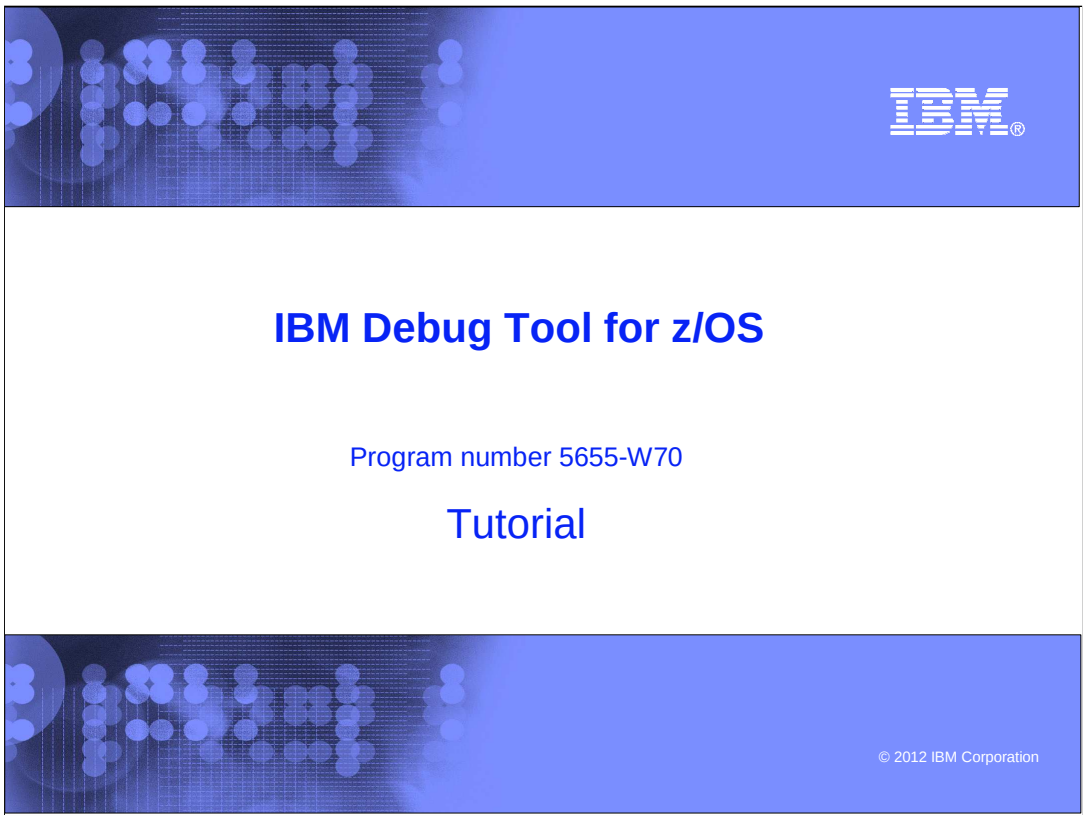
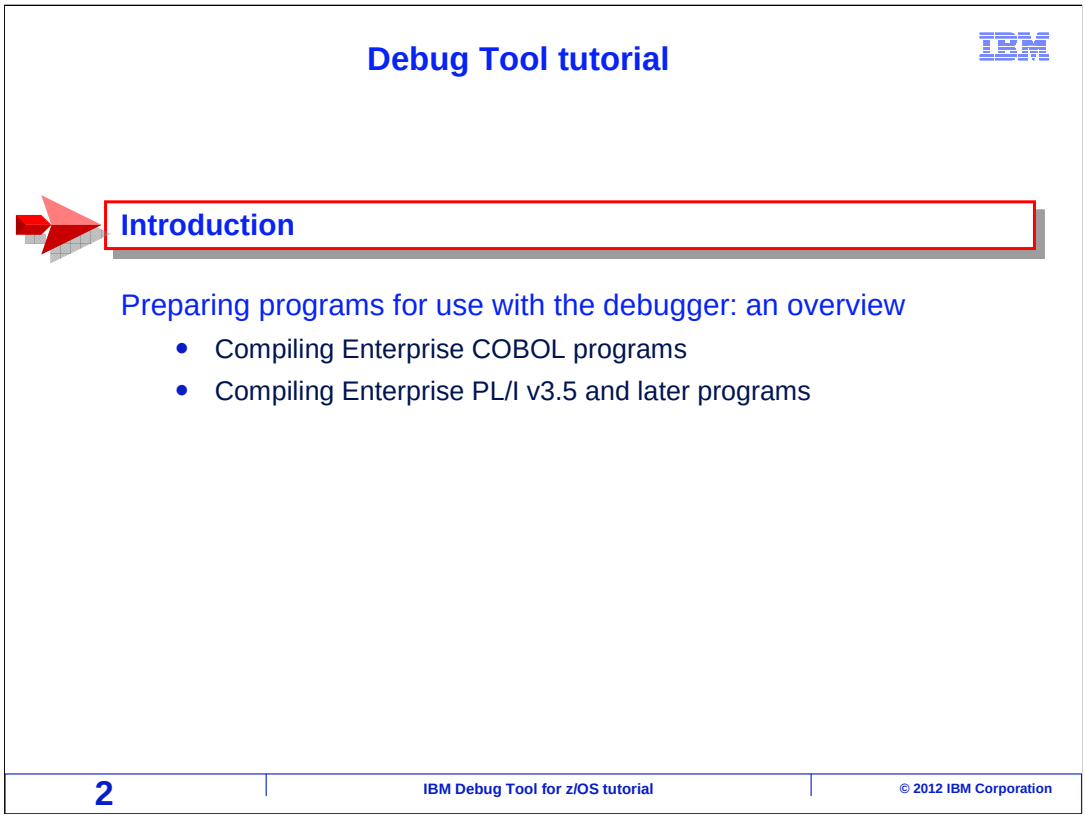


This is the tutorial for IBM Debug Tool for z/OS®, one of the IBM zSeries® problem determination tools.

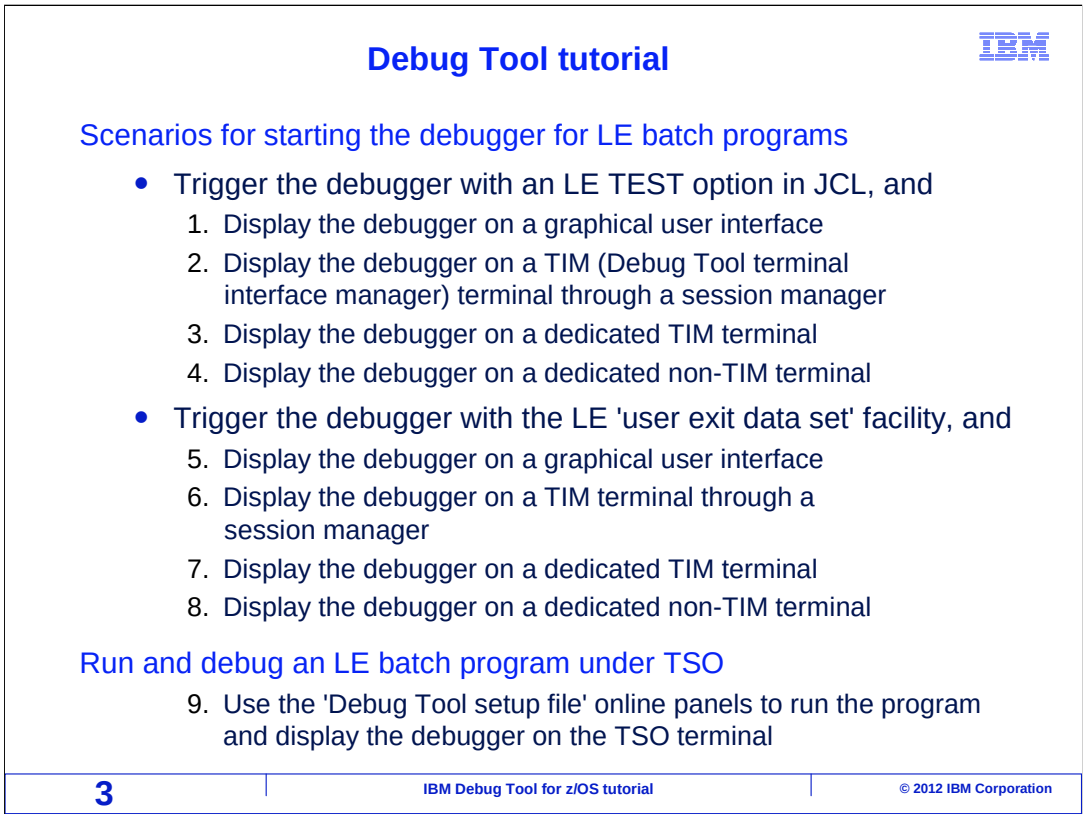


Take these tutorials to learn how to use Debug Tool to work with programs running on z/OS systems. Debug Tool is designed to address the needs of application developers working with complex programs and environments.

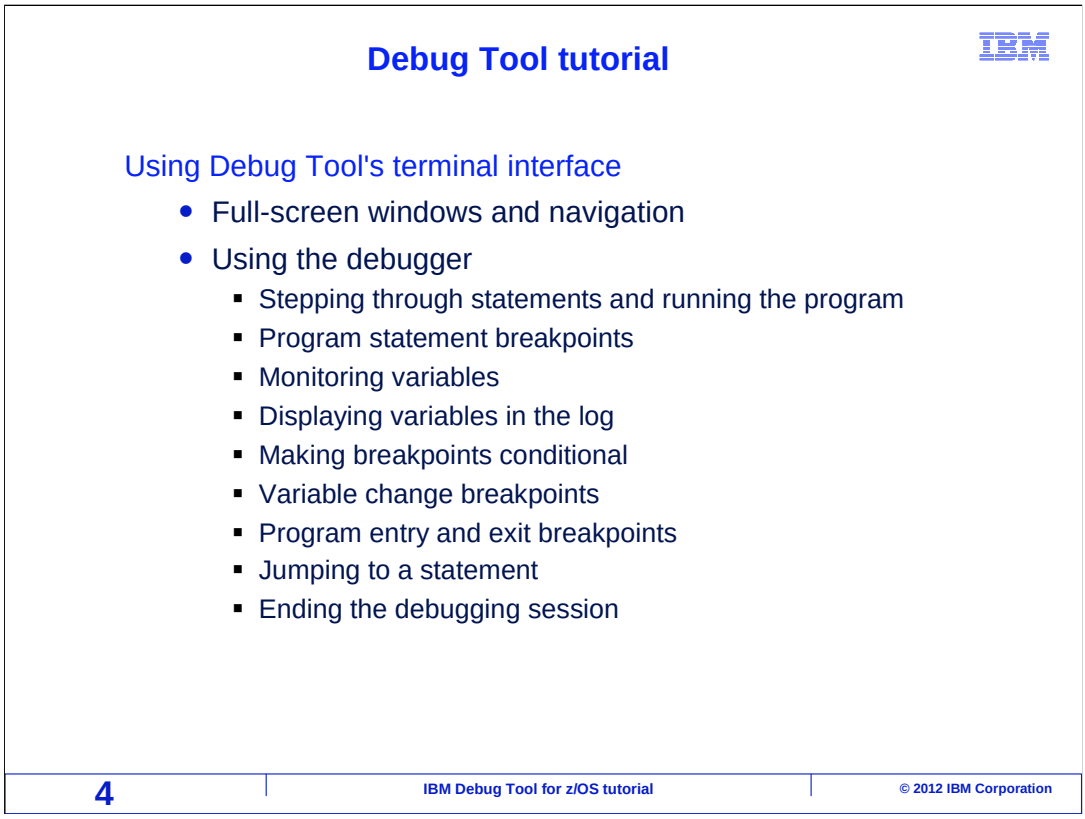
Take a look at how the course is structured. It is broken down into sections. If you have not already used Debug Tool, take each section in order. But if you have already been using it and want to learn more about a particular topic, you can go directly to that section.

This is the introduction, and it will describe how you can use Debug Tool and how to you can get the manuals from the IBM website.

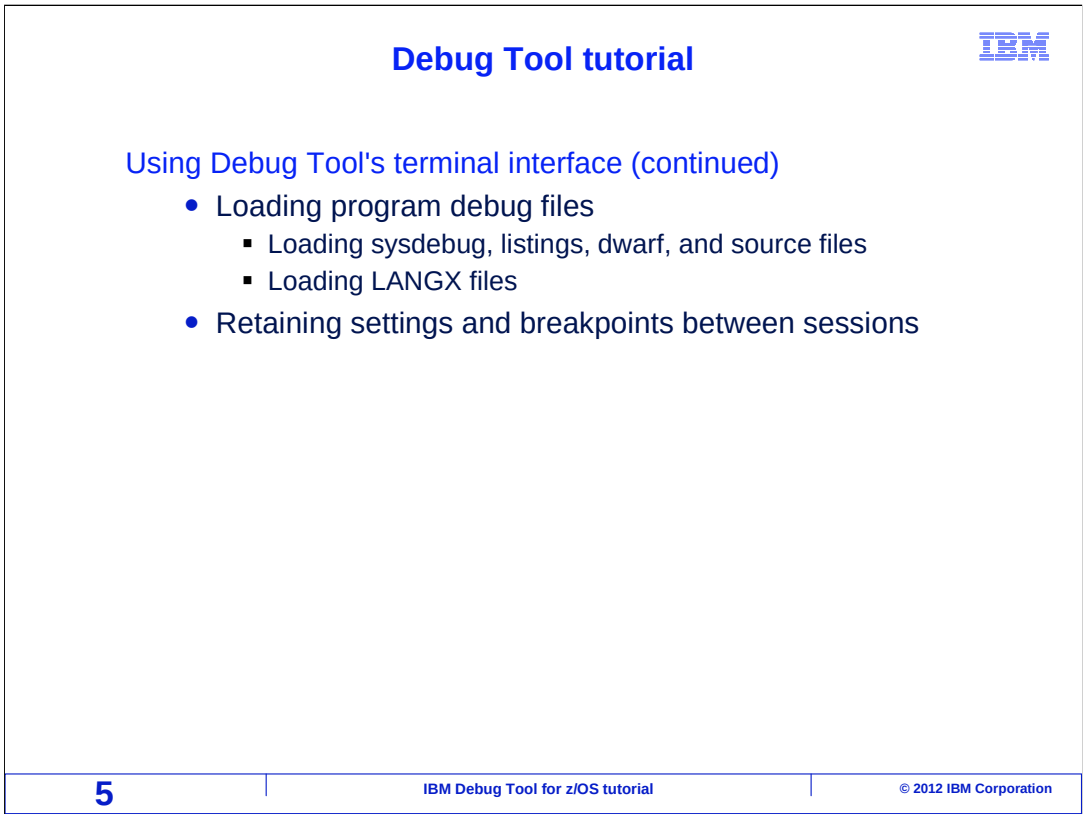
The next section, “Preparing programs for use with the debugger” provides an overview of how to compile programs so you can debug them, and shows where you can get detailed information about using the various compilers and assembler.



The section titled "Starting the debugger for batch LE programs" describes the various methods you can choose from to debug a batch program, and helps you select a method. After selecting a method, then view the section or sections that describe the specific methods you have chosen to debug batch programs.



The next sections describe how to use the 3270 terminal interface. Commands and features that you need to know to debug programs are described, along with hints, tips, and examples.



The next sections describe how to use the 3270 terminal interface. Commands and features that you need to know to debug programs are described, along with hints, tips, and examples.

Debug Tool tutorial

Using Debug Tool's graphical user interface

- Debug perspective views and navigation
- Using the debugger
  - Stepping through statements and running the program
  - Program statement breakpoints
  - Monitoring variables
  - Making breakpoints conditional
  - Watch breakpoints
  - Program entry and exit breakpoints
  - Ending the debugging session
- Loading program debug files
  - Loading sysdebug, listings, dwarf, and source files
  - Loading LANGX files

The next set of sections describe how to use the graphical user interface, showing buttons, commands, and features that you will use to debug programs.

Debug Tool tutorial

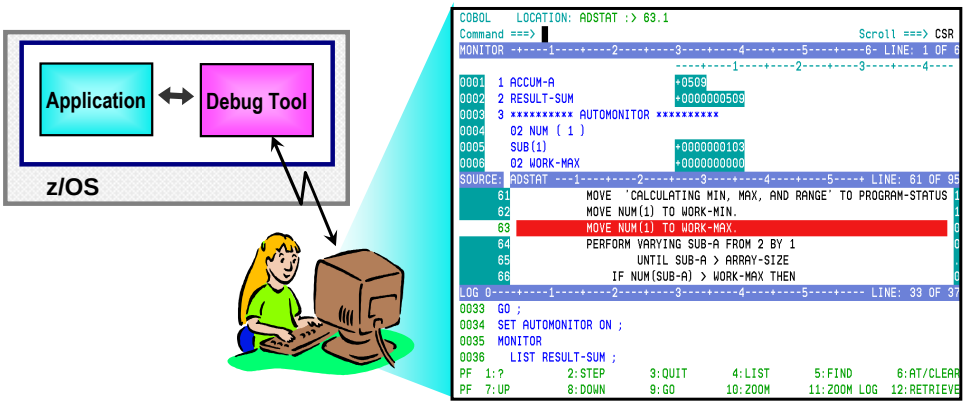
Debugging CICS® Applications

- Starting the debugger using the DTCN transaction
  - Creating a DTCN profile
  - Working with DTCN profiles
  - Setting the current terminal ID in a DTCN profile
- Creating and editing a DTCN profile using the GUI plug-in
- Starting the debugger using the CADP transaction
  - Creating a CADP profile
  - Working with CADP profiles
  - Setting the current terminal ID in a CADP profile

If you have programs that run under CICS, take the tutorial sections that describe debugging CICS applications.

What is IBM Debug Tool for z/OS ?

- An interactive program debugger you can use to control and monitor application programs while they run

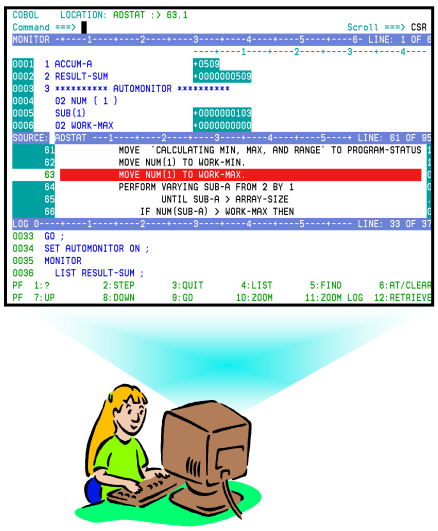


- And a set of related online panels and batch utilities

IBM Debug Tool for z/OS is an interactive program debugger you can use to control and monitor application programs while they run. It also provides a set of related online and batch utilities.

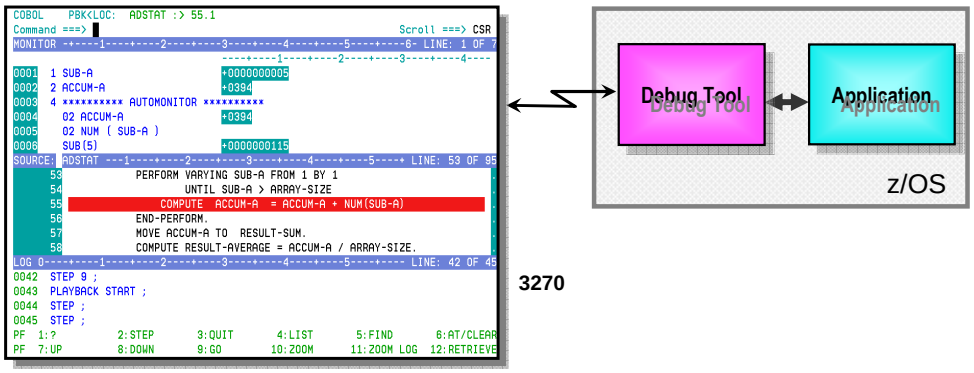
What features are provided by the interactive debugger?

- Use the debugger to:
  - Determine why a program is producing incorrect results
  - Locate an abending statement
  - See program statements, and step through statements one at a time
  - Set breakpoints at statements, programs, and conditions
  - Run until a breakpoint is reached
  - View and change the values of variables and storage
  - And much more



You can use the debugger to determine why a program is producing incorrect results, or to run a program that abends and have it stop automatically on the abending statement. As you are debugging, you can see program statements, and step through statements one at a time. You can set breakpoints at statements, at programs, and based on conditions, and then run the program until it reaches a breakpoint. You can view and change the values of variables in storage, trace the statement path of a program, and much more.

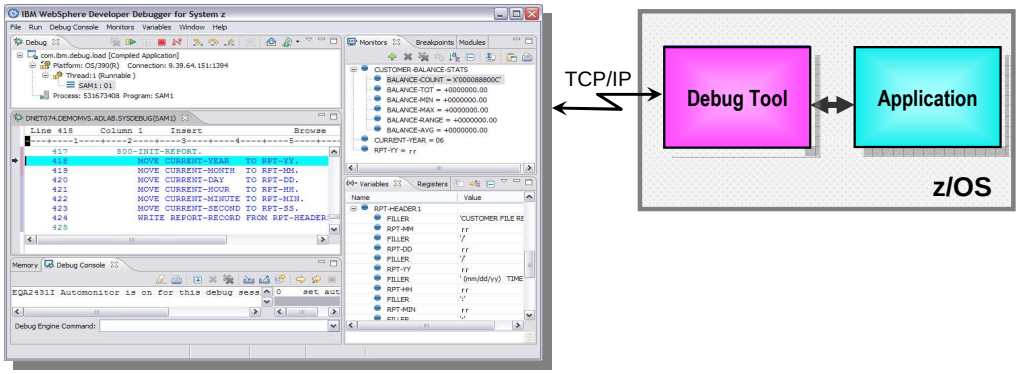
Interface with Debug Tool using a terminal



- Full-screen interface
  - A 3270 terminal-based interface is built into Debug Tool
  - Debug Tool displays on a 3270 terminal emulator on your workstation

Debug tool has more than one user interface. There is a 3270-based interface built in. The debugging engine controls the application as it runs, and communicates with a VTAM® terminal or 3270 emulator. This interface is called the MFI, which stands for mainframe interface. It is also called the full screen interface.

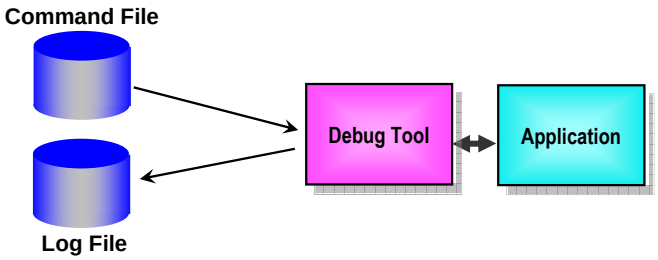
Interface with Debug Tool using a graphical user interface



- Remote graphical interfaces are available in Rational® Developer for System z® and CICS Explorer®
  - It is software that can be installed on your workstation
  - The application running on the mainframe is controlled by the same debugging engine that is used by the 3270 interface
  - Debug Tool displays on your workstation

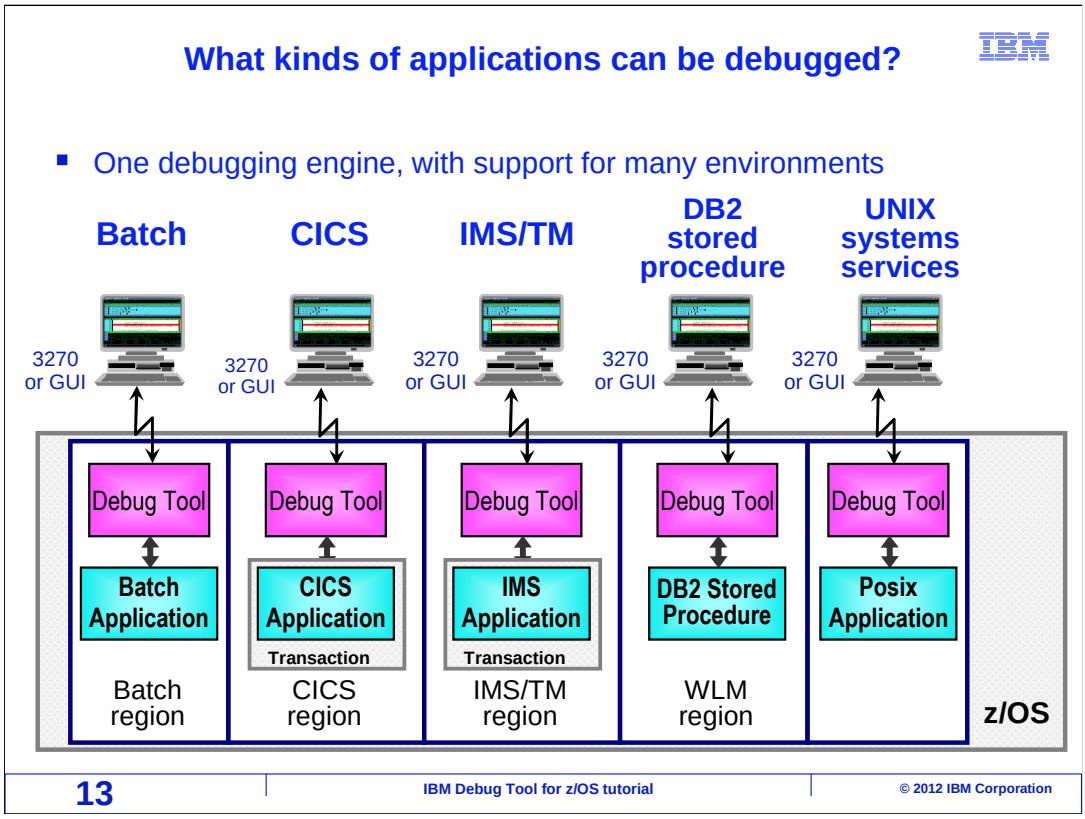
There are also remote graphical user interfaces for Debug Tool. The same debugging engine is used to control the application running on the mainframe, and it communicates with software installed on your workstation. You can install Rational Developer for System Z to provide a front end to Debug Tool. The GUI is also available as a plug-in to CICS explorer. You may prefer to use one of these interfaces instead of a 3270 terminal.

Automate debugging with your own script

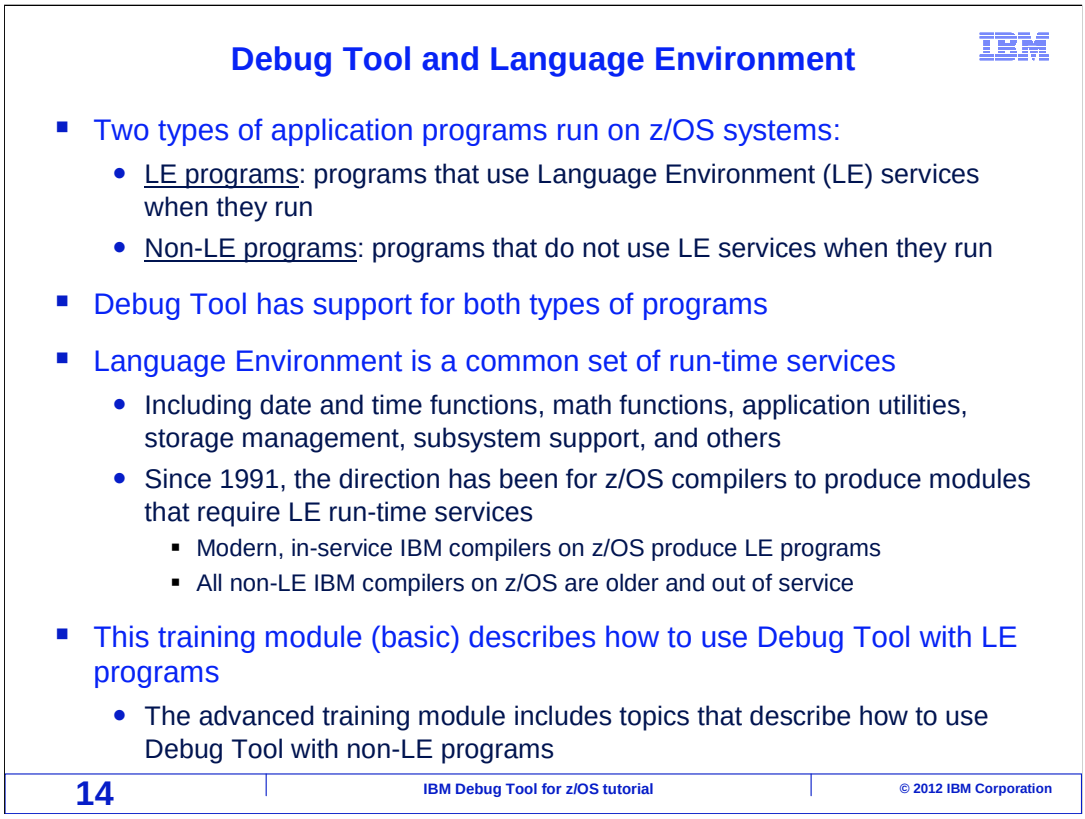


- "Batch mode" debugging
  - Can be unattended – everything can be scripted
  - A **command file** is used to run a series of debugger commands
  - The script controls the programs, and messages are written to a **log file**
  - Consider using it, for example:
    - instead of COBOL 'DISPLAY' statements
    - to display messages to help automate regression testing
  - If the script does not run the application to completion, the debugging interface displays after the script finishes

And there is one more way to interface with Debug Tool: do not interface with it at all. You can use something called batch mode debugging, which means that you do not work interactively with the debugger. Instead, you code a debugging script that controls the application. You write a command file, which contains a series of Debug Tool commands. Debug Tool uses your command file to control the application. During the run, your script writes messages to a log file. You might be wondering when you might use batch mode debugging instead of just interacting with the debugger from a terminal or workstation. And the answer is that typically you do not. However, you might consider using batch mode debugging instead of coding display statements into your COBOL programs, or to get a trace of statements as the program runs. This method can also be used to automate your regression testing runs, by checking for certain conditions at certain places in your programs, and generating messages based on the results. Batch mode debugging gives you a powerful way to programmatically get information about your application while it runs without changing the actual programs.



Debug tool provides a single debugging engine that supports many different environments. You can use the same debugger for your batch, CICS, and online IMS™ applications. It is the same debugging engine when you work with DB2® stored procedures and programs running with UNIX® systems services on z/OS. As a result, there is only a single debugger to learn. You do not need different debuggers in different application environments.

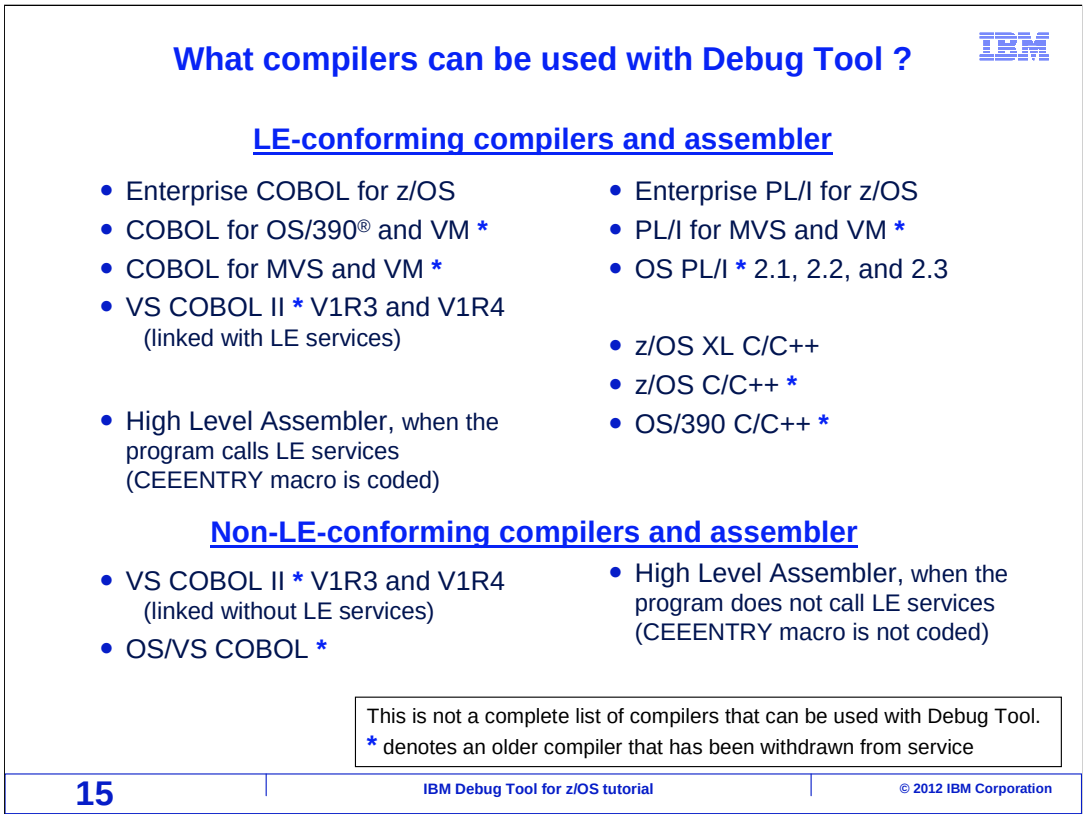


On a z/OS mainframe system, there are two types of programs, programs that use Language Environment services when they run, and programs that do not. IBM Debug Tool for z/OS supports both types of programs.

Language Environment is a common set of run-time services that a program accesses as it runs. These are services such as date and time functions, math functions, application utilities, storage management, support for subsystems such as CICS and IMS, and many others. Since the early 1990's, the direction for compilers on the z/OS platform has been that compilers produce modules that require Language Environment run-time services. This design allows for much smaller and more efficient application modules, since service routines do not have to be copied into program load modules, and service routines are accessed only when they are needed in real time.

Modern IBM z/OS compilers produce programs that require Language Environment services to be available at run-time. z/OS compilers that do not produce modules that use Language Environment services are old and no longer in service.

These tutorials describe how to use the debugger with programs that use Language Environment. The tutorials do not cover debugging older programs that do not use Language Environment, although those topics can be learned from product manuals or advanced on-site training classes.

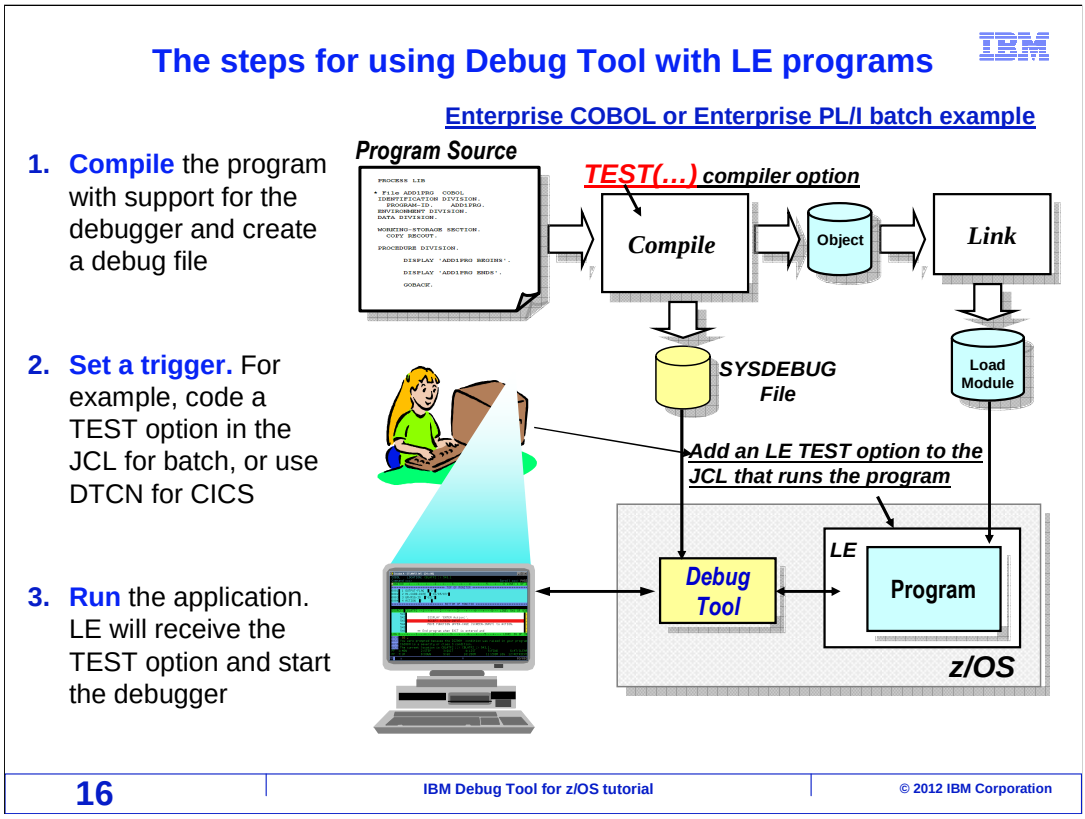


Here is a partial list of popular IBM compilers for the z/OS platform that can be used with IBM Debug Tool for z/OS. For COBOL, Enterprise COBOL for z/OS is the latest compiler. Other compilers are older, and if your organization still uses older COBOL compilers, there could be benefit in migrating to the latest compiler. Some compilers are no longer in support, such as the obsolete VS COBOL II and OS/VS COBOL compilers, but can still be used with Debug Tool.

For PL/I, Enterprise PL/I is the latest, but many older versions of PL/I can still be used with Debug Tool.

The debugger supports programs compiled using the latest z/OS XL C and C++ compiler, and some of the older C and C++ compilers. Debug Tool also provides robust support for programs written in assembler language.

The steps for using Debug Tool with LE programs



Here are the steps for starting Debug Tool with a Language Environment-conforming program. First, build the module. Compile and link a load module with support for the debugger. For modern compilers such as Enterprise COBOL and Enterprise PL/I, the compiler will produce a special file containing debugging information, such as a Sysdebug file.

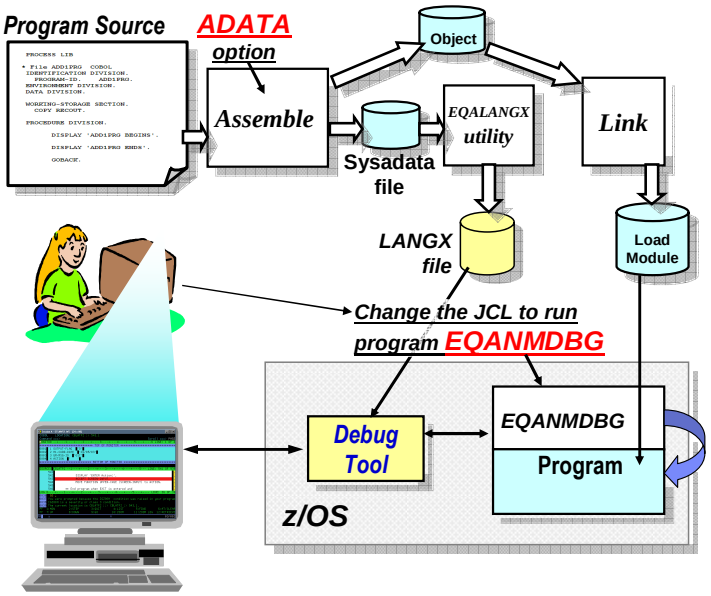
Next, set a debugging trigger. A debugging trigger is a setting or code that will cause the debugger to start when you run the program in a certain environment. Set it up so that at run time, a special TEST option will be passed to Language Environment when the program starts. There a many different ways to pass the TEST option. For example, it can be done for a batch program with a simple change to JCL. For CICS programs, a special transaction called DTCN can be used to turn on the TEST option for a specific program or transaction. Setting triggers for Language Environment conforming programs will be covered in detail in later sections.

Finally, run the application. When Language Environment receives the TEST run-time option, it starts the debugger. The debugger uses the special debug file created by the compiler to get information about the program, such as source statements and variable locations. The debugger is displayed on a 3270 terminal or graphical user interface.

The steps for using Debug Tool with non-LE programs

(non-LE batch Assembler example)

1. **Build:** Assemble or compile to create a load module and a debug file
2. **Set a trigger**: Set it up so Debug Tool will start when the program runs
3. **Run**: Run the application, and Debug Tool will start



17 | IBM Debug Tool for z/OS tutorial | © 2012 IBM Corporation

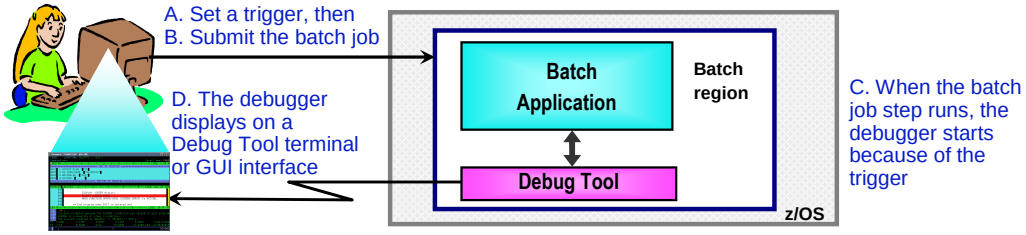
Here are the steps for starting Debug Tool with a program that does not use Language Environment services. Again, these are typically assembler programs or programs compiled with obsolete compilers. First, compile the program and link it into a load module. A special utility program that comes with Debug Tool is used to create a special LANGX file that contains information that the debugger will use.

Next, set a debugging trigger. The methods available for setting a trigger for programs that do not use Language Environment services will not be covered in this tutorial, with the exception of CICS programs. However, starting the debugger for all types of programs is described in the Debug Tool manuals, and can be covered in formal workshops.

Finally, run the application. The trigger causes the debugger to start, and it uses the LANGX file to get information about the program, such as source statements and variable locations. The debugger is displayed on a 3270 terminal or graphical user interface.

Debug batch LE applications

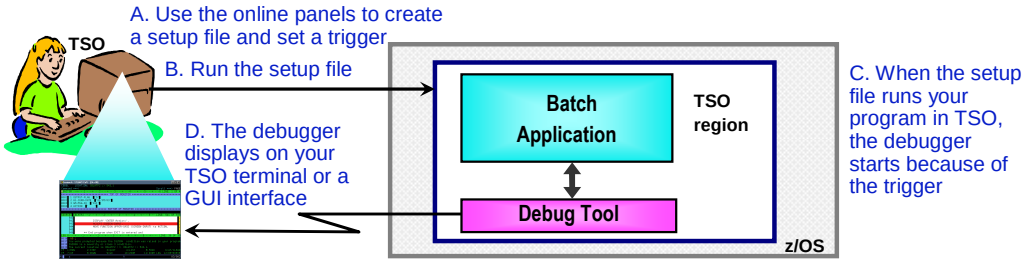
- Debug batch programs while they run in batch jobs
  - Including DB2 and IMS batch
- This workshop module describes two simple methods to trigger the debugger when an LE program runs in batch
  - Code an LE 'TEST' option in the run-time JCL, or
  - Use the Debug Tool 'User exit data set' online panels in TSO
- The advanced training module describes how to trigger the debugger for non-LE batch programs



You can use Debug Tool to work with batch programs, including programs that access DB2 and IMS databases. To start the debugger for a batch application, you set a program trap so Debug Tool will start automatically when the program runs. There are several ways to set a trap. For example, if the system is set up for it, you can specify the name of the batch program on a special Debug Tool panel in TSO. Then just submit the job, and Debug Tool starts automatically. Another way to set a trap is to code a special test option in your JCL, and then when you run the job, the debugger starts. Remember that the debugger can be displayed either on a 3270 terminal or on GUI debugging software on your workstation. When you set your debugging trap, you specify which interface you want to use. This is the first, and generally best, method available to debug batch applications. Run them in their native environment, as a batch job. However, there is another way to debug batch applications.

Debug batch applications in TSO

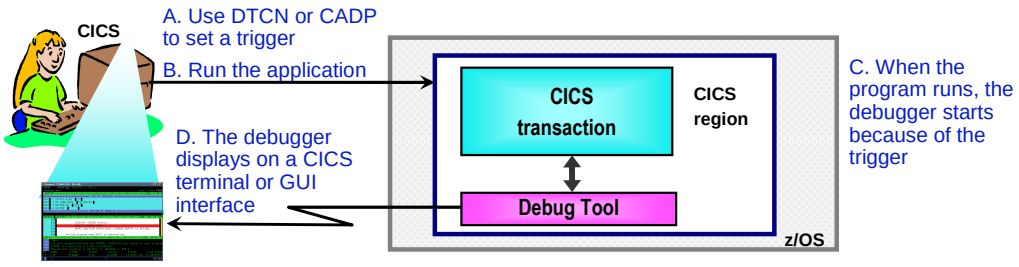
- You can run a batch application in TSO, instead of running it in batch, with the 'Debug Tool setup file utility' online panels
  - This is another, optional way to debug batch programs
- Use the online panels to create and run a "setup file"
  - A setup file serves a similar function as JCL, but runs an application in TSO instead of in batch
    - It contains the name of the program to execute, parameters, and a list of files (similar to DD statements)
  - Existing JCL can be copied to create a setup file



You can optionally debug your batch programs under TSO. Debug Tool gives you a special set of ISPF panels called the Debug Tool setup file utility that you can use to copy information from batch JCL and run your program. The debugger will be displayed on your TSO terminal or on GUI debugging software on your workstation.

Debug CICS applications

- Use the Debug Tool DTCN or CADP transaction to set a trigger
  - Define the trigger attributes that identify your application using a combination of: program name, transaction ID, terminal ID, user ID, and others
  - Specify where the debugger will be displayed
    - Single terminal mode: Debug Tool and the application share one terminal
    - Dual terminal mode: Debug Tool displays on a different CICS terminal (that you specify) than the application
    - GUI interface: Debug Tool displays on a GUI interface on your workstation



Use the same debugging engine to work with your CICS applications. To start the debugger for a CICS program, you will use a special Debug Tool transaction. Either DTCN or CADP. The Debug Tool transaction displays a screen where you specify information about the application that you want to debug. You can specify information such as the program name, the transaction ID, the CICS terminal ID, and the user ID. Then you save your debugging profile, and the trap is set. Then, just run your application normally in CICS. When it runs, Debug Tool is displayed on your CICS terminal or a remote GUI debugger.

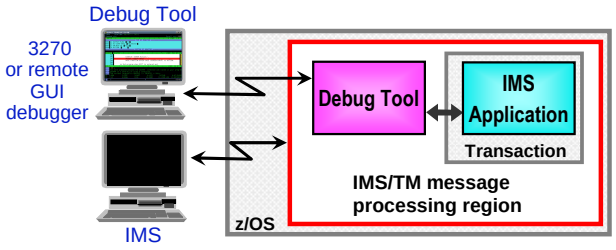
It is typical to debug CICS applications in something called single terminal mode. That means that Debug Tool is displayed on the same terminal where your transaction is running. The debugger and your application share the terminal. However, you can also in debug in dual terminal mode. That means your application is running on a CICS terminal, but the debugger is displayed on a different CICS terminal or on a remote GUI debugger. One of the benefits of dual terminal debugging is the ability for one person to run a CICS transaction, and another person to debug it. Also be aware that you can debug background CICS transactions, that is transactions that are not attached to a CICS terminal. For example, transactions that are initiated from web-based remote systems may run as background transactions.

Debug IMS/TM transaction applications

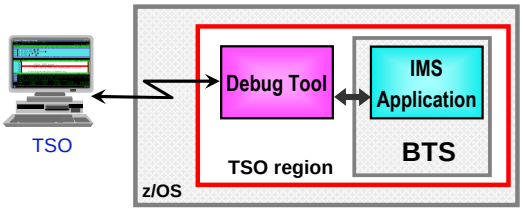
IMS Online Applications (IMS/TM)

- Set a program trap by specifying the program name on a Debug Tool panel in TSO

Dual Terminal mode:
Run the application as it normally runs in an online IMS region.
Debug Tool is displayed on a Debug Tool terminal or a GUI interface.



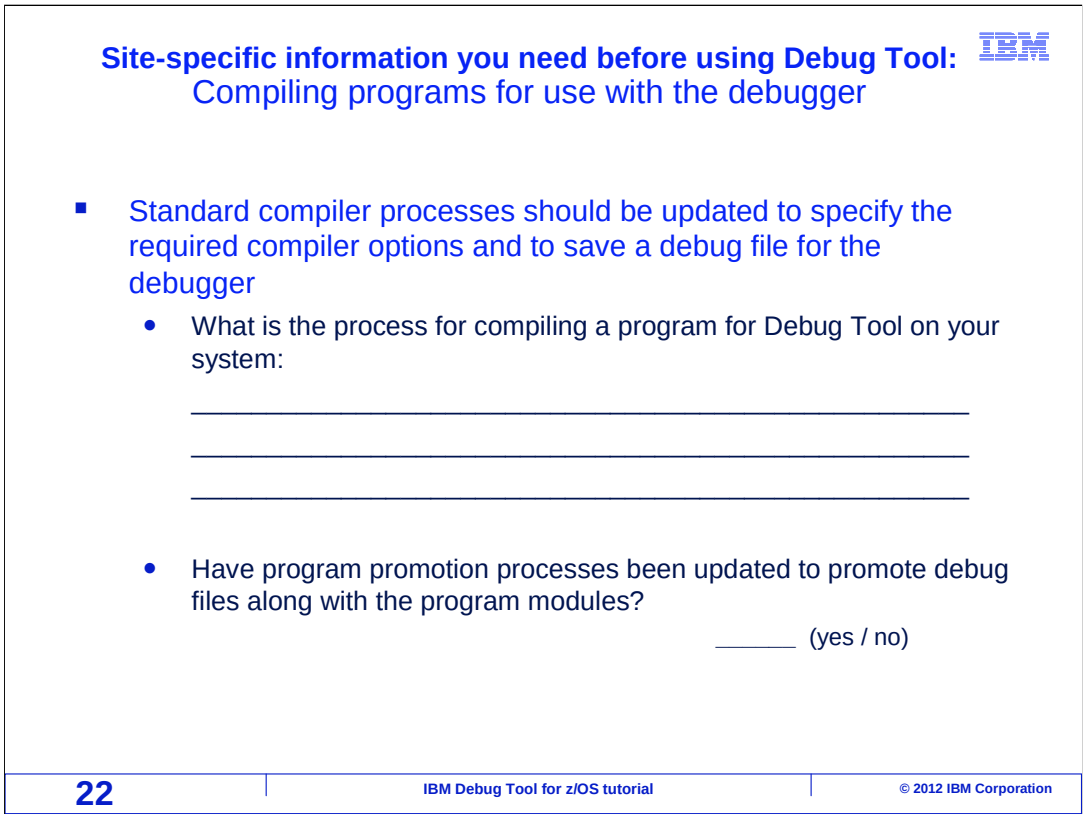
BTS (Batch Terminal Simulator):
Use Debug Tool online panels to create a setup file to run BTS in TSO. Run your IMS transaction under BTS. Debug Tool is displayed on your TSO terminal or a GUI interface.



21 | IBM Debug Tool for z/OS tutorial |

And use the same debugging engine to work with your IMS transaction manager transactions. To start the debugger for an online IMS program, use a special Debug Tool panel in TSO to specify the program name. Then, you can run your transaction normally in a regular IMS message processing region. When your program runs, Debug Tool is displayed on a 3270 terminal or a remote GUI debugger.

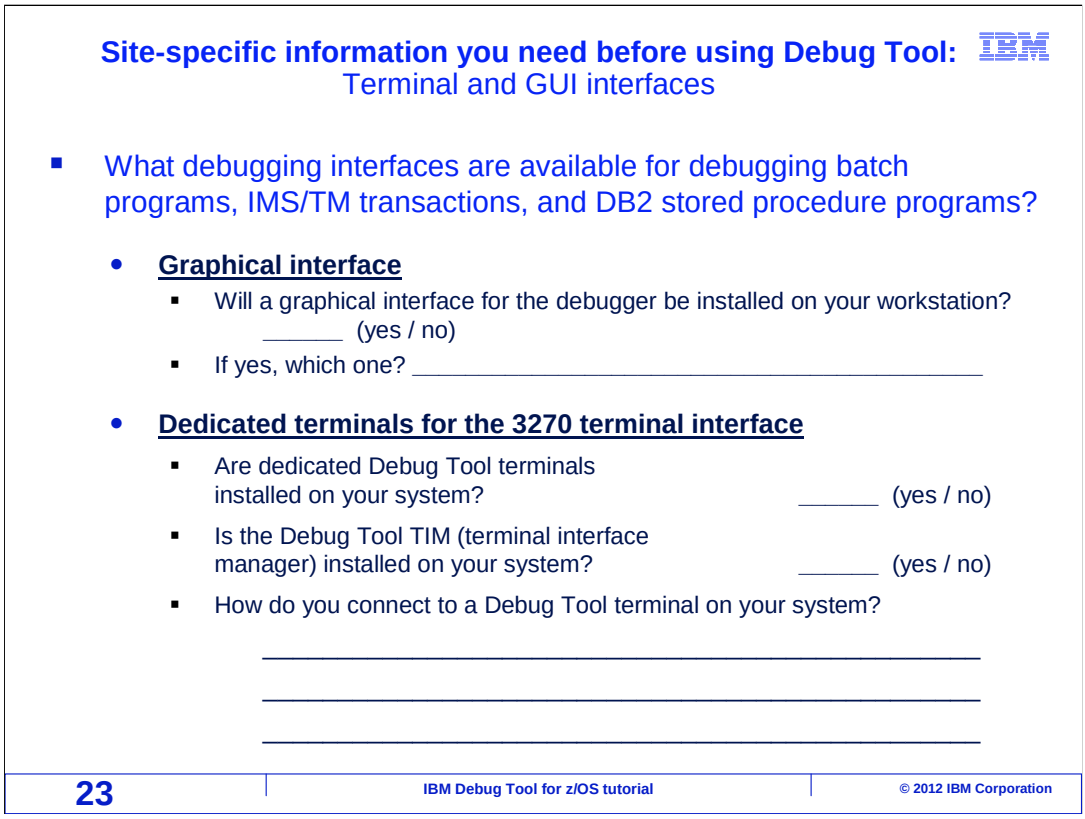
If you have IBM Batch Terminal Simulator software, you can debug an IMS transaction while it is running in your TSO region or in a batch job that you control.



The standard compiler processes should be updated so that the required compiler options are specified to prepare the program for debugging, and to save a debug file.

If you do not maintain your compile JCL, you may need to ask how to compile a program to use it with Debug Tool. Find out if the standard compiler processes have been updated for Debug Tool. If so, are there special parameters or selections needed for the debugger? Or will you make modifications to your own compiler JCL? These are often decisions made by the system programmer or source library administrator. If you are not sure, contact them to find out how compiler processes on your system have been customized for use with Debug Tool.

Also, find out if your program promotion processes have been customized so that the debug files are promoted along with the program. This promotion of debug files is needed if you want the ability to debug programs at different stages of their life cycle.

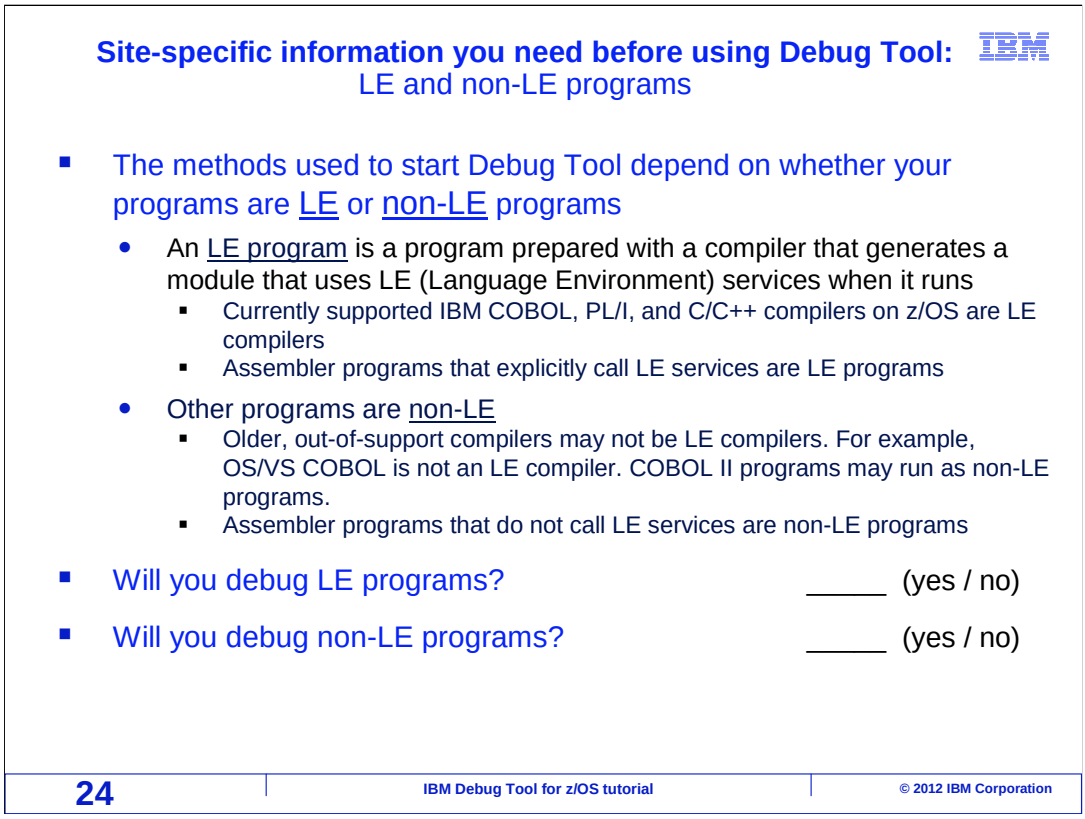


Before using the debugger, you need to know how it is configured on your system. If you are not sure about the answers, you should contact the person who installed Debug Tool. The answers will help you understand which sections of the this tutorial to take, since different sections apply to different ways that the debugger can be configured. Take only the sections of the tutorial that correspond to the way that Debug Tool is configured on your system.

First, determine the answer to the question: "Will GUI debugging software be installed on your workstation?". And if the answer is yes, which software product will be used? One product that includes the GUI debugger is Rational Developer for System z. Another way to make the GUI debugger available is to install CICS Explorer, and then install the Debug Tool plug-in.

Next, determine if you will be using the 3270 interface. If so, find out if dedicated terminals for the 3270 interface have been installed on your system. If they have, determine if those terminals use a service called the Debug Tool "terminal interface manager", or "TIM". You will need to know whether you will be using a TIM terminal when you start the debugger.

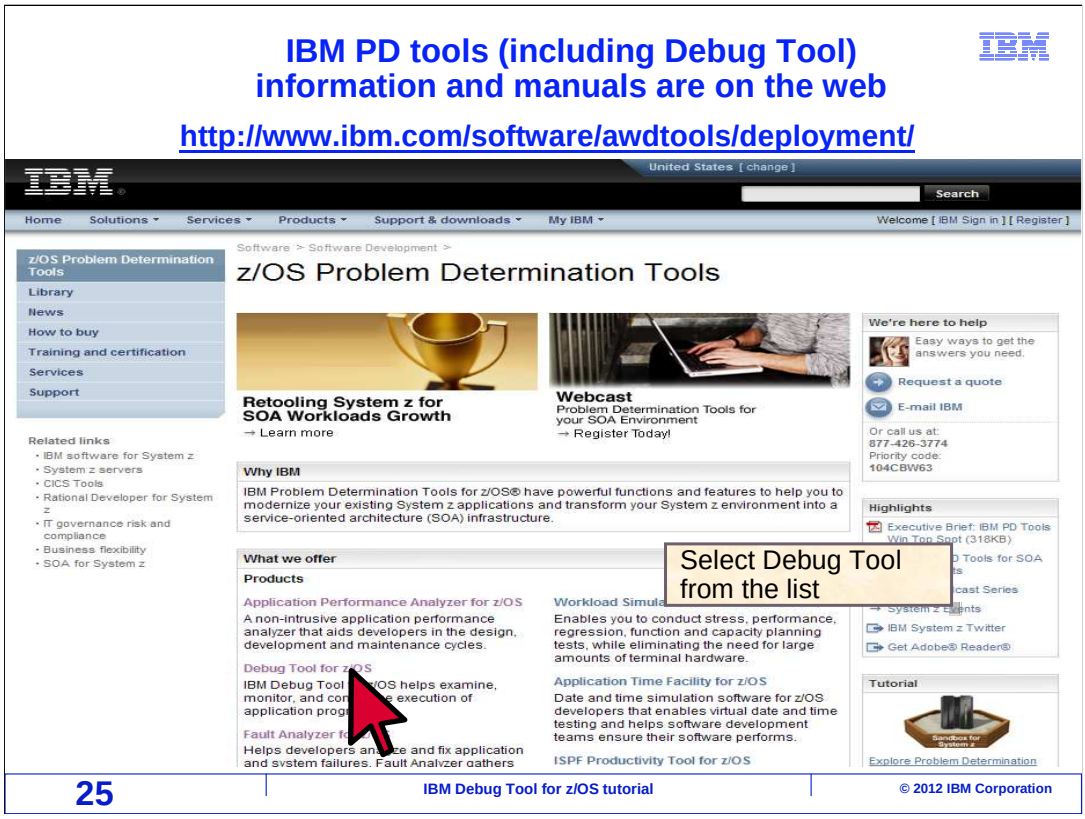
The procedure to connect to a Debug Tool terminal will be unique to your system, based on the terminal emulator used and the network address of your host systems. If you have not been shown how to connect to a Debug Tool terminal, check with the person or group that installed Debug Tool.



The way you start Debug Tool may depend on whether your application programs use Language Environment (LE) services or not. Language Environment is a collection of run-time services that IBM supplies that are used by many programs running on z/OS. A Language Environment-conforming program is one that is prepared with a compiler that generates a module that uses Language Environment services when it runs. If you are using currently supported IBM COBOL, PL/I, or C or C++ compilers, then you have Language Environment-conforming programs. Or if you have explicitly added code to your assembler programs to call Language Environment services, then those assembler programs are also Language Environment-conforming programs.

However, older, out of support compilers may not be LE compilers. For example, if you are using the old OS/VS COBOL compiler, those programs are non-Language Environment-conforming programs. If you have assembler programs that do not have Language Environment calls coded into them, then they are also non- Language Environment programs.

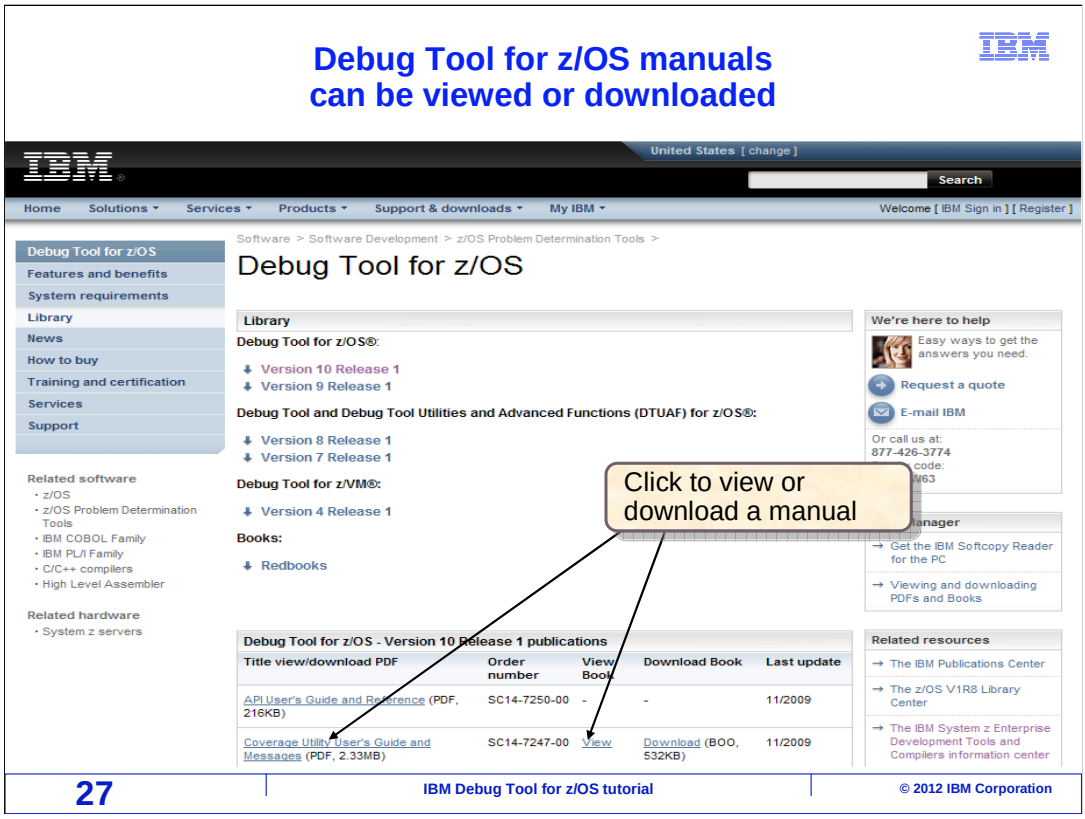
Make a note of which types of programs you will be using with Debug Tool, Language Environment programs, non- Language Environment programs, or some of each.



You can get more information about Debug Tool, and all of the IBM problem determination tools, on the web. Direct your browser to this URL: www.ibm.com/software/awdtools/deployment .

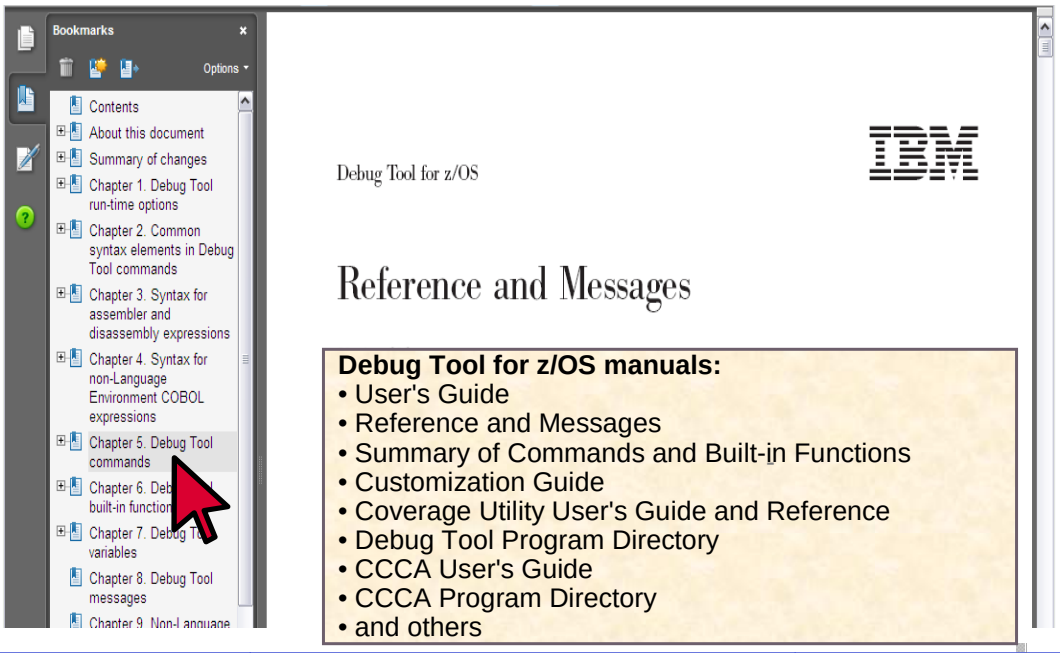
To get to the Debug Tool page, select the link for "Debug Tool for z/OS".

You can view or download manuals from the web. From the Debug Tool page, there is a link to the library. Click the word library on the left side of the screen.



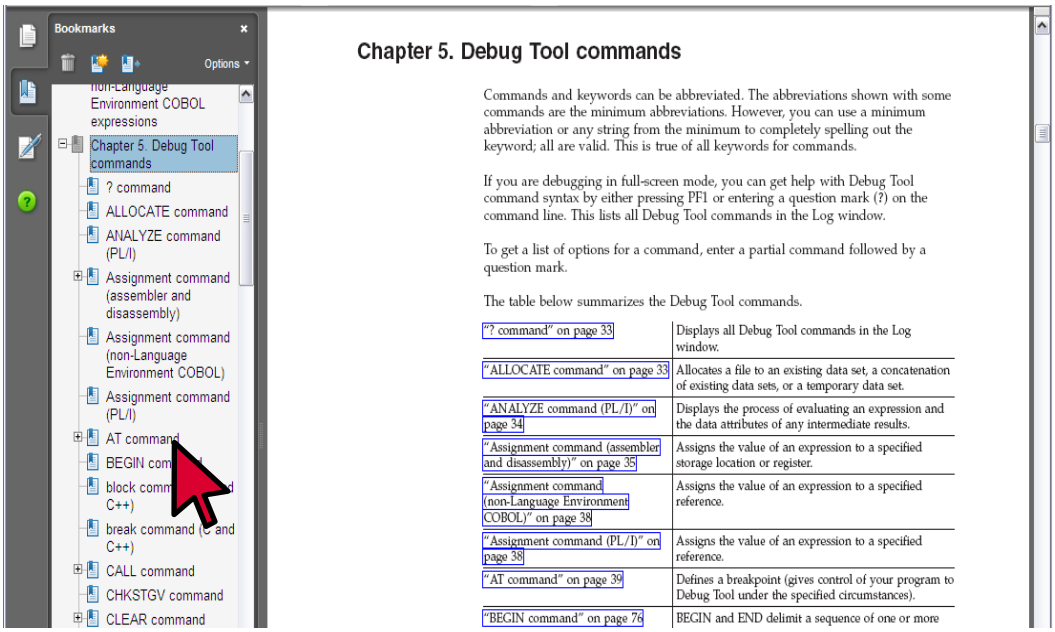
From the library page, you can view a manual by clicking a link in either the PDF or View book column, or you can download a manual by right clicking the link in the PDF column.

Debug Tool for z/OS manuals available from the web



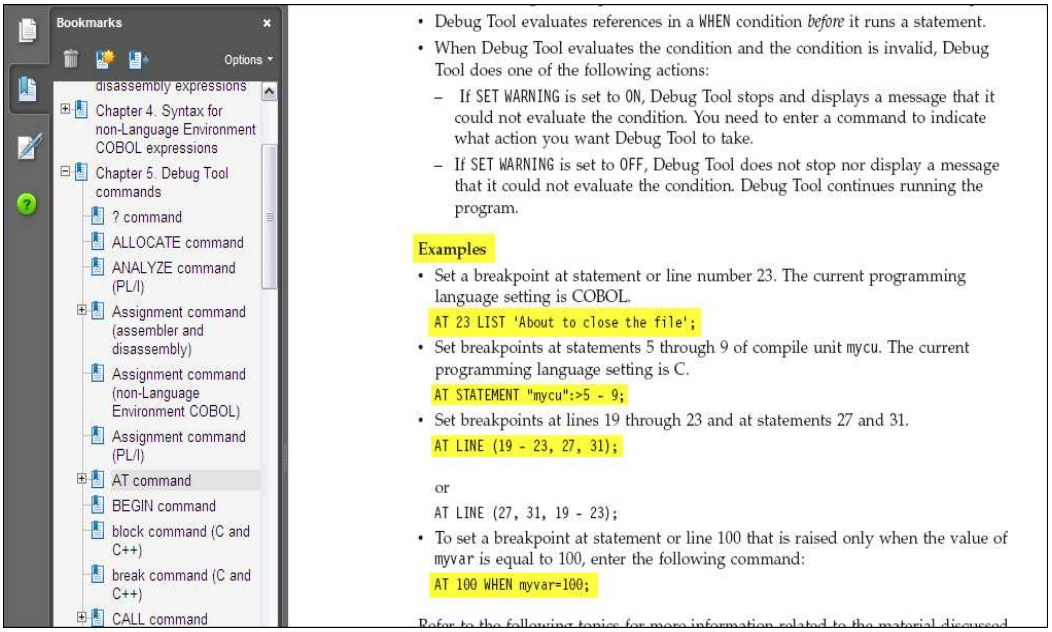
The Debug Tool library provides several manuals. You will find helpful information about the debugger in the “Summary of Commands and Built-in functions”, the “Users guide”, and in the “Reference and Messages” manuals. The Reference and Messages manual has a chapter that describes the commands in detail. In this example, the link for the "Debug Tool commands" chapter is selected.

Commands are described in detail in the Reference and Messages manual



A list of commands is shown. You can easily navigate to see more information by clicking on a link for a command on the left side of the page.

Commands syntax and examples are shown in the Reference and Messages manual



The Reference and Messages manual describes the syntax and parameters available for each command. For many commands, examples are shown and explained, as on this page which is showing variations of the AT command.

That is the end of this section, an introduction to Debug Tool.

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_DTv12s01Introduction.ppt

This module is also available in PDF format at: ../DTv12s01Introduction.pdf

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

Trademarks, copyrights, and disclaimers