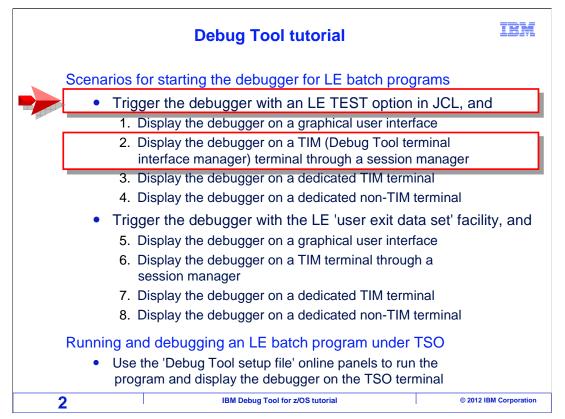
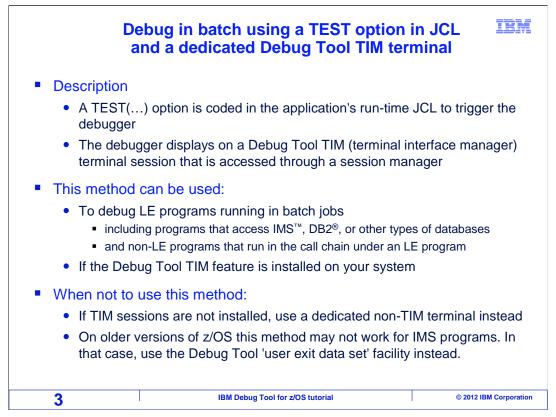


This is the tutorial for IBM Debug Tool for z/OS[®], one of the IBM zSeries[®] problem determination tools.



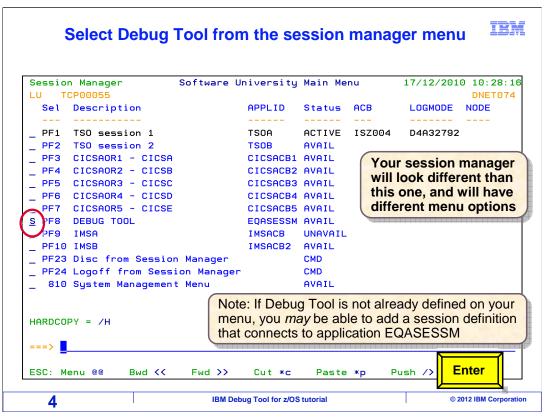
In this section, you will see a scenario for starting Debug Tool for a batch application. In this scenario the application runs as a batch job, and a TEST option is coded in the JCL that will trigger the debugger when the job runs. The debugger displays on a Debug Tool terminal interface manager terminal session that you access through a session manager. You can skip this section if you do not plan to use Debug Tool this way on your system.



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

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

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



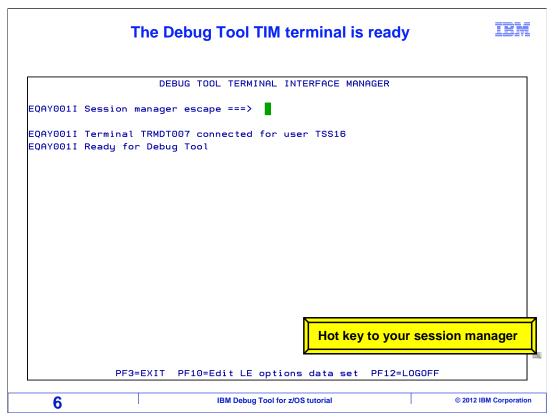
In this scenario, terminal session manager software is used to switch between TSO and a Debug Tool session. A terminal session manager may be installed on your system to give users an easy way to access different applications or systems from a single terminal. There are different terminal session manager products available. This example uses one called IBM session manager. There may be a different one on your system, so your session manager will probably look different than the one shown here, and the menu options will be different. Even so, most session managers provide a menu of systems that can be selected, or a special "hotkey" you can use to switch between sessions.

In this example the user is already logged on to TSO. If your session manager shows the debugger as one of the options, select it. If your menu does not have an option for the debugger, depending on your authorizations, you may be able to define a new session for it. If you define your own session, try connecting to an APPL with a name of EQASESSM, which is the default network name for the Debug Tool application. If your session manager does not give you a way to connect to Debug Tool, consider contacting your help desk or the Debug Tool installer so that the session manager can be configured correctly.

Log on to	the Debug Tool TIM session	n IBM
EQAY001I Session manager esca EQAY001I Enter your userid:		
EQAY001I Password:		
	Log on to Debug Tool with User ID and password	n your
	PF3=EXIT	Enter
5	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 the TIM feature is not installed on your system, take one of the tutorial sections that describes how to debug using a non-TIM terminal, instead of this section.

Next, log on to 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.



At this point your Debug Tool terminal is ready to receive a debugging session. Next the TSO terminal session will be selected. Typically, there is a special "hot key" you can press to jump to the session manager menu, such as a specific PA key or a key combination. Press whatever hot key or key combination you use to re-display your session manager menu.

		Software Un	niversity	Main Me	าน	17/12/2010	
LU TI Sel	CP00055 Description		APPLID	Status		LOGMODE	DNET074 NODE
S PF1	TSO session 1		TSOA	ACTIVE		D4A32792	
PF2	TSO session 2		TSOB	AVAIL			
_ PF3	CICSAOR1 - CICSA		CICSACB1	AVAIL			
_ PF4	CICSAOR2 - CICSB		CICSACB2	AVAIL			
_ PF5	CICSAOR3 - CICSC		CICSACB3	AVAIL			
_ PF6	CICSAOR4 - CICSD		CICSACB4	AVAIL			
_ PF7	CICSAOR5 - CICSE		CICSACB5	AVAIL			
_ PF8	DEBUG TOOL		EQASESSM	ACTIVE			
_ PF9	IMSA		IMSACB	UNAVAIL			
_ PF10	IMSB		IMSACB2	AVAIL			
_ PF23	Disc from Session	Manager		CMD			
_	Logoff from Sessi	_		CMD			
_ 810	System Management	: Menu		AVAIL			
HARDCO	PY = /H						
===>							

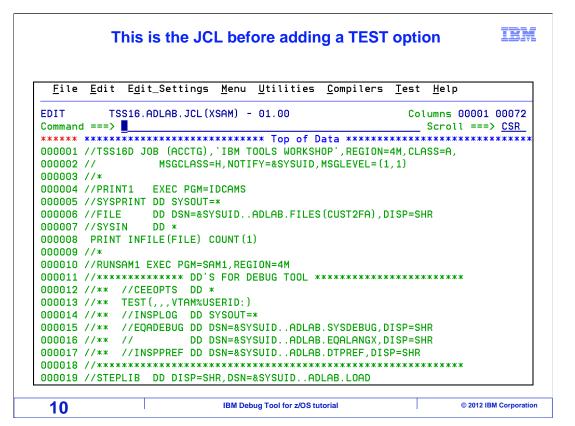
Select the TSO session.

	<u>M</u> enu <u>U</u> tilities	s <u>C</u> ompilers <u>O</u> ptions <u>S</u> tatus <u>H</u> elp	
0pt	tion ===> 2	OS/390 Primary Option Menu	
0	Settings	Terminal and user parameters	User ID . : TSS16
1	View	Display source data or listings	Time : 09:53
2	Edit	Create or change source data	Terminal. : 3278A
3	Utilities	Perform utility functions	Screen : 1
4	Foreground	Interactive language processing	Language. : ENGLISH
5	Batch	Submit job for language processing	Appl ID . : ISR
6	Command	Enter TSO or Workstation commands	TSO logon : TPROCO2
7	Dialog Test	Perform dialog testing	TSO prefix: TSS16
8	LM Facility	Library administrator functions	System ID : STLABF6
9	IBM Products	IBM program development products	MVS acct. : *
10	SCLM	SW Configuration Library Manager	Release . : ISPF 6.1
11	Workplace	ISPF Object/Action Workplace	
12	0S/390 System	OS/390 system programmer applications	
13	0S/390 User	OS/390 user applications	
DB	DataBase	Interactive Database Functions	
G	SWS/STL	Display SWS/STL Selection Panel	Enter

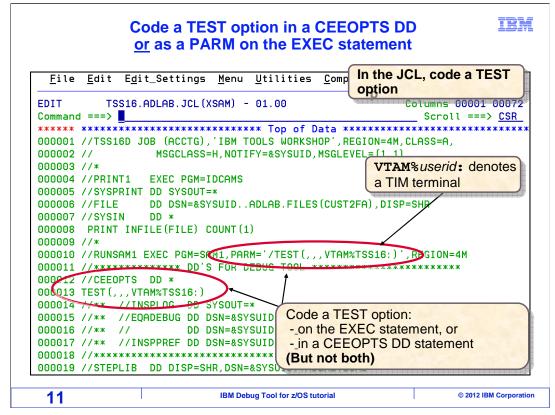
In TSO, open the JCL that runs the application you want to debug. In this example, the ISPF editor is selected.

		Open th	e JCL tha	t will ru	un the	batch p	orogram	IBM
<u>M</u> enu	<u>R</u> efList	R <u>e</u> fMode	<u>U</u> tilities	<u>W</u> orksta	ation <u>H</u> e	elp		
Command	===>			Entry Pa	nel			
Grou Type Memb Other P Name	ect	ADLAB JCL XSAM	tial or VSF	IM Data S	Set, or :		er selecti X file:	_ on list) +
Worksta	tion File Name .	:	Open the applicatio	JCL that n in batc	will run ch	the		
Profile Format Data Se	Name t Passwor	:::=		ME	Confirm (lixed Mod dit on l	de Jorkstat	ove/Replac ion rd lengt	e Enter
9			IBM Debug	Tool for z/OS t	utorial		© 201	2 IBM Corporation

Open the JCL in the editor.

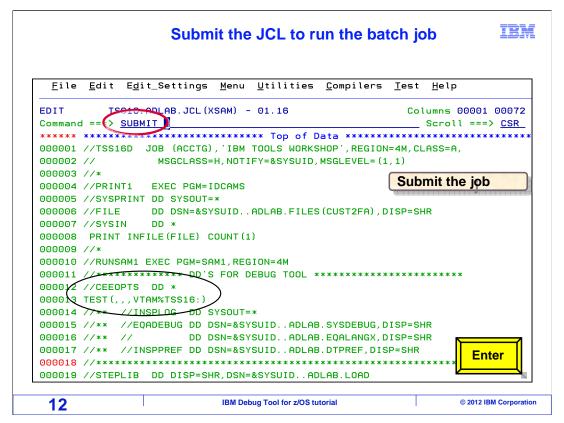


This is JCL that is used to run the application program in this example. It is shown here before any changes have been made.

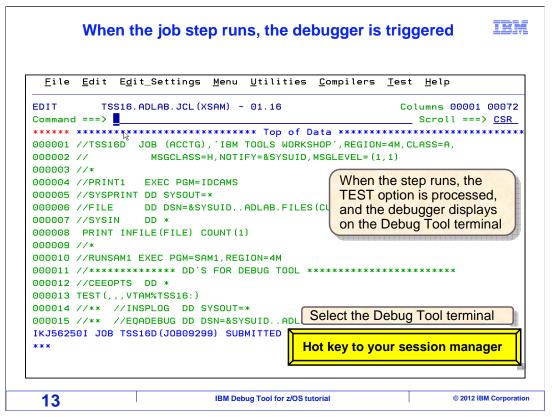


To set a trigger for the debugger, code a Language Environment TEST option in the JCL. There are two ways to do it. Depending on how the JCL is coded, you may be able to code a TEST option directly on the EXEC statement that runs the program. Or you can add a special DD statement called CEEOPTS, and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

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



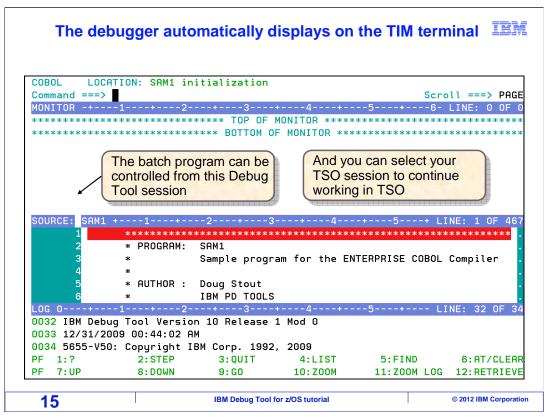
The job is ready to run. The JCL is submitted to batch.



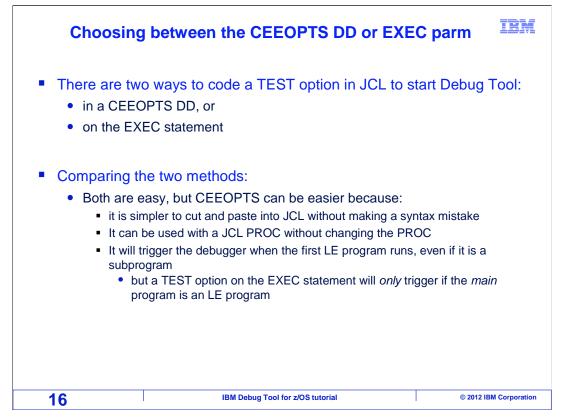
When a step runs that has a TEST option specified, Language Environment receives the TEST option and starts Debug Tool. The debugger is displayed in the TIM terminal session. Press whatever hot key or key combination is needed to display the session manager.

LU TO			niversity	Main Me	าน	17/12/2010	
Sel	CP00055 Description		APPLID	Status	ACB	LOGMODE	DNET074
PF1	TSO session 1		TSOA	ACTIVE	ISZ004	D4A32792	
PF2	TSO session 2		TSOB	AVAIL			
_ PF3	CICSAOR1 - CICSA		CICSACB1	AVAIL			
_ PF4	CICSAOR2 - CICSB		CICSACB2	AVAIL			
_ PF5	CICSAOR3 - CICSC		CICSACB3	AVAIL			
_ PF6	CICSAOR4 - CICSD		CICSACB4	AVAIL			
PF7	CICSAOR5 - CICSE		CICSACB5	AVAIL			
<u>s</u> PF8	DEBUG TOOL		EQASESSM	ACTIVE			
PF9	IMSA		IMSACB	UNAVAIL			
_ PF10	IMSB		IMSACB2	AVAIL			
_ PF23	Disc from Session	n Manager		CMD			
_ PF24	Logoff from Sess	ion Manager		CMD			
_ 810	System Managemen	t Menu		AVAIL			

Select the Debug Tool TIM terminal.



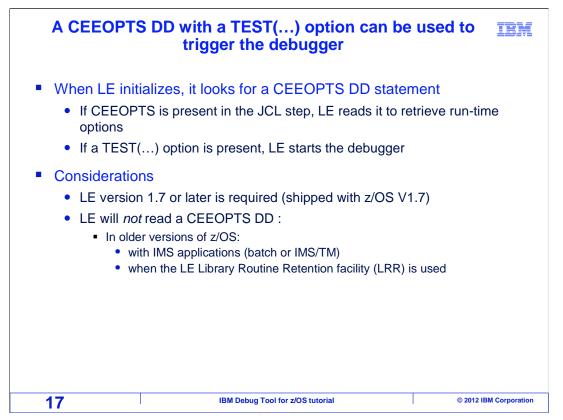
The batch job is running on the host z/OS system, and the debugger is communicating with the Debug Tool terminal. You can control the program from the Debug Tool terminal, and you can also select your TSO session to continue working in TSO at the same time. Be aware that the TSO session is no longer 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.



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

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

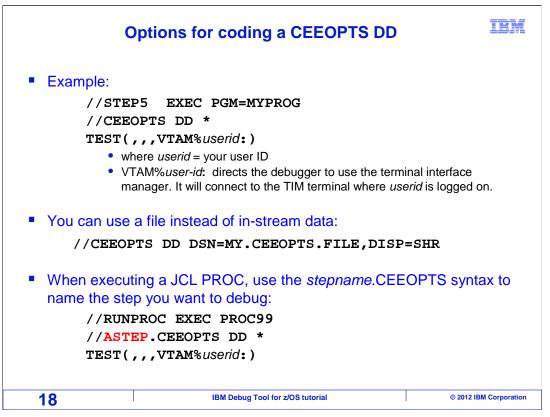
Finally, when you use CEEOPTS, Debug Tool will trigger when the first LE program is called, even if it is a subprogram. That is an advantage over the other method, because when you code a TEST option on the EXEC statement, Debug Tool will only trigger if the main program is an LE program.



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

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

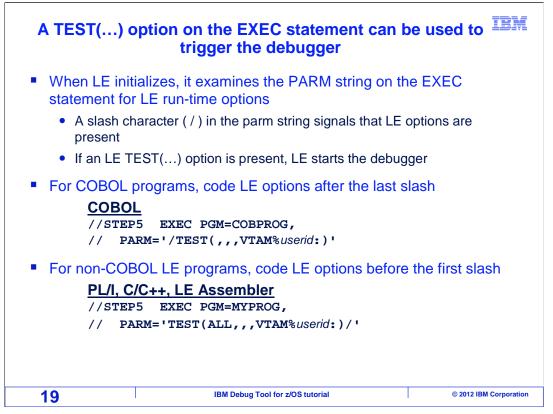
LE will not read a CEEOPTS DD statement with IMS applications on older versions of z/OS. So, if you want to debug an IMS batch application and you are running on an older system, you cannot use a TEST option in JCL to do it. In that case, use the 'Debug Tool user exit data set facility' to trigger the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



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

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

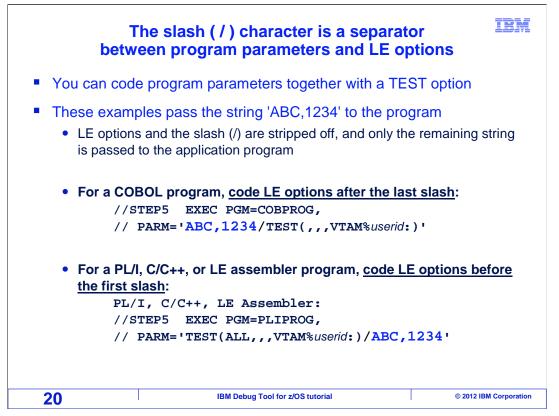
The third example shows how you can code a CEEOPTS DD statement when the JCL executes a PROC. In this case, notice that the EXEC statement runs a PROC named PROC99. The "ASTEP.CEEOPTS" DD statement will add the CEEOPTS DD to a proc step named ASTEP. When it is done this way, you do not have to make any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



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

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

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

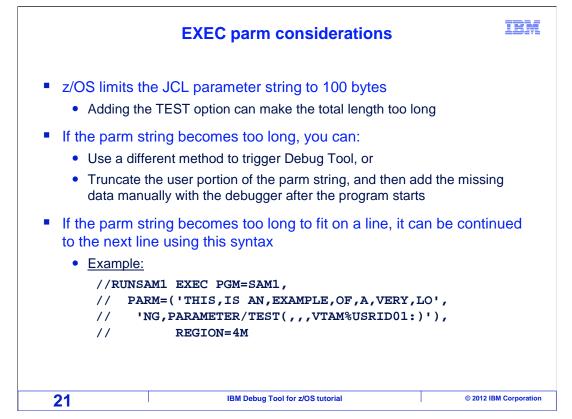


When you code a TEST option on the EXEC statement, you can still code parameters for your program. Here are examples that pass the data string "ABC,1234" to the application program as a parameter.

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

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

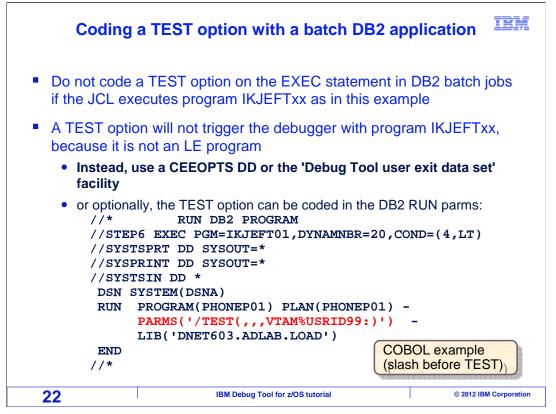
In either case, LE strips its options and the slash out of the parm string before control is passed to the application program. So the LE options are not received by your program.



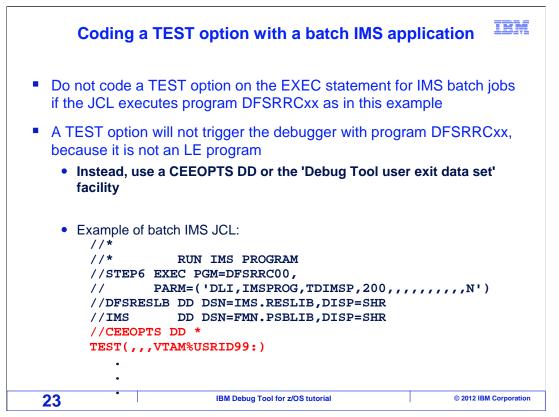
z/OS limits JCL parameters to a total of 100 bytes. It is possible to run into a situation where adding a TEST option will make the total length exceed 100 bytes.

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

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



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



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

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

	Feedback	IBM
Your feedback i	s valuable	
	rove the quality of IBM Education Assistant contends by providing feedback.	ent to better
Did you find to	this module useful?	
 Did it help yo 	u 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_DTv12s06bDebuggingBate	<u>chSMTimTest.ppt</u>
	This module is also available in PDF format at: /DTv12s06bDebuggingBatchSMTimTest.pdf	
24	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 disclair	mers
IBM, the IBM logo, ibm.com, DB2, IMS, VTAM, z/OS, and zSeries are trademarks or registered trademarks of Internat registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other cor IBM trademarks is available on the web at " <u>Copyright and trademark information</u> " at http://www.ibm.com/legal/copytrademarks is available on the web at " <u>Copyright and trademark information</u> " at http://www.ibm.com/legal/copytrademarks is available on the web at " <u>Copyright and trademark information</u> " at http://www.ibm.com/legal/copytrademarks is available on the web at " <u>Copyright and trademark information</u> " at http://www.ibm.com/legal/copytrademarks is available on the web at " <u>Copyright and trademark information</u> " at http://www.ibm.com/legal/copytrademarks is available on the web at " <u>Copyright and trademark information</u> " at http://www.ibm.com/legal/copytrademarks is available on the web at " <u>Copyright and trademark information</u> " at http://www.ibm.com/legal/copytrademarks is available on the web at " <u>Copyright and trademark information</u> " at http://www.ibm.com/legal/copytrademarks is available on the web at " <u>Copyright and trademarks</u> " (<u>Copyright and trademarks</u>) at the trademarks of the trademarks at the trademark information at the trademarks at the tradem	mpanies. A current list of other
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 O MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRES 'AS IS' WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BA PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHAL ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREA' REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONE OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.	ENTATION, IT IS PROVIDED SED ON IBM'S CURRENT L NOT BE RESPONSIBLE FOR (OTHER DOCUMENTATION. TING ANY WARRANTIES OR
© Copyright International Business Machines Corporation 2012. All rights reserved.	
25 IBM Debug Tool for z/OS tutorial	© 2012 IBM Corporation