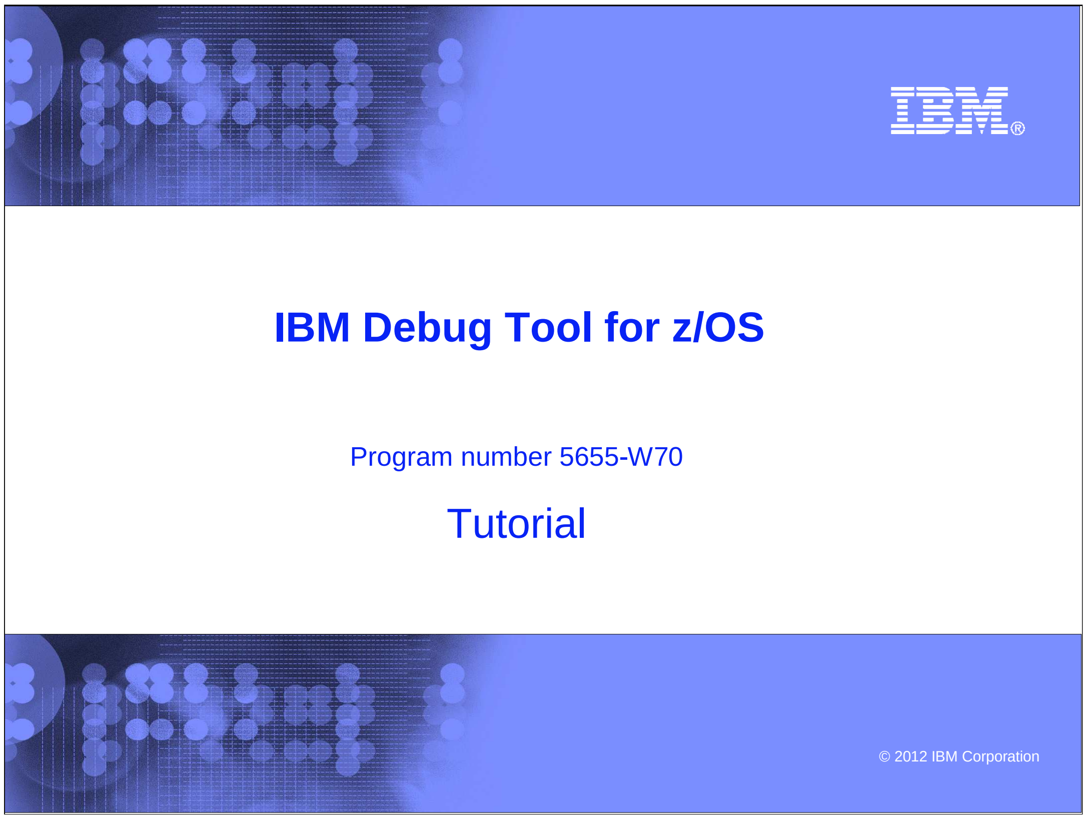
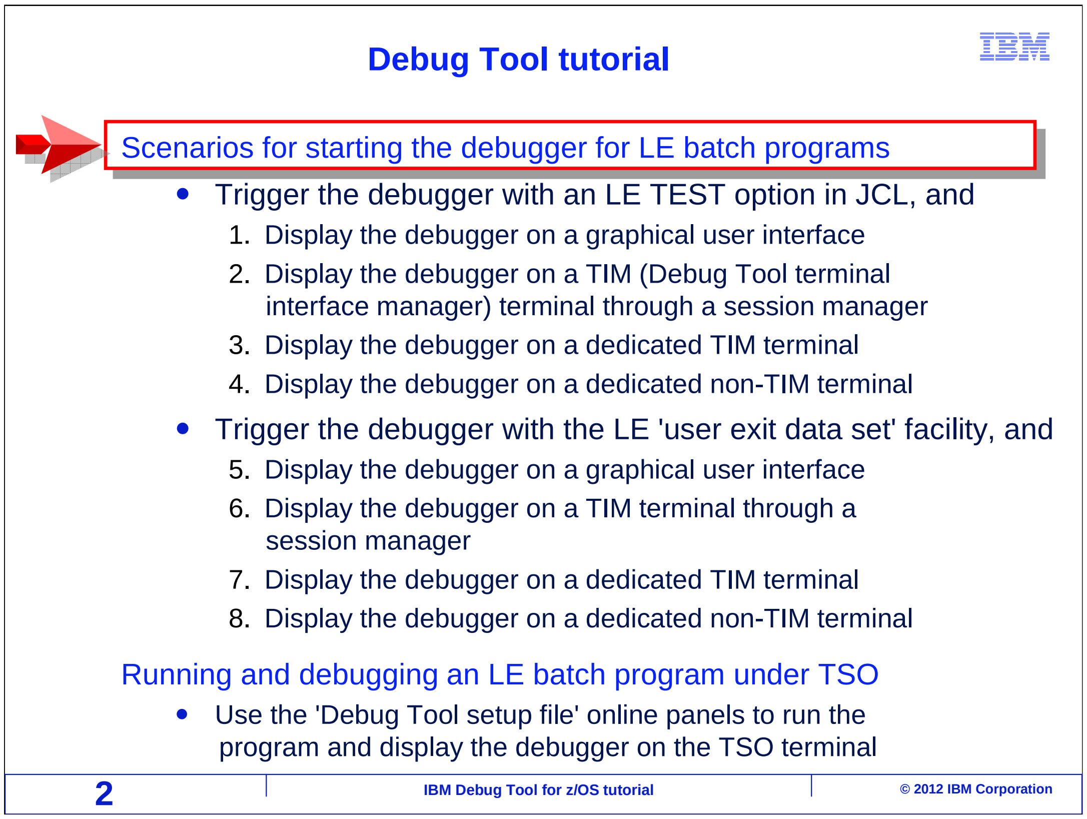


# IBM Debug Tool for z/OS

Program number 5655-W70

Tutorial

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



## Debug Tool tutorial

### Scenarios for starting the debugger for LE batch programs

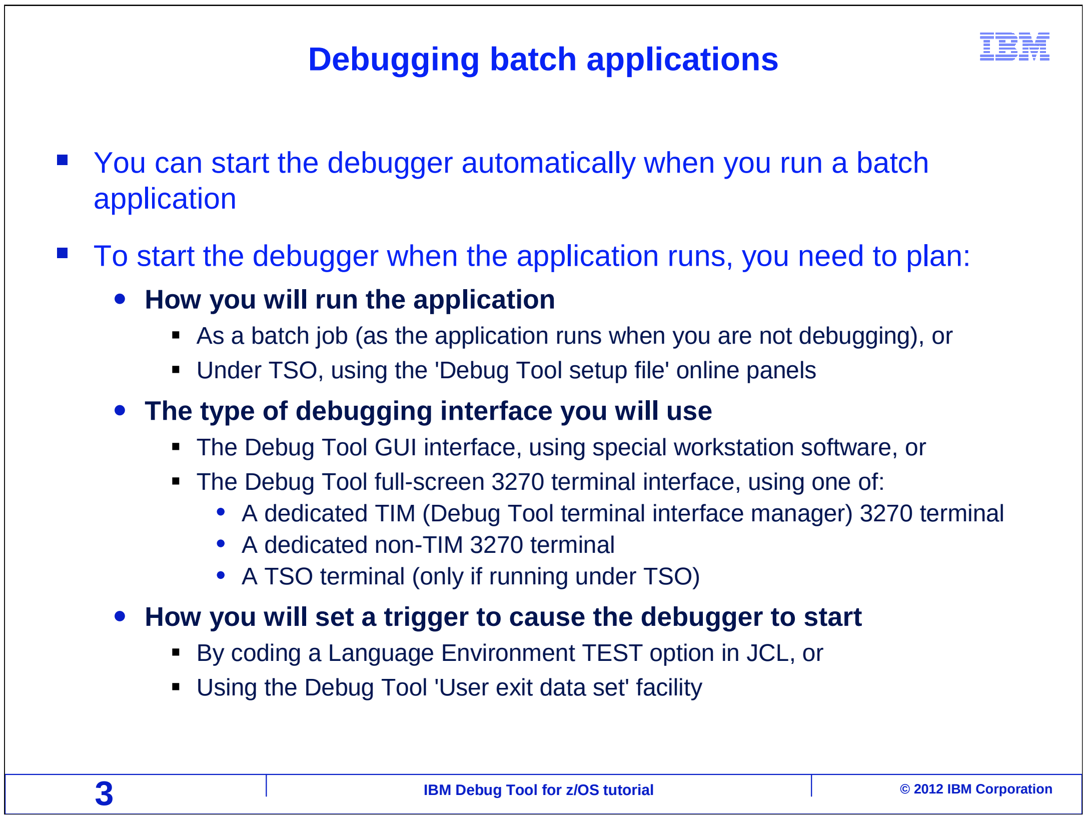
- Trigger the debugger with an LE TEST option in JCL, and
  1. Display the debugger on a graphical user interface
  2. Display the debugger on a TIM (Debug Tool terminal interface manager) terminal through a session manager
  3. Display the debugger on a dedicated TIM terminal
  4. Display the debugger on a dedicated non-TIM terminal
- Trigger the debugger with the LE 'user exit data set' facility, and
  5. Display the debugger on a graphical user interface
  6. Display the debugger on a TIM terminal through a session manager
  7. Display the debugger on a dedicated TIM terminal
  8. Display the debugger on a dedicated non-TIM terminal

### Running and debugging an LE batch program under TSO

- Use the 'Debug Tool setup file' online panels to run the program and display the debugger on the TSO terminal

2 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

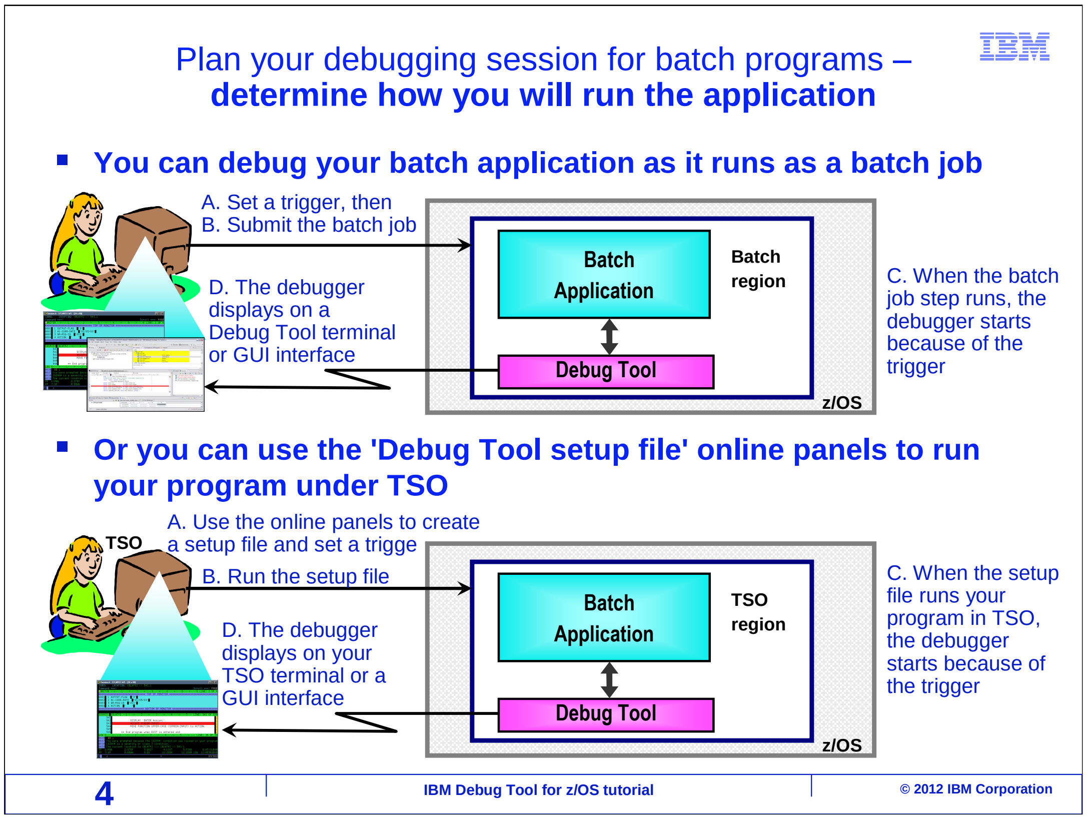
In this section, there will be a discussion of the methods available for you to debug batch programs that are compiled with Language Environment conforming compilers.



## Debugging batch applications

- You can start the debugger automatically when you run a batch application
- To start the debugger when the application runs, you need to plan:
  - **How you will run the application**
    - As a batch job (as the application runs when you are not debugging), or
    - Under TSO, using the 'Debug Tool setup file' online panels
  - **The type of debugging interface you will use**
    - The Debug Tool GUI interface, using special workstation software, or
    - The Debug Tool full-screen 3270 terminal interface, using one of:
      - A dedicated TIM (Debug Tool terminal interface manager) 3270 terminal
      - A dedicated non-TIM 3270 terminal
      - A TSO terminal (only if running under TSO)
  - **How you will set a trigger to cause the debugger to start**
    - By coding a Language Environment TEST option in JCL, or
    - Using the Debug Tool 'User exit data set' facility

A batch application is as a program or programs that run in a batch job on a z/OS system. It runs when JCL (job control language) is submitted on a z/OS system. You can debug batch applications by setting a trigger that causes the debugger to start when the application runs. Debug Tool provides several ways to debug a batch application, so there are some decisions to make. Before you use the debugger, you need to plan how you will run the application, the type of debugging interface you will use, and how you will set a trigger to cause the debugger to start.



## Plan your debugging session for batch programs – **determine how you will run the application**

- **You can debug your batch application as it runs as a batch job**

A. Set a trigger, then
B. Submit the batch job

D. The debugger displays on a Debug Tool terminal or GUI interface

C. When the batch job step runs, the debugger starts because of the trigger

- **Or you can use the 'Debug Tool setup file' online panels to run your program under TSO**

A. Use the online panels to create a setup file and set a trigge
B. Run the setup file

D. The debugger displays on your TSO terminal or a GUI interface

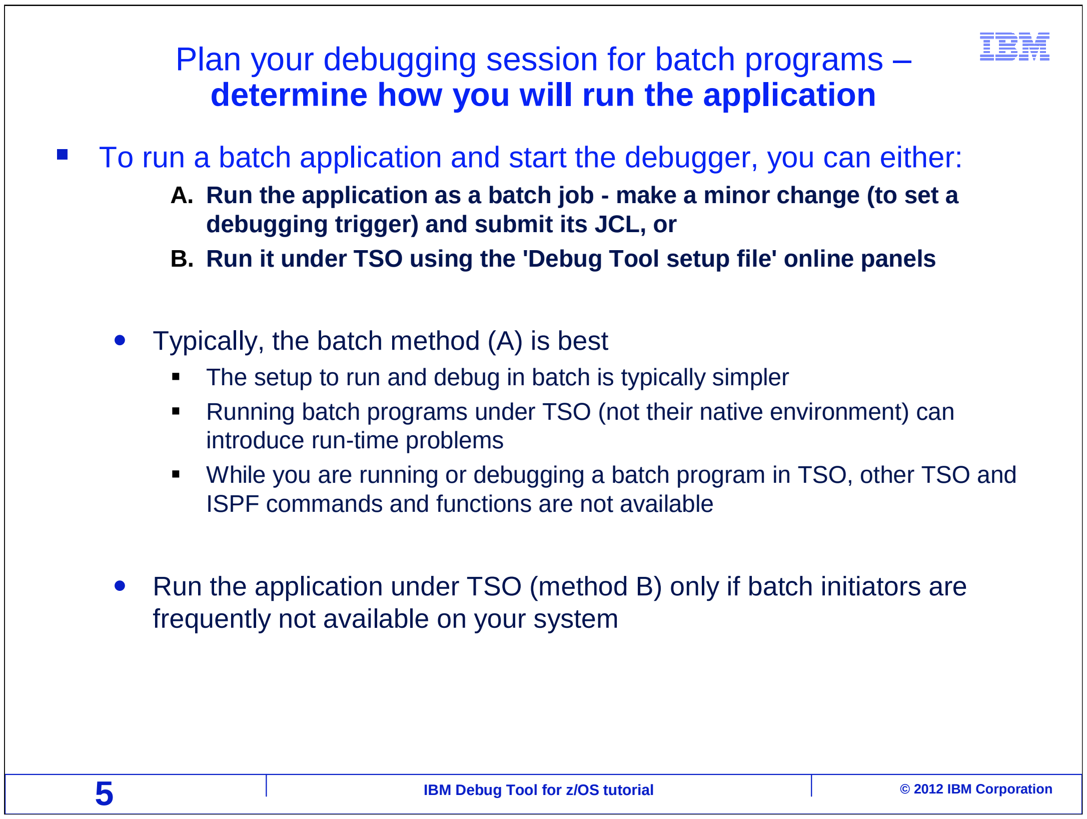
C. When the setup file runs your program in TSO, the debugger starts because of the trigger

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

When you run a typical batch application, it runs as a batch job, but does not trigger the debugger. But you can configure your batch application so that when it runs, the debugger starts automatically. There are two ways you can run your application when you plan to debug it.

First, you can run it normally as a batch job. Start with JCL that you already use to run the application, and make a minor change to the JCL that will cause the debugger to start. This change to the JCL is the debugging trigger. Examples of setting debugging triggers are shown in upcoming sections. Then submit the job. When it runs, the trigger causes the debugger to start, and it displays on either a dedicated Debug Tool terminal session, or on GUI debugging software on your workstation. This method, debugging your batch application while it runs natively in a batch job, is typically the best and simplest approach.

Or, there is another method – run your batch application under TSO. Use the 'Debug Tool setup file' panels in TSO to create something called a 'setup file', which has all the information needed to run your batch program. You can copy JCL into your setup file to expedite the preparation. When it is ready, run the setup file, which runs your program under TSO and triggers the debugger. The debugger displays on your TSO terminal or on GUI debugging software on your workstation.



## Plan your debugging session for batch programs – **determine how you will run the application**

- To run a batch application and start the debugger, you can either:
  - **A. Run the application as a batch job - make a minor change (to set a debugging trigger) and submit its JCL, or**
  - **B. Run it under TSO using the 'Debug Tool setup file' online panels**

  - Typically, the batch method (A) is best
    - The setup to run and debug in batch is typically simpler
    - Running batch programs under TSO (not their native environment) can introduce run-time problems
    - While you are running or debugging a batch program in TSO, other TSO and ISPF commands and functions are not available

  - Run the application under TSO (method B) only if batch initiators are frequently not available on your system

5 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

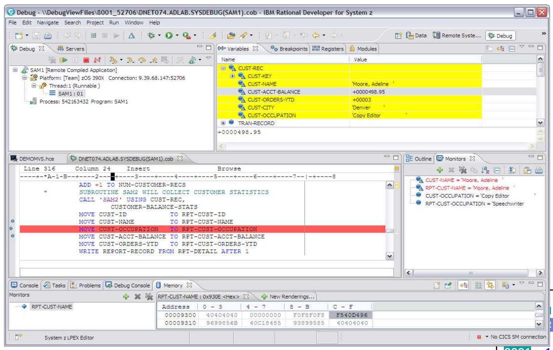
The two methods are: A) Run a batch application as a batch job, or B) Run it under TSO.

Typically the first option, (A) – running and debugging your batch application in its native environment, as a batch job – is the best and simplest method. That is because there is relatively little setup needed. You start with the JCL that runs the application, and only a minor modification is needed in the JCL to set a debugging trigger. Also, running under TSO can introduce unintended run-time problems, including anything from running out of storage to differences in internal system control blocks. Also, when you run in batch, you can debug a program and work in TSO at the same time. However, when you run and debug under TSO, you cannot enter ISPF commands or do other work in TSO while you are debugging.

Because of these reasons, running in batch is generally the better method. However, consider using the TSO method (B) if there are a limited number of batch initiators on your system, and you are frequently unable to run a batch job quickly because they queue up.

Decide which of these methods you will use, batch, or TSO.

## Plan your debugging session for batch programs – **select a debugging interface: terminal or GUI**



Debug - \\DebugViewFiles\8001_52706\DNET074.ADLAB.SYSDEBUG(SAM1).cob - IBM Rational Developer for System z

```
SAM1 [Remote Compiled Application]
  Platform: [Team] zOS 390X   Connection: 9.39.68.147:52706
    Thread:1 (Runnable )
      SAM1 : 01
  Process: 542163432  Program: SAM1
```

| Name | Value |
|---|---|
| CUST-REC | |
| CUST-KEY | |
| CUST-NAME | 'Moore, Adeline ' |
| CUST-ACCT-BALANCE | +0000498.95 |
| CUST-ORDERS-YTD | +00003 |
| CUST-CITY | 'Denver ' |
| CUST-OCCUPATION | 'Copy Editor ' |
| TRAN-RECORD | |

```
+0000498.95
```

```
 Line 316     Column 24    Insert             Browse
----+-*A-1-B--+----2---+----3----+----4----+----5----+----6----+----7--|-+----8
               ADD +1 TO NUM-CUSTOMER-RECS
      *        SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
               CALL 'SAM2' USING CUST-REC,
                       CUSTOMER-BALANCE-STATS
               MOVE CUST-ID           TO RPT-CUST-ID
               MOVE CUST-NAME         TO RPT-CUST-NAME
               MOVE CUST-OCCUPATION   TO RPT-CUST-OCCUPATION
               MOVE CUST-ACCT-BALANCE TO RPT-CUST-ACCT-BALANCE
               MOVE CUST-ORDERS-YTD   TO RPT-CUST-ORDERS-YTD
               WRITE REPORT-RECORD FROM RPT-DETAIL AFTER 1
```

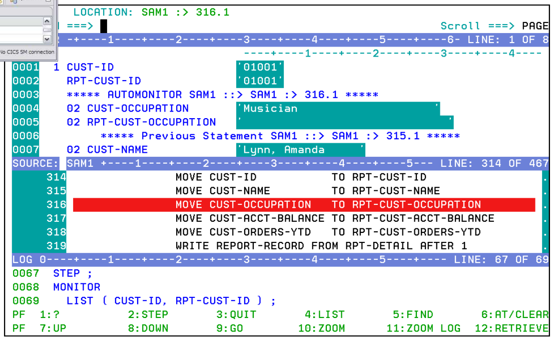
```
CUST-NAME = 'Moore, Adeline '
RPT-CUST-NAME = 'Moore, Adeline '
CUST-OCCUPATION = 'Copy Editor '
RPT-CUST-OCCUPATION = 'Speechwriter
```

RPT-CUST-NAME : 0x930E <Hex>

| Address | 0 - 3 | 4 - 7 | 8 - B | C - F |
|---|---|---|---|---|
| 00009300 | 40404040 | 00000000 | F0F8F0F8 | F540D496 |
| 00009310 | 9699856B | 40C18485 | 93899585 | 40404040 |

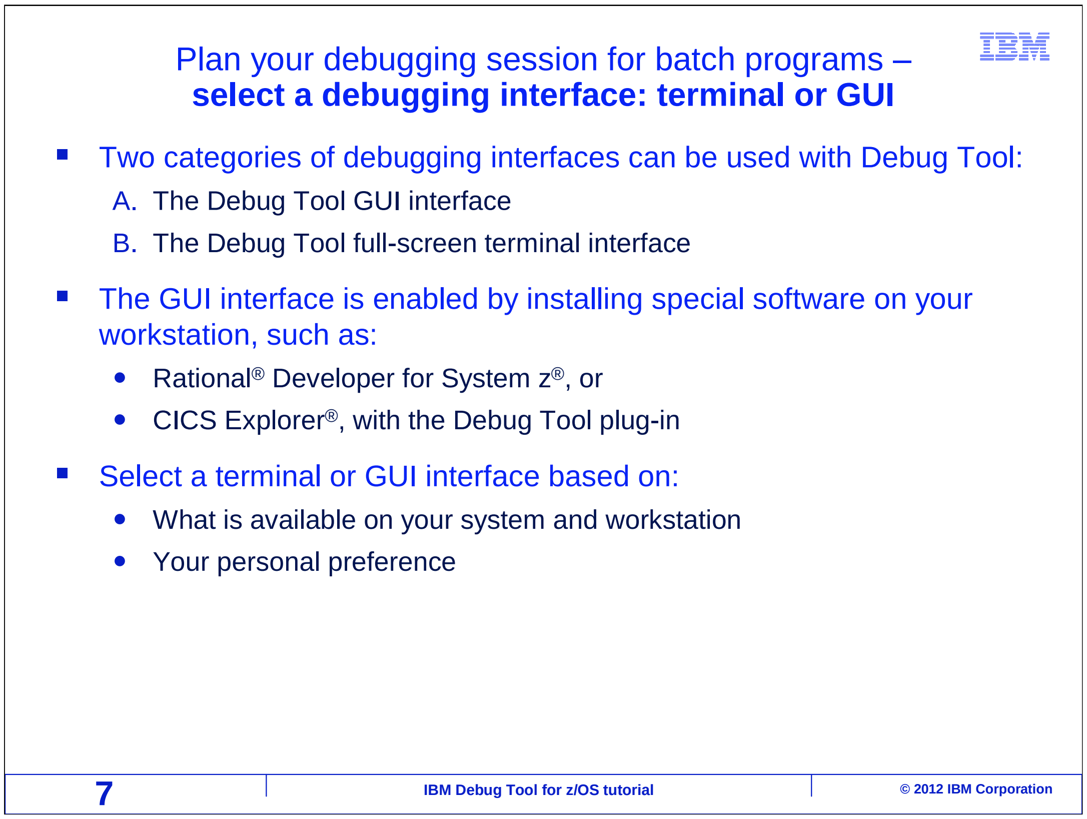
**GUI interface**

**3270 terminal interface**



```
  LOCATION: SAM1 :> 316.1
 ===>                                                        Scroll ===> PAGE
 -+----1----+----2----+----3----+----4----+----5----+----6- LINE: 1 OF 8
                                 ----+----1----+----2----+----3----+----4----
0001   1 CUST-ID                  '01001'
0002     RPT-CUST-ID              '01001'
0003     ***** AUTOMONITOR SAM1 ::> SAM1 :> 316.1 *****
0004     02 CUST-OCCUPATION       'Musician                    '
0005     02 RPT-CUST-OCCUPATION   '                              '
0006          ***** Previous Statement SAM1 ::> SAM1 :> 315.1 *****
0007     02 CUST-NAME             'Lynn, Amanda        '
SOURCE: SAM1 +----1----+----2----+----3----+----4----+----5--- LINE: 314 OF 467
     314               MOVE CUST-ID           TO RPT-CUST-ID
     315               MOVE CUST-NAME         TO RPT-CUST-NAME
     316               MOVE CUST-OCCUPATION   TO RPT-CUST-OCCUPATION
     317               MOVE CUST-ACCT-BALANCE TO RPT-CUST-ACCT-BALANCE
     318               MOVE CUST-ORDERS-YTD   TO RPT-CUST-ORDERS-YTD
     319               WRITE REPORT-RECORD FROM RPT-DETAIL AFTER 1
LOG 0----+----1----+----2----+----3----+----4----+----5----+---- LINE: 67 OF 69
0067  STEP ;
0068  MONITOR
0069    LIST ( CUST-ID, RPT-CUST-ID ) ;
PF  1:?          2:STEP        3:QUIT        4:LIST        5:FIND        6:AT/CLEAR
PF  7:UP         8:DOWN        9:GO         10:ZOOM       11:ZOOM LOG   12:RETRIEVE
```

Debug Tool can be used with either a graphical interface, or a full-screen 3270 terminal interface. Next, determine which type you plan to use.

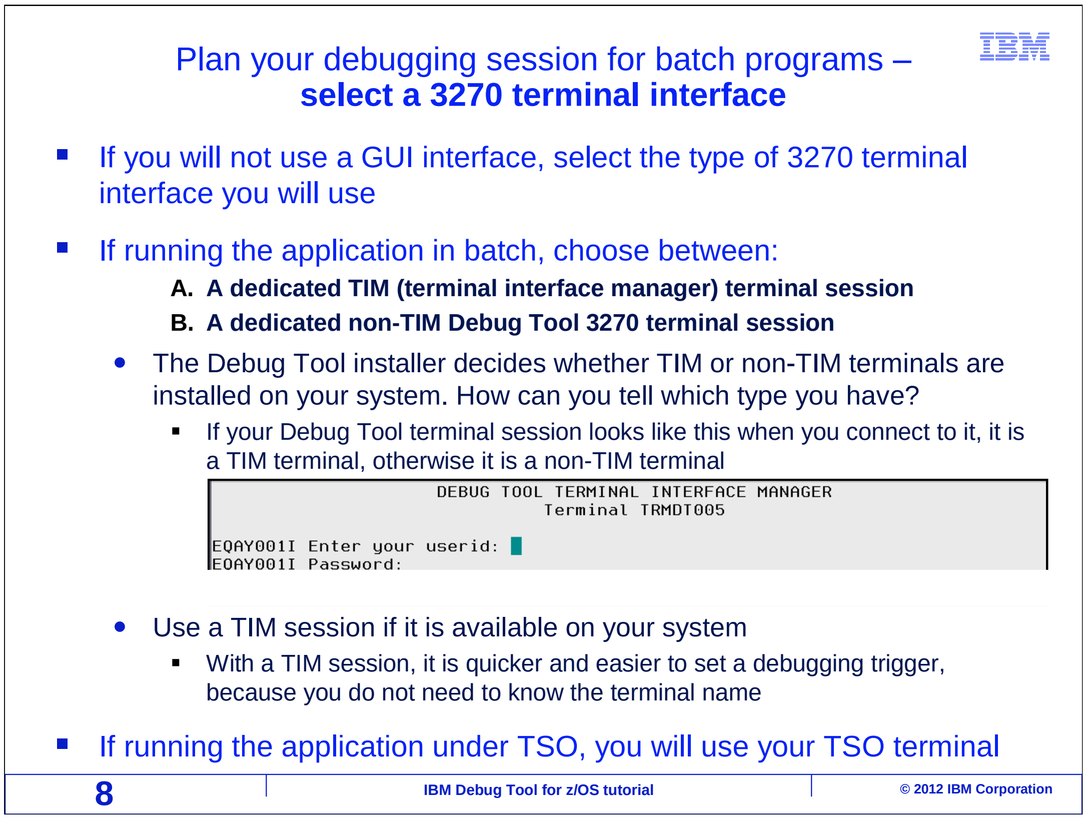


## Plan your debugging session for batch programs – **select a debugging interface: terminal or GUI**

- Two categories of debugging interfaces can be used with Debug Tool:
  A. The Debug Tool GUI interface
  B. The Debug Tool full-screen terminal interface
- The GUI interface is enabled by installing special software on your workstation, such as:
  - Rational® Developer for System z®, or
  - CICS Explorer®, with the Debug Tool plug-in
- Select a terminal or GUI interface based on:
  - What is available on your system and workstation
  - Your personal preference

7 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

The GUI is enabled by installing special software on your workstation, such as Rational Developer for System z, or CICS Explorer with the Debug Tool plug-in. On some systems, the installers of Debug Tool may have made only one or the other available. If both are available, you may be able to choose whichever you prefer.



## Plan your debugging session for batch programs – **select a 3270 terminal interface**

- If you will not use a GUI interface, select the type of 3270 terminal interface you will use
- If running the application in batch, choose between:
  - **A. A dedicated TIM (terminal interface manager) terminal session**
  - **B. A dedicated non-TIM Debug Tool 3270 terminal session**
  - The Debug Tool installer decides whether TIM or non-TIM terminals are installed on your system. How can you tell which type you have?
    - If your Debug Tool terminal session looks like this when you connect to it, it is a TIM terminal, otherwise it is a non-TIM terminal

```
                    DEBUG TOOL TERMINAL INTERFACE MANAGER
                             Terminal TRMDT005

EQAY001I Enter your userid:
EQAY001I Password:
```

  - Use a TIM session if it is available on your system
    - With a TIM session, it is quicker and easier to set a debugging trigger, because you do not need to know the terminal name
- If running the application under TSO, you will use your TSO terminal

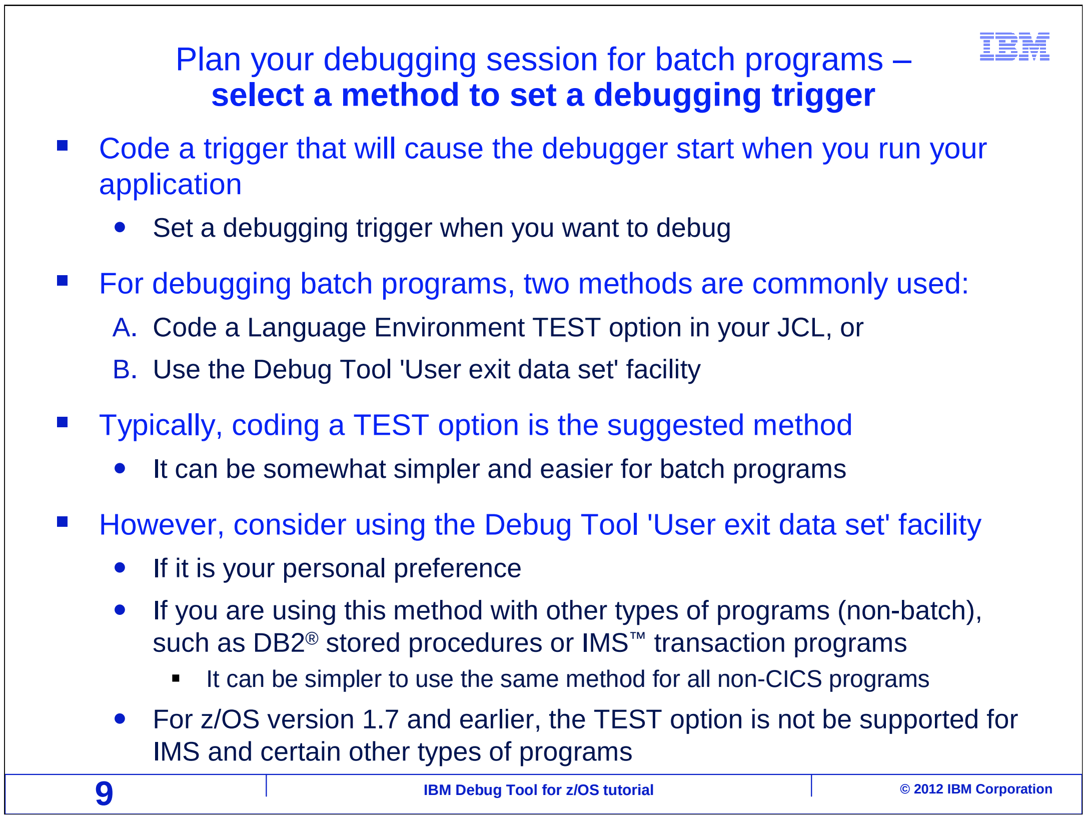
8 | IBM Debug Tool for z/OS tutorial | © 2012 IBM Corporation

If you will not be using a GUI, then determine the specific type of 3270 terminal debugging interface you will use.

If you are running the application in batch, choose between a Debug Tool terminal interface manager session (TIM for short), or a non-TIM terminal session. The Debug Tool installer decides whether Debug Tool TIM or non-TIM terminal sessions are installed on your network. How can you tell which type of terminal session you have? Look at your dedicated Debug Tool terminal session after you connect to it. If you see the title "Debug Tool terminal interface manager" at the top, then you know you have a TIM terminal. If it does not have this title, then you know it is a non-TIM terminal.

If both types are available, use a TIM terminal, because it makes it easier to set up your debugging trigger. With a non-TIM terminal, you have to specify the terminal name when you code the trigger. With a TIM terminal, you do not have to, which makes it a little simpler.

The use of a dedicated TIM or non-TIM terminal session only applies when you run your application as a batch job. If you run it under TSO, you will not use either of these. Instead, the debugger will display on your TSO terminal.



## Plan your debugging session for batch programs – **select a method to set a debugging trigger**

- Code a trigger that will cause the debugger start when you run your application
  - Set a debugging trigger when you want to debug
- For debugging batch programs, two methods are commonly used:
  - A. Code a Language Environment TEST option in your JCL, or
  - B. Use the Debug Tool 'User exit data set' facility
- Typically, coding a TEST option is the suggested method
  - It can be somewhat simpler and easier for batch programs
- However, consider using the Debug Tool 'User exit data set' facility
  - If it is your personal preference
  - If you are using this method with other types of programs (non-batch), such as DB2® stored procedures or IMS™ transaction programs
    - It can be simpler to use the same method for all non-CICS programs
  - For z/OS version 1.7 and earlier, the TEST option is not be supported for IMS and certain other types of programs

9 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

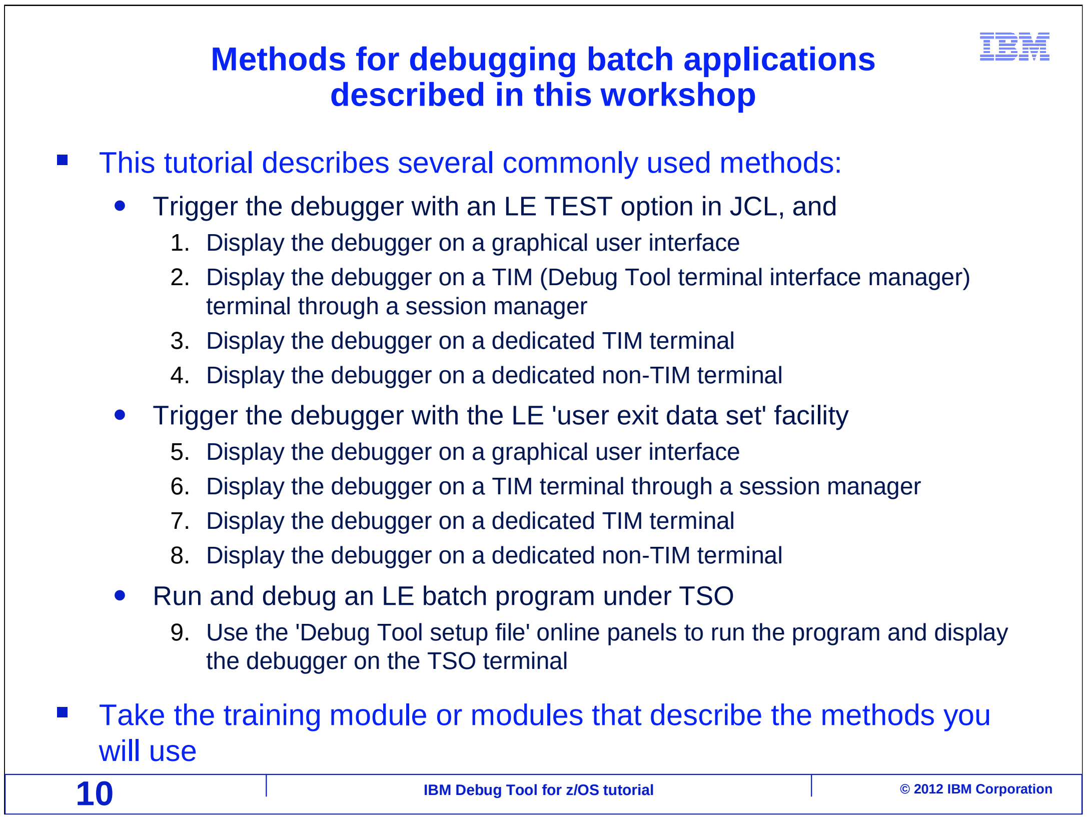
The last decision is which method you will use to set a trigger to start the debugger automatically when the application runs.

Two methods are commonly used. Either code a Language Environment TEST option in your JCL, or use the Debug Tool 'User exit data set' facility. Either of these methods can be used whether you run in batch or in TSO, and regardless of the type of debugging interface used.

A TEST option is typically the best method, because it is easy, and tends to be slightly quicker.

However, consider the other method, using the Debug Tool 'User exit data set' facility if you prefer it. You may find that the 'User exit data set' method makes it easy to debug types of programs other than batch, such as DB2 stored procedures or programs that run in IMS message processing regions under the IMS transaction manager. If you get into a habit of using this method for those other programs, you may opt to use it for batch programs too. That way, you could use the same method for all of your non-CICS programs. In older versions of z/OS, at version 1.7 and earlier, the TEST option may not work with certain types of programs, which could be a reason to use the "user exit data set' method.

You can see detailed examples of both of these methods in upcoming sections.



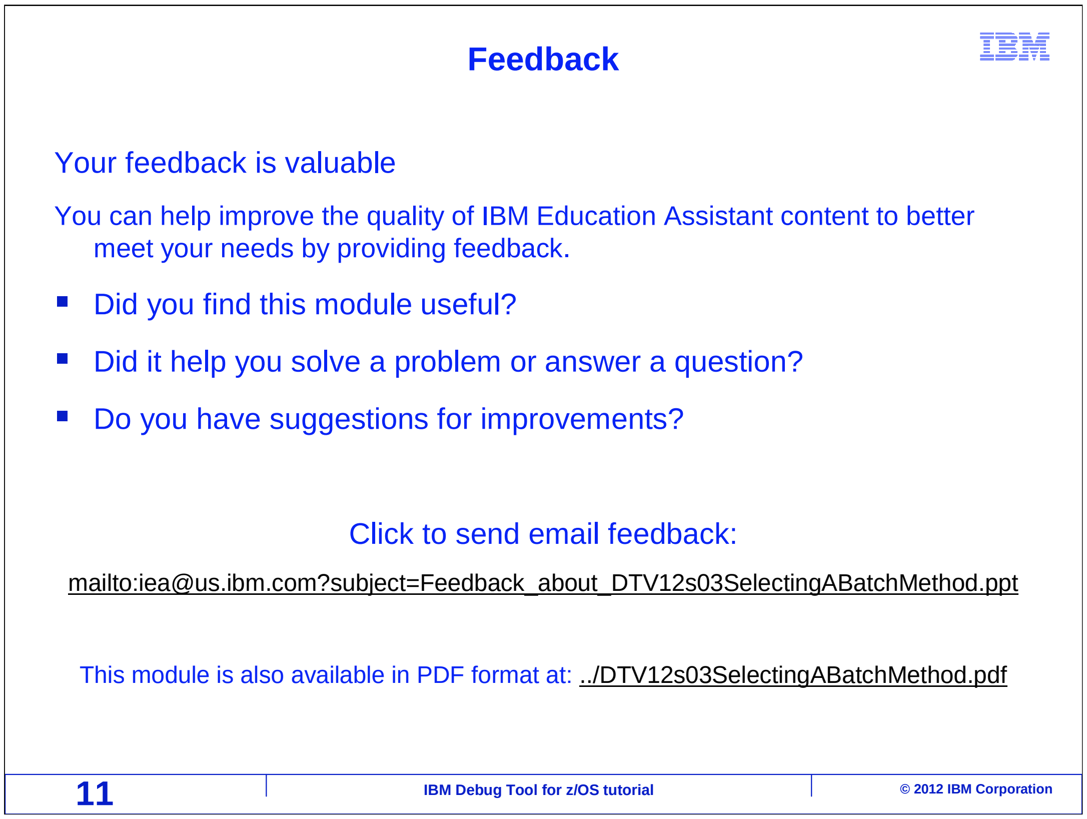
## Methods for debugging batch applications described in this workshop

- This tutorial describes several commonly used methods:
  - Trigger the debugger with an LE TEST option in JCL, and
    1. Display the debugger on a graphical user interface
    2. Display the debugger on a TIM (Debug Tool terminal interface manager) terminal through a session manager
    3. Display the debugger on a dedicated TIM terminal
    4. Display the debugger on a dedicated non-TIM terminal
  - Trigger the debugger with the LE 'user exit data set' facility
    5. Display the debugger on a graphical user interface
    6. Display the debugger on a TIM terminal through a session manager
    7. Display the debugger on a dedicated TIM terminal
    8. Display the debugger on a dedicated non-TIM terminal
  - Run and debug an LE batch program under TSO
    9. Use the 'Debug Tool setup file' online panels to run the program and display the debugger on the TSO terminal
- Take the training module or modules that describe the methods you will use

10 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

As you have seen, before you debug a batch application, you need to plan how you will run the application, the type of debugging interface you will use, and how you will set a trigger to make the debugger start. You can take different sections of the tutorial according to which methods you plan to use. Seven commonly used combinations of methods are shown in detail in different sections. View the section or sections that match how you will use the debugger, and skip sections that cover combinations of methods you will not use.

That is the end of this section, which described the methods available for you to debug a Language Environment conforming batch application.



## 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_DTV12s03SelectingABatchMethod.ppt

This module is also available in PDF format at: ../DTV12s03SelectingABatchMethod.pdf

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

## Trademarks, copyrights, and disclaimers