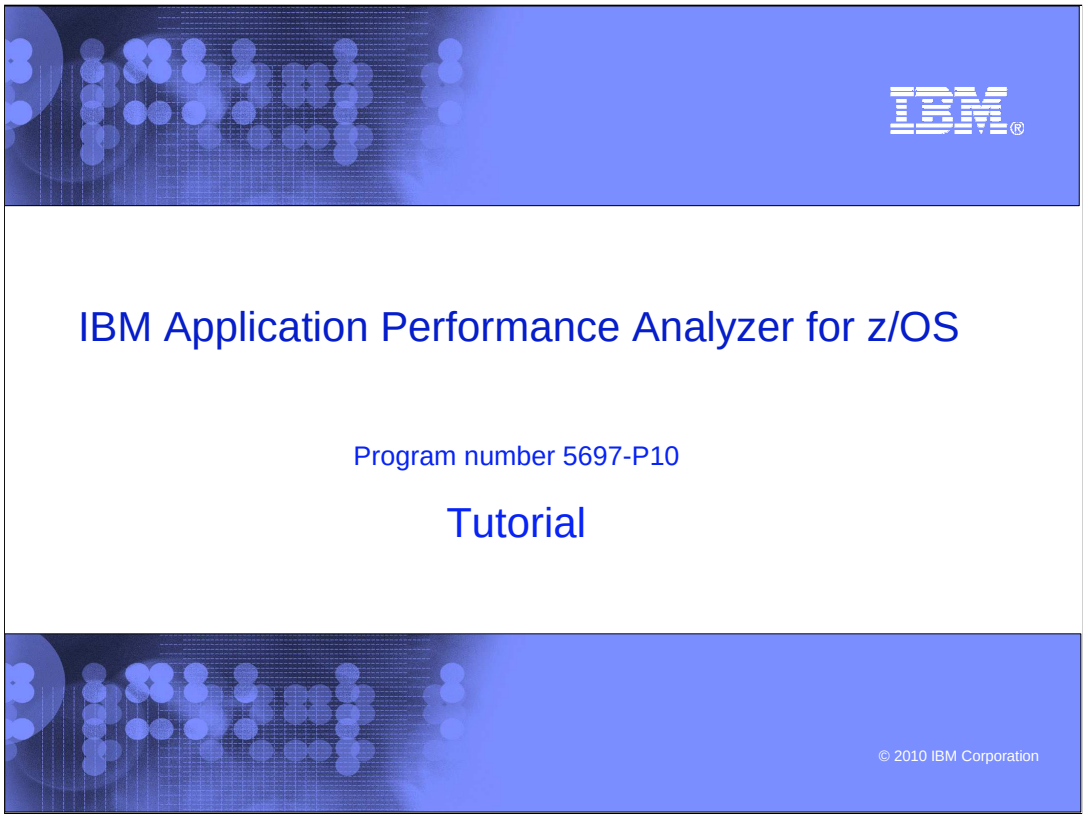
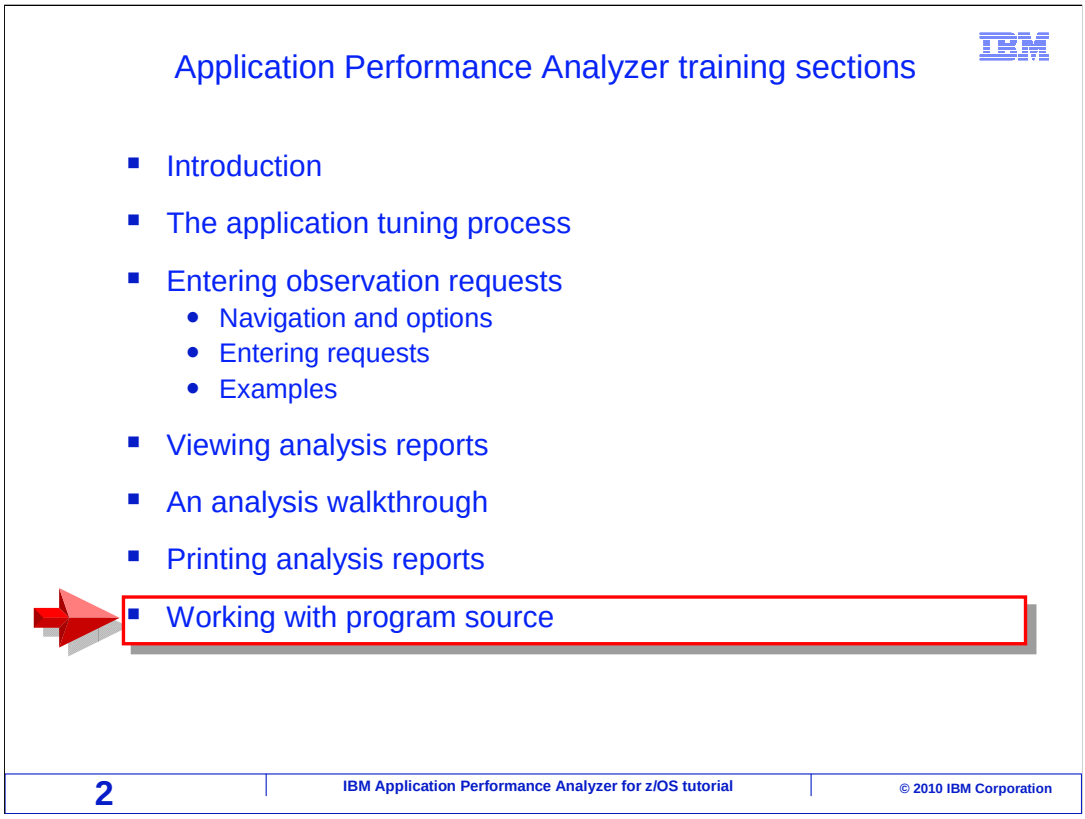


This is the tutorial for IBM's Application Performance Analyzer for z/OS®, one of the IBM zSeries® problem determination tools.

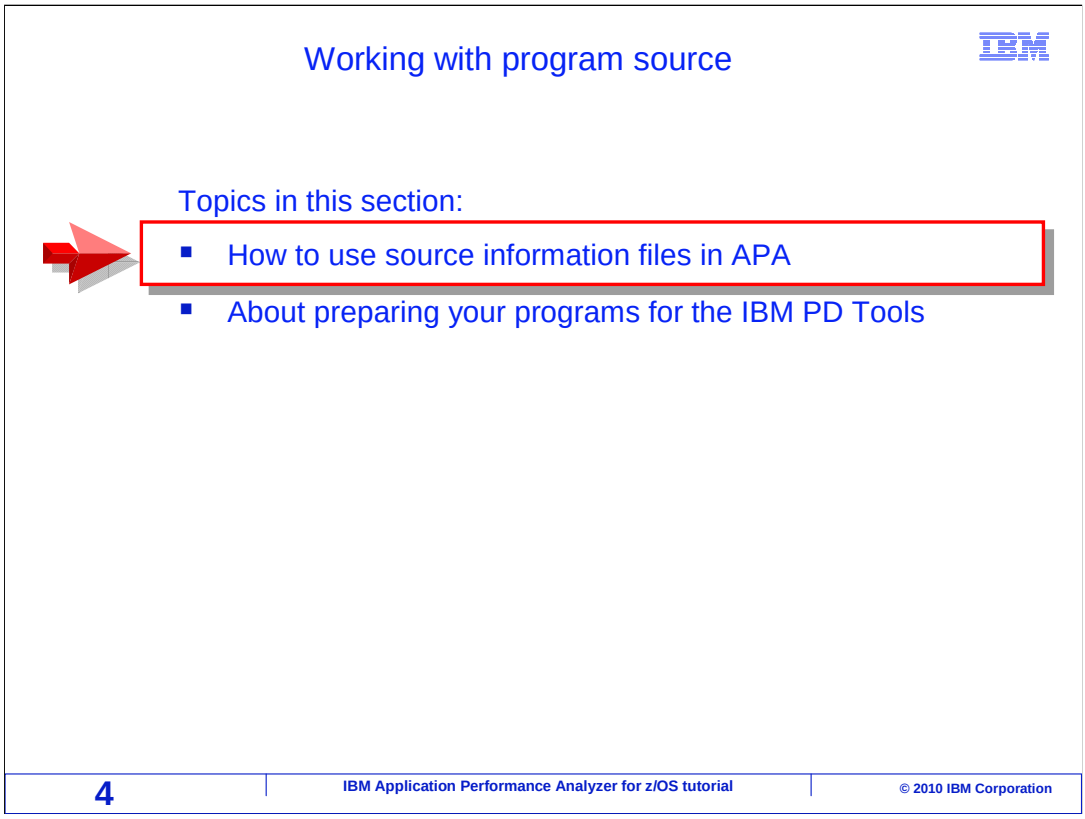


In this section “Working with program source” you will go through an overview of how to set up your programs and save source information files that can be used with the PD tools products. This includes APA, Fault Analyzer and Debug Tool.

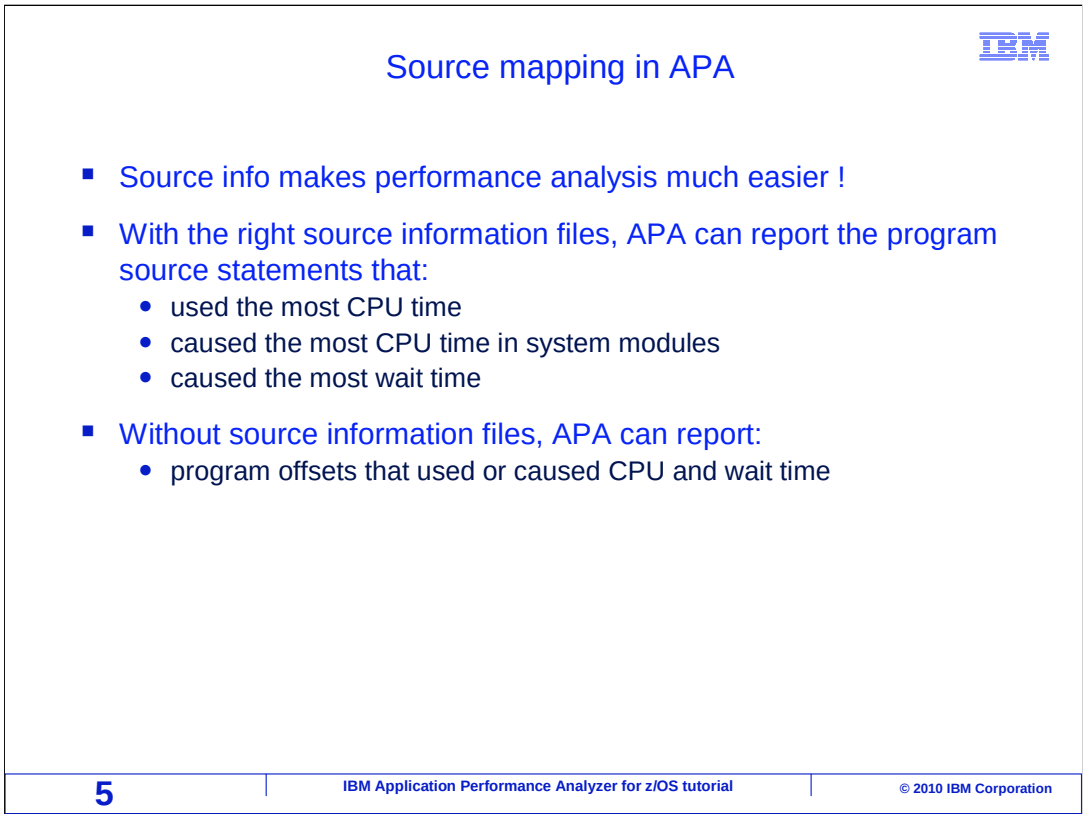
Working with program source

- Preparing programs and saving source information
  - For details about compiling programs for use with IBM PD Tools, see the Application Performance Analyzer for z/OS User's Guide.
  - Details are in Chapter 9: Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products.

See the Application Performance Analyzer for z/OS User's Guide for details about compiling programs for use with IBM PD Tools products,. Details can be found in Chapter nine: Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products.



The first topic “how to use source information files in APA” will explore APA’s source mapping function.



APA captures and reports performance information about an application. And it can do it whether source information for the application is available or not. However, if source information is available for programs, your performance analysis is much easier.

With a source information file, APA can perform source mapping. That saves you time, because it can pinpoint the program source statements that caused the most CPU and wait time. It can even identify the statements that caused system modules to be ran that used a lot of time.

Without a source information file, APA can report which programs and CSECTs are causing the most CPU and wait time. And, it can report the offsets of the machine instructions within programs. Given that, you can manually map the offsets to source statements, if you have a compiler listing with a storage map, although that can be a time-consuming and error-prone process.

Enabling source program mapping

- To use APA's Source Program Mapping functions, you must save program source information
- Depending on the compiler, the Source Info File can be:
  - a Compiler Listing
  - a SYSDEBUG file
  - CAZLANGX file
  - an ADATA file
- Save the Source information in a PDS
  - The member name must be the same as the program name

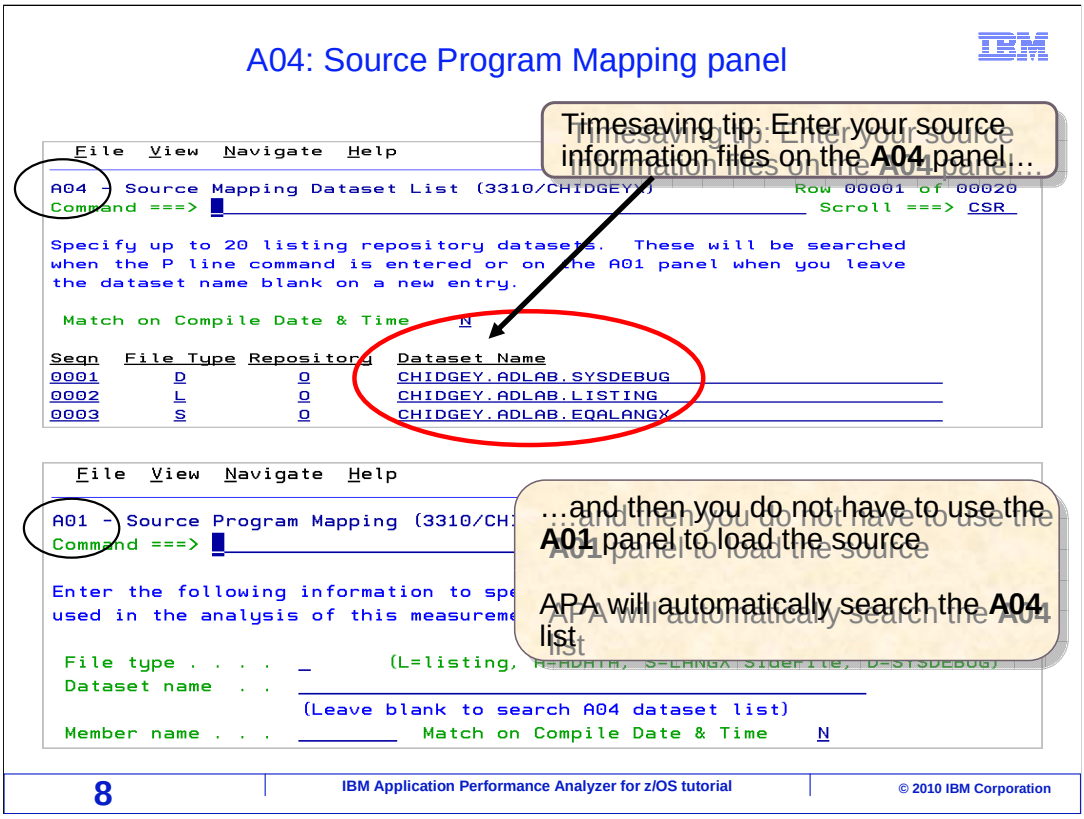
To use APA’s source mapping functions, you have to save program source information when your program is compiled. APA has the capability of reading source information files in several different formats. Each compiler can generate different types of source information. To determine which type of file you should use, you should research your options for each compiler. If someone at your organization has already set up your compile processes to produce the right source information files for APA, then you can just use your normal compilation process. And the right files are generated for you. However, if it is your responsibility to update the compile processes, then you should research the capabilities of each compiler individually. Review the document titled “Preparing programs for PD Tools”, which is available on the IBM Education Assistant website. It is best to have your source information files saved as PDS members, where the member name is the same as the program name.

Source information files

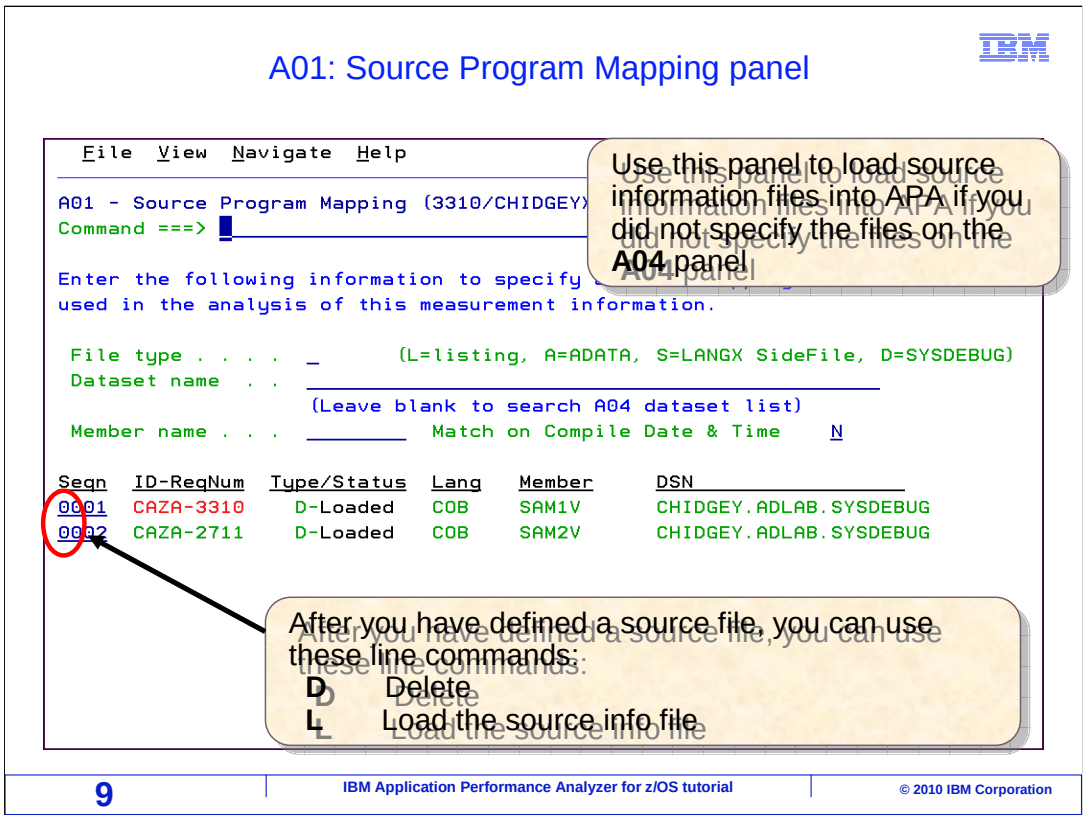
APA can read source information in these formats:

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				✓

For the LE COBOL compilers, and that includes the enterprise COBOL compiler, APA can read sysdebug files, compiler listings, and LANGX files. For the older COBOL compilers, including COBOL II and OS/VS COBOL, it can use either compiler listings or Langx files. For Enterprise PLI and PLI for MVS and VM, it can read Langx files. For OS PLI and C and C++, either compiler listings or Langx files can be used. And for the assembler, APA can use SYSADATA files.

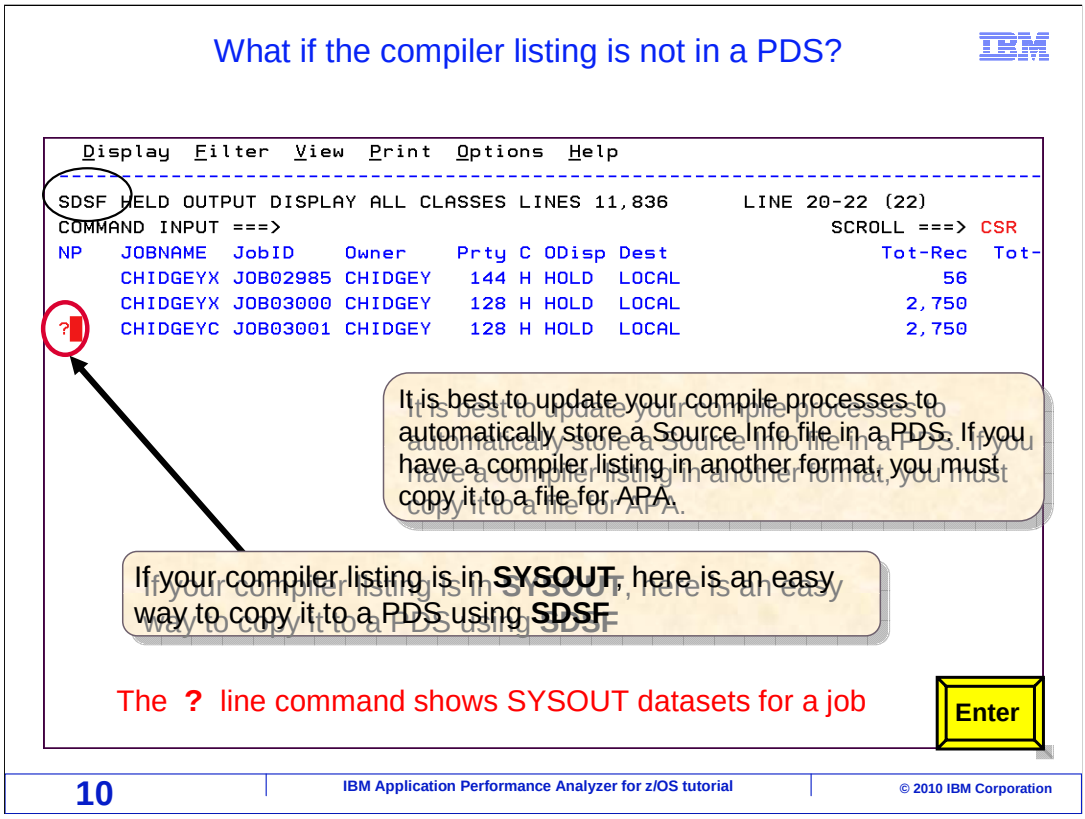


Here is a time-saving tip: If you navigate to the A04 panel, which is the Source Mapping Dataset list panel, you can specify the dataset names of your source information file libraries. If you do this, you do not need to specify the dataset name of the source information library on the A01 panel. APA will automatically search the list for your source.



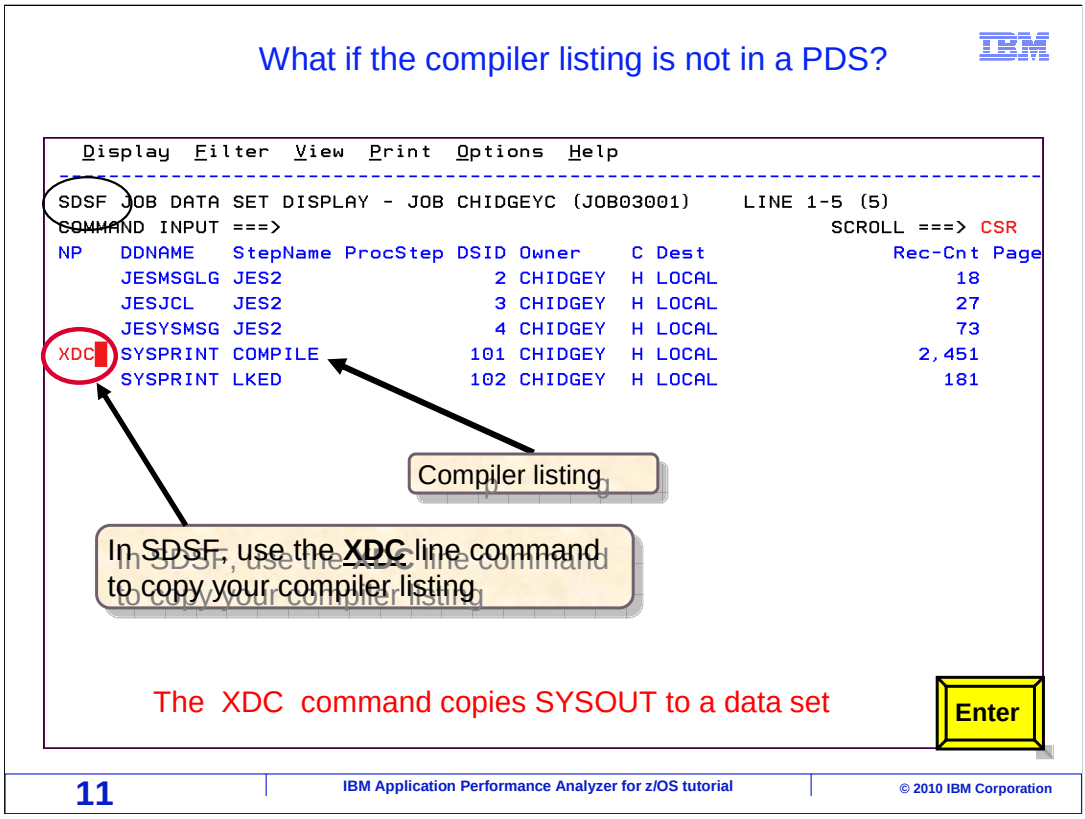
Once your compiler processes have been updated to save the right source information files, APA can use them. Here is how to do it. While you are viewing reports for an observation session, navigate to the A01 panel. On the A01 panel, you specify the location of your source information files. You only need to do this if you did not specify the location of your source information files on the A04:Source Mapping Dataset List panel.

In any case, once the location of your source information files have been indicated using the A04 panel or the A01 panel, use a “P” (for program) line command next to a CSECT entry to display source mapping.

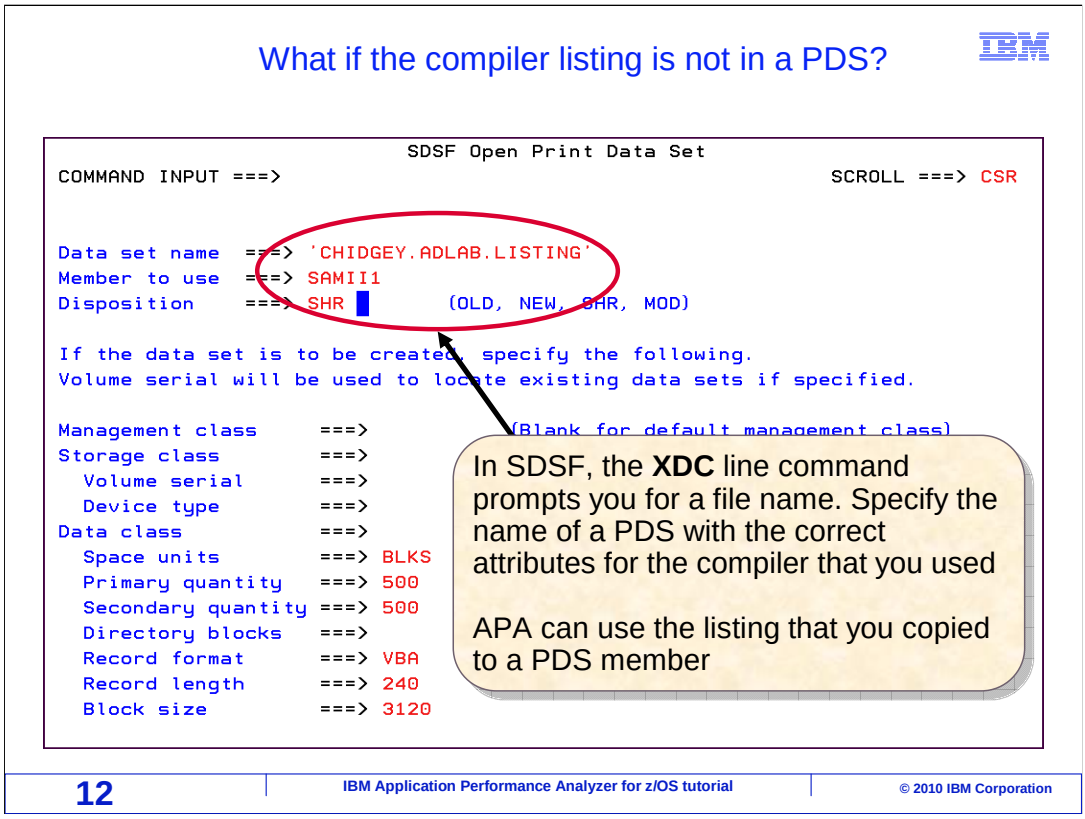


For some compilers, APA can use compiler listings that are stored in PDSes. And the question sometimes arises, can you use a compiler listing that is in the SYSOUT queue? Before you can use it with APA, you need to copy it to a PDS or sequential file. Most SYSOUT viewers give you an easy way to do that.

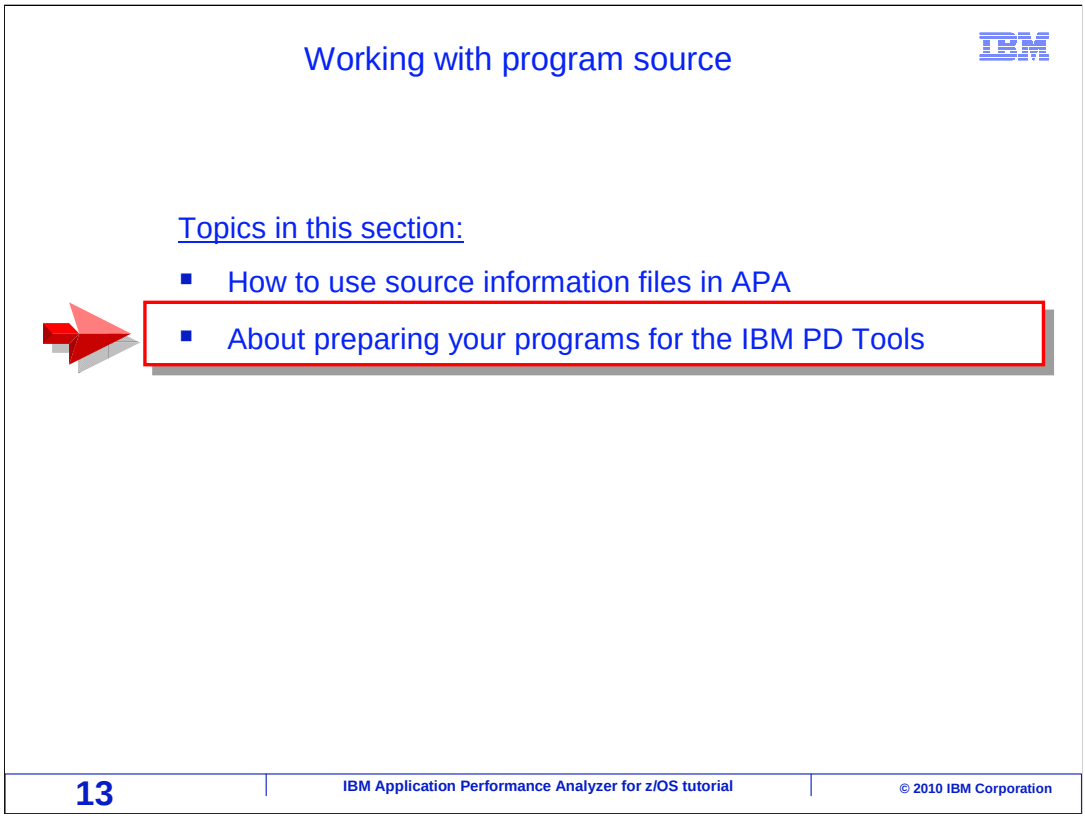
In this example, there is a compiler listing in SYSOUT. SDSF is also being used to work with the SYSOUT. You might have a different SYSOUT viewer that works differently. In SDSF, start by using a “?” line command next to the job, and press Enter.



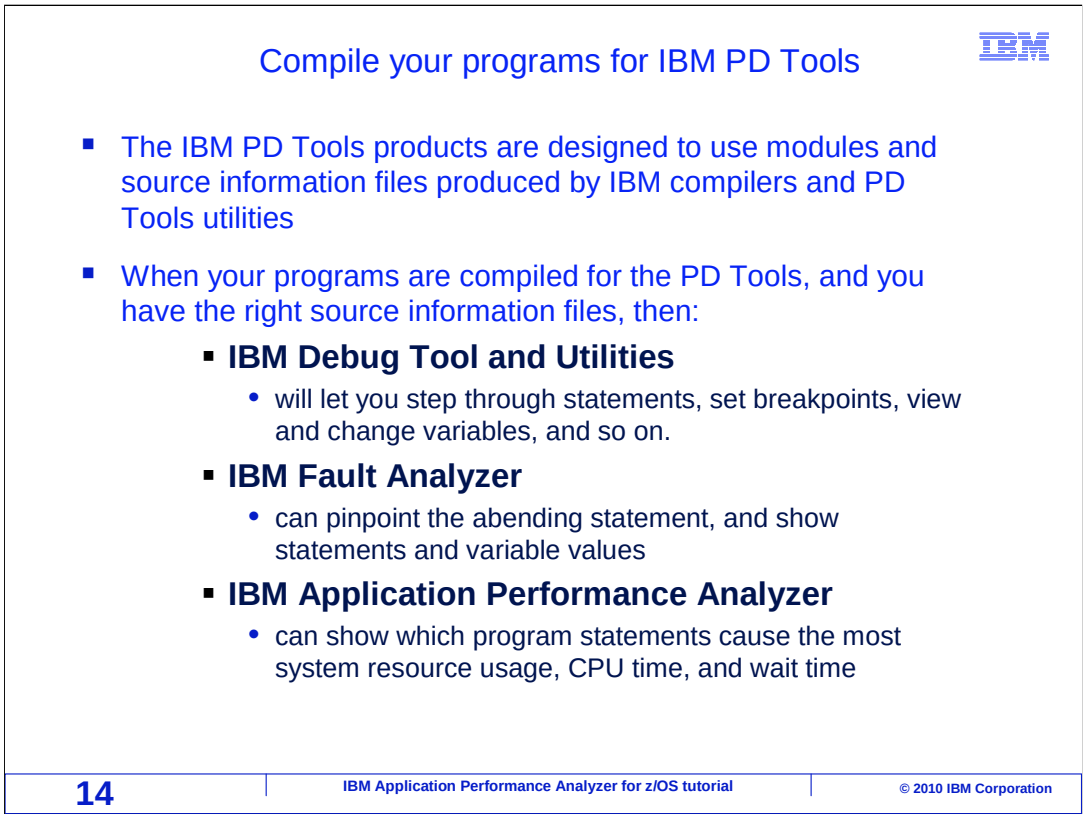
That will display a list of DDs generated by the job. In this case, the compiler listing is in the SYSPRINT DD from the compiler step. From here, type in an “XDC” line command. Press Enter.



That brings up this panel, where you can enter the name of a PDS member. When you hit Enter, the listing is copied into the file. After that, the listing is available for use with APA. It is important to note that the output file must have the correct file attributes for a compiler listing. Different compilers require different attributes for their compiler listing files.



That is the end of the topic titled “How to use source information files in APA”. Next, the topic entitled “About preparing your programs for IBM PD tools” is covered. This topic is an overview of how to compile and assemble your application programs for use with the IBM PD tools products.



The IBM PD tools products are designed to use modules and source information files that are produced by IBM compilers and utilities supplied by the PD tools products.

A very important part of installing any of the IBM PD tools products is updating your compilation processes. You do this to ensure that the modules produced by your compiler are compatible with the tools. And, that your compilers generate and save source information files that are needed by the tools.

For example, when your programs are compiled correctly and the correct source information files are saved. Then the IBM debug tool and utilities product will let you debug your programs by stepping through statements, setting breakpoints, viewing and changing variables, and controlling the execution of your programs.

When your compilation processes are correct, fault analyzer can help you research program abends by pinpointing the source statement that abended, and displaying program variable definitions and values.

Application performance analyzer can help you locate application performance inefficiencies by showing you exactly which application program statements cause the most system resource usage, such as CPU time and wait time. Of course, APA also depends on the compile processes being set up correctly to generate modules with the right options and save source information files in the right formats.

How do you prepare your programs for the IBM PD Tools?

1. For each compiler you use, research the options and methods available, and decide:
   - Which formats are best to store source information files
   - Which compiler options are needed
2. Change compile processes / JCL to:
   - Specify the right compiler options
   - Save source information files
   - For Debug Tool, for some compilers/environments, link-edit a DT start-up exit into the application load module
3. Change program promotion processes / JCL to:
   - Promote source information files along with load modules
   - Can be either a copy or recompile

If you are the person responsible for updating your organization's compilation processes, then you need the answer to this question: "How do you prepare your programs for the IBM PD tools products?". The most important thing to keep in mind as you are researching how to code your compile JCL is that each of the IBM compilers will need different options. And, each can produce different kinds of source information files.

Research how to set up compile JCL for each compiler separately. Do not assume that there is one compile process that will work universally for all of the different compilers. The reason is that they might all require unique changes, and it might be optimal to save source information files in different formats for each compiler.

First, review the options described in the document entitled "preparing programs for PD tools", and decide which types of source information files is to be saved, and which compiler options are needed.

Next, modify your compiler processes and JCL to specify the correct compiler options, save source information files, and possibly for debug tool, link-edit a debug tool start-up exit into the application load module.

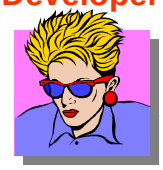
The task of updating the compilation processes is not complete until you also perform the third step, which is to update the program promotion processes to promote the source information files along with the program load modules.

Who makes the changes to the program compile and promotion processes?

Systems Programmer?

Library Manager?

Application Developer?



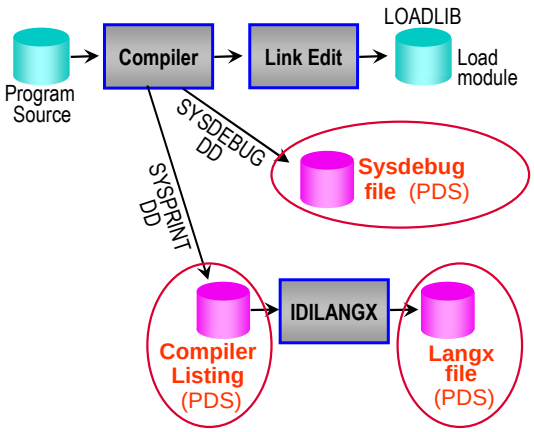
- The owner of the processes or JCL typically makes the changes
  - In many organizations, a person or group is responsible for program source management and program build processes
  - In some organizations, JCL is owned by each developer

In different organizations, different people might have the responsibility of updating compiler processes. In many organizations, an individual or department is responsible for program source management and the program compilation processes. In those organizations, that person or department is responsible for making the updates needed to enable the IBM PD tools. In other organizations, each individual application developer maintains their own compilation JCL. If you are a developer in an organization like that, then it is your responsibility to update your compilation JCL.

At all events, someone should have the responsibility of updating the compilation JCL and processes. The installation of the IBM PD tools products is not complete until the program compilation and promotion processes have been updated for them.

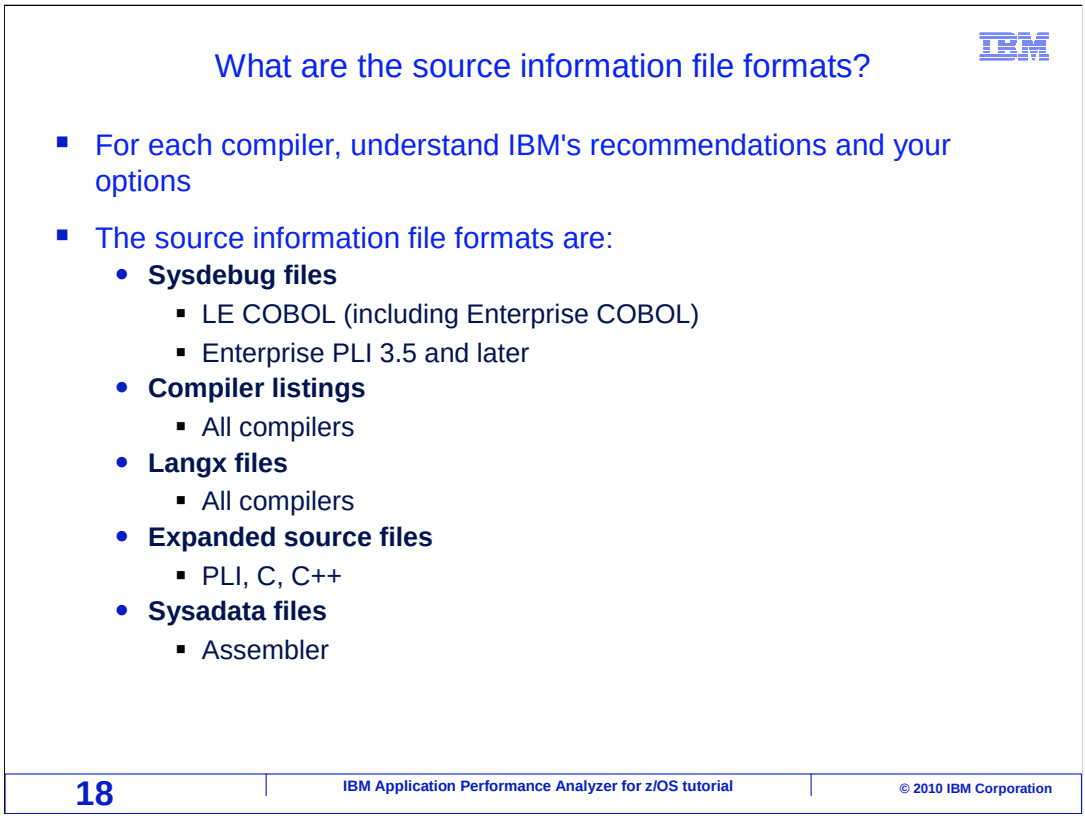
Which source information format should you use?

- Each compiler can generate different kinds of source information files
- Each PD Tools product can support different file formats for each compiler
- Select the best format for your combination of compilers and PD Tools products
- **It *might* be optimal to store in different file formats for different compilers**



If you plan to update compiler JCL for PD tools products, perhaps the first question you will have is “Which source information formats should you use?”

Remember that each compiler takes different options and has different capabilities in terms of the types of source information files that it can generate. Also, each PD tool product can support different types of source information files for each compiler. It is important to research your options, and select the best format of source information files for each of your compilers and for your combination of PD tools products. Keep in mind that it might be optimal for you to save more than one kind of source information file.

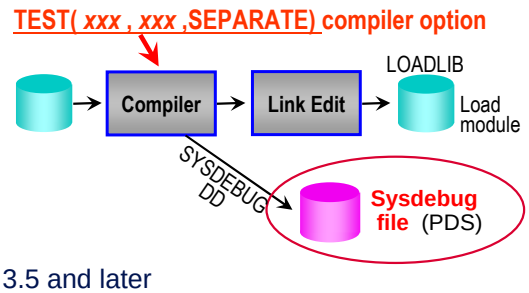


That brings you to the next question. “What kinds of source information files are there?”. To decide which types of source information files you should use, you should read the section for each compiler in the document titled “Preparing programs for PD tools”. It gives recommendations for each compiler, and examples of JCL.

There are five types of source information files used by the PD tools products. SYSDEBUG files, Compiler listings, LANGX files, expanded source files, and SYSADATA files.

Sysdebug file is a source information file

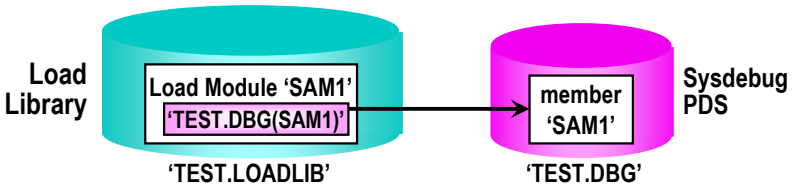
- A Sysdebug file is a special file format that contains program source and symbolics information
- It is a newer format available only in recent compilers:
  - LE COBOL compilers
  - Enterprise PLI compiler version 3.5 and later
- The compiler generates a sysdebug file when you specify the TEST compiler option with SEPARATE. Examples:
  - COBOL: TEST(NONE,SYM,SEPARATE)
  - PLI: TEST(ALL,NOHOOK,SYM,SEPARATE)
- File format:
  - PDSE or PDS
  - LRECL=(80 to 1024),RECFM=FB,BLKSIZE=(any multiple of lrecl)



The first type of source information files to be covered are the SYSDEBUG files. A SYSDEBUG file is a special file format that contains program source and symbolics information. It is a newer file format that is available only in recent compilers, the LE COBOL compilers including enterprise COBOL, and enterprise PLI version 3.5 and later. SYSDEBUG files are produced by a compiler when you specify the TEST compiler option with the separate sub-option. They are stored in a PDS or PDSE, into fixed block records, and the record length can be anything you choose in the range of 80 to 1024.

Advantages of Sysdebug files

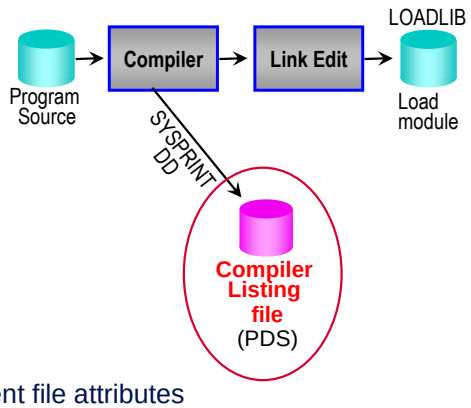
- For COBOL programs, the load module can be production-ready
  - COBOL can generate a module that can be debugged with no added run-time overhead
- Sysdebug files are smaller than compiler listings (less disk space)
- You can produce a listing from a sysdebug file with the Fault Analyzer IDILANGP utility or Debug Tool EQALANGP or APA CAZLANGP
  - The listing is similar to a compiler listing
- A Sysdebug file can be located automatically
  - The compiler stores the name of the Sysdebug file in the load module, so IBM PD Tools know where to find it



SYSDEBUG files are a newer format, and they have some advantages. For COBOL programs, the load module produced by the compiler can be production ready. That is because the SYSDEBUG file contains nearly all of the information that the PD tools require, and the application load module does not have any added run-time overhead. Also, SYSDEBUG files are smaller than compiler listings, so it takes less disk space to save them. Another advantage is that you can produce a printable listing from a SYSDEBUG file, using utilities that are supplied in fault analyzer, debug tool, and APA. The listing produced is similar to a compiler listing, so you might decide that you do not need to save your compiler listing. Another advantage of SYSDEBUG files is that the PD tools products might be able to locate them automatically. That is because the name of the SYSDEBUG file is actually stored right in the load module. So for example, fault analyzer can locate the SYSDEBUG file while it performs its real-time analysis, and perform source mapping, automatically.

Compiler listing is a source information file

- A Compiler listing is the report produced by a compiler
- Instead of writing it directly to SYSOUT, save it in a file
- Specify compiler options that the PD Tools require
  - Different for each compiler
- File format:
  - PDSE or PDS
  - Each compiler might require different file attributes
- It is a best practice to save compiler listings, in some form, for all of your production programs



The next type of source information file to be covered is a compiler listing file. This is the listing that is produced by the compiler, the same listing that you read to review compiler messages, and so on. In your compiler JCL, you point the listing to a file instead of writing it directly to SYSOUT. In order to use a compiler listing, you will need the right compiler options turned on so that the information that the PD tools need is in the compiler listing. Each compiler will have different options requirements. You should store your compiler listings in a PDS or PDSE, and the member name must be the same as the program name. Note that it is a best practice to save compiler listings, in some form, for all of your production programs. If you do not, then you do not have any documentation for the programs that you have running in production. So even if you do not plan to use compiler listings with PD tools, it is still important to save your compiler listing or to be able to re-create it from a SYSDEBUG file or LANGX file.

More information on compiler listings

- If you still have the original compiler listing for an existing load module, can you use it?
  - In many cases, FA and APA can use your existing listings
    - They require the right compiler options, so that needed information is in the listings
    - Check the required options for each compiler
  - For Debug Tool, probably not
    - For most languages, Debug Tool requires the TEST compiler parm, which prepares the load module for debugging
- Can you re-create a compiler listing for an existing load module by recompiling the program?
  - You can for Fault Analyzer, if:
    - The source code and compiler have not changed
  - Not for Debug Tool
    - It requires an exact time stamp match (listing / module)

You might be in a situation where one or more of the PD tools products has just recently been installed on your system. If so, you might have the question: “If you still have the original compiler listing for an existing load module, can you still use it with the PD tools now?”. You also compiled the load module before the compiler processes were modified for the PD tools. And the answer is, it depends. In many cases, fault analyzer and APA is able to use your old compiler listings. However, the right set of compiler options are needed. So, if your old listings were originally generated with the right options, then you can use them now in fault analyzer and APA.

For debug tool, probably not. For most languages, debug tool requires the TEST compiler parameter, and it is unlikely that the test parm was used when your program was originally compiled.

Another important question has to do with the ability to re-create a fresh compiler listing, and use the new listing with the PD tools products now, without the need to replace the original load module. You can do that for fault analyzer and APA.

Also, it is a very good approach because you do not have to replace all of your existing production load modules, which, of course can be a risky proposition. However, that will not work for debug tool. Debug tool requires an exact timestamp match between the source information file and the load module. Typically, however, that is not a problem because most debugging occurs in the test environment, and you can regenerate a module from the source if you need to interactively debug it.

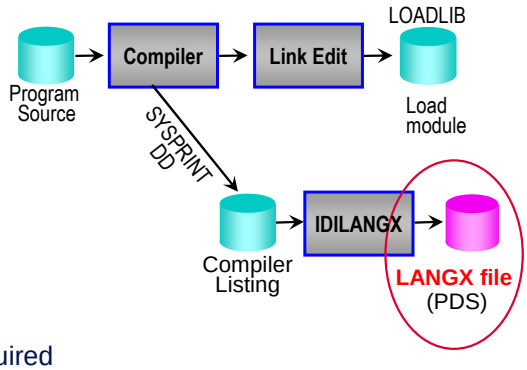
Compiler listing data set attributes

- To use compiler listings with PD Tools, store them in a PDS or PDSE (member name = program name)
- COBOL SYSPRINT (except OS/VS COBOL):
  - RECFM=FBA,LRECL=133
- OS/VS COBOL SYSPRINT:
  - RECFM=FBA,LRECL=121
- PLI SYSPRINT:
  - RECFM=VBA,LRECL=125 minimum
- Enterprise PLI SYSPRINT:
  - RECFM=VBA,LRECL=137 minimum
- C or C++ SYSPRINT
  - RECFM=VB or VBA,LRECL=137 minimum, or
  - RECFM=FB or FBA,LRECL=133

It is important to store compiler listings in files with the correct attributes. The correct file attributes for compiler listings generated by each compiler are indicated on this slide. If you do not store your listings and files with the correct attributes, the PD tools might not be able to use them.

Langx file is a source information file

- A Langx file (or "Side File") is a special file format that is produced by one of the PD Tools xxxLANGX utility programs
- They are an alternative to storing compiler listings
- Equivalent utilities are shipped with:
  - Fault Analyzer: IDILANGX
  - Debug Tool: EQALANGX
  - APA: CAZLANGX
- The utility reads a compiler listing or assembler sysadata file
  - Certain compiler options are required
  - Different options for each compiler
- File format:
  - PDSE or PDS,LRECL=1562,RECFM=VB,BLKSIZE=(1566 to 32K)



The next type of source information file to be covered is the Langx file. The Langx file is a special file format that is produced by one of the PD tools Langx utility programs. Langx files are also called side files. They are an alternative to storing compiler listings. Note that equivalent utilities to generate Langx files are shipped with fault analyzer, debug tool, and APA.

To create a Langx file, the utility reads a compiler listing or assembler SYSADATA file, and outputs a Langx file. In order to process the compiler listing, certain compiler options are required to ensure that the needed data is in the listing. Langx files are stored in PDSes or PDSEs, and have a variable blocked record format with a minimum record length of 1562.

More information about Langx files

- When should you use a Langx file?
  - For some languages, this file type is supported by more of the PD Tools products (DT, FA, APA) than any other file type.
    - In these cases, if you store a Langx file, you might not need to store other types of source information files

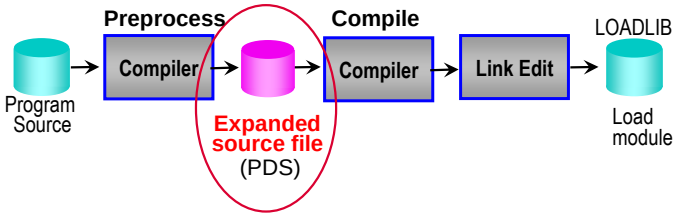
    - or -
  - For some languages, you can store Langx files instead of compiler listings
    - Langx files are smaller than compiler listings (less disk space)
    - However, Langx files are comparable in size to Sysdebug files
- You can produce a listing from a Langx file with the Fault Analyzer IDILANGP utility or APA CAZLANGP
  - The listing is similar to a compiler listing
  - However, Langx files are not directly "human readable"

"When should you opt to save a Langx file instead of a listing?" Langx files are supported by more of the PD tools products than other file types for some languages. So if you store a Langx file in these cases, then you might not need to store other types of source information files. Also, for some languages, you can store Langx files instead of compiler listings. You can save on disk space using Langx files since they are typically smaller than compiler listings. In addition, Langx files are comparable in size to SYSDEBUG files.

You can store a Langx file as an alternative to a compiler listing. That is because you can produce a listing from a Langx file using the LANGP utility that is shipped with the PD Tools products. While Langx files themselves are not "human readable", the listing that can be produced from one is similar to a compiler listing.

Expanded program source file is a source information file

- An expanded program source file is program source code that has been expanded (preprocessed) by the compiler
- It contains statements generated from PLI %include, C #include, in-line functions, macro expansions, and so on.
- Advantage: Use it with PLI, C, and C++ compilers to get complete source information in some PD Tools



- File format:
  - PDSE or PDS or sequential
  - any file attributes supported by the compiler

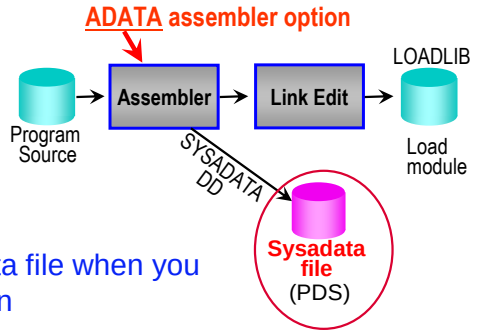
Expanded program source files are yet another type of source information files. It is nothing more than program source that has been expanded, that is preprocessed, by the compiler. It can be used with PLI, C, or C++ compilers, and it contains statements generated from PLI %include and C #include statements, in-line functions, macro expansions, and so on.

The advantage of using a program expanded source file is that it allows you to obtain complete expanded source information in some of the PD tools products.

They are stored in PDSes or PDSEs, in any file format supported as input by the compiler.

Sysadata file is a source information file

- A Sysadata file is a special file format that is produced by the assembler
- It can be processed directly by Fault Analyzer or APA, or used as input to the xxxLANGX utility to create a Langx file



- The assembler generates a sysadata file when you specify the ADATA assembler option
- File format:
  - PDSE or PDS
  - LRECL=(8188 to 32K-4),RECFM=VB,BLKSIZE=(lrecl+4 to 32K)

The last type of source information file to be covered is the Sysadata file. The Sysadata file is a special file format generated at assembly time. It is processed directly by Fault Analyzer or APA, or used as input to a Langx utility to generate a Langx file.

The assembler generates a Sysadata file when you specify the ADATA option. Sysadata files can be stored as members in PDSes or PDSEs. They have a variable blocked record format, with a record length of anywhere from 8188 to 32K-4.

At this point, you have gone through all of the different types of source information files.

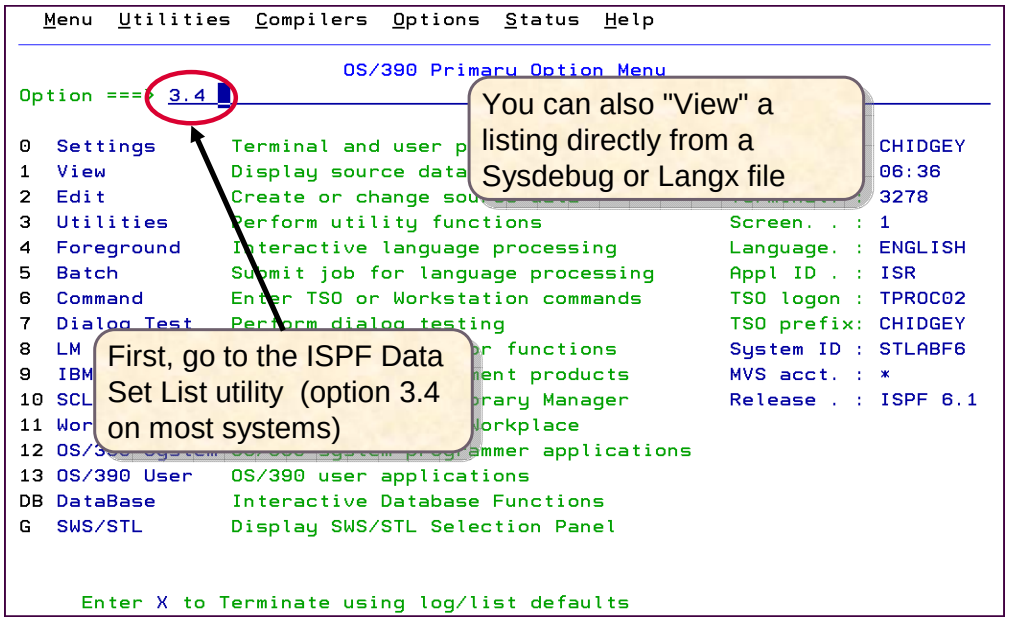
Tip: Print a listing from a Sysdebug or Langx file

- You can print a listing from a Sysdebug or Langx file
  - With the Fault Analyzer IDILANGP utility program
  - Or equivalent Debug Tool utility EQALANGP
- The sample JCL below will print a listing to SYSOUT:

```
EDIT       CHIDGEY.ADLAB.JCL(IDILANGP) - 01.00            Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ******************************
000001 //CHIDGEYC JOB (CHIDGEY,H244,090,CTKA),NOTIFY=CHIDGEY,
000002 //  MSGCLASS=H,TIME=59,REGION=0M RESTART=STEP011
000003 //*********************************************************************
000004 //*   PRINT A LISTING FROM A SYSDEBUG FILE
000005 //*********************************************************************
000006 //STEP1 EXEC PGM=IDILANGP,PARM='CHIDGEY.ADLAB.SYSDEBUG(SAM1)'
000007 //SYSPRINT DD SYSOUT=*
****** **************************** Bottom of Data ****************************
```

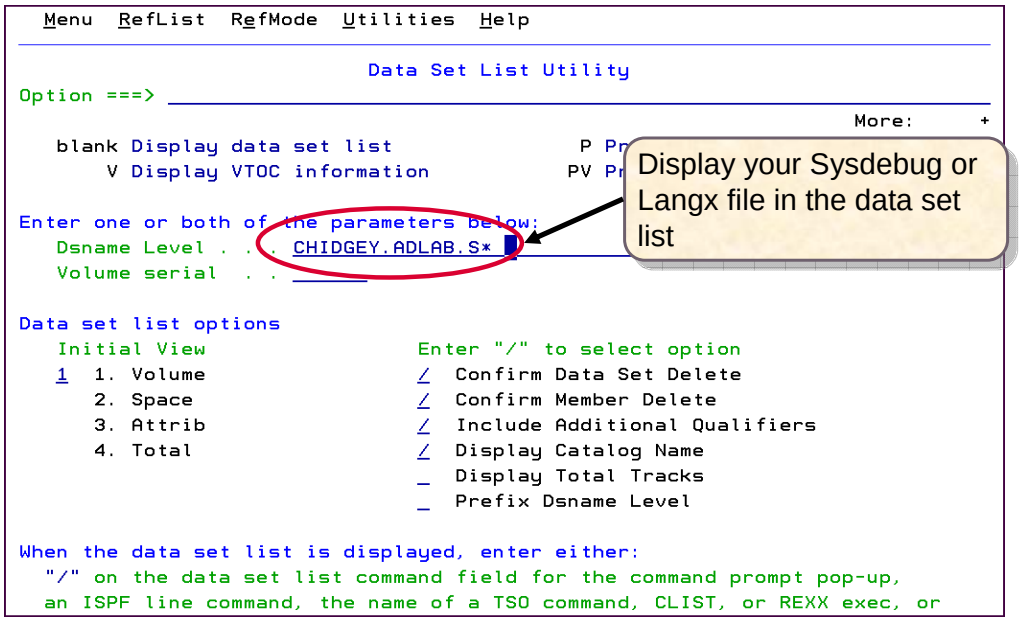

Here is a tip. You can print a listing from a Sysdebug or a Langx file by using one of the Langp utility programs. Here is an example of JCL you can use to print a listing from one of these files.

Tip: View a listing from a Sysdebug or Langx file



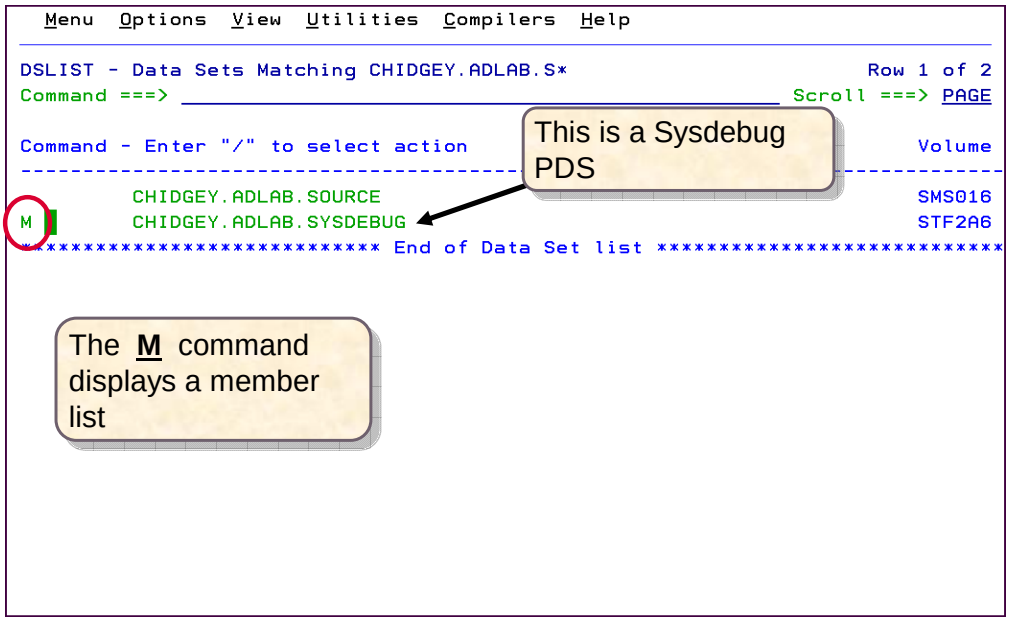
And there is another, even easier, way to view a listing directly from a Sysdebug or Langx file. For example. In ISPF, first go to the 3.4 data set list utility.

Specify the name of your Sysdebug or Langx file



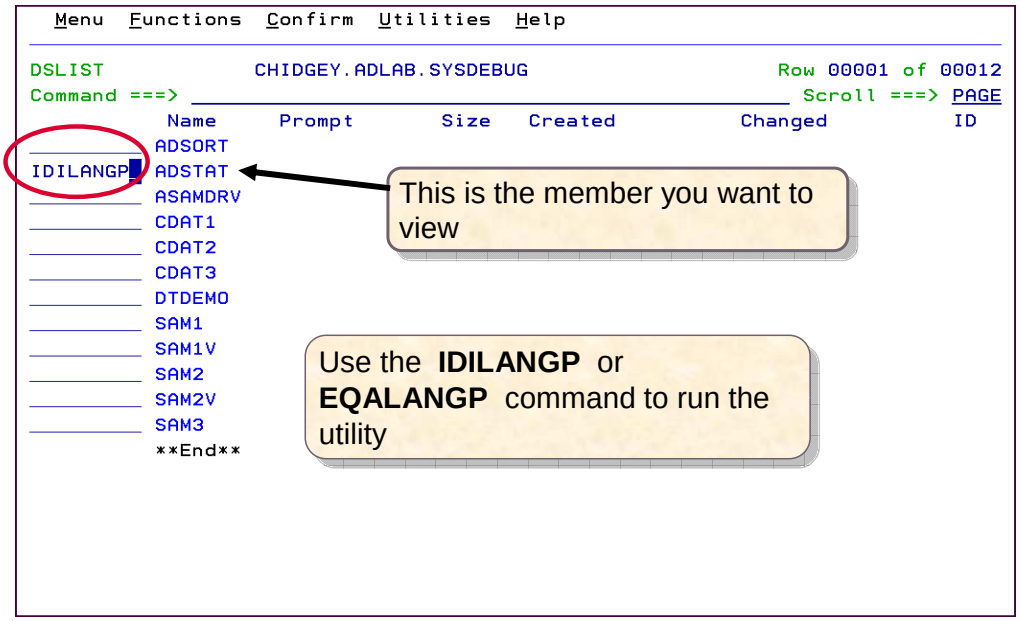
Once in the data set list utility, bring up a list of files that include your Sysdebug PDS library.

Use the “M” line command to display a member list

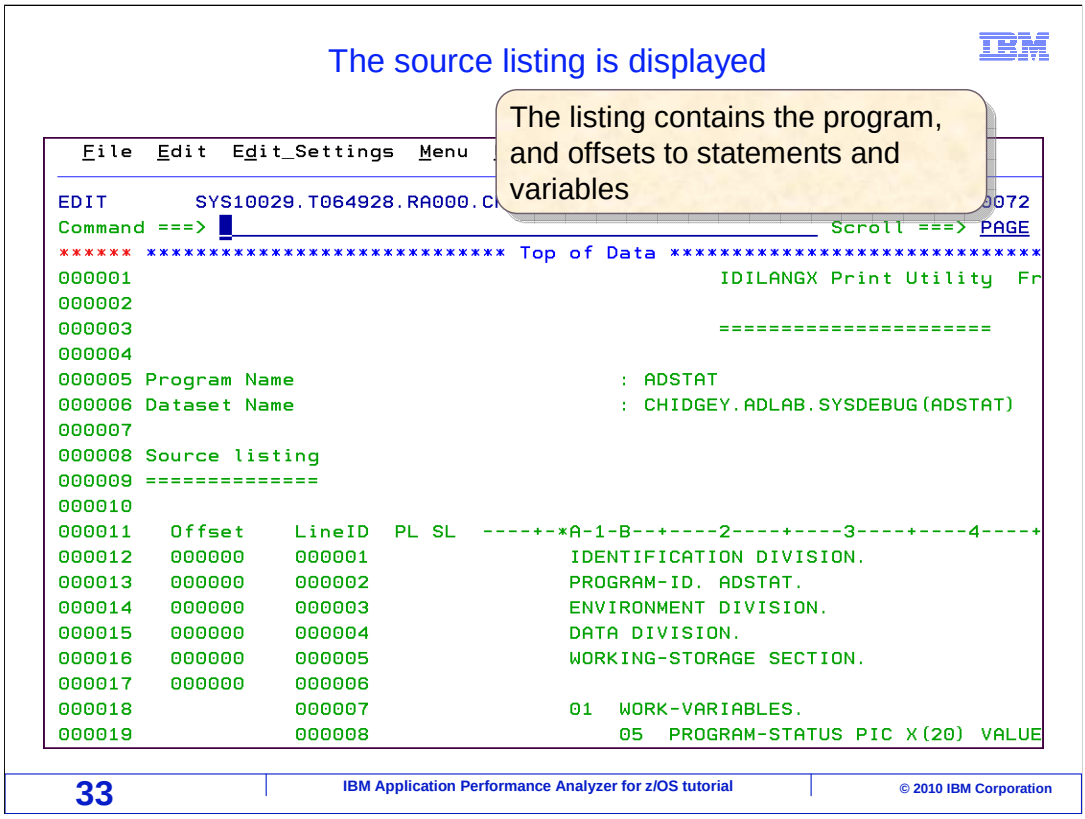


Now a data set list is shown. If you type an M line command next to the file name of your Sysdebug library, you can display a member list.

Use the IDILANGP or EQALANGP utility to get a listing



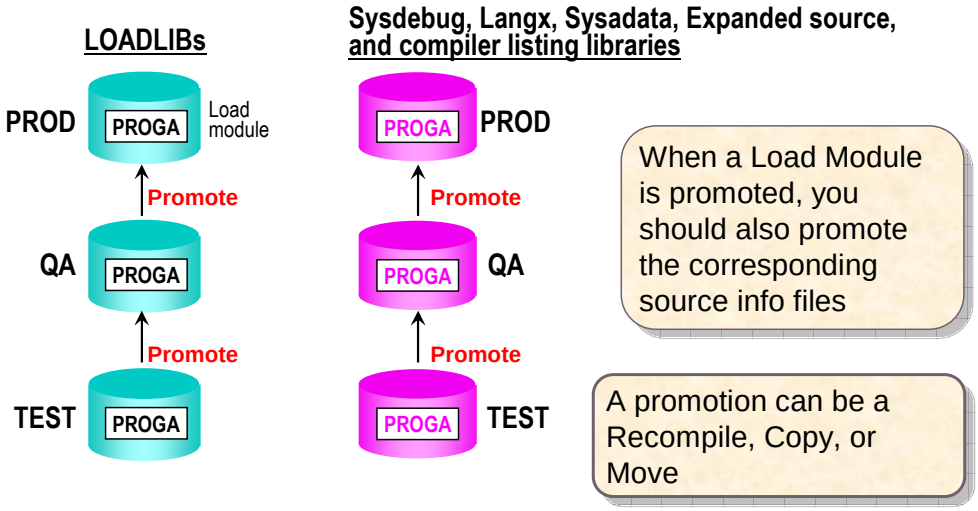
From the member list, you can enter the IDILANGP command next to the member that you want to process. Or, you can optionally use the equivalent APA utility CAZLANGP or the equivalent Debug Tool utility EQALANGP. After you press enter….



That runs the LANGP utility, and displays the listing right on the screen. Again, if you are storing either Sysdebug files or Langx files, then you might not find it necessary to also store the compiler listings. Since you are still able to reproduce similar listings as in this example.

Save source information for PD Tools

- Retain source information throughout the program life cycle
- Update your *promotion* processes to retain source info



The last topic to be covered in this section is the promotion of source information throughout the program life cycle. It is important that source information files are promoted alongside the application load modules.

A common mistake that some organizations make is to forget to promote the source information. Sometimes people will only update their compilation processes, and then the source information files are only available in their test environment. Promote the source information, because as a developer, you are much more productive if you can use the PD tools to work with your programs all the way through their life cycle. For example, suppose the source information is promoted for program all the way through to production. If the production program abends, you are able to use fault analyzer to pinpoint the exact program source statement that caused the abend. And, you are able to use APA to locate program statements that might be using excessive amounts of CPU time.

For every load library that you have, there should be a corresponding source information library or PDS. When a load module is promoted, say from test to QA, or from QA to production, the corresponding source information files should also be promoted. A promotion can be accomplished with recompile, a copy, or a move. You will not have to completely redesign your existing promotion processes to make this work. Whatever treatment is given to load modules during a promotion, just treat the source information files in the same manner.

That way, you are able to take advantage of the PD tools products throughout the entire life cycle of your applications.

That is the end of the section, working with program source information. Remember, the detailed sections, which describe exactly how to set up your compile processes, are covered in the Application Performance Analyzer for z/OS User's Guide. They can be found in Chapter nine: Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products.

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_APAv10s07ProgramSource.ppt

This module is also available in PDF format at: ../APAv10s07ProgramSource.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, 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

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.