

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 application runs as a batch job, and the Debug Tool 'user exit data set' utility is used to set a trigger that will start the debugger when the job runs. The debugger displays on graphical user interface software running on your workstation. You can skip this section if you do not plan to use Debug Tool this way on your system.

Debug in batch using the 'user exit data set' facility and a GUI interface



- **Description**
 - The 'User exit data set' online panels in TSO are used to set a trigger for the debugger.
 - A special library may be required in STEPLIB or JOBLIB to enable the Debug Tool LE user exit
 - The debugger displays on GUI debugging software on your workstation
- **This method can be used:**
 - To debug Language Environment (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 compatible GUI debugging software is installed on your workstation
 - Such as Rational® Developer for System z®, or CICS Explorer® with the Debug Tool plug-in
- **When not to use this method:**
 - If GUI debugging workstation software is not available, use a terminal interface instead

3

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

This is a commonly used, simple method to start the debugger and display it on GUI debugging software on your workstation. You set a trigger, to start the debugger when the application runs, using the 'Debug Tool user exit data set' utility panels in TSO.

This method can be used to debug Language Environment (LE) conforming programs running in a batch job, including programs that access DB2, IMS, and other types of databases. Even non-LE subroutines can be debugged using this method, if there is at least one LE program higher in the call chain. Software such as Rational Developer for System z or CICS Explorer with the Debug Tool plug-in must be installed on your workstation.

You cannot use this method if compatible GUI debugging software is not available.

How to start Debug Tool for a batch job with the 'User exit data set' facility and a remote GUI interface

Start GUI debugging software on your workstation

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

You must have remote GUI debugging software installed, such as the Debug Tool plug-in for CICS Explorer, or Rational Developer for System z

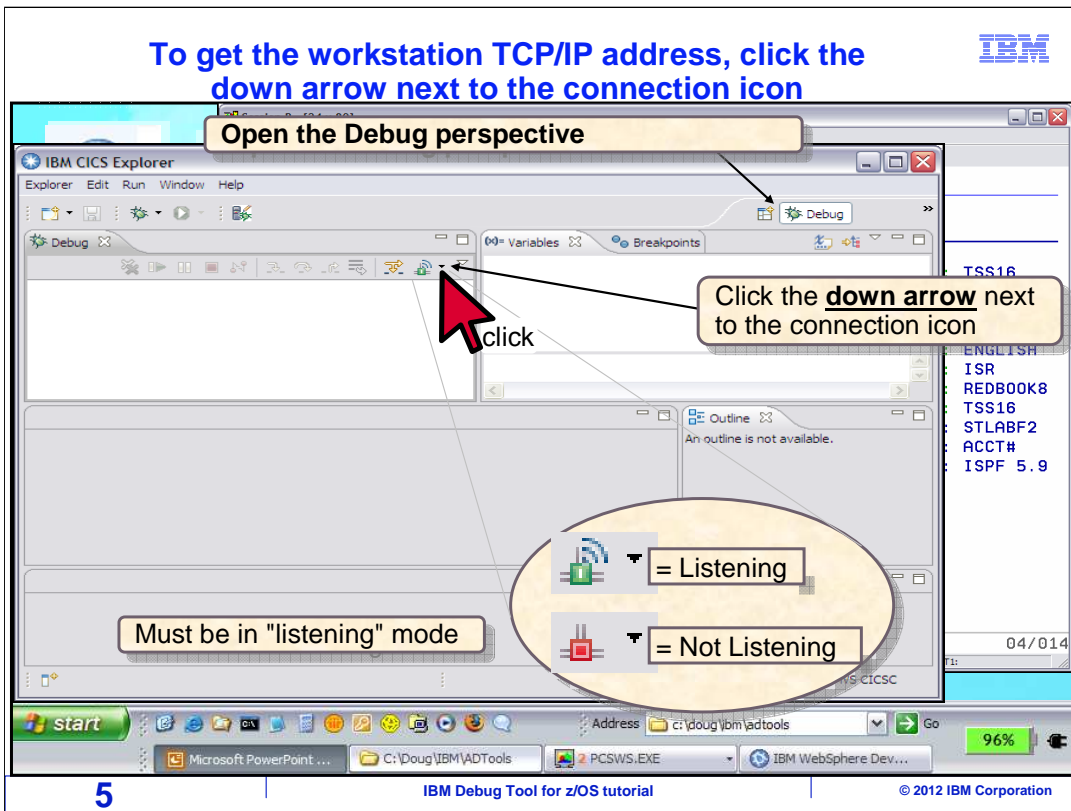
4

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

In this example, the user is already logged on to TSO. The first step is to start the remote GUI debugging software on your workstation. This could be software such as Rational Developer for System z or CICS Explorer with the Debug Tool plug-in.

Here, CICS Explorer is started.



That started CICS Explorer. If the "Debug" view is not displayed, switch to the Debug perspective. It must be in "listening" mode. There is an icon on the toolbar that shows the listening status. If it is not listening, the listener icon is red, and you can click the icon to switch it on. The icon is green when the listener is on.

You will need to determine your workstation's IP address. Click the small black down arrow next to the listening icon.

Click Get Workstation IP...



The screenshot shows the IBM CICS Explorer interface. A context menu is open over a debug message that reads "Debug UI daemon is listening on port: 8001". The menu options are "Stop listening", "Change Port...", and "Get Workstation IP...". A red mouse cursor is pointing at the "Get Workstation IP..." option. A callout box with the text "Click 'Get Workstation IP'" has an arrow pointing to the menu item. Another callout box with the text "This is the listener's TCP/IP port number. (8001 in this example)" has an arrow pointing to the port number "8001" in the debug message. On the right side of the interface, a list of system parameters is visible: TSS16, 23:38, 3278A, 1, ENGLISH, ISR, REDBOOK8, TSS16, STLABF2, ACCT#, ISPF 5.9. The Windows taskbar at the bottom shows the Start button, several application icons, and the system tray with a battery level of 96%.

6

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

That displays a menu. Make a note of the listener's IP port. In this example, it is 8001. Next, click "Get Workstation IP".

The TCP/IP address of your workstation is shown



IBM CICS Explorer

Workstation IP

IBM-4DC71712818

Workstation IP

9.76.135.133 AGN Virtual Network Adapter - AGN Filter Interface

192.168.1.110 Intel(R) 82567LM Gigabit Network Connection - AGN Filter Interface

OK

This is the TCP/IP address of the workstation. You will need this address and the port number to start the debugger.

TSS16
23:38
3278A
1
ENGLISH
ISR
REDBOOK8
TSS16
STLABF2
ACCT#
ISPF 5.9

7

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

That displays your workstation's IP address. Make a note of it. At this point, you have prepared the remote GUI debugging software, and it ready to receive a debugging session. Click OK to close the pop-up window. Now go back to TSO by clicking your TSO terminal session.

Navigate to the Debug Tool Utilities menu in TSO



Option ==>

0 Settings
1 View
2 Edit
3 Utilities
4 Foreground
5 Batch
6 Command
7 Dialog Test
8 LM Facility
9 IBM Products
10 SCLM
11 Workplace
D DB2/DXT/QMF
R Redbook

Follow the menus on your system to navigate to the Debug Tool panels

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

Enter X to Terminate using log/list defaults

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

In TSO, navigate to the Debug Tool ISPF panels. The person who installed Debug Tool on your system may have defined a menu option for you to get there.

Select the Debug Tool user exit data set utility option from the menu



The screenshot displays the IBM CICS Explorer interface. The main window shows the 'Debug Tool Utilities' menu with the following options:

- 0 Job Card
Create Job Card image.
- 1 Program Preparation
Convert, compile, assemble or link edit program.
- 2 Debug Tool Setup File
Manage setup files and start debug session in TSO foreground or batch.
- 3 Code Coverage
Measure code coverage in programs.
- 4 IMS TM Setup
Update Language Environment run-time options in IMS. Create message region.
- 5 Load Module Analyzer
Analyze load modules and each CSECT in the load module.
- 6 Debug Tool User Exit Data Set
Modify the data set used by user exit during program initial

Option 6 is circled in red, and a yellow 'Enter' button is positioned to its right. The interface also shows a status bar at the bottom with the text '9', 'IBM Debug Tool for z/OS tutorial', and '© 2012 IBM Corporation'.

The 'Debug Tool Utilities' menu is displayed. Select option 6 for the 'Debug Tool user exit data set' panels.

Specify the name of the Debug Tool user exit data set



```
----- Debug Tool - Manage TEST Run-time Option Data Set -----
Command ==> █

Specify the name of a TEST run-time option data set that you want to
create or edit.

Press Enter to edit the data set.
Press Exit or Cancel to exit.

The data set provides a TEST run-time option for debugging application
that uses Debug Tool Language Environment user exit.

Use Help and the Debug Tool User Guide section: Preparing a program
using Debug Tool Language Environment user exit, for more information.

Data Set Name:
Data Set Name . . . 'TSS16.DBGTOOL.EQUAUMPTS'
Volume Serial . . . (If not cataloged)

Enter

Use the default file name unless instructed
otherwise, or if a different user ID is needed
```

10

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

In this utility, you store your debugging profile in a file. The file name must follow a specific convention. By default, it is yourID.DBGTOOL.EQUAUMPTS.

If you have not used this panel before, it will automatically fill in a file name in the 'data set name' field. Use the default name unless you have been instructed otherwise by your system programmer. The naming convention may have been customized for your system, and the convention must be followed for this feature to work. Later, when you run your batch job, a Debug Tool system exit will look for a file according to the convention.

Press Enter. If the file does not already exist, it will be created automatically.

If the Debug Tool user exit data set does not already exist, this allocation panel is displayed

```
----- Debug Tool - Allocate TEST Run-time Option Data Set-----
Command ==> _____

Specify allocation parameters for the new setup file:
TSS16.DBGTOOL.EQAUOPTS
Press ENTER to create Setup file and continue with edit
Press CANCEL or EXIT to cancel

New Setup File Dataset Allocation Parameters
Management class . . . . . _____ (Blank for default management class)
Storage class . . . . . _____ (Blank for default storage class)
Volume serial . . . . . _____ (Blank for system default volume) **
Device type . . . . . _____ (Generic unit or device address) **
Data class . . . . . _____ (Blank for default data class)
Space units . . . . . _____ (BLKS, TRKS, CYLS, KB, MB, BYTES
or RECORDS)
Average record unit _____ (M, K, or U)
Primary quantity . . . 2 _____ (In above units)
Secondary quantity . . 2 _____ (In above units)
Directory blocks . . . 0 _____ (Zero for sequential data set)
Record format . . . . . _____ (FB or VB)
Record length . . . . . _____
Block size . . . . . _____
```

More: +

Use the default settings, unless required by your system

Enter

11

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

In this example, the file did not exist, so the file allocation panel is displayed where you can specify attributes for the new file. On many systems, you do not need to change anything and you can take the defaults. Press Enter to continue.

Specify one or more program names



```
----- Debug Tool - Edit TEST Run-time Option Data Set -----
Command ==>  More: +
Enter test program names: (* is a valid wild card, by itself or as the last
character of a name)
Name 1: * Name 2: Name 3: Name 4:
Name 5: Name 6: Name 7: Name 8:
Enter IMS identifiers: (only valid IMS Subsystem ID: IMS Tra
Select Test Options:
Test Option ==> TEST Test/Notest
Test Level ==> ALL All/Error/None
Commands File ==> *, DDname, or Data Set Name
Prompt Level ==> * Prompt, NoPrompt, ;, *, command
Preference File ==> PROMPT *, DDname, or Data Set Name
EQAOPTS File ==> * Data Set Name or blank
```

* = any program

Specify the names of **main** programs that should trigger the debugger when they run

Specify the debugging profile

12

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

Now the Debug Tool user option data set has been created, and the panel titled: 'Edit test run-time option data set' is displayed. Here you specify the name or names of the main programs that should trigger the debugger when they run.

Only the highest level LE program, typically the main program, will trigger the debugger. There is no reason to specify the names of lower level programs or subprograms. You do not need to specify all of the subprograms that you want to debug. In fact, entering subprogram names on this panel will have no effect at all. After the debugger has triggered, you will be able to use debugging commands to tell the debugger to stop at any subprogram you want to see.

Notice that you can specify an asterisk in the program name, which specifies that the debugger will trigger for the first LE program that runs in the call chain, regardless of its name. This can be useful if you do not know the name of the highest level LE program, or if you want to use the same debugging profile with different programs.

Specify the name of one or more main programs that will trigger the debugger when they run

----- Debug Tool - Edit TEST Run-time Option Data Set -----

Command ==> More: +

Enter test program names: (* is a valid wild card, by itself or as the last character of a name)

Name 1: SAM1 Name 2: ABC* Name 3: _____ Name 4: _____
Name 5: _____ Name 6: _____ Name 7: _____ Name 8: _____

Enter IMS identifiers: (only valid
IMS Subsystem ID: _____ IMS Tra

Select Test Options:

Test Option	==> <u>TEST</u>	Test/Notest
Test Level	==> <u>ALL</u>	All/Error/None
Commands File	==> <u>*</u>	*, DDname, or Data Set Name
Prompt Level	==> <u>PROMPT</u>	Prompt, NoPrompt, ;, *, command
Preference File	==> <u>*</u>	*, DDname, or Data S
EQA0PTS File	==> <u>*</u>	Data Set Name or blank

F8

Specify the names of **main** programs that should trigger the debugger when they run

Specify the debugging profile

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

In this example, SAM1 is specified as the name of the main program that will trigger the debugger when it runs. You can specify up to eight program names. Notice that you can enter a partial program name using an asterisk as a wildcard. Scroll down with the F8 key.

Specify the debugging interface and connection information



```
----- Debug Tool - Edit TEST Run-time Option Data Set -----
Command ==> _____ More: -
                ==> PROMPT
Preference File ==> _____ *, DDname, or Data Set Name
                ==> *

Select (/) a session type and provide parameters:

- Full-screen mode
  Network name ==> _____ Blank or Dedicated terminal Network name
  LU name      ==> _____ Dedicated terminal LU name

- Full-screen mode using the Debug Tool Terminal Interface Manager
  User ID     ==> DNET074 User ID

/ Remote debug mode
  Address    ==> 9.76.135.133
  Port      ==> 8001

Other run-time options: _____
```

Select the Remote debug mode option

When it triggers, the debugger will connect to the workstation at this address

Specify the debugging profile

Enter

14

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

Since a graphical interface is being used, select the option named "Remote debug mode" with a slash. Specify the IP address of your workstation in the "address" field. Also specify the "port" field. These are the TCP/IP address and port number where the GUI debugging software is listening. That is all that is needed. Press Enter.

The user exit data set was updated by pressing Enter



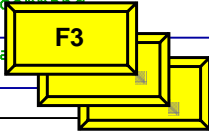
```
----- Debug Tool - Edit TEST Run-time Option Data Set -----
Command ==> █
                                                                 More:  +
Enter test program names: (* is a valid wild card, by itself or as the last
character of a name)

Name 1: SAM1      Name 2: ABC*      Name 3: _____ Name 4: _____
Name 5: _____ Name 6: _____ Name 7: _____ Name 8: _____

Enter IMS identifiers: (only valid for IMS user exit)
IMS Subsystem ID: _____ IMS Transaction ID: _____

Select Test Options:

Test Option   ==> TEST                Test/Notest
Test Level    ==> ALL                All/Error/None
Commands File ==> *                        *, DDname, or Data Set Name
Prompt Level  ==> *                        Prompt, NoPrompt, ;, *, or Command
Preference File ==> PROMPT                *, DDname, or Data Set Name
EQA0PTS File  ==> *                        Data Set Name or blank
```



15

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

At this point, the name of the program that will trigger Debug Tool, and the location of the debugging interface have been specified. Press F3 several times to save the options in the user exit data set and exit from the Debug Tool utilities.

Next, open the JCL that will run the program in batch



```
File Edit Edit_Settings Menu Utilities Compilers Test Help
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(CUST
000007 //SYSIN DD *
000008 PRINT INFILE(FILE) COUNT(1)
000009 //*
000010 //RUNSAM1 EXEC PGM=SAM1,REGION=4M
000011 //STEPLIB DD DISP=SHR,DSN=&SYSUID..ADLAB.LOAD
000012 //          DD DISP=SHR,DSN=DEBUG.TOOL.BATCH.EXITLIB
000013 //***** OPTIONAL DD'S FOR DEBUG TOOL *****
000014 /*** //CEE0PTS DD *
000015 /*** TEST(,,VTAM%TSS16
000016 /*** //INSPLOG DD SYSO
000017 /*** //EQADEBUG DD DSN=
000018 /*** //          DD DSN=
000019 /*** //INSPREF DD DSN=
```

If needed, add a special library to STEPLIB or JOBLIB

Depending on how your program was linked, you may need to add a special Debug Tool library to STEPLIB or JOBLIB. The name of the library will be provided by your system programmer.

Next, open the JCL that you will use to run your batch application in the editor. On some systems, you may need to add a special Debug Tool library to the STEPLIB or JOBLIB DD. This library enables the system exit that reads the 'User exit data set' to determine whether to start the debugger. If this library is needed, get its name from your system programmer or help desk.

Submit the JCL to run the batch job



Session A - [24 x 80]

File Edit View Communication Actions Window Help

IBM CICS Explorer

Command ==> **SUBMIT** Columns 00001 00072 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 //STEPLIB DD DISP=SHR, DSN=&SYSUID..ADLAB.LOAD
000012 //          DD DISP=SHR, DSN=DEBUG.TOOL.BATCH.EXITLIB
000013 //***** OPTIONAL DD'S FOR DEBUG TOOL *****
000014 //** //CEEOPTS DD *
000015 //** TEST(,, ,VTAM%TSS16:)
000016 //** //INSPLOG DD SYSOUT=*
000017 //** //EQADEBUG DD DSN=&SYSUID..ADLAB.SYSDEBUG, DI
000018 //** //          DD DSN=&SYSUID..ADLAB.EQUALANGX, DISP=SHR
000019 //** //INSPREF DD DSN=&SYSUID..ADLAB.DTPREF, DISP=SHR
```

Submit the job

Enter

04/021

Connected to remote server/host 9.30.128.24 using lu.pool.TCP00016 and port 23

Print to Disk - Append

DemoMVS CICSC

start

Address c:\doug\ibm\adtools Go

96%

Microsoft PowerPoint ... C:\Doug\IBM\ADTools PCSWS.EXE IBM WebSphere Dev...

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

Submit the JCL to run the batch job.

When the job step runs, the debugger is triggered



When the step runs, the Debug Tool LE user exit is invoked. The exit opens your user exit data set. If the main program matches any of the program entries in the profile, the debugger is triggered and is displayed in the GUI interface

Select the GUI interface

click

18

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

When the step that was modified in the batch job runs, the Debug Tool LE user exit runs automatically. The exit looks for your Debug Tool user exit data set. If the data set does not exist, no action is taken and the step runs normally. If the data set is found, it is read. If the highest level LE program does not match any of the program names in the profile, the step runs normally. But if there is a match on the program name, the exit automatically formats and passes a TEST option to Language Environment to trigger the debugger. The debugger is displayed by the GUI debugging software.

The debugging software is selected by clicking it.

The debugger automatically displays



The batch program can be debugged from the GUI interface

And you can click your TSO terminal to continue working in TSO

19

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 over the network with the GUI software. Notice that you can control the program from the GUI debugger, but 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.

How the Debug Tool user exit data set facility works



How it works:

- The user exit data set utility saves your debugging profile in a 'user exit data set'
 - By default, the naming convention is: user-ID.DBGTOOL.EQUAUMPTS
 - The system programmer can customize the naming convention on your system. If that is the case:
 - You need to learn the naming convention
 - Ensure that you name your user exit data set correctly
- Debug Tool provides special LE exits. If an exit is enabled when the program runs:
 - It uses the user ID to determine the name of your data set
 - The exit reads your user option data set
 - If the main program name matches any of the programs in the profile, the exit passes a TEST option to LE to start the debugger

That was an example of how to start Debug Tool for a batch application using the Debug Tool 'User Exit data set' utility and remote GUI debugging software. At this point, you may be comfortable with the process, and have all of the information you need to start the debugger. The rest of this section goes into technical details about how it works. You can opt to skip the rest of this section, if you are not interested in the details.

Here is how it works. The 'User exit data set' utility saves your Debug Tool profile in a file. The test option file has to be named using a specific naming convention. By default, user ID.DBGTOOL.EQUAUMPTS. However, your systems programmer can customize the naming convention. Use the default naming convention unless you have been instructed otherwise by your system programmer.

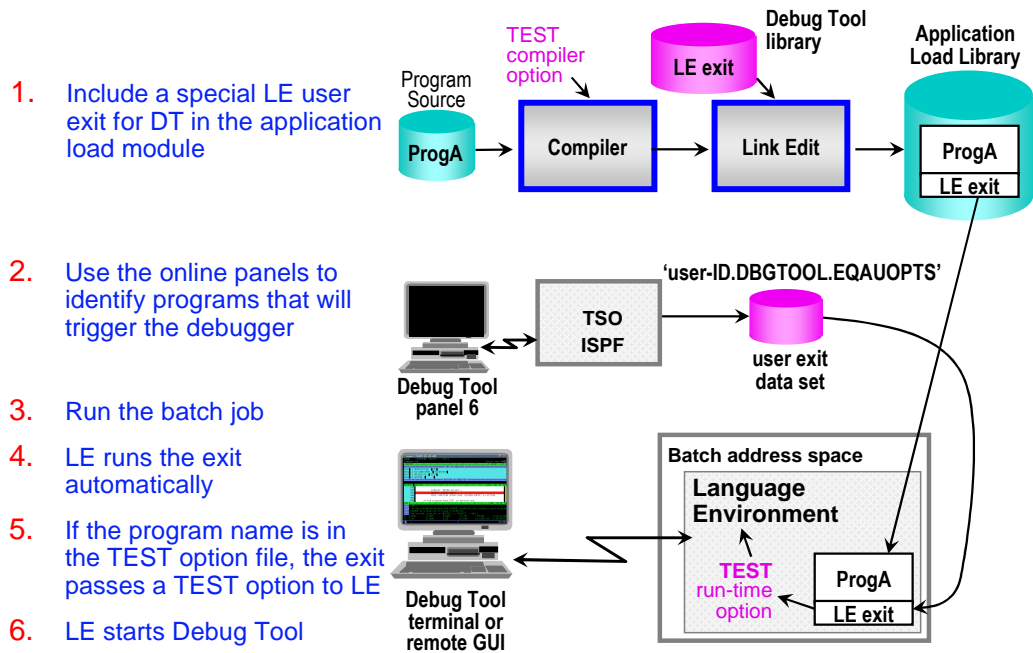
Debug tool provides special exits for Language Environment. If one of these exits is enabled when your program runs, it uses the user ID of the job to determine the user ID portion of the file name. The exit program then reads your test option data set. If it finds a match on the program name in your profile, the exit passes a TEST option to Language Environment, which causes the debugger to start.

- There are two ways to enable a Debug Tool LE exit program
 - Option 1: A supplied Debug Tool exit is link-edited into the application load module
 - Option 2: A supplied Debug Tool exit is made available in STEPLIB or JOBLIB in run-time JCL
 - The system programmer must set up a special library that Debug Tool users can add to their STEPLIB or JOBLIB concatenation

There are two ways to enable a Debug Tool LE exit program. The first way is to link-edit one of the supplied Debug Tool exits into an application load module. Then it will be enabled automatically whenever the program runs. The second way is to make the exit available by adding a special library to STEPLIB or JOBLIB of the run-time JCL. The system programmer must set up this special library so that Debug Tool users can use it.

Enabling the user exit data set utility - how it works

Option 1: The exit is linked into application load module



1. Include a special LE user exit for DT in the application load module
2. Use the online panels to identify programs that will trigger the debugger
3. Run the batch job
4. LE runs the exit automatically
5. If the program name is in the TEST option file, the exit passes a TEST option to LE
6. LE starts Debug Tool

22

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

This chart shows how the supplied LE exit works if your organization has chosen to link edit the exit into your application load modules. In the first step, the LE exit is included into your application load module when the program is compiled and linked. This could be made part of the standard compile process for all of your batch programs.

In the second step, the Debug Tool 'User exit data set' panels are used to identify the name of the programs that will trigger the debugger, and the terminal or workstation where the debugger will be displayed. These settings are stored in the file. In the third step, the job is submitted.

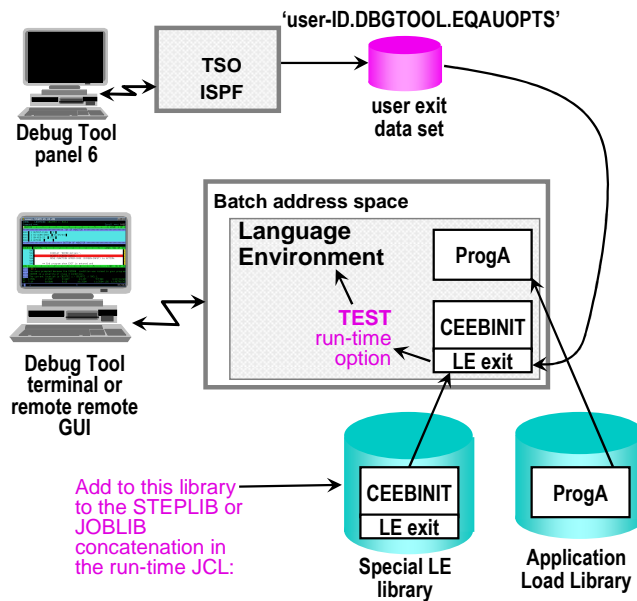
When the program runs, Language Environment automatically runs the exit program because it was included in the load module. This capability is a normal part of Language Environment initialization. The exit uses the user ID of the job to determine the name of the user exit data set. The exit opens the file and reads the options stored there. If there is match on the program name, then the exit passes a TEST option to Language Environment, and Language Environment starts Debug Tool.

Enabling the user exit data set utility - how it works

Option 2: STEPLIB or JOBLIB to the exit



1. The system programmer prepares a special library with the exit
2. Use the utility to identify programs that will trigger the debugger
3. Add the special library to STEPLIB in run-time JCL
4. Run the batch job
5. LE runs the exit automatically
6. If the program name is in the TEST option file, the exit passes a TEST option to LE
7. LE starts Debug Tool



23

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

This chart shows how the supplied LE exit works if your organization has chosen NOT to link edit the exit directly into your application load modules, but instead has chosen to make the exit available by adding it to your STEPLIB concatenation. In the first step, the system programmer prepares a special library that contains the exit module.

In the second step, the Debug Tool 'User exit data set' panels are used to identify the name of the programs that will trigger the debugger, and the terminal or workstation where the debugger will be displayed. These settings are stored in the file.

In the third step, the JCL to run the job is edited. The special load library is added to the STEPLIB or JOBLIB concatenation. In the fourth step, the modified JCL is submitted. When the program runs, Language Environment automatically runs the exit program that was made available in STEPLIB or JOBLIB. The exit uses the user ID of the job to determine the name of the user exit data set. The exit opens the file and reads the options stored there. If there is match on the program name, then the exit passes a TEST option to Language Environment, and Language Environment starts Debug Tool.

Enabling the user exit data set utility - how it works



Option 2: STEPLIB to the exit

- Example:

- Note: Get the name of the library from your system programmer

```
//STEP5 EXEC PGM=MYPROG
//STEPLIB DD DSN=MY.PROGRAM.LIB,DISP=SHR
//          DD DSN=DBGTOOL.BATCH.LOADLIB,DISP=SHR
```

- Tip: You can override STEPLIB in a step that is in a PROC without changing the PROC

- This example adds or overrides the third library in the STEPLIB concatenation of the step named ASTEP in PROC99

```
//RUN99 EXEC PROC99
//ASTEP.STEPLIB DD
//          DD
//          DD DSN=DBGTOOL.BATCH.LOADLIB,DISP=SHR
```

Here are examples of coding your JCL if you are using the method where you add a special library to STEPLIB or JOBLIB. In the first example, the library is added to the existing STEPLIB concatenation.

That is fine, if your JCL is self-contained. However, your JCL may run a JCL PROC, and the STEPLIB that you need to modify may be hard coded in the PROC. If that is the case, you can code JCL as shown in the second example. That example assumes that STEPLIB is already coded in a step named ASTEP in a proc named PROC99. In this example the first two DD statements in the overridden STEPLIB concatenation remain unchanged, and a third library is either added or overridden.

Comparing the two ways to enable the user exit data set utility



- In option 1, a special Debug Tool module is link-edited into the application load module
 - Advantage:
 - No JCL changes are needed at run-time to start Debug Tool !
 - Considerations:
 - The exit module always runs once each time the main program starts, but incurs negligible overhead when not debugging
 - It may be preferred, but not technically necessary, to remove the exit module before promoting to production
- In option 2, the exit is made available at run-time in STEPLIB or JOBLIB
 - Advantages:
 - No change to the compile process (to link the exit into the application module)
 - You can control when the exit runs with JCL
 - Consideration:
 - Run-time JCL must be modified to start Debug Tool

Two options have been described for enabling the Language Environment exit that Debug Tool supplies. If it is up to you to choose which option to use, here is a comparison.

In option one, the supplied Language Environment exit is link edited directly into the application load module. The advantage to this method is that no JCL changes are needed at runtime to start the debugger. You just submit your JCL without any modifications whatsoever. You control whether to start the debugger strictly by specifying program names in the 'user exit data set' panel. A consideration for this method is that the exit will always run once each time a main program is executed, although this incurs negligible overhead. Even so, it may be preferred to remove the exit module before promoting the application module to production, to eliminate any possible risk of inadvertently triggering the debugger during a production run.

In option 2, the exit is made available at run time by adding a special library to STEPLIB or JOBLIB in your JCL. The advantages of this option are that no changes are needed to your compilation processes, and you can control when the exit runs with changes to your JCL. The consideration is that your JCL requires this simple modification before you can trigger the debugger.

Enabling the user exit data set utility A deep-dive into the details



- Debug Tool provides four versions of LE exit programs:
 - EQAD3CXT: a single exit that can be used for all LE batch, IMS/TM, and DB2 SPs
 - EQADBCXT: can be used for batch LE applications only (including IMS and DB2 batch)
 - EQADICXT: can be use for online IMS (IMS/TM) LE applications only
 - EQADDCXT: can be used for DB2 stored procedures only
 - Installation and use are described in the Debug Tool User's Guide
 - Source for the exits (in library SEQASAMP) can be customized to:
 - Change the naming convention of the TEST option data set and produce diagnostic messages
- To enable them:
 - Option 1: Link the appropriate version of the exit into each application load module, or
 - Option 2: The system programmer can create 3 new load libraries, which can be added to STEPLIB or JOBLIB to enable the exits
 - A loadlib for batch applications containing a copy of LE module CEEBINIT linked with exit module EQAD3CXT or EQADBCXT
 - A loadlib for IMS/TM applications containing a copy of LE module CEEBINIT linked with exit module EQAD3CXT or EQADICXT
 - A loadlib for DB2 stored procedures containing a copy of LE module CEEPIPI linked with exit module EQAD3CXT or EQADDCXT

If you are curious about exactly how to implement the supplied Language Environment exits, here is a deep-dive into the details. Debug Tool provides three exit programs that work with the 'user exit data set' utility. A module named EQADBCXT is used to start Debug Tool for **batch** Language Environment applications. A module named EQADICXT can be used to start Debug Tool for **online IMS/TM applications**, and a module named EQADDCXT is used to start Debug Tool for **DB2 stored procedures**. These modules are shipped in Debug Tool library SEQAMOD. Source code for the exits is also shipped with Debug Tool in the SEQASAMP sample library. The source code can be modified and assembled to change the naming convention for user exit data sets, and to request that the exit produce messages as an aid to the installer.

To enable the exits, one option is to link edit the appropriate version of the exit into each application load module. The exit is enabled because it is present in the module. The second option to enable the exits is to create new load libraries, which the Debug Tool user can add to their STEPLIB or JOBLIB. First, create a load library for batch applications. It will contain module CEEBINIT, from the Language Environment SCEERUN library, with the supplied exit module EQADBCXT link-edited into it. Also, create a load library for IMS/TM applications. It will contain module CEEBINIT, from the Language Environment SCEERUN library, with the supplied exit module EQADICXT link-edited into it. Finally, create a load library for DB2 stored procedure programs. It will contain module CEEPIPI, from the Language Environment SCEERUN library, with the supplied exit module EQADDCXT link-edited into it.

That is the end of this section, which described a program running in batch, triggering the debugger by using the 'user exit data set' utility, and displaying the debugger on a remote GUI interface.

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_DTv12s05DebuggingBatchGUIExit.ppt

This module is also available in PDF format at: [../DTv12s05DebuggingBatchGUIExit.pdf](http://DTv12s05DebuggingBatchGUIExit.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, CICS, CICS Explorer, DB2, IMS, Rational, System z, 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.