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

IBM

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 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 a Debug Tool terminal interface manager terminal session that you connect to directly. 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 dedicated Debug Tool TIM terminal

- 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 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 the TIM is not installed, use a dedicated non-TIM terminal instead

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

This is a commonly used, simple method used to start the debugger. The debugger is displayed on a Debug Tool terminal interface manager, or TIM, terminal. 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.

If Debug Tool terminal interface manager sessions have not been installed on your system's network, consider using a non-TIM terminal or a GUI interface instead, which are described in other sections.

How to trigger Debug Tool for a batch job

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.

dbl click

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 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

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

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 TIM 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
Edit your LE options data set

At this point your Debug Tool terminal is ready to receive a debugging session. Next, press F10 to edit your debugging profile data set.

Specify the name of the Debug Tool user exit data set

IBM

```
                    DEBUG TOOL TERMINAL INTERFACE MANAGER
                   * Specify TEST Run-time Option Data Set *


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

Data Set Name . . . 'TSS16.DBGTOOL.EQAUOPTS'
Volume Serial . . . █_____  (If not cataloged)
```

Specify the correct name of your user option data set. There is a specific naming convention that must be used on each system.

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

Enter

```
              PF1=Help   ENTER=Process   PF12=Cancel
```

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

You will store your debugging profile in a file. The file name must follow a specific convention. By default, it is yourID.DBGTOOL.EQAUOPTS.

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. The naming convention may have been customized for your system, and the convention must be followed exactly 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 run-time option data set does not already exist, this allocation panel is displayed

```
                   DEBUG TOOL TERMINAL INTERFACE MANAGER
                   * Allocate TEST Run-time Option Data Set *
Data set name: 'TSS16.DBGTOOL.EQAUOPTS'

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_____      (BLKS, TRKS, CYLS, KB, MB, BYTES
                                           or RECORDS)
  Average record unit .  _               (M, K, or U)
  Primary quantity  . .  20_____       (In above units)
  Secondary quantity  .  20_____       (In above units)
  Directory blocks  . .  0_____         (Zero for sequential data set)
  Record format . . . .  FB____           (FB or VB)
  Record length . . . .  80____
  Block size  . . . . .  32720_
  Data set name type  .
                                    ┌──────────────────────┐
( ** Only one of these fields may be│ Use the default settings,  │    ┌────────┐
                                    │ unless changes are         │    │ Enter  │
                                    │ required by your system    │    └────────┘
                                    └──────────────────────┘
                 PF1=Help   ENTER=Create file   PF12=Cancel
```

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.

A panel is displayed where you specify your debugging profile

```
                 DEBUG TOOL TERMINAL INTERFACE MANAGER
                   * Edit   TEST Run-time Option Data Set *
    Enter test program names:

     Program name(s) ==> █_____  ==> _____  ==> _____  ==> _____
                      ==> _____  ==> _____  ==> _____  ==> _____


    Select Debug Tool options

     Test Option      ==> TEST____   Test/Notest
     Test Level       ==> ALL_____   All/Error/None          ⤷
     Commands File    ==> _____
     Prompt Level     ==> PROMPT_____
     Preferences File ==> _____


     EQAOPTS File     ==> _____


     Other run-time options:
     ==> _____


    Current debug display information:
      VTAM%TSS16


              PF1=Help  PF4=Save  PF9=Change Display Info  PF12=CANCEL
```

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

Now the run-time option data set has been created, and the panel titled: 'Edit test run-time option data set' is displayed, where you can specify your debugging profile.

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

```
                    DEBUG TOOL TERMINAL INTERFACE MANAGER
                    * Edit   TEST Run-time Option Data Set *
 Enter test program names:
                                            Specify the names
                                            of main programs
 Program name(s) ==> SAM1     ==> ABC*     ==> _____  ==> _____
                 ==> _____  ==> _____ ==> _____  ==> _____

 Select Debug Tool options                 * = wildcard
                                            Note: * can be used by itself
 Test Option     ==> TEST      Test/Notest
 Test Level      ==> ALL       All/Error/None
 Commands File   ==> _____
 Prompt Level    ==> PROMPT_____
 Preferences File ==> _____

 EQAOPTS File    ==> _____

 Other run-time options:
 ==> _____

 Current debug display information:              F9
   VTAM%TSS16

          PF1=Help  PF4=Save  PF9=Change Display Info  PF12=CANCEL
```

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

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 names of lower level programs or subprograms that you want to debug. In fact, entering subprogram names on this panel will not have any effect at all. After the debugger starts, you will be able to enter commands to tell the debugger to stop at any subprogram you want to see.

 For any individual batch job step, there is only one main program. However, you can use the same profile to trigger different jobs by specifying more than one main program name. You can specify an asterisk as a wildcard as the last character in the program name. If you specify an asterisk as the full name by itself, any time your debugging profile is active, the highest level LE program will trigger the debugger, regardless of its name.

  Change the 'test option' field to NOTEST to turn the debugging profile off, or leave it as TEST if you want the debugger to trigger. You can specify the name of a file that contains a debugging script in either the 'commands file' or 'preferences file' fields. A script is a file that contains a series of debugger commands that you want to run automatically when the debugger starts.

  Notice the line near the bottom of the panel that reads 'current debug display information'. In this example, under that line is the setting 'VTAM%TSS16', which indicates that the debugger will display on a TIM terminal identified by user ID TSS16. If you want to change where the debugger will display, press F9.

You can control where the debugger is displayed

```
              DEBUG TOOL TERMINAL INTERFACE MANAGER
                      * Change session type *

 Select (/) a session type and provide parameters:

  / Full-screen mode using the Debug Tool Terminal Interface Manager

    User ID        ==> TSS16     User ID

  _ Remote debug mode

    Address        ==> _____
    Port           ==> _____



                    PF1=Help   PF4=Save   PF12=Cancel
```

By default, the debugger will display on a TIM terminal identified by your ID

Specify a different ID if you want the debugger to appear on a colleague's TIM terminal.

Or select "remote" and your workstation's IP address for the graphical user interface.

F4

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

The 'change session type' panel gives you options to control where the debugger will be displayed when it triggers. You can send it to your TIM terminal by leaving the defaults: the 'Full-screen mode' option is selected and your ID is in the 'user ID' field. You can work with a colleague who has logged on to their own Debug Tool terminal. Enter their ID, and the debugger will display on their TIM terminal when it starts.

You can optionally specify that the debugger should display on a graphical user interface by selecting 'remote debug mode', and the IP address and port number of your workstation. This option assumes that you have software installed on your workstation that provides the graphical user interface, such as the Debug Tool plug-in for CICS Explorer®.

Press F4 to save your changes and return.

That returns to the 'edit test run-time option data set' panel. Press F4 again to save your profile.

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.

```
  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 //**   //CEEOPTS  DD *
 000015 //**   TEST(,,,VTAM%TSS16;)
 000016 //**   //INSPLOG  DD SYSO
 000017 //**   //EQADEBUG DD DSN=
 000018 //**   //         DD DSN=
 000019 //**   //INSPPREF 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.

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

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

Submit the JCL to run the batch job.
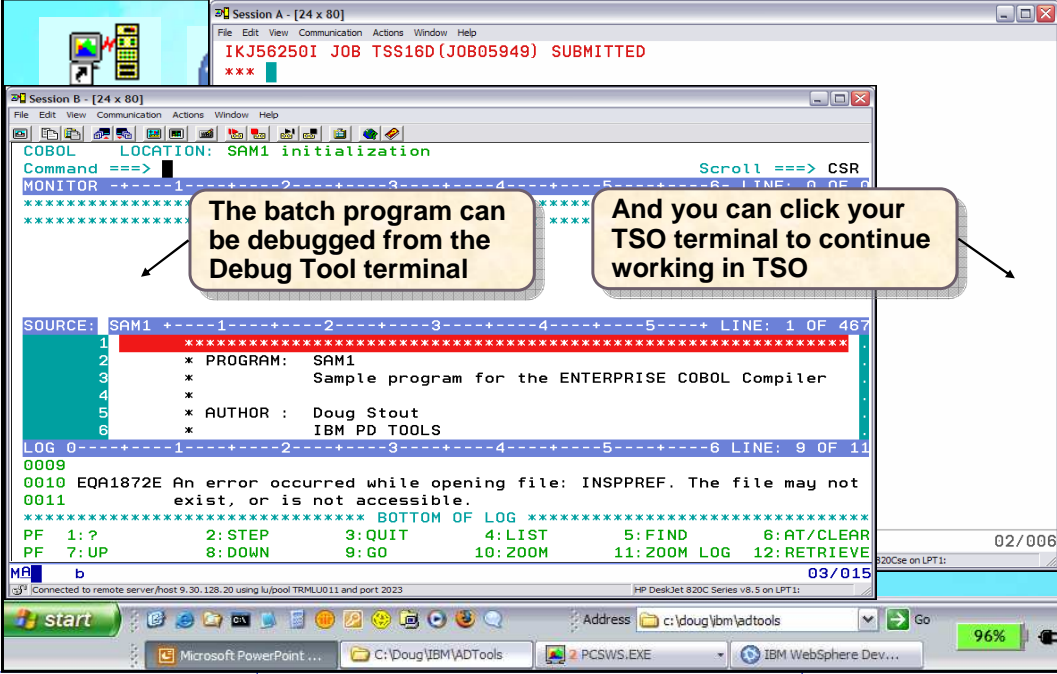
When the step that was modified in the batch job runs, the Debug Tool LE user exit runs automatically. The exit looks for your Debug Tool user exit data set. If the data set does not exist, no action is taken and the step runs normally. If the data set is found, it is read. If the highest level LE program 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 on the Debug Tool terminal.

The Debug Tool terminal window is selected by clicking on it.

The debugger automatically displays on the TIM terminal

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.

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.EQAUOPTS
    - 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 a TIM terminal. 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.EQAUOPTS. 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.

Enabling the Debug Tool LE user exit

■ 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

IBM
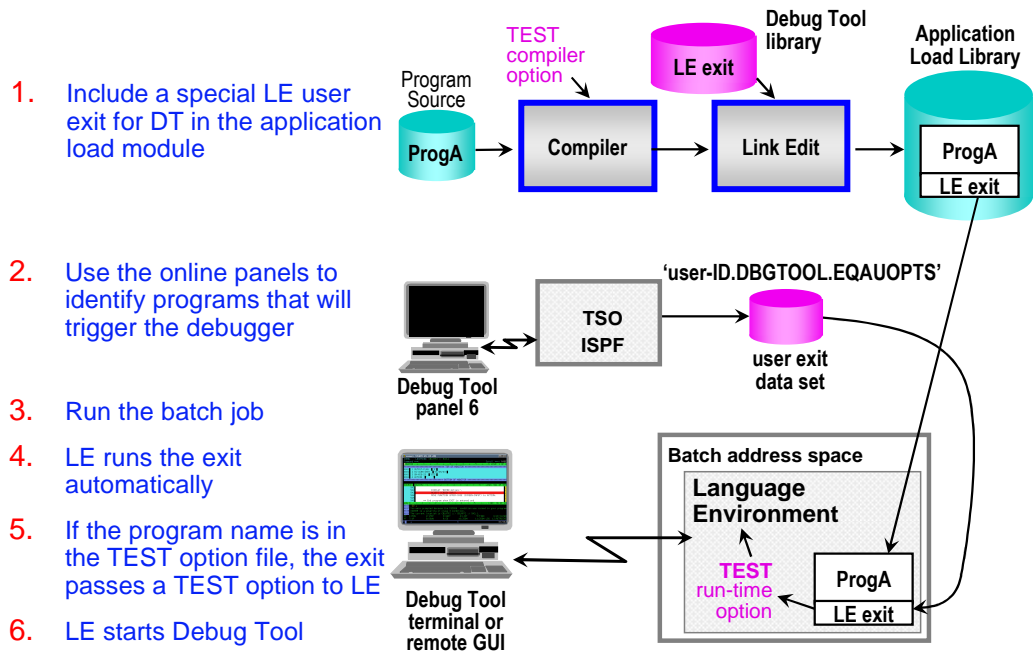
1. Include a special LE user exit for DT in the application load module

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

Program Source — ProgA → Compiler → Link Edit → Application Load Library → ProgA / LE exit

TEST compiler option

Debug Tool library — LE exit

'user-ID.DBGTOOL.EQAUOPTS'

Debug Tool panel 6 — TSO ISPF → user exit data set

Debug Tool terminal or remote GUI

Batch address space
Language Environment
TEST run-time option
ProgA / LE exit

20   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.

DTv12s07DebuggingBatchTimExit.ppt

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

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.
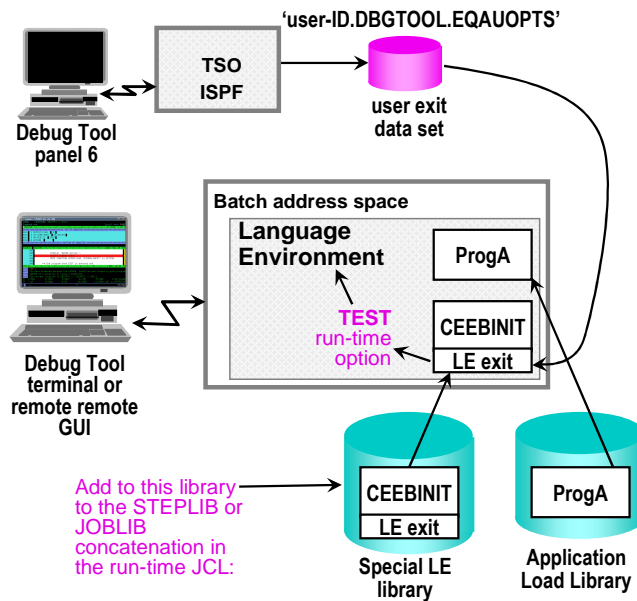
In the third step, the JCL to run the job is edited. The special load library is added to the STEPLIB or JOBLIB concatenation. In the fourth step, the modified JCL is submitted. When the program runs, Language Environment automatically runs the exit program that was made available in STEPLIB or JOBLIB. 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
    ```

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

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 you're 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.

- 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

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

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

| **24** | **IBM Debug Tool for z/OS tutorial** | **© 2012 IBM Corporation** |

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 starting the debugger for a program running in batch, triggering the debugger using the Debug Tool 'user exit data set' utility, and displaying the debugger on a Debug Tool TIM terminal.

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

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

IBM

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, CICS, CICS Explorer, DB2, IMS, z/OS, and zSeries are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

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.

| 26 | IBM Debug Tool for z/OS tutorial | © 2012 IBM Corporation |