



IBM Software Group

The Integrated Debugger in IBM WebSphere Development Studio Client for iSeries

iSeries Application Development Team
IBM Toronto

WebSphere. software



August 2003 | WDS V5R2 and WDSC 5.1

© 2003 IBM Corporation

The Integrated Debugger is part of the IBM WebSphere Development Studio Client for iSeries (WDSC is the short form). This presentation gives an overview of the Integrated Debugger and its features.

Table of contents

Overview

- Debugger Startup from the Remote System Explorer
- Debug Perspective
- Debugger Functions
- Launch Configurations and Settings
- Demo

This presentation first gives a high level overview of WDSC and where the Integrated Debugger fits in, as well as a look at the different ways to start the debugger and at its features.

IBM Software Group | WebSphere software IBM

WebSphere Development Studio (Advanced) V5.1

NEW!

Unlimited Licenses

RPG **Cobol**

C/C++ **PDM, SEU
SDA, RLU**

V5R1 or V5R2
5722-WDS

www.ibm.com/software/awdtools/wds400

iSeries	iSeries	iSeries*	iSeries	Web Facing*	iSeries Projects
Java	Debug	Struts Web	Web Service	RSE	
Trace	Profiling	DB	XML	App Server	EJB* J2EE*
					Test Cases*

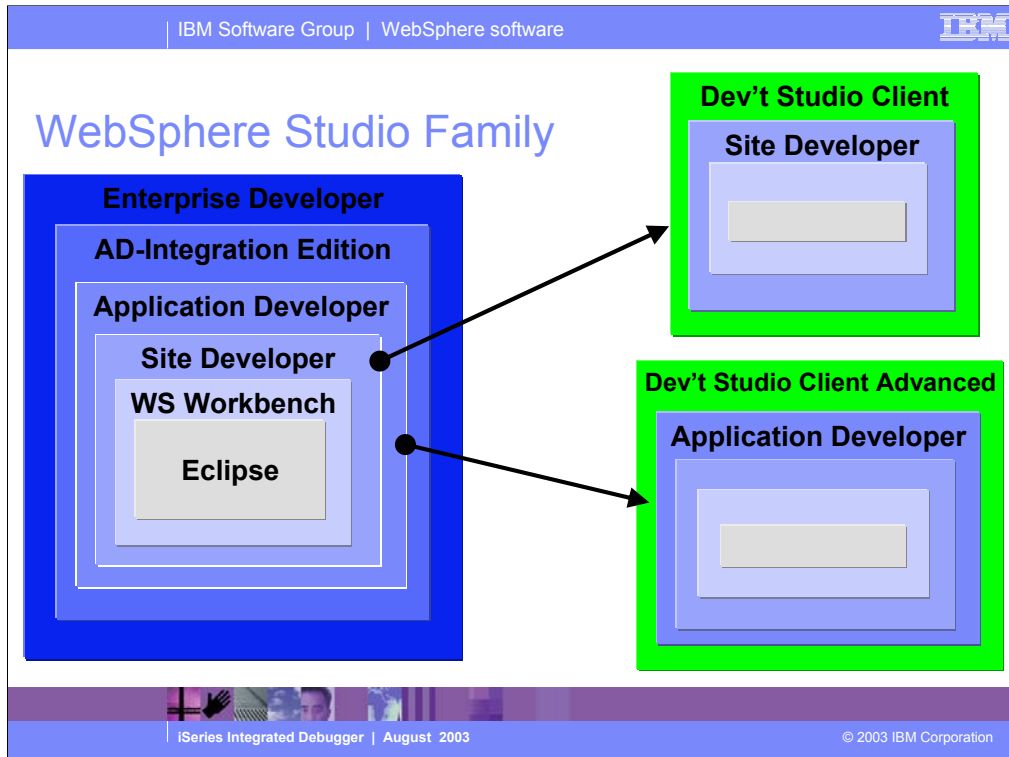
+CODE
+VisualAge RPG

*WebSphere Development Studio Client Advanced V5.1

iSeries Integrated Debugger | August 2003 © 2003 IBM Corporation

The Advanced edition of Development Studio Client V5.1, and Development Studio V5.1, will be available October 17, 2003.

The difference of Development Studio Client Advanced over Development Studio Client is that it has additional tools (blue boxes with asterisks) and some enhancements to existing iSeries tools (green boxes with asterisks). The majority of the additional tools are related to Enterprise Java Bean (EJB) development. You'll see later that Development Studio Client Advanced is based on WebSphere Application Developer (Application Developer) versus WebSphere Studio Site Developer (Site Developer).



Here you see that Development Studio Client is based on WebSphere Studio Site Developer, while Development Studio Client Advanced is based on WebSphere Studio Application Developer.

The Workbench is based on the open-source Eclipse technology. It is not for sale, but is the basis of all IBM WebSphere Studio products, and is available to business partners.

Site Developer is IBM's entry level offering based on eclipse, and it is for building dynamic Web sites out of non-EJB Java. Application Developer extends Site Developer and adds support for EJBs. Application Developer-Integration Edition extends Application Developer and adds support for JCA Connectors and for Workflow. Enterprise Developer extends Application Developer-Integration Edition and adds support for S/390 and Enterprise Generation Language (EGL), the follow-on to VisualAge Generator.

Debuggers in WSDC

- **Integrated iSeries Debugger**
- Java iSeries Debugger
- WebSphere Application Server debug adapter
- IBM Distributed Debugger
 - Object Level Trace
- Compiled language debugger
- Java Script debug adapter
- Active Script debugger

WSDC ships with 7 different debuggers, each one for a different user scenario.

Integrated iSeries Debugger – for all your host applications in RPG, Cobol, CL, C and C++

Java iSeries Debugger – for your Java development

WAS debug adapter – for EJBs, JSPs and servlets running on WAS

IBM Distributed Debugger and OLT – for Was 3.5, Java JNI calls and CODE users

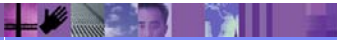
Compiled language debugger – for workstation development

Java Script debug adapter – server side Java Script with WAS debug adapter

Active Script debugger

Integrated iSeries Debugger - Overview

- RPG, Cobol, CL, C, and C++
- ILE and non ILE
- Source and Listing view
- Batch and interactive
- Multi-Threaded Applications
- Client/Server Applications
- Distributed Applications



Debugger Invocation - Overview

- Remote Systems view in RSE and Project perspective
 - Pop up menu of a program or service program with source in library system or IFS
 - Pop up menu of a job
- iSeries Table view
 - Pop up menu of a program or service program
- Workbench
 - Tool bar: Debug pull down menu
 - Menu bar: Run menu

In a typical scenario, the user makes some changes to the source, runs a verify and compile and then debugs the program.

All these tasks are integrated into the RSE and can be invoked from menu items, tool buttons or pop up menus.

Table of contents

Overview

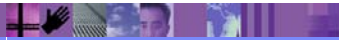
 **Debugger Startup from the Remote Systems view**

Debug Perspective

Debugger Functions

Launch Configurations and Settings

Demo



Debugger Invocation from Remote Systems view

Debug As... from Popup menu of:

Program
Service program
iSeries job

Debug As...

Batch
Interactive
Multi-threaded
Job

You can run and debug programs from the Remote Systems view or the iSeries Table view in three ways:

- In a batch job
- In an interactive job
- In a server job

In the third case, running the program will use the same job as the Remote System Explorer communications server job. With batch and interactive jobs, you cannot monitor the status as easily, however, you do not tie up your communications server and you are notified when the program command ends. Batch jobs work as you would expect and do not require any initial setup.

Note: A multi-threaded debug session creates a new server job and this way keeps the RSE communications server job free for other tasks.

Where does the Application run?

Debug As...

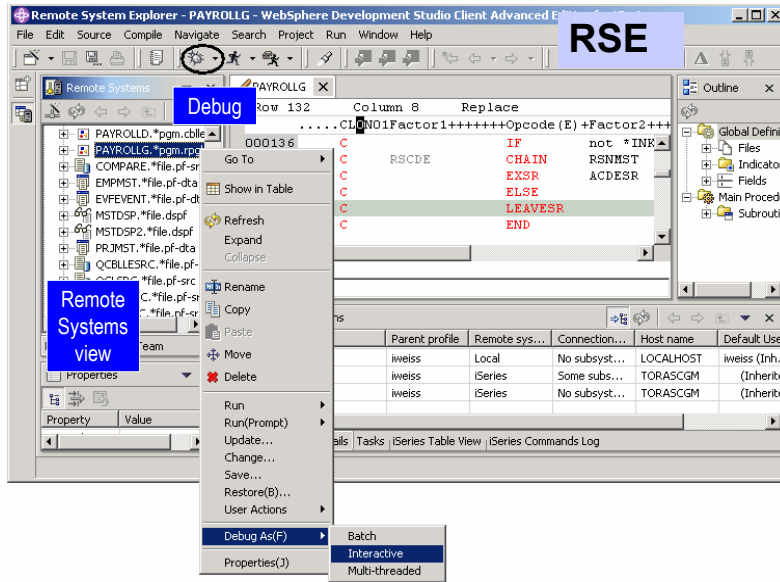
Batch	Submitted to batch
Interactive	5250 emulation, STRRSESVR
Multi-threaded	creates BCI job
Job	Specified job

Depending on the debugging mode you selected, the application will run in different jobs. Debugging in batch or interactive uses the same type of job as running the application.

Debugging Interactive requires a 5250 emulation session where the STRRSESVR command has been run.

For multi-threaded applications, instead of using the RSE server job, the debugger creates a BCI job and calls your application in that job.

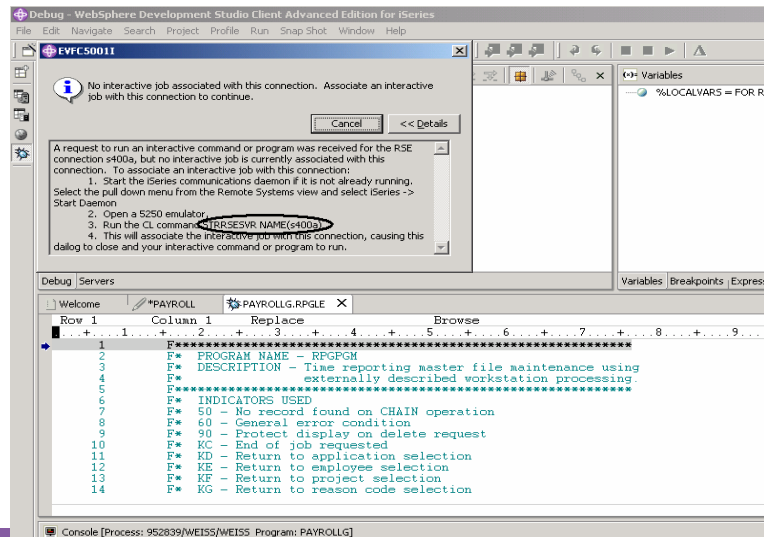
Selecting debug as job allows the user to attach to a job on the iSeries. This can be a batch, interactive, multi-threaded or even Java job.



When the debugger is invoked from the pop up menu, it will start for the selected program, step into it and terminate the debug session when the program ends.

For service programs, a dialog is displayed to collect information about the starting program.

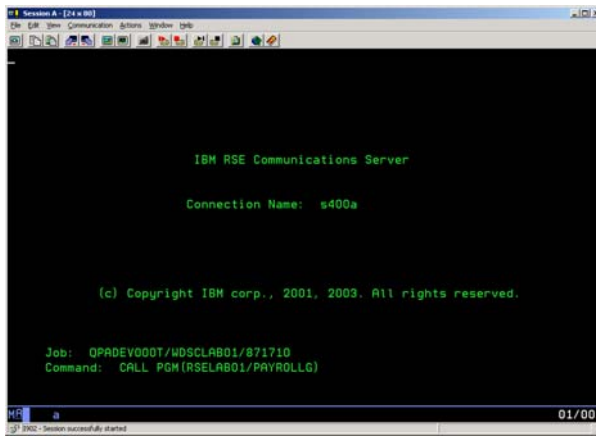
Debugging – Interactive



If you selected Debug as Interactive but there is currently no interactive RSE session, a message is displayed informing you that there is no such session and also giving you instructions how to remedy the situation.

Tip: You can copy the command from the message (STRRSESVR NAME(yourServerName)) and paste it into the 5250 command line.

Interactive RSE Session



```
IBM RSE Communications Server

Connection Name: s400a

(c) Copyright IBM corp., 2001, 2003. All rights reserved.

Job:  OPADVE000T/WDCLAB01/871710
Command:  CALL PGM(RSELAB01/PAYROLLG)

01/001
```

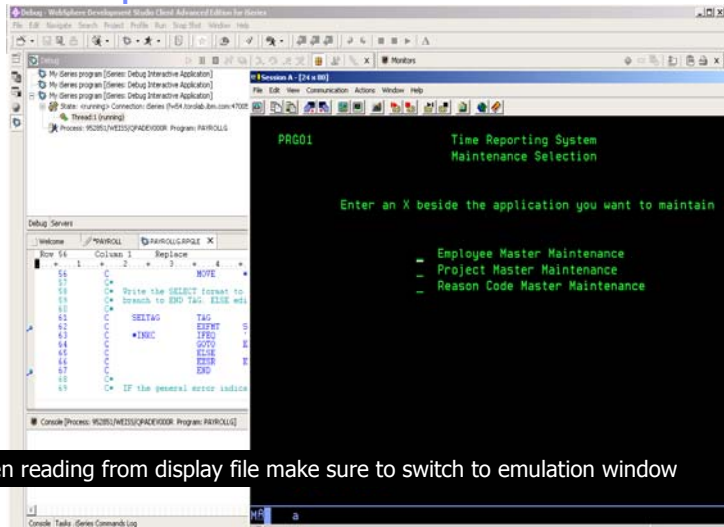
STRRESVR
command

Message removed
when session established

Once the STRRESVR command has been run, your 5250 session is associated with the RSE server and blocked from other use. The screen will look similar to the one above.

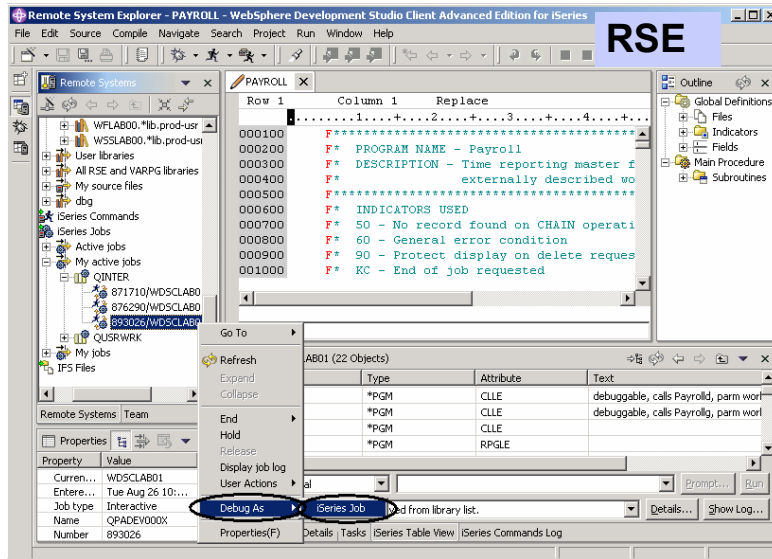
If you want control over the 5250 session back, use the pop up menu of one of the RSE subsystems (iSeries Objects, iSeries Jobs, iSeries Commands or IFS Files) and select 'Release Interactive job'.

Program output to emulation window



When reading from display file make sure to switch to emulation window

When you run an interactive program, the program waits for input from the 5250 emulation session.



This is the Remote System Explorer perspective. The Remote System view is the primary drill-down view, similar to PDM. You double-click a member to open the built-in LPEX editor shown in the middle. Notice the Outline view to the right. When you click on an element in the Outline view, the cursor is positioned to that element in the editor. The Table view below shows a PDM like list of the selected object . There are other tabs behind the Table view such as the Error List, and the iSeries Commands Log. As you can see there are numerous useful views in the Remote System Explorer perspective. You can launch the debugger from a pop up menu which is available for programs and service programs in the RSE view or the Table view as well as from the pop up menu of a job.

iSeries Projects Perspective

The screenshot displays the iSeries Projects Perspective in the WebSphere Development Studio Client Advanced Edition for iSeries. The interface is divided into several panes:

- Project Navigator (Left):** Lists local files in the project, including source files like `QCCLLESRC`, `PAYROLL.C.CBLE`, and `COMPARE.*file.pf-src`. A callout box indicates it can be used to "Expand to work with local source".
- Editor (Center):** Shows the source code for a program, with a callout box identifying it as the "RSE Lpex Editor".
- Outline View (Right):** Provides a hierarchical view of the program's structure, with a callout box labeled "Outline View".
- Build Monitor (Bottom):** A table for monitoring build jobs, with a callout box labeled "Build Monitor".
- Remote Systems View (Bottom Left):** A list of remote systems, with a callout box labeled "Remote Systems view".

The bottom of the window shows the "Remote Systems | Properties" and "Tasks | iSeries Commands Log | iSeries Build Job Status" panes.

This is the iSeries Projects perspective. The navigator view on the left is the primary view that drives the other views. It lists all the local files in the project. The LPEX editor is the same rich editor we saw for the Remote System Explorer.

When editing is complete and the project is pushed to the library and built, the build job is monitored in the job status window. When the build job is finished, you select the job and right click to see it's error list, which uses the same iSeries Error List window as the Remote System Explorer.

The debugger can be invoked from the pop up menu of the RSE view in the Project perspective.

Table of contents

Overview

Debugger Startup from the Remote Systems
view

 **Debug Perspective**

Debugger Functions

Launch Configurations and Settings

Demo

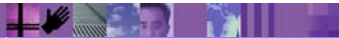


Table of contents

Overview

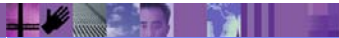
Debugger Startup from the Remote Systems
view

Debug Perspective

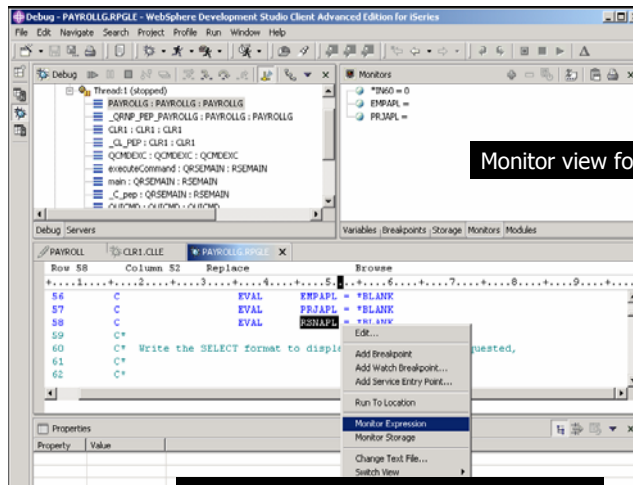
❖ **Debugger Functions**

Launch Configurations and Settings

Demo



Display Variable Content

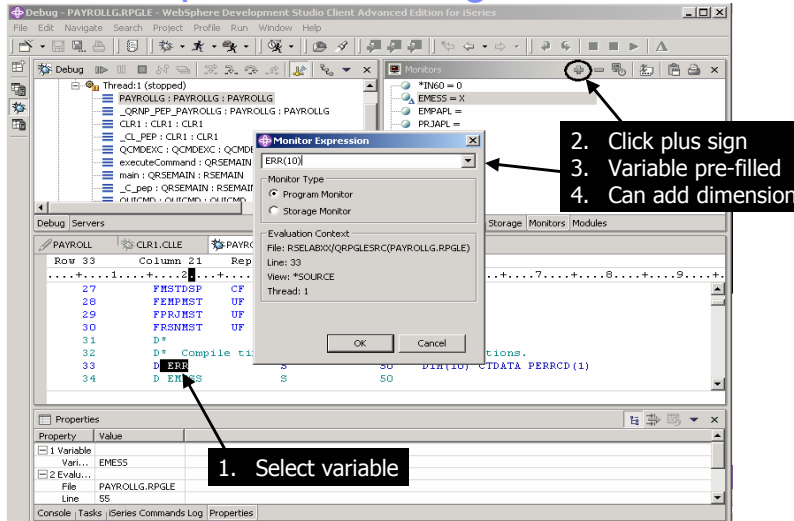


Monitor view for variable content

Selecting variable to display their content

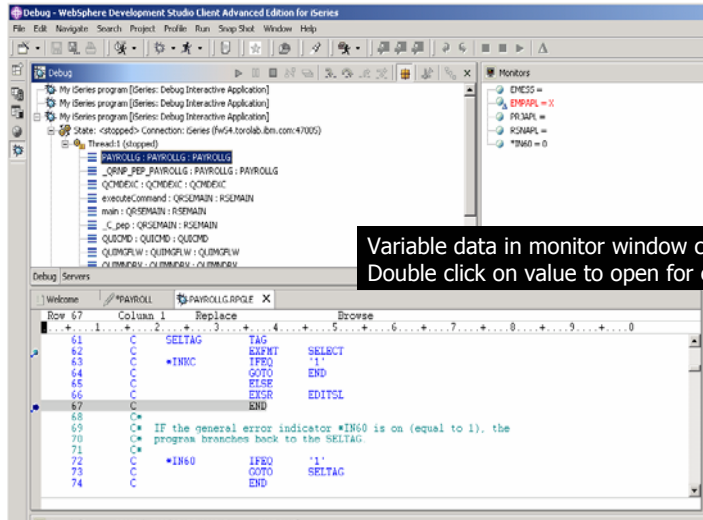
To monitor a variable, select it in the source by double clicking and use the Monitor Expression menu option.

Monitor Expression Dialog



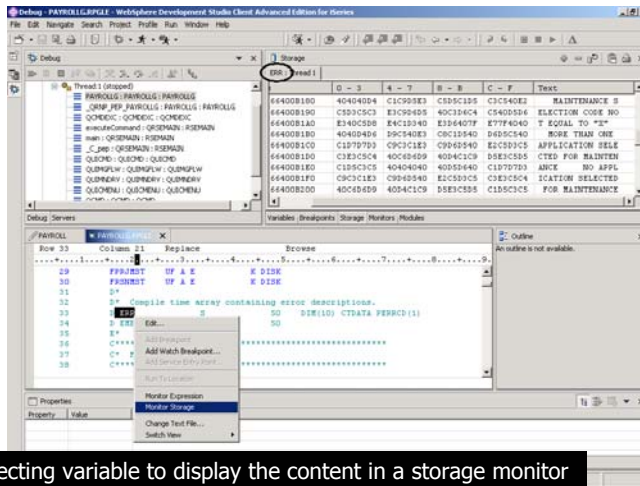
You can also monitor a variable from the Monitor Expression dialog, which is available from the Monitors view. The dialog is pre-filled with any variable that is selected in the source. This dialog is especially useful when you want to monitor one specific array element or an element of a structure.

Editing Variable Data



When the value of a variable changes, it will be highlighted in red. You can change the value of a variable while debugging by double clicking on the value in the Monitors view. You can also change the representation of the variable for example to hexadecimal.

Display Storage Content

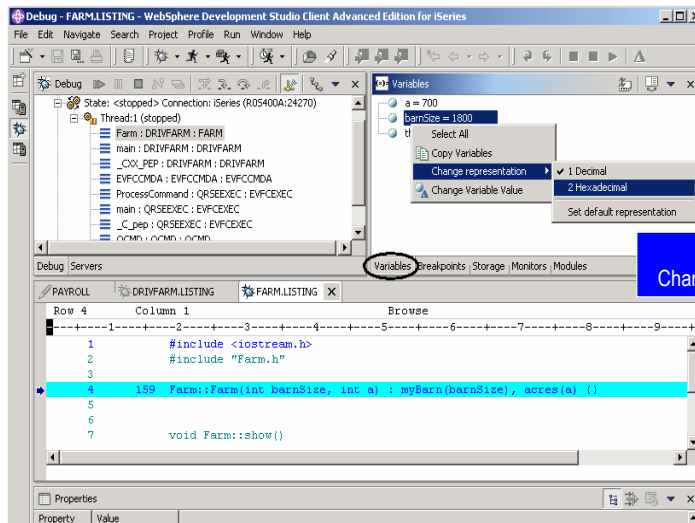


Selecting variable to display the content in a storage monitor

To display the storage starting with the address of a selected variable use the Monitor Storage menu option.

Note: Teraspace enabled programs will allow you to modify the storage content.

Local Variables



Change Value
Change Representation

The Variables view contains the local variables that are currently in scope.
Note: This feature is supported for C, C++ and Java.

Setting Breakpoints

Setting Breakpoints:

- Double click in prefix area
- Prefix area pop up menu
- Source pop up menu
- Breakpoints view pop up menu

Row	Column	Replace	Browse
61	C*		
62	C*		
63	C	DOU not *IN60	
64	C	EXHBT SELECT	
65	C	*INCK IFEQ '1	
66	C	LEAVESR	
67	C	ELSE	
68	C	EXSR	

You can only set breakpoints at executable lines. All executables lines are displayed in blue.

Conditional Breakpoints

The image displays two screenshots of the 'Add a Line Breakpoint' dialog box in the IBM iSeries Integrated Debugger. The left screenshot shows the 'Required information' tab, and the right screenshot shows the 'Optional parameters' tab.

Required information
Sets a breakpoint to stop execution at a specific source line

Program: *PGM RSELAB01/CLR1
Module: CLR1
Views: *SOURCE, *LISTING, *STATEMENT
Source(optional): CLR1.CLE
Line: 8

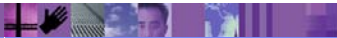
Optional parameters
Make the breakpoint conditional upon the following parameters

Tthread: Every
Frequency: From: 99, To: Infinity, Every: 1
Expression:

You can also set conditional breakpoints. The frequency allows you to limit the number of stops. Specifying an Expression will only stop program execution when the condition is true. The type of expression allowed depends on the programming language.

Step, Run, Run To Location

- **Step Into**
 - Debug the next call level
- **Step Over**
 - Run the next call level and stop at the next statement
- **Resume**
 - Run until an event is encountered
- **Run To Location**
 - Run and stop at the current cursor position



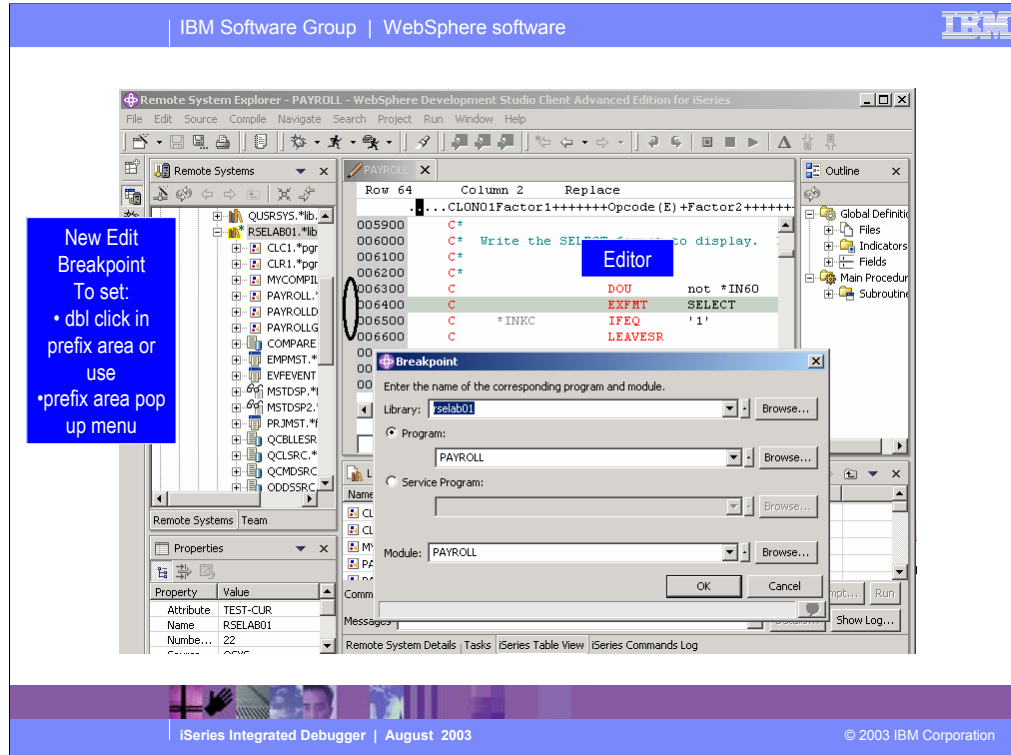
Setting Watch Breakpoints

The screenshot shows the IBM WebSphere Development Studio Client Advanced Edition for iSeries. The main window displays a source code editor with a context menu open over a line of code. The menu options include 'Add watch breakpoint', 'Add storage breakpoint', 'Run to Location', 'Monitor Expression', 'Monitor Storage', 'Change Text File...', and 'Switch View...'. A callout box on the right explains the steps for setting watch breakpoints.

Setting Watch Breakpoints:

- Double click variable and
- Source pop up menu or
- Breakpoints view pop up menu

Watch breakpoints allow you to stop program execution when the value of the watched variable changes. The program stops at the line after the change occurred. All breakpoints are listed in the Breakpoints view and can be manipulated from there, i.e. they can be disabled, enabled, removed and line breakpoints can be modified.



This is the dialog for edit breakpoints. Enter the library name and select Program or Service Program. The breakpoint is shown as a dot in the prefix area. If there is an active debug session, the breakpoint will be set in that session and there will be no marker in the editor. The breakpoint is then listed in the Breakpoints view in the Debug perspective.

Debugging in V5R2

- In V5R2 you can set Service entry points
 - You know the program you want to debug but the program gets started in a server environment
 - You know the userid the program will be running under
 - You don't know the job name where the program will be running
- To set a service entry point
 - Debug the program
 - Identify the line you want to stop at
 - Set the Service entry point
 - Start the program

You use service entry points when you wish to debug an application that makes use of the Toolbox or multiple jobs. Examples of cases where you would want to use a service entry point include:

- Applications that are invoked by a Toolbox program call. In this case, you would set a service entry point in the application that will be called by the Java application. When the application is called and the code where the service entry point is set is about to execute, the debugger can take control of the application and stop at that line. With this technique, you can put the program invoked by the Toolbox under debug when you do not know which job it will be running in.
- Programs that are spawned by other programs. In this case, you would set a service entry point in the application that will be spawned. When the program is spawned and the line where the service entry point is set is about to execute, operation will be suspended and the debugger will be able to gain control of the program and stop at that line. When a service entry point is set, it is triggered when the application not currently under debug is called.

To set a service entry point, start a debug session for the program. Select **Add Service Entry Point** from the pop-up menu of the source or the prefix area for the line where you want your program to stop. This will invoke the **Add Service Entry Point** dialog box, which displays the program, module, source file, and line number of the service entry point that will be created. In this dialog box, specify the user profile for which the service entry point will be activated. By default, the user profile is set to *CURRENT (the user profile for the current debug session).

Note: Whenever the program gets invoked with the specified userid regardless in which job it is running, the program will be stopped at the line with the Service Entry breakpoint and the Integrated Debugger on your workstation will start a debug session for the specified program.

Debugging in V5R2 (continued)

- Great for:
 - Any batch program you want to debug
 - WebFaced applicatons
 - Web applications
 - Toolbox call

You can debug all sorts of applications such as any batch program, a WebFaced application or a Web application.

Programs/Modules view

The screenshot displays the IBM Series Integrated Debugger's Programs/Modules view. The main window shows a tree structure of loaded programs and modules. A context menu is open over a selected entry, with options to 'Add Program', 'Remove Program', and 'Display Source'. The 'Add Program' dialog box is also visible, allowing the user to specify a program name and type (Program, Service Program, or Java Class). The console window at the bottom shows the current execution state, including thread information and a list of active programs.

Annotations in the image provide the following instructions:

- Click plus to add a Program, Service Program or Java Class
- pop up menu to remove programs from debug
- Expand and double click to display source

The Programs view lists all programs, modules and procedures of the current debug session. You can use the Add Program dialog to add programs, service programs and Java classes to your debug session. Click the plus sign to bring up the dialog. The pop up menu of a program, service program or Java class allows you to remove the selected entry from the debug session. The initial program and the one you are currently stopped at cannot be removed. Double clicking on a source or procedure entry displays its source.

iSeries specific Help

Help ->
Help
Contents

The screenshot shows the 'Contents' pane on the left with 'iSeries application developer' and 'Debuggers' circled. The main pane displays 'Optional breakpoint parameters' with a table and descriptive text.

Optional breakpoint parameter	Description	Type of breakpoint supported
Thread	This selection list lets you choose what threads to set the breakpoint in. To select a thread from the list, highlight the thread where you want to set the breakpoint. This list is available only on platforms that support multithreaded programs.	This parameter is supported by line and watch breakpoints.
	Use the Frequency controls to tell the debugger when to stop on a breakpoint and when to skip it. The debugger keeps track of how many times each breakpoint is encountered. The fields in this section tell the debugger on which encounter of a breakpoint the debugger should first stop, how often it should stop, and on which encounter the debugger should no longer stop. The following parameters are used to set the breakpoint frequency:	
From	Enter the first breakpoint encounter you want the debugger to stop on. For example, if you want the debugger to skip over the breakpoint the first five times it is encountered, enter "6".	This parameter is supported by...

Table of contents

Overview

Debugger Startup from the Remote Systems
view

Debug Perspective

Debugger Functions

➤ **Launch Configurations and Settings**

Demo



Launch Configuration

Default name: My iSeries Program

Debug Launch configuration

- Persist debug session information
- Specify Service program to be debugged

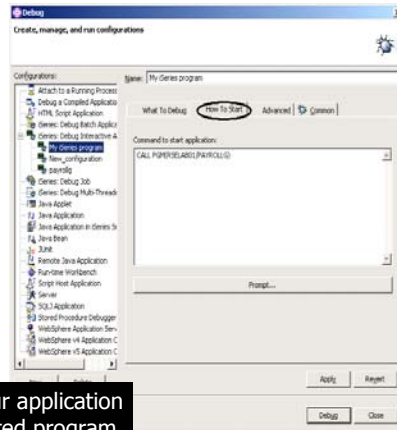
iSeries Integrated Debugger | August 2003

© 2003 IBM Corporation

You can start the Debugger in several ways: direct from the Remote Systems view, or from the Launch Configuration dialog. Starting directly from the pop up menu does not allow you to specify parameters to be passed to the program. The Launch Configuration dialog allows you to modify how the program is invoked including to specify parameters.

When you invoke the debugger from the pop up menu, a default launch configuration called My iSeries program is created with the information of the selected object.

Launch Configuration – How to Start

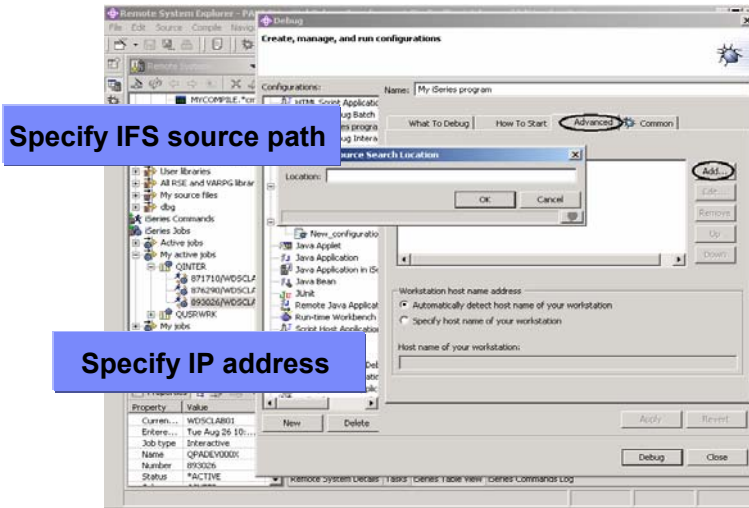


Command to start application:
required for service programs

How to invoke your application
Default: Call selected program

In the How To Start page of the Launch Configuration you specify the command that invokes your application. By default, this page contains a call command for the selected program. You can modify this command to add parameters or you could specify a different command or program that invokes your application. You can use the Prompt button to get a prompt dialog for the specified command.

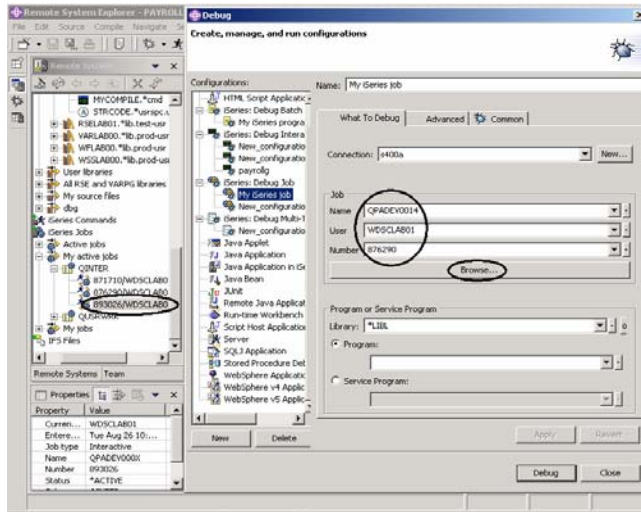
Launch Configuration – Advanced Page



The Advanced page allows you to set an IFS source path so that the debugger does not have to prompt you for the source location. The different elements of the path are listed in form of a table. The buttons on the right enable you to reorder the entries, edit or remove them.

Normally the debugger is able to resolve the address or hostname of your workstation. However, if your workstation has more than one IP address, for example when you are connecting to the iSeries via VPN,

Debug Job



Selecting Debug Job from the pop up menu will create a launch configuration with the name My iSeries job. If you selected a job in the RSE, this job will be put into debug mode and when prompted, you start your application in that job. Step into is selected which means that the debugger stops at the first executable statement it encounters. Terminate debug session on program completions is not available in this case since the debugger was started for a job, not a program.

If you want to debug a program in a specific job, you have to use the iSeries: Debug Job Launch Configuration and specify job and program.

Note: There is no 'How To Start' page in the job launch configuration. It is up to the user to start the application in the specified job when prompted to do so.

Library List

Initial library list from user profile

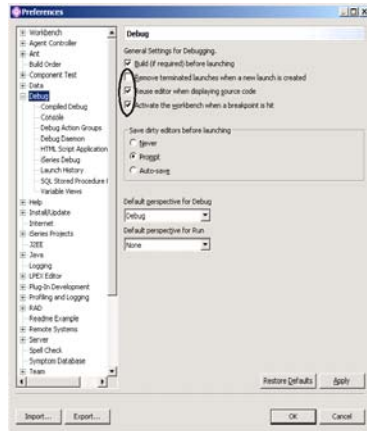
- **Change for this session from pop up menu of Library List**
 - Add Library List Entry
 - Change Current Library
- **Change in Properties from pop up menu of any subsystem**
 - Initial Library List node**
 - Add Library List Entry
 - Change Current Library
 - Re-order library list entries

Changes to the library list affect the RSE server job as well as jobs submitted to Batch and BCI jobs created to debug multi-threaded applications.

The interactive RSE session uses the library list that is set for the 5250 session before the STRRSESVR command is run. Added to that list are the libraries that are set in the Properties. Changes to the properties settings will be used after the connection has been disconnected and connected again.

Tip: You can create multiple connections to the same iSeries host and set different properties for each one.

Debug Preferences

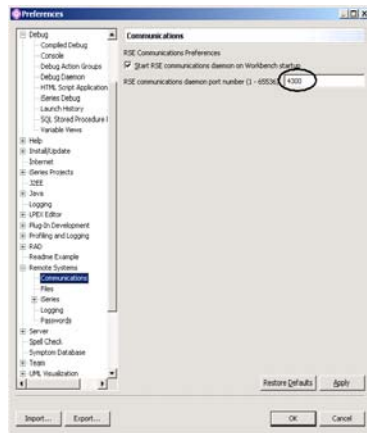


**Select:
Windows -> Preferences**

Check the 'Reuse editor when debugging source code' check box if you want only one tab for multiple source in the debug editor. Switching to a different source can then be done from the call stack or the Programs view. If de-selected, each source will have its own tab and selecting its tab will display the source.

Check the 'Remove terminated launches when a new launch is created' check box, to delete the messages in the call stack that belong to terminated debug sessions. That way only the currently active sessions are listed in the call stack.

Preferences – Remote Systems



**Select:
Windows -> Preferences**

**Remote Systems
Communications
RSE port number**

The port number specified in Remote Systems -> Communications is used by the RSE communications daemon which includes the debugger communication. If you are working from behind a firewall, opening the specified port will give you access to the debugger.

Table of contents

Overview

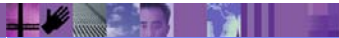
Debugger Startup from the Remote Systems
view

Debug Perspective

Debugger Functions

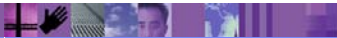
Launch Configurations and Settings

 **Demo**



Summary

- Source level debugger
- Fully integrated
- Common user interface
- Off-load host
- Improved developer productivity



Additional Information

- **homepage:**
<http://ibm.com/software/adwtools/iseries>
Select Library link for Labs, Tutorials, Presentations
- **newsgroup:**
<news://news.software.ibm.com/ibm.software.websphere.code400>



Trademarks & Disclaimers

© IBM Corporation 1994-2003. All rights reserved.

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

AS/400	IBM (logo)
AS/400e	iSeries
e (logo) business	OS/400
IBM	

Lotus, Freelance Graphics, and Word Pro are registered trademarks of Lotus Development Corporation and/or IBM Corporation. Domino is a trademark of Lotus Development Corporation and/or IBM Corporation.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Information is provided "AS IS" without warranty of any kind.

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.

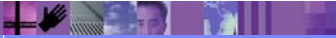
Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

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 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 throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.



Disclaimer

- **Acknowledgement:**

This presentation is a collaborative effort of the IBM Toronto iSeries Application Development presentation team, including work done by:

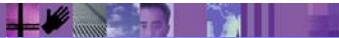
Vadim Berestetsky, Phil Coulthard, George Farr, Claus Weiss, Don Yantzi,

- **Disclaimer:**

The information contained in this document has not been submitted to any formal IBM test and is distributed on an as is basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customers' ability to evaluate and integrate them into the customers' operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

- **Reproduction:**

The base presentation is the property of IBM Corporation. Permission must be obtained PRIOR to making copies of this material for any reason.





IBM Software Group

Thank you for coming.
Enjoy IBM WebSphere Development Studio
Client for iSeries!

WebSphere. software



August 2003 | WDS V5R2 and WDSC 5.1

© 2003 IBM Corporation