Enhanced Application Debugging in CICS TS V2.3 or later

Darren Beard

CICS Developer

Darren_Beard@uk.ibm.com

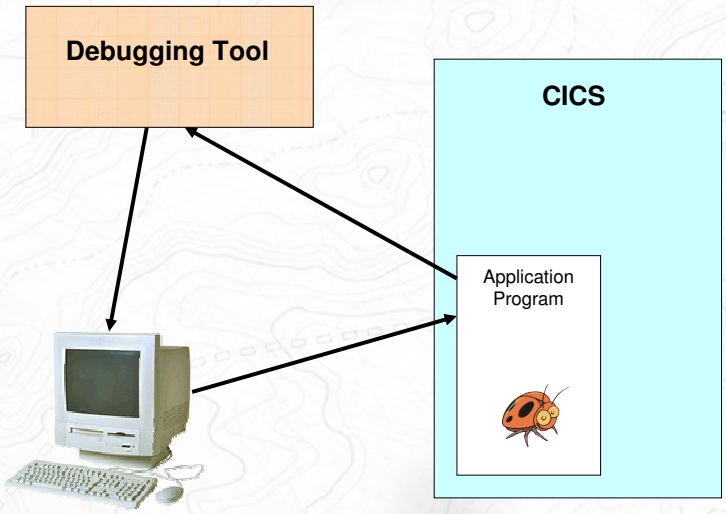
4948

impact·venture*

This presentation deals with the enhanced ability to set up CICS for the debugging of application programs. This support was shipped in CICS TS V2.3.

IMPACT IBM

Debugging



3

IMPACT IBM

The foil provides an overall picture of the main things involved in the debugging scenario. A terminal is in session with CICS and starts executing an application program. The program needs to be debugged, so CICS triggers a debugging tool. The debug tool then starts a session with the original terminal where the programmer can attempt to diagnose the problem.

4

IMPACT IBM

Interactive Debugging

- CEDF
  - CICS and DB2 commands
- Source Debuggers
  - IBM Debug Tool
  - Workstation debuggers
    - Visual age for COBOL
    - Visual age for Java
    - WebSphere Studio family
- Enhanced CICS support for source debuggers
  - Documented interfaces
    - Available to all software vendors and developers
  - Utilized by IBM debuggers

IMPACT IBM

Interactive debugging currently uses CEDF and/or source debuggers. CEDF is limited to CICS and DB2 commands. For source debuggers, there are varying amounts of setup which need to be performed. The new support in CICS TS V2.3 provides enhanced support for source debuggers.

IMPACT IBM

Debugging applications

- Debugging from a workstation
  - Debugger client – Workstation
    - IBM Distributed Debugger
    - WebSphere Studio product family
    - VisualAge family of products
  - Debugger server – CICS
    - IBM Debug Tool
    - Java Virtual Machine
  - Types of applications that can be debugged
    - Compiled (LE) programs, including mixture of languages
    - Java programs
    - Mixed compiled and Java programs
- Debugging from a 3270
  - Restricted to compiled (LE) programs, including a mixture of languages
  - IBM Debug Tool

IMPACT IBM

The foil shows the options available for debugging applications whether from a 3270 terminal or from a workstation. For debugging Java applications, a workstation based debugger is essential, since a 3270 display will not be able to handle Java.

The types of applications to be debugged are becoming more complex. If it is required to put Java applications into production along with existing COBOL applications, then there is an increased likelihood that it will be necessary to debug applications in a mixture of programming languages.

IMPACT IBM

IBM Debug Tool

- Feature of compiler
  - Need only purchase the feature for one compiler
- Available as standalone product
  - This is the recommended way to obtain the product
- Interfaces with LE
  - Uses documented LE debugging interface
  - Supports compiled (LE run-time) languages
    - COBOL, PL/I, C, C++, [hpj]
    - Limited support for non-LE enabled compilers
      - OS/VS COBOL, VS COBOL II
  - Version 5.1 supports High Level Assembler

9

IMPACT IBM

Language Environment supports, but is not required for, an interactive debug tool for debugging applications in a native z/OS environment. The IBM interactive Debug Tool is available with z/OS, or with the latest releases of the C/C++, COBOL, PL/I and VisualAge for Java compiler products.

The mainframe interactive debug tool is offered with the Enterprise COBOL compiler in what is called the Full Function offering. This debug tool is a common facility which supports:-

Enterprise COBOL for z/OS and OS/390
COBOL for OS/390 & VM
COBOL for MVS™ & VM
z/OS C/C++
OS/390 C/C++
C/C++ for MVS
Enterprise PL/I for z/OS and OS/390
PL/I for MVS & VM
High Level Assembler for MVS & VM & VSE

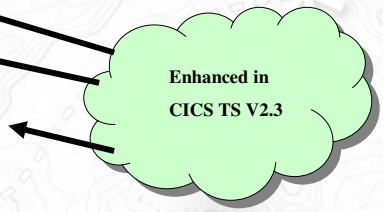
Only one Full Function offering is required for debugging applications when using any of these programming products. An Alternative Function offering is available for customers who wish to receive the Enterprise COBOL for z/OS and OS/390 compiler but not the Debug Tool.

The Debug Tool is also offered as a separate product, IBM Debug Tool for z/OS and OS/390 V3.1. For more information about the functionality offered in the Debug Tool and availability of the product, refer to http://www-306.ibm.com/software/awdtools/debugtool/

10

Debugging in CICS TS V2.3

- Preparing your programs for debugging
- Setting up debugging support
- Creating debugging profiles
- Supplying debugging parameters
- Specifying JVMprofile for debugging
- Invoking the debugger



This foil shows some of the items involved in setting up programs for debugging. The indicated items have been enhanced in CICS TS V2.3.

Setting up the debugging support has been enhanced because there are fewer necessary steps in preparing an application program for debugging. It may also be possible to reduce the amount of configuration in CICS, for example if the Sockets Feature is being used only for Debug Tool, then this will no longer be needed.

In previous releases of CICS, to debug a Java program required a lot of CICS definitions to be set up. These would need to point to a different JVM profile which allows the JVM to be debuggable. Typically, the task would need to run under a different transaction ID. By using the debugging profiles introduced in CICS TS V2.3, it is possible to change the JVM profile during transaction attach. This makes it unnecessary to provide a lot of alternative definitions in CICS and keeps the debugging more closely tied to the person requiring the debug function.

IMPACT IBM

What is required?

- Application program preparation
- CICS setup
- Getting it to work at application runtime

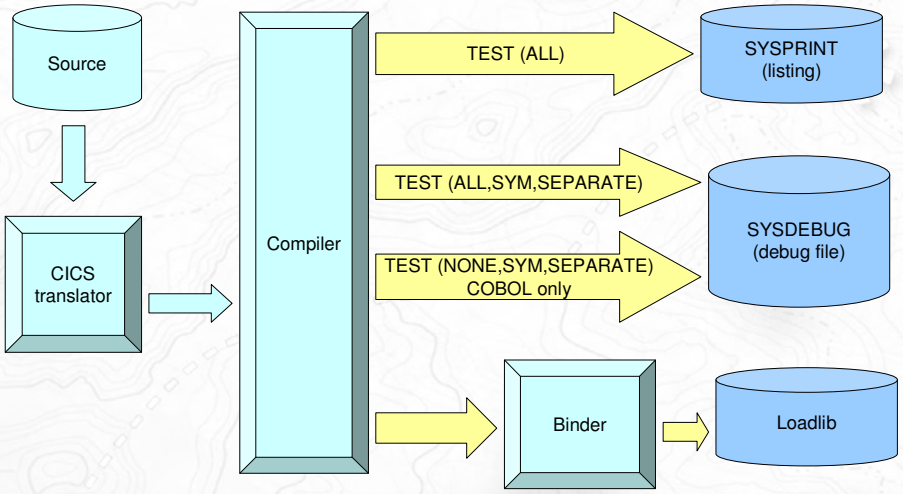
13

IMPACT IBM

In order to debug an application program, various things need to be set up. The application programs need to be set up for debugging and CICS needs to be set up for debugging. Once this has been done, it is then possible for CICS to trigger a debug tool at application runtime. The rest of the presentation explains these steps and explains how CICS 'knows' when to trigger the debug tools.

14

Preparing COBOL and PL/I programs



The foil shows how to get the source of a program into a form which can be displayed by Debug Tool. TEST(ALL) puts hooks into the file so that Debug Tool can set breakpoints easily. Using the SEPARATE option produces a side file in a SYSDEBUG file. This is what is used by Debug Tool to display source code. Unless the side file is produced, source will not display. In order to use the SEPARATE option, the SYM option must be used. This generates symbolic information.

Not using any options means that the program is put into a load library but unless overlay hooks are possible (which needs MVS and Debug Tool setup) then it will not be possible to set breakpoints or display source.

NB. It is not necessary to link-edit CEEBXITA into the CICS program any more. This was required with previous releases of CICS, but is one of the things which has been made simpler in CICS TS V2.3.

SIT Parameters

- Two new SIT parameters have been introduced
  - DEBUGTOOL
    - Can be YES or NO. The default is NO.
  - INFOCENTER
    - Specifies the Universal Resource Locator (URL) of the root of the CICS Information Center directory structure.
      - If set, this enables context sensitive help to generate hypertext links directly into the infocenter.

The foil details the new SIT parameters which are introduced in CICS TS V2.3 which are relevant to this feature.

If DEBUGTOOL is set to NO (the default) then the checking of debugging profiles as a match to a running transaction instance is turned off completely. Debugging profiles may still be created and activated, but if the CICS region to which the profile refers has DEBUGTOOL=NO, then debugging will not be triggered on that system.

Setting DEBUGTOOL to NO does not prevent the use of debuggable JVMs. It only means that the debugging profiles cannot be used to control the triggering of a debug tool or the swapping of JVM profiles at run time.

If DEBUGTOOL is set to YES, then debugging profiles are matched against transaction instances and debugging will be triggered as appropriate.

INFOCENTER is set to the URL of the root of the CICS information directory structure. This allows CICS to construct links to the information center. The web interface to debugging profiles uses this as the main method of providing help information. If this parameter is not specified, then CICS is unable to construct links and the help available is quite limited.

CICS Supplied Definitions (1)

- Definitions for the new supplied transactions
  - CADP and CIDP and the mapset for CADP are supplied in group DFHDP. This is included in list DFHLIST.
- Definitions for ADPM (web interface)
  - supplied in group DFHDPWB. This is also in list DFHLIST.

The foil shows the CICS supplied definitions which define the new transactions and the web interface. They are supplied in groups which are included in list DFHLIST.

CICS Supplied Definitions (2)

- Definitions for the debugging profiles repository files are supplied in three groups
  - DFHDPVR
    - this group defines the files to use RLS
  - DFHDPVSL
    - this group defines the files as local files not using RLS
  - DFHDPVSR
    - this group defines the files as non-RLS and remote
- In DFHDEFDS, the JCL is provided to define the datasets

The foil shows the CICS supplied definitions which define the files needed to store debugging profiles. DHFDEFDS provides sample JCL to create the files so that they can be local or remote and may or may not use RLS.

When creating the files, there is the option to create some sample debugging profiles. If these are created, they can be used as models for 'real' profiles. Anyone can delete them when they are no longer required. It is not necessary to generate them at all. The files can be produced as empty files.

Debugging Profiles

- Debugging Profiles are used to control debugging of
  - compiled language programs
  - Java application programs
  - Enterprise Java beans
  - CORBA stateless objects
- Profiles are stored in a VSAM file
  - may be shared across multiple CICS regions
  - profiles persist across a CICS restart

Debugging profiles are used to control what is to be debugged. Profiles are of two basic types, those for LE languages and those for Java related items. The latter category covers Java programs, Enterprise Java Beans and Corba stateless objects.

The profiles are stored in a VSAM file, which means that they survive a CICS restart. They can also be shared across multiple CICS regions. This is different from Debug Tool profiles, which are lost on CICS restart and which cannot be shared across multiple CICS regions.

Debugging Profiles Usage

- Profiles are either active or inactive
  - no need to delete a profile to end debugging, just make it inactive
- Profiles are useable by multiple users
  - only the profile owner can alter but others may copy and activate or inactivate
- Used to specify debug criteria
  - this allows targeting of the debugging activity

Profiles which might be needed in future but are not needed now do not need to be deleted in order to get them out of the way. They can be changed to 'inactive' state. Any time they are needed, they can be made 'active' again and used.

Debugging profiles can be used by many users in many CICS regions, assuming that the repository file is shared. This may be useful if complex profiles have been set up and they need to be shared.

By keeping the profiles specific, it is possible to target debugging activity closely to particular programs or transactions which are causing problems.

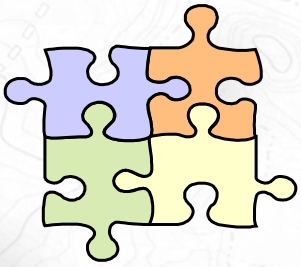
Debugging Profiles - Parameters

- Profile parameters
  - LE programs
    - the transaction under which the program is running
    - the terminal associated with the transaction
    - the netname associated with the transaction
    - the name of the compile unit (COBOL only)
    - the userid of the signed on user.
    - the applid of the CICS region in which the transaction is running
  - Java programs
    - the transaction under which the program is running
    - the class name (for Java programs and CORBA stateless objects)
    - the bean name (for Enterprise Java Beans)
    - the userid of the signed on user
    - the applid of the CICS region in which the transaction is running

The foil summarises the items which may be specified in the two types of debugging profile at creation time. The parameters listed here are relevant to the program to be debugged. There are other parameters (discussed later) which relate to the debugging environment. The parameters listed here are required to match a starting task in CICS if a debug tool is to be triggered. The parameters may be specified completely, or may contain an asterisk "*" at the end so as to make them generic to a greater or lesser extent.

Pattern Matching

Debugging profile	TranID	TermID	Program	UserID	Applid	Bean	Method
Profile1	TRA1	T001	PYRL01	TESTER1	CICSTST1		
Profile2	TRA1	*	PYRL02	TESTER*	*		
Profile3	TRA2	*	*	*	CICSTST2		
Profile4	TRA3			TESTER1	CICSTST1	Account_check	Get_balance



Profile1 is very specific in what needs to be debugged. The UserID, TermID, Program, Applid and TranID are all specified.

Profile2 is more generic. Any user with a userid starting with TESTER on any CICS region on any terminal running TRA1 will trigger a debug tool if this profile is active.

Profile3 is very generic. Anyone running TRA2 in region CICSTST2 will trigger a debug tool if this profile is active.

Profile4 is more specific again.

NB. It is recommended that profiles are kept as specific as possible and that only those which are being used should be kept active. This will prevent accidental and/or unnecessary triggering of a debug tool.

Creating Debugging Profiles

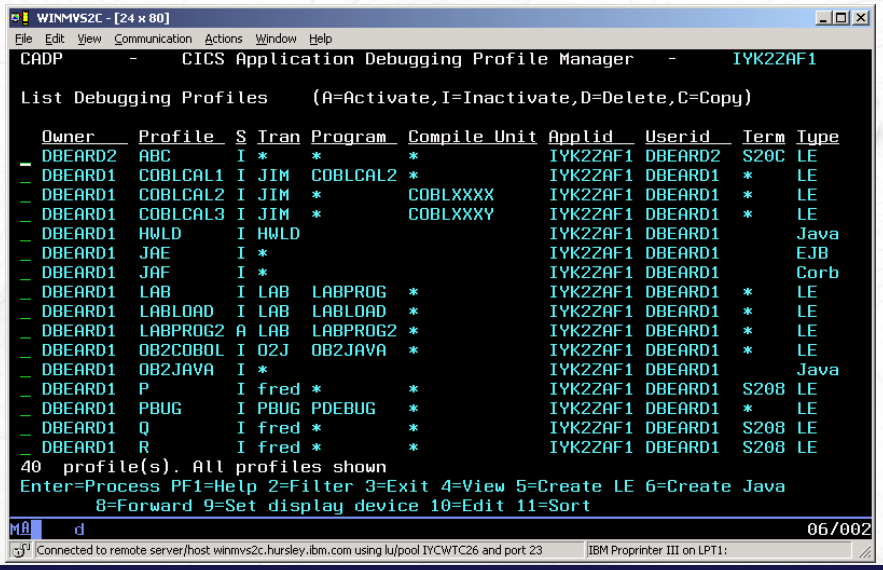
- Debugging Profile Manager
  - CADP – 3270 terminal
    - logical replacement for DTCN (the Debug Tool supplied transaction)
  - ADPM – web interface
    - Uses CICS web support
    - invoke with http://mvs_address:port/CICS/CWBA/dfhdpwb
    - help can link to the CICS Information Center
      - requires the setting of the INFOCENTER SIT parameter

Creation of debugging profiles is either via CADP (a new CICS supplied transaction) or through a web interface.

CADP is a replacement for DTCN, which is a Debug Tool supplied transaction used for defining debugging profiles. There was no web front end to Debug Tool profiles. The Debug Tool documentation will state that CICS TS V3.1 is the last version of CICS for which DTCN is supported. For CICS TS V2.3 and above, an on screen message will also be issued to suggest to users of DTCN that CADP should be used instead.

IMPACT IBM

Initial CADP screen



IMPACT IBM

The foil shows a CICS screen after running transaction CADP. There are already 40 debug profiles defined. The display shows the first 16 profiles and PF8 can then be used to scroll the screen to the next page of profiles. The PF keys currently active are displayed at the bottom of the screen. The information message line explains what is being shown.

Using CADP to create an LE profile

```
CADP      -    CICS Application Debugging Profile Manager   -     IYK2ZAF1

Create LE Debugging Profile ==>           for DBEARD1


CICS Resources To Debug (use * to specify generic values e.g. *, A*, AB*, etc.)
Transaction      ==>                                  Applid   ==> IYK2ZAF1
Program          ==>                                  Userid   ==> DBEARD1
Compile Unit     ==>                                  Termid   ==> TC26
                                                      Netname  ==> IYCWTC26

Debug Tool Language Environment Options
Test Level       ==> All                                       (All,Error,None)
Command File     ==>
Prompt Level     ==> PROMPT
Preference File  ==>

Other Language Environment Options
==>
==>
==>
==>

Enter=Create PF1=Help 2=Save options as defaults 3=Exit 10=Replace 12=Return
```


The foil shows a typical screen displayed after pressing PF5 from the initial CADP screen. This screen is used to define things which must be matched when an LE language program is being run in order for a debug tool to be triggered.

The ring highlights the CICS resources to debug section of the screen. These are the items to be used for “pattern matching” as explained earlier.

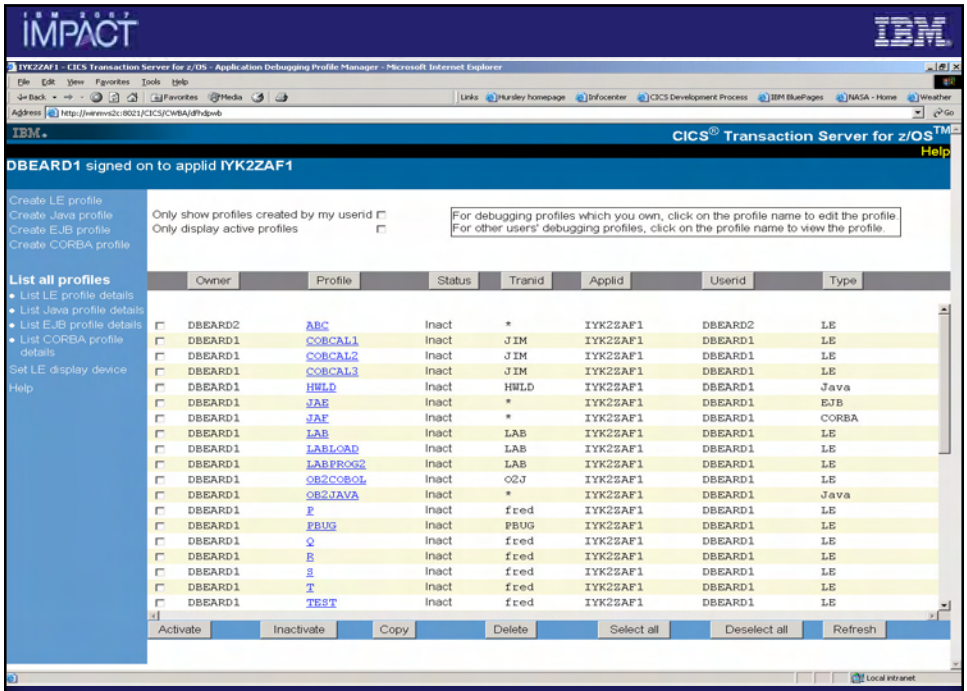
Using CADP to create a Java profile

```
CADP      -    CICS Application Debugging Profile Manager   -     IYK2ZAF1

Create Java Debugging Profile ==>            for DBEARD1

CICS Resources To Debug (use * to specify generic values e.g. *, A*, AB*, etc.)
Transaction      ==>                          Applid       ==> IYK2ZAF1
                                              Userid       ==> DBEARD1

Debugging Options
JVM Profile      ==>

Java Resources To Debug
Type             ==> J       (J=Java Applications, E=Enterprise Beans, C=Corba)

Class (Java Applications or Corba)
==>
==>
==>
==>

Press PF8 to set Bean and Method

Enter=Create PF1=Help 2=Save options as defaults 3=Exit 8=Forward
10=Replace 12=Return
```


The foil shows the screen as a result of pressing PF6 from the main CADP screen. Again, it is possible to specify CICS resources to debug.

Note that it is also possible to set a JVM profile to be used. This allows the program to be run in a debuggable JVM for the debugging session but other users of the program on the CICS region will still run in the normal production JVM. The JVM profile will be changed by CICS at transaction attach time. Prior to CICS TS V2.3, it was necessary to define the transaction to use a debuggable JVM. This then affected all users of the program on the system.

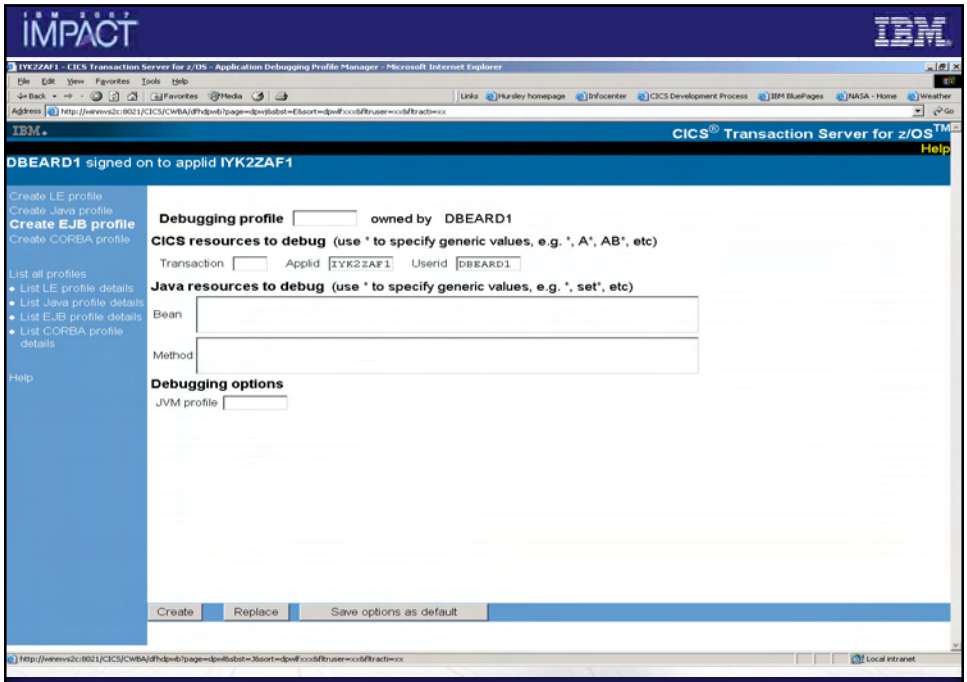
The Java class to be debugged can also be entered on this screen. Pressing PF8, pages to another input screen which allows entry of the bean and method to be debugged.

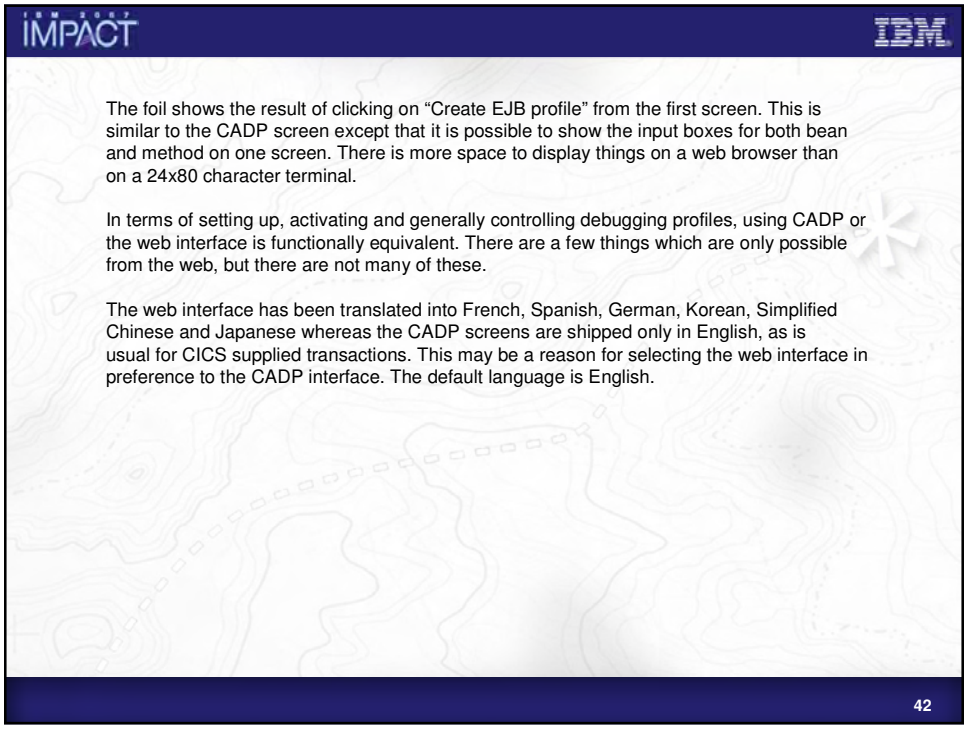


39

This foil shows the same information as the CADP foil, but this time using the web interface. Clearly, the screen layout is very different, but examination of the two foils will show that the information is the same. There are some functions available from the web interface which are not available from the 3270 interface.

40


IMPACT
IBM
IYK2ZAF1 - CICS Transaction Server for z/OS - Application Debugging Profile Manager - Microsoft Internet Explorer
CICS® Transaction Server for z/OS™
Help
DBEARD1 signed on to applid IYK2ZAF1
Create LE profile
Create Java profile
Create EJB profile
Create CORBA profile
List all profiles
List LE profile details
List Java profile details
List EJB profile details
List CORBA profile details
Help
Debugging profile
owned by DBEARD1
CICS resources to debug (use * to specify generic values, e.g. *, A*, AB*, etc)
Transaction
Applid IYK2ZAF1
Userid DBEARD1
Java resources to debug (use * to specify generic values, e.g. *, set*, etc)
Bean
Method
Debugging options
JVM profile
Create
Replace
Save options as default
41


IMPACT
IBM
The foil shows the result of clicking on “Create EJB profile” from the first screen. This is similar to the CADP screen except that it is possible to show the input boxes for both bean and method on one screen. There is more space to display things on a web browser than on a 24x80 character terminal.
In terms of setting up, activating and generally controlling debugging profiles, using CADP or the web interface is functionally equivalent. There are a few things which are only possible from the web, but there are not many of these.
The web interface has been translated into French, Spanish, German, Korean, Simplified Chinese and Japanese whereas the CADP screens are shipped only in English, as is usual for CICS supplied transactions. This may be a reason for selecting the web interface in preference to the CADP interface. The default language is English.
42

Activating Debugging Profiles

- When profiles are created, they are INACTIVE
- To be used, a profile must be activated
  - CADP can be used to activate profiles
  - the web interface can be used to activate profiles

When a debugging profile is first created, it will always be in INACTIVE state. This is so that debug activity is not started accidentally before it is really intended. This is especially important if some quite generic profiles are being set up.

Before debugging activity can take place, the profiles created have to be activated. Either CADP or the web interface can be used for this.

Activating using CADP

```
CADP      -    CICS Application Debugging Profile Manager   -     IYK2ZAF1

List Debugging Profiles    (A=Activate,I=Inactivate,D=Delete,C=Copy)

  Owner    Profile  S Tran Program  Compile Unit Applid   Userid   Term Type
a DBEARD2  ABC      I *    *        *            IYK2ZAF1 DBEARD2  S20C LE
_ DBEARD1  COBLCAL1 I JIM  COBLCAL2 *            IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  COBLCAL2 I JIM  *        COBLXXXX     IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  COBLCAL3 I JIM  *        COBLXXXY     IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  HWLD     I HWLD                       IYK2ZAF1 DBEARD1       Java
_ DBEARD1  JAE      I *                          IYK2ZAF1 DBEARD1       EJB
_ DBEARD1  JAF      I *                          IYK2ZAF1 DBEARD1       Corb
_ DBEARD1  LAB      I LAB  LABPROG  *            IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  LABLOAD  I LAB  LABLOAD  *            IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  LABPROG2 A LAB  LABPROG2 *            IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  OB2COBOL I O2J  OB2JAVA  *            IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  OB2JAVA  I *                          IYK2ZAF1 DBEARD1       Java
_ DBEARD1  P        I fred *        *            IYK2ZAF1 DBEARD1  S208 LE
_ DBEARD1  PBUG     I PBUG PDEBUG   *            IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  Q        I fred *        *            IYK2ZAF1 DBEARD1  S208 LE
_ DBEARD1  R        I fred *        *            IYK2ZAF1 DBEARD1  S208 LE
40  profile(s). All profiles shown
Enter=Process PF1=Help 2=Filter 3=Exit 4=View 5=Create LE 6=Create Java
      8=Forward 9=Set display device 10=Edit 11=Sort
```


The foil shows how to activate a debugging profile. The letter ‘a’ (capital letters would also work) is entered next to the profile which requires activation. Multiple profiles can be activated in one operation by entering an ‘a’ next to several of the profiles. When the ENTER key is pressed, the requested actions are processed.

NB. With CADP it is possible to enter combinations of a, d, i and c next to several profiles. When the ENTER key is pressed, all of the requested operations will be actioned.

Selecting the output device

```
CADP      -    CICS Application Debugging Profile Manager   -     IYK2ZAF1

Set LE Debugging Display Device

Debugging Display Device
Session Type             ==>  TCP                        (3270,TCP)
3270 Display Terminal    ==>  TC26

TCP/IP Name Or Address
==> 9.20.221.189
==>
==>
==>
Port                     ==>  08001

Type of socket communication ==> Single                  (Single,Multiple)


Display this panel on LE profile activation ==> YES



Enter=Save and return PF1=Help 3=Exit 12=Cancel
```


If display of the “Set LE debugging display device” panel is set to “yes” then the screen shown in the foil is displayed. The default is to display this screen, but it can be suppressed if not required. It can be accessed directly by pressing PF9 from the initial CADP screen.

There are several important items on this screen. Firstly, this screen selects whether the debugging output is to be directed to a 3270 screen, which may or may not be the screen from which CADP is being run, or whether the output is to be sent to a TCP/IP address. The screen defaults the 3270 display terminal to be the terminal from which CADP is running. If TCP is selected, then the TCP/IP address and port number need to be specified. The default port number used is 8001, since this is the default listening port for the WebSphere Studio and Rational families of products.

Another very important field for TCP/IP output is the “Type of socket communication” field. This accepts single or multiple.

NB. The options set on this screen belong to the transaction user and not the debugging profile. The association between user preferences and debugging profiles is made at profile activation time.

Single or Multiple sockets?

- Single Socket Configuration
  - this is the default
  - it is the preferred value when the debugging client is a WebSphere Studio product
- Multiple Socket Configuration
  - this must be used if an old VisualAge product is used as the debugging client
  - this *can* be used for WebSphere Studio products, but is not preferred
  - some firewalls prevent this mode of operation

The choice between single or multiple sockets for communication is mainly dictated by which client debug tool is being used and the presence of any firewalls between CICS and the workstation.

Likely problems if the wrong setting is chosen:-

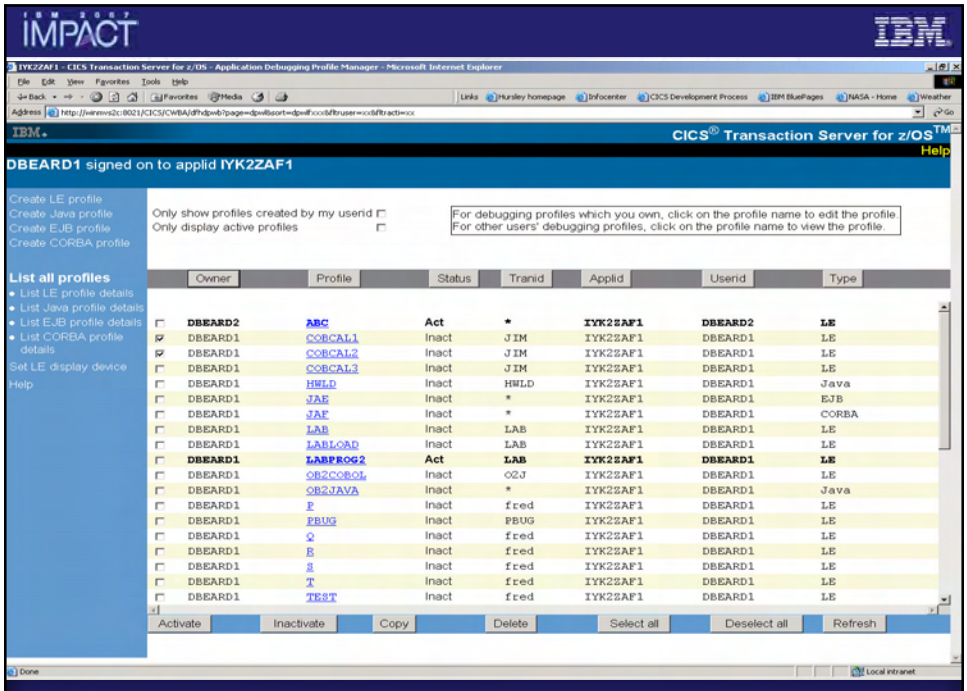
Symptom: transaction to debug is run but abends 4038
Cause: Single sockets has been used to a workstation debug product which required multiple sockets.

Symptom: transaction to debug is run and output starts at a 3270 instead of the workstation expected
Cause: The workstation product is not listening or the debugging profile points to an invalid TCP/IP address. IBM Debug Tool detects this and sends the output to the 3270 instead.

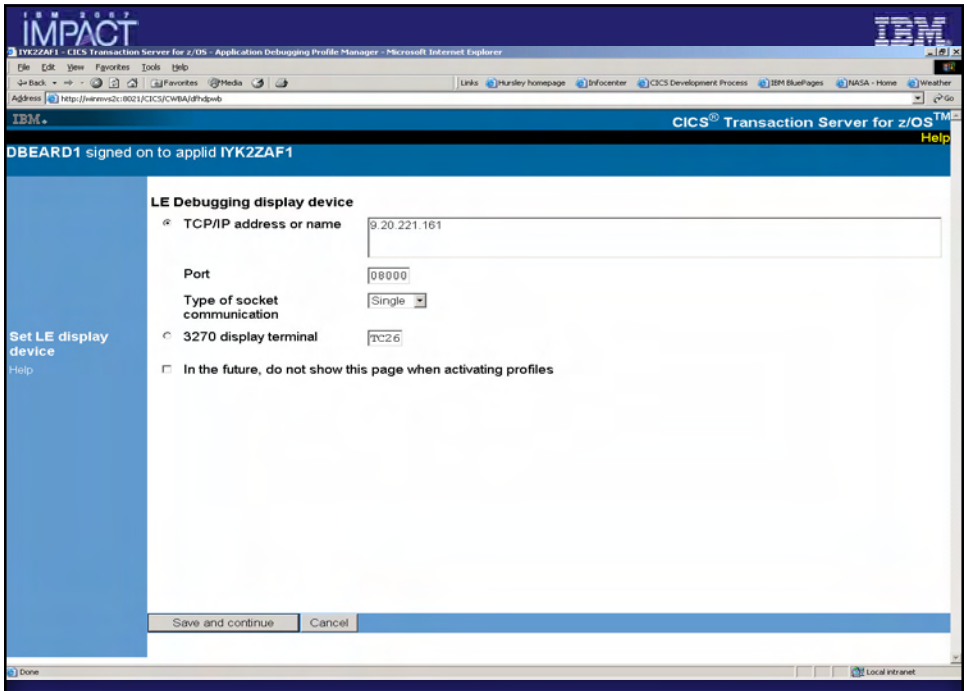
Activation Completed

```
CADP      -    CICS Application Debugging Profile Manager   -     IYK2ZAF1

List Debugging Profiles    (A=Activate,I=Inactivate,D=Delete,C=Copy)

  Owner    Profile  S Tran Program  Compile Unit Applid   Userid   Term Type
_ DBEARD2  ABC      A *    *        *            IYK2ZAF1 DBEARD2  S20C LE
_ DBEARD1  COBLCAL1 I JIM  COBLCAL2 *            IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  COBLCAL2 I JIM  *        COBLXXXX     IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  COBLCAL3 I JIM  *        COBLXXXY     IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  HWLD     I HWLD                       IYK2ZAF1 DBEARD1       Java
_ DBEARD1  JAE      I *                          IYK2ZAF1 DBEARD1       EJB
_ DBEARD1  JAF      I *                          IYK2ZAF1 DBEARD1       Corb
_ DBEARD1  LAB      I LAB  LABPROG  *            IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  LABLOAD  I LAB  LABLOAD  *            IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  LABPROG2 A LAB  LABPROG2 *            IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  OB2COBOL I O2J  OB2JAVA  *            IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  OB2JAVA  I *                          IYK2ZAF1 DBEARD1       Java
_ DBEARD1  P        I fred *        *            IYK2ZAF1 DBEARD1  S208 LE
_ DBEARD1  PBUG     I PBUG PDEBUG   *            IYK2ZAF1 DBEARD1  *    LE
_ DBEARD1  Q        I fred *        *            IYK2ZAF1 DBEARD1  S208 LE
_ DBEARD1  R        I fred *        *            IYK2ZAF1 DBEARD1  S208 LE
1  activate(s) processed
Enter=Process PF1=Help 2=Filter 3=Exit 4=View 5=Create LE 6=Create Java
      8=Forward 9=Set display device 10=Edit 11=Sort
```


The foil shows the screen after the activation has completed. The information line displays a message stating that 1 activation has completed successfully and the status of the selected debugging profile has changed from I (inactive) to A (active).



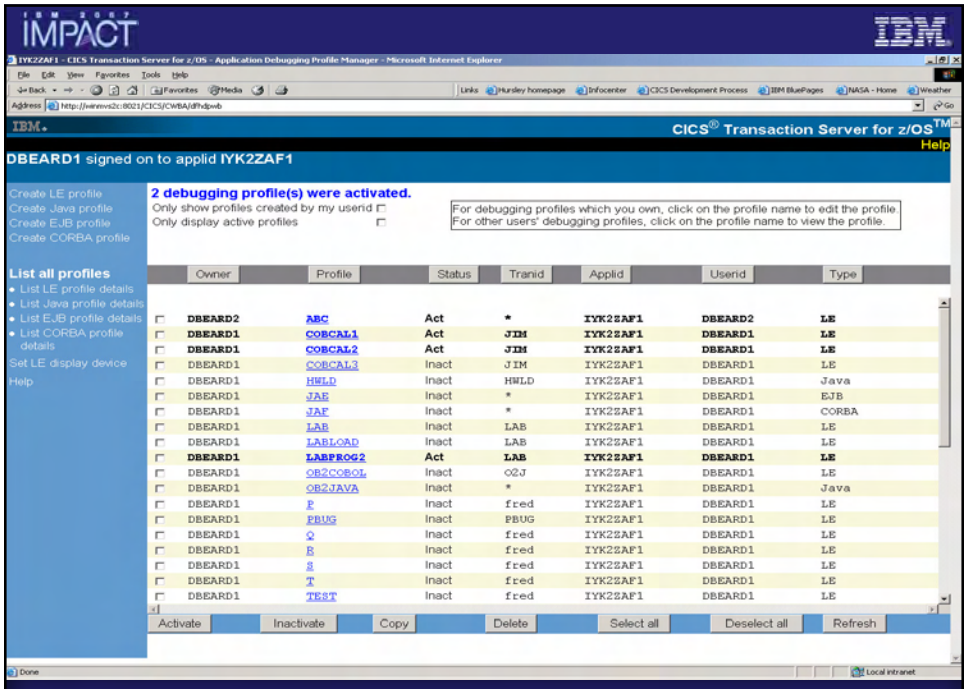
Two profiles to be activated have been selected by clicking in the boxes next to them. The “Activate” button then needs to be clicked so as actually to implement the activation requests.



55

This screen is basically the same as the corresponding CADP screen. The same selections need to be made before the “Save and continue” button is clicked.

56

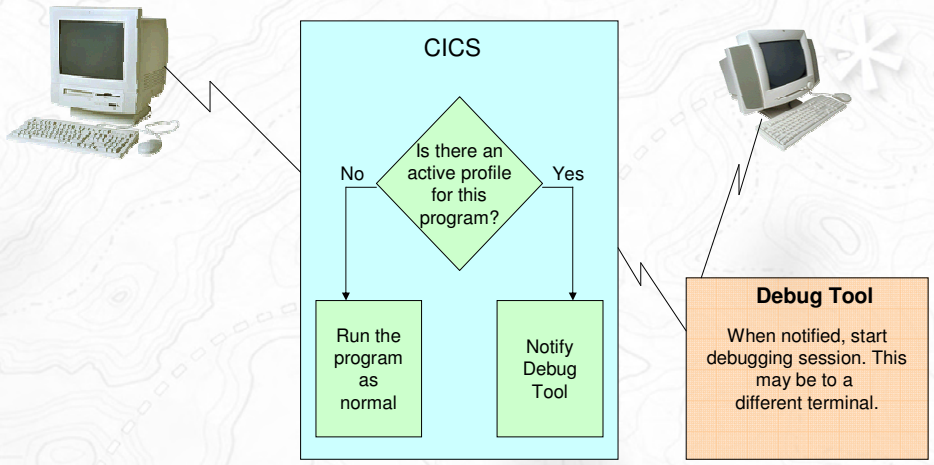


57

The selected two profiles are now active. This is indicated by the status field showing “Act” and by the fact that the profiles now appear in bold font on the display.

58

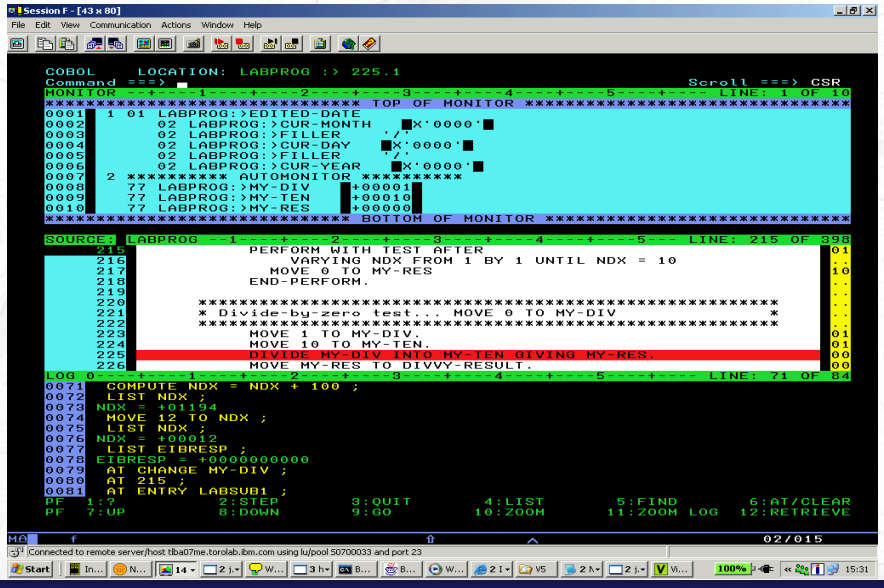
Using Debugging Profiles



When a transaction is being attached in CICS, assuming that debugging is activated in CICS, then checks are made to see whether there exists an active debugging profile which matches the transaction instance being attached. If not, then everything runs as normal. If there is, then a debugging tool can be triggered and the output of the debug session may be routed either to the terminal which started the transaction or to another terminal. The foil shows the debug session being routed to a second terminal.

IMPACT IBM

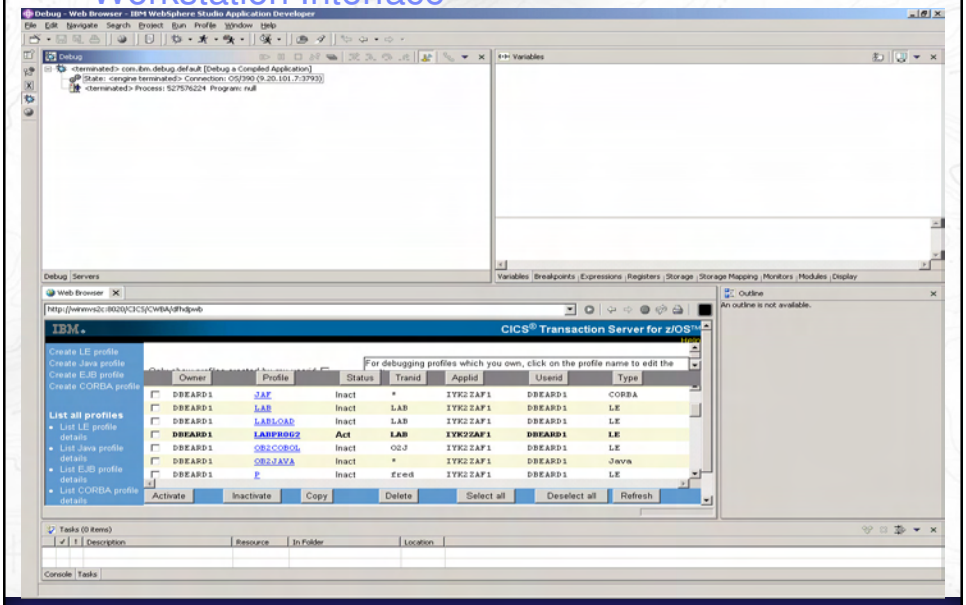
3270 Debug Session



IMPACT IBM

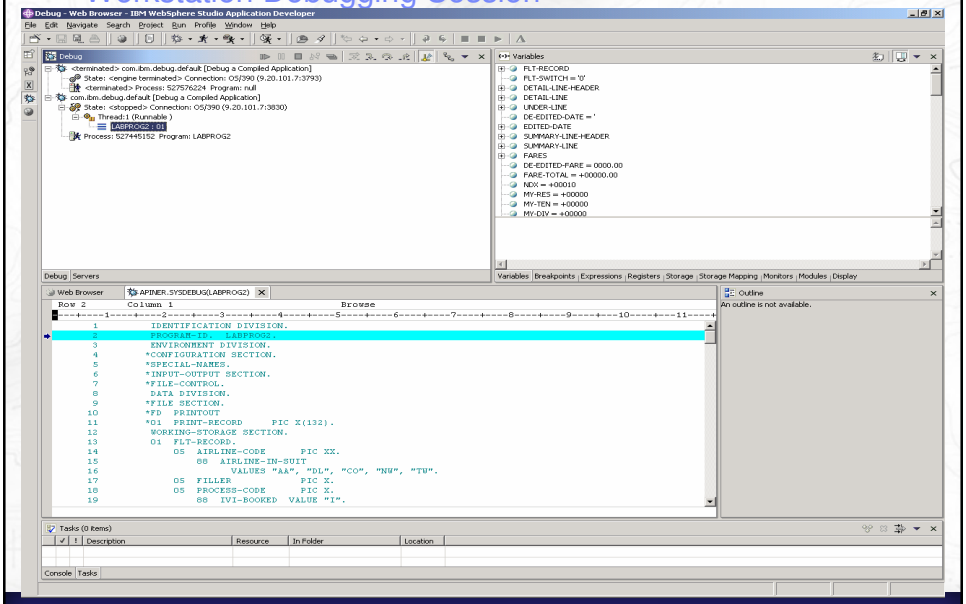
The foil shows a 3270 screen display as a result of running a transaction for which there was a matching debugging profile. The debug tool in use is the IBM Debug Tool. Once the tool has triggered, the actual debugging is not affected. The enhancements in CICS TS V2.3 are only concerned with triggering a debug tool when it is required. It does not affect the debugging activity once the tool is running.

Workstation Interface



The foil shows a typical workstation controlled debugging session. This is a screen capture from using WebSphere Studio Application Developer 5.1. A web browser pane has been opened in the debug perspective. This allows access to the web interface to the CICS debugging profiles from within WSAD. The LABPROG2 profile is shown in BOLD and is ACTIVE.

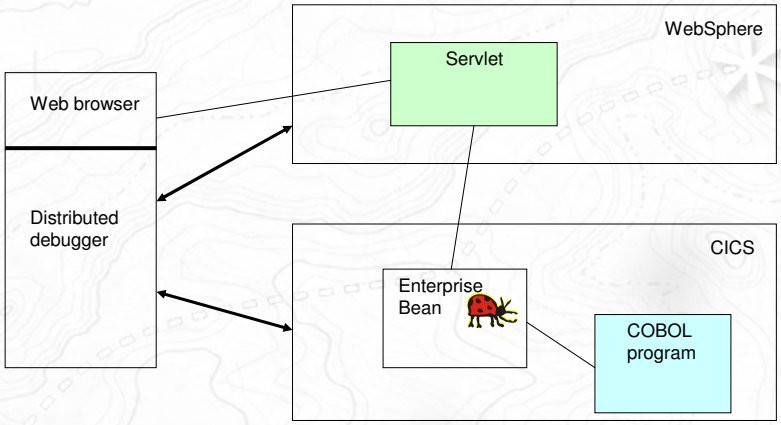
Workstation Debugging Session



A source pane has opened automatically when the program is run on CICS. The output has been directed to the WSAD running on the workstation instead of to the 3270 from where the transaction was started. The variables pane now shows the variables in the program. From here, the usual debugging activity can be performed. The code can be single stepped and variables and storage examined.

The ability to control the debugging profiles and perform the debugging activity from within one tool, such as WSAD or WSED, is clearly very convenient from an application programmer's point of view.

"End to End" Debugging



The foil attempts to show an "end to end" debugging scenario. A servlet accessed from a web browser appears to have a problem. However, the servlet actually executes an EJB in CICS which in turn links to a COBOL program. It is possible to have a debugging session simultaneously with WebSphere and CICS and to control the debugging profiles in CICS from the workstation running the web browser which invokes the servlet. This allows a more ready access to the code in the servlet, EJB and the COBOL application which should help the 'bug' to be located.

CEMT interface

```
I SYS
  STATUS:  RESULTS - OVERTYPE TO MODIFY
    Aging( 00500 )                  Progautoexit( DFHPGADX )
    Akp( 04000 )                    Progautoinst( Autoactive )
    Cicstslevel(020300)             Reentprotect(Noreentprot)
    Cmdprotect(Cmdprot)             Release(0630)
    Db2conn()                       Runaway( 0020000 )
    Debugtool( Debug )              Scandelay( 0050 )
    Dfltuser(DBEARD1)               Sdtran(CESD)
    Dsalimit( 04194304 )            Sosstatus(Notsos)
    Dsrtprogram( NONE     )         Storeprotect(Inactive)
    Dtrprogram( DFHDYP    )         Time( 0000100 )
    Dumping( Sysdump )              Tranisolate(Inactive)
    Edsalimit( 0209715200 )
    Forceqr( Noforce )
    Logdefer( 00005 )
    Maxtasks( 020 )
    Mrobatch( 001 )
    Oslevel(010400)
    Progautoctlg( Ctlgmodify )

                                                SYSID=CICS APPLID=IYK2ZAF1
   RESPONSE: NORMAL                          TIME:  16.55.24  DATE: 15.05.07
PF 1 HELP       3 END       5 VAR       7 SBH 8 SFH 9 MSG 10 SB 11 SF
```


The setting of DEBUGTOOL can be changed dynamically in a running CICS region by using CEMT. The setting has been added to the CEMT INQUIRE SYSTEM command as indicated in the foil. CEMT displays the status as Debugtool (Debug) or (Nodebug).

CIDP – Inactivating Profiles

- Debugging profiles
  - survive a CICS restart (of any type: Initial, Cold or Warm)
  - maintain their status, 'active' or 'inactive'
  - have scope of sysplex
- How can all profiles be inactivated?
  - Using CADP or ADPM
  - Transaction CIDP
    - Can be used as part of PLT processing
- CIDP
  - inactivates *ALL* profiles in the repository file. This affects all regions which share the repository file
- Setting DEBUGTOOL=NO
  - has the same effect as inactivating profiles related to a *SINGLE* region (although the profiles are still active)

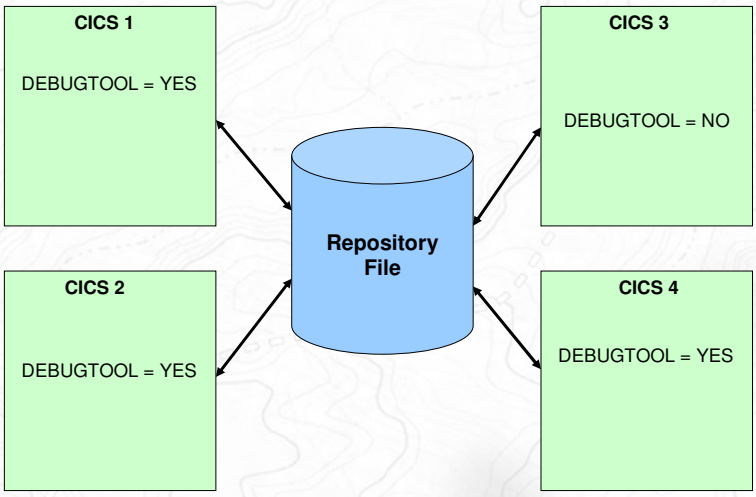
Unlike DTCN profiles, the new CICS debugging profiles were designed to be more permanent. They survive restarts. This includes maintaining their status across restarts. Active profiles will remain active.

As a result of customer feedback during early testing of CICS TS V2.3, it was thought that this behaviour of profiles would sometimes be inappropriate. Although setting DEBUGTOOL=NO could be used to inactivate a single CICS region, it was felt that a global mechanism would be useful.

The response to this requirement was to produce the CIDP transaction. This transaction can be entered at a terminal and will inactivate all profiles in the repository file. The transaction can also be invoked in a PLT (2nd phase startup or shutdown). The program which implements the transaction (DFHDPIN) can also be linked to as a CICS program.

The transaction results in message DFHDP0300 which notifies the user as to how many profiles have been inactivated.

Scope of Effect



The four CICS regions are all sharing the same debugging profiles repository file. CICS 3 has DEBUGTOOL=NO specified, so users on that system may define profiles but they will not result in any debug tools being triggered for transactions run on that system. As far as CICS 3 is concerned, the profiles are inactive. However, should a profile be defined on CICS 3 which happens to match something running in CICS 1, then a debug tool would be triggered.

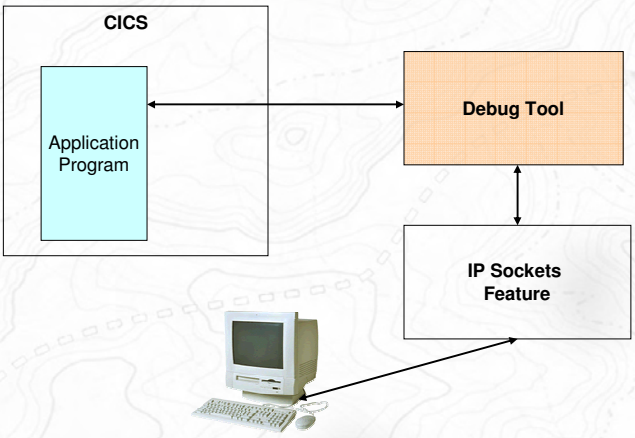
If CICS 1 is restarted and executes CIDP as part of it's restart processing, then **all** active profiles in the repository file are made inactive. This will affect any debugging activity which might still be occurring on CICS 2 and CICS 4.

Sockets changes

- To send a debugging session to a workstation based debug tool with DTCN
  - configure the DTCN profile to send the output to a TCP/IP address and port
  - install and configure the IP sockets feature
    - set up a configuration file and associated definitions
    - run the transaction EZAC to set up the file for the CICS region
    - run the transaction EZAO to start the feature for the CICS region
- Since CICS has a sockets domain, why not use it?

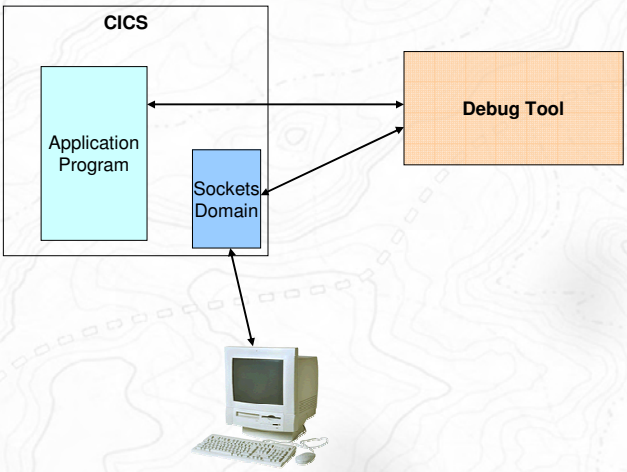
The use of the IP sockets feature is needed for Debug Tool communication to a workstation. However, it seems better to use the CICS sockets support for this communication since this simplifies the configuration of the system. Also, if the only requirement for the IP sockets feature is purely for Debug Tool use, then it will no longer be needed in the system at all.

IP Sockets use



An application program running in CICS results in Debug Tool being triggered as a result of a successful match against a debugging profile. The debugging profile specifies that the debugging session is to go to a TCP/IP address. Prior to CICS TS V2.3, this meant that Debug Tool would have to use the IP sockets feature to perform the actual communications with the workstation.

Sockets use in CICS TS V2.3



Since the application program is running in CICS and Debug Tool is already calling CICS in order to check the debugging profiles, it is logical for Debug Tool to use CICS sockets domain for communication with the workstation. This removes the requirement for the IP sockets feature in this scenario.

Software Requirements

- Debug Tool for z/OS and OS/390 V3.1, with PTF UQ77541 for APAR PQ73643, Debug Tool for z/OS V4.1 or V5.1
- WebSphere Studio Application Developer 5.0.1 or higher (or equivalent vendor product)
- WebSphere Studio Enterprise Developer 5.0.1 or higher (or equivalent vendor product)

The foil lists the software levels required for the IBM products solution for “end to end” debugging. With WSAD and WSED prior to 5.0.1, there are problems with task termination. The communications hang and never free up.

Note : CICS TS V3.1 has a pre-req of Debug Tool V5.1. There is no plan to support CICS TS V3.1 on older levels of Debug Tool.

Debug Tool 5.1 supports Assembler (even non-LE assembler)

CICS TS V3.2 requires Debug Tool for z/OS V7.1 (5655-R44)

Useful Tips

- CADP 'cares' about 4 termids
  - the termid set in the debugging profile on which a transaction must run if it is to enter a debug session
  - the termid to which the debugging session is to be directed, if not being directed to a TCP/IP address and port
  - the current LE output device setting
  - the termid of the current CADP user
- Three userids are important
  - the userid of the person who created a debugging profile
  - the userid of the current CADP or ADPM user
  - the userid under which a running transaction must execute if it is to enter a debug session

CADP has to be aware of several TERMIDs and USERIDs. The foil lists these. It is useful to try and be aware of these when setting up debugging profiles. CADP will default the TERMID field in most of the input screens to be the TERMID of the current session. This might not be what is required.

More Useful Tips

- Profiles remain ACTIVE even across a CICS restart
  - be sure to inactivate any profiles which are not required
- Make profiles as specific as possible
  - profiles which are too generic will cause debugging sessions to be started when they are not required
- Set DEBUGTOOL to NO (or use CEMT to do this) when debugging is not required
  - there is a performance overhead if DEBUGTOOL is set to YES. For debugging, this is not important. In a production region, it is.

Active debugging profiles remain active even following a CICS restart. This can be a surprise since DTCN defined profiles were lost completely by performing a restart.

Generic profiles are useful for catching problems which it is not possible to narrow down easily. However, having lots of generic profiles in a system is likely to result in debug tools being triggered when this is not required.

Having DEBUGTOOL set to YES results in CICS performing checks at transaction attach time to see whether there is a matching debugging profile for the current transaction instance. To avoid this performance overhead, it is best to set DEBUGTOOL to NO unless debugging is really being performed. In a test region, the overhead is probably not important, but in a production region it could be.

Summary

- CICS provided support for debugging
  - Support for LE and Java debuggers
    - regardless of vendor
  - Simplified setup
    - may remove requirement for IP sockets feature
  - Improved management of debugging profiles
    - stored in repository, active or inactive
    - web interface in addition to 3270
  - Dynamic switching of JVM profiles
  - Greater integration for debugging of mixed-language programs and applications
- Result : Enhanced capability to debug CICS applications

The foil summarises the main benefits and enhancements made to the ability to debug application programs in CICS TS V2.3.

More information

- CICS TS V2.3 Infocenter
  - Look up the CADP transaction
  - Look up 'debug'
- IBM Redbook
  - Introduction to the IBM Application Development Tools for Z/OS and OS/390 (SG24-6887-00)
- CICS Update
  - October 2004 issue pp 3 – 9

The foil summarises some of the places where you can find out more information should you desire to do so.

IBM 2007 IMPACT

IBM

Questions
and
Answers

impact·venture*

IBM 2007 IMPACT

IBM

The end.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. 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. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. 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 the applicable license agreement governing the use of IBM software.

References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM trademarks, see www.ibm.com/legal/copytrade.shtml
AIX, CICS, CICSPlex, DB2, DB2 Universal Database, i5/OS, IBM, the IBM logo, IMS, iSeries, Lotus, OMEGAMON, OS/390, Parallel Sysplex, pureXML, Rational, RCAF, Redbooks, Sametime, System i, System i5, System z , Tivoli, WebSphere, and z/OS.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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