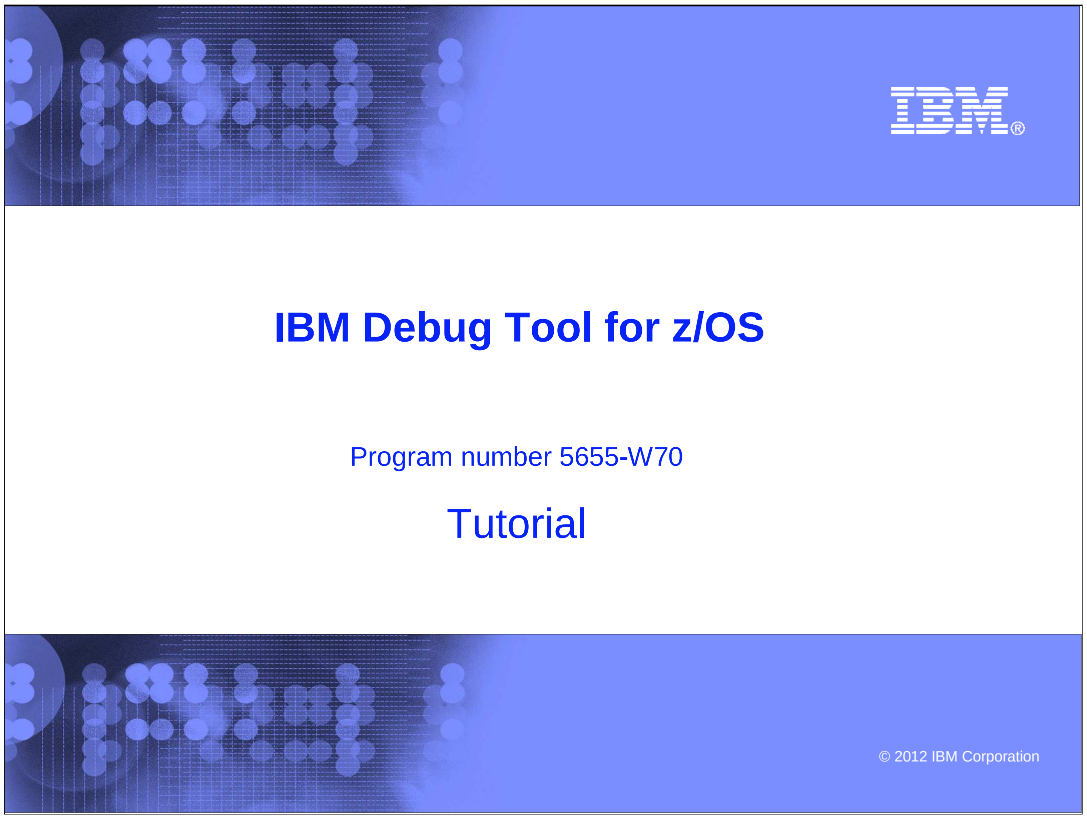
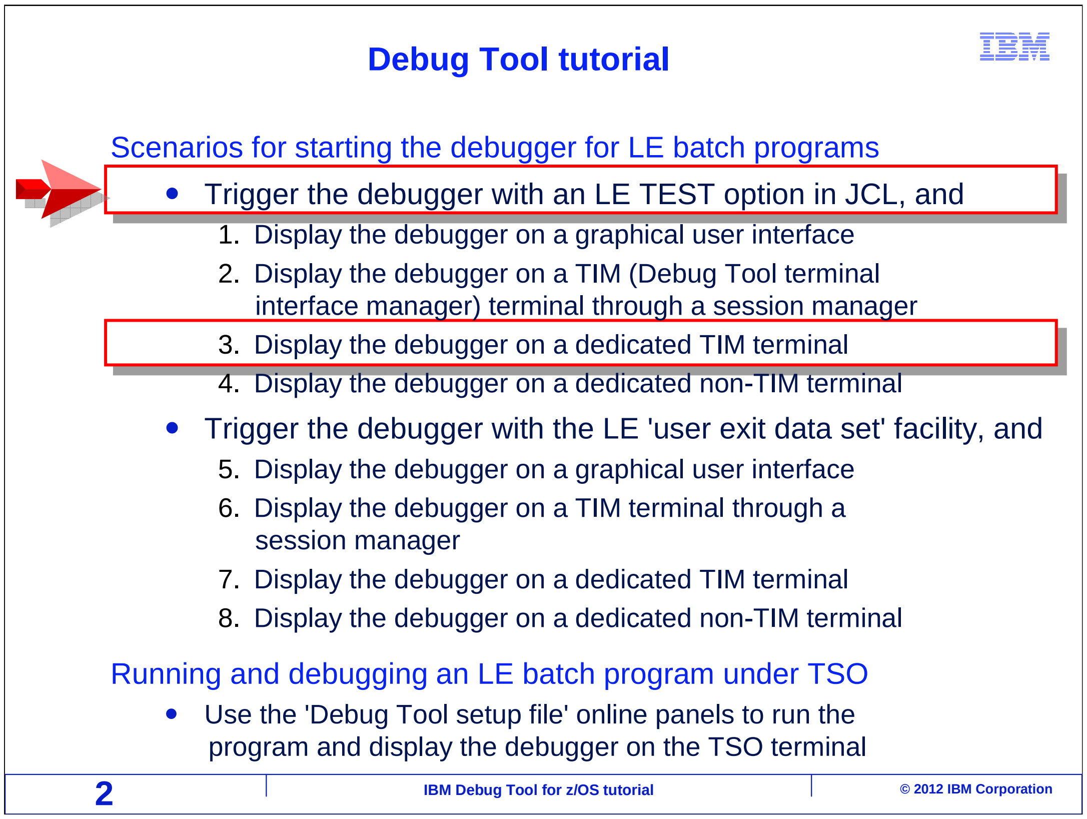


IBM Debug Tool for z/OS

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

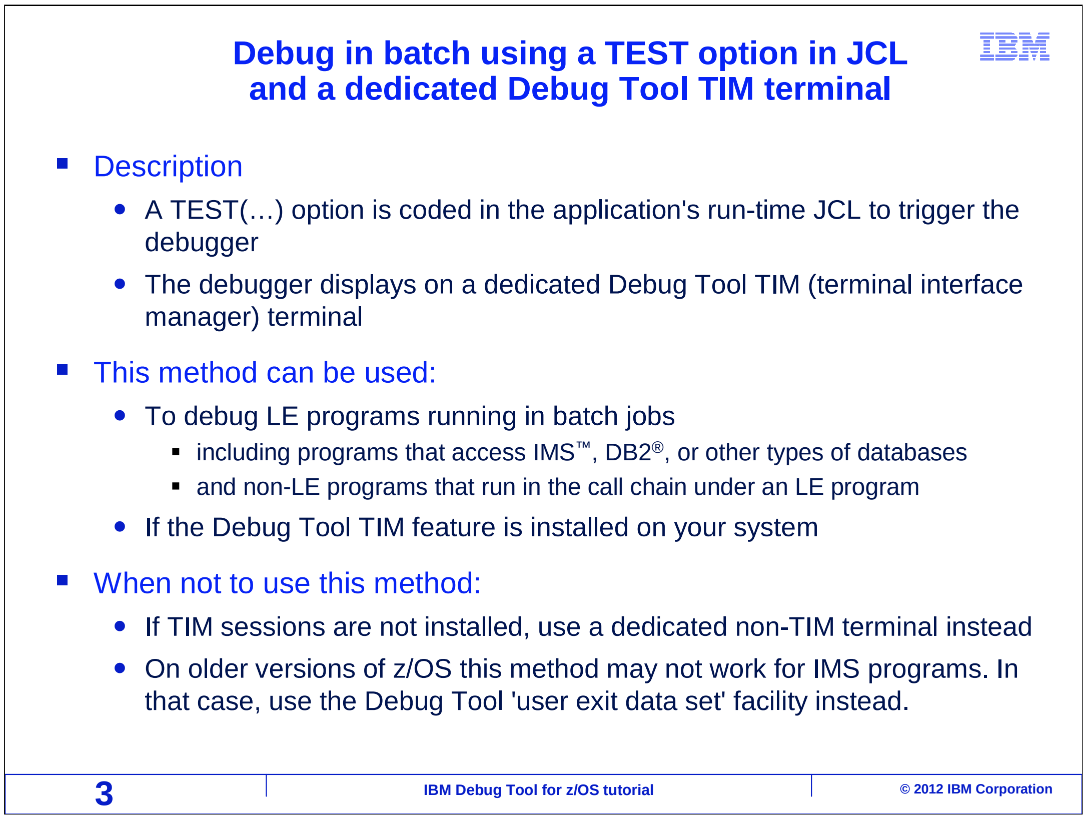
- Trigger the debugger with an LE TEST option in JCL, and
  1. Display the debugger on a graphical user interface
  2. Display the debugger on a TIM (Debug Tool terminal interface manager) terminal through a session manager
  3. Display the debugger on a dedicated TIM terminal
  4. Display the debugger 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, you will see a scenario for starting Debug Tool for a batch application. In this scenario the application runs as a batch job, and a TEST option is coded in the JCL that will trigger the debugger when the job runs. The debugger displays on a dedicated Debug Tool terminal interface manager terminal session. You can skip this section if you do not plan to use Debug Tool this way on your system.



Debug in batch using a TEST option in JCL and a dedicated Debug Tool TIM terminal

- Description
  - A TEST(…) option is coded in the application's run-time JCL to trigger the debugger
  - The debugger displays on a dedicated Debug Tool TIM (terminal interface manager) terminal
- This method can be used:
  - To debug LE programs running in batch jobs
    - including programs that access IMS™, DB2®, or other types of databases
    - and non-LE programs that run in the call chain under an LE program
  - If the Debug Tool TIM feature is installed on your system
- When not to use this method:
  - If TIM sessions are not installed, use a dedicated non-TIM terminal instead
  - On older versions of z/OS this method may not work for IMS programs. In that case, use the Debug Tool 'user exit data set' facility instead.

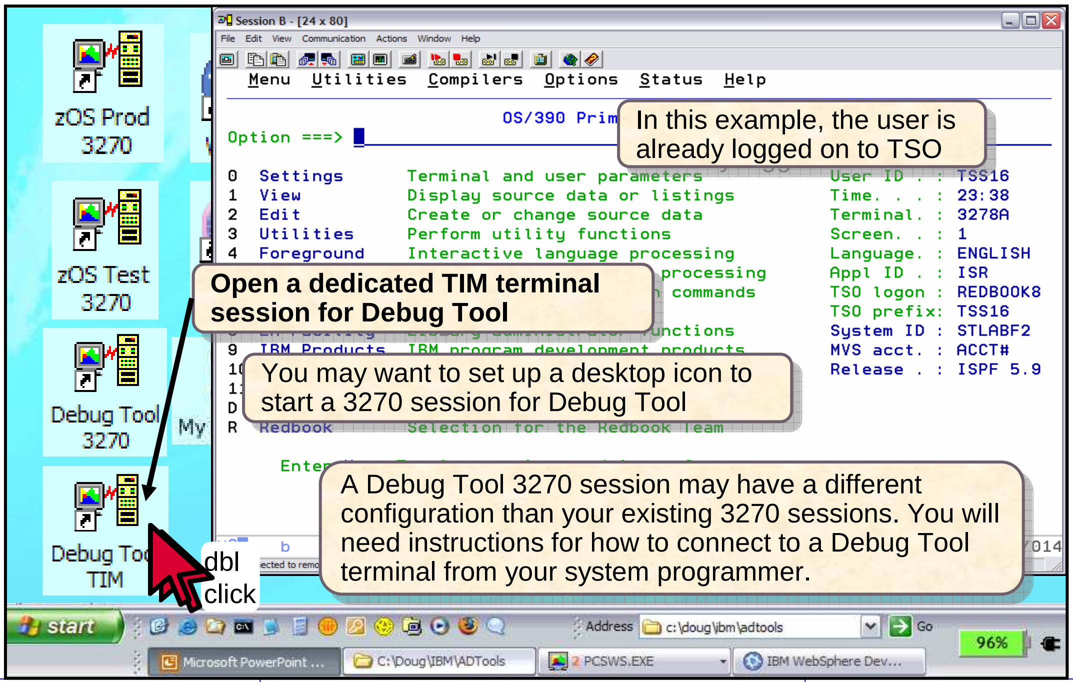
3 | IBM Debug Tool for z/OS tutorial | © 2012 IBM Corporation

This is a commonly used, simple method to start the debugger. The debugger is displayed on a Debug Tool terminal interface manager terminal. You set a trigger, to start the debugger when the application runs, by coding a Language Environment (LE) TEST option in the program's run-time JCL.

You can use this method to debug batch LE programs, including programs that access DB2, IMS, and other types of databases. Even non-LE subroutines can be debugged using this method, as long as there is at least one LE program higher in the call chain. This method assumes that the application is running in a batch job, and that the Debug Tool terminal interface manager is installed on your system.

This is generally the best method when a TIM terminal is used. However, you cannot use this method if TIM terminal sessions have not been defined on your system's network. If not, consider displaying the debugger on a non-TIM terminal or a GUI debugger instead. Also, if you are running on an older version of z/OS (1.7 or earlier), a TEST option may not trigger the debugger for IMS batch programs. In that case, the 'user exit data set' method could be used instead.

How to trigger Debug Tool for a batch job



```
 Menu  Utilities  Compilers  Options  Status  Help
                         OS/390 Prim
Option ===>
0  Settings      Terminal and user parameters        User ID . : TSS16
1  View          Display source data or listings     Time. . . : 23:38
2  Edit          Create or change source data        Terminal. : 3278A
3  Utilities     Perform utility functions           Screen. . : 1
4  Foreground    Interactive language processing     Language. : ENGLISH
                                        processing   Appl ID . : ISR
                                       commands      TSO logon : REDBOOK8
                                                     TSO prefix: TSS16
                                        unctions     System ID : STLABF2
9  IBM Products  IBM program development products    MVS acct. : ACCT#
                                                     Release . : ISPF 5.9
R  Redbook       Selection for the Redbook Team

   Enter
```

In this example, the user is already logged on to TSO

Open a dedicated TIM terminal session for Debug Tool

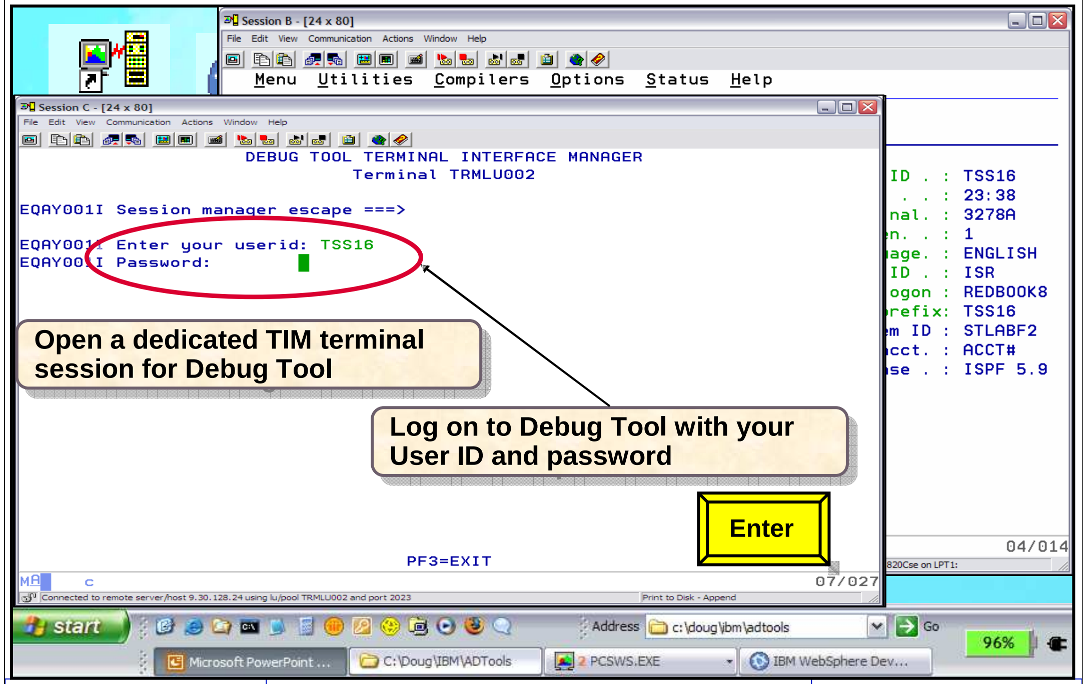
You may want to set up a desktop icon to start a 3270 session for Debug Tool

A Debug Tool 3270 session may have a different configuration than your existing 3270 sessions. You will need instructions for how to connect to a Debug Tool terminal from your system programmer.

In this example, the user is already logged on to TSO. The first step is to open a dedicated terminal session for the debugger. You may find it easiest to set up a desktop icon so that you can quickly start a terminal session for the debugger. A Debug Tool terminal session may be configured differently than your other 3270 sessions. You may need instructions from your system programmer describing how to configure a 3270 emulator session to connect to a dedicated Debug Tool terminal.

In this example, the user has already configured a special terminal session for Debug Tool, and set up an icon for it on the desktop. The icon is double clicked.

Open a Debug Tool TIM terminal and log on



```
                         DEBUG TOOL TERMINAL INTERFACE MANAGER
                                  Terminal TRMLU002

EQAY001I Session manager escape ===>

EQAY001I Enter your userid: TSS16
EQAY001I Password:

                                   PF3=EXIT
```

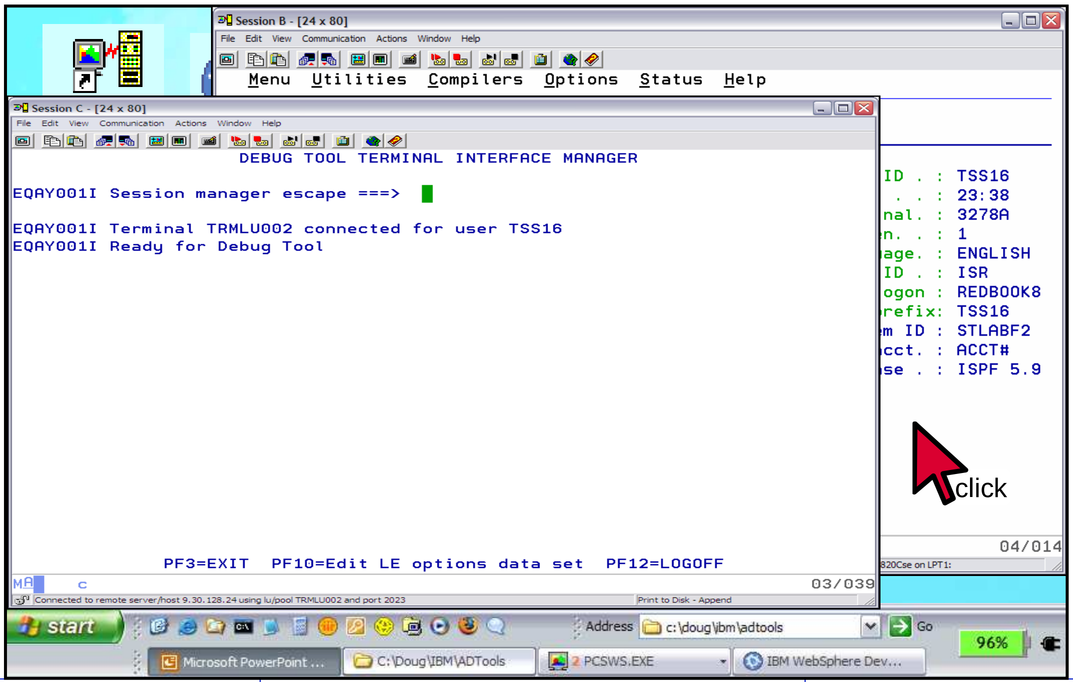
```
ID . : TSS16
 . . : 23:38
nal. : 3278A
n. . : 1
age. : ENGLISH
ID . : ISR
ogon : REDBOOK8
refix: TSS16
m ID : STLABF2
cct. : ACCT#
se . : ISPF 5.9
```

Session B - [24 x 80]
Menu Utilities Compilers Options Status Help
Session C - [24 x 80]
Open a dedicated TIM terminal session for Debug Tool
Log on to Debug Tool with your User ID and password
Enter
07/027
04/014
Connected to remote server/host 9.30.128.24 using lu/pool TRMLU002 and port 2023
start
Microsoft PowerPoint ...
C:\Doug\IBM\ADTools
PCSWS.EXE
IBM WebSphere Dev...

That opened a terminal session for the debugger. Notice the title "Debug Tool terminal interface manager" on the Debug Tool terminal. That indicates that it is a TIM terminal. If your Debug Tool terminal does not have this title, then you either have not correctly connected to it, or you have a non-TIM terminal session. If you do not have a TIM terminal, take one of the tutorial sections that describes how to debug using a non-TIM terminal, instead of this section.

Next, log on to the Debug Tool terminal with your ID and password. Typically, this is the same ID and password you use to log onto TSO. Do not be confused by thinking that you are logging onto TSO twice. You are not logging onto TSO here, you are logging on to Debug Tool. The ID and password are typed in, and enter is pressed.

The Debug Tool TIM terminal is ready

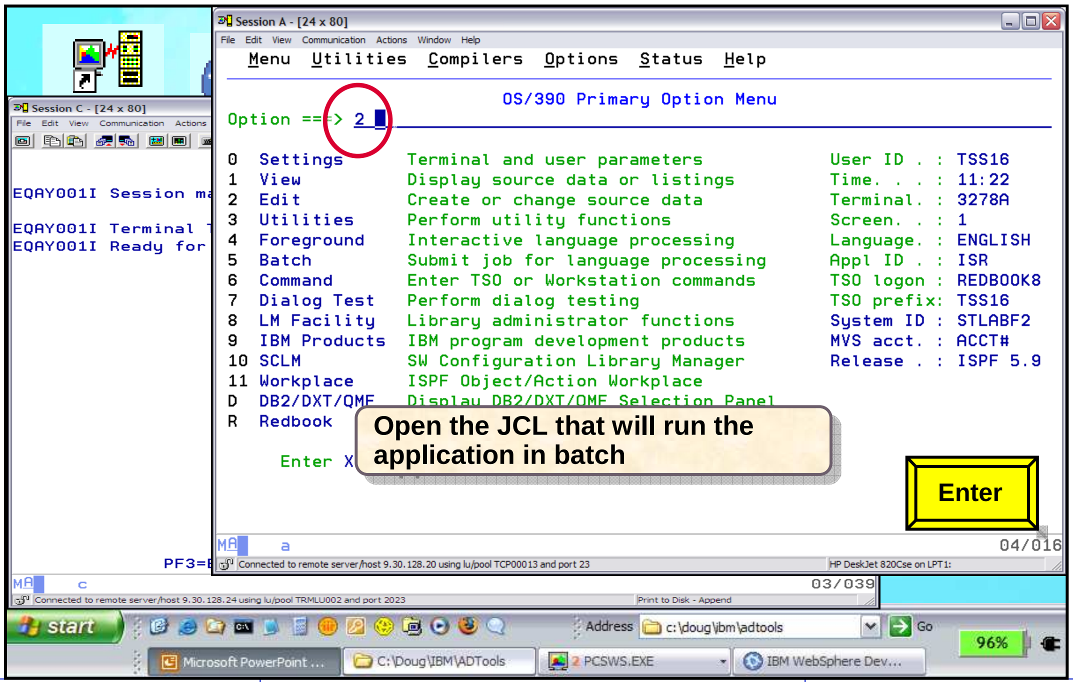


```
DEBUG TOOL TERMINAL INTERFACE MANAGER

EQAY001I Session manager escape ===>

EQAY001I Terminal TRMLU002 connected for user TSS16
EQAY001I Ready for Debug Tool


            PF3=EXIT  PF10=Edit LE options data set  PF12=LOGOFF
```

```
ID . . : TSS16
 . . . : 23:38
nal. . : 3278A
n. . . : 1
age. . : ENGLISH
ID . . : ISR
ogon . : REDBOOK8
refix: TSS16
m ID . : STLABF2
cct. . : ACCT#
se . . : ISPF 5.9
```

At this point your Debug Tool terminal is ready to receive a debugging session. Next, the TSO terminal session is selected again by clicking on it.

Open the JCL that will run the batch program

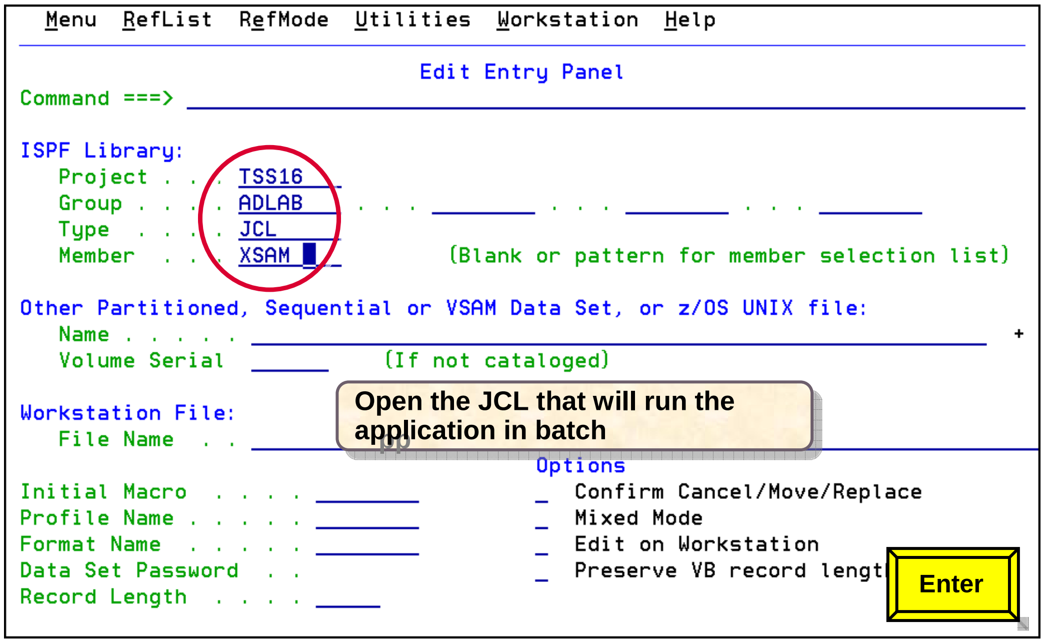


```
 Menu  Utilities  Compilers  Options  Status  Help
                              OS/390 Primary Option Menu
Option ===> 2

0  Settings      Terminal and user parameters            User ID . : TSS16
1  View          Display source data or listings         Time. . . : 11:22
2  Edit          Create or change source data            Terminal. : 3278A
3  Utilities     Perform utility functions               Screen. . : 1
4  Foreground    Interactive language processing         Language. : ENGLISH
5  Batch         Submit job for language processing      Appl ID . : ISR
6  Command       Enter TSO or Workstation commands       TSO logon : REDBOOK8
7  Dialog Test   Perform dialog testing                  TSO prefix: TSS16
8  LM Facility   Library administrator functions         System ID : STLABF2
9  IBM Products  IBM program development products        MVS acct. : ACCT#
10 SCLM          SW Configuration Library Manager        Release . : ISPF 5.9
11 Workplace     ISPF Object/Action Workplace
D  DB2/DXT/QMF   Display DB2/DXT/QMF Selection Panel
R  Redbook

   Enter X
```

```
EQAY001I Session ma
EQAY001I Terminal T
EQAY001I Ready for
```

In TSO, open the JCL that runs the application you want to debug. In this example, the ISPF editor is selected.

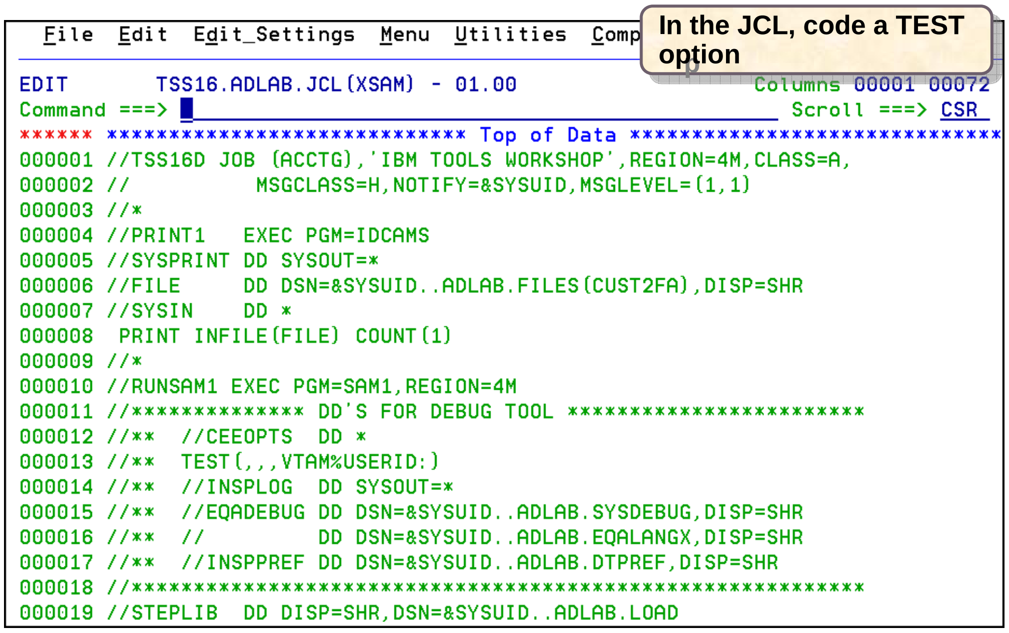
Open the JCL that will run the batch program



```
   Menu  RefList  RefMode  Utilities  Workstation  Help
                          Edit Entry Panel
Command ===> 

ISPF Library:
   Project . . . TSS16
   Group . . . . ADLAB     . . .          . . .          . . .
   Type  . . . . JCL
   Member  . . . XSAM              (Blank or pattern for member selection list)

Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:
   Name . . . . . .                                                    +
   Volume Serial           (If not cataloged)

Workstation File:
   File Name  . .
                                       Options
Initial Macro  . . . .                 _  Confirm Cancel/Move/Replace
Profile Name . . . . .                 _  Mixed Mode
Format Name  . . . . .                 _  Edit on Workstation
Data Set Password  . .                 _  Preserve VB record lengt
Record Length  . . . .
```

Open the JCL in the editor.

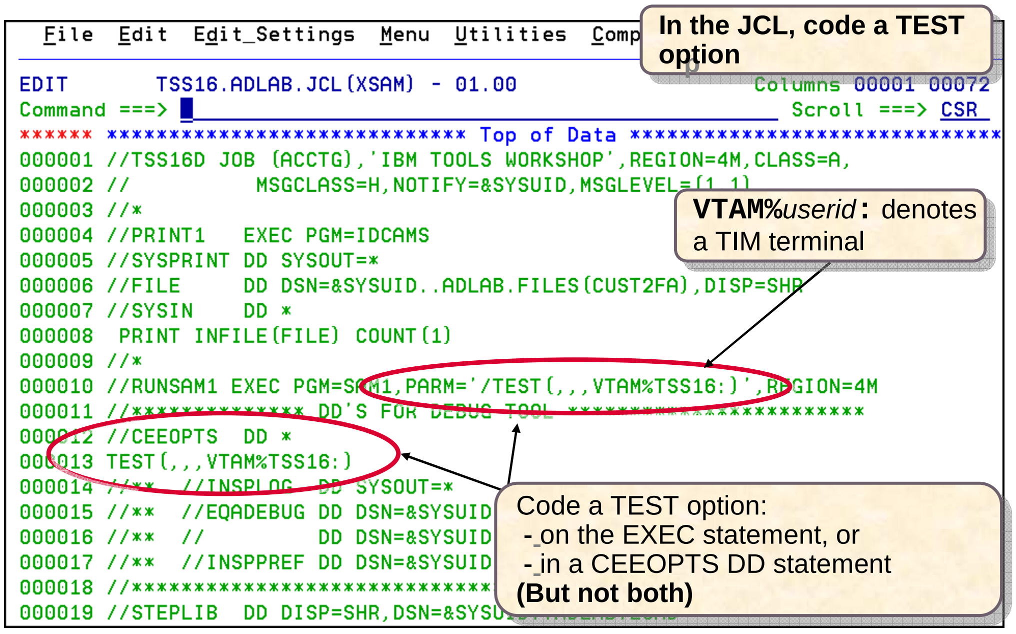
This is the JCL before adding a TEST option


File Edit Edit_Settings Menu Utilities Comp
In the JCL, code a TEST option

```
EDIT       TSS16.ADLAB.JCL(XSAM) - 01.00                  Columns 00001 00072
Command ===> _                                              Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 //TSS16D JOB (ACCTG),'IBM TOOLS WORKSHOP',REGION=4M,CLASS=A,
000002 //          MSGCLASS=H,NOTIFY=&SYSUID,MSGLEVEL=(1,1)
000003 //*
000004 //PRINT1   EXEC PGM=IDCAMS
000005 //SYSPRINT DD SYSOUT=*
000006 //FILE     DD DSN=&SYSUID..ADLAB.FILES(CUST2FA),DISP=SHR
000007 //SYSIN    DD *
000008  PRINT INFILE(FILE) COUNT(1)
000009 //*
000010 //RUNSAM1 EXEC PGM=SAM1,REGION=4M
000011 //************** DD'S FOR DEBUG TOOL ***********************
000012 //**  //CEEOPTS  DD *
000013 //**  TEST(,,,VTAM%USERID:)
000014 //**  //INSPLOG  DD SYSOUT=*
000015 //**  //EQADEBUG DD DSN=&SYSUID..ADLAB.SYSDEBUG,DISP=SHR
000016 //**  //         DD DSN=&SYSUID..ADLAB.EQALANGX,DISP=SHR
000017 //**  //INSPPREF DD DSN=&SYSUID..ADLAB.DTPREF,DISP=SHR
000018 //*********************************************************
000019 //STEPLIB  DD DISP=SHR,DSN=&SYSUID..ADLAB.LOAD
```

This is JCL that is used to run the application program in this example. It is shown here before any changes have been made.

Code a TEST option in a CEEOPTS DD or as a PARM on the EXEC statement



```
  File  Edit  Edit_Settings  Menu  Utilities  Comp
EDIT       TSS16.ADLAB.JCL(XSAM) - 01.00                    Columns 00001 00072
Command ===>                                                  Scroll ===> CSR
****** ***************************** Top of Data ******************************
//TSS16D JOB (ACCTG),'IBM TOOLS WORKSHOP',REGION=4M,CLASS=A,
//          MSGCLASS=H,NOTIFY=&SYSUID,MSGLEVEL=(1,1)
//*
//PRINT1   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//FILE     DD DSN=&SYSUID..ADLAB.FILES(CUST2FA),DISP=SHR
//SYSIN    DD *
 PRINT INFILE(FILE) COUNT(1)
//*
//RUNSAM1 EXEC PGM=SAM1,PARM='/TEST(,,,VTAM%TSS16:)',REGION=4M
//************* DD'S FOR DEBUG TOOL ***********************
//CEEOPTS  DD *
TEST(,,,VTAM%TSS16:)
//**  //INSPLOG  DD SYSOUT=*
//**  //EQADEBUG DD DSN=&SYSUID
//**  //         DD DSN=&SYSUID
//**  //INSPPREF DD DSN=&SYSUID
//******************************
//STEPLIB  DD DISP=SHR,DSN=&SYSU
```

In the JCL, code a TEST option

VTAM%*userid***:** denotes a TIM terminal

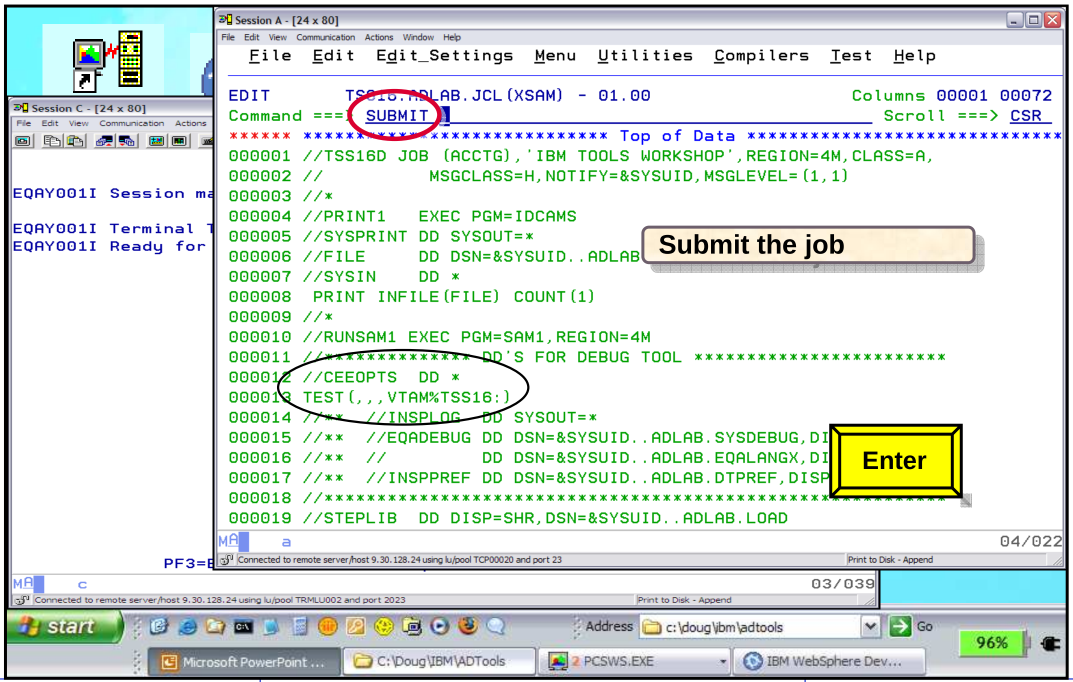
Code a TEST option:
- on the EXEC statement, or
- in a CEEOPTS DD statement

(But not both)

To set a trigger for the debugger, code a Language Environment TEST option in the JCL. There are two ways to do it. Depending on how the JCL is coded, you may be able to code a TEST option directly on the EXEC statement that runs the program. Or you can add a special DD statement called CEEOPTS, and code a TEST option as input data in the CEEOPTS DD. Although both methods are shown in this example, only code one or the other.

Notice how the TEST option is coded. After the third comma, it has "VTAM%user ID:". The VTAM® keyword specifies that the Debug Tool terminal interface manager will be used. The user ID is coded after the % sign. This is the user ID that was used to log onto the Debug Tool TIM terminal, and it will be used by Debug Tool to locate the terminal.

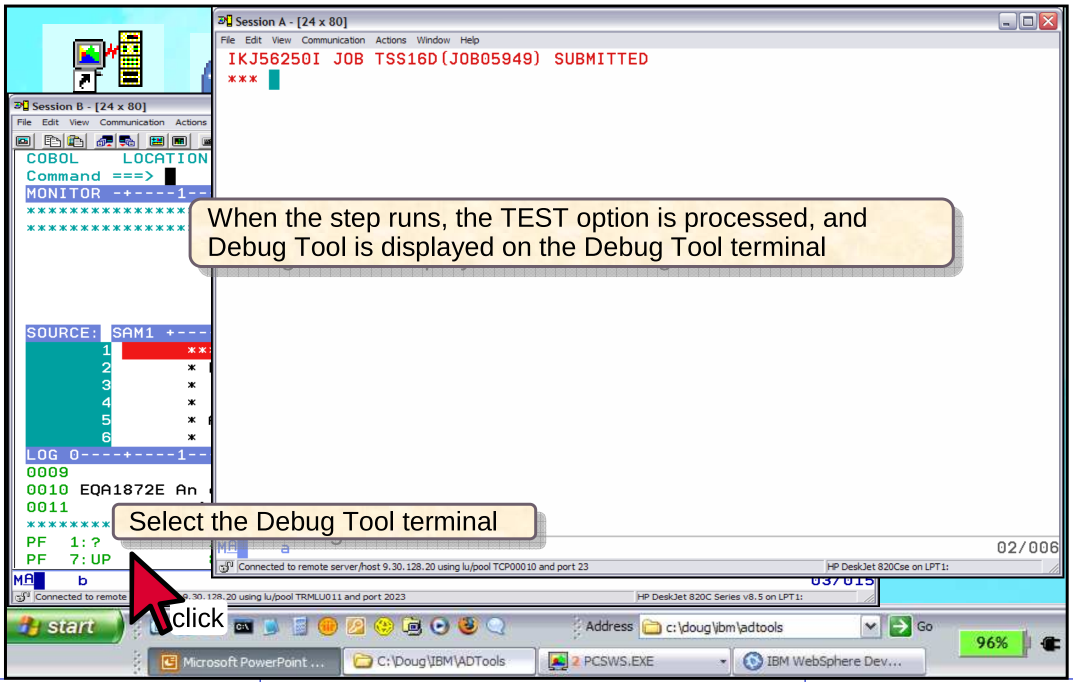
Submit the JCL to run the batch job



```
EDIT       TSS16.ADLAB.JCL(XSAM) - 01.00                   Columns 00001 00072
Command ===> SUBMIT                                            Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 //TSS16D JOB (ACCTG),'IBM TOOLS WORKSHOP',REGION=4M,CLASS=A,
000002 //          MSGCLASS=H,NOTIFY=&SYSUID,MSGLEVEL=(1,1)
000003 //*
000004 //PRINT1   EXEC PGM=IDCAMS
000005 //SYSPRINT DD SYSOUT=*
000006 //FILE     DD DSN=&SYSUID..ADLAB
000007 //SYSIN    DD *
000008  PRINT INFILE(FILE) COUNT(1)
000009 //*
000010 //RUNSAM1 EXEC PGM=SAM1,REGION=4M
000011 //*************** DD'S FOR DEBUG TOOL ***********************
000012 //CEEOPTS  DD *
000013 TEST(,,,VTAM%TSS16:)
000014 //**  //INSPLOG  DD SYSOUT=*
000015 //**  //EQADEBUG DD DSN=&SYSUID..ADLAB.SYSDEBUG,DI
000016 //**  //         DD DSN=&SYSUID..ADLAB.EQALANGX,DI
000017 //**  //INSPPREF DD DSN=&SYSUID..ADLAB.DTPREF,DISP
000018 //***********************************************************
000019 //STEPLIB  DD DISP=SHR,DSN=&SYSUID..ADLAB.LOAD
```

The job is ready to run. The JCL is submitted to batch.

When the job step runs, the debugger is triggered



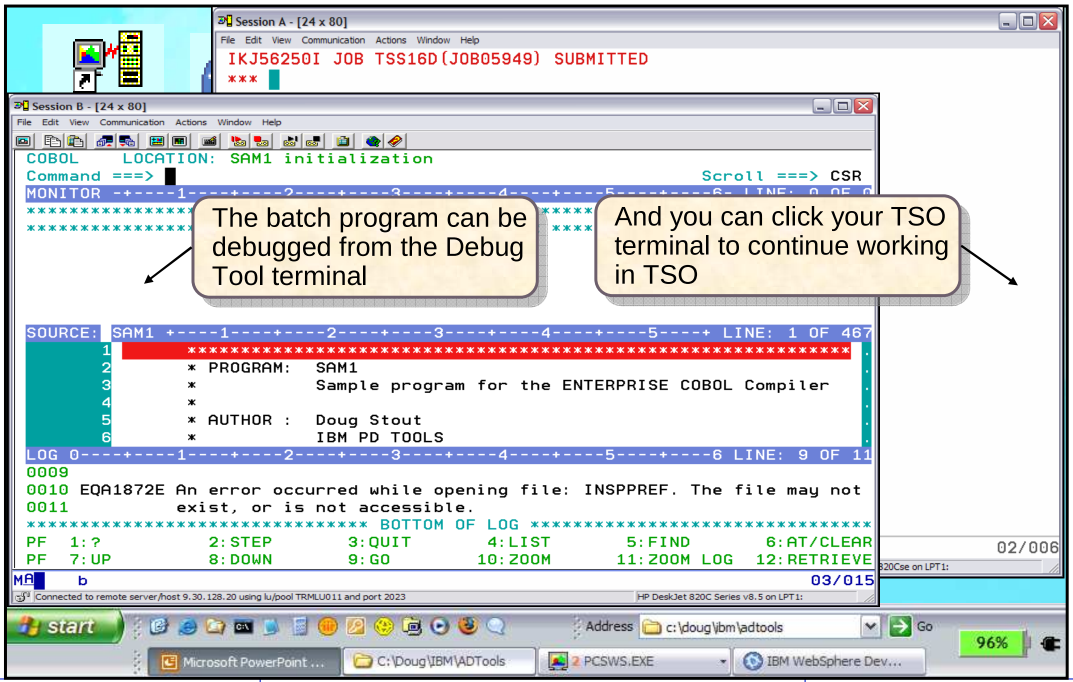
```
IKJ56250I JOB TSS16D(JOB05949) SUBMITTED
***
```

```
COBOL    LOCATION
Command ===>
MONITOR -+----1--
***************
***************
SOURCE: SAM1 +---
LOG 0----+----1--
0009
0010 EQA1872E An
0011
********
PF   1:?
PF   7:UP
```

Session A - [24 x 80]
Session B - [24 x 80]
When the step runs, the TEST option is processed, and Debug Tool is displayed on the Debug Tool terminal
Select the Debug Tool terminal
click
Connected to remote server/host 9.30.128.20 using lu/pool TCP00010 and port 23
Connected to remote
9.30.128.20 using lu/pool TRMLU011 and port 2023
start
Microsoft PowerPoint ...
C:\Doug\IBM\ADTools
PCSWS.EXE
IBM WebSphere Dev...

When a step runs that has a TEST option specified, Language Environment receives the TEST option and starts Debug Tool. The debugger is displayed in the TIM terminal session. Here, the Debug Tool terminal emulator window is selected by clicking on it.

The debugger automatically displays on the TIM terminal

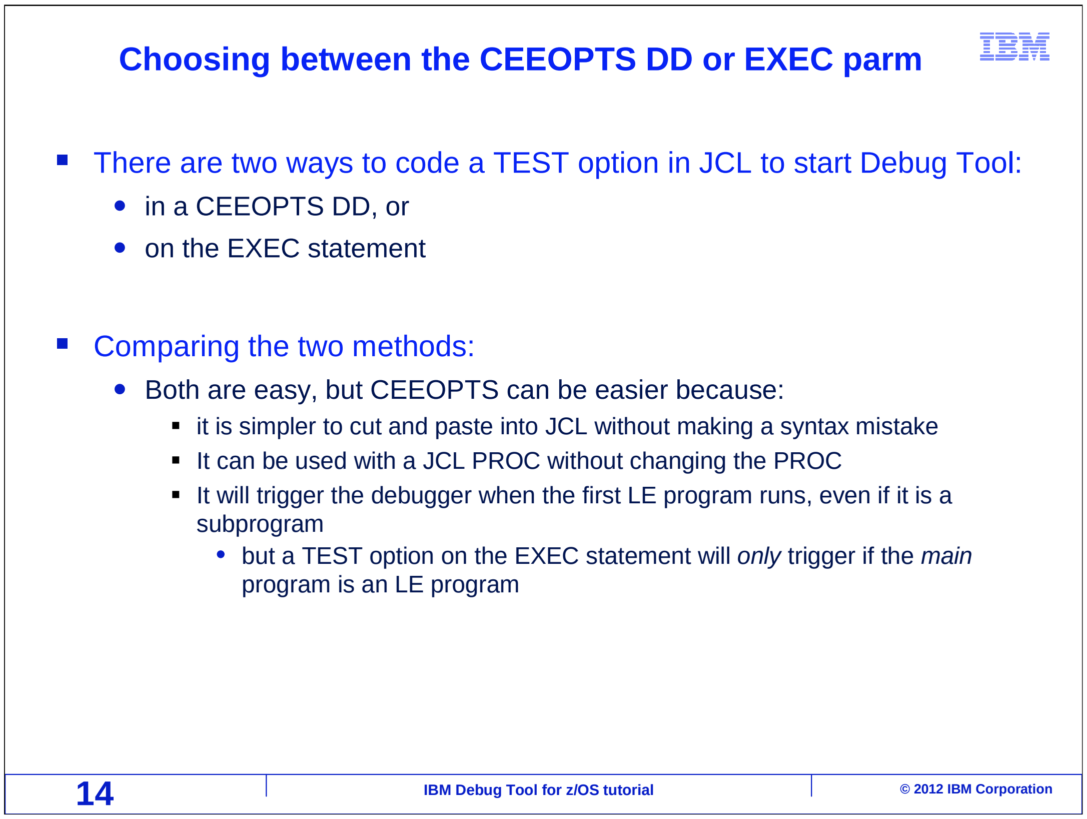


```
IKJ56250I JOB TSS16D(JOB05949) SUBMITTED
***
```

```
COBOL     LOCATION: SAM1 initialization
Command ===>                                                   Scroll ===> CSR
MONITOR -+----1----+----2----+----3----+----4----+----5----+----6- LINE: 0 OF 0
SOURCE: SAM1 +----1----+----2----+----3----+----4----+----5----+ LINE: 1 OF 467
     1       ***************************************************************
     2       * PROGRAM:   SAM1
     3       *            Sample program for the ENTERPRISE COBOL Compiler
     4       *
     5       * AUTHOR :   Doug Stout
     6       *            IBM PD TOOLS
LOG 0----+----1----+----2----+----3----+----4----+----5----+----6 LINE: 9 OF 11
0009
0010 EQA1872E An error occurred while opening file: INSPPREF. The file may not
0011          exist, or is not accessible.
******************************* BOTTOM OF LOG ********************************
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
```

Session A - [24 x 80]
Session B - [24 x 80]
File Edit View Communication Actions Window Help
The batch program can be debugged from the Debug Tool terminal
And you can click your TSO terminal to continue working in TSO
Connected to remote server/host 9.30.128.20 using lu/pool TRMLU011 and port 2023
start
Address c:\doug\ibm\adtools
Microsoft PowerPoint ...
C:\Doug\IBM\ADTools
PCSWS.EXE
IBM WebSphere Dev...

The batch job is running on the host z/OS system, and Debug Tool is communicating with the Debug Tool terminal. Notice that you can control the program from the Debug Tool terminal, and you can also click your TSO terminal window to continue working in TSO at the same time. Be aware that the TSO session is no longer needed. You could log off from TSO and continue to debug, because Debug Tool is controlling the application that is running in its own batch address space.



Choosing between the CEEOPTS DD or EXEC parm

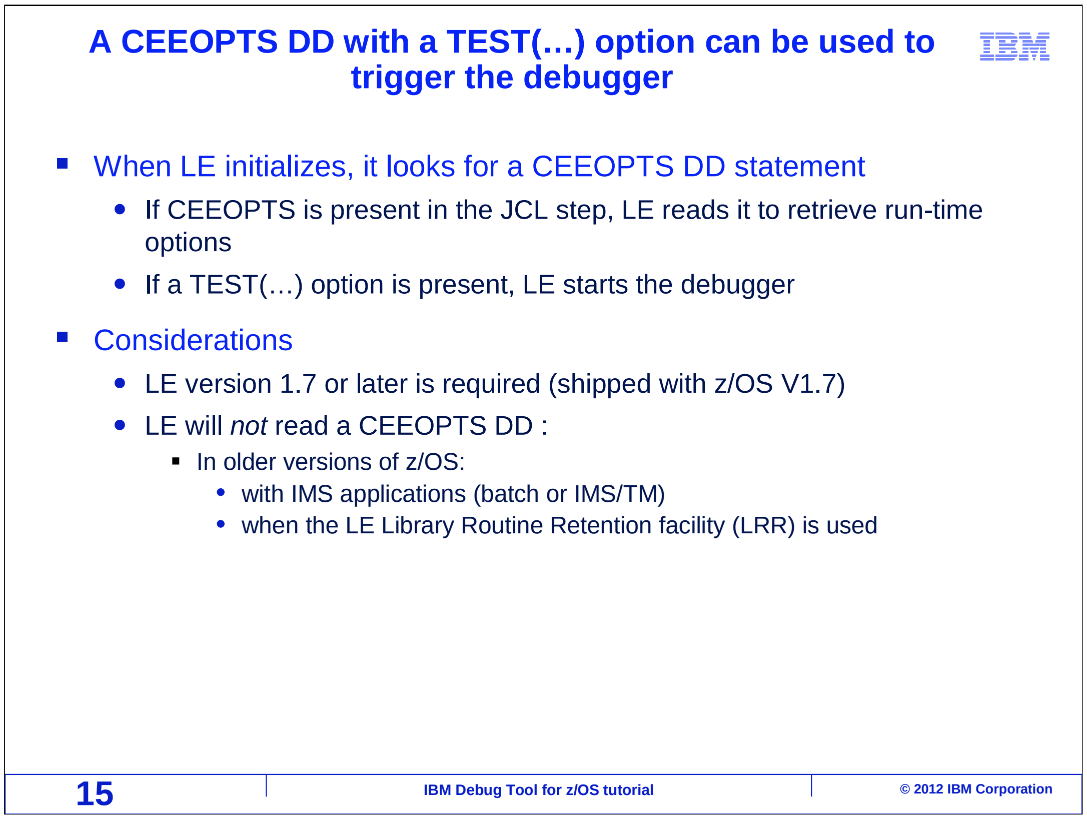
- There are two ways to code a TEST option in JCL to start Debug Tool:
  - in a CEEOPTS DD, or
  - on the EXEC statement
- Comparing the two methods:
  - Both are easy, but CEEOPTS can be easier because:
    - it is simpler to cut and paste into JCL without making a syntax mistake
    - It can be used with a JCL PROC without changing the PROC
    - It will trigger the debugger when the first LE program runs, even if it is a subprogram
      - but a TEST option on the EXEC statement will *only* trigger if the *main* program is an LE program

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

There are two ways to code a TEST option in JCL to start Debug Tool. On the EXEC statement, or with a CEEOPTS DD statement. For most people, the CEEOPTS DD method is a little easier. Here is why.

When you use the CEEOPTS method, it can be easier to “cut and paste” a TEST option into JCL from other JCL where you have coded it before. Also, if a cataloged JCL PROC is used, you may not be able to code a parm on the EXEC statement, because it is embedded in the PROC. With a CEEOPTS DD, it is easy to code it in your JCL to make it take effect in any step you want in your PROC. You will see an example of that in a minute.

Finally, when you use CEEOPTS, Debug Tool will trigger when the first LE program is called, even if it is a subprogram. That is an advantage over the other method, because when you code a TEST option on the EXEC statement, Debug Tool will only trigger if the main program is an LE program.



A CEEOPTS DD with a TEST(…) option can be used to trigger the debugger

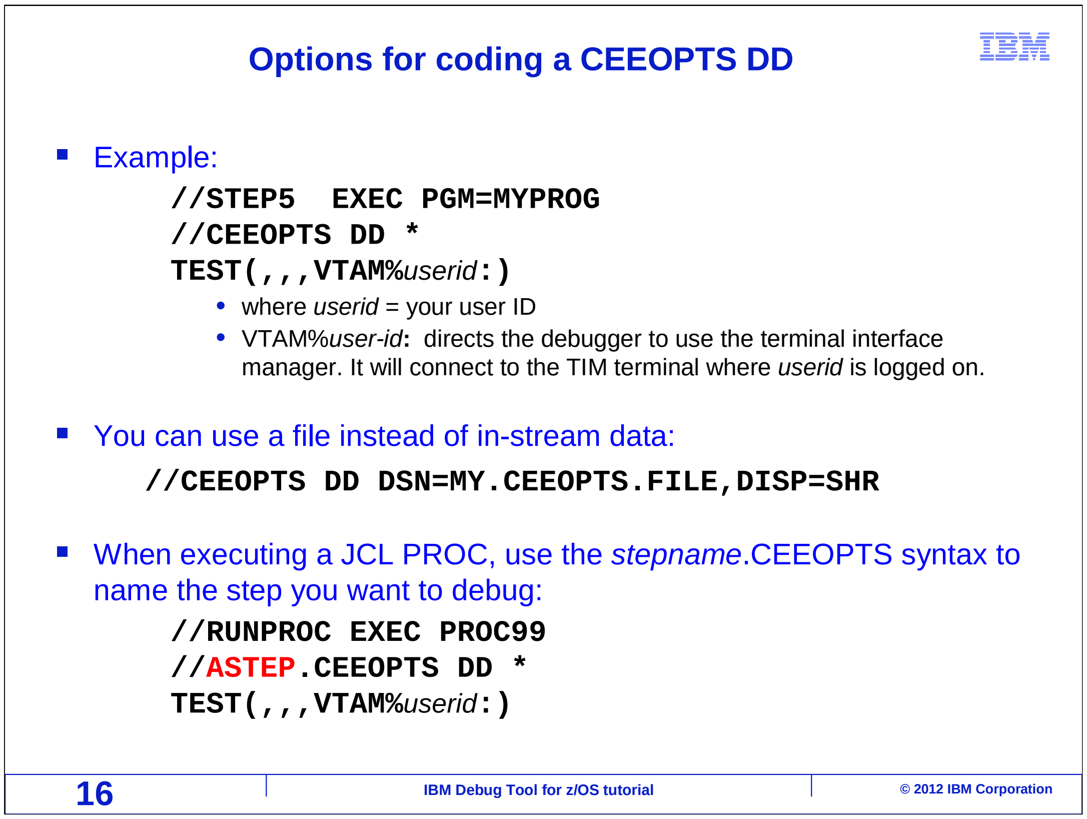
- When LE initializes, it looks for a CEEOPTS DD statement
  - If CEEOPTS is present in the JCL step, LE reads it to retrieve run-time options
  - If a TEST(…) option is present, LE starts the debugger
- Considerations
  - LE version 1.7 or later is required (shipped with z/OS V1.7)
  - LE will *not* read a CEEOPTS DD :
    - In older versions of z/OS:
      - with IMS applications (batch or IMS/TM)
      - when the LE Library Routine Retention facility (LRR) is used

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

When you run an LE program, which includes programs compiled with compilers such as Enterprise COBOL and Enterprise PL/I, Language Environment initializes when the program starts. When LE initializes, as part of its normal processing, it looks for a CEEOPTS DD statement. If it finds one, it reads it to retrieve run-time options. When you pass a TEST option to LE, it will start Debug Tool.

But there are a couple of considerations and restrictions. First, you must be at LE version 1.7 or later. Since LE is shipped with z/OS, typically that means that the z/OS operating system must be at version 1.7 or later.

LE will not read a CEEOPTS DD statement with IMS applications on older versions of z/OS. So, if you want to debug an IMS batch application and you are running on an older system, you cannot use a TEST option in JCL to do it. In that case, use the 'Debug Tool user exit data set facility' to trigger the debugger.



Options for coding a CEEOPTS DD

- Example:

```
//STEP5  EXEC PGM=MYPROG
//CEEOPTS DD *
TEST(,,,VTAM%userid:)
```

  - where *userid* = your user ID
  - VTAM%*user-id*: directs the debugger to use the terminal interface manager. It will connect to the TIM terminal where *userid* is logged on.

- You can use a file instead of in-stream data:

```
//CEEOPTS DD DSN=MY.CEEOPTS.FILE,DISP=SHR
```

- When executing a JCL PROC, use the *stepname*.CEEOPTS syntax to name the step you want to debug:

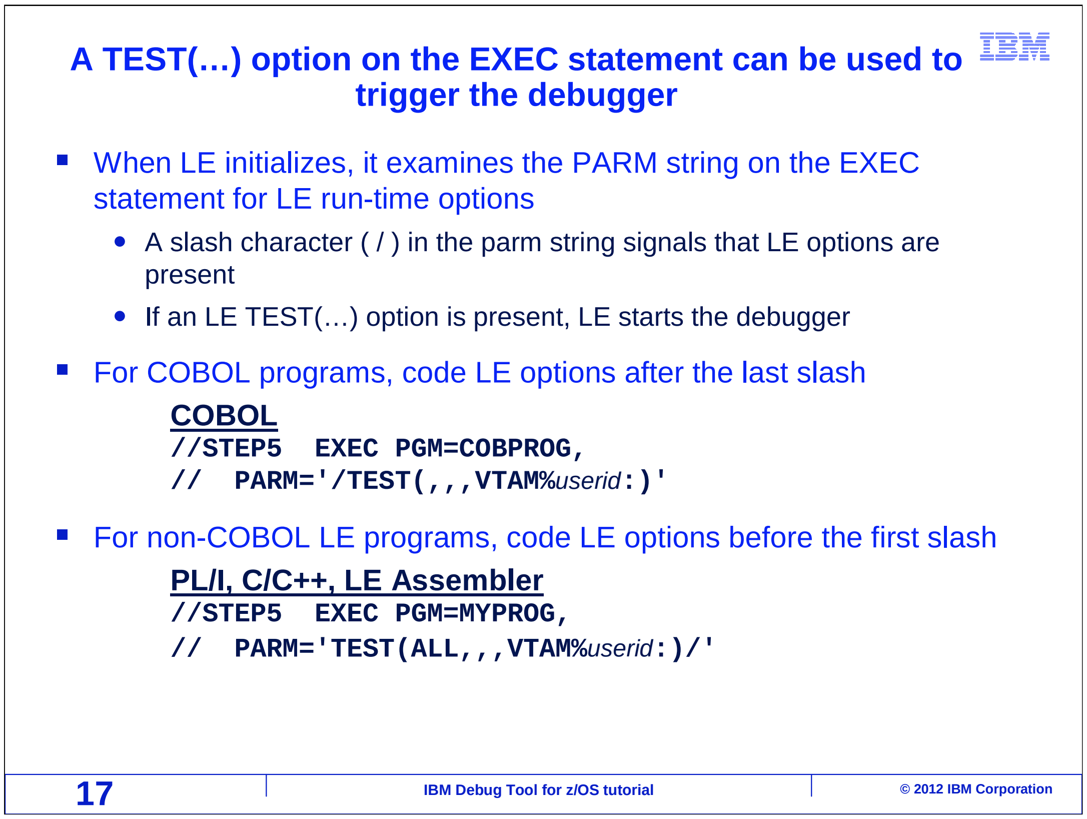
```
//RUNPROC EXEC PROC99
//ASTEP.CEEOPTS DD *
TEST(,,,VTAM%userid:)
```

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

Here are examples of coding CEEOPTS DD statements in your JCL. In the first example, CEEOPTS is coded as one of the DD statements in a step. The TEST option can be coded as in-stream data as is shown.

The second example shows that you can have your TEST option coded in a file if you find that more convenient.

The third example shows how you can code a CEEOPTS DD statement when the JCL executes a PROC. In this case, notice that the EXEC statement runs a PROC named PROC99. The “ASTEP.CEEOPTS” DD statement will add the CEEOPTS DD to a proc step named ASTEP. When it is done this way, you do not have to make any changes to the PROC itself.



A TEST(…) option on the EXEC statement can be used to trigger the debugger

- When LE initializes, it examines the PARM string on the EXEC statement for LE run-time options
  - A slash character (/) in the parm string signals that LE options are present
  - If an LE TEST(…) option is present, LE starts the debugger
- For COBOL programs, code LE options after the last slash

  COBOL
  ```
  //STEP5   EXEC PGM=COBPROG,
  //   PARM='/TEST(,,,VTAM%userid:)'
  ```
- For non-COBOL LE programs, code LE options before the first slash

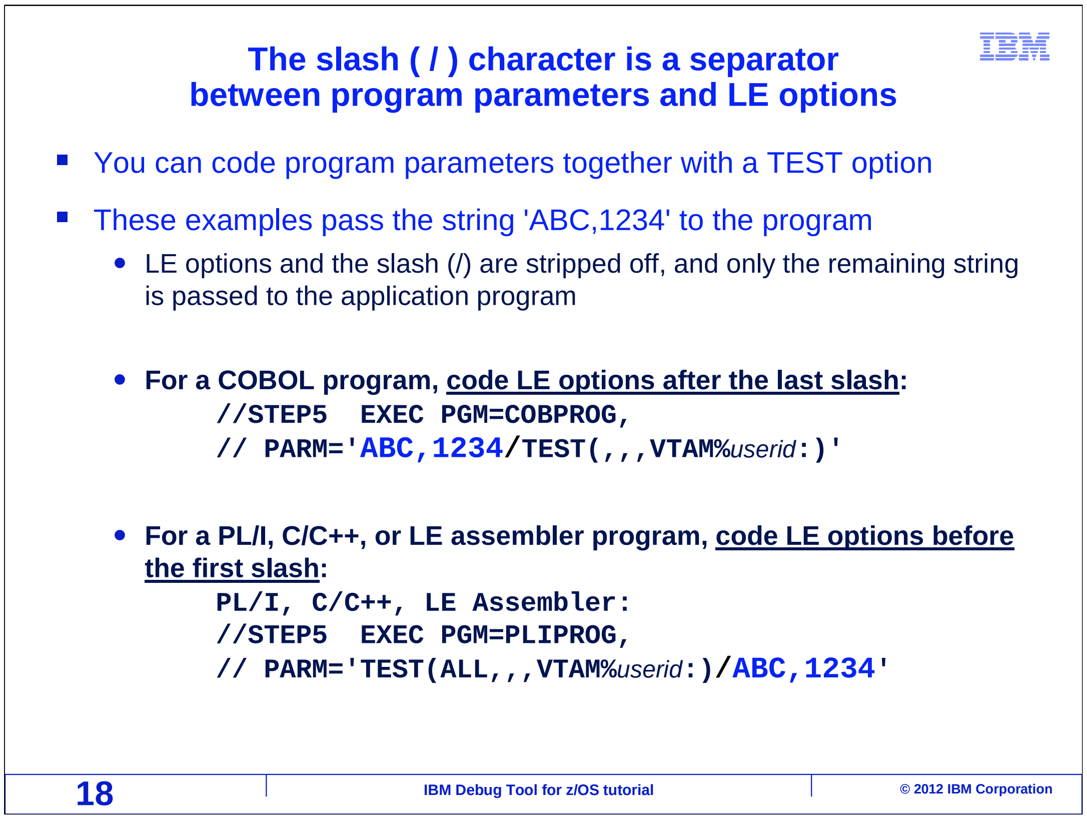
  PL/I, C/C++, LE Assembler
  ```
  //STEP5   EXEC PGM=MYPROG,
  //   PARM='TEST(ALL,,,VTAM%userid:)/'
  ```

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

There are two ways to code a TEST option in JCL. Either use a CEEOPTS DD statement as you have already seen, or code a TEST option on the EXEC statement.

Here is how to code a TEST option on the EXEC statement. When LE initializes, it examines the PARM string on the EXEC statement for LE run-time options and looks for a slash. For COBOL programs, everything after the last slash is taken as LE options. For non-COBOL LE programs, everything before the first slash is taken as LE options. In either case, you must code a slash at the appropriate location in your PARM string.

Here is an important restriction. LE will examine the PARM string from the EXEC statement only if the main program is an LE program. That means that you cannot use this method to start Debug Tool if the main program is a non-LE program.



The slash (/) character is a separator between program parameters and LE options

- You can code program parameters together with a TEST option
- These examples pass the string 'ABC,1234' to the program
  - LE options and the slash (/) are stripped off, and only the remaining string is passed to the application program
  - **For a COBOL program, <u>code LE options after the last slash</u>:**

```
//STEP5  EXEC PGM=COBPROG,
// PARM='ABC,1234/TEST(,,,VTAM%userid:)'
```

  - **For a PL/I, C/C++, or LE assembler program, <u>code LE options before the first slash</u>:**

```
PL/I, C/C++, LE Assembler:
//STEP5  EXEC PGM=PLIPROG,
// PARM='TEST(ALL,,,VTAM%userid:)/ABC,1234'
```

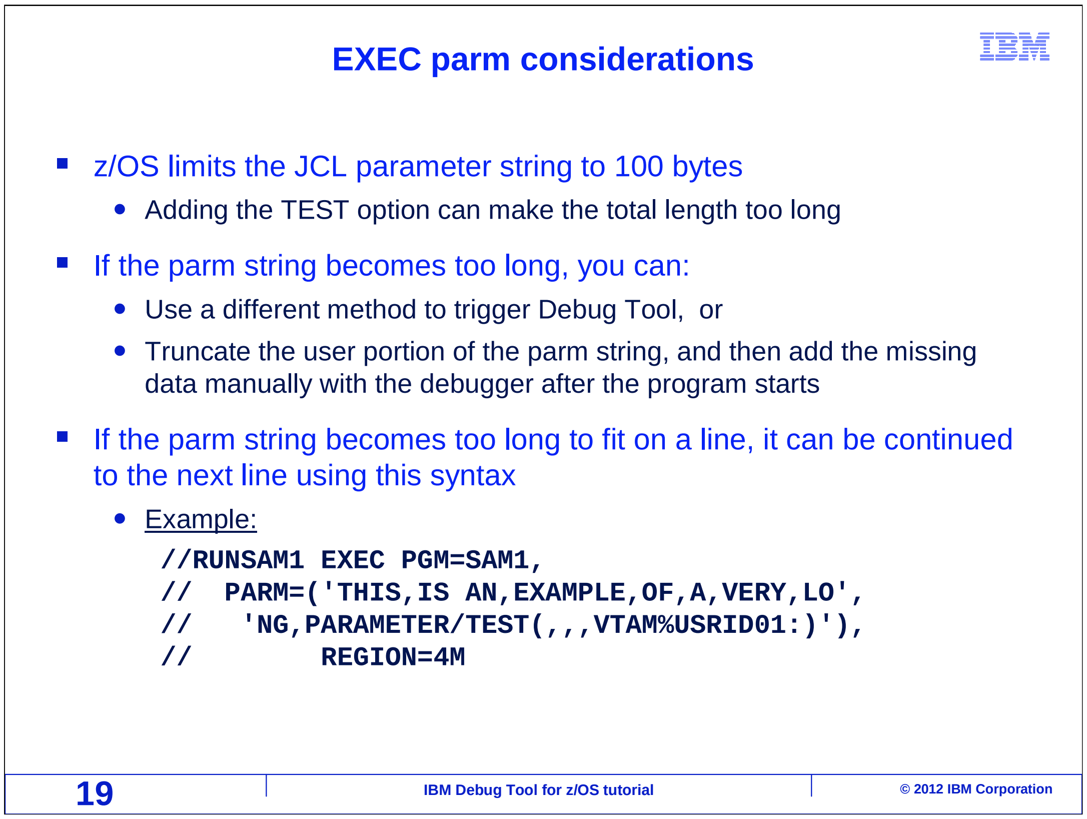
18 | IBM Debug Tool for z/OS tutorial | © 2012 IBM Corporation

When you code a TEST option on the EXEC statement, you can still code parameters for your program. Here are examples that pass the data string “ABC,1234” to the application program as a parameter.

Remember that for COBOL programs, all parameters coded after the last slash are taken as LE options. So to pass parameters to the program, code them all before the last slash.

For non-COBOL LE programs, all parameters coded before the first slash are taken as LE options. So to pass parameters to your program, code them all after the first slash.

In either case, LE strips its options and the slash out of the parm string before control is passed to the application program. So the LE options are not received by your program.


EXEC parm considerations

- z/OS limits the JCL parameter string to 100 bytes
  - Adding the TEST option can make the total length too long
- If the parm string becomes too long, you can:
  - Use a different method to trigger Debug Tool, or
  - Truncate the user portion of the parm string, and then add the missing data manually with the debugger after the program starts
- If the parm string becomes too long to fit on a line, it can be continued to the next line using this syntax
  - Example:

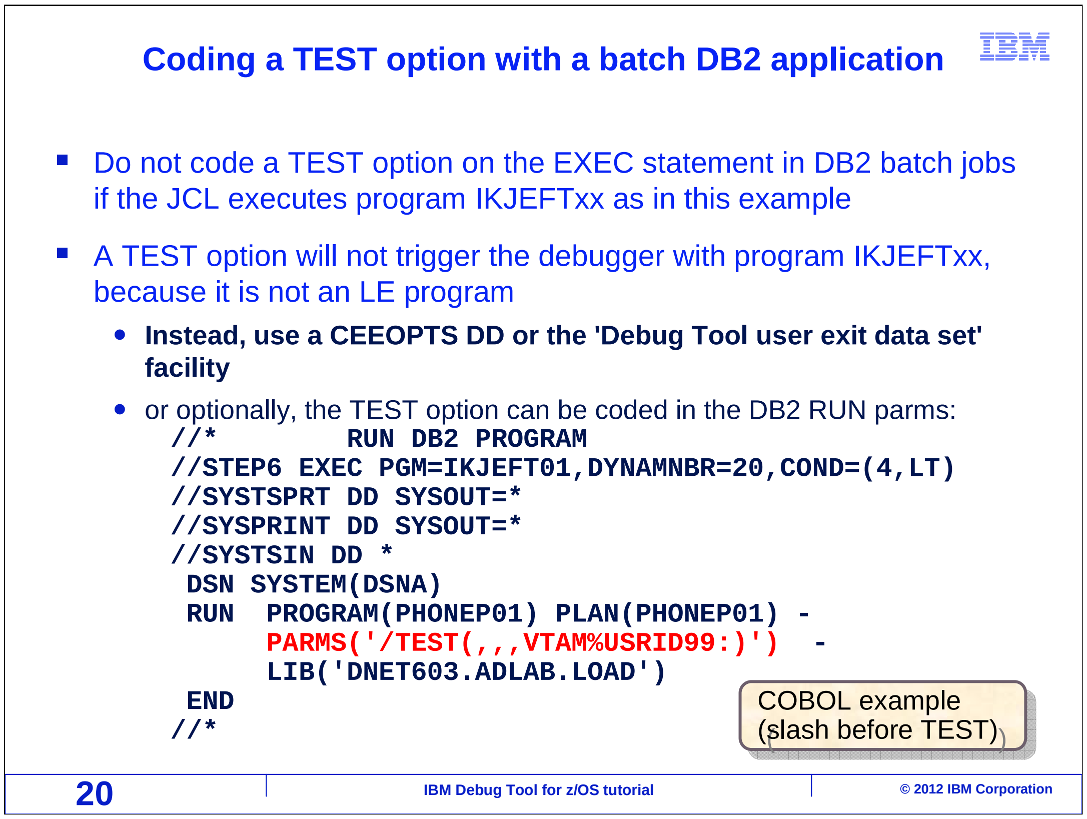
```
//RUNSAM1 EXEC PGM=SAM1,
//   PARM=('THIS,IS AN,EXAMPLE,OF,A,VERY,LO',
//    'NG,PARAMETER/TEST(,,,VTAM%USRID01:)'),
//         REGION=4M
```

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

z/OS limits JCL parameters to a total of 100 bytes. It is possible to run into a situation where adding a TEST option will make the total length exceed 100 bytes.

If the parameter string becomes too long, you have a couple of options. First, you can use a different method to start Debug Tool. Either use a CEEOPTS DD statement, or use the 'Debug Tool user exit data set' facility. Another option is to truncate the user portion of the string, and then add the missing data manually using the debugger after the program starts.

If the string becomes too long to fit on one line, it can be continued to the next line. An example is shown for the syntax used to code a long parameter string that spans multiple lines.



Coding a TEST option with a batch DB2 application

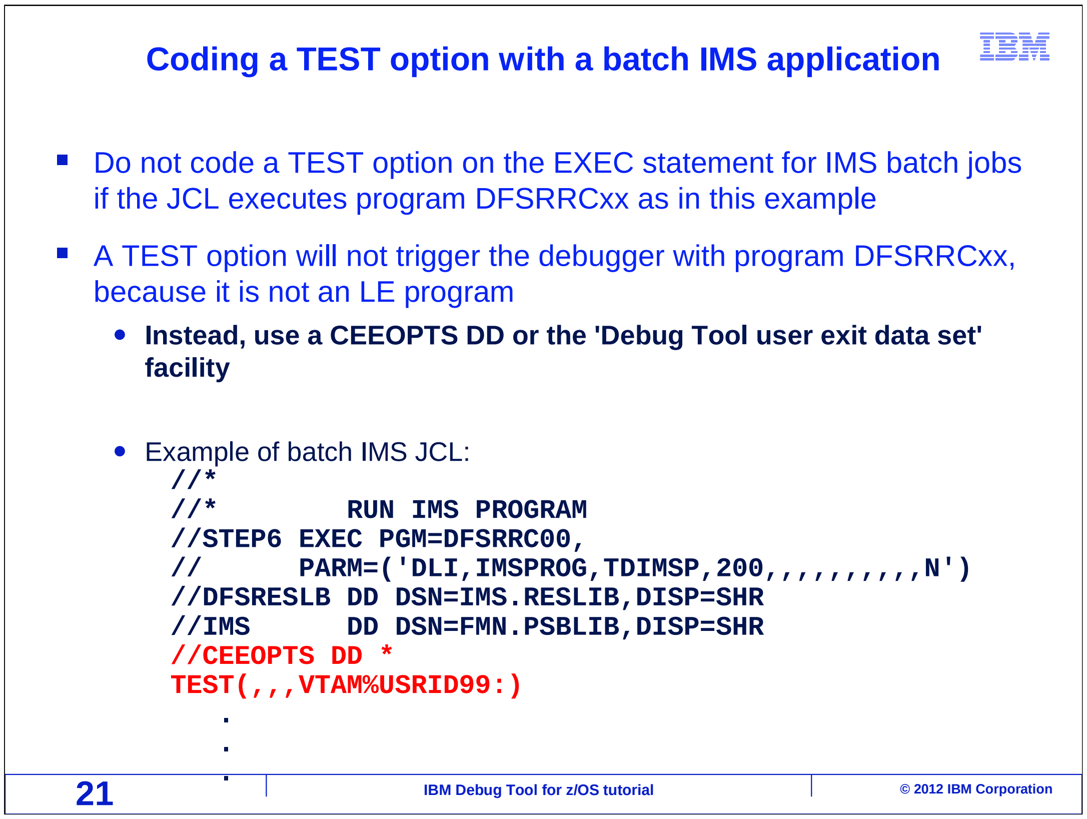
- Do not code a TEST option on the EXEC statement in DB2 batch jobs if the JCL executes program IKJEFTxx as in this example
- A TEST option will not trigger the debugger with program IKJEFTxx, because it is not an LE program
  - **Instead, use a CEEOPTS DD or the 'Debug Tool user exit data set' facility**
  - or optionally, the TEST option can be coded in the DB2 RUN parms:

```
//*         RUN DB2 PROGRAM
//STEP6 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD *
 DSN SYSTEM(DSNA)
 RUN  PROGRAM(PHONEP01) PLAN(PHONEP01) -
      PARMS('/TEST(,,,VTAM%USRID99:)')  -
      LIB('DNET603.ADLAB.LOAD')
 END
//*
```

COBOL example
(slash before TEST)

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

Do not code a TEST option on the EXEC statement when running a DB2 batch application that executes program IKJEFT01. The debugger will not be triggered, since that program is not an LE program. Instead, consider using a CEEOPTS DD. Or optionally, a TEST option can be coded in the run parms in SYSTSIN as in the example.



Coding a TEST option with a batch IMS application

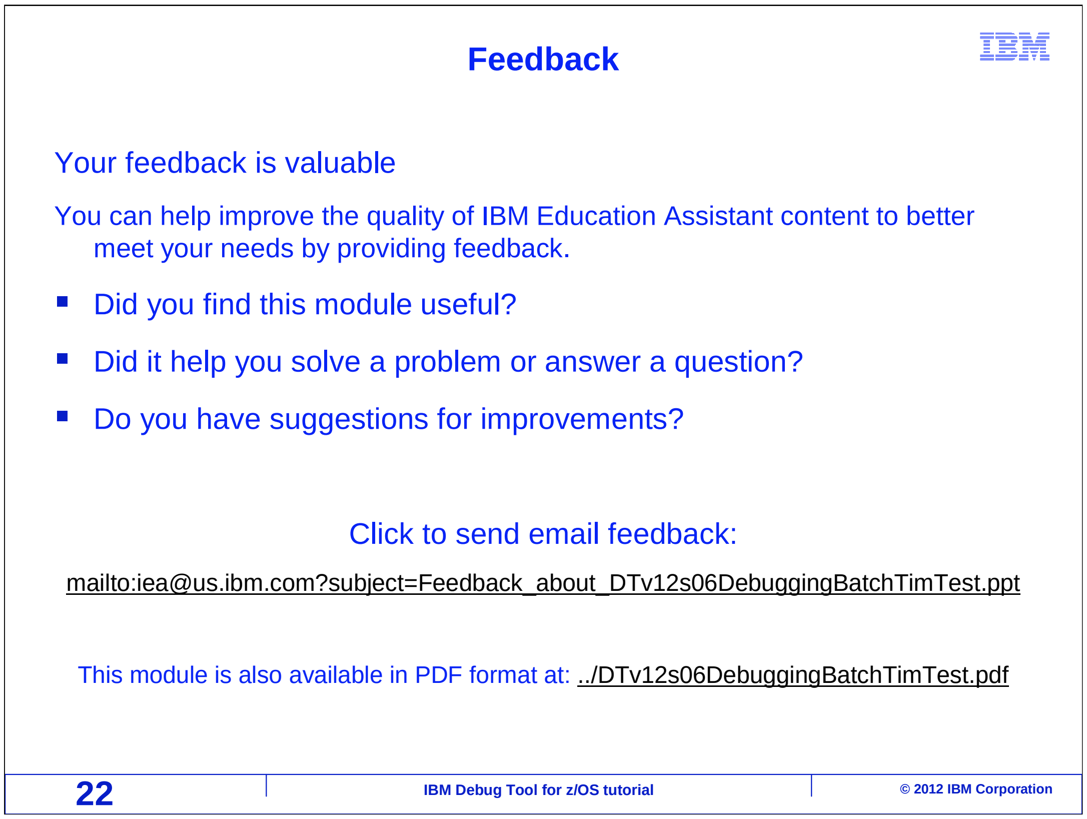
- Do not code a TEST option on the EXEC statement for IMS batch jobs if the JCL executes program DFSRRCxx as in this example
- A TEST option will not trigger the debugger with program DFSRRCxx, because it is not an LE program
  - **Instead, use a CEEOPTS DD or the 'Debug Tool user exit data set' facility**
  - Example of batch IMS JCL:

```
//*
//*         RUN IMS PROGRAM
//STEP6 EXEC PGM=DFSRRC00,
//      PARM=('DLI,IMSPROG,TDIMSP,200,,,,,,,,,,N')
//DFSRESLB DD DSN=IMS.RESLIB,DISP=SHR
//IMS      DD DSN=FMN.PSBLIB,DISP=SHR
//CEEOPTS DD *
TEST(,,,VTAM%USRID99:)
   .
   .
   .
```

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

Also, do not code a TEST option on the EXEC statement when running an IMS batch application that executes program DFSRRC00. The debugger will not be triggered, since that program is not an LE program. Instead, consider using a CEEOPTS DD statement or the 'Debug Tool user exit data set' facility.

That is the end of this section, which described starting the debugger for a program running in batch, triggering the debugger by coding a TEST option in JCL, and displaying the debugger on Debug Tool TIM terminal.



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_DTv12s06DebuggingBatchTimTest.ppt

This module is also available in PDF format at: ../DTv12s06DebuggingBatchTimTest.pdf

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

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

Trademarks, copyrights, and disclaimers