with the debugger

- Before you use a program with Debug Tool, it must be compiled with certain compiler options to:
  - prepare the module for debugging, and
  - create a program debug file
- Debug files are created when programs are compiled
  - The debugger reads these files to display program source and variables
- Programs can be compiled so that source support can be provided in each of the IBM problem determination products:
  - IBM Debug Tool for z/OS
  - IBM Fault Analyzer for z/OS
  - IBM Application Performance Analyzer for z/OS
- If there are standard program compile processes, they should be updated to enable source support for these products

Before you use a program with Debug Tool, you should compile it with certain compiler options that Debug Tool requires. 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 can be produced.

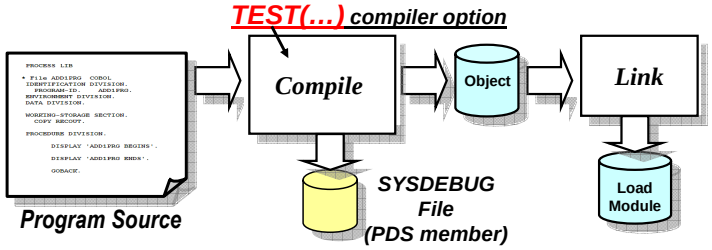
A source information, or debug, file is created when a program is compiled. It contains program statements and information about the statements and variables, such as offsets and lengths. The debugger uses this information to display the program and variables.

A program can be prepared so that source support is available in each of the IBM problem determination tools: Debug Tool, Fault Analyzer, and Application Performance Analyzer.

If your organization has standard program compile processes in place, those processes should be updated to enable source support for these products.

Compiling a program for use with IBM PD Tools, including IBM Debug Tool for z/OS

- When using the Enterprise COBOL compiler, for example, a TEST(…) compiler option with appropriate sub-options produces a sysdebug file



- A COBOL sysdebug file can be used by Debug Tool, Fault Analyzer, and Application Performance Analyzer
- Other compilers require different options, and generate different kinds of debug files
  - For details about compiling programs, see the Debug Tool for z/OS User's Guide: Appendix C. Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products

When you use either the Enterprise COBOL or Enterprise PL/I compiler, a TEST(…) compiler option with appropriate sub-options tells the compiler to produce a SYSDEBUG file.

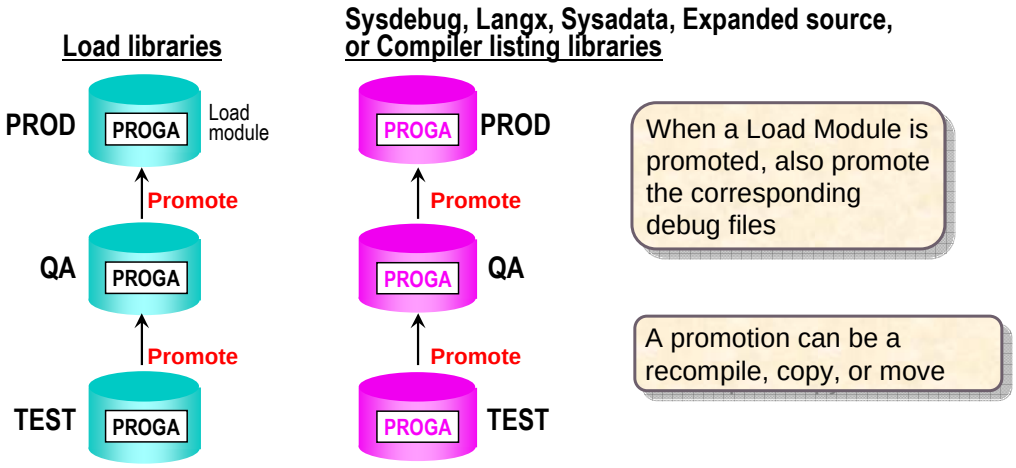
A sysdebug file produced by Enterprise COBOL can be used by Debug Tool, Fault Analyzer, and Application Performance Analyzer.

Other compilers require different compiler options, and generate different kinds of debug files.

For details about compiling programs, see the Debug Tool for z/OS User's Guide: Appendix C.

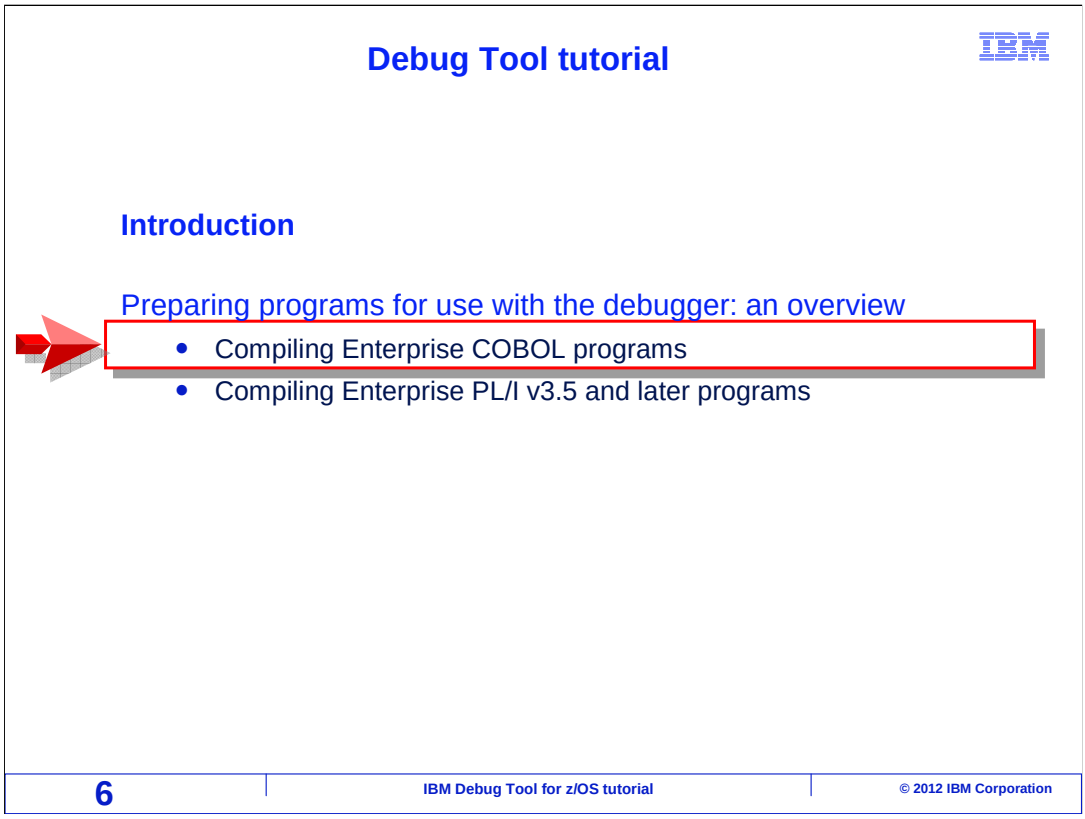
Promote debug files for IBM PD tools

- Do you want the ability to use Debug Tool, Fault Analyzer, or APA throughout a program's life cycle?
- Then the *promotion* processes should promote debug files files along with program load modules

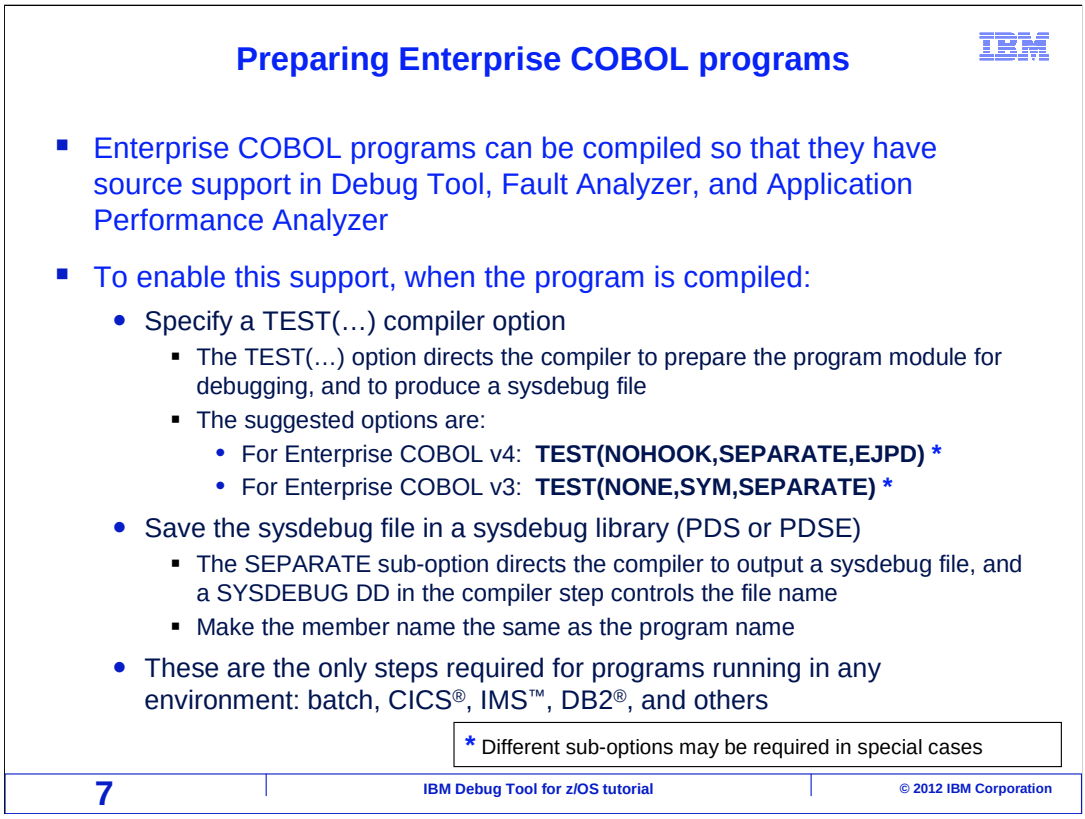


Typically, as a program is tested, program load modules are promoted through different stages and eventually reach production. For example, when a new program is compiled for the first time, it may be placed into a 'test' load library. After unit testing is completed, perhaps it is promoted to a 'quality assurance' environment. And finally it may be promoted into production. On your system, you may know these stages by different names. "Unit test", "System test", and "Model office" are common names for the various stages.

Consider whether you want the ability to use Debug Tool, Fault Analyzer, and Application Performance Analyzer throughout your programs' life cycle. Even if you do not plan to use Debug Tool with production programs, Fault Analyzer and Application Performance Analyzer are very useful in those stages. To enable these tools at each stage, update your promotion processes to retain the debug files. Promotions can be accomplished by performing a recompile, a copy, or a move. Give your debug files the same treatment as your load libraries. For each load library, you should have a corresponding set of debug libraries. When the load module is promoted, promote the debug files right along with it. That way, you can continue to take advantage of the tools at all stages of your program's life cycle.



Next, you will see a suggested method for compiling a program when the Enterprise COBOL compiler is used.



When you use the Enterprise COBOL compiler, programs can be prepared so that source support is enabled in Debug Tool, Fault Analyzer, and Application Performance Analyzer.

If your organization has standard compile processes in place, then those processes should be updated with these changes. Once the changes are made, you can use the standard processes to compile your programs.

However, if you use your own JCL to compile programs, or if it is your responsibility to update compile processes, here are the changes needed to enable source support when the program is compiled.

Specify a TEST compiler option. The TEST option directs the compiler to prepare the program module for debugging, and to produce a sysdebug file. For Enterprise COBOL version 4, the suggested TEST sub-options are NOHOOK, SEPARATE, and EJPD. For Enterprise COBOL version 3, the suggested TEST sub-options are NONE, SYM, and SEPARATE.

Save the generated sysdebug file in a PDS or PDSE library by adding a sysdebug DD statement to the compile JCL. Make the member name the same as the program name. These are the only changes needed to your compile process for programs running in any environment, such as batch, CICS, IMS, DB2, and others.

Considerations for preparing Enterprise COBOL programs for use with the debugger

- The sysdebug file is the only debug file needed, and it enables source support in:
  - IBM Debug Tool for z/OS
  - IBM Fault Analyzer for z/OS
  - IBM Application Performance Analyzer for z/OS
- Enterprise COBOL modules produced this way are production-ready
  - With the NOHOOK or NONE sub-options, the program does not incur significant added run-time overhead when running without the debugger

The sysdebug file is the only file needed to enable source support in Debug Tool, Fault Analyzer, and Application Performance Analyzer.

The resulting load module, while having full debugging support, is ready for production use. Because a NOHOOK or NONE sub-option of the TEST compiler option is specified, the program does not incur significant added run-time overhead when running without the debugger. There is typically no reason to recompile to remove debugging capability before promoting the module to production.

Considerations for preparing Enterprise COBOL programs for use with the debugger

- Using the OPT (optimize) compiler option
  - Optimized Enterprise COBOL programs are supported by the debugger
  - In Enterprise COBOL v3, the debugger commands 'JUMPTO' and 'GOTO' are disabled when debugging optimized programs
  - In Enterprise COBOL v4, this restriction is removed when using the EJPD sub-option of the TEST compiler option
- Using 'Debug Tool user exit data set' panels to trigger the debugger
  - Typically, this capability is enabled by adding a library to STEPLIB or JOBLIB in your application's run-time JCL
  - However, if a special Debug Tool LE exit module is linked into the load module of a main program, the 'Debug Tool user exit data set' facility can be used to trigger the debugger *without any JCL modifications*
  - Link module EQAD*****CXT into the load module during the linkage editor step (from the DT SEQAMOD library)
    - ***** = (“B” for batch programs, “I” for IMS/TM, “D” for DB2 stored procedures)

You can compile an Enterprise COBOL program with the optimize compiler option, and it is still supported by the debugger. When the version 3 compiler is used with the optimize option, all commands and features of the debugger are available, with the exception that JUMPTO and GOTO debugger commands cannot be used. When the version 4 compiler is used, it is suggested to also code the EJPD sub-option of the TEST compiler option. This removes the JUMPTO and GOTO restriction.

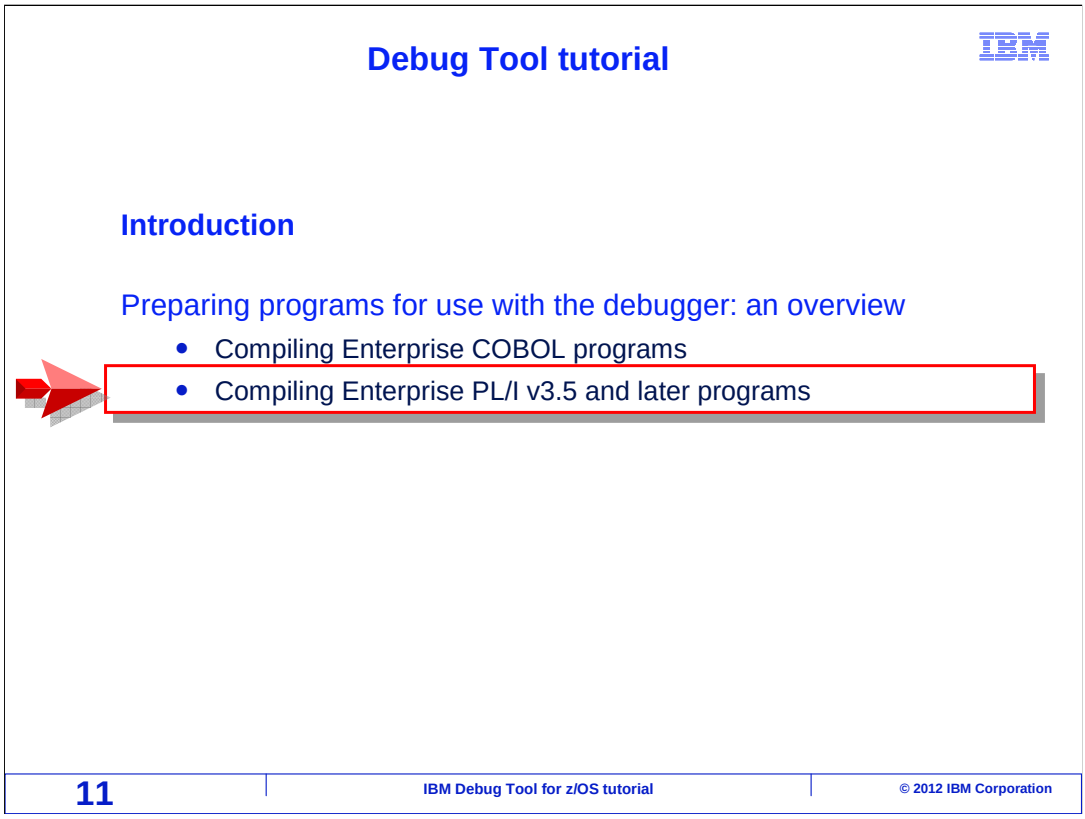
One final consideration at compile time is whether you plan to enable support for the 'Debug Tool user exit data set' facility to trigger the debugger. This facility will be described in detail in later sections. You may not plan to use this facility at all, and even if you do, it is typically enabled by adding a library to the STEPLIB or JOBLIB concatenation of your application's run-time JCL.

However, if you link a special Debug Tool Language Environment exit module into the load module of a main program, the 'Debug Tool user exit data set' facility can be used to trigger the debugger **without any JCL modifications** when your program runs. If this capability is wanted, then link a special Debug Tool module into the application load module.

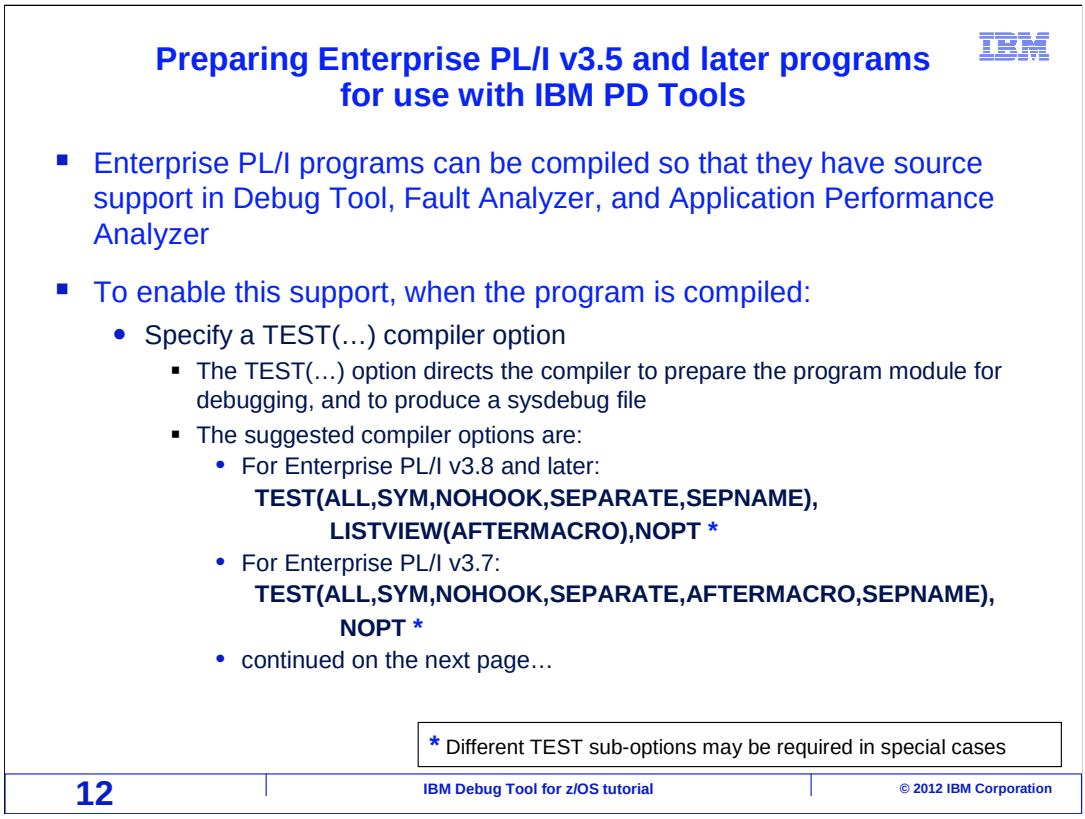
Where to find more details

- For details about compiling programs for use with IBM problem determination tools products, including JCL examples:
  - See the Debug Tool for z/OS User's Guide:
    - Appendix C. Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products

To get more in-depth information about compiling programs for use with Debug Tool, to see JCL examples, or to learn how to prepare program using other compilers, see the Debug Tool for z/OS User's Guide, Appendix C.



Next, you will see a suggested method for compiling a program when the Enterprise PL/I compiler is used.

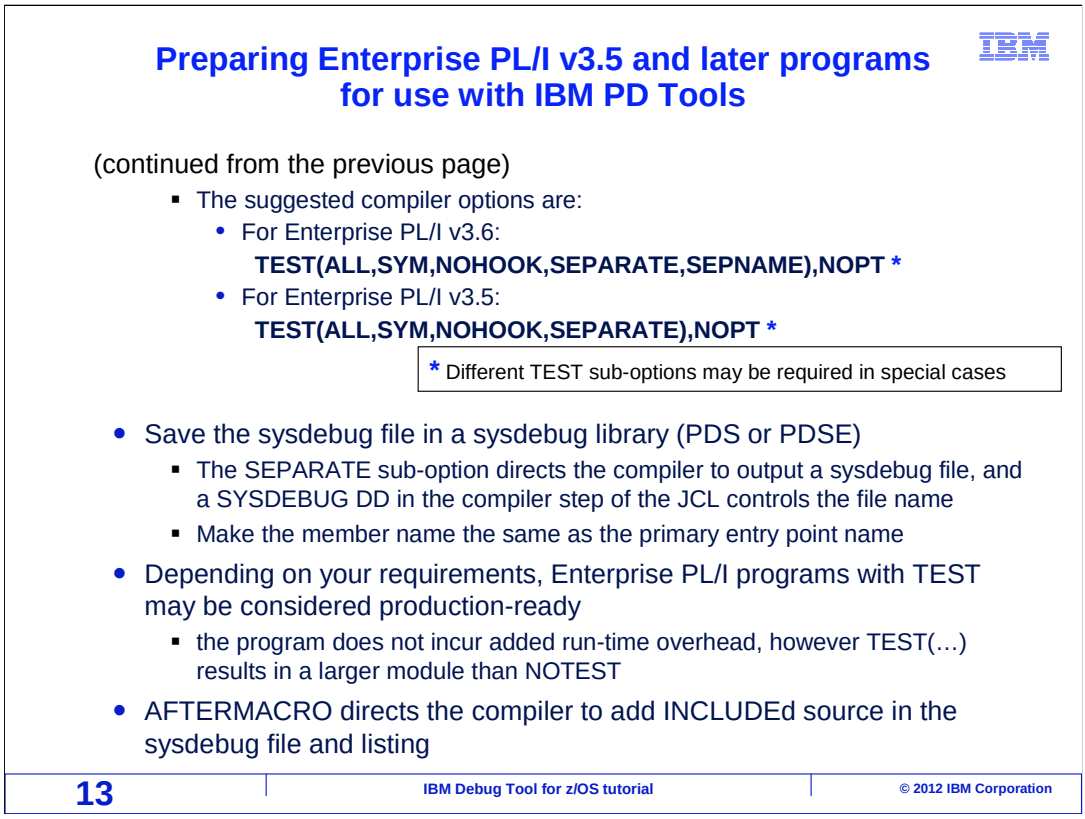


When you use the Enterprise PL/I compiler, programs can be prepared so that source support is enabled in Debug Tool, Fault Analyzer, and Application Performance Analyzer.

If your organization has standard compile processes in place, then those processes should be updated with these changes. Once the changes are made, you can use the standard processes to compile your programs.

However, if you use your own JCL to compile programs, or if it is your responsibility to update compile processes, here are the changes needed to enable source support when the program is compiled.

Specify the needed compiler options. The TEST option directs the compiler to prepare the program module for debugging, and to produce a sysdebug file. For Enterprise PL/I version 3.8 and later, specify the TEST option with sub-options of ALL, SYM, NOHOOK, SEPARATE, and SEPNAME. Also specify LISTVIEW(AFTERMACRO), and NOPT. For Enterprise PL/I version 3.7, specify the TEST option with sub-options ALL, SYM, NOHOOK, SEPARATE, AFTERMACRO, and SEPNAME. Also specify NOPT.



For Enterprise PL/I version 3.6, specify the TEST option with sub-options ALL, SYM, NOHOOK, SEPARATE, and SEPNAME. Also specify the NOPT option. For Enterprise PL/I version 3.5, specify the TEST option with sub-options ALL, SYM, NOHOOK, and SEPARATE, and the NOPT option.

Save the generated sysdebug file in a PDS or PDSE library by adding a sysdebug DD statement to the compile JCL. Make the member name the same as the primary entry point name.

Depending on your organization's policies, Enterprise PL/I programs compiled with the TEST compiler option may or may not be considered ready for use in a production environment. Because of the NOHOOK sub-option, the program does not incur significant additional run-time overhead when it runs without the debugger. However, the resulting module is larger than when compiled with NOTEST.

The AFTERMACRO sub-option directs the compiler to put expanded copybooks and macros into the sysdebug file and compiler listing. That way, the fully expanded source is shown to the programmer by the debugger during a debugging session.

Preparing Enterprise PL/I v3.5 and later programs for use with IBM PD Tools

For Enterprise PL/I version 3.6 and earlier, a two-stage compile is suggested to expand macros and INCLUDEs

- The first stage outputs a source file with expanded macros and INCLUDEs
- The second stage is the actual compile. The expanded source file from the first stage is input.
- A two-stage compile is not needed for Enterprise PL/I v3.7 and later when the AFTERMACRO sub-option is used

For Enterprise PL/I before version 3.7, a two-stage compile is suggested to expand macros and copy books. The first stage only expands macros and INCLUDEs, and outputs a source file with the full expansion. The second stage is the actual compile. It reads the expanded source file created by the first stage. A two-stage compile is not needed for Enterprise PL/I version 3.7 and later when the AFTERMACRO sub-option is specified.

Considerations for preparing Enterprise PL/I programs for use IBM PD Tools

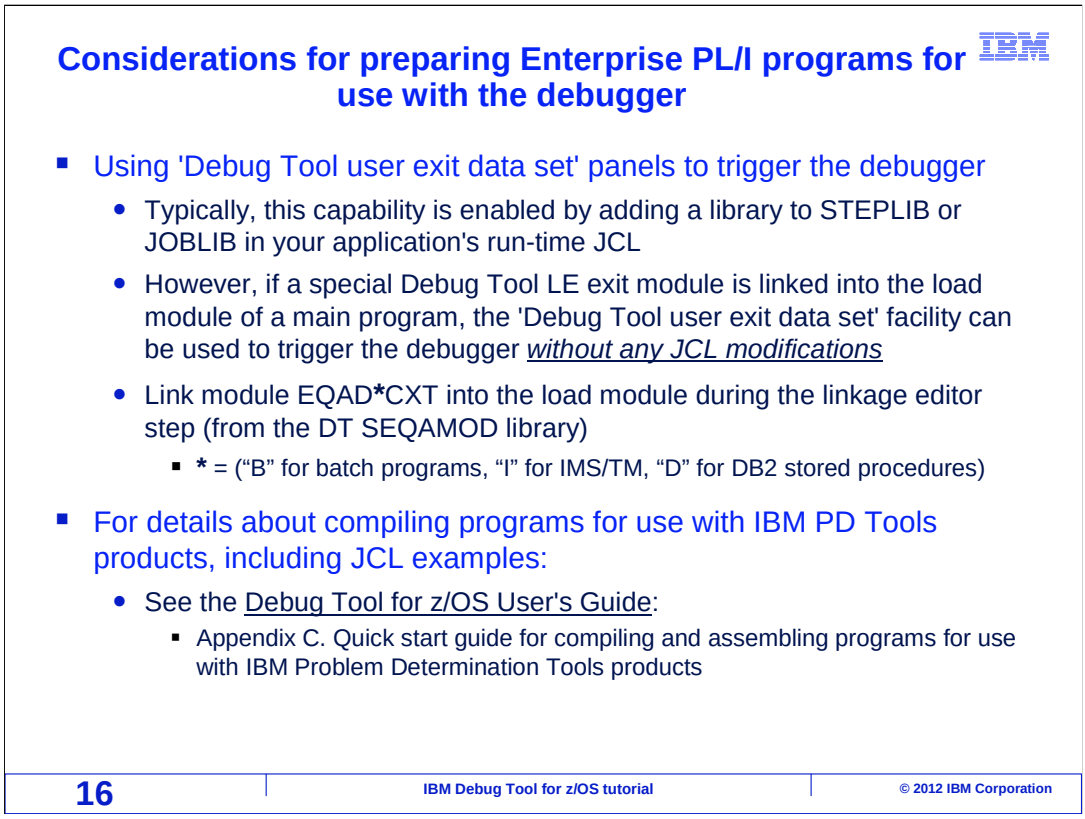
- Specify the NOPT (no optimize) compiler option when the TEST option is specified
  - Do not use the OPT (optimize) compiler option with Debug Tool
- The sysdebug file can be used to enable source support in:
  - IBM Debug Tool for z/OS
  - IBM Fault Analyzer for z/OS
- If the sysdebug file will not be made available in production, consider creating a Langx file for use in production by IBM Fault Analyzer for z/OS and IBM Application Performance Analyzer for z/OS
  - Add a step after the compile to run Debug Tool utility program EQALANGX
  - It reads the compiler listing and produces a Langx file
  - Save the Langx file in a Langx library (PDS or PDSE), making the member name the same as the primary entry point name

Specify the NOPT option when compiling for use with the debugger.

The sysdebug file is the only file needed to enable source support in Debug Tool and Fault Analyzer.

Depending on the policies in your organization, Enterprise PL/I modules compiled with a TEST compiler option may or may not be considered to be production ready. If your organization decides to use programs compiled with TEST in production, then the sysdebug file can be promoted along with the module and made available in production.

However, if your organization opts to recompile programs without the TEST option before promoting them to production, then sysdebug files will not be available for your production programs. If that is the case, consider adding a step to the compile process to run Debug Tool utility program EQALANGX. This program reads the compiler listing and creates a special Langx file. Langx files can then be used by Fault Analyzer and Application Performance Analyzer to provide source support for your production programs. Save the Langx file in a PDS or PDSE data set, making the member name the same as the primary entry point name.



One final consideration at compile time is whether you plan to enable support for the 'Debug Tool user exit data set' facility to trigger the debugger, which will be described in detail in later sections. You may not plan to use this facility at all, and even if you do, it is typically enabled by adding a library to the STEPLIB or JOBLIB concatenation of your application's run-time JCL.

However, if you link a special Debug Tool Language Environment exit module into the load module of a main program, the 'Debug Tool user exit data set' facility can be used to trigger the debugger **without any JCL modifications** when your program runs. If this capability is wanted, then link a special Debug Tool module into the application load module.

To get more in-depth information about compiling programs for use with Debug Tool, to see JCL examples, or to learn how to prepare program using other compilers, see the Debug Tool for z/OS User's Guide, Appendix C.

That is the end of this section, an overview of compiling Enterprise COBOL and Enterprise PL/I programs for use with the debugger.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_DTV12s02PreparingProgramsForPDTools.ppt

This module is also available in PDF format at: ../DTV12s02PreparingProgramsForPDTools.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, CICS, DB2, IMS, z/OS, and zSeries are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.