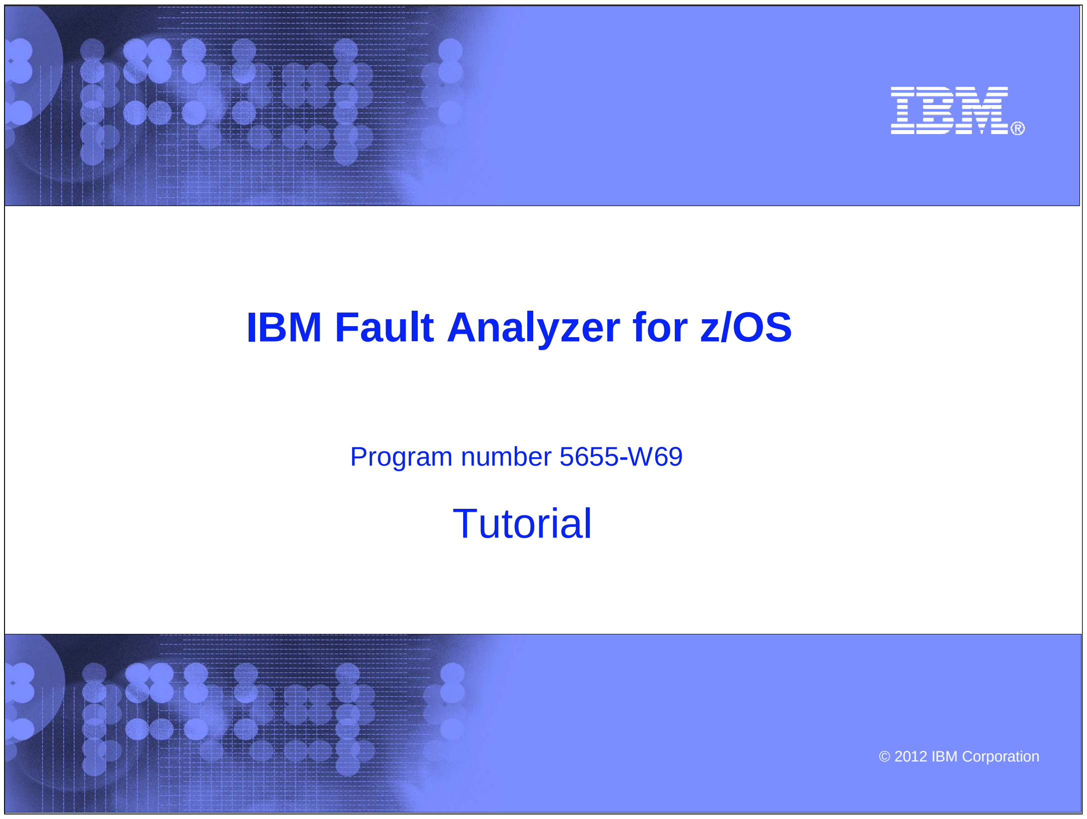


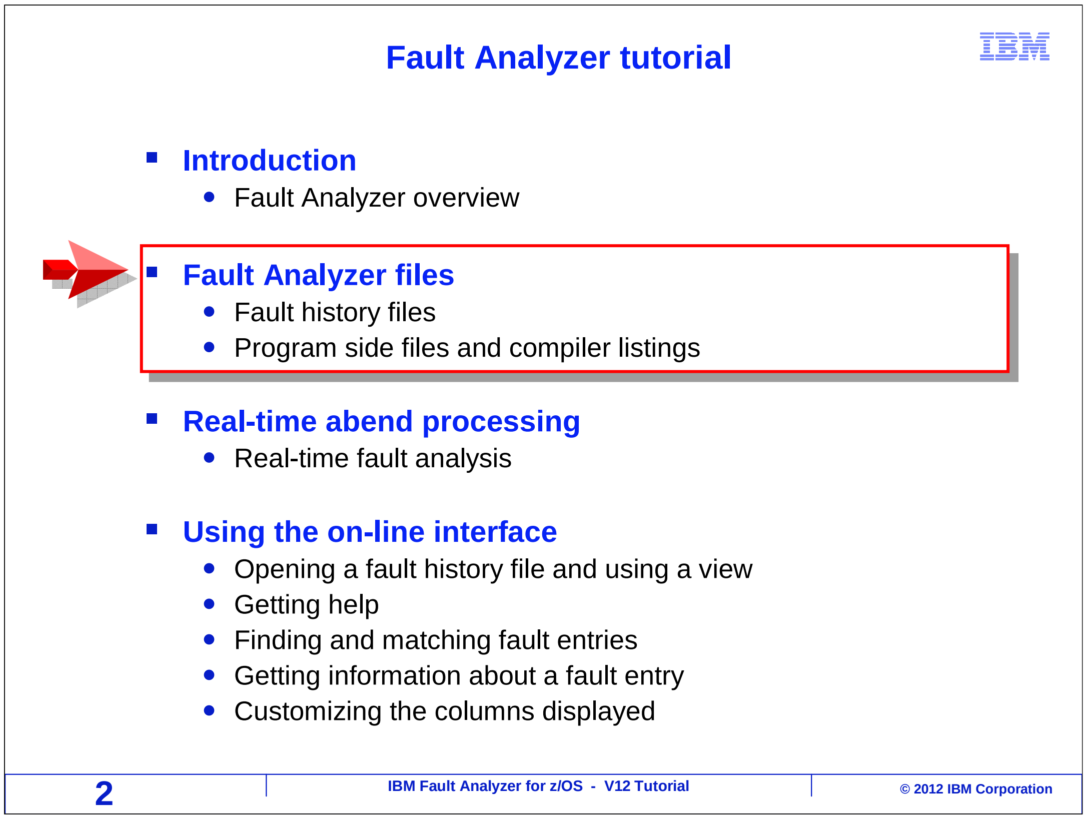
IBM Fault Analyzer for z/OS

Program number 5655-W69

Tutorial

© 2012 IBM Corporation

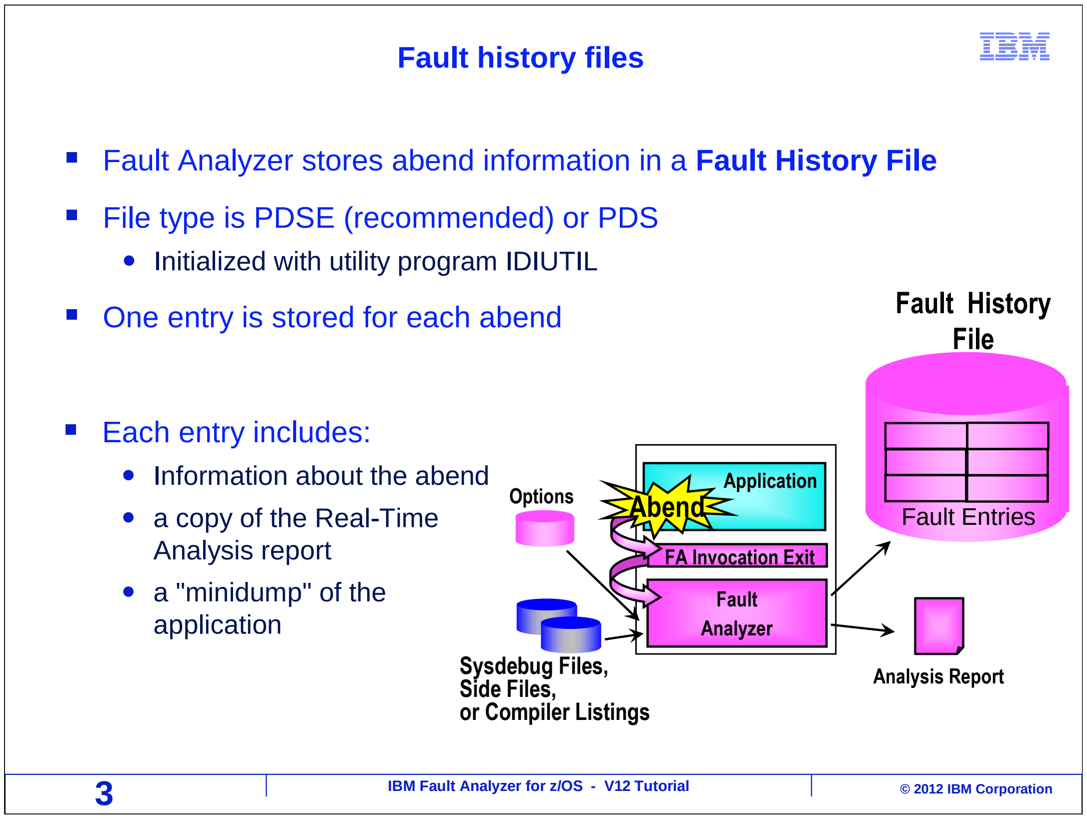
This is the tutorial for IBM's Fault Analyzer for z/OS®, one of the IBM zSeries® problem determination tools.



Fault Analyzer tutorial

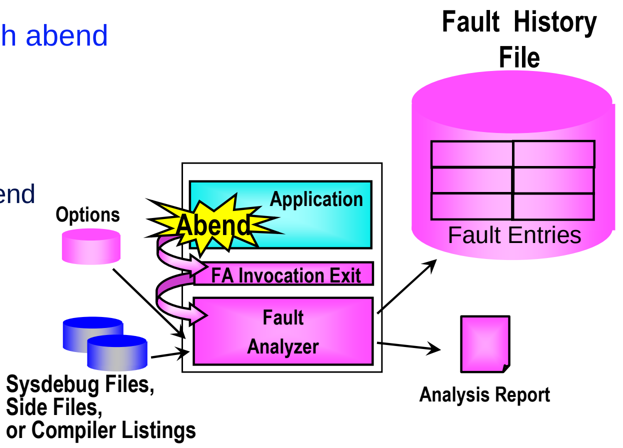
- **Introduction**
  - Fault Analyzer overview
- **Fault Analyzer files**
  - Fault history files
  - Program side files and compiler listings
- **Real-time abend processing**
  - Real-time fault analysis
- **Using the on-line interface**
  - Opening a fault history file and using a view
  - Getting help
  - Finding and matching fault entries
  - Getting information about a fault entry
  - Customizing the columns displayed

In this section you will learn about fault history files, and how side files and compiler listings can be used by Fault Analyzer.



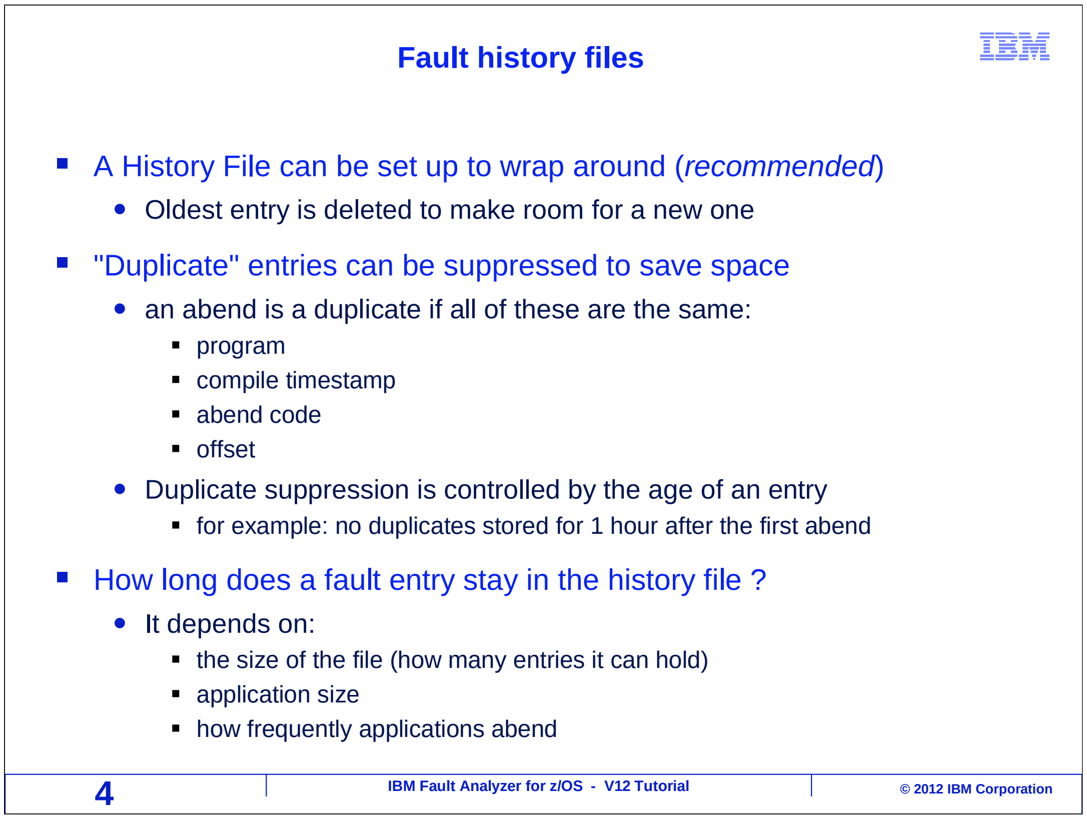
Fault history files

- Fault Analyzer stores abend information in a **Fault History File**
- File type is PDSE (recommended) or PDS
  - Initialized with utility program IDIUTIL
- One entry is stored for each abend
- Each entry includes:
  - Information about the abend
  - a copy of the Real-Time Analysis report
  - a "minidump" of the application



A fault history file is a repository where Fault Analyzer stores information about abends. There can be one or more history files on each system. They are large partitioned data sets – PDS or PDSE libraries – and are typically set up by the installer.

Fault Analyzer stores one member, called a fault entry, for each abend that it collects. In the entry, it stores a complete copy of the real-time analysis report and a mini-dump.



Fault history files

- A History File can be set up to wrap around (*recommended*)
  - Oldest entry is deleted to make room for a new one
- "Duplicate" entries can be suppressed to save space
  - an abend is a duplicate if all of these are the same:
    - program
    - compile timestamp
    - abend code
    - offset
  - Duplicate suppression is controlled by the age of an entry
    - for example: no duplicates stored for 1 hour after the first abend
- How long does a fault entry stay in the history file ?
  - It depends on:
    - the size of the file (how many entries it can hold)
    - application size
    - how frequently applications abend

4 IBM Fault Analyzer for z/OS - V12 Tutorial © 2012 IBM Corporation

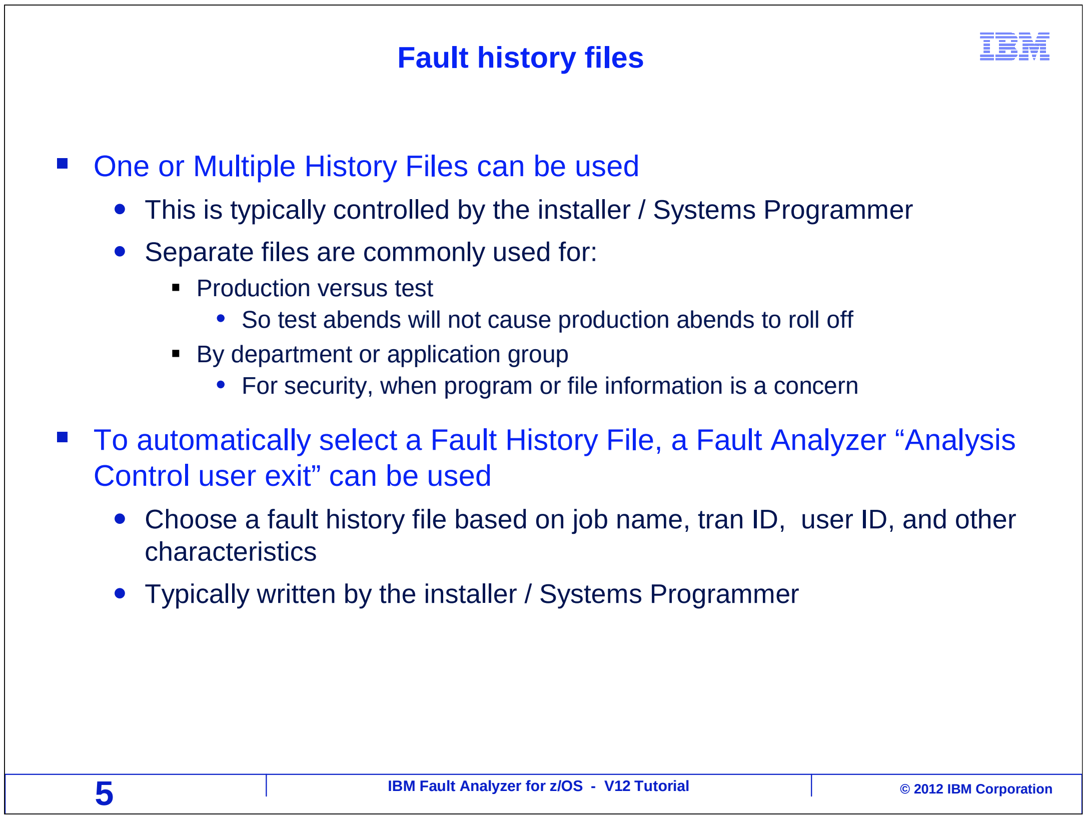
A history file can, and should, be set up to **wrap around**. Then when it becomes full, the old entries are automatically deleted to make room for new ones.

Fault Analyzer can be configured so that duplicate abends are not stored. This can be done to save space on a fault history file. For example, a batch job abends and fault analyzer processes it. A new fault entry is added to a fault history file, and the information is stored there. Then, the same job is submitted again and abends again for exactly the same reason. With duplicate suppression, Fault Analyzer does not store information for the second abend.

To be considered a duplicate, an abend has to occur in the same program, at the same offset in the program, with the same abend code, and the program must have the same compile timestamp. The idea is that if you fix the first one, you have fixed them all. After some period of time has elapsed, then fault analyzer will store one more abend if the job fails again.

As a general recommendation, suppressing duplicates for batch jobs may not save that much space. However, it can be a good idea to suppress duplicates in high-volume online systems. If an application goes bad in a system like that, rapid fire abends can fill up a fault history file in a hurry.

Eventually, a fault entry will roll off of the history file. How long it stays there depends on the size of the history file and how often abends occur. If your abends are being deleted too quickly, give your systems programmer a call and ask them to increase the size of the file. If you are concerned about losing a particular fault entry, you can copy it, as you would copy any PDS member. You will see more about copying fault entries in a later section.



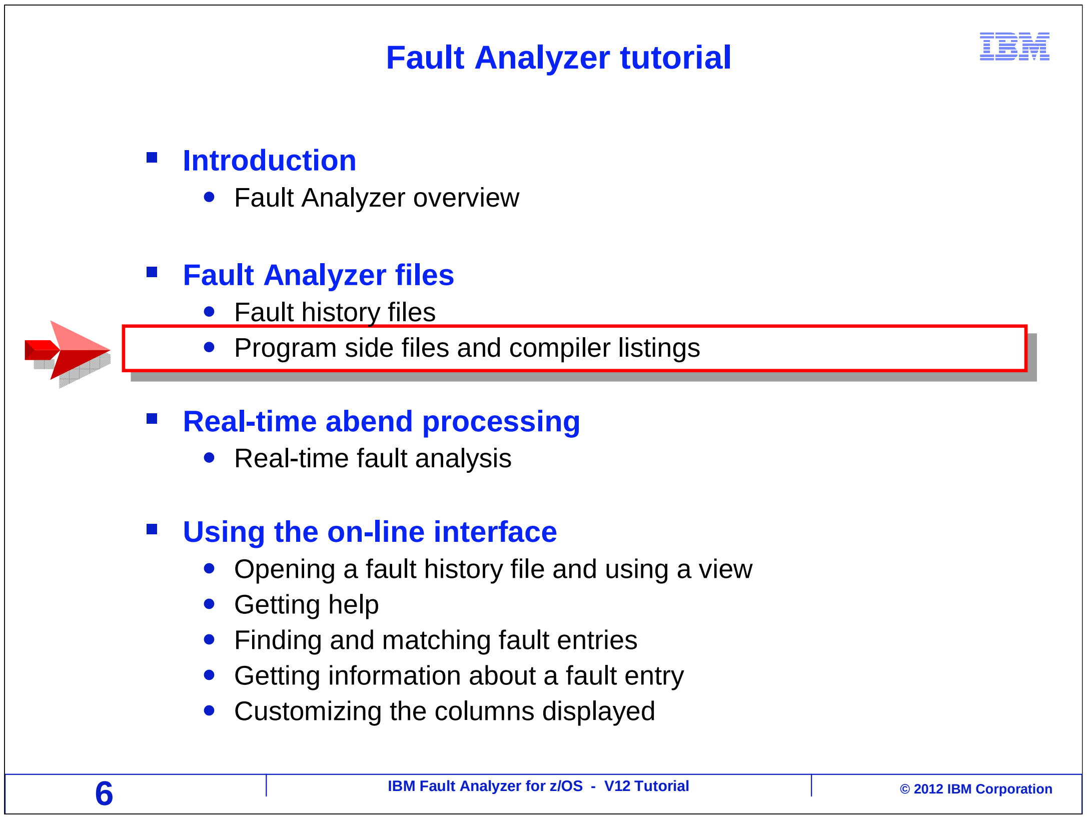
Fault history files

- One or Multiple History Files can be used
  - This is typically controlled by the installer / Systems Programmer
  - Separate files are commonly used for:
    - Production versus test
      - So test abends will not cause production abends to roll off
    - By department or application group
      - For security, when program or file information is a concern
- To automatically select a Fault History File, a Fault Analyzer “Analysis Control user exit” can be used
  - Choose a fault history file based on job name, tran ID, user ID, and other characteristics
  - Typically written by the installer / Systems Programmer

5 | IBM Fault Analyzer for z/OS - V12 Tutorial | © 2012 IBM Corporation

Your system can have one big history file for everything, or several history files. Some organizations do well with only one file. However, it is typically best to separate test and production abends, if only to prevent a large number of test abends from deleting production entries. Some customers separate batch and online abends. And there can even be different history files for different groups or applications. It just depends on the complexity of your system and how often abends occur.

The installer can customize a special Fault analyzer exit to control which abends are stored in which history files. The exit can examine the job name, program name, user id, and other information to make it is determination.

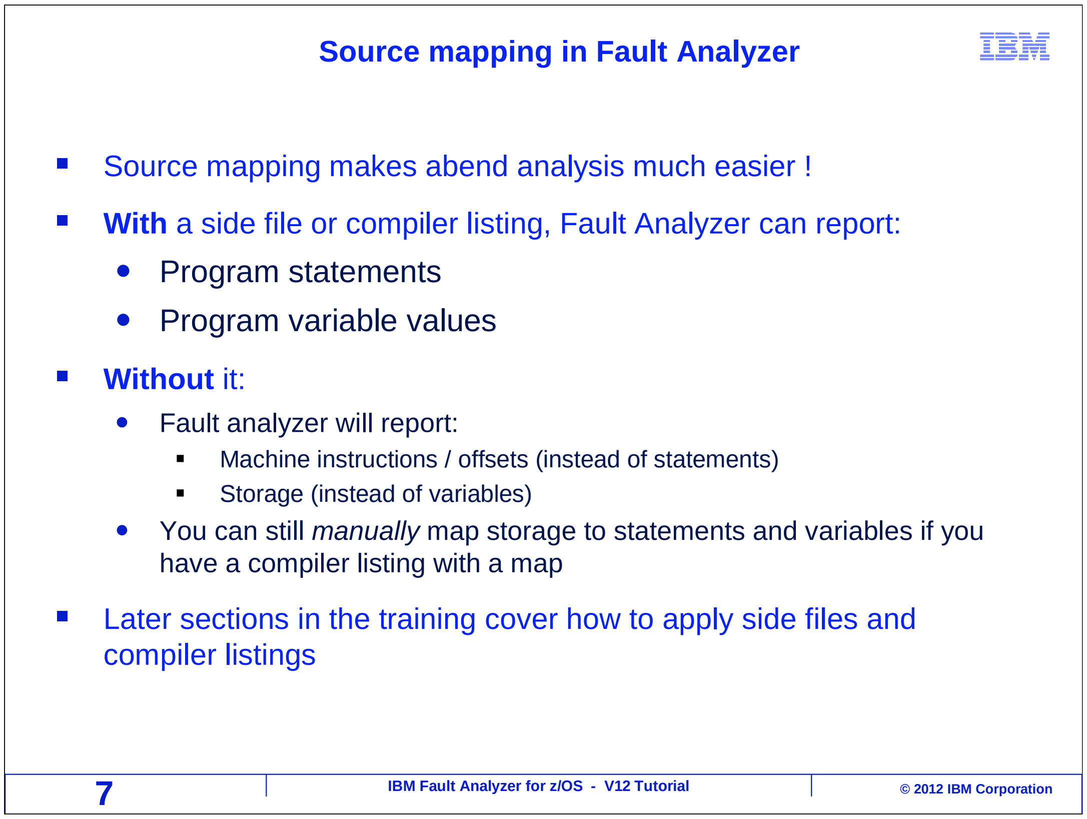


Fault Analyzer tutorial

- **Introduction**
  - Fault Analyzer overview
- **Fault Analyzer files**
  - Fault history files
  - Program side files and compiler listings
- **Real-time abend processing**
  - Real-time fault analysis
- **Using the on-line interface**
  - Opening a fault history file and using a view
  - Getting help
  - Finding and matching fault entries
  - Getting information about a fault entry
  - Customizing the columns displayed

6 | IBM Fault Analyzer for z/OS - V12 Tutorial | © 2012 IBM Corporation

Next you will see an overview of using side files and compiler listings to provide source level mapping, to display program statements and variable values.



Source mapping in Fault Analyzer

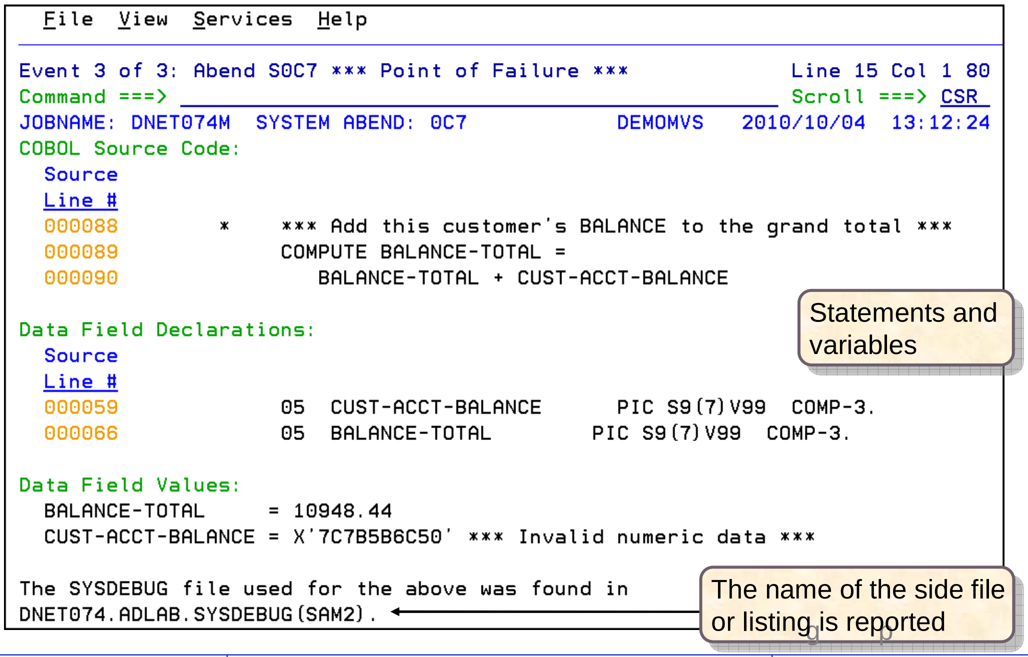
- Source mapping makes abend analysis much easier !
- **With** a side file or compiler listing, Fault Analyzer can report:
  - Program statements
  - Program variable values
- **Without** it:
  - Fault analyzer will report:
    - Machine instructions / offsets (instead of statements)
    - Storage (instead of variables)
  - You can still *manually* map storage to statements and variables if you have a compiler listing with a map
- Later sections in the training cover how to apply side files and compiler listings

7 IBM Fault Analyzer for z/OS - V12 Tutorial © 2012 IBM Corporation

Fault analyzer captures and reports information about program abends, whether source information is available for the programs or not. However, if you have side files or compiler listings, your work can be much easier because Fault Analyzer can pinpoint the abending program statement and show the values of variables.

If side files or compiler listings are not available, Fault Analyzer still reports a lot of detailed information, but it cannot show you source-level detail. Instead of program statements and variables, it will show the offset of the abend, machine instructions, and storage. Use the source mapping capability if you can, because it can save you a lot of time.

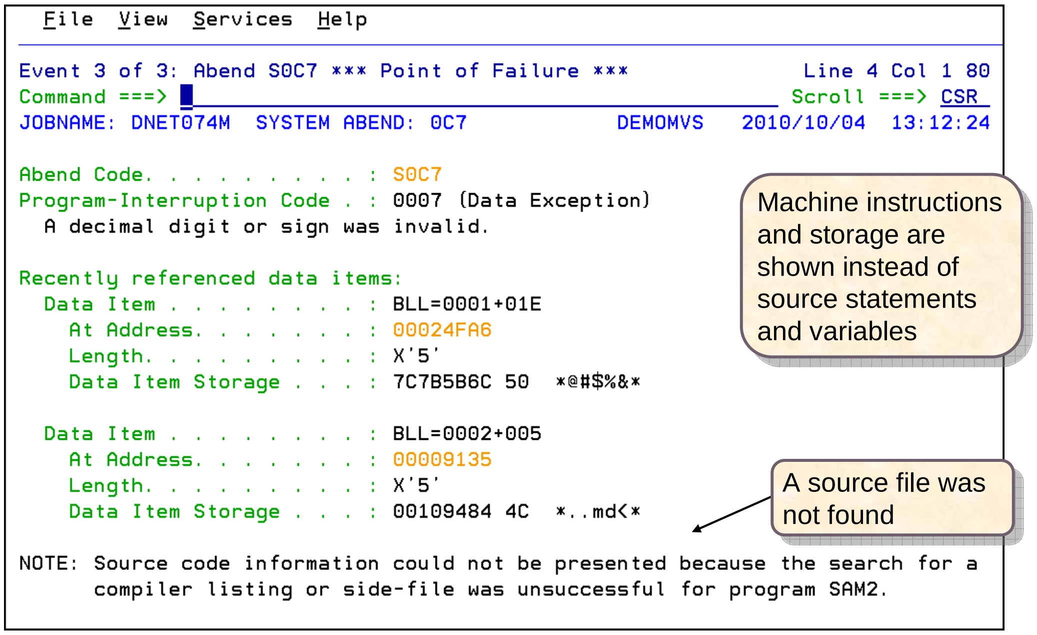
If a side file or compiler listings is found program statements and variables can be reported



```
   File  View  Services  Help

 Event 3 of 3: Abend S0C7 *** Point of Failure ***              Line 15 Col 1 80
 Command ===> ________________________________________________  Scroll ===> CSR
 JOBNAME: DNET074M  SYSTEM ABEND: 0C7            DEMOMVS   2010/10/04  13:12:24
 COBOL Source Code:
   Source
   Line #
   000088        *    *** Add this customer's BALANCE to the grand total ***
   000089             COMPUTE BALANCE-TOTAL =
   000090                BALANCE-TOTAL + CUST-ACCT-BALANCE

 Data Field Declarations:
   Source
   Line #
   000059             05  CUST-ACCT-BALANCE      PIC S9(7)V99  COMP-3.
   000066             05  BALANCE-TOTAL        PIC S9(7)V99  COMP-3.

 Data Field Values:
   BALANCE-TOTAL     = 10948.44
   CUST-ACCT-BALANCE = X'7C7B5B6C50' *** Invalid numeric data ***

 The SYSDEBUG file used for the above was found in
 DNET074.ADLAB.SYSDEBUG(SAM2).
```

If Fault Analyzer finds a side file or compiler listing for a program, it can show program statements and variable values. In this example, it is showing a COBOL program statement where an abend occurred, and the values of the variables that it references. The name of the file used to perform source mapping is reported.

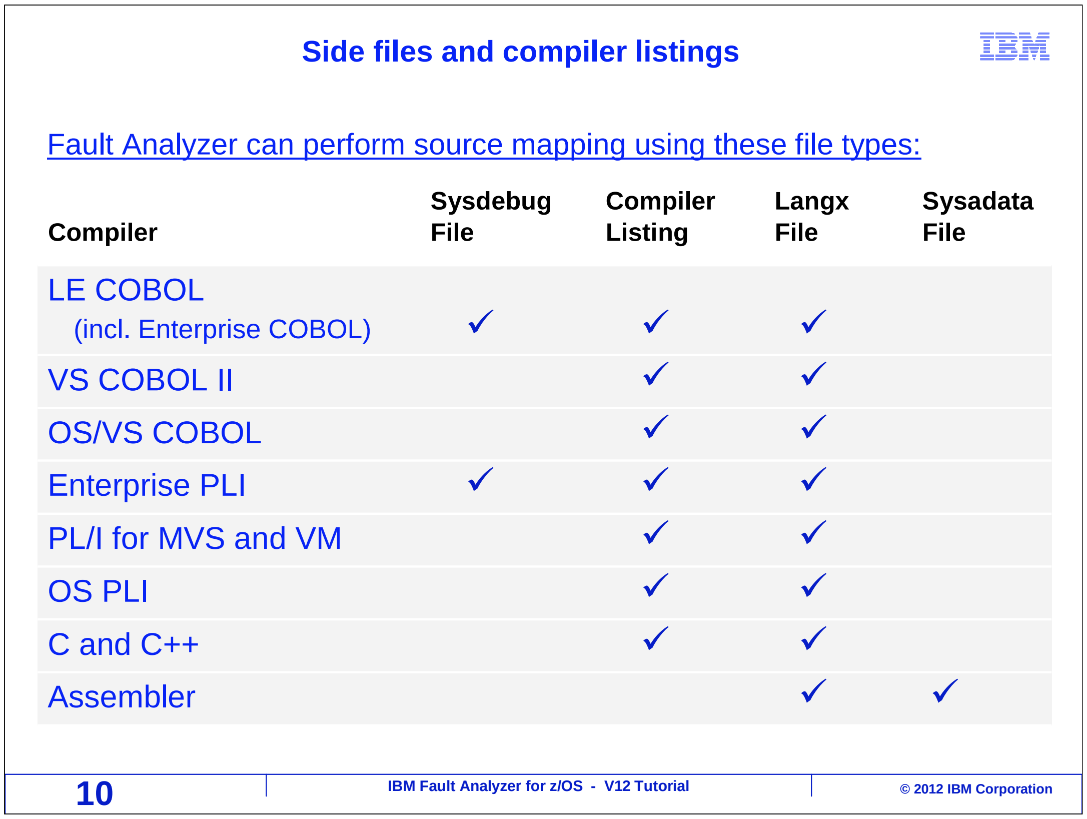
If a side file or compiler listing is not found program statements and variables are not reported



```
  File  View  Services  Help

Event 3 of 3: Abend S0C7 *** Point of Failure ***              Line 4 Col 1 80
Command ===>                                                    Scroll ===> CSR
JOBNAME: DNET074M  SYSTEM ABEND: 0C7            DEMOMVS   2010/10/04  13:12:24

Abend Code. . . . . . . . . . : S0C7
Program-Interruption Code . : 0007 (Data Exception)
  A decimal digit or sign was invalid.

Recently referenced data items:
  Data Item . . . . . . . . . : BLL=0001+01E
    At Address. . . . . . . . : 00024FA6
    Length. . . . . . . . . . : X'5'
    Data Item Storage . . . . : 7C7B5B6C 50   *@#$%&*

  Data Item . . . . . . . . . : BLL=0002+005
    At Address. . . . . . . . : 00009135
    Length. . . . . . . . . . : X'5'
    Data Item Storage . . . . : 00109484 4C   *..md<*

NOTE: Source code information could not be presented because the search for a
      compiler listing or side-file was unsuccessful for program SAM2.
```

Machine instructions and storage are shown instead of source statements and variables
A source file was not found

If, on the other hand, Fault Analyzer does not find a side file or compiler listing, it still produces a detailed report. But it cannot display program source statements and variables. Instead, it shows machine instructions, storage, and offsets. A message is displayed indicating that a corresponding source mapping file could not be found for the program.



Side files and compiler listings

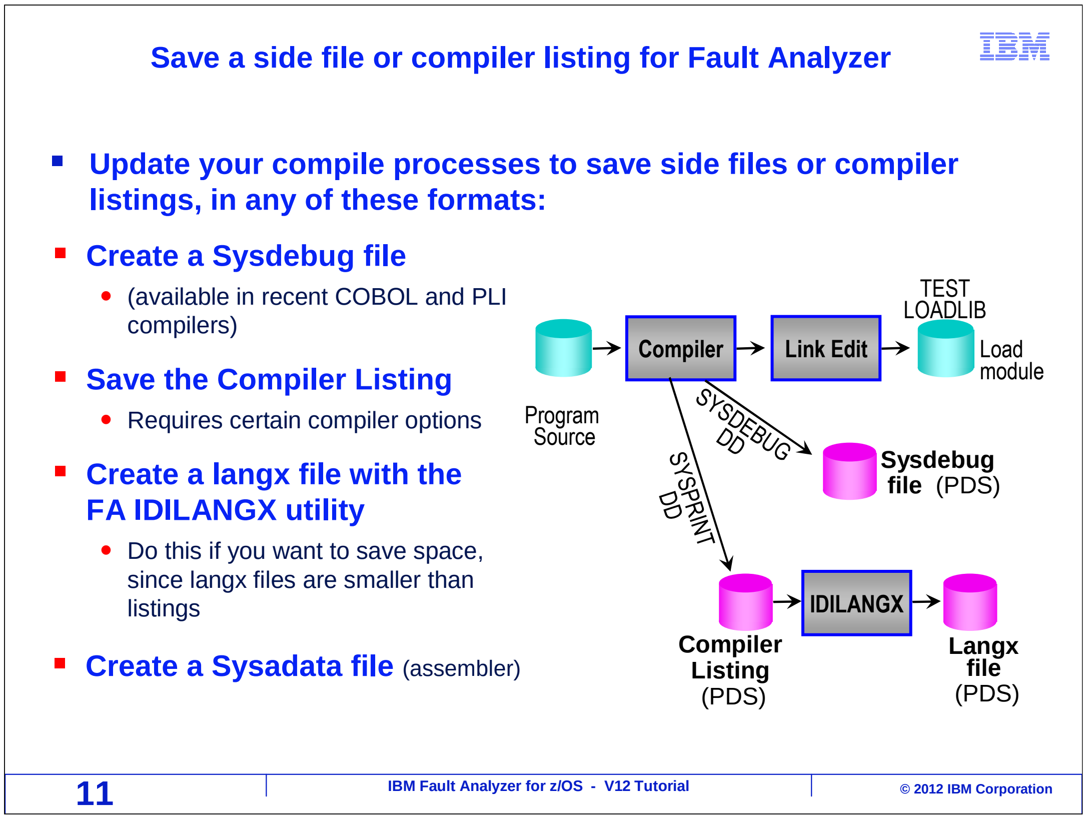
Fault Analyzer can perform source mapping using these file types:

Compiler	Sysdebug File	Compiler Listing	Langx File	Sysadata File
LE COBOL (incl. Enterprise COBOL)	✓	✓	✓	
VS COBOL II		✓	✓	
OS/VS COBOL		✓	✓	
Enterprise PLI	✓	✓	✓	
PL/I for MVS and VM		✓	✓	
OS PLI		✓	✓	
C and C++		✓	✓	
Assembler			✓	✓

10 IBM Fault Analyzer for z/OS - V12 Tutorial © 2012 IBM Corporation

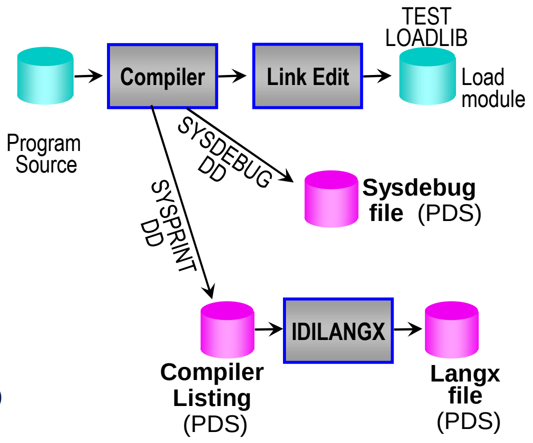
If someone in your organization has already set up your compile processes for Fault analyzer, then the right files are generated for you when you compile a program. However, if it is your responsibility to update the compile processes, then research how to set up each compiler individually.

For the LE COBOL compilers, including Enterprise COBOL, and for recent versions of Enterprise PLI, fault analyzer supports sysdebug files, compiler listings, and LANGX files. For all other compiled languages shown on this list, fault analyzer supports compiler listings and LANGX files. For assembler programs, fault analyzer supports LANGX files and SYSADATA files.



Save a side file or compiler listing for Fault Analyzer

- **Update your compile processes to save side files or compiler listings, in any of these formats:**
- **Create a Sysdebug file**
  - (available in recent COBOL and PLI compilers)
- **Save the Compiler Listing**
  - Requires certain compiler options
- **Create a langx file with the FA IDILANGX utility**
  - Do this if you want to save space, since langx files are smaller than listings
- **Create a Sysadata file** (assembler)



11 IBM Fault Analyzer for z/OS - V12 Tutorial © 2012 IBM Corporation

A side file or compiler listing can be created when you compile a program. Different compilers can create different kinds of side files. Some newer compilers produce a **sysdebug** file if you specify the right compiler options. This is the **best** method if your compiler supports them because they are relatively compressed in size, and they can be used by other IBM software tools such as debug tool.

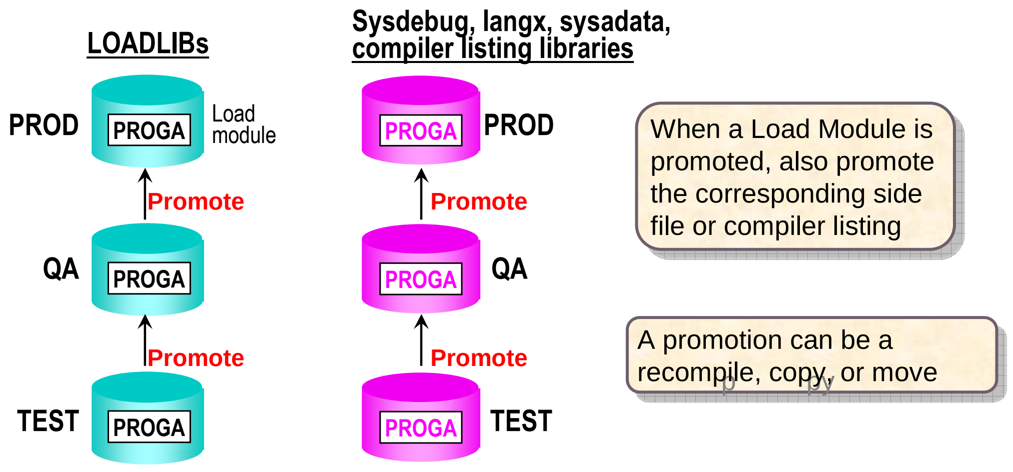
Or, you could save your **compiler listing** in a file. To be able to use a listing, there are certain compiler options that have to be turned on. It is a **best practice** to store compiler listings, whether you are using fault analyzer or not. A compiler listing documents how your program was generated, and **should** be saved for as long as the program is in use.

If your compiler is not capable of generating sysdebug files, and your organization chooses not to store compiler listings, there is a third option. Fault Analyzer provides a utility program called IDILANGX that reads compiler listings and saves the needed information in something called a LANGX side file. LANGX files are a good deal smaller than compiler listings, but comparable in size to Sysdebug files.

For assembler programs, either save a SYSADATA file or a LANGX file.

Promote side files or compiler listings for PD tools

- Keep files used for source mapping throughout the program life cycle
- Update your promotion processes to retain the files



When a Load Module is promoted, also promote the corresponding side file or compiler listing

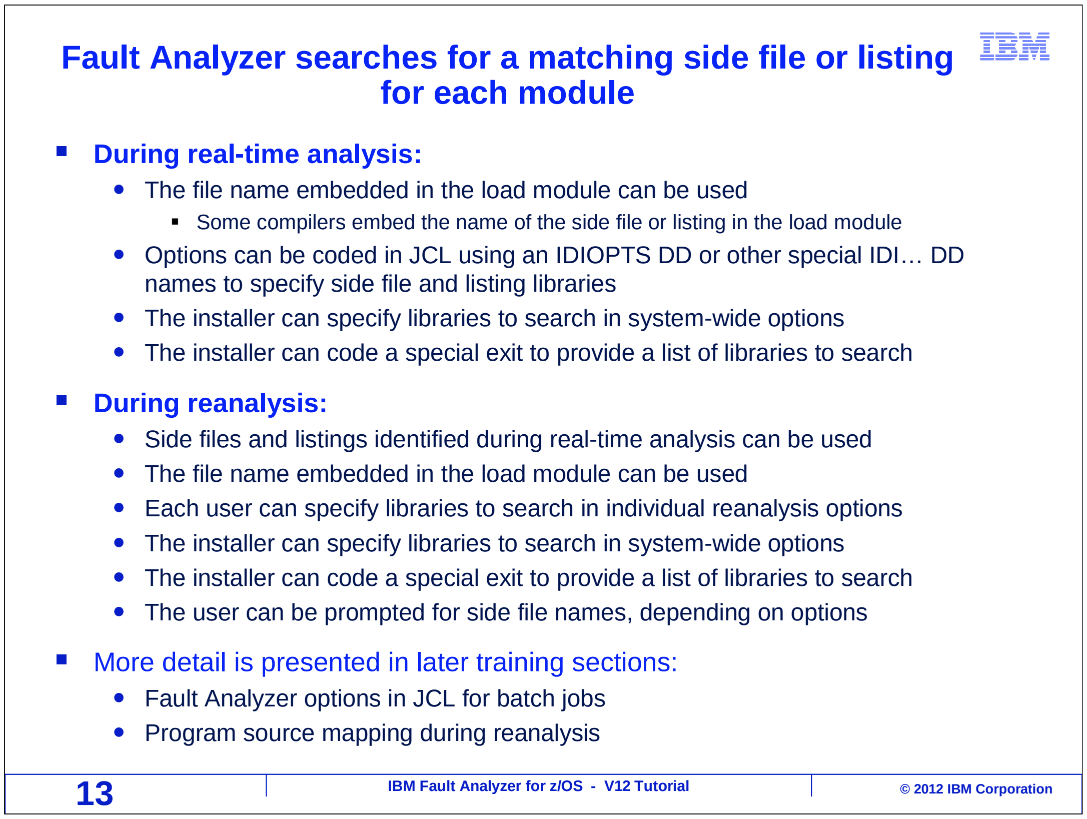
A promotion can be a recompile, copy, or move

12 IBM Fault Analyzer for z/OS - V12 Tutorial © 2012 IBM Corporation

If you want to enable source mapping with **production** abends, your program promotion processes should take that into account. When a program is promoted, say from test to QA, or from QA to production, then the **side file or compiler listing** should be promoted along with it.

That way, you will have source information for your **production** programs as well. Typically, for any **load library** you have that contains an executable program, you should have a corresponding side file or listing library.

When the **load module** gets promoted, the corresponding member or members in the side file or listings libraries should also get promoted. Depending on how your promotion processes are designed, it can be done as a recompile, a copy, or a move.



Fault Analyzer searches for a matching side file or listing for each module

- **During real-time analysis:**
  - The file name embedded in the load module can be used
    - Some compilers embed the name of the side file or listing in the load module
  - Options can be coded in JCL using an IDIOPTS DD or other special IDI… DD names to specify side file and listing libraries
  - The installer can specify libraries to search in system-wide options
  - The installer can code a special exit to provide a list of libraries to search
- **During reanalysis:**
  - Side files and listings identified during real-time analysis can be used
  - The file name embedded in the load module can be used
  - Each user can specify libraries to search in individual reanalysis options
  - The installer can specify libraries to search in system-wide options
  - The installer can code a special exit to provide a list of libraries to search
  - The user can be prompted for side file names, depending on options
- More detail is presented in later training sections:
  - Fault Analyzer options in JCL for batch jobs
  - Program source mapping during reanalysis

13 | IBM Fault Analyzer for z/OS - V12 Tutorial | © 2012 IBM Corporation

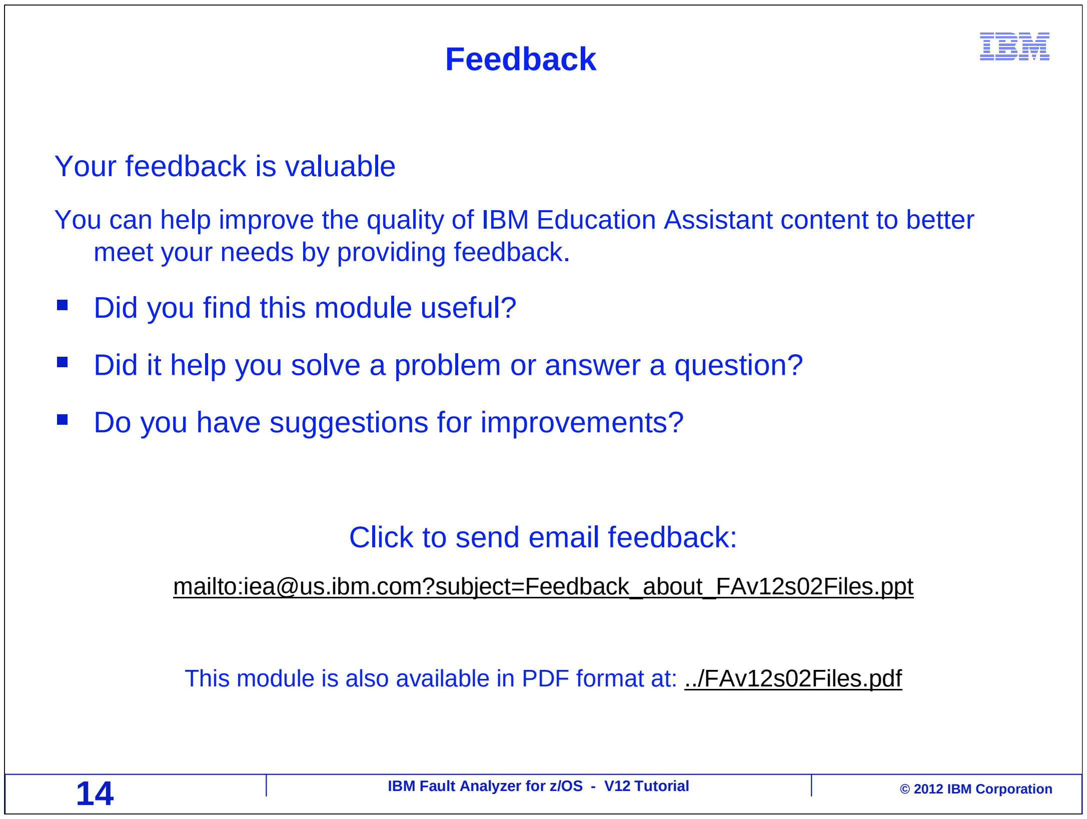
Fault Analyzer automatically searches for matching side files or compiler listings. As it searches for a matching file, it compares the time stamp and contents of the load module against the time stamp and contents of the side file or listing looking for the best match.

During real-time analysis, it first examines a program's load module. Some compilers embed the name of a SYSDEBUG file or listing in the module. If a name is embedded, that file is checked to see if is a match. Fault Analyzer options can optionally be specified in the application's run-time JCL to name side file and compiler libraries. The installer can also specify libraries to be searched in Fault Analyzer's system-wide options, and a Fault Analyzer exit can also be coded to provide the names of libraries to be searched. All of these can be searched to find the best match.

During interactive reanalysis, if a side file or listing was identified during real-time analysis, that file can be used. Next, the file name embedded in the load module can be checked. Each user can code personal Fault Analyzer options to specify libraries to be searched. After that, system-wide options are checked for libraries to search, and an exit can be installed to provide library names. If after searching all of these, a match is not found, the user may be prompted during interactive reanalysis to name a side file or compiler listing for each program.

How to control source mapping is presented in more detail in later sections of this training.

That is the end of this section, an overview of Fault Analyzer Files.



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_FAv12s02Files.ppt

This module is also available in PDF format at: ../FAv12s02Files.pdf

14 IBM Fault Analyzer for z/OS - V12 Tutorial © 2012 IBM Corporation

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

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, 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.

© Copyright International Business Machines Corporation 2012. All rights reserved.