

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



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

In this section, you will see a scenario for starting Debug Tool for a batch application. In this scenario, the debugger displays on a Debug Tool terminal that is not controlled by the Debug Tool terminal interface manager. The application runs as a batch job, and a TEST option is coded in the JCL to trigger the debugger. 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 non-TIM Debug Tool 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 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 dedicated Debug Tool terminals are installed on your system
- **When not to use this method:**
 - 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
 - If dedicated Debug Tool terminals are not installed, use the 'Debug Tool setup file' and run under TSO instead

3

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

In this scenario, the debugger is displayed on a dedicated Debug Tool terminal. The terminal is not attached to the Debug Tool terminal interface manager (or TIM), so it can be referred to as a "non-TIM" 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.

If TIM terminal sessions are available on your system, it is generally better to use a TIM terminal, instead of this method. However, this is generally the best method when a non-TIM terminal is used. You cannot use this method if dedicated Debug Tool terminal sessions have not been defined on your system's network. If not, consider displaying the debugger on a GUI interface 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



The screenshot shows a terminal window titled "Session B - [24 x 80]" with a menu of options: 0 Settings, 1 View, 2 Edit, 3 Utilities, 4 Foreground, 9 IBM Products, 10 IBM Products, 11 IBM Products, 12 IBM Products, 13 IBM Products, 14 IBM Products, 15 IBM Products, 16 IBM Products, 17 IBM Products, 18 IBM Products, 19 IBM Products. System parameters are listed on the right: User ID: TSS16, Time: 23:38, Terminal: 3278A, Screen: 1, Language: ENGLISH, Appl ID: ISR, TSO logon: REDBOOK8, TSO prefix: TSS16, System ID: STLABF2, MVS acct.: ACCT#, Release: ISPF 5.9.

Callouts in the image:

- "In this example, the user is already logged on to TSO" (pointing to the terminal window)
- "Open a dedicated non-TIM terminal session for Debug Tool" (pointing to the "Debug Tool 3270" icon)
- "You may want to set up a desktop icon to start a 3270 session for Debug Tool" (pointing to the "Debug Tool 3270" icon)
- "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." (pointing to the "Debug Tool 3270" icon)

At the bottom of the screenshot, there is a slide number "4", the text "IBM Debug Tool for z/OS tutorial", and the copyright notice "© 2012 IBM Corporation".

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 non-TIM Debug Tool terminal



The screenshot shows a terminal window titled "Session B - [24 x 80]". The terminal displays the following text:

```
CONNECTED TO SYSTEM DEMOMVS
TERMINAL ID: TRMLU011
```

A callout box with an arrow points to the terminal ID "TRMLU011" and contains the text: "Note the terminal ID".

Another callout box contains the text: "A Debug Tool terminal on your system may look different than this one".

A third callout box contains the text: "DO NOT LOG ON to the non-TIM Debug Tool terminal".

On the right side of the terminal window, there is a list of system parameters:

```
ID . . : TSS16
. . . : 23:38
nal . . : 3278A
n . . . : 1
age . . : ENGLISH
ID . . : ISR
ogon . . : REDB00K8
refix . : TSS16
m ID . : STLABF2
cct . . : ACCT#
se . . . : ISPF 5.9
```

A red mouse cursor is pointing at the bottom right of the terminal window with the text "click".

At the bottom of the terminal window, there is a status bar with the text: "03/015" and "620Cse on LPT1:". Below the terminal window, there is a Windows taskbar showing the start button, several open applications (Microsoft PowerPoint, IBM WebSphere Dev...), and a system tray with a battery icon at 96%.

At the bottom of the slide, there is a footer with the number "5", the text "IBM Debug Tool for z/OS tutorial", and the copyright notice "© 2012 IBM Corporation".

That started a special 3270 terminal session for Debug Tool. Non-TIM Debug Tool terminal sessions have a unique appearance on every system. It may look different on your system than the example shown here. Make a note of the terminal ID, which may be displayed on the text of the screen, or in your terminal emulator's status bar. Do not log on to the Debug Tool terminal. If you log on to the terminal, it will be in use. And if it is in use, Debug Tool will not be able to connect to it.

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

Open the JCL that will run the batch program



Option ==> 2

Option	Description	System Information
0	Settings	Terminal and user parameters
1	View	Display source data or listings
2	Edit	Create or change source data
3	Utilities	Perform utility functions
4	Foreground	Interactive language processing
5	Batch	Submit job for language processing
6	Command	Enter TSO or Workstation commands
7	Dialog Test	Perform dialog testing
8	LM Facility	Library administrator functions
9	IBM Products	IBM program development products
10	SCLM	SW Configuration Library Manager
11	Workplace	ISPF Object/Action Workplace
D	DB2/DXT/QMF	Display DB2/DXT/QMF Selection Panel
R	Redbook	

Enter X

Open the JCL that will run the application in batch

Enter

04/016

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

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 length
Record Length _____	

Enter

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

Open the JCL in the editor.

This is the JCL before adding a TEST option



In the JCL, code a TEST option

```
File Edit Edit_Settings Menu Utilities Comp
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 //** //CEEOPST 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 //** //INSPREF DD DSN=&SYSUID..ADLAB.DTPREF,DISP=SHR
000018 //*****
000019 //STEPLIB DD DISP=SHR,DSN=&SYSUID..ADLAB.LOAD
```

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

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) - 01.01      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=SRM1,PARM='/TEST(,,MFI%TRMLU011:)',REGION=4M
000011 //***** OPTIONAL DD'S FOR DEBUG TOOL *****
000012 //CEEOPTS DD *
000013 TEST(,,MFI%TRMLU011:)
000014 //** //INSPLDG DD SYSOUT=*
000015 //** //EQADEBUG DD DSN=&SYSUID
000016 //** //          DD DSN=&SYSUID
000017 //** //INSPREF DD DSN=&SYSUID
000018 //*****
000019 //STEPLIB DD DISP=SHR,DSN=&SYSU
```

In the JCL, code a TEST option

MFI%terminal-id: denotes a dedicated Debug Tool terminal

Code a TEST option:
- on the EXEC statement, or
- in a CEEOPTS DD statement
(But not both)

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

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 "MFI%terminal ID:". MFI stands for "mainframe interface". The MFI keyword specifies that the debugger will connect to a dedicated Debug Tool non-TIM terminal identified by the terminal ID. Code the terminal ID after the % sign and before the required colon.

Submit the JCL to run the batch job



The screenshot displays the IBM Debug Tool for z/OS interface. The main window, titled 'Session A - [24 x 80]', shows a JCL job being edited. The command 'SUBMIT' is circled in red. The JCL code includes job information, dataset definitions, and program execution commands. A yellow callout box labeled 'Submit the job' points to the 'SUBMIT' command. Another yellow callout box labeled 'Enter' points to the end of the JCL code. The interface also shows a terminal window on the left with the message 'CONNECTED TO SYS' and 'TERMINAL ID: TRM'. The bottom of the screen features a Windows taskbar with various applications and a system tray showing 96% battery.

```
EDIT      TSS16D.ADLAB.JCL (XSAM) - 01.01          Columns 00001 00072
Command == SUBMIT
***** 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=IDCAM5
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 //***** OPTIONAL DD'S FOR DEBUG TOOL *****
000012 //CEEOPDS DD *
000013 TEST(,,MFI%TRMLU011;)
000014 //** //INSPLOG DD SYSOUT=*
000015 //** //LOADEBUG DD DSN=&SYSUID..ADLAB.SYSDEBUG,DISP=SHR
000016 //** //                DD DSN=&SYSUID..ADLAB.EQUALANGX,DISP=SHR
000017 //** //INSPREF DD DSN=&SYSUID..ADLAB.DTPREF,DISP=SHR
000018 //*****
000019 //STEPLIB DD DISP=SHR,DSN=&SYSUID..ADLAB.LOAD
```

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

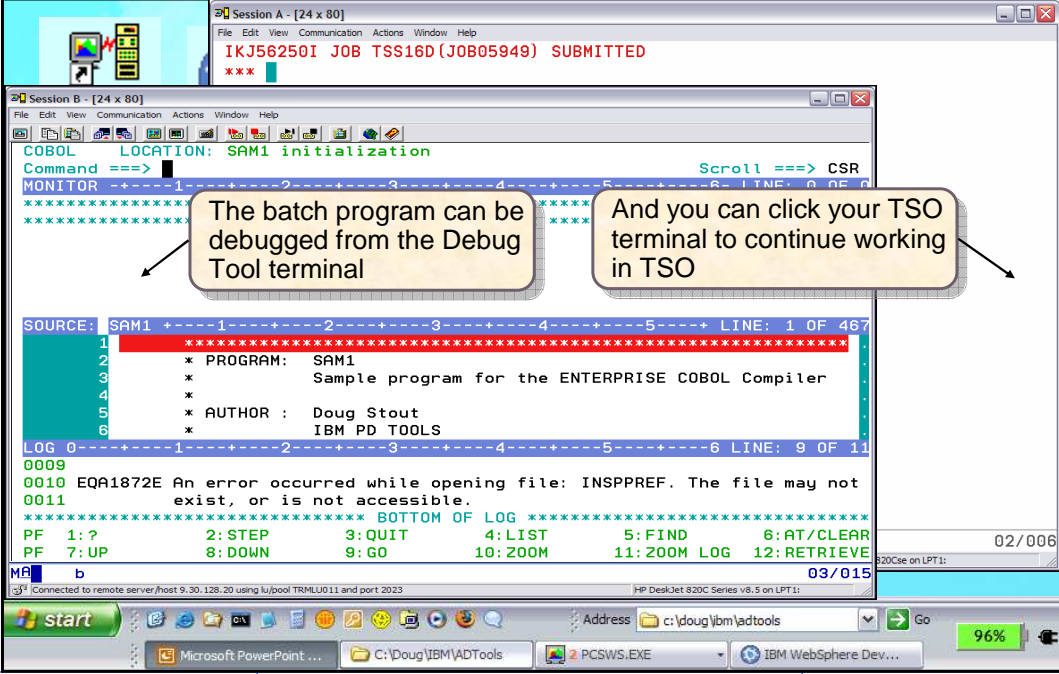
When the job step runs, the debugger is triggered



The screenshot displays the IBM Debug Tool for z/OS interface. At the top, a window titled 'Session A - [24 x 80]' shows the message 'IKJ56250I JOB TSS16D (JOB05949) SUBMITTED ***'. Below this, a 'Session B - [24 x 80]' window shows a COBOL program named 'LOCATION'. The program code includes a 'MONITOR' section with asterisks, a 'SOURCE' section with lines 1 through 6, and a 'LOG' section with lines 0009 through 0011. A callout box with a yellow background and black border points to the 'MONITOR' section, containing the text: 'When the step runs, the TEST option is processed, and Debug Tool is displayed on the Debug Tool terminal'. Another callout box with a yellow background and black border points to the 'LOG' section, containing the text: 'Select the Debug Tool terminal'. A red mouse cursor is positioned over the 'LOG' section, with the word 'click' written next to it. The bottom of the screenshot shows a Windows taskbar with several open applications, including 'Microsoft PowerPoint...', 'C:\Doug\IBM\ADTools', 'PCSWS.EXE', and 'IBM WebSphere Dev...'. The system tray shows the date '02/006' and a battery level of '96%'. The slide number '11' is in the bottom left corner, and the text 'IBM Debug Tool for z/OS tutorial' and '© 2012 IBM Corporation' are in the bottom right corner.

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 Debug Tool terminal session. The Debug Tool terminal emulator window is selected by clicking it.

The debugger automatically displays on the TIM terminal



12

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

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

13

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

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(,,,MFI%terminal-id:)
```

- where *terminal-id* = the Debug Tool terminal name
- MFI%*terminal-id*: directs the debugger to connect to the Debug Tool terminal by with that name (also known as VTAM® terminal name or LU name)

- 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(,,,MFI%terminal-id:)
```

15

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

Here are examples for 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 (put the slash *before* the LE options) :

```
//STEP5 EXEC PGM=COBPROG,  
// PARM= '/TEST( , , ,MFI%terminal-id: ) '
```

- For non-COBOL LE programs, code LE options before the first slash

PL/I, C/C++, LE Assembler (put the slash *after* the LE options) :

```
//STEP5 EXEC PGM=MYPROG,  
// PARM= 'TEST(ALL , , ,MFI%terminal-id: ) / '
```

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, **code LE options after the last slash:**

```
//STEP5 EXEC PGM=COBPROG,  
// PARM='ABC,1234/TEST(,,,MFI%terminal-id:).'
```

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

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

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(,,,MFI%TRMLU099:)',  
//      REGION=4M
```

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 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=*
//REPORT   DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DSNA)
RUN  PROGRAM(PHONEP01) PLAN(PHONEP01) -
      PARS(' /TEST(,,MFI%TRMLU005:)' ) -
      LIB('DNET603.ADLAB.LOAD')
END
//*
```

COBOL example
(slash before TEST)

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  
.  
.  
.
```

20

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 with a TEST option in JCL, and displaying the debugger on dedicated non-TIM Debug Tool terminal session.

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 DTv12s08DebuggingBatchNonTimTest.ppt](mailto:iea@us.ibm.com?subject=Feedback%20about%20DTv12s08DebuggingBatchNonTimTest.ppt)

This module is also available in PDF format at: [../DTv12s08DebuggingBatchNonTimTest.pdf](.../DTv12s08DebuggingBatchNonTimTest.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, DB2, IMS, VTAM, 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](http://www.ibm.com/legal/copytrade.shtml)" at <http://www.ibm.com/legal/copytrade.shtml>

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.

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