



# Tutorial: Maintaining an ILE RPG or ILE COBOL application

*Version 5.1*



IBM WebSphere Development Studio Client for iSeries



# Tutorial: Maintaining an ILE RPG or ILE COBOL application

*Version 5.1*



---

# Contents

<b>Introduction to this tutorial</b> . . . . .	<b>v</b>
Learning objectives . . . . .	vi
Prerequisite knowledge . . . . .	vii
<b>Chapter 1. Reviewing the 5250 payroll application</b> . . . . .	<b>1</b>
Starting the 5250 application . . . . .	1
Maintaining an employee record. . . . .	2
Maintaining a project code. . . . .	2
Maintaining a reason code. . . . .	4
Recap. . . . .	4
<b>Chapter 2. Introducing Development Studio Client and iSeries Development Tools</b> . . . . .	<b>5</b>
Introducing Development Studio and Development Studio Client . . . . .	5
Introducing iSeries Application Development Tools . . . . .	7
Remote System Explorer . . . . .	7
LPEX Editor . . . . .	7
Shells and Commands in the Remote Command view . . . . .	7
Program Verifier . . . . .	8
CODE Designer . . . . .	8
iSeries Debugger . . . . .	8
Checkpoint . . . . .	9
Recap . . . . .	11
<b>Chapter 3. Starting Development Studio Client</b> . . . . .	<b>13</b>
Starting Development Studio Client . . . . .	13
Checkpoint . . . . .	14
Recap . . . . .	15
<b>Chapter 4. Opening a perspective</b> . . . . .	<b>17</b>
Introducing a perspective. . . . .	17
Opening the Remote System Explorer perspective . . . . .	17
Checkpoint . . . . .	18
More practice. . . . .	19
Recap . . . . .	19
<b>Chapter 5. Configuring a connection to an iSeries system and connecting to an iSeries</b> . . . . .	<b>21</b>
Configuring a connection to an iSeries system. . . . .	22
Connecting to an iSeries system . . . . .	24
Viewing and accessing objects in the Remote Systems view. . . . .	27
Opening a second source member. . . . .	31
Displaying an outline of a structured file . . . . .	31
Checkpoint . . . . .	33
More practice. . . . .	35
Recap . . . . .	35

<b>Chapter 6. Editing Source</b> . . . . .	<b>37</b>
Introducing the editor . . . . .	38
Editing columns in RPG source. . . . .	38
Entering SEU commands . . . . .	41
Requesting undo and redo operations . . . . .	41
Invoking language-sensitive help . . . . .	42
Prompting RPG language specifications . . . . .	49
Indenting RPG source . . . . .	51
Finding and replacing text . . . . .	53
Filtering lines by string . . . . .	54
Filtering lines by type . . . . .	57
Searching multiple files . . . . .	58
Comparing file differences from the Remote Systems view. . . . .	60
Checking syntax. . . . .	63
Checkpoint . . . . .	67
More practice. . . . .	70
Recap . . . . .	70
<b>Chapter 7. Verifying source</b> . . . . .	<b>71</b>
Verifying RPG or COBOL source . . . . .	71
Fixing RPG errors . . . . .	73
Fixing COBOL errors . . . . .	75
Saving the source and verifying source again . . . . .	76
Checkpoint . . . . .	76
More practice. . . . .	77
Recap . . . . .	77
<b>Chapter 8. Compiling source</b> . . . . .	<b>79</b>
Compiling interactively . . . . .	79
Specifying RPG compile options . . . . .	81
Specifying COBOL compile options . . . . .	83
Submitting iSeries commands in the iSeries table view. . . . .	84
Running commands and programs . . . . .	86
Starting an interactive connection . . . . .	87
Running the payroll program . . . . .	88
Checkpoint . . . . .	89
More practice. . . . .	90
Recap . . . . .	90
<b>Chapter 9. Designing screens</b> . . . . .	<b>91</b>
Opening a DDS member . . . . .	92
Viewing the DDS tree . . . . .	92
Selecting the DDS object . . . . .	93
Designing the DDS screen . . . . .	94
Creating groups from existing records . . . . .	95
Creating new screens . . . . .	97
Adding fields to the subfile record. . . . .	98
Switching between multiple records . . . . .	101
Adding field error handling . . . . .	103
Accessing field properties . . . . .	105
Adding new keywords . . . . .	106
Verifying the source changes . . . . .	108

Switching between designing and editing the screen . . . . .	109
Compiling your source changes and closing the Designer . . . . .	110
Checkpoint . . . . .	111
More practice . . . . .	111
Recap . . . . .	112

**Chapter 10. Debugging a program . . . 113**

Introducing the iSeries Integrated Debugger . . . . .	113
Starting the integrated debugger . . . . .	114
Setting breakpoints . . . . .	122
Monitoring variables . . . . .	126
Stepping into a program. . . . .	128
Listing call stack entries . . . . .	130
Setting breakpoints in PAYROLLG or PAYROLLD . . . . .	131
Removing a breakpoint in PAYROLLG or PAYROLLD . . . . .	132
Monitoring variables in PAYROLLG or PAYROLLD . . . . .	133
Adding a storage monitor . . . . .	135
Setting watch breakpoints . . . . .	136
Closing the debug session . . . . .	139

Checkpoint . . . . .	139
More practice . . . . .	140
Recap . . . . .	140

**Chapter 11. Exploring Remote System Explorer . . . . . 143**

Introducing the Remote System Explorer . . . . .	143
Creating a library filter . . . . .	144
Creating an object filter . . . . .	147
Creating a user action . . . . .	151
Running commands from the Remote System Explorer . . . . .	156
More practice . . . . .	157
Recap . . . . .	158

**Chapter 12. Module checkpoint answers. . . . . 159**

**Appendix. Notices . . . . . 161**

Programming interface information . . . . .	162
Trademarks . . . . .	163

---

## Introduction to this tutorial

The iSeries™ server development capabilities of the new integrated development environment (IDE) provides many compelling reasons for you to upgrade from Application Development Toolset (ADTS) or CoOperative Development Environment (CODE).

The IDE reduces the learning curve by providing a consistent interface for developing server applications and e-business applications. This allows you to progress to new levels of application development. The IDE delivers on the promise of tool integration with best-of-breed tools from IBM® and several partners to support the end-to-end application development life cycle.

The IDE also includes a robust, easy-to-use development environment for creating, building and maintaining iSeries RPG, COBOL, C, C++ applications, and Web-enabled applications using the IBM WebFacing Tool.

ADTS has been the traditional method for developing and maintaining server-side iSeries applications. But now there is a new set of highly integrated and highly extendible tools for iSeries RPG, COBOL, C, C++, CL and DDS development. These new tools offer you a development experience that is consistent with the experience for developing Java™, Web, Web Services, and XML applications. The new tools also allow you to leverage the classic CODE tools for extremely rich editing and DDS design support. These new generation tools offer rich support for exploring the file system, compiling/building, editing, running, and debugging. They offer significant productivity and usability gains, support for disconnected and team development, and a common harness for the tight integration of IBM and partner-supplied tools for server development.

In this tutorial you will follow the typical iSeries application development life cycle tasks using a payroll application written in RPG or COBOL. You will work through a series of exercises that will teach you what the new tools are and how to use them.

By the end of this tutorial you will be able to realize significant productivity and usability gains in your iSeries server application development and be on your way to increasing your skills to make the transition into new programming models such as Java, Web, Web Services, and XML.

**Tutorial purpose:**

Before you begin, take a minute to read the learning objectives for the tutorial.

**Length of time:**

This tutorial takes approximately 2 hours to complete.

**Modules:**

Work on the modules in sequence. The pictures in the modules show similar tasks. Some of the names and icons may be different from the environment you will be working with when you complete the modules. Following each module it is recommended that you take the checkpoint quiz. Taking each quiz will help you to determine if you have mastered the module content and are ready to move on to the next module. The quiz answers are included at the end of this tutorial so you can check your

answers. Following each module checkpoint is an opportunity for more practice. This gives you the chance to apply what you have learned in the exercise. Often you will complete different but similar tasks to what you have just learned in the module. The Development Studio Client for iSeries online help can assist you in completing the practice tasks.

### **Prerequisites**

You should also learn about the prerequisite knowledge for the tutorial before beginning. This section describes the type of knowledge you need to already have to fully benefit from this tutorial.

---

## **Learning objectives**

In order to successfully edit, compile and debug the sample payroll application, you will need to understand many concepts. This tutorial is designed to help you learn about those concepts while working through maintaining a simple payroll application written in RPG and COBOL. In this tutorial, you will learn how to take the first step in modernizing your iSeries applications by moving to the next generation of iSeries application development tools. You will learn about the first application development tool which is the Remote System Explorer which has its own tools and views. Remote System Explorer is similar to Programming Development Manager (PDM) in that it allows you to drill down to the QSYS file system, or use filters to list specific objects within the QSYS file system. You will learn that the Remote Systems Explorer goes well beyond PDM however! You will learn how to explore iSeries jobs and commands, and the IFS file system. Further, you will learn that it can also be used to explore the file system of remote Linux, UNIX<sup>®</sup> and Windows<sup>®</sup> systems.

This tutorial is broken into eleven modules, each with their own specific learning objectives. Each module is composed of several exercises. These exercises must be completed in order to successfully achieve the module's goals.

The first module, Reviewing the 5250 payroll application, helps you learn about the 5250 payroll application before you use the new iSeries Development tools to maintain it. This application is written in ILE RPG and ILE COBOL.

The second module, Introducing Development Studio Client and Remote System Explorer, introduces you to IBM WebSphere<sup>®</sup> Development Studio for iSeries (Development Studio) and describes its relationship to IBM WebSphere Development Studio Client for iSeries. You learn which product makes up the host components and which product makes up the workstation components. You learn about the iSeries application development tools included with Development Studio Client for iSeries programmers. Finally, you learn about the Remote System Explorer, the launching point for the iSeries application development tools.

The third module, Starting Development Studio Client helps you learn about the workbench and workspace. You will then learn how to start Development Studio Client.

The fourth module, Opening a perspective helps you learn about a perspective, and specifically the Remote System Explorer perspective. You will then learn how to open the Remote System Explorer perspective.

The fifth module, Configuring a connection to an iSeries system and connecting to an iSeries, explains how to create a connection to an iSeries server and select objects using the Remote System Explorer perspective. You will learn the steps to



create a connection, how to find a library in the library list and finally how to open a member in the Remote Systems LPEX Editor. You will also learn about several views such as the Remote Systems view, iSeries Table view, and the Outline view.

The sixth module, Editing source, helps learn about some of the language features available in the Remote Systems LPEX Editor while you edit source.

The seventh module, Verifying source, explains how to verify source in the Remote Systems LPEX Editor. The program verifying is one of the final steps in compiling a program or module. When errors are found by verifying, the iSeries Error List appears. The iSeries Error List is a powerful tool that manages errors found by verify utility in the Remote Systems LPEX Editor. You will become familiar with the verify tool, the various capabilities of the iSeries Error List and the program that you will create.

The eighth module, Compiling source, explains how to compile source in the Remote Systems LPEX Editor. The program compiling step is one of the final steps in compiling a program or module. When errors are found by the compile step, the iSeries Error List appears. The iSeries Error List is a powerful tool that manages errors found by the compile utility in the Remote Systems LPEX Editor. You will become familiar with the compile tool, the various capabilities of the iSeries Error List and the program that you will create.

The ninth module, Designing screens, helps you learn about the various aspects of the CODE Designer while modifying a display file to add a screen.

The tenth module, Debugging a program, explains the Debug perspective and how to start the debugger, set breakpoints, monitor variables, run and step into a program, view the call stack in the Debug view, remove a breakpoint, add a storage monitor, and set watch breakpoints.

The eleventh module, Exploring Remote System Explorer helps you learn more about using the Remote System Explorer perspective to work with the iSeries objects that you used in the previous modules. You will learn how easy it is to define filters, perform actions and define your own actions. You will learn how Remote System Explorer can organize and integrate your work and make that work easier.

Be sure to read about the Prerequisite knowledge for the tutorial before beginning. This section describes the type of knowledge you need to already have to fully benefit from this tutorial.

---

## Prerequisite knowledge

In order to complete this tutorial end to end, you should already have working knowledge of the following:

- Basic Microsoft® Windows operations such as working with the desktop and basic mouse operations such as opening folders and performing drag-and-drop operations.

It is also useful, but not necessary, for you to have basic knowledge of the following:

- ILE RPG or ILE COBOL
- DDS

When you are ready to begin, start with the first module.

---

## Chapter 1. Reviewing the 5250 payroll application

In this module, you are going to explore the application by invoking it from a 5250 session and working with the applications green screens and command keys.

You will learn about the 5250 application, before you start to use the new iSeries Development tools to maintain it. This will allow you to become familiar with the 5250 screens and the behavior of the application.

Remember: Before beginning this module, you should learn about the Prerequisite knowledge for the tutorial.

In order to review the 5250 order entry application, there are several steps you will need to learn about and follow:

- Start the 5250 application
- Know what screens the application provides
- Know how the application works in its 5250 environment
- Know what command keys are supported and what functions they allow

In order to accomplish these learning objectives, there are several steps that are involved, including:

- Starting the 5250 application
- Maintaining an employee record
- Maintaining a project record
- Maintaining a reason code

The exercises in this module must be completed in order. Start with the first exercise when you are ready to begin.

**Length of time:**

This module will take approximately 5 minutes to complete.

---

### Starting the 5250 application

To start the order entry application you need a 5250 emulator on your workstation.

On the 5250 emulation screen on your desktop, displaying the sign-on screen:

1. Enter your User ID.
2. Enter your password.
3. On the command line of a 5250 screen, add the library RSELABXX to your library list.

```
ADDLIBLE RSELABXX
```

4. Now, invoke the Payroll application:

```
CALL PAYROLL
```

The application starts and shows the first panel. This first screen asks you to enter an X beside the application you want to maintain.

5. Choose the application that you want to maintain.

---

## Maintaining an employee record

To maintain an employee record:

1. Type x beside **Employee Master Maintenance**.



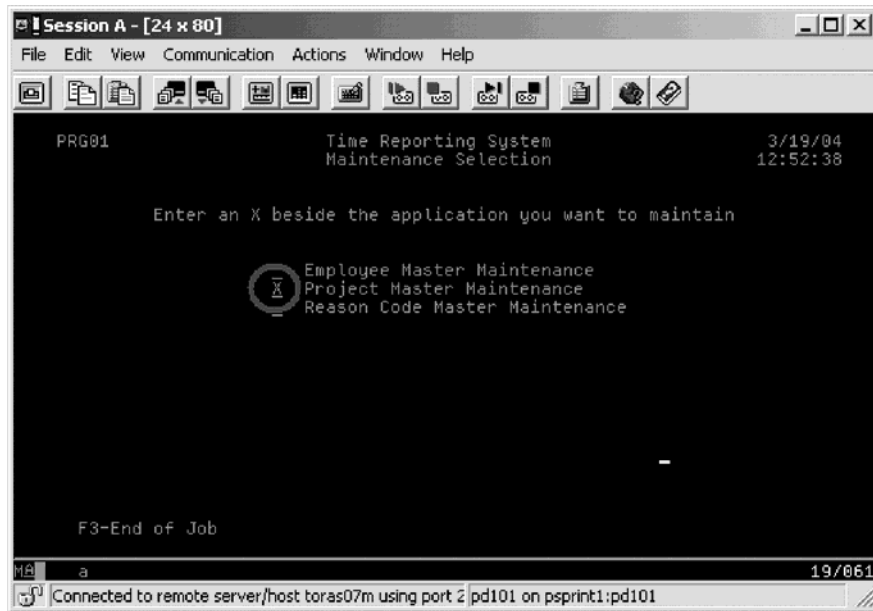
2. Press **Enter**.
3. Type 123 for the Employee Number.
4. Type A for the Action Code to add employee **123**.
5. Press **Enter**.
6. Type any information you like about the employee.
7. Press **Enter**.
8. Play in the employee records as much as you like.
9. Press **F4** to return to the maintenance selection.

---

## Maintaining a project code

To maintain a project code:

1. Type x beside **Project Master Maintenance**.



2. Press **Enter**.
3. Type 123 for the Project Code.
4. Type A for the Action Code to add project **123**.



5. Press **Enter**.
6. Type any information you like about the project.
7. Press **Enter**.
8. Play in the project records as much as you like.
9. Press **F4** to return to the maintenance selection.

---

## Maintaining a reason code

To maintain a reason code:

1. Type x beside **Reason Code Master Maintenance**.
2. Press **Enter**.
3. Type 123 for the Reason Code.
4. Type A for the Action Code to add reason code **123**.
5. Press **Enter**.
6. Type any information you like about the reason code.
7. Press **Enter**.
8. Play in the reason codes as much as you like.
9. Press **F3** to end the PAYROLL program.

---

## Recap

Congratulations! You have completed this module. You should now understand how to:

- Start the payroll application
- Maintain an employee record
- Maintain a project code
- Maintain a reason code
- Exit the application.

You have reviewed all the panels in the payroll application. Continue to the next module.

---

## Chapter 2. Introducing Development Studio Client and iSeries Development Tools

In this module, you will be introduced to IBM WebSphere Development Studio for iSeries (Development Studio) product and its relationship to IBM WebSphere Development Studio Client for iSeries. You learn which product makes up the host components and which product makes up the workstation components. You recognize the iSeries application development tools included with Development Studio Client for iSeries programmers. You then are introduced to Remote System Explorer the launching point for iSeries application development tools.

In order to gain a introductory understanding of Development Studio Client and the iSeries development tools, there are several steps you will need to learn about and follow:

- Describe the Development Studio product
- Describe the Development Studio Client product
- Describe how Development Studio Client fits into the WebSphere Studio family of products
- Explain the tools available to iSeries programmers for iSeries application development

In order to accomplish these learning objectives, there are several steps that are involved, including:

- Introducing Development Studio and Development Studio Client
- Introducing iSeries Development Tools

The exercises in this module must be completed in order. Start with the first exercise when you are ready to begin.

**Length of time:**

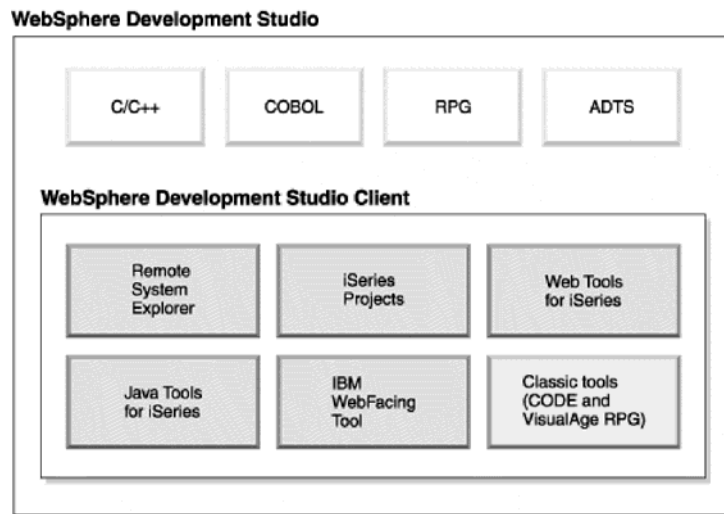
This module will take approximately 10 minutes to complete.

---

### Introducing Development Studio and Development Studio Client

With Development Studio Client, you can quickly develop and deploy traditional and e-business applications for your iSeries system. You receive unlimited licenses of this powerful suite of tools when you use Development Studio for your host development. The following diagram illustrates the interaction between host and

client tools:



Development Studio Client is designed to help you:

1. Develop and maintain iSeries applications using the Remote System Explorer
2. Develop Web GUIs for iSeries classic applications using the IBM WebFacing Tool and other Web Tools
3. Develop client and server applications for iSeries using Java Tools
4. Work with other Integrated Site Developer Tools (XML, Web Services, SQL, Relational Databases)

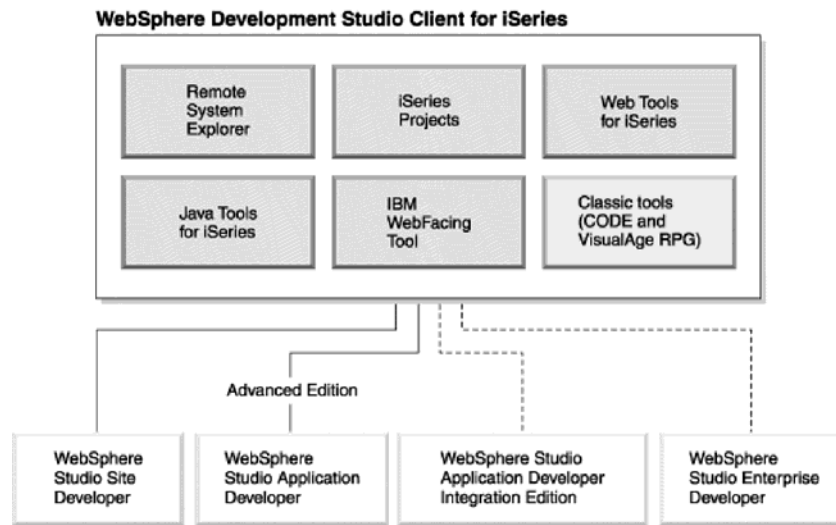
Now, You know Development Studio Client makes up the workstation tools while Development Studio makes up the host tools. But there's more. WebSphere Studio is IBM's solution for application and Web development. Both versions of our product come with an additional base WebSphere Studio product.

- IBM WebSphere Development Studio Client for iSeries includes WebSphere Studio Site Developer. Site Developer includes support for Web services, XML development tools, and core support for Java and Web development tools.
- IBM WebSphere Development Studio Client Advanced Edition for iSeries includes WebSphere Studio Application Developer. This base product provides end-to-end support for the creation and maintenance of J2EE applications and Web services. It also provides extensive support for Enterprise Java Beans, and for Java Messaging services.

And if you happen to have other WebSphere products installed you can also install either edition of the product on top of WebSphere Studio Application Developer - Integration Edition or WebSphere Studio Enterprise Developer. The following diagram illustrates how this product fits into the WebSphere Studio family of



products:



---

## Introducing iSeries Application Development Tools

Now, you know what the two flavors are of Development Studio Client and why you would want to use each one. Next let's look at those next generation iSeries server application development tools mentioned at the beginning of this tutorial. Just what are they and what do they do.

### Remote System Explorer

You can manage your development-cycle tasks in the Remote System Explorer. This is an enhanced and more flexible workstation version of Program Development Manager (PDM). You can create and manage development projects on your iSeries system from your Windows-based workstation with the Remote System Explorer and iSeries projects. With these tools, you can connect to an iSeries remote host, view iSeries libraries, files, and members. You can also launch the host compilers, the workstation editor, a program verifier and various debuggers all from the Remote System Explorer. This tool also supports other system types, such as UNIX(R), Linux, and Windows.

### LPEX Editor

Your program editing tasks are simplified with the Remote Systems LPEX Editor. This is a powerful language-sensitive editor that you can customize. Token highlighting of source makes the various program elements stand out. It has SEU-like specification prompts for RPG and DDS to help enter column-sensitive fields. Local syntax checking and semantic verification for your RPG, COBOL and DDS source makes sure it will compile cleanly on an iSeries system. If there are verification errors, an Error List lets you locate and resolve problems quickly. On-line programming guides, language references, and context-sensitive help make finding the information you need just a keystroke away.

### Shells and Commands in the Remote Command view

You can use the Remote Commands view to run and interact with commands and command shells on universal systems. A universal system includes Windows, Linux, and UNIX system types. Specifically, use the view to:

- Run commands in a command shell

- Display and interpret the output of a program
- Enter input to a program
- Display and manage different commands and shells from the same view. Multiple commands can be run in a single shell (one command at a time per shell), multiple shells may be run on a single system, and multiple systems may be running shells.

Whenever a command shell is launched or a command is run from the Remote System Explorer, the Remote Commands view displays the output and provides a way to work with that output.

## Program Verifier

One of the most powerful and unique features of the Remote System Explorer is the Program Verifier. Before you compile your code on an iSeries system, you can ensure that there are no errors by invoking the Program Verifier. The verifier checks for semantic (compile) errors on your workstation so that you can guarantee a clean compile on the iSeries. Think of the host cycles you'll save. It is especially handy when you are writing code but you are disconnected from an iSeries system. You can do this because Remote System Explorer ported the parsing and checking code from the iSeries host compilers to the workstation. The Error List window lists the errors that are found and their severity, inserts the error messages directly into the source and helps you to navigate between the errors.

## CODE Designer

Using an editor to create and maintain DDS source for your display and printer files can be a frustrating and difficult task. What would be great is a graphical design tool that lets you design your screens and reports visually and then generate the DDS source for you. Well, that's exactly what the CODE Designer does for you.

The CODE Designer interface was designed to help the novice DDS programmer create screens, reports and databases quickly and easily without worrying about the details of the DDS language, while at the same time letting the expert DDS programmer get access to all the features and power of the language. CODE Designer Is not fully integrated into the workbench yet, but you can launch it as a separate tool from the workbench.

## iSeries Debugger

With the integrated iSeries debugger you can debug an application that is running on an iSeries system. It provides an interactive graphical interface that makes it easy to debug and test your iSeries programs. It is fully integrated into the workbench. You can also set breakpoints before running the debugger, by inserting breakpoints directly in your source while editing. The integrated iSeries debugger client user interface also enables you to control program execution. For example, you can run your program, set line, watch, and service entry point breakpoints, step through program instructions, examine variables, and examine the call stack. You can also debug multiple applications, which may be written in different languages, from a single debug window. Each session you debug is listed separately in the Debug view.

---

## Checkpoint

Complete the checkpoint below to determine if you are ready to move on to the next module.

1. WebSphere Development Studio for iSeries:
  - a. Includes all four host compilers and all traditional tools (ADTS)
  - b. Includes all four host compilers, all traditional tools (ADTS) and unlimited licenses of the workstation-based tools named Development Studio Client
  - c. Includes only the workstation-based tools named Development Studio Client
  - d. Includes only the four host compilers
2. WebSphere Development Studio Client for iSeries Version 5 includes:
  - a. WebSphere Studio Site Developer Version 5 for e-business development
  - b. Cooperative development environment (CODE)
  - c. VisualAge® RPG
  - d. Java tools
  - e. Web tools
  - f. WebFacing tool
  - g. All of the above
3. WebSphere Development Studio Client for iSeries Advanced Version 5 includes:
  - a. WebSphere Studio Application Developer Version 5 for e-business development
  - b. Cooperative development environment (CODE)
  - c. VisualAge RPG
  - d. Java tools
  - e. Web Tools
  - f. WebFacing Tool
  - g. All of the above
4. WebSphere Studio Application Developer includes support for:
  - a. Creation and maintenance of J2EE applications
  - b. Creation and maintenance of Web services
  - c. Enterprise Java Beans
  - d. Java Messaging Services
  - e. All of the above
5. WebSphere Studio Site Developer includes support for:
  - a. Web services
  - b. XML development tools
  - c. Java tools
  - d. Web tools
  - e. All of the above
6. You can manage your development-cycle tasks in:
  - a. The Remote System Explorer
  - b. iSeries Projects
  - c. The IBM WebFacing tool
  - d. All of the above

7. With the Remote System Explorer and iSeries Projects you can view iSeries libraries, files and members. You can also launch the host compilers, the workstation editor, and various debuggers. (T, F)
8. Your program editing tasks are simplified with the:
  - a. The Remote System Explorer
  - b. iSeries Projects
  - c. The IBM WebFacing tool
  - d. The LPEX Editor
  - e. All of the above
9. The editor can access source files on your workstation or on your iSeries system directly. When a compilation results in errors, you can jump from the compiler messages to an editor containing the source. The editor opens with the cursor positioned at the offending source statements so that you can correct them. (T, F)
10. You can debug your program running on the iSeries system from your workstation using:
  - a. The Remote System Explorer
  - b. iSeries Projects
  - c. The IBM WebFacing tool
  - d. The LPEX Editor
  - e. The Integrated iSeries Debugger
  - f. All of the above
11. The graphical design tool that lets you design your screens and reports visually and then generates DDS source for you is:
  - a. The Remote System Explorer
  - b. CODE Designer
  - c. The IBM WebFacing tool
  - d. The LPEX Editor
  - e. The Integrated iSeries Debugger
12. Before you compile your code on an iSeries system, you can ensure that there are no errors by invoking the:
  - a. The Remote System Explorer
  - b. CODE Designer
  - c. The IBM WebFacing tool
  - d. The LPEX Editor
  - e. The Integrated iSeries Debugger
  - f. Program Verifier
13. You can use the Remote Commands view to:
  - a. Run commands in a command shell
  - b. Display and interpret the output of a program
  - c. Enter input to a program
  - d. Display and manage different commands and shells from the same view
  - e. All of the above
14. The integrated iSeries Debugger enables you to run your program, set line, watch, and service entry point breakpoints, step through program instructions, examine variables, and examine the call stack. (T, F)

15. The iSeries Projects perspective is designed more specifically to support structured programming, offline development, and team collaboration. (T, F)
  16. If the Advanced Version of the product is not installed on your workstation then you will not see the word Advanced in the Start menu. (T, F)
- 

## Recap

Congratulations! You have completed this module. You should now understand:

- What components make up Development Studio and Development Studio Client
- What components are included in the advanced edition of Development Studio Client
- What components are included in Application Developer
- What components are included in Site Developer
- What tool you use to manage your development cycle tasks
- What the benefits are of Remote System Explorer
- What tool is used to edit source members
- What tool is used to debug programs
- What tool is used to design screens
- What tool you used to save compile time
- The purpose of the Remote Commands view
- The capabilities of each iSeries development tool

Now that you have this introductory knowledge of Development Studio Client, you can continue with the next module.



---

## Chapter 3. Starting Development Studio Client

In this module you are introduced to workbench and the workspace. You then learn how to start Development Studio Client..

In order to get started with Development Studio Client and the Remote System Explorer, there are several steps you will need to learn about and follow:

- Explain workspace and workbench
- Start Development Studio Client

In order to accomplish these learning objectives, you are going to complete the step: Starting Development Studio Client.

The exercises within this module must be completed in order. Start with the first exercise when you are ready to begin.

**Length of time:**

This module will take approximately 5 minutes to complete.

---

### Starting Development Studio Client

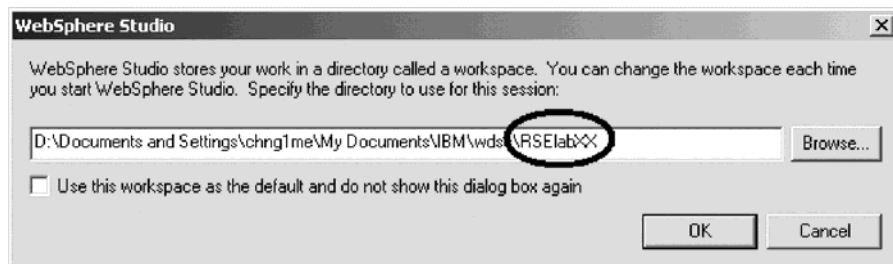
To start Development Studio Client:

1. Click **Start** on the task bar of your desktop
2. Select **Programs > IBM WebSphere Studio > Development Studio Client Advanced Edition for iSeries 5.1**



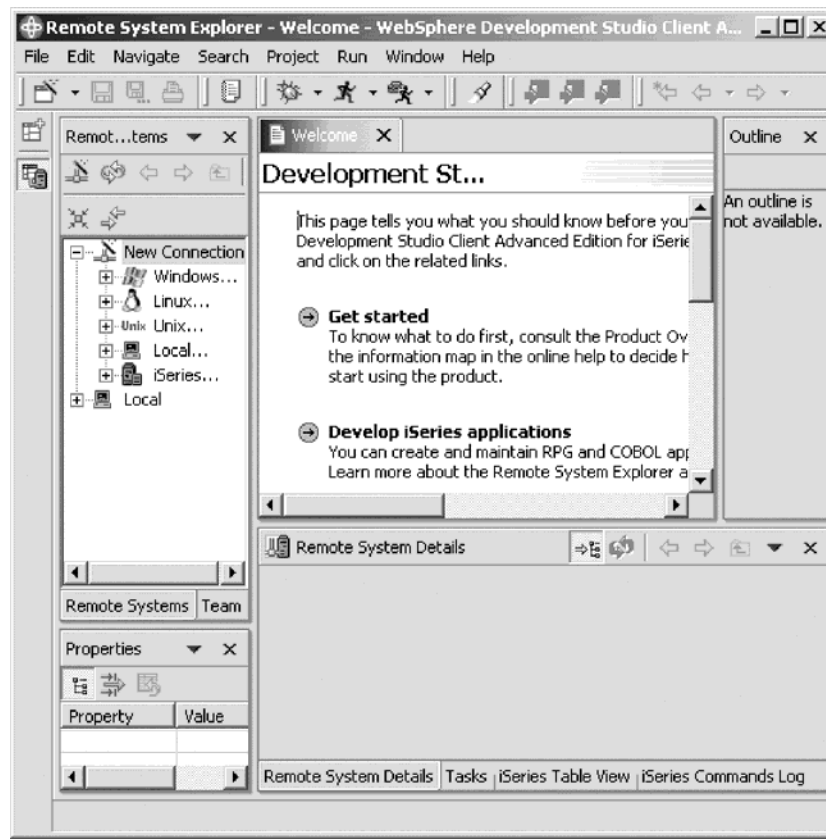
**Note:** If the Advanced Edition Version of the product is not installed on your workstation then you will not see the words "Advanced Edition" in the Start menu

3. A dialog may appear. Here you specify the name of the workspace where your projects and other resources such as folders, subfolders and files that you are developing in the workbench will reside.



4. (Optional) Change the field in this dialog and use a unique directory name, for example, RSElabXX (where XX is a unique number).

5. Click OK to open the workbench.



The workbench refers to the desktop development environment. The workbench aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workbench resources. Each workbench window contains one or more perspectives.

---

## Checkpoint

Complete this checkpoint to determine if you are ready to move on to the next module.

1. A workspace:
  - a. Aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workbench resources.
  - b. Defines the initial set and layout of views in the Workbench window.
  - c. Refers to the desktop development environment.
  - d. Specifies where your projects and other resources such as folders, subfolders and files that you are developing in the workbench will reside
2. A workbench:
  - a. Aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workbench resources.
  - b. Defines the initial set and layout of views in the Workbench window.
  - c. Refers to the desktop development environment.



- d. Specifies where your projects and other resources such as folders, subfolders and files that you are developing in the workbench will reside.
- e. A and C

---

## Recap

Congratulations! You have completed this module. You should now understand:

- The concepts of the workbench and the workspace
- How to start Development Studio Client

Now that you have started Development Studio Client, you can move on to opening a perspective. Continue to the next module.



---

## Chapter 4. Opening a perspective

In this module you are introduced to a perspective. You then learn how to open a perspective.

In order to open a perspective there are several steps you will need to learn about and follow:

- Explain perspectives
- Describe the Remote System Explorer perspective
- Open the Remote System Explorer perspective

In order to accomplish these learning objectives, you are going to complete the following steps:

- Introducing perspectives
- Opening the Remote System Explorer perspective

The exercises within this module must be completed in order. Start with the first exercise when you are ready to begin.

**Length of time:**

This module will take approximately 5 minutes to complete.

---

### Introducing a perspective

A perspective defines the initial set and layout of views in the Workbench window. Within the window, each perspective shares the same set of editors. Each perspective provides a set of capabilities aimed at accomplishing a specific type of task or working with specific types of resources. For example, the Java perspective combines views that you would commonly use while editing Java source files, while the Debug perspective contains views that you would use while debugging Java programs. Perspectives contain views and editors and control what appears in certain menus and tool bars.

---

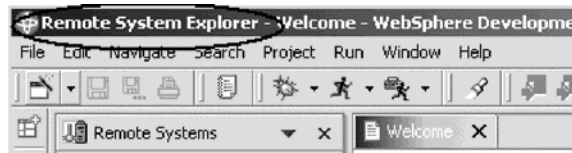
### Opening the Remote System Explorer perspective

When you develop and maintain iSeries application you will typically work in the Remote System Explorer perspective. This perspective is for an iSeries programmer to display the connections that you have already configured, create a new connection, connect to and disconnect from the connections that you have defined, work with iSeries files, commands, jobs, and integrated file system files.

This perspective will be active when you start Development Studio Client with a new workspace. If you had used the workspace before then, the workbench would come up with the perspective that you last opened. You will learn more about the Remote System Explorer perspective in the coming exercises as this is where you launch the iSeries programmer tools and views from the workbench.

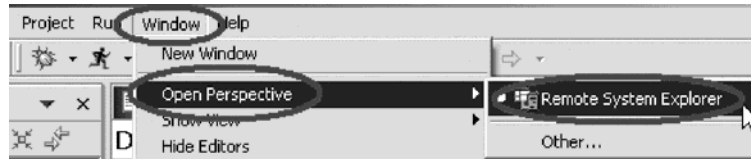
To open the Remote System Explorer perspective:

1. Check for the name of the perspective.



If you see a different perspective, not the **Remote System Explorer** open in the workbench or no perspective:

2. Click **Window > Open Perspective > Remote System Explorer** from the workbench menu.



The Remote System Explorer perspective opens.

---

## Checkpoint

Complete this checkpoint to determine if you are ready to move on to the next module.

1. A perspective:
  - a. Aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workbench resources.
  - b. Defines the initial set and layout of views in the Workbench window.
  - c. Refers to the desktop development environment.
  - d. Specifies where your projects and other resources such as folders, subfolders and files that you are developing in the workbench will reside.
2. Match the perspective with its correct definition:
  - a. Combines views that you would commonly use while editing Java source files
  - b. Contains views that you would use while debugging Java programs
  - c. Contains views that you would use while developing Web applications
  - d. Contains views that you would use while maintaining iSeries applications.
  - a. Java perspective
  - b. Web perspective
  - c. Remote System Explorer perspective
  - d. Debug perspective
3. In the Remote System Explorer perspective you can:
  - a. Display configured connections
  - b. Create a new connection
  - c. Connect and disconnect defined connections
  - d. Work with iSeries files, commands, jobs, IFS files
  - e. All of the above

---

## More practice

Want more practice? Try this!

Given your experience in opening the a perspective, open the Web perspective. See a list of tools and views available to the Web developer. Next open the Java perspective. See a list of tools and views available to the Java developer. Now since you are supposedly in the Java perspective, open the Web perspective. Be careful not to open another Web perspective.

Tip: Look in the workbench left-frame for the Web perspective icon. Now close both the Java perspective and the Web perspective.

---

## Recap

Congratulations! You have completed this module. You should know understand:

- The concept of a perspective
- The concept of the Remote System Explorer perspective
- How to open the Remote System Explorer perspective

Now that you have opened a perspective, you can move on to getting connected to an iSeries system. Continue to the next module.



---

## Chapter 5. Configuring a connection to an iSeries system and connecting to an iSeries

In this module, you will create a connection to an iSeries server and select objects. You will learn the steps to create a connection.

You will then learn how to find a library in your library list. Finally you open a member in the Remote Systems LPEX Editor. You also learn about several views such as the Remote Systems view, iSeries Table view, and the Outline view.

In order to configure and then connect, there are several steps you will need to learn about and follow:

- Explain the Remote Systems view
- Configure a connection to an iSeries system
- Connect to an iSeries system
- Describe what you need to set up a connection
- Describe a profile
- Find an object in the Remote Systems view
- Explain subsystems
- View and access objects
- Explain the Outline view
- Explain the iSeries Table View
- Explain the difference between the iSeries Table View and the Remote Systems view
- Show source physical file members in an iSeries Table view
- Lock and unlock the iSeries Table view
- Open a second source member
- View the outline of a file

In order to accomplish these learning objectives, there are several steps involved, including:

- Configuring a connection to an iSeries system
- Connecting to an iSeries system
- Viewing and accessing objects in the Remote Systems view
- Opening a second source member
- Displaying an outline of a structured file

The exercises within this module must be completed in order. Start with the first exercise when you are ready to begin.

**Length of time:**

This module will take approximately 10 minutes to complete.

---

## Configuring a connection to an iSeries system

When you first start the workbench and open a perspective, you are not connected to any system except your local hard drive on your workstation. To connect to a remote iSeries system, you need to define a connection. When you define a connection, you specify the name or IP address of the remote system and you give your connection a unique name that acts as a label in your workspace so that you can easily connect and disconnect. When you connect to the iSeries system, the workbench prompts you for your user ID and password on that host.

The first time you connect to an iSeries system, you need to define a profile.

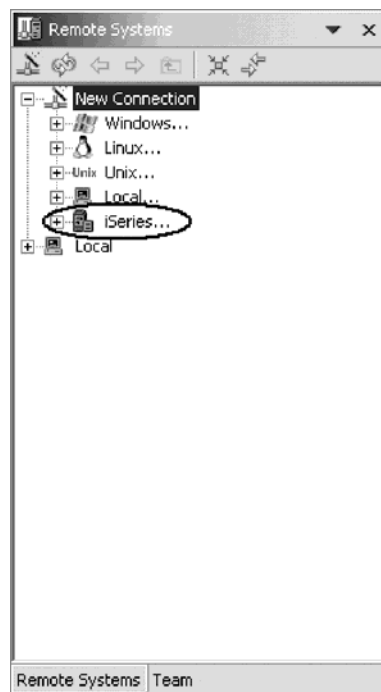
All connections, filters, and filter pools belong to profiles. We will describe filters in a later exercise. We will discuss profiles when you create your first connection.

Ok, let's get started.

Remember you have already opened the Remote System Explorer perspective in the previous module.

To configure a connection:

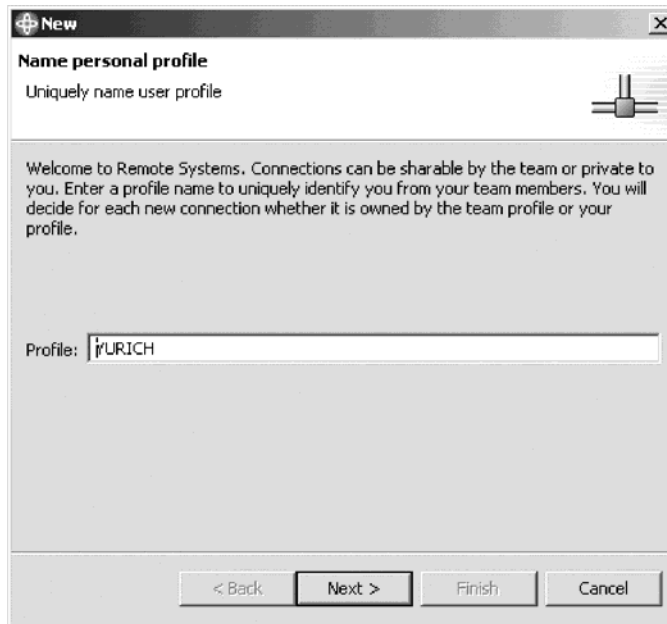
1. In the Remote Systems view, **New Connection** is automatically expanded to show the various remote systems types you can connect to through the Remote Systems view.



Expand **iSeries** to configure a connection to an iSeries system.



The Name personal profile page opens:

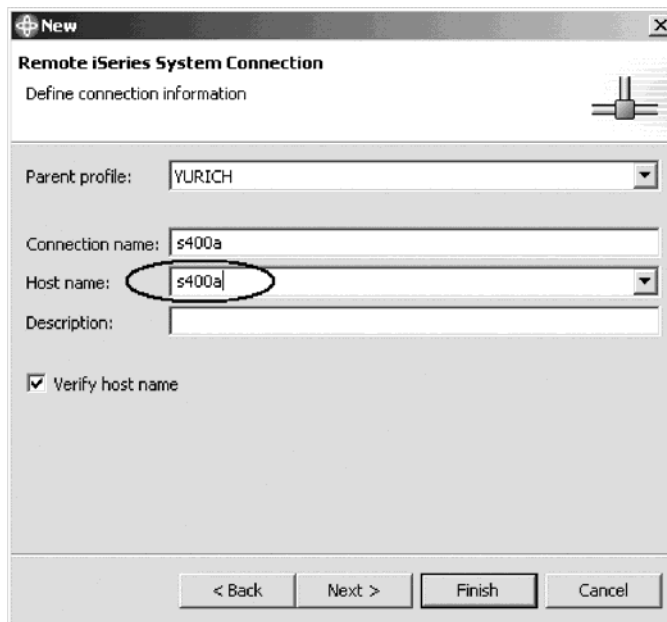


On this page of the New connection wizard and since it is the first time you connect to an iSeries system, you need to define a profile. All connections, filters, and filter pools belong to profiles. Profiles offer a way to group connections, share connections, or keep them private, and they help you partition data if you have a lot of connections or filter pools.

Your first profile will be for your local workstation. This is your personal profile. You use this profile as you want to keep your connection private. You do not want to share resources and information with other people.

2. Leave the **Profile** information default value. Click **Next**.

The Remote iSeries System Connection page opens:



On this second page you specify the information for your connection. The cursor on this page is positioned in the **Host Name** field.

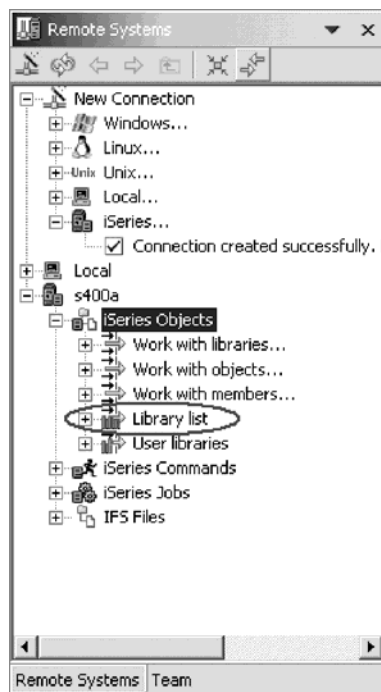
3. In the **Host name** field, type the name of your host system.  
The Connection name is automatically filled with the host name. Leave it this way. This name displays in your Remote Systems view and must be unique to the profile.
4. Leave the **Parent profile** default value. You don't need to change it. As the files will be kept private the parent profile defaults to the default profile shown on the previous page of the wizard. If you were sharing resources you would select a team profile that you would have created earlier.
5. Leave the **Verify host name** check box selected.
6. Click **Finish** to define your system.

---

## Connecting to an iSeries system

After you configure a connection to an iSeries system, you can easily connect and expand your new connection to reveal your subsystems. Subsystems are a function grouping of the various types of remote resources that can be explored in the remote system. There are four subsystems:

- iSeries Objects is a PDM-like group, allowing access to libraries, objects and members
- iSeries Commands allow you to predefine command sets each of which contain one or more often used commands. When run, all commands in a command set are sent to the remote system and executed, and the results are logged in the Commands view
- iSeries Jobs allow you to see various jobs, subset by job attributes, and to perform a limited number of operations on those jobs.
- IFS Files allow you to explore folders and files in the Integrated File System of the remote iSeries system



To connect to an iSeries system:

1. In the Remote Systems view, your new connection is expanded to reveal your subsystems. The **iSeries Objects** subsystem is the subsystem you will use most often! It is very similar to PDM, in that it allows you to access objects in the QSYS file system, and perform actions on those objects.
2. Notice the first three entries under the **iSeries Objects** subsystem are named after the PDM options, because they have similar capabilities:
  - **Work with libraries** (similar to WRKLIBPDM)
  - **Work with objects** (similar to WRKOBJPDM)
  - **Work with members** (similar to WRKMBRPDM)

In addition there are entries for working with library lists and user libraries:

- **Library list** (to simulate PDMs WRKLIBPDM LIB(\*LIBL) you can start with the pre-defined Library list filter, that when expanded lists all libraries in your library list.)
- **User libraries**

You also have more entries to work with under the connection itself and you can see from these entries that Remote System Explorer goes well beyond PDM! It allows you to explore iSeries jobs and commands and the IFS file system.

Now let's work with a library in our library list.

3. Expand the **Library list** folder.

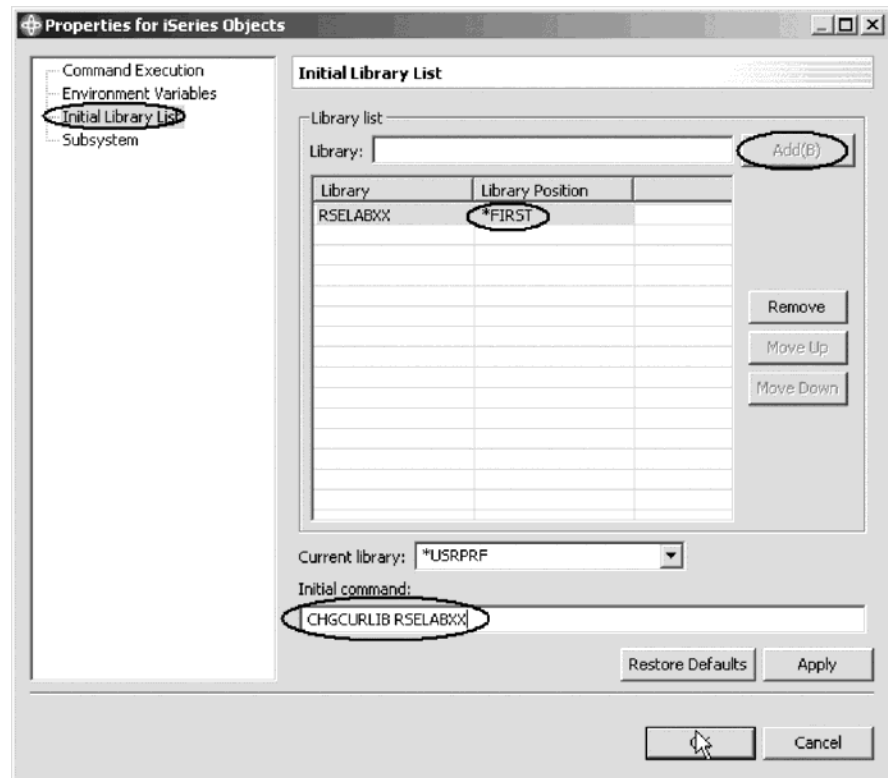
Now the connection will be activated and you will be prompted for a user ID and password.



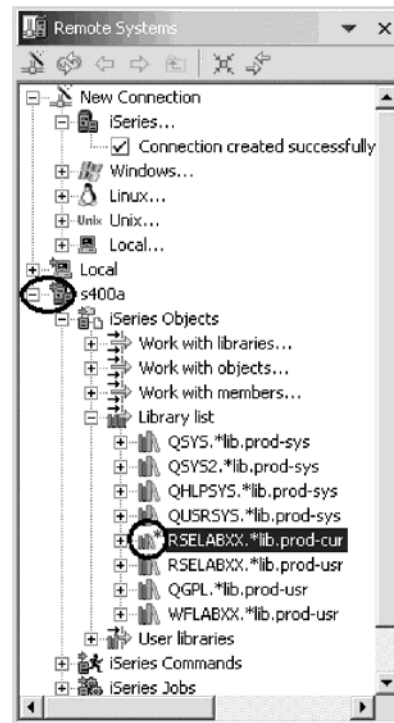
4. Enter your user ID and password.
5. Select the **Permanently change user ID** check box.
6. Select the **Save password** check box.
7. Click **OK**.

**Note:** Make sure that your user ID has been setup, so that your library has been added to the library list automatically. You can use the properties for iSeries Objects dialog to set connection information such as adding a library to a library list and changing the current library. You can right-click on iSeries Objects then click Properties to see this dialog. You

will need this setup to run jobs interactively from your workstation.



Back in the workbench you will see the libraries in your job's library list.



Notice that the s400a node now has a small green arrow in the icon to indicate it is an active connection.

Also your RSELABXX library should have a small green star that indicates it is the current library.

**Note:** If you do not see RSELABXX in your library list you can right-click on the Library list and use the Add Library List Entry window to add RSELABXX to your library list. If you also don't see RSELABXX as the current library you can right-click Library list and use the Change Current Library window to change the current library to RSELABXX.

For each library, you can right-click and select from a number of actions. For example, there is an action to create a new source file within the selected library, to refresh the contents of the library if it is expanded, to rename the library, copy the library or delete the library.

---

## Viewing and accessing objects in the Remote Systems view

Now you are ready to view and access objects in your current library RSELABXX.

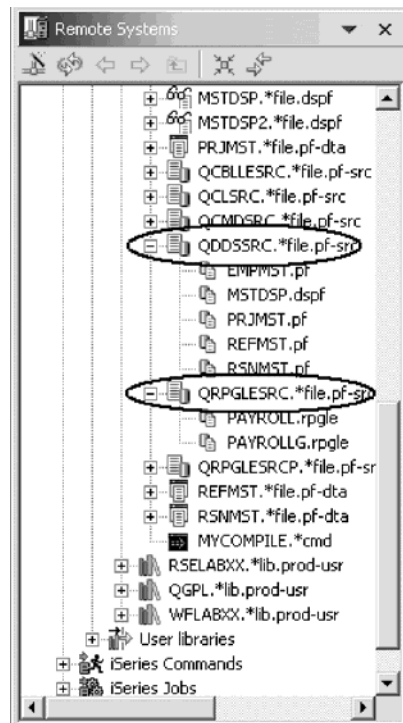
To view and access an object:

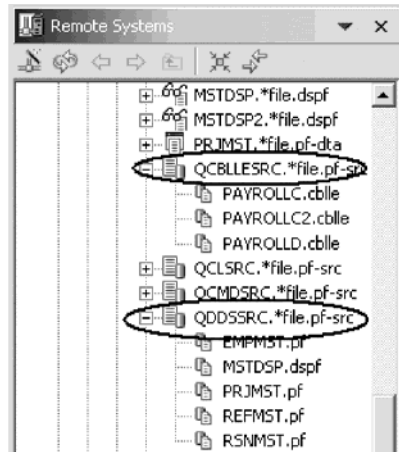
1. Expand library RSELABXX.

You will see all objects in this library appear in the Remote Systems view. For each object you can right-click and select from a number of actions. The list of actions depends on the object selected and whether you selected one or multiple objects.

For a source file the pop-up menu has an action to create a new member within the selected file, to refresh the contents of a file if it is expanded, to rename the file, copy the file and delete the file. These actions remotely run the appropriate iSeries command and you will see it logged in the Commands view.

2. Drill-down through the files in the Remote Systems view until you find QDDSSRC source file and then expand it.
3. Drill-down through the files in the Remote Systems view until you find QRPGLSRC source file if you want to work with RPG source or QCBLLSRC source file if you want to work with COBOL source and expand it as well.

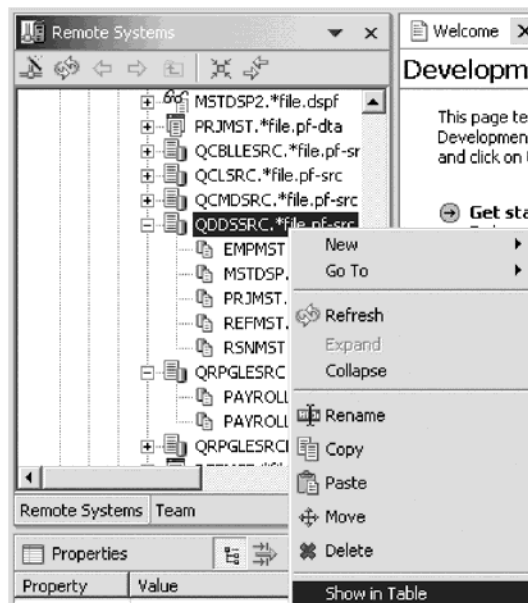




Now you can see and access the members in these two source files. For each member you can right-click and select from a number of actions. The exact list of actions depends on whether the member is a data file or source file and whether you select one or multiple members. For a member, the pop-up menu actions are edit, rename, copy, move, delete, and compile.

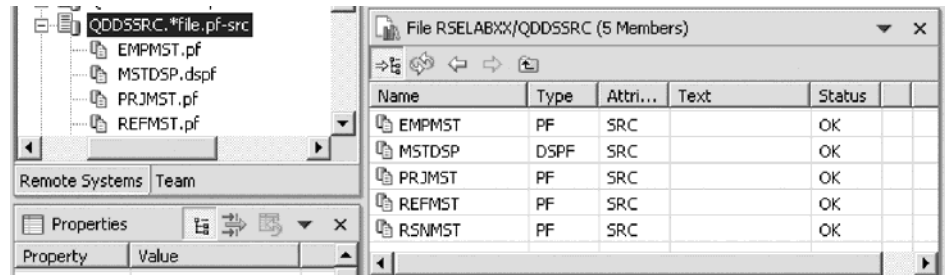
Before you go ahead and work with these members, let's see the members in the iSeries Table view as well because that is more similar to the view you are used to from PDM. You use this view to display a list of items, for example, members or objects, in a table format similar to PDM. You can also perform actions against these items such as editing and compiling.


4. Right-click the QDDSSRC file and then click **Show in Table** on the pop-up menu.



The iSeries Table view takes the selected object in the Remote Systems view as input, and displays the contents in the table. For source physical files, this step displays the members inside, their names, types, attributes, text descriptions,

and status.




The top of the iSeries Table view contains a lock icon  that controls the correlation between the Remote Systems view and the iSeries Table view.

If the lock is disabled (the lock icon's appearance is raised) then whenever you click an object or library in the Remote Systems view, the associated contents of that item automatically populate the iSeries Table view.

If the lock is enabled (the lock icon's appearance is depressed) then when you click on various items in the Remote Systems view, this view does not change the input to the iSeries Table view.

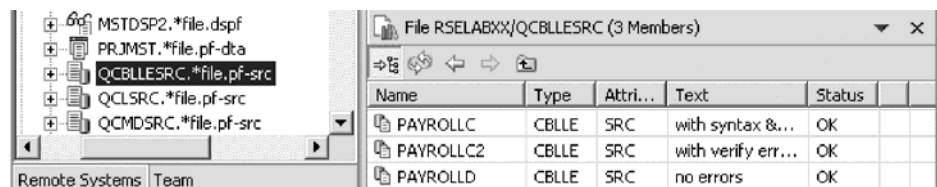
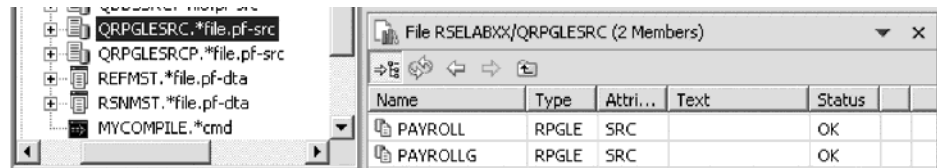
To enable or disable the lock, you can click it once to change its state. You can click on the columns heading to sort the view by column.

5. In the iSeries Table view toolbar make sure the lock/unlock button  is in the unlock position.

This means now the table will automatically be updated when a different object is selected in the Remote Systems view. This is a shortcut to open a pop-up menu for an object in the Remote Systems view and then clicking Show in Table.

6. In the Remote Systems view, select QRPGLSRC for RPG source or QCBLLSRC for COBOL source, right-click and click **Show in Table** on the pop-up menu.

The table shows the members in QRPGLSRC or the members in QCBLLSRC.



Now you are ready to use the Remote Systems LPEX Editor to edit the member MSTDSP found in QDDSSRC.

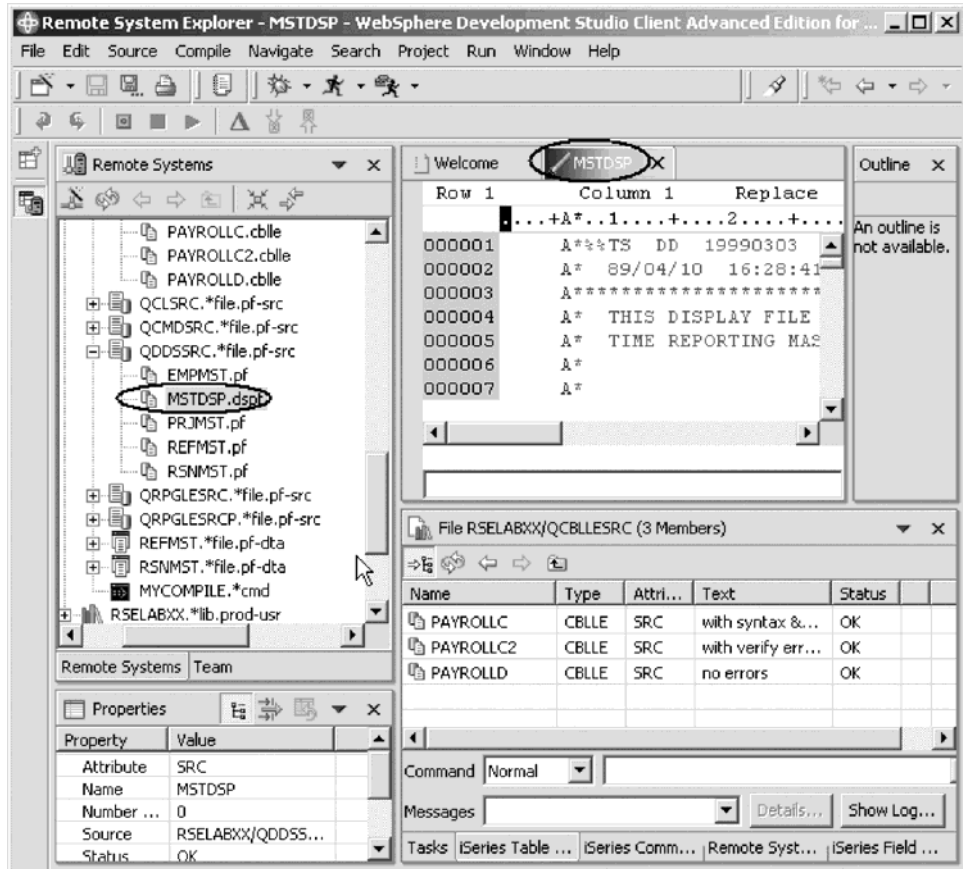
7. From the Remote Systems view double-click member MSTDSP in the QDDSSRC source file.

You can do this in the Remote Systems view or in the iSeries Table view.

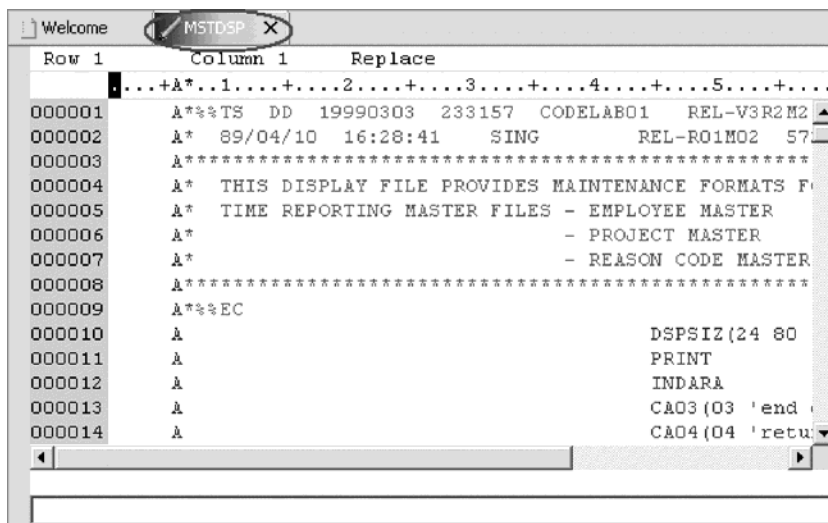
The Remote Systems LPEX Editor opens. It is built right into the workbench, with rich editing functions and is iSeries aware! It is a superset of SEU! The syntax checker is ported down from SEU, the compilers embedded for verifying errors and the reference manuals are built-in and F1 cursor sensitive.

The Outline view will show the program hierarchy. There is explicit and rich iSeries support for verify, compile, run and debug of RPG, COBOL, C, C++, and CL from the Remote Systems LPEX Editor. Many of the editing features offered in the CODE Editor for RPG, COBOL, CL, and DDS source, such as syntax checking, automatic uppercasing, program verification, and so on, are now available in the Remote Systems LPEX Editor.

8. Double-click the **MSTDSP** tab to maximize the Editor window.



9. Double-click the **MSTDSP** tab again to return the window to its original size.





---

## Opening a second source member

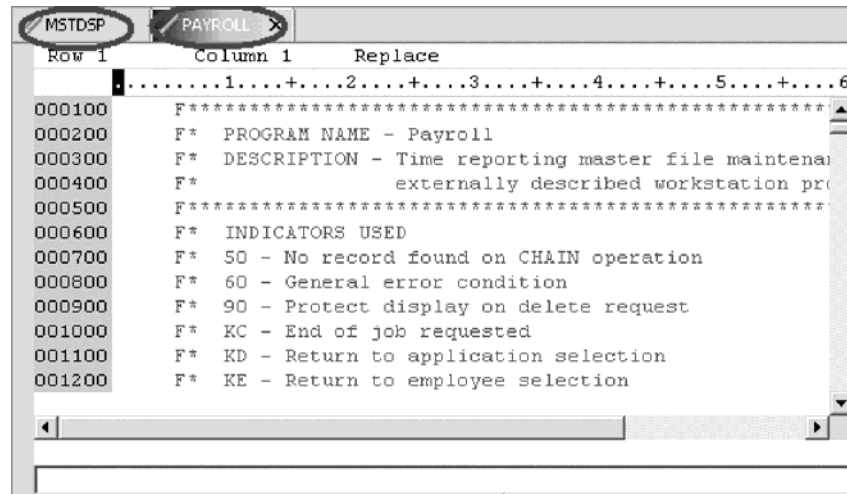
Next let's open a second member in the editor.

To open a second source member:

1. In the Remote Systems view, double-click member PAYROLL in the QRPGLSRC source file to open an RPG source member or PAYROLLC in QCBLLSRC to open a COBOL source member.

This member will be loaded into the editor as well.

Your Editor window will look something like:

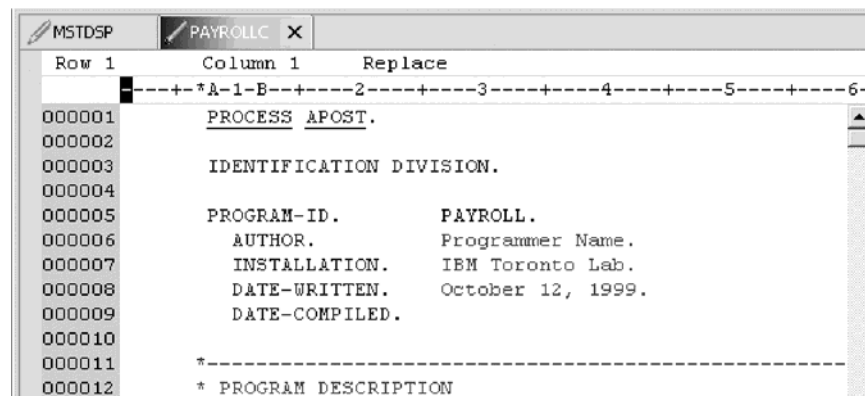


```

MSTDSP  PAYROLL X
Row 1   Column 1   Replace
.....1.....2.....3.....4.....5.....6
000100  F*****
000200  F*  PROGRAM NAME - Payroll
000300  F*  DESCRIPTION - Time reporting master file mainten
000400  F*                      externally described workstation pro
000500  F*****
000600  F*  INDICATORS USED
000700  F*  50 - No record found on CHAIN operation
000800  F*  60 - General error condition
000900  F*  90 - Protect display on delete request
001000  F*  KC - End of job requested
001100  F*  KD - Return to application selection
001200  F*  KE - Return to employee selection

```

or this:



```

MSTDSP  PAYROLLC X
Row 1   Column 1   Replace
---+*A-1-B---+---2---+---3---+---4---+---5---+---6-
000001  PROCESS APOST.
000002
000003  IDENTIFICATION DIVISION.
000004
000005  PROGRAM-ID.          PAYROLL.
000006  AUTHOR.              Programmer Name.
000007  INSTALLATION.       IBM Toronto Lab.
000008  DATE-WRITTEN.      October 12, 1999.
000009  DATE-COMPILED.
000010
000011  *-----
000012  * PROGRAM DESCRIPTION

```

Notice the two tabs in the editor pane.

2. Click on each tab to switch from one edit session to another edit session. Notice the outline view which you will see in the next exercise.

---

## Displaying an outline of a structured file

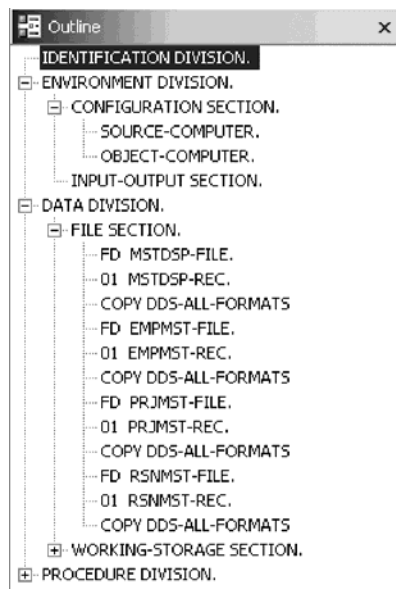
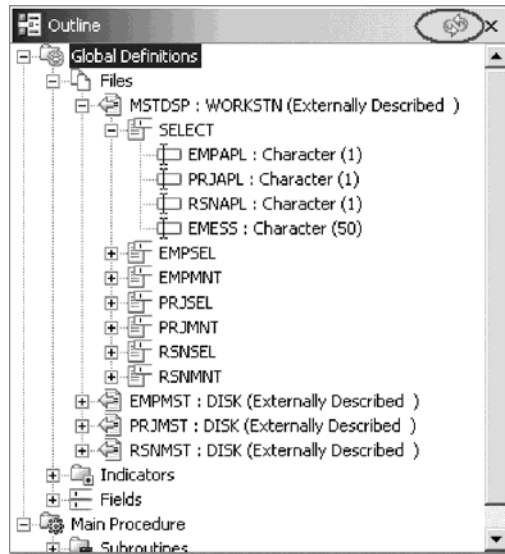
Next let's look at the Outline view. This view displays an outline of a structured file that is currently open in the editor area, and lists structural elements. The contents of the Outline view and toolbar are editor-specific.

To see an Outline view of your source:

1. Click **Refresh** on the Outline view toolbar for RPG source.

When you open COBOL source the outline view will open.

The Outline view contains your source program in a tree view without the lines containing logic.



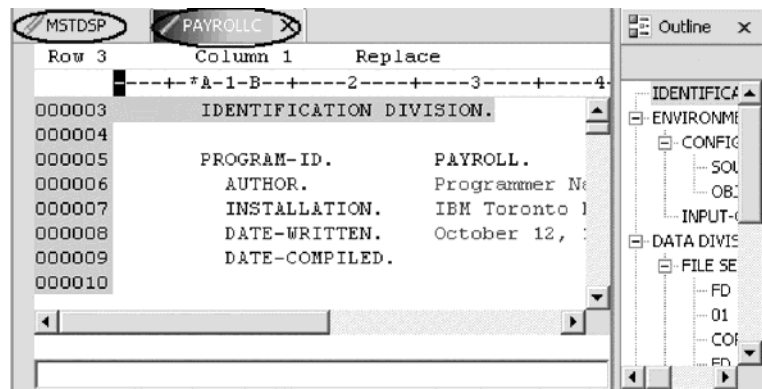
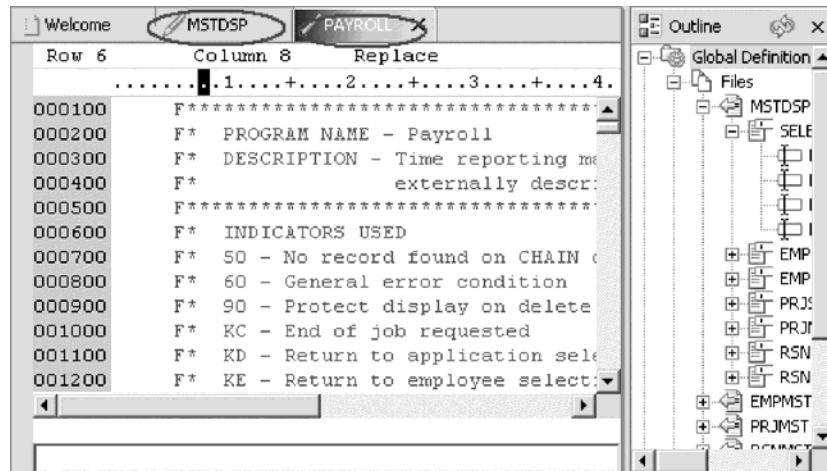
Now you want to see more details of your source member.

2. Expand Files then MSTDSP workstation file and then the SELECT record format for RPG source.
3. Expand ENVIRONMENT DIVISION then CONFIGURATION SECTION, then DATA DIVISION and then the FILE SECTION for COBOL source.
4. Double-click on any of the entries in the Outline view.

This will position the source editor accordingly.

Now if you want to switch to a different member loaded in the source editor just click on its tab and the Source Editor window with the corresponding

member comes into focus.



5. Click the **PAYROLL** tab or **PAYROLLC** tab to give focus to the edit session. Now you want to switch back to the DDS source member.
6. Click the **MSTDSP** tab.
7. Click the **PAYROLL** tab or **PAYROLLC** to get the **PAYROLL** or **PAYROLLC** Editor window in focus for the next module.

## Checkpoint

Complete the checkpoint below to determine if you are ready to move on to the next module.

1. When you first start the workbench and open a perspective, you are not connected to any system except your local workstation. To connect to a remote iSeries system, you need to:
  - a. Start Remote System Explorer communication server
  - b. Start a 5250 session
  - c. Define a connection. Specify the name or IP address of a remote system
  - d. Define a profile
2. The first time you connect to an iSeries system, you need to define a:
  - a. Profile
  - b. Filter
  - c. Filter pool
  - d. Connection

- e. All of the above
- 3. Profiles:
  - a. Help partition data when you have a lot of connections or filter pools
  - b. Include all connections, filters, and filter pools
  - c. Group connections
  - d. Share connections
  - e. Keep connections private
  - f. All of the above
- 4. There are several types of profiles:
  - a. Team
  - b. Personal
  - c. Both
- 5. A team profile is used to share resources and information with other people. (T, F)
- 6. Subsystems include:
  - a. iSeries Objects
  - b. iSeries Jobs
  - c. IFS Files
  - d. iSeries Commands
  - e. All of the above
- 7. iSeries objects include:
  - a. Work with libraries
  - b. Work with objects
  - c. Work with members
  - d. Library list
  - e. All of the above
- 8. The iSeries Table view is used to:
  - a. Display a list of items, for example, members or objects in a table format similar to PDM
  - b. Perform actions against a list of items, such as editing and compiling
  - c. Both
- 9. The lock icon controls the correlation between the Remote Systems view and the iSeries Table view. (T, F)
- 10. A disabled lock means when you click on an object or library in the Remote Systems view
  - a. The associated contents of that item automatically populate the iSeries Table view
  - b. Only the Remote Systems view is updated
  - c. Only the iSeries Table view is updated
  - d. None of the above
- 11. An enabled lock means when you click on an various items in the Remote Systems view
  - a. The associated contents of that item automatically populate the iSeries Table view
  - b. The iSeries Table view input does not change
  - c. Only the Remote Systems view is updated

- d. Only the iSeries Table view is updated
  - e. None of the above
12. You can maximize the Editor window by double clicking on its window heading. You can get back to its original size, by double-clicking on the heading again (T, F).
13. The outline view:
- a. Displays an outline of a structured file currently open in the editor.
  - b. Lists structural elements.
  - c. Contents and toolbar are editor specific
  - d. All of the above

---

## More practice

Want more practice? Try this!

Given that you have access to your own iSeries systems, configure a new connection and connect to this iSeries system. Now rename the connection, move the connection up or change the attributes of the connection. Then use the iSeries objects subsystem to list the libraries in your library list.

Use the iSeries Tables view to see the objects in your library. Subset objects in the iSeries Table view. Open the Remote Systems LPEX Editor from the iSeries Table view.

Use the Development Studio Client for iSeries online help to assist you in these tasks.

---

## Recap

Congratulations! You have completed this module. You should now understand:

- The concepts of profiles, subsystems, iSeries objects
- The concept of the iSeries Tables view
- How to define connection information for an iSeries server
- How to select the iSeries Objects subsystem and view the libraries in the library list
- How to expand the current library and look at the iSeries Table view of QDDSSRC
- How to lock and unlock the iSeries Table view
- How to open several source members in the Remote System LPEX Editor
- How to maximize the Editor window
- How to open an Outline view and switch from the Outline view to a member ready for edit

Now that you have opened a member for edit, you can move on to simplifying your editing tasks with the Remote Systems LPEX Editor. Remember you are editing right from the Remote Systems view! And keep in mind you can use SEU like format line rules, SEU specification prompting and SEU style commands in the Remote Systems LPEX Editor.

Continue with the next module.



---

## Chapter 6. Editing Source

In this module, you will edit ILE RPG source member PAYROLL or ILE COBOL source member PAYROLLC and learn about some of the Remote Systems LPEX Editor's language features.

In order to edit source, there are several steps you will need to learn about and follow:

- Describe the features of the Remote Systems LPEX Editor
- Edit columns in RPG source
- Invoke SEU commands in the prefix area
- Undo and redo edit changes
- Invoke RPG programming language specific help
- Prompt for language specific information
- Indent RPG source
- Find and replace
- Filter source
- Find a string in multiple files with the search utility
- Compare two files with the compare utility
- Identify lines with syntax errors
- Correct syntax errors

In order to accomplish these learning objectives there are several steps involved, including:

- Introducing the Editor
- Editing columns in RPG source
- Entering SEU commands
- Requesting Undo and Redo operations
- Invoking Language-Sensitive help
- Displaying COBOL line types
- Prompting RPG Language Specifications
- Indenting RPG source
- Finding and replacing text
- Filtering lines by string
- Searching multiple files
- Comparing file differences from the Remote Systems view
- Optional. Comparing file changes from the CODE editor
- Checking syntax

The exercises within this module must be completed in order. Start with the first exercise when you are ready to begin. You might want to maximize the Editor window during this module.

**Length of time:**

This module will take approximately 20 minutes to complete.

---

## Introducing the editor

Your program editing tasks are simplified with the Remote Systems LPEX Editor. The editor can access source files on your workstation or your iSeries system directly. When a compilation results in errors, you can jump from the compiler messages to an editor containing the source. The editor opens with the cursor positioned at the offending source statements so that you can correct them.

Here is a list of the some of the basic editor features that you would expect in a workstation editor:

- Cut, copy, and paste
- Block marking of lines, characters, or rectangles with copy, move, and delete operations
- Powerful find and replace function
- Unlimited undo and redo

In addition there are a few more functions that you may not have seen in a workstation editor:

- Token highlighting where different language constructs are highlighted using different colors and fonts to help identify them in a program
- SEU-like format-line rulers to show the purpose of each column for column-sensitive languages like RPG and DDS. These rulers can automatically update themselves to reflect the current specification.
- SEU-like specification prompting for RPG and DDS
- Sequence numbers, which allow SEU-style commands in the prefix area
- Intelligent tabbing between columns for column-sensitive languages
- Automatic uppercasing for languages that expect uppercase
- Settings for column-sensitive languages simplify text insertions and deletions
- On-line language reference

LPEX provides special support for insertion and deletion in column-sensitive languages. When column sensitive editing is selected, each column is considered as a separate entry space. For example, in an RPG source member, if you are inserting or deleting characters into a string that is in the Factor 2 entry, the Result field entry does not move. The default editor preference is that column sensitive editing is off. You can switch this support on by going to the workbench preferences window. The LPEX Editor has predefined settings, but also has an associated preference page containing settings that you can define. The name of the category is LPEX Editor and it appears in the tree view of the Preferences window.

Now let's try some of these features.

---

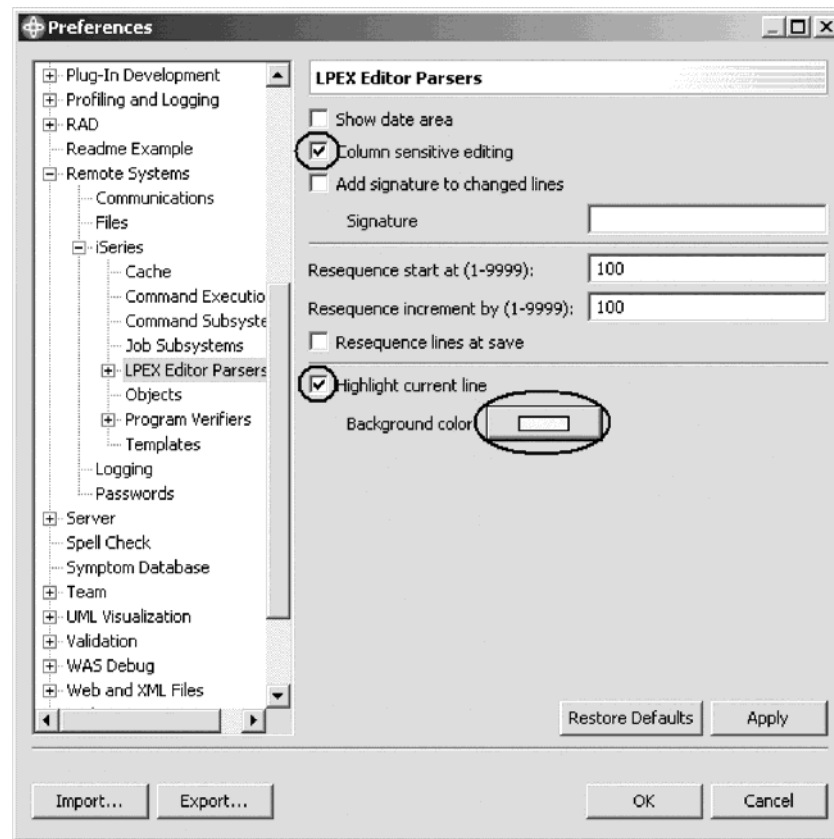
## Editing columns in RPG source

To edit columns:

1. Click **Window > Preferences** from the workbench menu.



The Preferences window opens.



2. In the left pane of the Preferences window, expand Remote Systems.
3. Expand iSeries under Remote Systems.
4. Click LPEX Editor Parsers under iSeries.

The right pane allows you to set preferences for this feature.

5. In the right pane of the Preferences window, select the **Column sensitive editing** check box.

When selected, each column is considered as a separate entry space.

6. In the right pane of the Preferences window, select the **Highlight current line** check box.

This option highlights the line that the cursor is currently on. The option applies to all source files opened in the editor area.

7. In the right pane of the Preferences window, click **Background color**.

The Color palette opens.

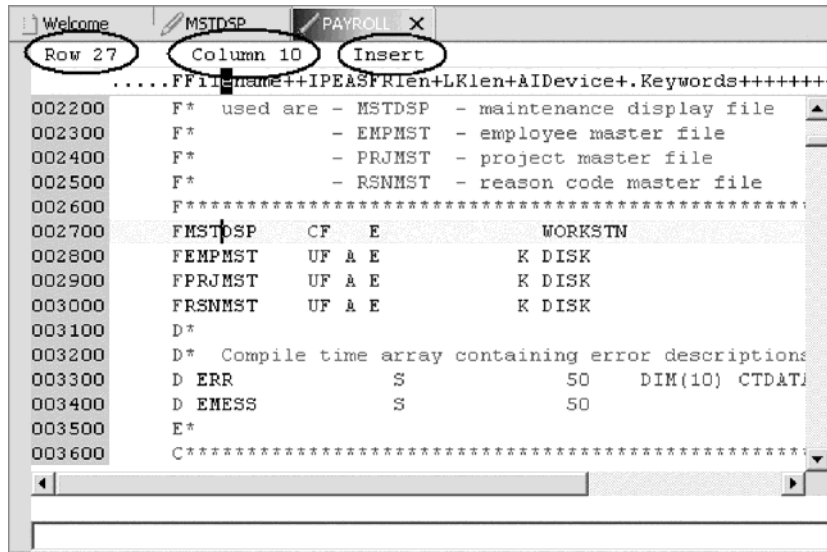
8. Select **light yellow** from the Color palette.
9. Click **OK**.

The Preferences window shows.

Other interesting preference settings are located under LPEX Editor Parsers. When you expand this entry you see preference settings for the individual language environments.

10. Click **OK** again on the Preferences window.

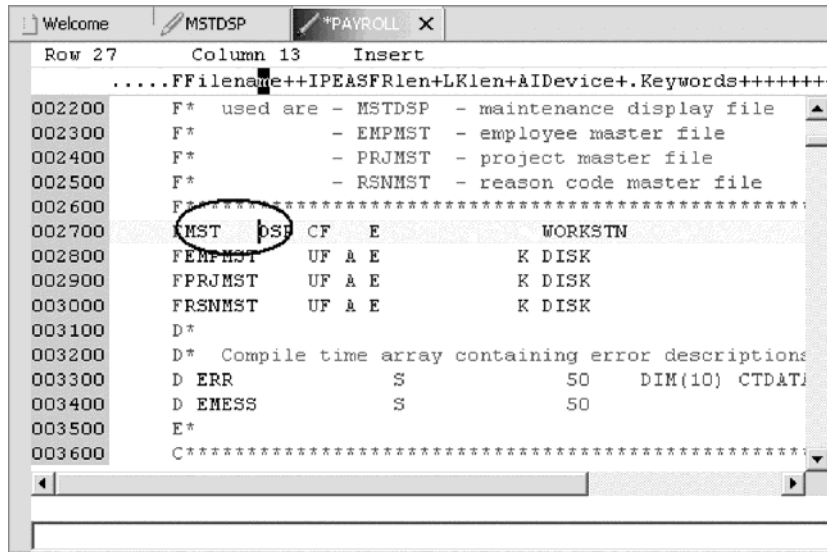
Now let's see what this does.



Return to the Editor window.

11. Move the cursor to line 27, column 10.
12. Make sure the editor is in **Insert** mode. If the status area shows **Replace** press the **Insert** key.
13. Press the spacebar 3 times.

Notice that only the filename is shifted but none of the other columns to the right are effected.



14. Press the **Backspace** 3 times.  
Once again the filename is shifted but no other columns are effected.

---

## Entering SEU commands

You can configure the LPEX Editor to adopt the keyboard and command personalities of many popular editors. Most editor profiles differ only in the keys and commands used to perform various editor tasks. Some base editor profiles, listed below, also add prefix information and command area at the start of each line:

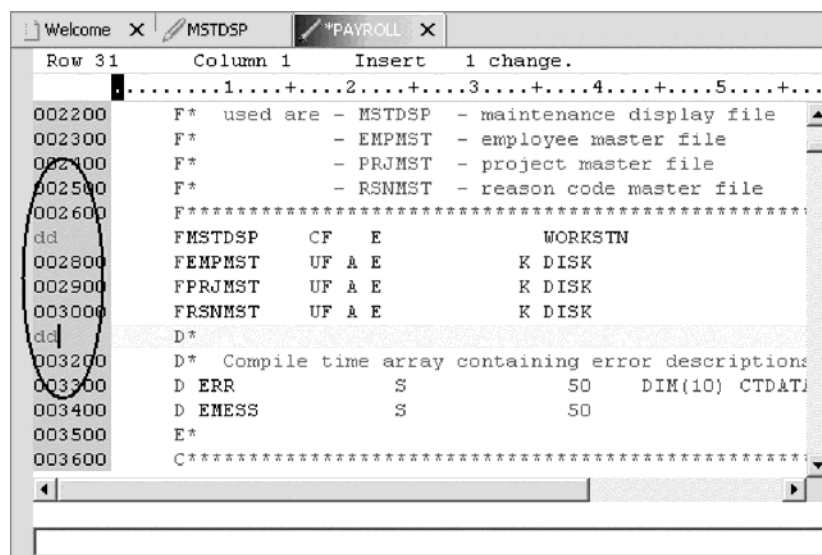
- ispf
- seu
- xedit

The editor recognizes prefix commands used by these editor profiles. Depending on which profile you are using, you can enter SEU, XEDIT, or ISPF commands when the prefix area is active.

If you are an SEU expert you will appreciate the ability to use SEU commands.

To enter SEU commands:

1. Move the cursor into the gray sequence number area to the left of the edit area for PAYROLL or PAYROLLC.



2. On any sequence number type dd.
3. Go down a few lines and type dd again and press **Enter**.  
Notice that the lines have been deleted.
4. Now type i5 in the sequence number area.
5. Make sure the cursor is within the sequence number area.
6. Press **Enter**.  
Five new lines are inserted.

---

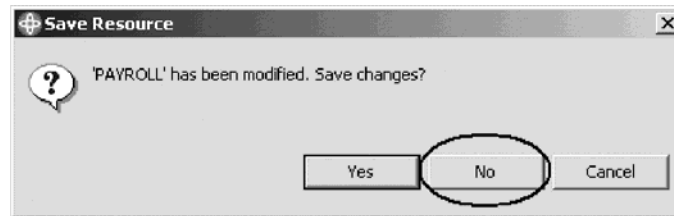
## Requesting undo and redo operations

The editor records each set of changes you make to a file in the Editor window. The number of changes made since the last file save is displayed on the status line. If you want to undo a set of changes made to a file you use the Undo operation. You can also cancel the effects of an Undo operation by using the Redo operation.

Now you are going to undo some of the changes you just made to the file. Then you will cancel the changes by using the Redo operation. Finally you will reload the source so that it is back to its original form.

To undo and redo edit changes:

1. Select **Edit > Undo** from the workbench menu.  
Notice that the 5 new lines disappear.
2. Press **Ctrl+Z** to undo the first change.  
Notice that the deleted lines reappear.
3. Select **Edit > Redo** from the workbench menu.  
Notice that the lines are deleted again.  
At this point you will reload the source from the iSeries to make sure that it is back in its original form.
4. Select **File > Close** option on the workbench menu.  
A Save Resource dialog opens asking if you want to save the latest changes.



5. Click **No**.
6. Go back to the workbench to the Remote System Explorer perspective and open the PAYROLL member in the QRPGLESRC file or the PAYROLLC member in the QCBLLSRC file.

---

## Invoking language-sensitive help

Inside the editor, there is cursor-sensitive language-reference help available. This help is invaluable if you cannot remember the order of fields in an RPG specification or the possible values for a variable field. This help is available from the LPEX Editor window.

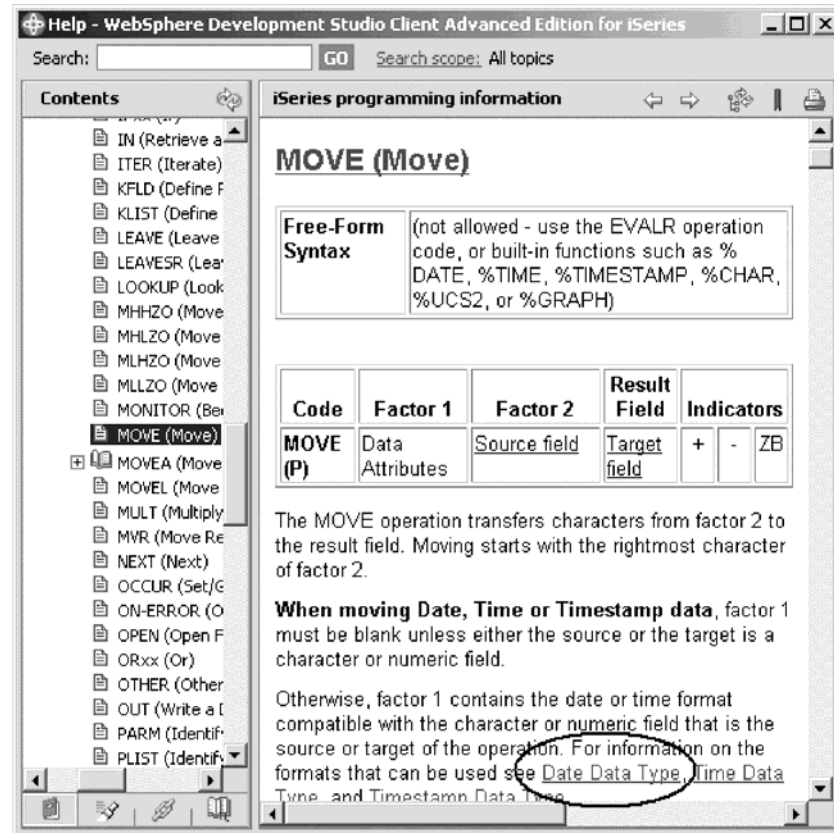
To receive language sensitive help, press F1 in an Edit window. If the cursor is on an operation code, you receive help for that operation code; otherwise, you receive help for the current specification.

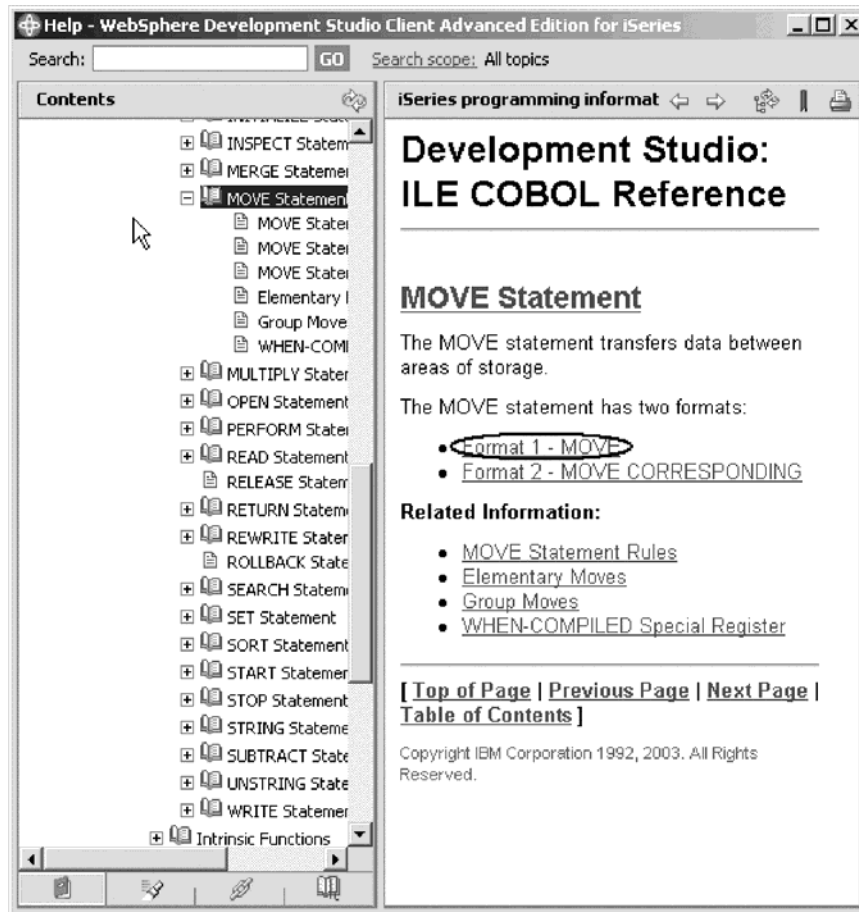
To access language sensitive help:

1. Position the cursor over the word MOVE in line 112 of the ILE RPG source or 231 in the ILE COBOL source.
2. Press **F1**.

Language-sensitive help for the MOVE operation code appears in a Help window.

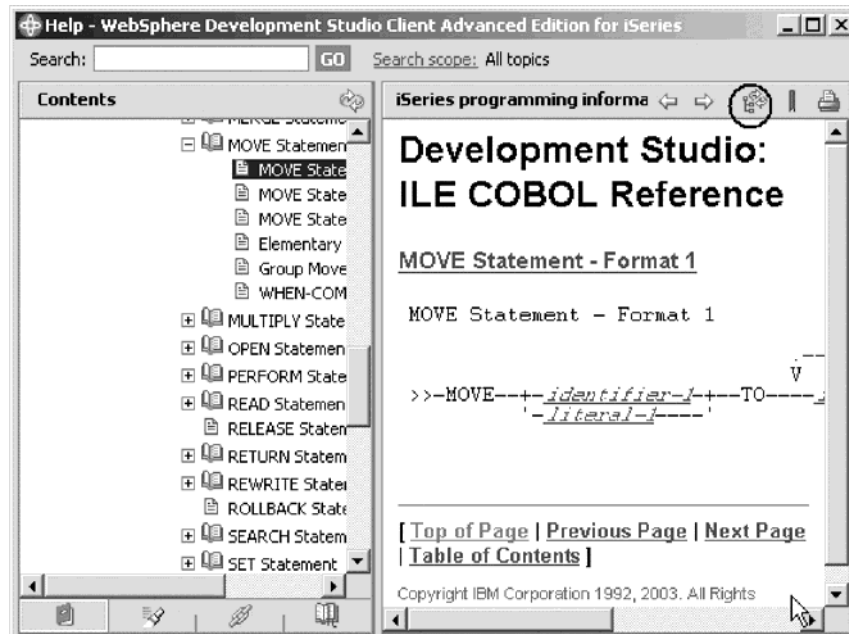
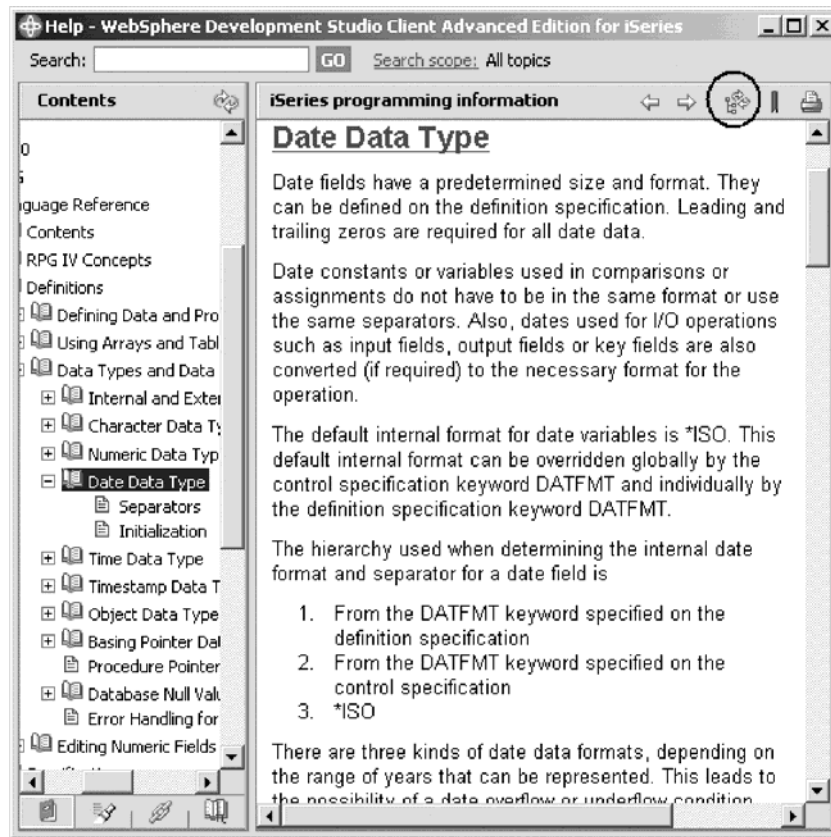
Text marked in blue in the Help window contains the link to detailed information about the topic in blue.





3. Click the link Date Data Type for the RPG help or Format 1–MOVE for COBOL help.

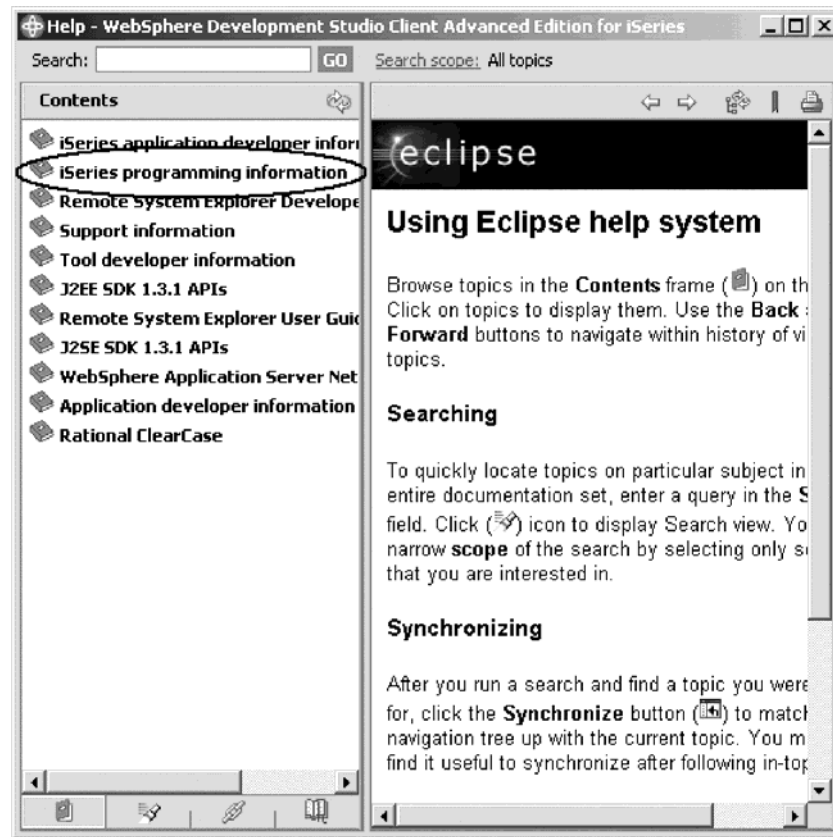
4. Click the **Show in Table of Contents** icon on the Help window toolbar.



This will synchronize the topics in the Contents pane on the left to the help topic you are viewing in the main help pane.

5. Play around in the Help window to see what else is available.
6. Minimize the Help window.
7. Select **Help > Help Contents** on the workbench menu to see a list of all help that is available in the product.

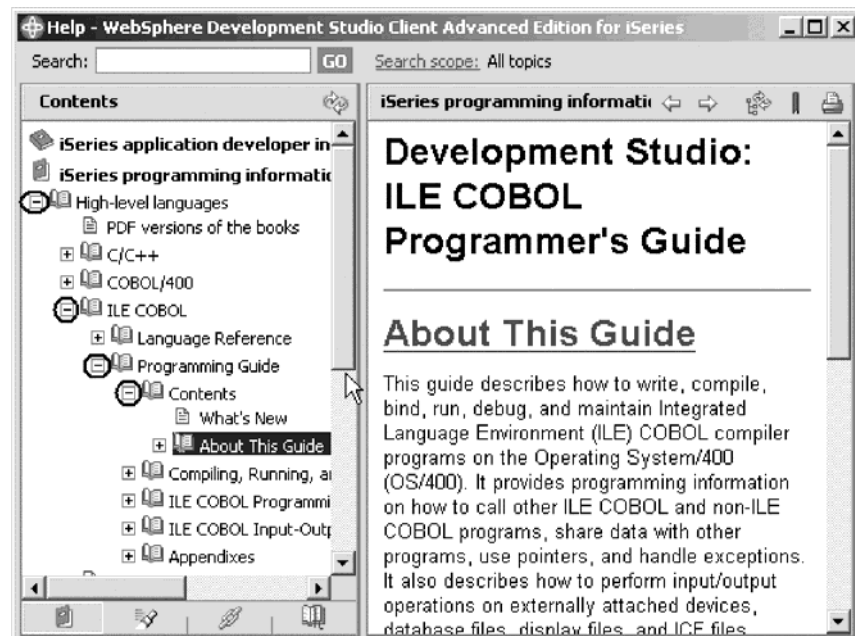
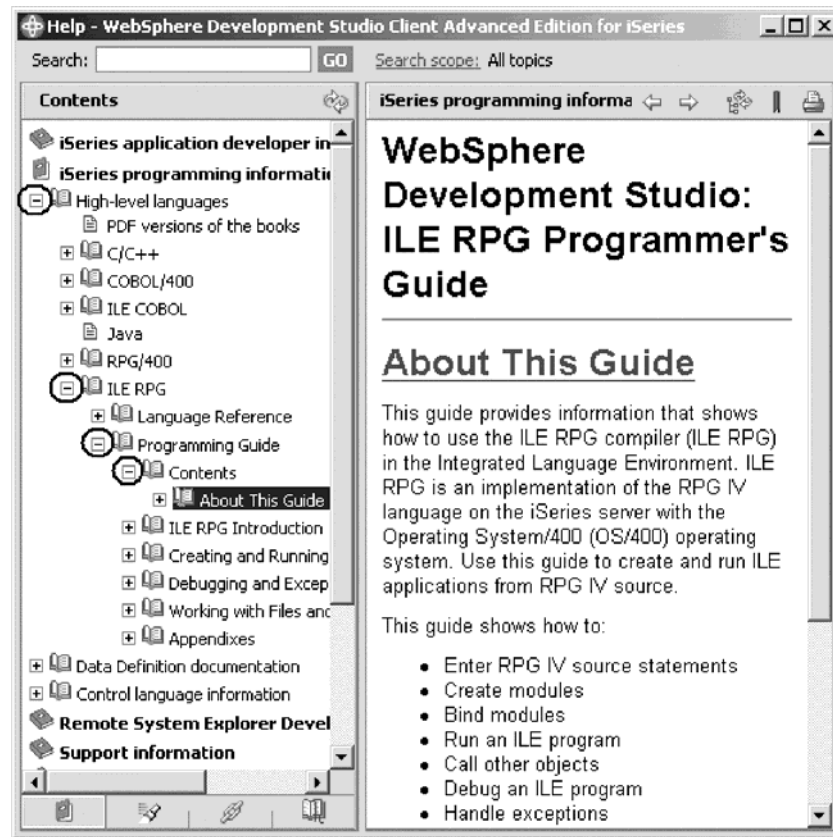
Here you will find the table of contents and you can select a topic of interest.



8. In the left pane of the Help window, click iSeries programming information.
9. Expand High level languages.
10. Expand ILE RPG or ILE COBOL.
11. Expand Programming Guide.
12. Expand Contents.



13. Click About This Guide.



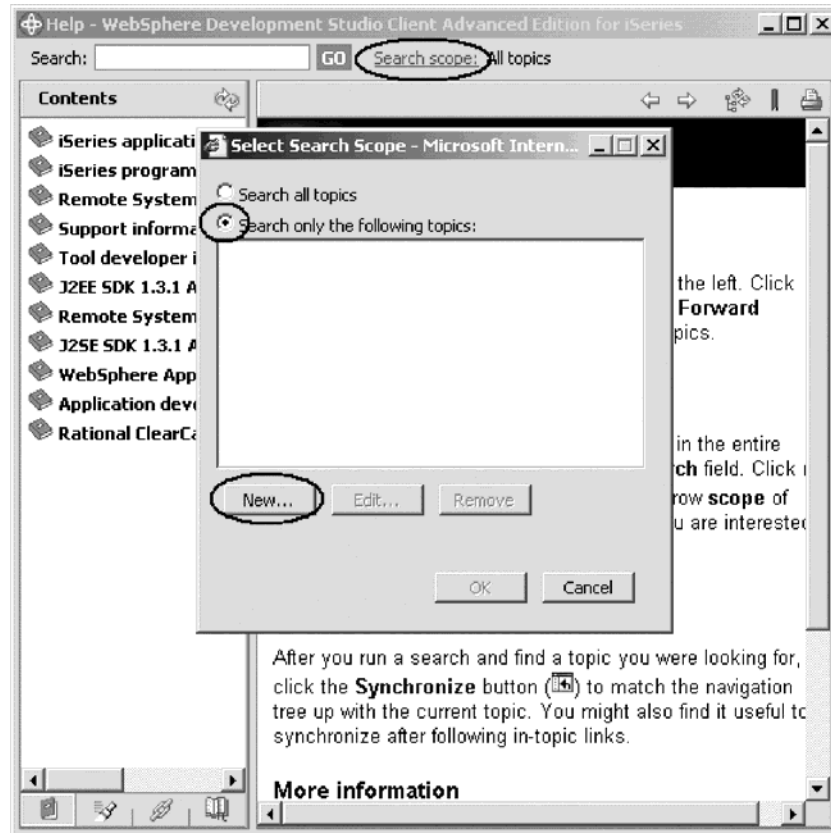
Having the latest version of the manuals at your fingertips will make it easier to find programming information. There is also the option to search the help by specifying a search string. By default, the complete help will be searched.

To limit the search to specific documents:

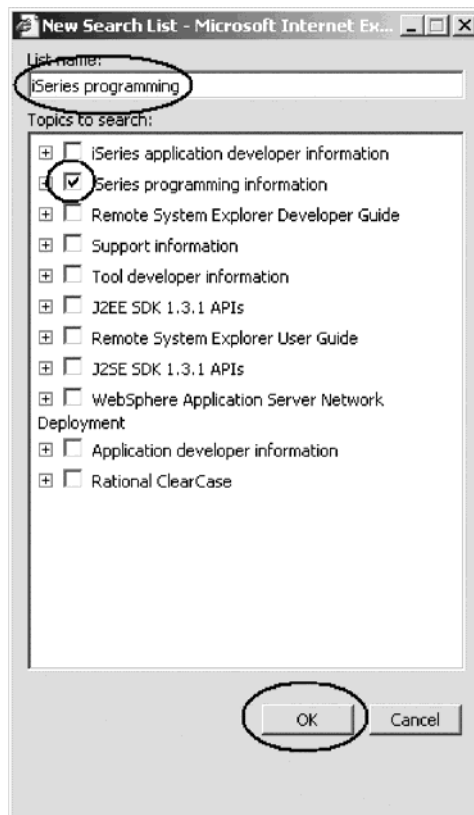
1. Click **Search scope**.

The Select Search Scope dialog opens.

2. Click the **Search only the following topics** radio button.
3. Click **New**.



The New Search List dialog opens:



4. In the **List name** field, type iSeries programming for example.
5. Select the **iSeries programming information** check box.
6. Click **OK**.
7. Click **OK** in the Select Search Scope dialog.

Now, if you wanted to do another search, you would only search in the iSeries programming information topics.

8. Minimize the Help window.

---

## Prompting RPG language specifications

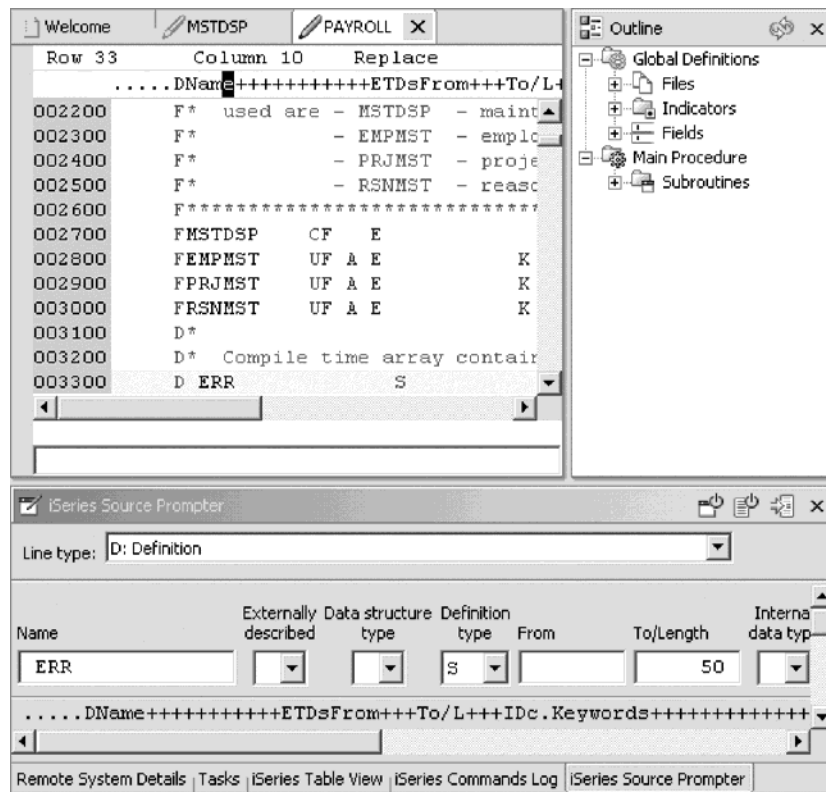
Instead of entering or changing code directly in the Editor window, you can use prompts. When you request a prompt for a specification line, a window appears where you can enter or change that line using entry fields.

To access prompts:

1. Return to the workbench. In the Editor window move your cursor to the D-spec on line 33.
2. Click **Source** from the workbench menu.
3. Click **Prompt** (or press **F4**).

The Editor window will reduce in size and allow you to see the iSeries Source Prompter at the bottom of the workbench. The iSeries Source Prompter shows

the specification line broken down into its individual fields.



On the iSeries Source Prompter toolbar you can use the three push buttons to: Disable source prompt view, Disable syntax checking, and Change to insert mode.

To display context sensitive help for any field in the iSeries Source Prompter:

1. Tab to the **Keywords** field.
2. Press **F1** to see help for this field.

The Help window with help for the D spec keywords opens. If it doesn't appear automatically, you might have to bring it to the foreground by clicking on its icon on the Windows task bar.

You will see words in the help that appear in a different color than the regular text. These are help links, and they show that there is additional help available on that word or phrase.

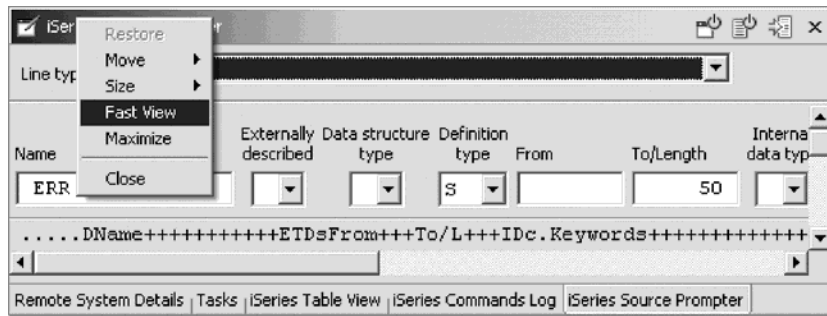
3. Click on any link to see specific help for that item.
4. Minimize the Help window.

To get to the iSeries Source Prompter from the maximized Editor window without bringing the size of the Editor window down, position the iSeries Source Prompter on the left task bar as a Fast view.

To create a fast view:

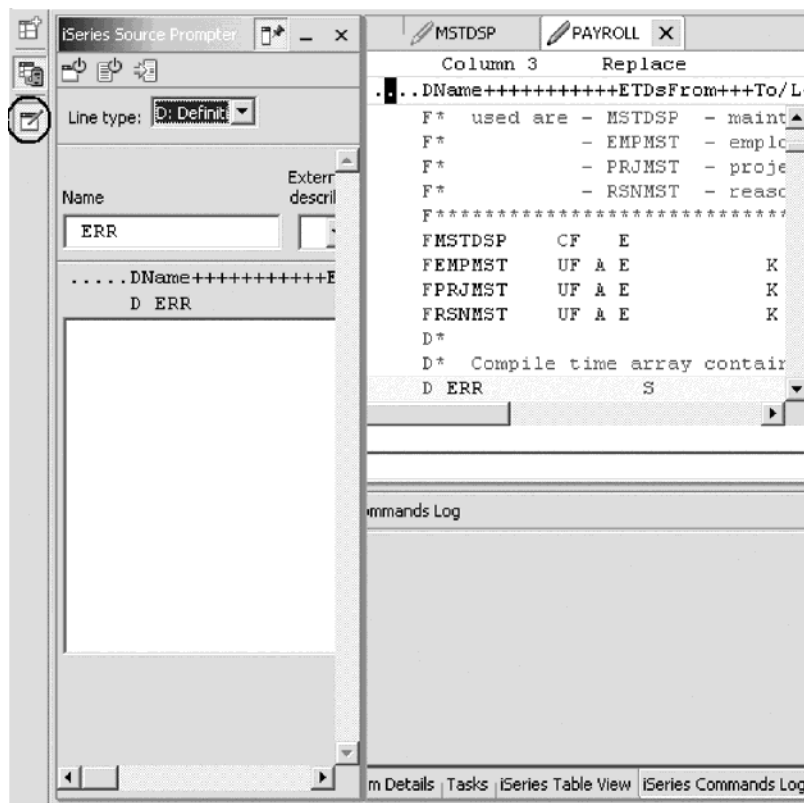
1. Right-click the iSeries Source Prompter title bar.

2. Click **Fast view** on the pop-up menu.



You will see the iSeries Source Prompter icon on the task bar on the left hand side of the workbench

3. Double-click the **PAYROLL** tab to maximize the Editor window.
4. Select a line in the Editor window and press **F4** to prompt for a line.  
The iSeries Source Prompter will be placed on top of the Editor window.



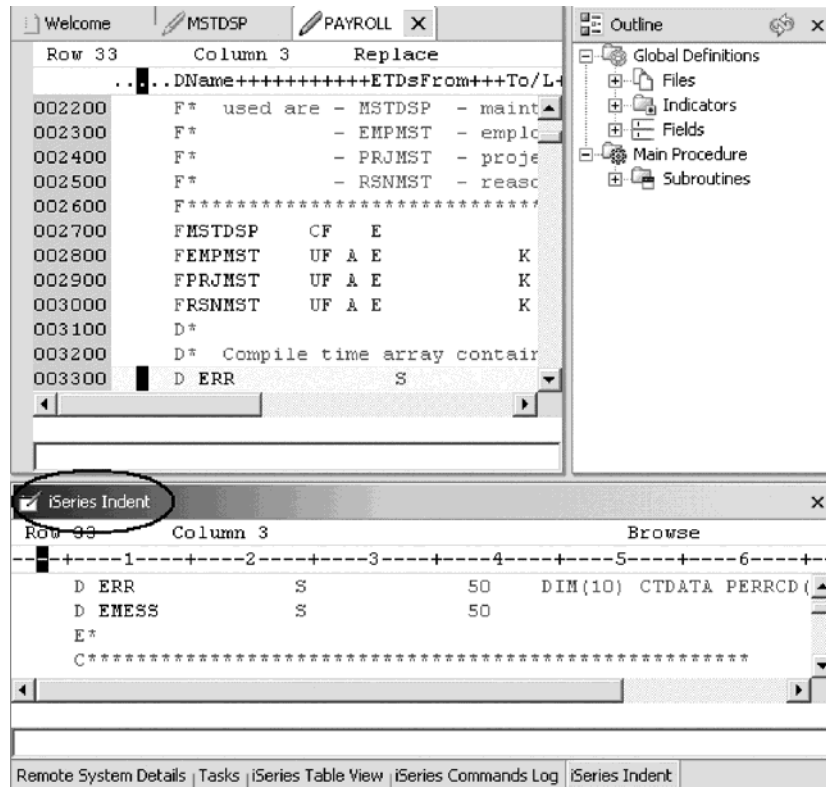
5. Click the **iSeries Source Prompter** icon on the task bar to minimize the iSeries Source Prompter.
6. Double-click the **PAYROLL** tab to return the Editor window to its original size.

## Indenting RPG source

When editing ILE RPG source, it can be difficult to determine the beginning and ending of constructs. The indent option allows you to view your source with constructs in an indented mode. By default, the indent option will split the screen horizontally and show the indented view in the bottom pane.

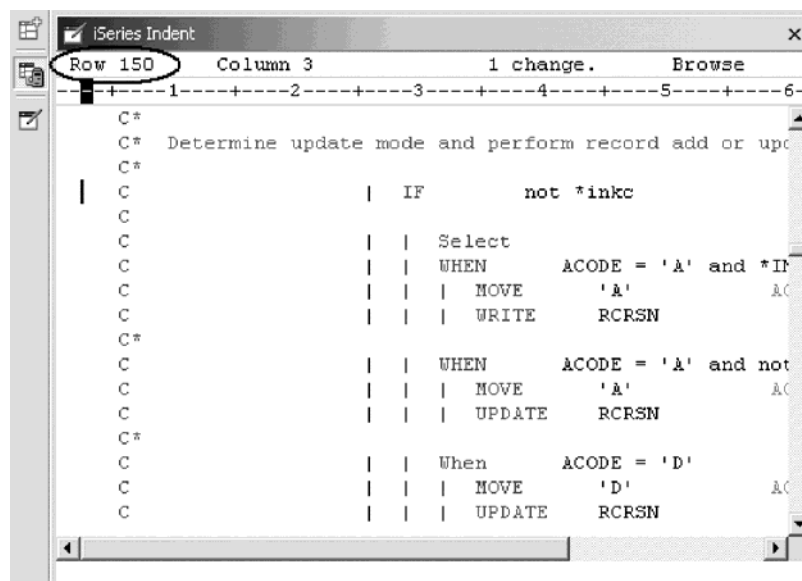
To indent source:

1. Click **Source > Show Indentation** from the workbench menu.  
The iSeries Indent view opens at the bottom of the workbench.



You can display the iSeries Indent view as a full view.

2. Double-click the iSeries Indent view title bar.



3. Scroll down to line 150.  
In this area you see some nested conditions with indented lines. As you will notice this helps to recognize the beginning and ending of these conditions.  
You can move this view as a Fast view to the left task bar as well.

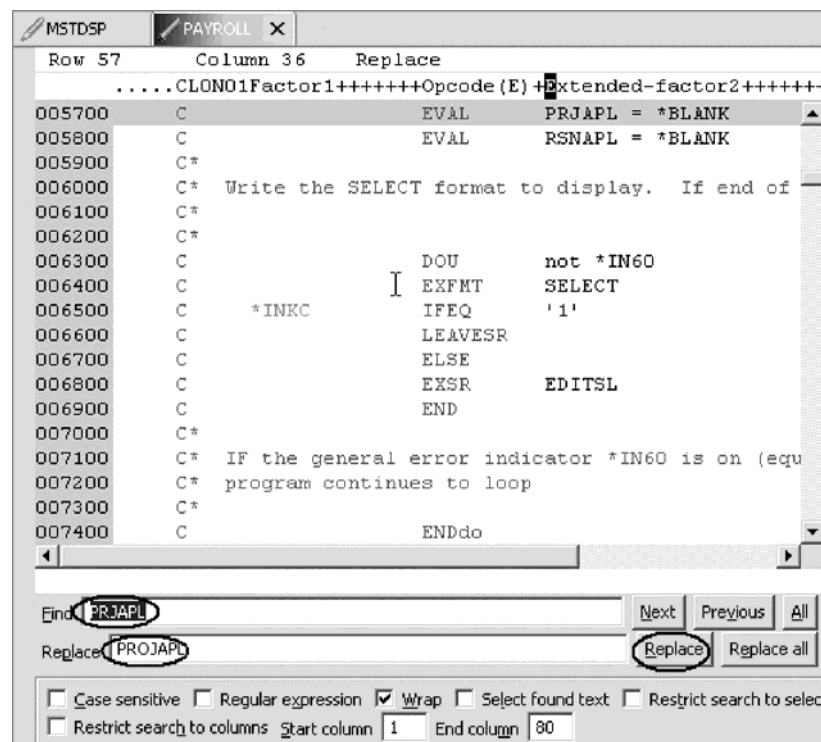
- Note:** The Indented view is Browse mode only and cannot be edited.
- Click the X in the top right corner of the view to close the iSeries Indent view.
  - This returns you to the Editor window with the PAYROLL program.

## Finding and replacing text

The LPEX Editor also has a powerful find and replace text feature. You use the Find and Replace window to search for an item. You can search for a word, a partial word, or a sequence of such. You can also enter a pattern you want to match, provided that the pattern follows the rules of regular expression. You can replace the found search item. If the entered text or pattern is found, the cursor moves to either the next or previous occurrence of the search item, according to your chosen search direction, and replaces the found text according to your selections.

To find and replace text:

- Press **Ctrl+Home** to go to the top of the PAYROLL file or the PAYROLLC file.  
Tip: When you press **Ctrl+Home** to get to the top or **Ctrl+End** to get to the bottom, a quick mark is set at your cursor position. This allows you to get back to that line by pressing **Alt+Q**. **Ctrl+Q** will set a quick mark.
- Double-click the **PAYROLL** tab or **PAYROLLC** tab to maximize the Editor window.
- Click **Edit > Find/Replace** from the workbench menu or press **Ctrl+F**.  
The Find/Replace window opens at the bottom of the Editor window.



At the bottom of this window, you will notice that you have some options to select from, for example, search only in certain columns, etc. You want to find the first occurrence of PRJAPL and replace it by the word PROJAPL.

- In the **Find** field, type PRJAPL.
- Press the tab key to move to the **Replace** field and type PROJAPL.

The Editor moves the active line to line 57 in the RPG source or 194 in the COBOL source, which contains the first PRJAPL in the file.

6. Click **Replace** to replace PRJAPL with PROJAPL.
7. Click **Next** to go to the next location of PRJAPL in the file.
8. Click **Replace all** to replace all occurrences of PRJAPL with PROJAPL.
9. Double-click member **PAYROLL** tab or **PAYROLLC** tab in the Editor window to return the PAYROLL or PAYROLLC Editor window to its original size.
10. Close the PAYROLL or PAYROLLC window.  
A window will appear asking you to save the latest changes.
11. Click **No**.
12. From the Remote Systems view, double-click PAYROLL to reload the RPG original source or PAYROLLC to reload the COBOL original source.

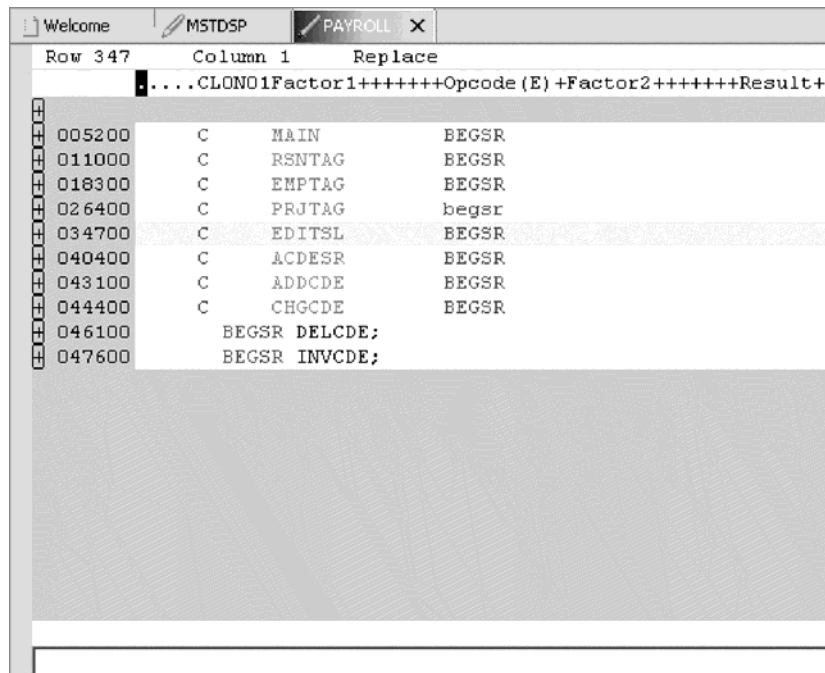
## Filtering lines by string

The editor allows you to filter or subset your source so that you see only lines containing a given string. Filtering lines makes it quick and easy to find lines without having to scroll through your source.

```
011000      C      RSNTAG      BEGSR
000029      * are: MSTDSP - maintenance display file
000030      *      EMPMST - employee master file
```

To filter source by string:

1. Double-click the BEGSR in PAYROLL or MSTDSP in PAYROLLC in the Edit window to select it.
2. Select **Edit > Selected > Filter Selection** from the workbench menu.





```

MSTDSP PAYROLL X
Row 29 Column 14 Replace
-----+*A-1-B-+-----2-----3-----4-----5-----+
000029 * are: MSTDSP - maintenance display file
000037 SELECT MSTDSP-FILE
000038 ASSIGN TO WORKSTATION-MSTDSP-S
000068 FD MSTDSP-FILE.
000069 01 MSTDSP-REC.
000070 COPY DDS-ALL-FORMATS OF MSTDSP.
000132 01 MSTDSP-IND-USAGE.
000154 OPEN I-O MSTDSP-FILE
000160 INITIALIZE MSTDSP-REC.
000167 CLOSE MSTDSP-FILE
000179 WRITE MSTDSP-REC FORMAT IS 'SELECT'
000183 READ MSTDSP-FILE FORMAT IS 'SELECT'
000220 INITIALIZE MSTDSP-REC
000294 WRITE MSTDSP-REC FORMAT IS 'EMPSEL'
000298 READ MSTDSP-FILE FORMAT IS 'EMPSEL'
000343 WRITE MSTDSP-REC FORMAT IS 'EMPMT'
000347 READ MSTDSP-FILE FORMAT IS 'EMPMT'
000394 WRITE MSTDSP-REC FORMAT IS 'PRJSEL'
000398 READ MSTDSP-FILE FORMAT IS 'PRJSEL'
000443 WRITE MSTDSP-REC FORMAT IS 'PRJMT'
000447 READ MSTDSP-FILE FORMAT IS 'PRJMT'
000494 WRITE MSTDSP-REC FORMAT IS 'RSNSEL'

```

3. Move the cursor down a few lines to line 347.
4. Click the plus sign beside line 347 to expand this section.

```

Welcome MSTDSP PAYROLL X
Row 347 Column 1 Replace
....CLONO1Factor1+++++Opcode(E)+Factor2+++++Result++
005200 C MAIN BEGSR
011000 C RSNTAG BEGSR
018300 C EMPTAG BEGSR
026400 C PRJTAG begsr
034700 C EDITSL BEGSR
034800 C*
034900 C* Housekeeping, clear display fields and reset
035000 C*
035100 C MOVE *BLANKS EMESS
035200 C EVAL *IN60 = *OFF
035300 C*
035400 C* The following IF AND OR combination checks the
035500 C* selection fields to ensure that only one appli
035600 C* selected.
035700 C*
035800 C EMPAPL IFEQ 'X'
035900 C PRJAPL ANDEQ 'X'
036000 C EMPAPL OREQ 'X'
036100 C RSNAPL ANDEQ 'X'
036200 C* THE BUG IS HERE

```

```

MSTDSP PAYROLLC X
Row 347 Column 1 Replace
---+*A-1-B-+---2---+---3---+---4---+---5---+
000160 INITIALIZE MSTDSP-REC.
000167 CLOSE MSTDSP-FILE
000179 WRITE MSTDSP-REC FORMAT IS 'SELECT'
000183 READ MSTDSP-FILE FORMAT IS 'SELECT'
000220 INITIALIZE MSTDSP-REC
000294 WRITE MSTDSP-REC FORMAT IS 'EMPSEL'
000298 READ MSTDSP-FILE FORMAT IS 'EMPSEL'
000343 WRITE MSTDSP-REC FORMAT IS 'EMPMT'
000347 READ MSTDSP-FILE FORMAT IS 'EMPMT'
000348 INDICATORS ARE IND-TABLE
000349 END-READ.
000350
000351 INITIALIZE RCEMP.
000352 MOVE CORR EMPMT-I TO RCEMP.
000353 MOVE WS-EMPNO TO EMPNO OF RCEMP.
000354 MOVE WS-ACREC TO ACREC OF RCEMP.
000355
000356 IF IND-OFF(IND-MAINT) AND IND-OFF(IND-EOJ)
000357 AND IND-OFF(IND-EMPS)
000358 EVALUATE WS-ACODE
000359 WHEN 'A'
000360 MOVE 'A' TO ACREC OF RCEMP
000361 IF WS-ACREC = 'D'

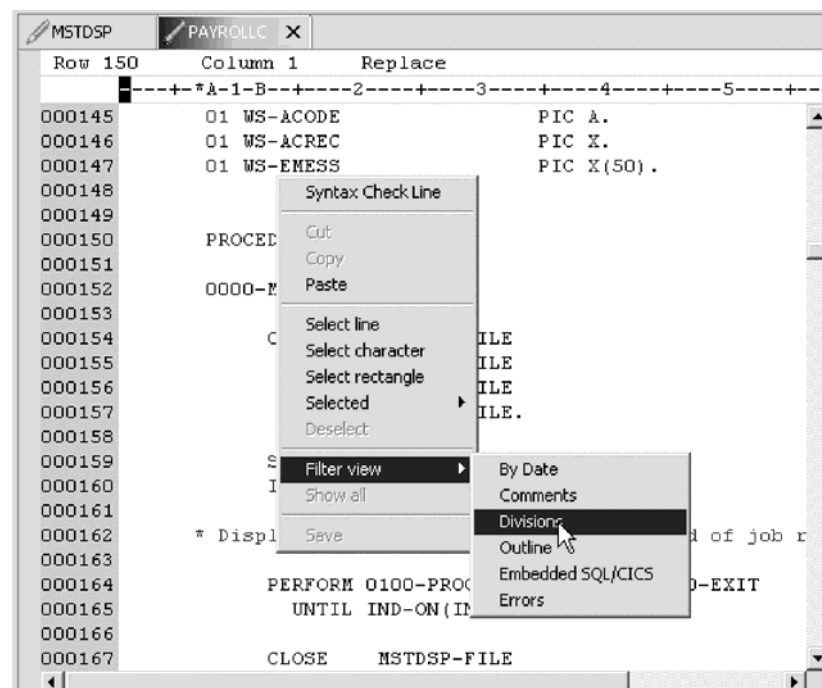
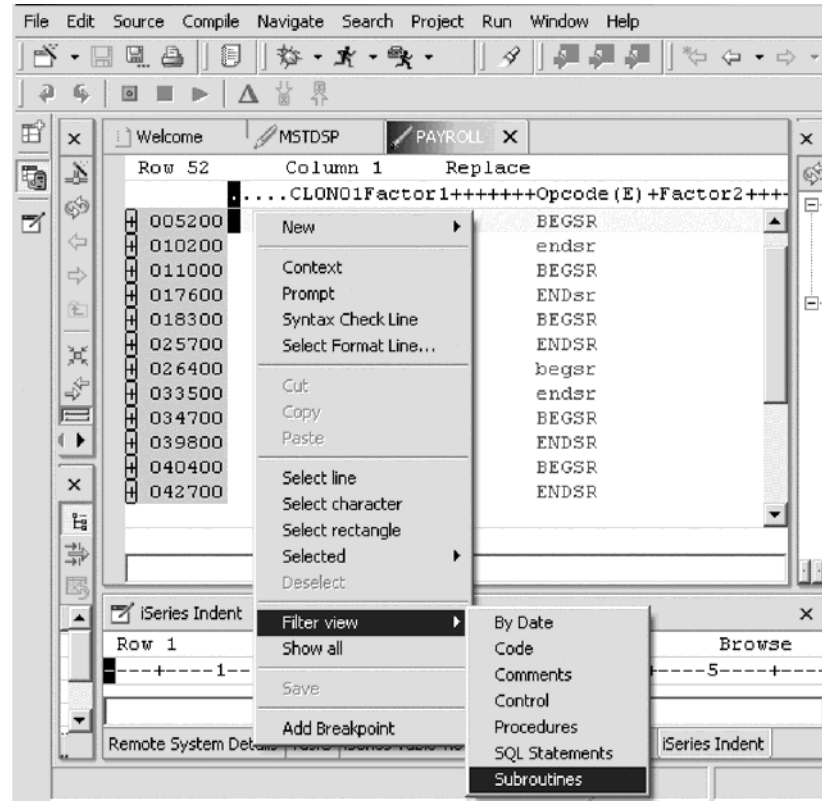
```

Now you want to show the entire source again.

5. Click **Edit > Show all** from the workbench menu or press **Ctrl+W**.  
Your cursor is still positioned on the same line that you moved the cursor to, even though all lines are now showing.

## Filtering lines by type

To help you navigate quickly through your ILE RPG or ILE COBOL source the Editor allows you to filter lines based on the line type. Imagine you want to see where all the subroutines are defined in your RPG source or where all the divisions are defined in your COBOL source.



To filter lines by type:

1. Right-click in the Editor window with the PAYROLL program or PAYROLLC program.
2. Click **Filter view > Subroutines** on the pop-up menu for PAYROLL or click **Filter view > Divisions** on the pop-up menu for PAYROLLC.

For PAYROLL all subroutines specifications are displayed allowing you to move quickly and easily to the area in your file where the desired subroutine is.

For PAYROLLC all divisions are displayed allowing you to move quickly and easily to the area in your file where the desired division is.

The screenshot shows the PAYROLL editor window with the following content:

```

Welcome MSTDSP PAYROLL X
Row 176 Column 39 Replace
.....CLON01Factor1+++++Opcode (E) +Factor2+++++
005200 C MAIN BEGSR
010200 C endsr
011000 C RSNTAG BEGSR
017600 C ENDSr
018300 C EMPTAG BEGSR
025700 C ENDSR
026400 C PRJTAG begsr
033500 C endsr
034700 C EDITSL BEGSR
039800 C ENDSR
040400 C ACDESR BEGSR
042700 C ENDSR
043100 C ADDCDE BEGSR
044000 C ENDSR
044400 C CHGCDE BEGSR
045400 C ENDSR
046100 BEGSR DELCDE;
047200 ENDSR;
047600 BEGSR INVCDE;
047900 ENDSR;

```

The screenshot shows the PAYROLLC editor window with the following content:

```

MSTDSP PAYROLLC X
Row 150 Column 1 Replace
-----+*A-1-B-----2-----3-----4-----5-----+
000003 IDENTIFICATION DIVISION.
000005 PROGRAM-ID. PAYROLL.
000018 ENVIRONMENT DIVISION.
000064 DATA DIVISION.
000150 PROCEDURE DIVISION.

```

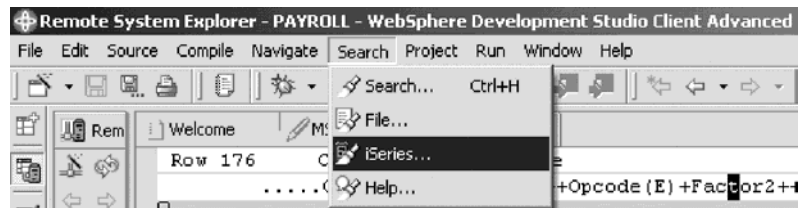
3. Move your cursor to the line with the subroutine declaration CHGCDE (line 444) in PAYROLL or the procedure division (line 150) in PAYROLLC.
4. Expand the declaration to show all lines in this subroutine or division.  
Now you could work with the source inside this subroutine or division.
5. Right-click in the Editor window and click **Show All** on the pop-up menu.

## Searching multiple files

If you would like to search through the members in a source physical file or through the files in a local directory, you can use the Search tool. The Multi-File Search utility allows you to search for a particular string of text in many members on the host. This function can also be used on local files.

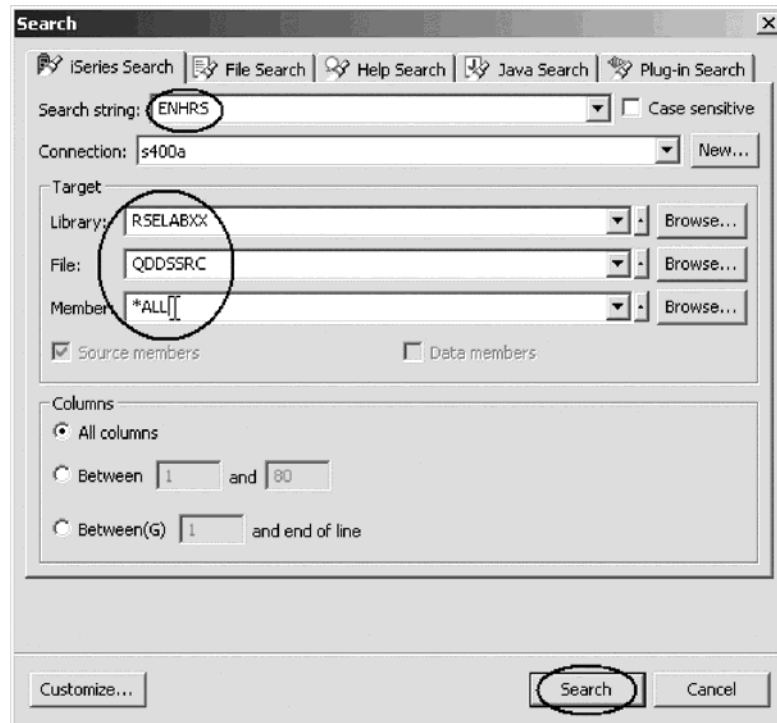
To search multiple files:

1. Click **Search > iSeries** from the workbench menu.



The Search window opens.

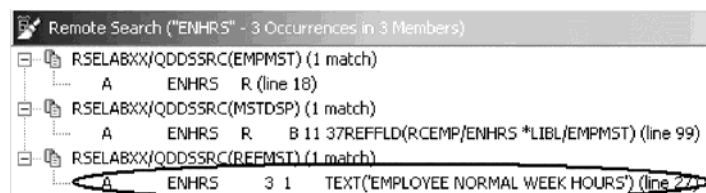
2. In the **Search string** field, type ENHRS.



The **Connection** field should contain your iSeries server name, otherwise enter it there.

3. Under **Target** in the **Library** field, type RSELABXX.
4. Under **Target** in the **File** field, type QDDSSRC to search all members in this source physical file.
5. Under **Target** in the **Member** field, select \*ALL.
6. Click **Search**.

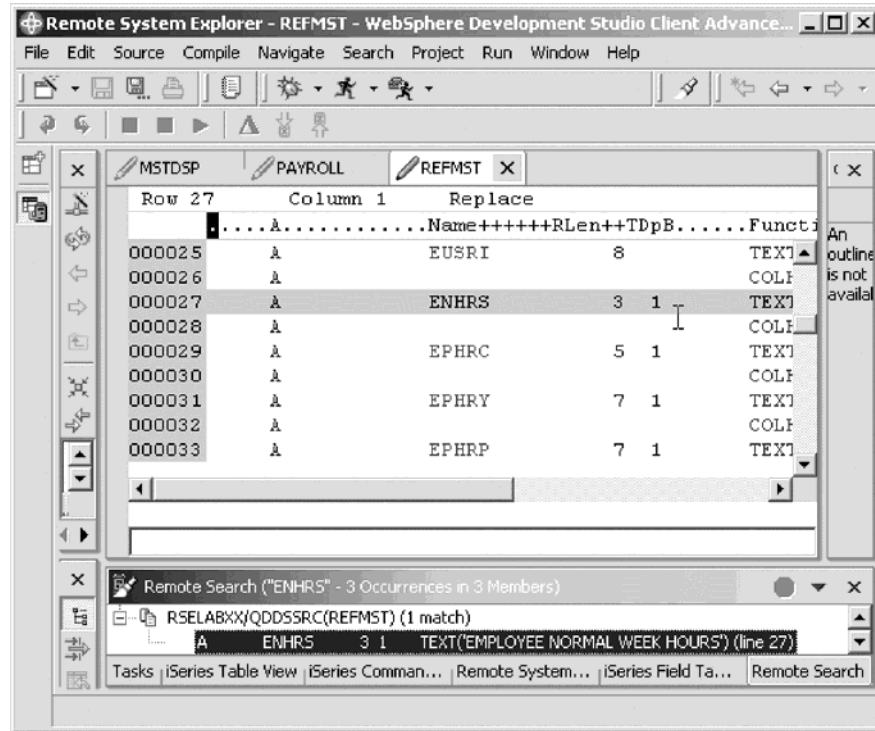
The Multi-File Search window lists all the lines in all the files that reference ENHRS.



7. Double-click the last line in the list.

```
A          ENHRS 3 1      TEXT('EMPLOYEE NORMAL WEEK HOURS')
```

The member REFMST is automatically loaded into the editor and the cursor is placed on the correct line.



8. Close REFMST.

## Comparing file differences from the Remote Systems view

If your product undergoes many changes, you will find the Compare utility useful. It allows you to compare different versions of a program and find the differences. There are two ways to do a compare: use the Compare utility in the workbench or use the Compare utility in the CODE tool. The compare in the CODE tool is more intuitive but requires you to start the CODE Editor outside of the workbench.

Using the compare utility in the workbench you can view the differences between two files by comparing them. You can compare different files, and you can compare versions in the Workbench with versions in the repository or with the local edit history. In some cases you can compare three files (when a common ancestor exists).

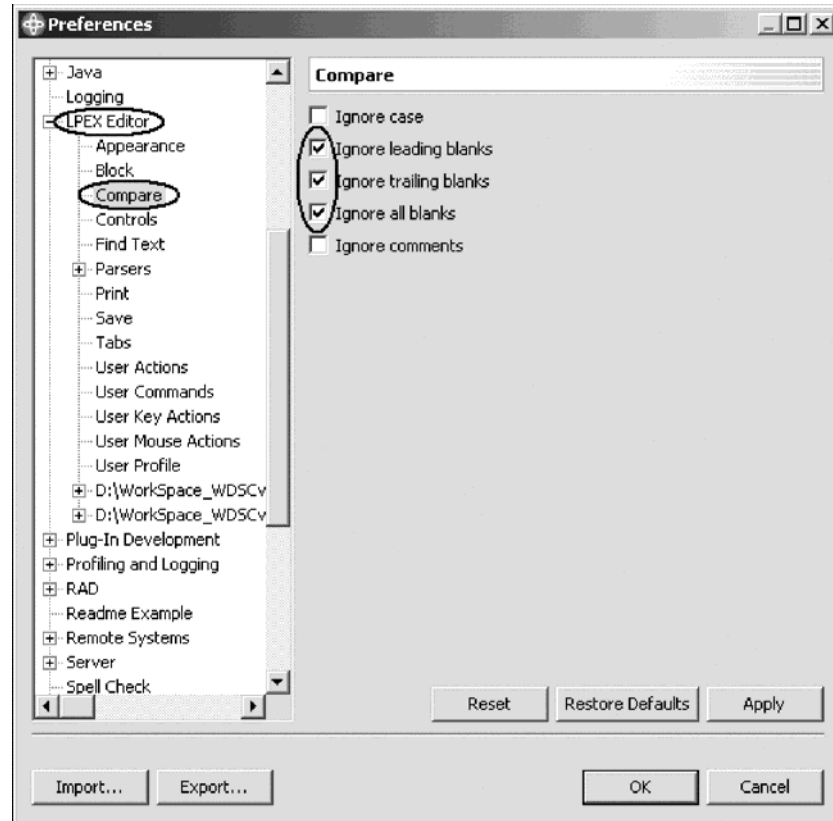
After a comparison is carried out, the Compare Editor opens in the editor area. In the compare Editor, you can browse through all the differences and copy highlighted differences between the compared resources. You can save changes to resources that are made in the comparison editor.

Using the compare utility in CODE you can also view the differences between two files by comparing them. You enter a name of a file to compare against the file in the CODE Editor view. You can type the name of a file, or you can select one from the list of files already open in the editor. If you type the name of the file that is not already open in the editor, it is loaded into the editor. If no file is specified, the current file is compared against a new, untitled file. The current file appears on the left side of the Compare view, and the specified file on the right. You use the Compare menu to view the next and previous mismatch and to select options such as ignore case, font, protect view and show mismatches only.

**Note:** Make sure all lines show in the source before starting the Compare tool. You can do this by right-clicking in the Editor window and clicking **Show All** on the pop-up menu.

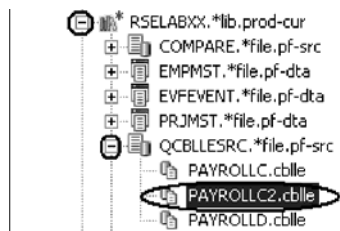
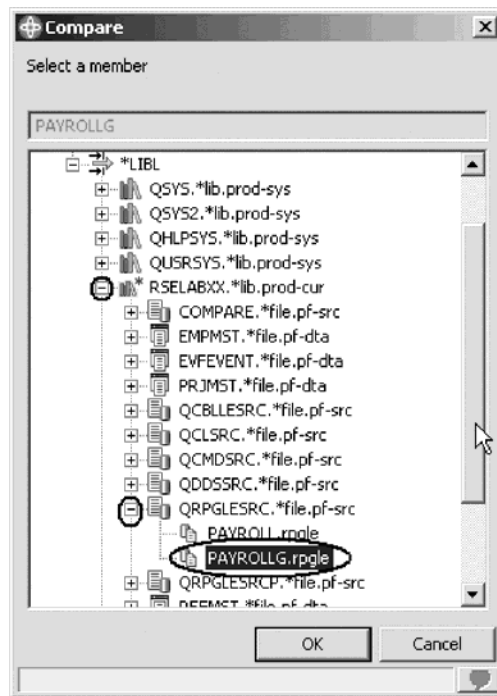
To compare files in the workbench:

1. Click **Window > Preferences** from the workbench menu.  
The Preferences window opens.
2. In the left pane of the Preferences window, expand LPEX Editor.



3. Click Compare under LPEX Editor.  
In the right pane of the Preferences window make sure that the **Ignore blanks** check boxes are selected.
4. Click **OK** in the Preferences window.
5. Back in the Edit window of the PAYROLL member or PAYROLLC member double-click the **PAYROLL** tab or **PAYROLLC** tab.
6. Click **Edit > Compare > Compare** from the workbench menu.  
The Compare window opens.
7. Expand your connection.
8. Expand \*LIBL.
9. Expand RSELABXX.
10. Expand QRPGLSRC or QCBLLESRC.

11. Select member PAYROLLG or PAYROLLC2.



12. Click OK.

The editor now will show the differences of these two members PAYROLL and PAYROLLG or PAYROLLC and PAYROLLC2.

You can move from mismatch to mismatch by going back to the Compare menu under the Edit menu.

Mismatches in PAYROLL and PAYROLLG or PAYROLLC and PAYROLLC2 are highlighted in different colors so that you know where the mismatched lines



are in each file.

```
Row 52      Column 1      Replace
...CLON01Factor1+++++Opcode(E)+Factor2+++++Result+++
003800      C*****
003900      C* This mainline routine controls the display file pr
004000      C* editing. Using the function keys described on ee
004100      C* format, you can transfer from one maintenance appl
004200      C* another. The action code you select on the select
004300      C* determines if the program will add a new record to
004400      C* update an existing record in the file.
004500      C*****
004600      C* Housekeeping, clear display fields and reset indic
004700      C
004800      C              EXSR      MAIN
004900      C* If MAIN is done program ends
005000      C              eval      *INLR = *on
005100      C * MAIN SUBROUTINE
005200      C      MAIN      BEGSR      *INKC
005300      C              dou      *INKC
005400      C              EVAL      *IN60 = *OFF
005500      C              EVAL      EMESS = *BLANK
005600      C              EVAL      EMPAPL = *BLANK
005700      C              EVAL      PRJAPL = *BLANK
```

```
Row 87      Column 1      Replace
---+---*A-1-B---+---2---+---3---+---4---+---5---+---
000085      01 ERROR-MESSAGES.
000086      05 MSG-TABLE.
000087      10 MSG-1          PIC X(50) VALUE
000088      10 MSG-1          PIC X(50) VALUE
000089      ' MAINTENANCE SELECTION CODE NOT EQUAL
000090      10 MSG-2          PIC X(50) VALUE
000091      'MORE THAN ONE APPLICATION SELECTED FOR
```

Next, end the compare session.

13. Click **Edit > Compare > Clear** from the workbench menu.
14. Double-click the **PAYROLL** tab or the **PAYROLLC** tab to return the Editor window to its original size.

## Checking syntax

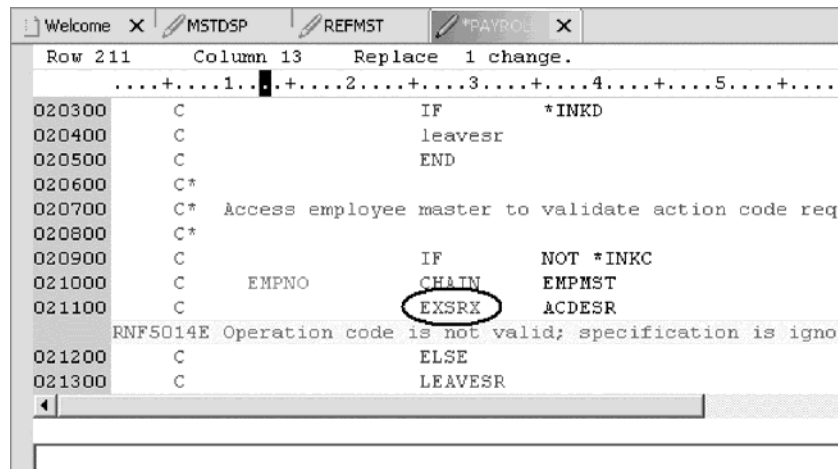
One of the powerful features that the LPEX Editor shares with SEU is its ability to syntax check your source. Syntax checking can be done either when the cursor leaves each line of source or all at once on either the currently selected source or on the entire file.

Now you will create a syntax error and watch for the prompt to correct it.

To syntax check the RPG file:

1. In the **PAYROLL** Editor window move the cursor to line 211 which contains **EXSR ACDESR**.  
You might already be on that line but if not, type the line number in the sequence number column, or scroll down.
2. Append an **X** to the **EXSR** op-code to make it **EXSRX**.
3. Move the cursor off of the line.

An error message appears to draw attention to the error.

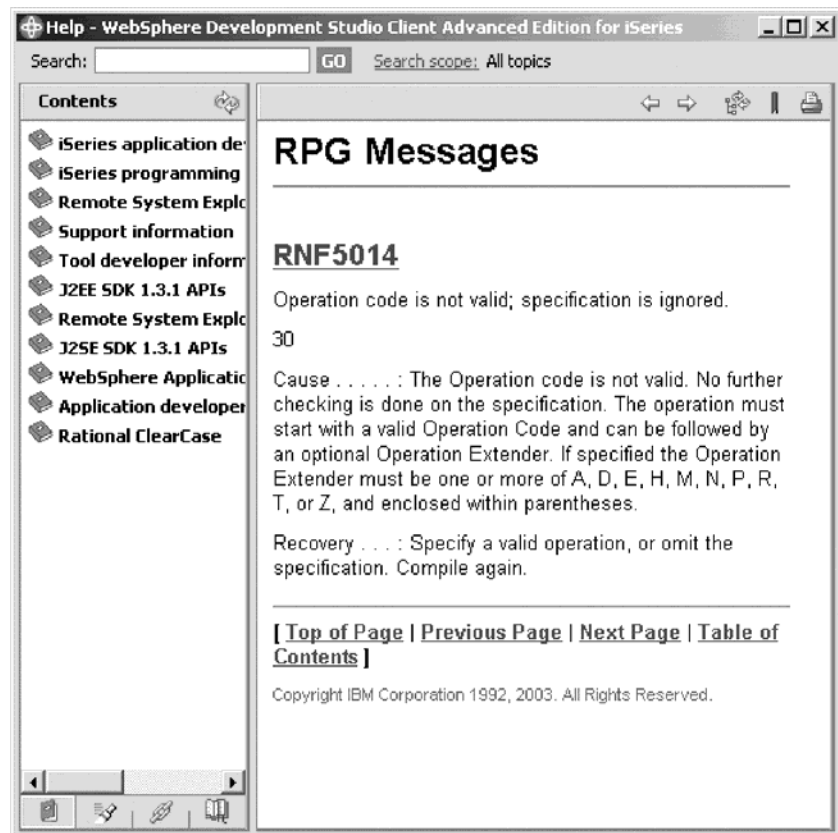


The screenshot shows a COBOL editor window with several tabs: 'Welcome', 'MSTDSP', 'REFMST', and '\*PAYROL'. The main window displays a table of code with columns for Row, Column, and Replace. The code is as follows:

Row	Column	Replace
020300	C	IF *INKD
020400	C	leavesr
020500	C	END
020600	C*	
020700	C*	Access employee master to validate action code req
020800	C*	
020900	C	IF NOT *INKC
021000	C EMPNO	CHATN EMPMST
021100	C	EXSRX ACDESR
RNF5014E Operation code is not valid; specification is igno		
021200	C	ELSE
021300	C	LEAVESR

The error message 'RNF5014E Operation code is not valid; specification is igno' is highlighted in pink. The code 'EXSRX' in the previous row is circled in black.

4. Move the cursor onto the pink error message.
5. Press F1.



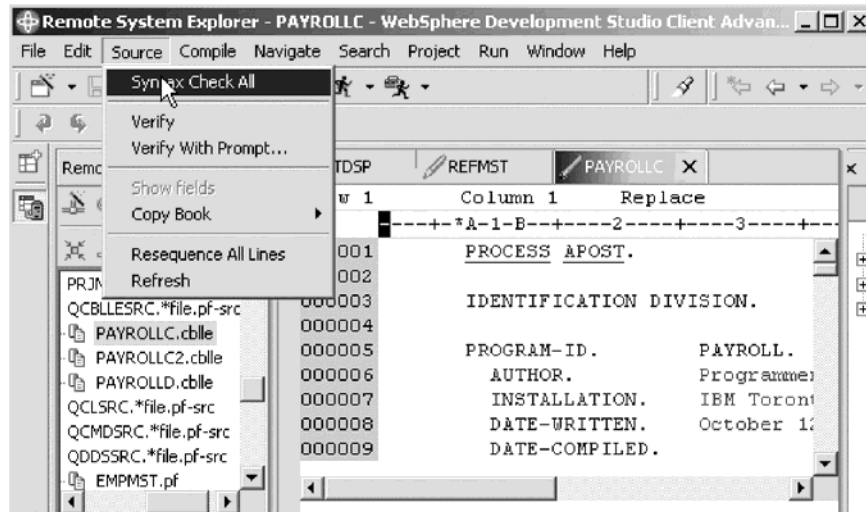
This opens a window with second level help for the error.

6. Minimize the Help window.
7. Change 'EXSRX' to 'EXSR' to correct the error.
8. Move the cursor off the line you just fixed.

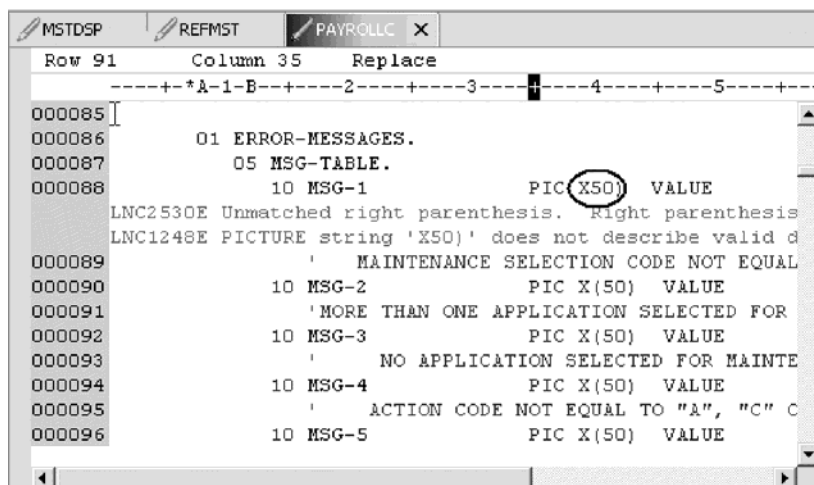
The error message is automatically removed from the editor.

To syntax check the COBOL file:

1. Click the PAYROLLC Editor window, click **Source** and then click **Source > Syntax Check All** on the workbench menu.

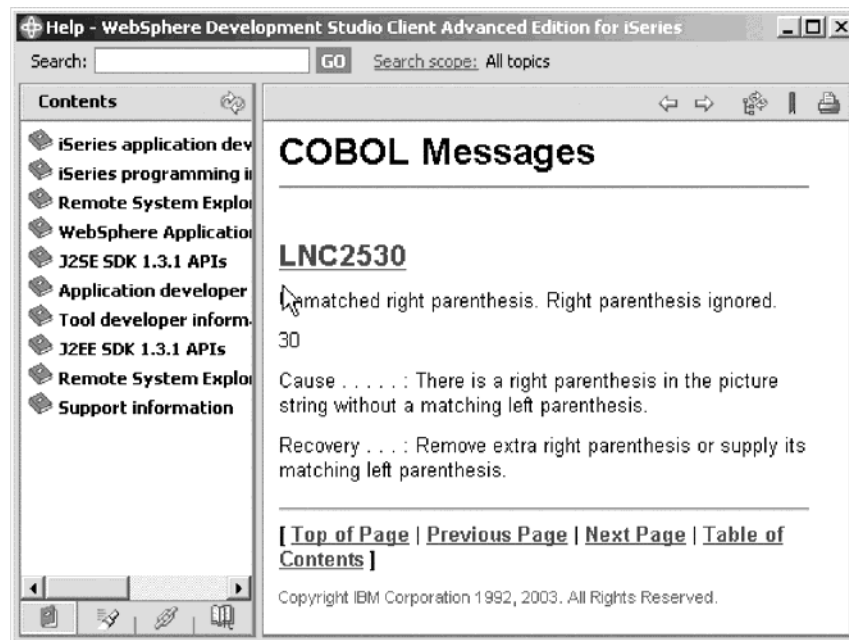


Error messages appear to draw attention to the errors.



2. Move the cursor onto a pink error message.

3. Press F1.

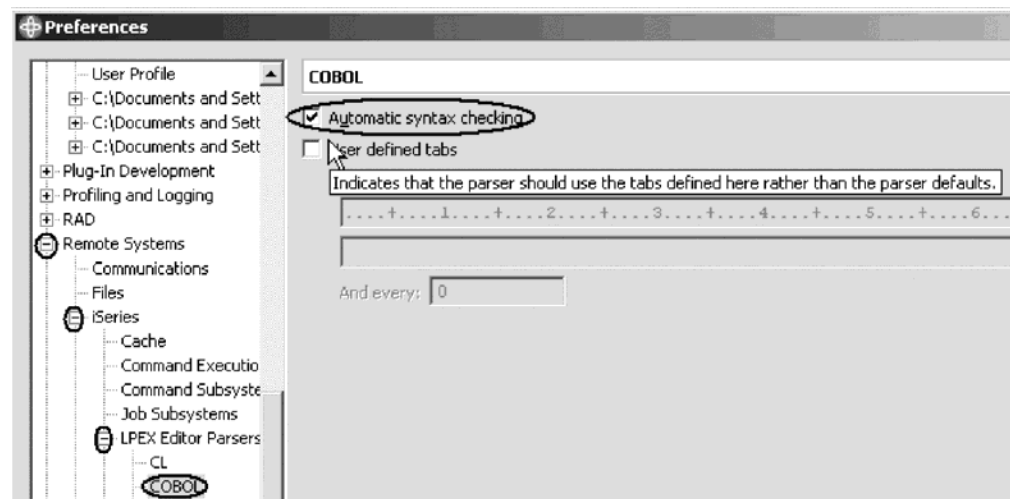
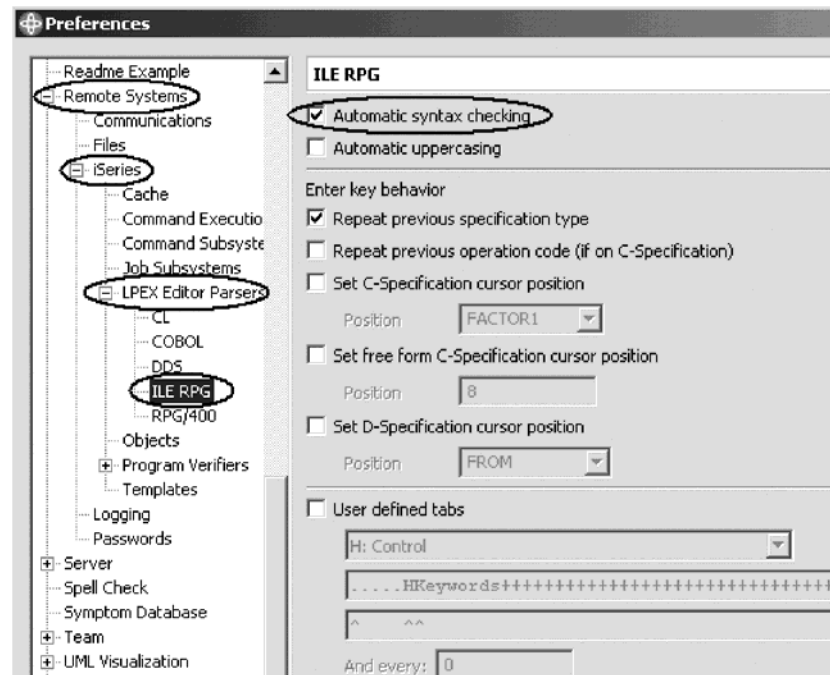


This opens a window with second level help for the error.

4. Minimize the Help window.
5. Add the required left parenthesis to correct the error.
6. Move the cursor off the line you just fixed.  
The error message is automatically removed from the editor.
7. Close the file and save the changes.

Tip: You can toggle automatic syntax checking. Click **Window > Preferences** from the workbench menu and then expand Remote Systems, iSeries, LPEX Editor Parsers, ILE RPG or COBOL in the left pane of the Preferences window, select the

Automatic syntax checking check box, then click OK.



## Checkpoint

Complete the checkpoint below to determine if you are ready to move on to the next module.

1. When column sensitive editing is selected:
  - a. Each column is considered a separate entry space
  - b. If you are inserting or deleting characters into a string that is in the Factor 2 entry, the result field entry does not move
  - c. The default editor preference is that column sensitive editing is off
  - d. All of the above
2. The LPEX Editor has predefined settings, but also has an associated preference page containing settings that you can define. (T, F)
3. LPEX Editor preferences are set in the:

- a. Preferences window
  - b. Editor window
  - c. New wizard
  - d. Remote Systems view
4. You can configure the LPEX Editor to adopt the keyboard and command personalities of many popular editors. (T, F)
5. If you want to undo a set of changes made to a file you must use the \_\_\_\_\_ operation. Name the operation.
- a. Insert
  - b. Replace
  - c. Redo
  - d. Undo
6. You can also cancel the effects of an undo operation by using the \_\_\_\_\_ operation. Name the operation.
- a. Insert
  - b. Replace
  - c. Redo
  - d. Undo
7. This help is invaluable if you cannot remember the order of fields in an RPG specification or the possible values for a variable field. Name the help. Choose the best answer.
- a. Context sensitive help
  - b. Language sensitive help
  - c. Display help
  - d. Field help
8. To receive language sensitive help, press \_\_\_\_ key in an Edit window. Name the key:
- a. F2
  - b. F3
  - c. F1
  - d. F4
9. If the cursor is \_\_\_\_\_ a field, you receive help for that field.
- a. Before
  - b. After
  - c. On
  - d. Off
10. Instead of entering or changing code directly in the Editor window, you can use \_\_\_\_\_.
- a. Prompts
  - b. Filters
  - c. SEU commands
  - d. Format line
  - e. All of the above
11. \_\_\_\_\_ views are hidden views, which can be quickly made visible. They work identical to normal views, the only difference being that when hidden they do not take up screen space on your workbench window.
- a. Remote Systems

- b. Navigator
  - c. Outline
  - d. Fast
12. The \_\_\_\_\_ option allows you to view your source with constructs in an indented mode. By default, the \_\_\_\_\_ option will split the screen horizontally and how show the indented view in the bottom pane.
- a. Fast view
  - b. Outdent
  - c. Indent
  - d. Edit
13. The indented view is browse mode only and cannot be edited. (T, F)
14. You use the \_\_\_\_\_ window to search for an item. Choose the best answer.
- a. Search
  - b. Find
  - c. Edit
  - d. Find and Replace
15. You can search for:
- a. A word
  - b. A partial word
  - c. A sequence of words
  - d. A pattern if it follows the rules of regular expression
  - e. All of the above
16. The LPEX Editor allows you to \_\_\_\_ or subset your source so that you see only lines containing a given string.
- a. Search
  - b. Find
  - c. Sort
  - d. Filter
17. To help you navigate quickly through your ILE RPG source the editor allows you to filter lines based on the \_\_\_\_\_. Choose the best answer.
- a. String
  - b. Line type
  - c. Line number
  - d. All of the above
18. If you would like to search through the members in a source physical file or through the files in a local directory, you can use the \_\_\_\_\_ tool. Choose the best answer.
- a. Compare
  - b. Search
  - c. Find
  - d. Edit
19. The \_\_\_\_\_ tool allows you to compare different versions of a program and find the differences. Choose the best answer.
- a. Convert
  - b. Migrate
  - c. Compare

- d. Search
20. There are two ways to compare files. They are Compare tool in the workbench and the Compare tool in the CODE Editor. (T, F)
21. Syntax checking can be done either when the cursor leaves each line of source or all at once on either the currently selected source or on the entire file. (T, F)
- 

## More practice

Want more practice? Try this!

Given your experience in this exercise using the Remote Systems LPEX Editor, try these new tasks: Check out how token highlighting works in your ILE RPG source. Display a format line and use it to enter your source. Filter source by comments. Set auto-uppercase on. Use the Development Studio Client for iSeries online help to assist you in these tasks.

---

## Recap

Congratulations! You have completed this module. You should now understand:

- The benefits of each language editing feature
- How to do column sensitive editing
- How to enter SEU commands
- How to undo and redo editing
- How to access help for language elements
- How to prompt on language elements
- How to indent source
- How to find and replace language elements
- How to filter the source by string and by line type
- How to search files
- How to compare files
- How to syntax check files

Now that you have mastered editing source, you can move on to verifying your source to ensure you have a clean compile on the iSeries system. This approach saves you iSeries cycles! Continue to the next module.



---

## Chapter 7. Verifying source

In this module, you will learn how to verify source in the Remote Systems LPEX Editor. The program verifying and program compiling are the final steps in compiling a program or module. When errors are found by either, the iSeries Error List appears. The iSeries Error List is a powerful tool that manages errors found by verify and compile utilities in the Remote Systems LPEX Editor. You will become familiar with these tools, the various capabilities of the iSeries Error List and the program that you have created.

In order to verify source, there are several steps you will need to learn about and follow:

- Describe the purpose of the Program Verifier
- Invoke the Program Verifier
- Insert messages and checking that errors were fixed using the iSeries Error list
- Submit iSeries commands using the iSeries Table view

In order to accomplish these learning objectives, there are several steps that are involved, including:

- Invoking the Program Verifier
- Reviewing the iSeries Error List
- Fixing Errors

The exercises within this module must be completed in order. Start with the first exercise when you are ready to begin.

**Length of time:**

This module will take approximately 10 minutes to complete.

---

### Verifying RPG or COBOL source

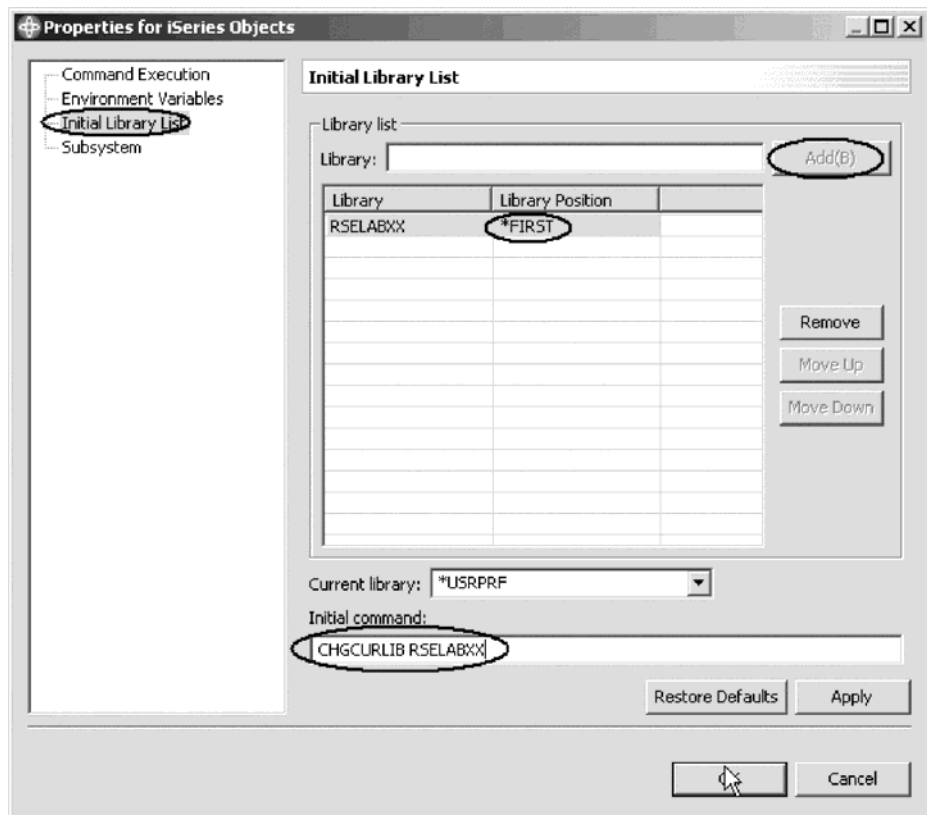
Now you get to play with one of the most powerful and unique features of the Remote System Explorer – the Program Verifier.

Now you get to play with one of the most powerful and unique features of the Development Studio Client — the Program Verifier.

Before you compile your code on an iSeries, you can make certain that there are no errors by invoking the Program Verifier. The verifier checks for semantic (compile) errors on your workstation so that you can guarantee a clean compile on the iSeries. Think of the host cycles you'll save. It is especially handy when you are writing code but you are disconnected from an iSeries. You can do this because Development Studio Client ported the parsing and checking code from the iSeries system compilers to the workstation. The iSeries Error List window lists the errors that are found and their severity, inserts the error messages directly into the source and helps you to navigate between the errors.

**Note:** Make sure that your user ID has been setup, so that your library has been added to the library list automatically. You can use the properties for iSeries Objects dialog to set connection information such as adding a library to a library list and changing the current library. From the Remote Systems view,

you can right-click on iSeries Objects then click Properties to see this dialog. You will need this setup to run jobs interactively from your workstation.



To invoke the verifier:

1. Click the **PAYROLL** tab to verify RPG source or **PAYROLLC2** tab (you will need to open the PAYROLLC2 file for edit first) to verify COBOL source.
2. Click **Source > Verify** from the workbench menu.

After a moment the verifier will display an iSeries Error List below the Editor window.

ID	Message	S...	L...	Location	Connection
RNF...	ENDSR operation is missin...	3...	1	RSELABXX/QRPG...	s400a
RNF...	The name or indicator EME...	3...	3...	RSELABXX/QRPG...	s400a
RNF...	The name or indicator RSN...	3...	9...	RSELABXX/QRPG...	s400a
RNF...	The operand RSNTAX of E...	3...	9...	RSELABXX/QRPG...	s400a
RNF...	Factor 2 must be '0' or '1' ...	3...	3...	RSELABXX/QRPG...	s400a
RNF...	Move operands ERR and E...	3...	3...	RSELABXX/QRPG...	s400a
RNF...	RPG provides Separate-In...	0	2...	RSELABXX/QRPG...	s400a

ID	Message	S...	Line	Location	Connection
LNC1904	Program PAYROLL syntax checked. Errors ...		40	1	RSELABXX/QCBL... s400a
LNC1463	'EMPNO' is not unique in this context. Use ...		30	302	RSELABXX/QCBL... s400a
LNC1326	'PRCD OF PRJSEL-I' not defined name. Sta...		30	402	RSELABXX/QCBL... s400a
LNC1329	Subscript value '14' exceeds maximum occ...		30	606	RSELABXX/QCBL... s400a
LNC0407	AT END phrase missing from sequential RE...		20	184	RSELABXX/QCBL... s400a
LNC0407	AT END phrase missing from sequential RE...		20	299	RSELABXX/QCBL... s400a
LNC0407	AT END phrase missing from sequential RE...		20	348	RSELABXX/QCBL... s400a
LNC0407	AT END phrase missing from sequential RE...		20	399	RSELABXX/QCBL... s400a
LNC0407	AT END phrase missing from sequential RE...		20	448	RSELABXX/QCBL... s400a
LNC0407	AT END phrase missing from sequential RE...		20	499	RSELABXX/QCBL... s400a
LNC0407	AT END phrase missing from sequential RE...		20	548	RSELABXX/QCBL... s400a

The error list shows you:

- The error message itself
- The severity
- The line number
- The source location
- The connection name

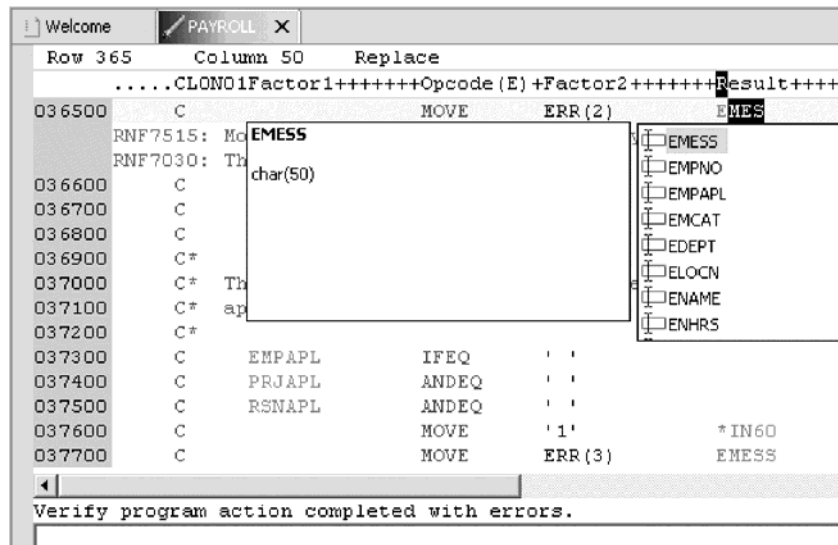
## Fixing RPG errors

Next you fix the errors in your source.

To fix an error in your source go the error list:

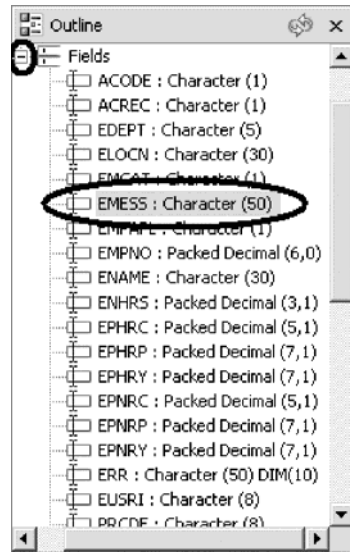
1. Double-click the error RNF7030.

You are automatically brought back into the Editor window to the line where the error occurred. The error on line 365 is a typo. The variable EMES is really EMESS. One good way of finding the correct name is the content assist.



2. Refresh the Outline view.
3. Select the misspelled variable and press **CRTL+spacebar**.  
If the variable starts correctly the selection presented contains the correct name.
4. Double-click the variable EMESS in the list to correct the variable name.

Another way to find the variable name is to use the Outline view and see what variables are declared.



The next error is a RNF7030 as well.

5. Double-click RNF7030. Fix it in the editor.
6. RSNTAX should really be RSNTAG.

Make the appropriate change.

7. Go to the next error RNF5184.

The next error is actually RNF7018 but it is related to the first one and can be ignored. It shows the same line number, which is an indication that both errors are related.

8. Double-click RNF5184.

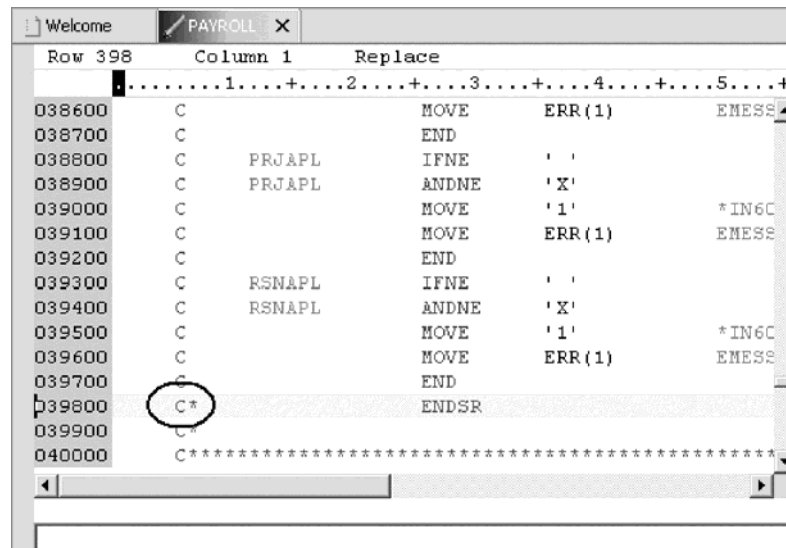
Certainly somebody didn't know their RPG and tried to assign a value 2 to an indicator.

9. Fix this by replacing the 2 with a 1.

Error RNF7515 is related to the first error you fixed. It has the same line number. You can ignore it. The only serious remaining error is RNF5178. This error is caused by a missing ENDSR.

**Note:** The verifier could not determine where the ENDSR is missing so the line number reported is 1, for this reason you can't just double click on the error message. You need to investigate where the missing statement belongs.

- Move to line 398 in the editor.



- Remove the asterisk (\*) from the C-spec.  
You can use the Tab key to quickly move to the appropriate column.  
All the non-informational errors are now fixed. You can filter out different severities by using the filter menu.



- Click the arrow in the iSeries Error List title bar.
- Click **Show Severity** on the pop-up menu.
- Clear the severities you don't want to see in the list (Warning for example).

## Fixing COBOL errors

Next you fix the errors in your source.

To fix an error in your source go the error list:

- Double-click the error LNC1463.  
You are automatically brought back into the Editor window to the line where the error occurred. The error is caused by EMPNO not being uniquely defined. Go to line 302 where EMPNO is defined. The variable EMPNO should be EMPNO OF EMPSEL-I.
- Make the change.  
The next error is LNC1326.
- Double-click LNC1326. Fix it in the editor.
- PRCD should really be PRCDE.  
Make the appropriate change on line 402.
- Double-click LNC1329.  
This error is because the array was declared with 4 elements. Go to line 606.

- Change the index of the array from 14 to 4.  
The next errors are LNC0407. You can ignore these errors as they are severity 20.  
All the severity 30 errors and above are now fixed. You can filter out different severities by using the filter menu.



- Click the arrow in the iSeries Error List title bar.
- Click **Show Severity** on the pop-up menu.
- Clear the severities you don't want to see in the list (Warning for example).

## Saving the source and verifying source again

Now before you loose any of your changes, it's a good idea to save them. Make sure the PAYROLL member is selected. You then verify the source again to make sure that all the errors are fixed.

You can save the member using one of these ways:

- Click **File > Save** from the workbench menu.
- Click the Save icon in the workbench toolbar.
- Press **Ctrl+S**.  
Changes are uploaded to the iSeries.
- Verify your source again; **Source > Verify**.

The screenshot shows the 'iSeries Error List' window for source '<TORA507M>RSELABXX/QRPGLESRC(PAYROLL)'. The table displays several error messages with severity 0. The columns are ID, Message, Sev..., Line, and Location.

ID	Message	Sev...	Line	Location
RNF7089	RPG provides Separate-Indicat...	0	27	RSELABXX/QRPGL...
RNF7031	The name or indicator *IN03 is ...	0	1	RSELABXX/QRPGL...
RNF7031	The name or indicator *IN04 is ...	0	1	RSELABXX/QRPGL...
RNF7031	The name or indicator *IN05 is ...	0	1	RSELABXX/QRPGL...
RNF7031	The name or indicator *IN06 is ...	0	1	RSELABXX/QRPGL...
RNF7031	The name or indicator *IN07 is ...	0	1	RSELABXX/QRPGL...

The screenshot shows the 'iSeries Error List' window for source '<TORA507M>RSELABXX/QCBLLSRC(PAYROLLC2)'. The table displays several LNC0407 error messages with severity 20. The columns are ID, Message, S..., Line, Location, and Connection.

ID	Message	S...	Line	Location	Connection
LNC0407	AT END phrase missing from sequential RE...	20	184	RSELABXX/QCBLL...	s400a
LNC0407	AT END phrase missing from sequential RE...	20	299	RSELABXX/QCBLL...	s400a
LNC0407	AT END phrase missing from sequential RE...	20	348	RSELABXX/QCBLL...	s400a
LNC0407	AT END phrase missing from sequential RE...	20	399	RSELABXX/QCBLL...	s400a
LNC0407	AT END phrase missing from sequential RE...	20	448	RSELABXX/QCBLL...	s400a
LNC0407	AT END phrase missing from sequential RE...	20	499	RSELABXX/QCBLL...	s400a
LNC0407	AT END phrase missing from sequential RE...	20	548	RSELABXX/QCBLL...	s400a

Everything should be ok. You are ready to compile the program.

## Checkpoint

Complete the checkpoint below to determine if you are ready to move on to the next module.

1. The \_\_\_\_\_ tool checks for semantic (compile) errors on your workstation so that you can guarantee a clean compile on the iSeries.
  - a. Compile
  - b. LPEX Editor
  - c. Program Generator
  - d. Program Verifier
2. The \_\_\_\_\_ view identifies each error with an icon that identifies the severity level of the error, the ID of the error, the message, the severity, the line in the source member that caused the error, the location of the source member that produced the error, and the connection name.
  - a. Remote Systems
  - b. Outline
  - c. Navigator
  - d. iSeries Error List
3. You can sort the entries in the iSeries Error List view by:
  - a. ID
  - b. Message
  - c. Severity
  - d. Line
  - e. Location
  - f. Connection
  - g. All of the above
4. If you used the local program verifier, then your host compiles should be successful; no wasted iSeries cycles. (T, F)

---

## More practice

Want more practice? Try this!

Given your experience in this exercise using the Program Verifier and that you have your own source on your own iSeries system try these new tasks: Check out Verify again on your source files. Assuming you receive errors in your source (add some errors into your source if you don't have any) when you verify your source, choose insert all error messages into the editor from the Error list. Use the Development Studio Client for iSeries online help to assist you in these tasks.

---

## Recap

Congratulations! You have completed this module. You should now understand:

- The purpose of the Program Verifier
- The purpose of the iSeries Error List
- How to invoke the Program Verifier
- How to view and fix errors using the iSeries Error list
- How to save the fixed source

Now that you know how to verify source, you can move on to compiling the source. Continue to the next module.





---

## Chapter 8. Compiling source

In this module, you will learn how to compile source in the Remote Systems LPEX Editor. The program compiling are the final steps in compiling a program or module. When errors are found by either, the iSeries Error List appears. The iSeries Error List is a powerful tool that manages errors found by compile utilities in the Remote Systems LPEX Editor. You will become familiar with this tool, the various capabilities of the iSeries Error List and the program that you have created.

In order to compile source, there are several steps you will need to learn about and follow:

- Submit iSeries commands using the iSeries Table view in Remote System Explorer
- Insert messages and checking that errors were fixed using the iSeries Error list
- Select compile options and compiling source using the Compile utility
- Explain the generated program called PAYROLL

In order to accomplish these learning objectives, there are several steps that are involved, including:

Compiling interactively

Specifying compile options

Submitting the iSeries commands in the iSeries Table view

Running commands and programs

The exercises within this module must be completed in order. Start with the first exercise when you are ready to begin.

**Length of time:**

This module will take approximately 10 minutes to complete.

---

### Compiling interactively

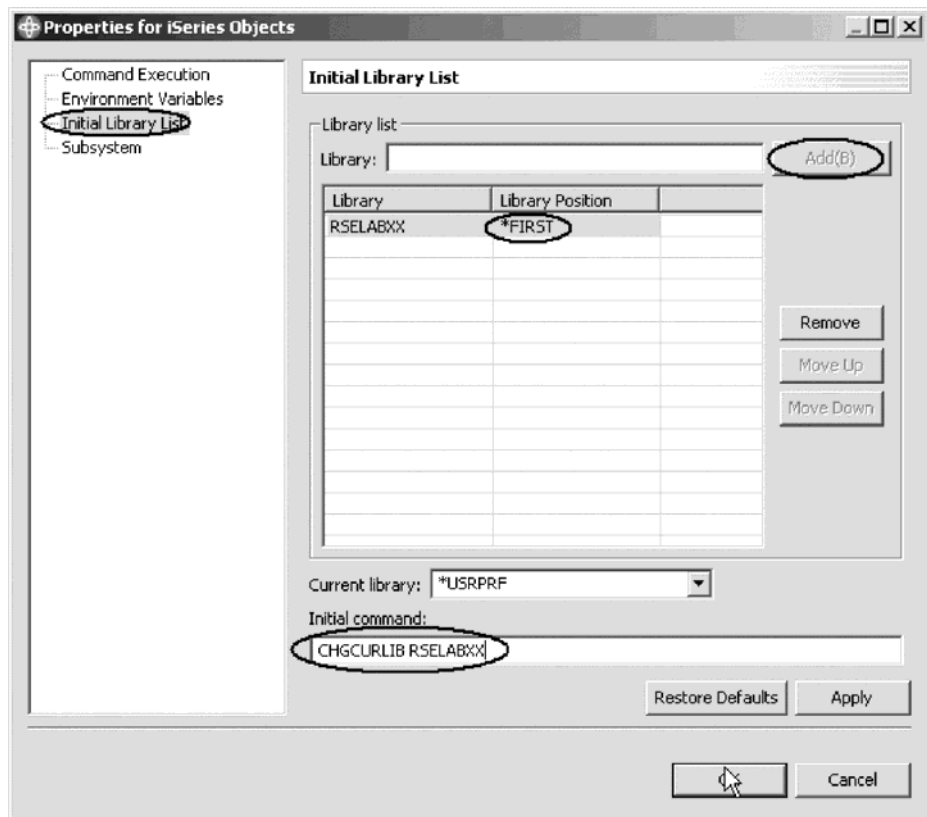
The remote compile capability is part of the Remote System Explorer. It gives you a workstation interface to submit requests to iSeries to compile, bind, or build objects on the host. It allows for easy access to all the compile options available for all the supported CRTxxx commands.

If you used the local program verifier, then your host compiles should be successful -- no wasted iSeries cycles. However, if there are errors, the host compiler will send the error information back to the workstation and they will be loaded into the iSeries Error List window, which behaves just as it did when you did a program verify.

The default for compiling programs is to submit the compile to the batch job queue. Here in this exercise you can run the compile interactive.

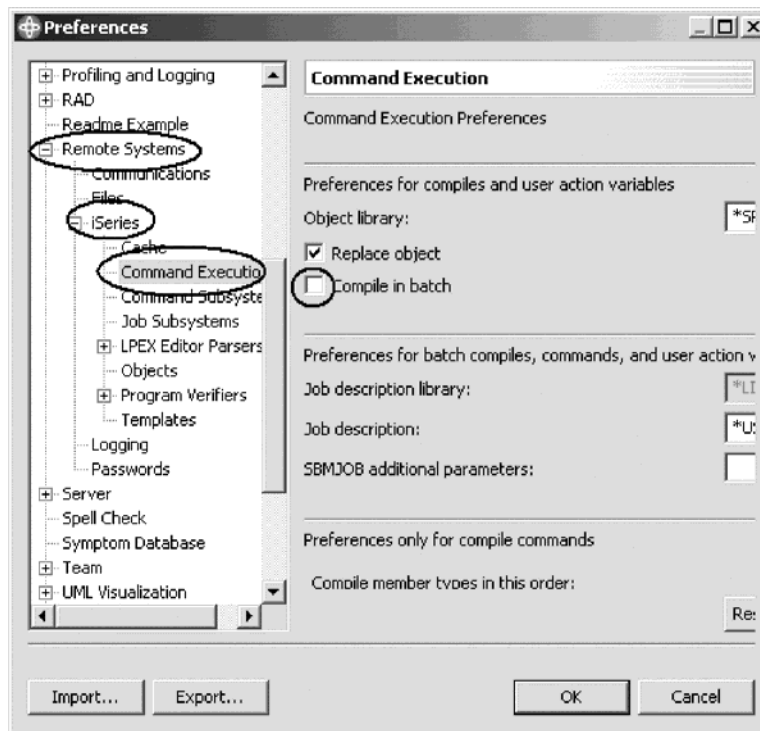
**Note:** Make sure that your user ID has been setup, so that your library has been added to the library list automatically. You can use the properties for iSeries

Objects dialog to set connection information such as adding a library to a library list and changing the current library. You can right-click on iSeries Objects then click Properties to see this dialog. You will need this setup to run jobs interactively from your workstation.



To change the preferences to run the compile interactive:

1. Click **Window > Preferences** from the workbench menu.



2. In the left pane of the Preferences window, expand Remote Systems.
3. Expand iSeries under Remote Systems.
4. Click Command Execution under iSeries.
5. In the right pane of the Preferences window, clear the **Compile in batch** check box.
6. Click **OK** to return to the Remote System Explorer perspective.

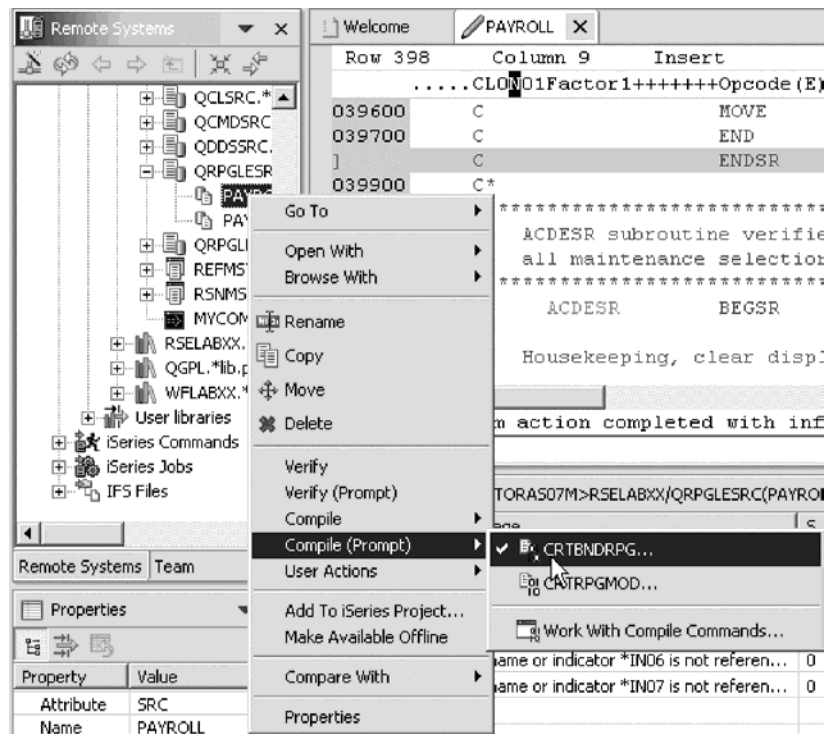
---

## Specifying RPG compile options

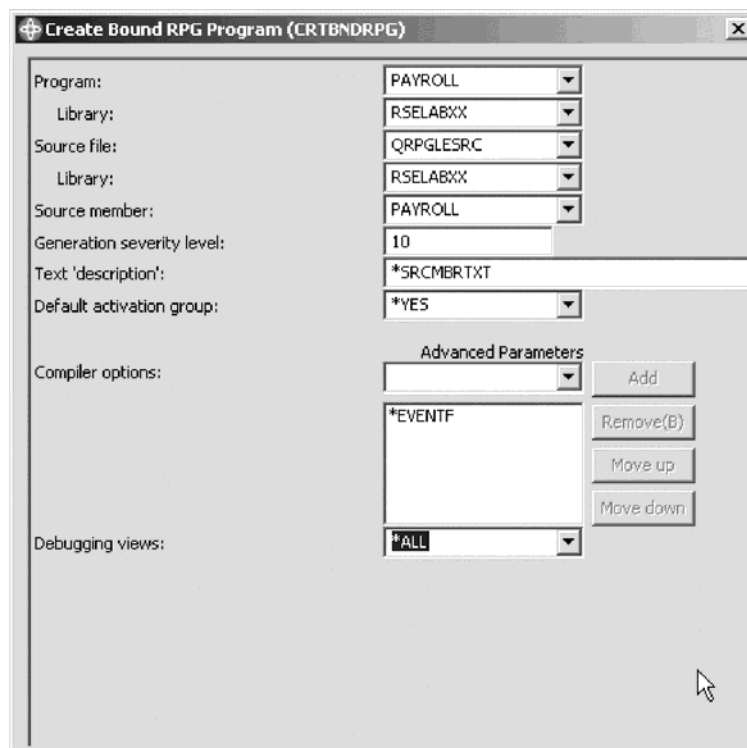
You will now use the prompt for the CRTBNDRPG command to specify your compile parameters. All entry fields pertaining to names are already filled in with the correct information.

To compile source:

1. Right-click the PAYROLL member in QRPGLSRC.



2. Click **Compile (Prompt) > CRTBNDRPG** on the pop-up menu. The Create Bound RPG Program (CRTBNDRPG) dialog opens.
3. In the **Debugging views** list, select the **\*ALL** parameter.

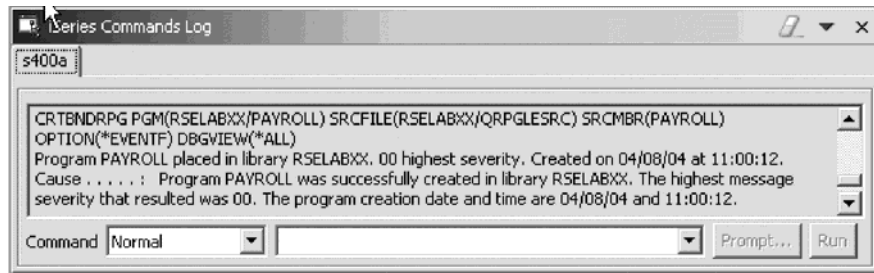


If you want check out the other parameters available, click **Advanced**.

4. Click **OK** when you are done.

The progress bar on the workbench (bottom right corner) will indicate that the compile runs then the error list will be shown, with no errors, just information messages.

If you are not sure that the compile was successful. You can check the iSeries Commands Log.



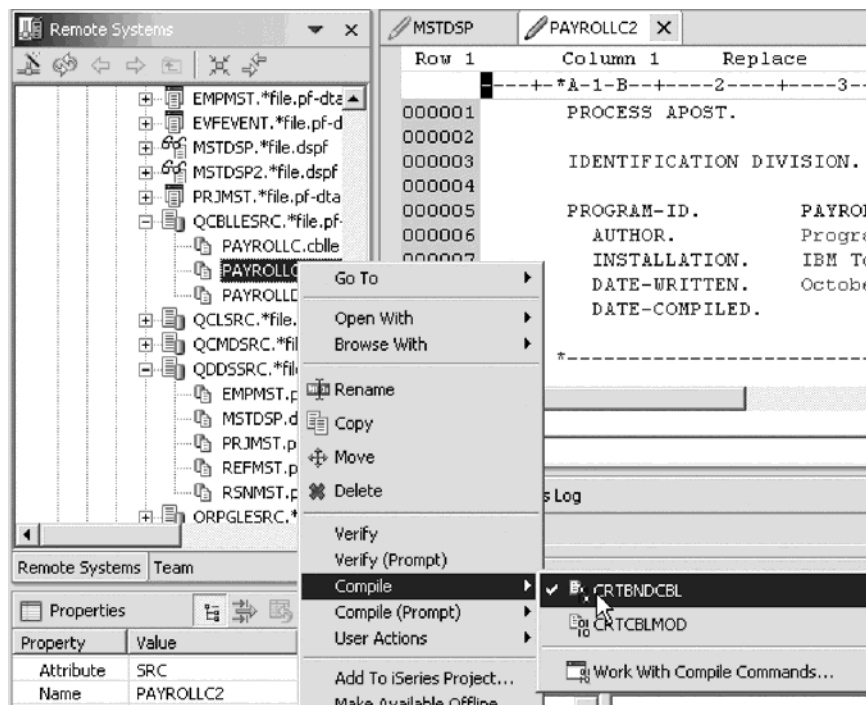
5. Click the **iSeries Commands Log** tab at the bottom of the workbench. This log shows a list of all commands run on the remote system and the messages returned for each command.

## Specifying COBOL compile options

You will now use the prompt for the CRTBNDCBL command to specify your compile parameters. All entry fields pertaining to names are already filled in with the correct information.

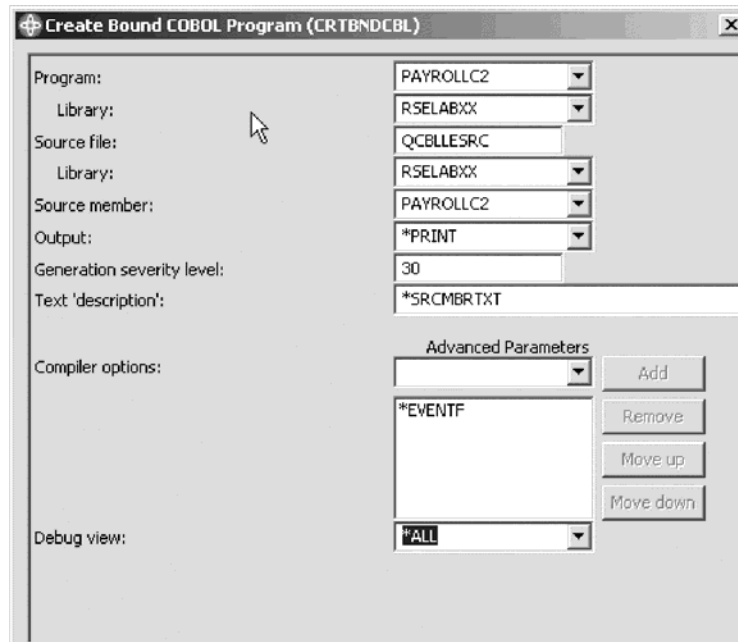
To compile source:

1. Right-click the PAYROLLC2 member in QCBLLSRC.



2. Click **Compile (Prompt) > CRTBNDCBL** on the pop-up menu. The Create Bound COBOL Program (CRTBNDCBL) dialog opens.

- In the **Debugging views** list, select the **\*ALL** parameter.

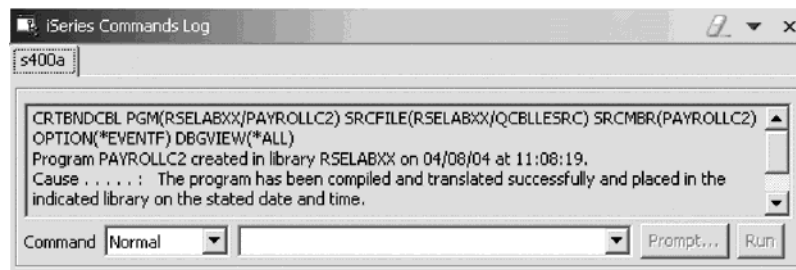


If you want check out the other parameters available, click **Advanced**.

- Click **OK** when you are done.

The progress bar on the workbench (bottom right corner) will indicate that the compile runs then the error list will be shown, with no errors, just information messages.

If you are not sure that the compile was successful. You can check the iSeries Commands Log.



- Click the **iSeries Commands Log** tab at the bottom of the workbench.

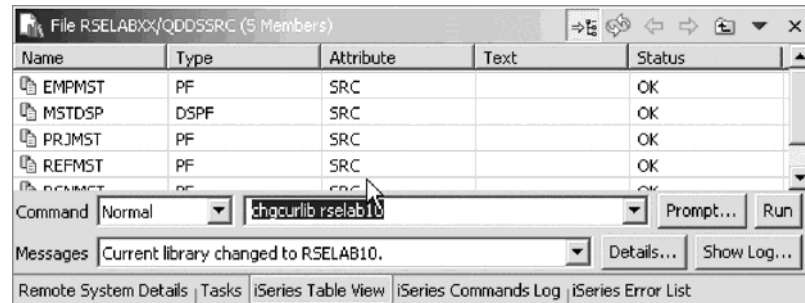
This log shows a list of all commands run on the remote system and the messages returned for each command.

## Submitting iSeries commands in the iSeries table view

You can use the iSeries Table view inside the Remote System Explorer to submit commands to the iSeries. You can run commands in the Commands field beneath the iSeries Table view, and view messages in the Messages field. After you populate the table, you can enter a command and select either Prompt to specify parameters and then Run, or just select Run (which prompts and runs the command as normal). When you run a command, the Messages drop-down field is populated with the messages from the command. When you select a message, the Details button is enabled. When you press this button, the message and its help is displayed.

Also note that you can use the iSeries Table view or the Remote Systems view to run commands and programs. In the iSeries Table view, you can see the properties of all items at the same time; they are displayed as rows across the table. In the Remote Systems view, you have greater ease of navigation; you can work from your Library list in the iSeries Objects subsystem, and you can see the contents of many items before selecting the one you want to run.

To change the library list:

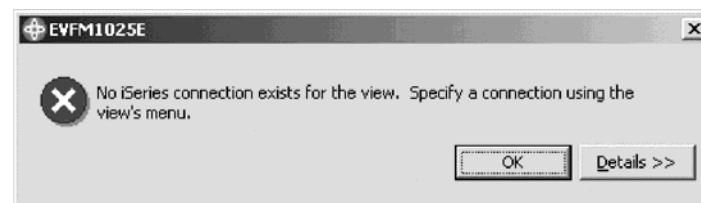


1. Click the **iSeries Table View** tab from the views at the bottom of the workbench.
2. In the **Command** field type, CHGCURLIB RSELAB10 for example.

**Note:** Use a library that is on your iSeries system.

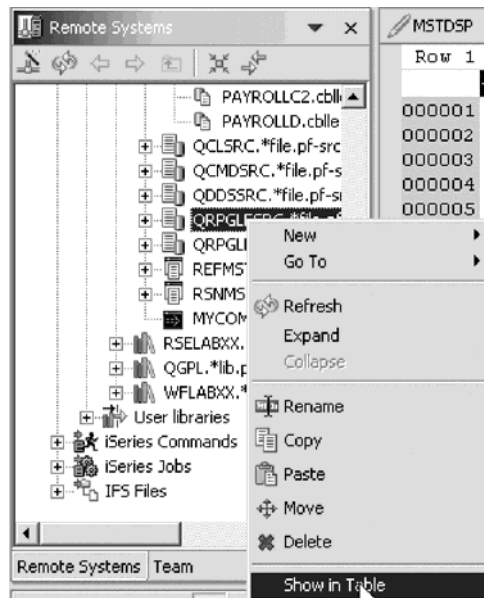
3. Click **Run**.

If you haven't used the iSeries Table view to show iSeries objects in this view you will get this error message because the table view is not linked to an active connection.

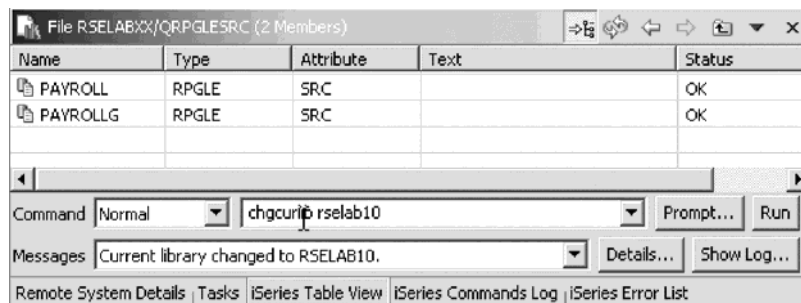


If you see this message, click **OK** and complete the next steps 4, 5 and 6.

- In the Remote Systems view, right-click QRPGLSRC or QCBLLESRC.



- Click **Show in Table** on the pop-up menu.  
The table view is now populated with the member in the selected source file.
- Run the CHGCURLIB command again.  
The command will run on the iSeries and after completion you will see the completion message on the bottom of the iSeries Table view.



You can also connect to other than iSeries systems with the Remote System Explorer and Launch commands for these systems as well, for example, your local system, or LINUX.

## Running commands and programs

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

- In the Remote System Explorer communications server job.  
This is the one you are using currently.
- In a batch job.
- In an interactive job (to test 5250 applications).

Using the first option lets you run the program in the same job as the communications server. 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



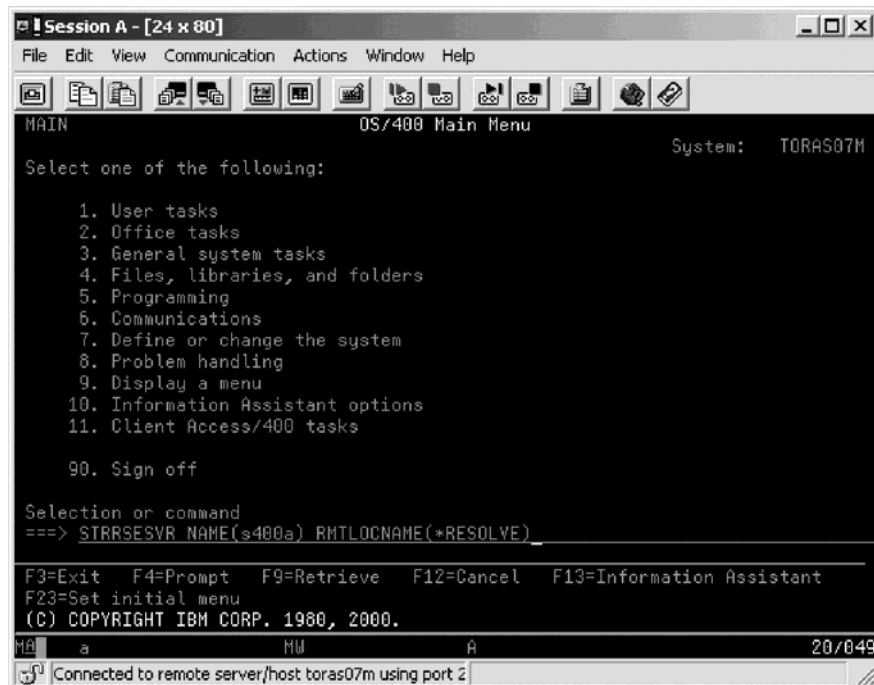
expect and do not require any initial setup. Interactive programs require a 5250 emulator, so you need to first run a STRRSESVR connectionName command to associate the emulator with a particular connection in the Remote System Explorer communications server.

## Starting an interactive connection

To start an interactive connection:

1. Start a 5250-emulation session.
2. Sign-on to the iSeries with your User ID and password.

**Note:** Instead of the **Enter** key, you may have to use the **Ctrl** key in your 5250-emulation session.



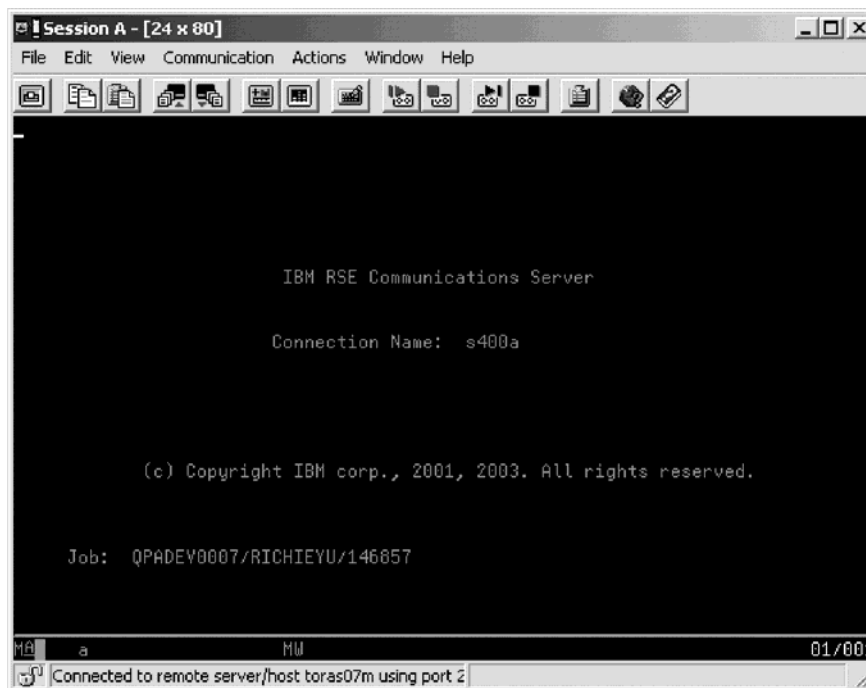
```
Session A - [24 x 80]
File Edit View Communication Actions Window Help
MAIN OS/400 Main Menu System: TORAS07M
Select one of the following:
1. User tasks
2. Office tasks
3. General system tasks
4. Files, libraries, and folders
5. Programming
6. Communications
7. Define or change the system
8. Problem handling
9. Display a menu
10. Information Assistant options
11. Client Access/400 tasks
90. Sign off
Selection or command
==> STRRSESVR NAME(s400a) RMTLOCNAME(*RESOLVE)_
F3=Exit F4=Prompt F9=Retrieve F12=Cancel F13=Information Assistant
F23=Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 2000.
MA a MW A 20/049
Connected to remote server/host toras07m using port z
```

3. In the command line, type the command:  
STRRSESVR connectionName RMTLOCNAME(\*RESOLVE)
4. Press Enter.

The connectionName parameter is the name of your connection defined in the Remote Systems view. This associates the interactive job with the Remote System Explorer communications server. The \*RESOLVE keyword will get the IP address of your workstation and with this information the Remote System Explorer communications server will communicate with the Remote System Explorer daemon that runs on your workstation.

**Notes:**

- a. The connection name is case sensitive and must already be defined in the Remote System Explorer.



- b. This screen will stay this way. Don't wait for it to finish. This session is the interactive session for interactive commands started from the Remote System Explorer.

## Running the payroll program

Now you are ready to run the PAYROLL or PAYROLLC2 program that you just compiled.

**Note:** You may need to refresh the Remote Systems view to see the PAYROLL program or PAYROLLC2 program in the Remote Systems view under the RSELABXX library.

Return to the workbench.

To run the payroll program:

1. In the Remote Systems view, locate the PAYROLL program or PAYROLLC2 that you created.
2. Right-click the PAYROLL program or PAYROLLC2 program.
3. Click **Run > Interactive** on the pop-up menu.
4. Switch to your 5250-emulation session.

You will see the payroll program menu.



5. Type x beside **Employee Master Maintenance**.
6. Press **Enter**.
7. Type 456 for the Employee Number.
8. Type A for the Action Code to add the employee 456.
9. Press **Enter**.
10. Type any information you like about the employee.
11. Press **Enter**.
12. Play in the application as much as you like.
13. Press **F3** to end the PAYROLL program.

If you want to release the interactive job, you could right-click on iSeries Objects and click **Release Interactive Job** on the pop-up menu. However, you may not want to release the interactive job if you expect to debug your program later in this tutorial.

---

## Checkpoint

Complete the checkpoint below to determine if you are ready to move on to the next module.

1. A compile command can be run on an iSeries server from the Remote System Explorer, and you can retrieve error feedback from the compile. (T, F)
2. Each profile in the Remote System Explorer has a set of source member types, and each source type has a set of compile names associated with them. (T, F)
3. There are several ways to compile. They are:
  - a. Prompted
  - b. Non-prompted
  - c. Interactive
  - d. Batch
  - e. Remote System Explorer communications server
  - f. All of the above
4. The iSeries Table view displays the same information as the \_\_\_\_\_ view, with the ability to sort items and perform numerous actions.
  - a. Remote Systems

- b. Outline
  - c. Navigator
  - d. iSeries Error List
5. From the iSeries Table view you can:
    - a. List and sort libraries, objects, and members
    - b. Copy, rename, delete, edit, compile and run any item in the view from the pop-up menu
    - c. Transfer files from one system to another
    - d. All of the above
  6. You can run programs and commands from the Remote Systems view or the iSeries Table view in:
    - a. A Remote System Explorer communications server job
    - b. A batch job
    - c. An interactive job
    - d. All of the above
  7. Interactive programs require a 5250 emulator, so you need to first run a STRRSESVR connectionName command to associate the emulator with a particular connection in the Remote System Explorer Communications server. (T, F)

---

## More practice

Want more practice? Try this!

Given your experience in this exercise using the Compile command, and that you have your own source on your own iSeries system try these new tasks: Check out Compile (Prompt) Work with Compile commands. Assuming you receive errors in your source (add some errors into your source if you don't have any) when you compile your source, choose insert all error messages into the editor from the Error list. Use the Development Studio Client for iSeries online help to assist you in these tasks.

---

## Recap

Congratulations! You have completed this module. You should now understand:

- The purpose of the iSeries Error List
- The purpose of the iSeries Table view
- How to view and fix errors using the iSeries Error list
- How to save the fixed source
- How to run a remote compile from the Remote Systems view
- How to use the iSeries Table view to enter an iSeries command
- How to start an interactive connection to the iSeries
- How to run the program from the Remote Systems view

Now that you know how to compile RPG source from the Remote Systems view, you can move on to visually working with DDS source to design a screen for the RPG payroll program using the CODE Designer tool. Continue to the next module.

---

## Chapter 9. Designing screens

In this module, you will become familiar with the various aspects of the CODE Designer while modifying a display file to add a screen. You will step through each part of the CODE Designer tool interface and update some DDS as well. In the workbench, in the Remote System Explorer perspective you will use the connection that you used in the module before.

In order to design screens, there are several steps you will need to learn about and follow:

- Describe the features of the CODE Designer
- Start the CODE Designer
- Open, modify and save a display file
- Add a group
- Add a record
- Add, delete and modify various DDS fields
- Verify DDS
- Compile DDS

In order to accomplish these learning objectives, there are several steps that are involved, including:

- Opening a DDS member
- Viewing the DDS tree
- Selecting the DDS object
- Designing your DDS screen
- Creating groups from existing records
- Creating new screens
- Adding fields to the subfile record
- Switching between multiple records
- Adding field error handling
- Accessing field properties
- Adding new keywords
- Verifying your source changes
- Switching between design and editing your screen
- Compiling your source changes and closing CODE Designer

The exercises within this module must be completed in order. Start with the first exercise when you are ready to begin.

**Length of time:**

This module will take approximately 30 minutes to complete.

---

## Opening a DDS member

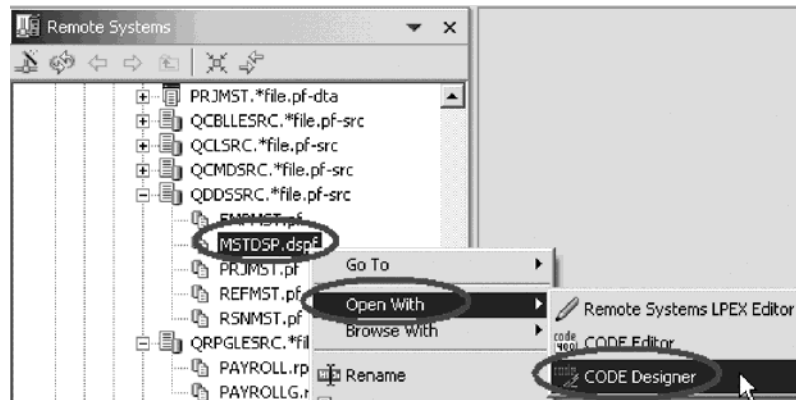
Using an editor to create and maintain DDS source for your display and printer files can be a frustrating and difficult task. What would be great is a graphical design tool that let's you design your screens and reports visually and then generate the DDS source for you. Well, that's exactly what CODE Designer does for you.

CODE Designer helps the novice DDS programmer create screens, reports and databases quickly and easily without worrying about the details of the DDS language, while at the same time letting the expert DDS programmer get access to all the features and power of the language.

**Note:** Make sure MSTDSP is closed from a previous LPEX Editor exercise.

To open a DDS file member:

1. Expand the Library List filter if is not already expanded.
2. Expand the QDDSSRC file in library RSELABXX.



3. Right-click MSTDSP member.
4. Click **Open With > CODE Designer** on the pop-up menu.

The member MSTDSP will be downloaded to the workstation and loaded into CODE Designer.

---

## Viewing the DDS tree

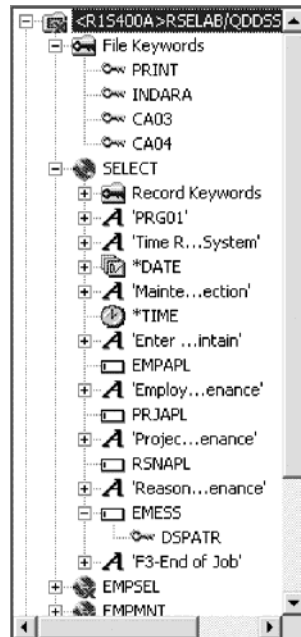
What you are looking at now is basically an explorer view of the DDS. The DDS Tree view on the left-hand side of the Designer displays the DDS source in its file, record, field, and keyword hierarchy. It is a familiar and intuitive way to see the overall structure of the DDS source and to navigate through it quickly. Don't worry if you're not a DDS expert, we will explain everything you need to know.

The DDS Tree is located on the left-hand side of the CODE Designer window. This view presents the loaded DDS source as a tree structure, showing the hierarchy of files, records, fields, help specifications, keys, and keywords in each selected DDS object. The DDS Tree shows groups of records, which represent the screens or reports you are designing, as peers of the file in the tree hierarchy.

In this view, you can create groups, and copy or move keys, keywords, fields, and records. If any DDS object contains an error, the icon representing it displays a red X.

To open selection in the DDS tree to see file-level keywords and the record SELECT:

1. Expand the <Servername>RSELABXX/QDDSSRC(MSTDSP) folder.
2. Expand the File Keywords folder.
3. Expand the SELECT record.
4. Expand the Record Keywords folder.
5. Expand the EMESS field.



The DDS tree now shows you a summary of the file-level keywords and of the record SELECT.

---

## Selecting the DDS object

In the upper right-hand side of the Designer is the Workbook with several different tabbed pages. The Workbook is the area of the CODE Designer where you design display files or printer files. You can view this notebook on the top right-hand side of the CODE Designer window. The top page is called Details and it provides a detailed view of the DDS objects selected by the DDS Tree. You can view this page in either details mode or list mode by selecting View > List.

In details mode, columns display information about each DDS object. You can use this mode if you want to know more about each DDS object (record, help specification, field, keyword, or keys).

In list mode, the DDS objects appear as columns of named icons. You can use this mode if you want to see more DDS objects within the page.

In the bottom right-hand side of the Designer is the Utility notebook. It lets you view specific elements of the DDS source such as errors and selected objects. The notebook contains several views: Selected DDS page, Error List page, Create keywords page, and Comments page. The Selected DDS page in the notebook shows the actual DDS source for the currently selected item.

To work with the DDS record SELECT:

1. In the DDS tree click the SELECT record.

The Details page lists all the fields in the record SELECT and summarizes some of their properties. The Selected DDS page shows the DDS for the SELECT record.

Field	Positi...	Len...	Type	Shift	Usa...	Sample
A 'PRG01'	2, 5	5	Text const...			PRG01
A 'Time R...System'	2, 30	21	Text const...			Time Reporting Syste
<input checked="" type="checkbox"/> *DATE	2, 70	6	Date const...			MM/DD/YY
A 'Mainte...ection'	3, 30	21	Text const...			Maintenance Selectic
<input checked="" type="checkbox"/> *TIME	3, 70	6	Time const...			HH:MM:SS
A 'Enter ...intain'	6, 14	54	Text const...			Enter an X beside the
<input type="checkbox"/> EMPAPL	9, 25	1	Alphanumeric	A - Alphanum...	Both	B
A 'Employ...enance'	9, 28	27	Text const...			Employee Master Mai
<input type="checkbox"/> PRJAPL	10, 25	1	Alphanumeric	A - Alphanum...	Both	B
A 'Projec...enance'	10, 28	26	Text const...			Project Master Mainte
<input type="checkbox"/> RSNAPL	11, 25	1	Alphanumeric	A - Alphanum...	Both	B

2. In the DDS tree click Record keywords immediately below SELECT.

The Details page shows the current record-level keywords. The Selected DDS page still shows the DDS for the SELECT record.

3. In the DDS tree click the EMESS field.

The Details page shows its field-level keywords. The Selected DDS page now shows the DDS for the EMESS field.

Create keywords	Web Settings	Comments	Error list
<pre> A      EMESS      50A  O 21 16 A 60                                     DSPATR(HI) </pre>			

Even this relatively small and simple DDS source member demonstrates how much easier it is to use the Designer to navigate through your DDS source. The syntax is being interpreted in intuitive graphical ways making it an ideal tool for learning DDS. But to get orders of magnitude improvement in your productivity what you really need is to work with your screens and reports in a WYSIWYG fashion, completely oblivious to the DDS required to make things appear the way they do. You need the Design Page.

---

## Designing the DDS screen

You will spend most of your time creating, updating, and designing your DDS screens and reports in the Design page. The Design pages allow you to design your screens or reports visually using an intuitive graphical user interface. The Design page shows the DDS source as it would appear on either a screen (for display files) or a printed page (for printer files). It allows you to design your application's screens or reports by laying out records and fields in a graphical user interface.

On the Design page, you can easily create, edit, resize, and move DDS objects graphically. Create new records, fields, and constants directly on the Design page by using the palette push buttons to the left of the Design area or from the pop-up



menus. The toolbar above the Design area allows quick access to many of the editing features as well as information about the currently selected object.

Click the MAIN\_MENU tab in the workbook.



In order to understand where MAIN\_MENU came from, we need to describe the concept of a group. A group is simply a collection of one or more DDS records that represent how a screen or report would be assembled at runtime. It allows you at development time to work with screens or reports as they would appear when they get assembled by your programs at runtime. To work with groups in CODE designer you need to tell CODE designer which record formats make up a screen. In this case we have done this for the screen you see in the design page. We have created a group MAIN\_Menu and CODE designer saved this information in the DDS source in comment lines. Any groups that you create are persisted as comment lines in the DDS so you can re-use these groups in subsequent CODE designer sessions.

The groups you create will appear in the tree view as well as in the workbook as a Design page tab for each group defined, to allow quick access to each group of records.

## Creating groups from existing records

If you are working with existing DDS, you will want to create groups that will correspond to how the records are being used. In this example you will create a group for the next screen, where the user selects which employee in the payroll database to maintain. The screen is made up of record format EMPSEL.

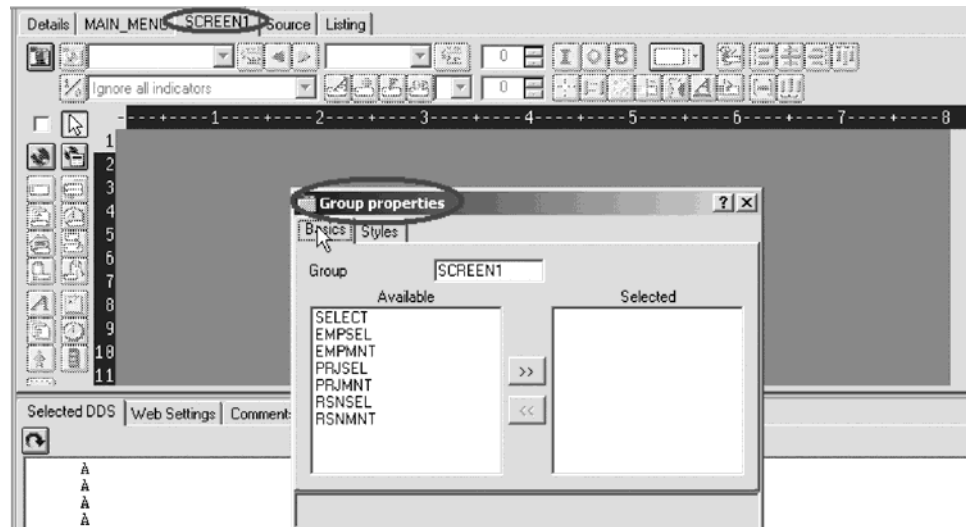
To create a new group:

1. Scroll to the bottom of the DDS tree and expand the MAIN\_MENU group.  
The SELECT record appears as the only record in this group.



2. Right-click the MAIN-MENU group.
3. Click **Insert group** on the pop-up menu.

A Group Properties notebook opens and a blank Design page for the group SCREEN1 also opens.



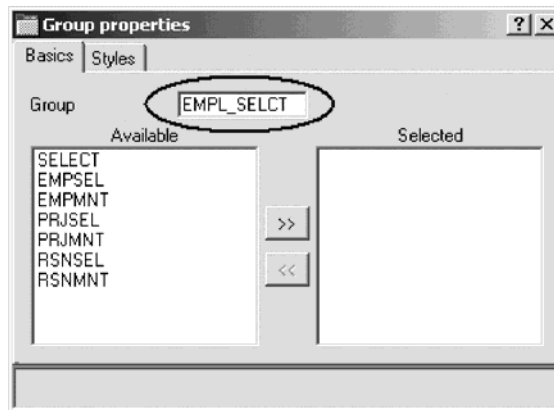
The Properties notebook lets you view and update the properties of the currently selected DDS object. You can open this notebook from any view, pop-up menu, or menu of the CODE Designer. The properties notebook is modeless. When you change an object's properties, the selected object changes immediately.

4. In the Group Properties notebook, select EMPSEL record from the **Available** list and click the **>>** button.

For simplicity this is the only record you will add for now. The Design page now shows us what the record EMPSEL looks like.



5. In the group field, type EMPL\_SELCT over SCREEN1 to rename the group.



6. Close the Group properties notebook. Click the X in the top right corner of the Group Properties notebook.

You have finished creating a group. You could now work in the Design page with the record formats contained in this group. Now you'll create a new record format.

It appears that this is one of those unusable applications where you have to know the employee number ahead of time instead of being able to browse what is in the database. What we really need is a subfile. But aren't those difficult to code, you ask? Not with CODE Designer.

---

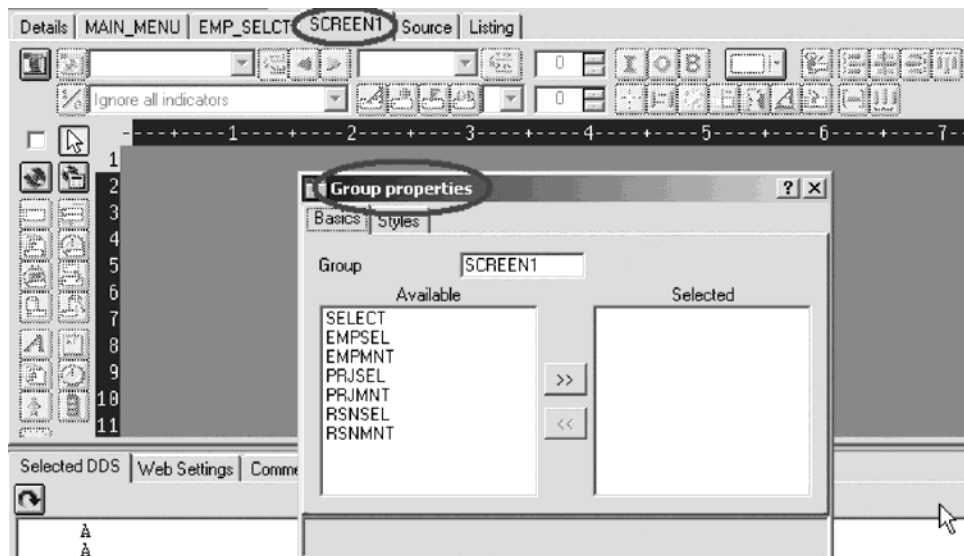
## Creating new screens

To create a new record screen on the Design page you need to create a group that will create an empty page you can work with.

To create a new group:

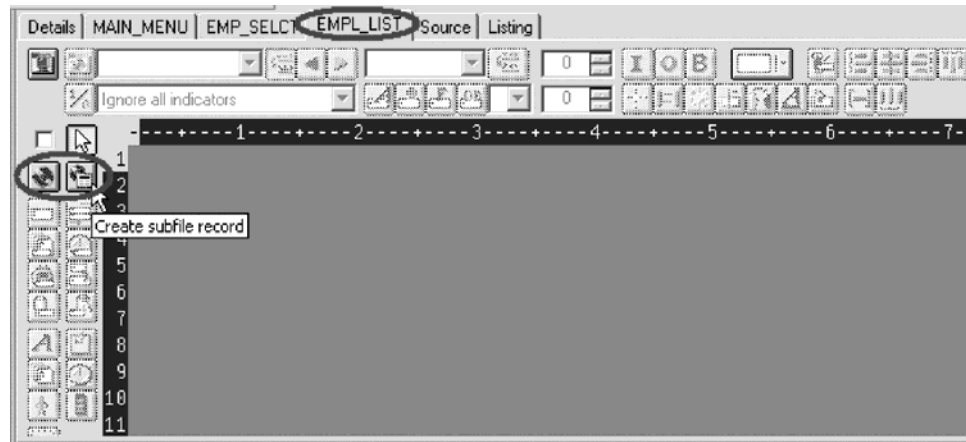
1. Right-click the new EMPL\_SELCT group in the DDS tree.
2. Click **Insert group** on the pop-up menu.

A Group Properties notebook appears and a blank Design page for the group SCREEN1 also appears.

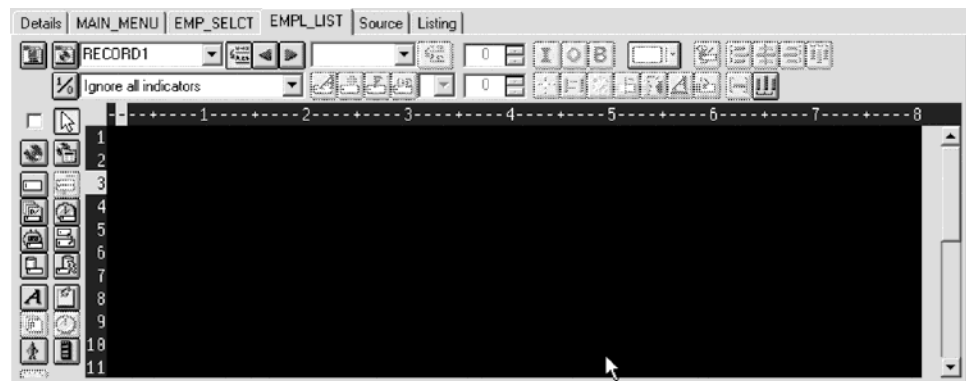


3. Rename the group to EMPL\_LIST and close the Group Properties notebook.

- You can create things on the Design page by selecting the appropriate tool from the palette on the left-hand side and then click on the Design page where you want it to be created. Right now, most things are disabled in the palette because there is no record in which to create fields. The only two tools available are **Create standard record** and **Create subfile record**. If you leave the mouse over a button for a second or two, flyover help will appear describing the indicated button.




- Click the **Create subfile** record button and then click in the dark gray area. A subfile and a subfile control record pair are created.



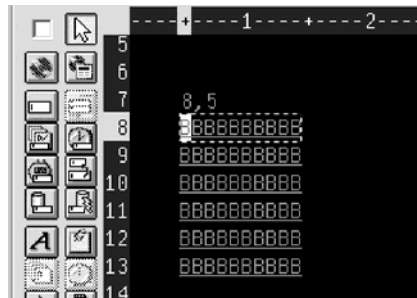
## Adding fields to the subfile record

Now you add some columns to the subfile using the Design page. The subfile should be positioned on row eight. You use the cursor to specify the location of the part you want to put on the screen, in this case your subfile.

To add fields on the subfile:

- Click **Create named field** button  and then **click** somewhere on **row 8**. Six fields appear in a vertical column. This is because the subfile you created, currently specified a SFLPAGE (visible list size) of six.
- Click the top field and **hold** the mouse button down and **move** it to **row 8, column 5**.

Note the current row and column appear just above the field as you move it.




3. Move the cursor over to the **right edge** of the field. It turns into a double-headed arrow. Hold down the mouse button and **move** it to the **left**. The size of the field will be reduced. The current size will appear just above the field. When the size is **3**, let go of the mouse.

The toolbar at the top of the Design page is a very convenient place to monitor and manipulate the currently selected parts.

4. Rename the record from RECORD1 to EMPLSTSFL and the field from FIELD1 to OPCODE by typing over the text in each list.



5. Click the Color palette  button and select pink to change the color of the field.

To create an additional column:


1. Click the **I** button to change the usage of the field to input.  
Now you will create an additional column in the subfile.
2. Position the cursor at **row 8, column 9**.

**Note:** The bottom right of the CODE Designer window shows the current cursor position 

If you can't see the field with the cursor position on your screen, click the **Maximize** button in the top right corner of the screen. You can use the cursor keys or the mouse to move the cursor.

3. If you are creating a long field with an exact length, the SDA syntax can be easier. Type:  
+0(30)

and then press the **back arrow** (not Backspace!) to select the text you created. Notice from the Selected DDS page that you have created a text constant containing +0(30).

4. Click **Convert string to field**  button on the toolbar or press F11 to convert the SDA syntax into a character output field of length 30.



5. Rename the new field to ENAME using the toolbar. This will show the name of the employee.

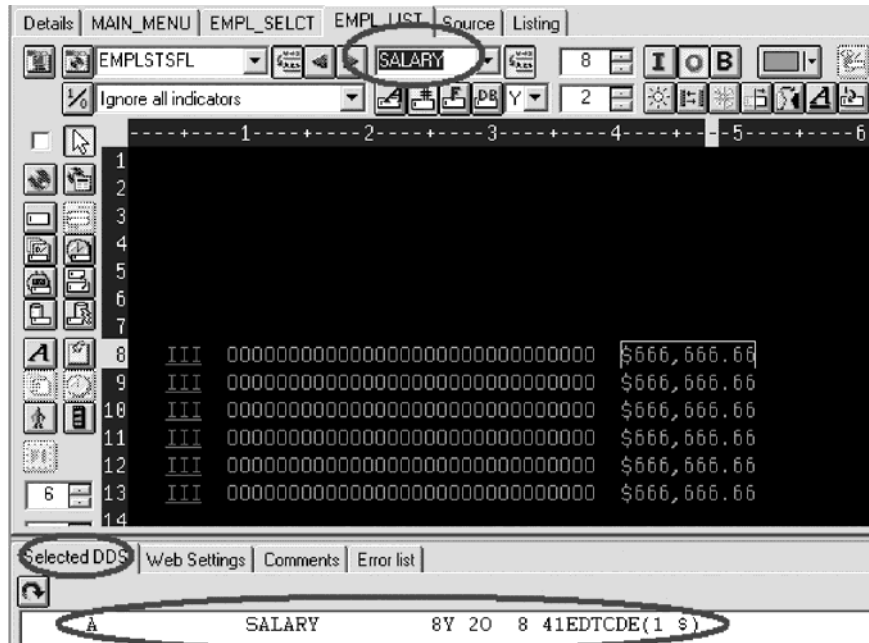


6. Position the cursor to 8, 41.
7. Now you will add a field for the employee's salary. Type \$666,666.66

and then press the back arrow.

Now wouldn't it be better if we could just tell the Designer what we wanted the number to look like and then have Designer generate all the cryptic EDTCDEs to make it happen?

8. Press F11 to convert this field into an output numeric field with comma delimiters, two decimal positions, a currency symbol and no sign. Look at the Selected DDS page to see what was generated for you. Impressive!



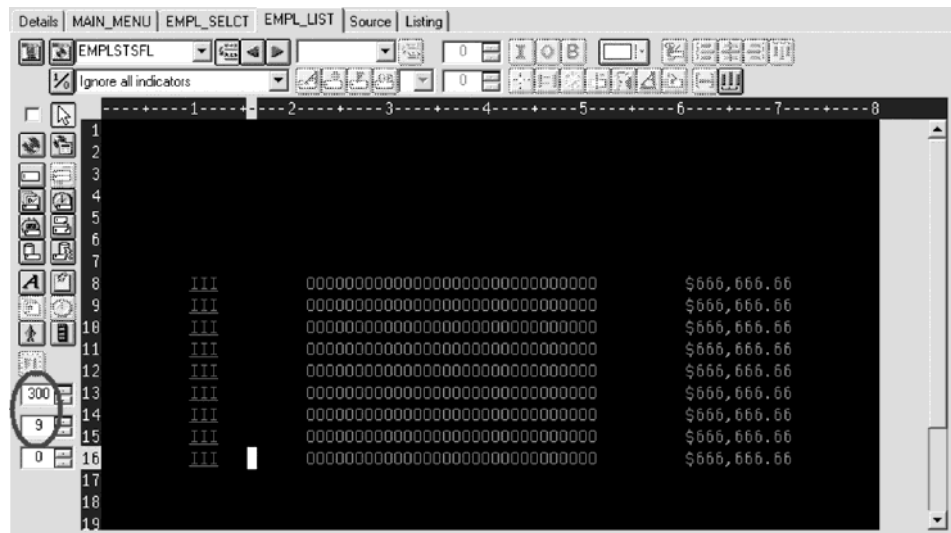
9. Rename the field to SALARY and change its color to yellow, using the toolbar.
10. The subfile seems compacted to the left. It would be better to space it out evenly. Just select the field and click the space horizontally button on the far right side of the toolbar. The other alignment buttons will align fields, left, right, center and top.



Just below the palette there are three spin buttons. The top one, **Subfile size**, specifies the total number of entries in the list that will be filled in by the application. The second one, **Subfile page size**, is how many entries appear on the screen.

11. In the **Subfile size** field, type 300.

- In the **Subfile page size** field, type 9.




The Design page is updated accordingly.

## Switching between multiple records

Now let's fix up the Subfile control record. The group you created contains 2 records. You can verify this by looking at the record list in the toolbar:

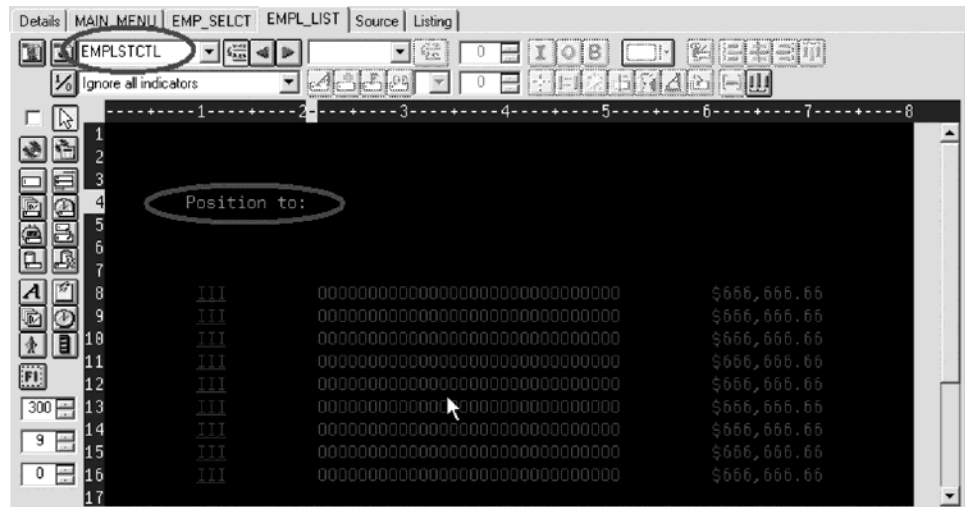





- Change the current record by selecting **RECORD1CTL** from the record list or click next record  or press **Alt+End**.

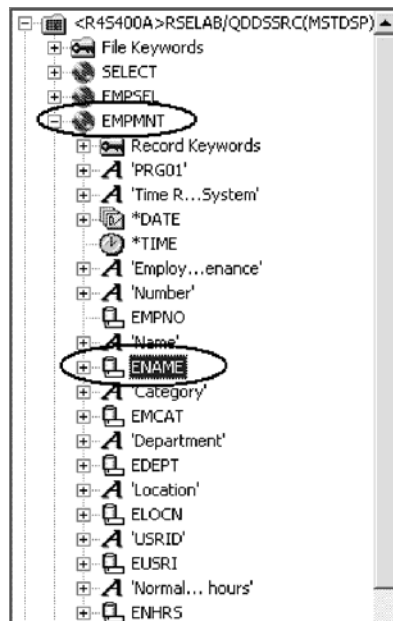
The fields in the subfile still appear so that column heading can be lined up, but they appear at half-intensity so that they can be distinguished from the fields of the current record.

- Rename the record to **EMPLSTCTL** using the toolbar. Let's provide a 'position to' entry in the subfile control header.
- Position the cursor at **4, 9** and type:

Position to:



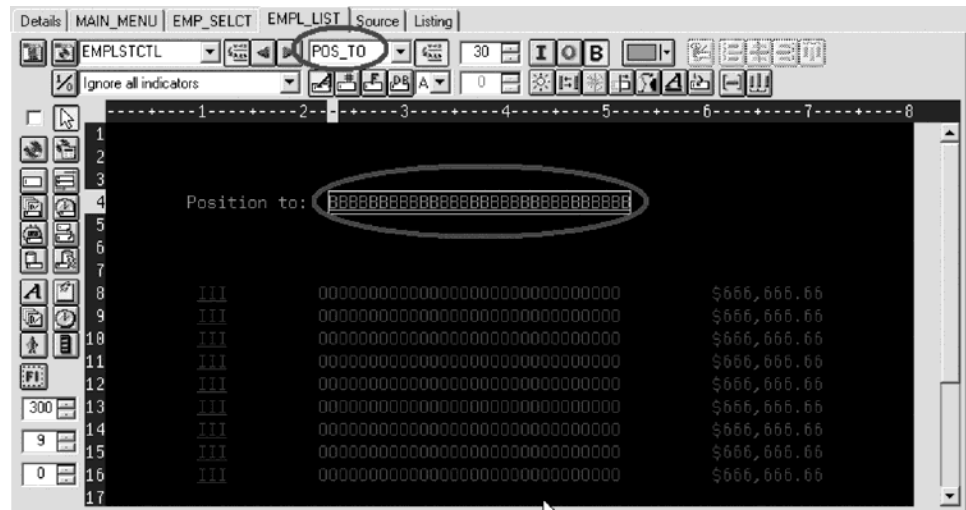
4. Now you need an employee name field.  
You could create a named field with the right characteristics like you did in the subfile, or you could create a source reference using the Create source reference field  button in the palette or you could reference the original database field using one of the Create database reference field  Create database reference field(s) by selection  buttons. But there is an even simpler way. Use copy and paste!
5. In the DDS tree expand the EMPMNT record.
6. Click the ENAME field and press **Ctrl+C**.



- (The pop-up menu or Edit menu shows the Copy menu item as well).
7. Position the cursor to 4, 23 and press **Ctrl+V**. Now that was easy!



- Rename the field to POS\_T0.




## Adding field error handling

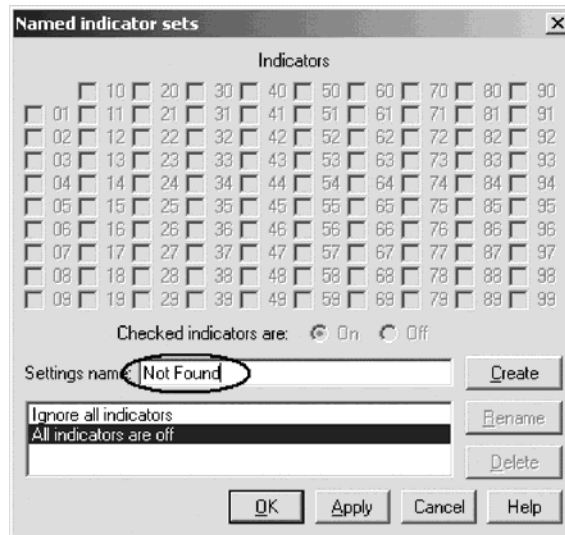
Let's put in some error handling for the 'position to employee name' field. If the employee name is not found in the database, the program will turn on indicator 60. An indicator is a two digit variable (01, 02, 03, ... 98, 99) to signal whether or not certain events have occurred during processing. Using indicators enables program events to control screen displays. These indicators are set 'on' or 'off' in one part of the program while their status is referenced in another part of the program. By setting the indicators 'on' or 'off', you can associate fields with indicators to control whether or not a field will appear on the screen.

The screen should turn the field red, reverse image and position the cursor to it. Now wouldn't it be better if we could work something higher level and easier to remember than some arbitrary number from 1 to 99.

To set indicators:

- Click the Changed named indicator sets  button on the Design page toolbar (or press F7).  
The Named indicator sets window opens.
- In the **Settings name** field, type: Not Found.

- Click **Create**.



- Select the check box next to **60** and click **OK**.

The Not found indicator set is now in effect. The Design area is shown as if indicator 60 was on and all other indicators were off. The Design page toolbar shows the current indicator set in the Select named indicator set list on the bottom left.

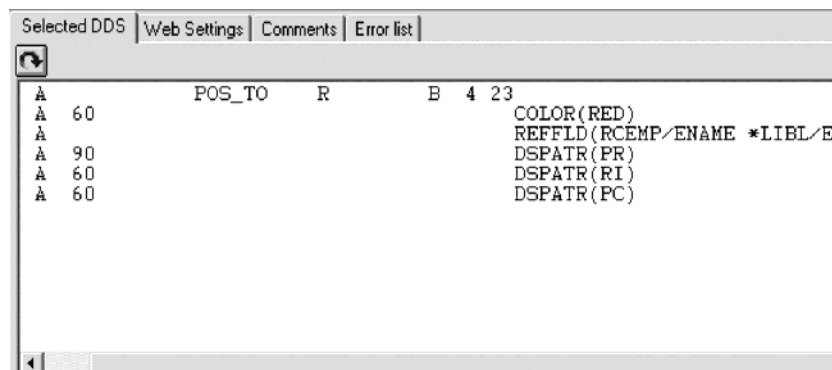


- Now select the **POS\_TO** field.
- On the toolbar, select the color **red** and the display attributes **reverse image** and **position cursor**. (The set of toolbar buttons representing the current display attributes is found just below the color button).

The toolbar should look as follows:

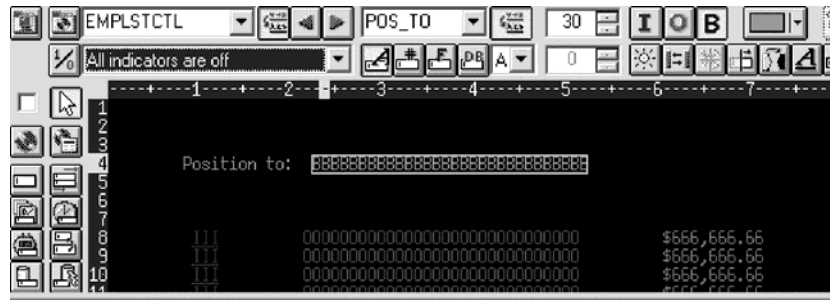


- Examine the DDS generated in the Selected DDS page.



Notice that all the new keywords were created with a condition of 60. (The DSPATR(PR) was pasted with the field originally).

- Now let's try it out! From the **Select named indicator sets** list, select **All indicators are off**.



Wow! This really works!

- From the **Select named indicator sets** list, select **Not found**.  
The field appears red and reverse imaged.

## Accessing field properties

Second to the direct manipulation and the toolbar on the Design page, the easiest and quickest ways of getting access to the properties of a field, record, or entire file is a Properties notebook.

The Properties notebook lets you view and update the properties of the currently selected DDS object. You can open this notebook from any view, pop-up menu, or menu of the CODE Designer.

The Properties notebook is modeless. When you change an object's properties, the selected object changes immediately.

You can get to a Properties notebook from the **Selected** menu, by pressing **F4**, or double-click on anything in the DDS tree or the Details page or Design page.

To open the Properties notebook:

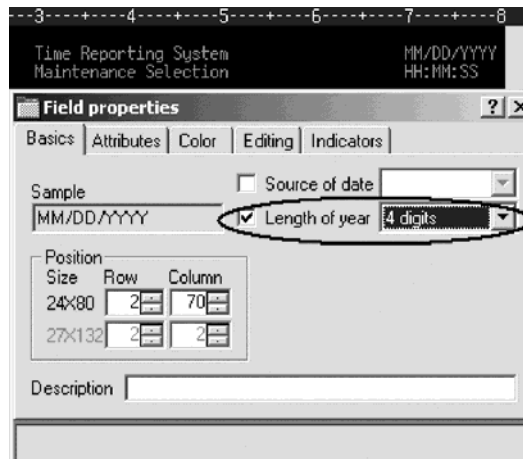
- In the DDS tree, click the record **SELECT** and press **F4** to see the Record properties.



As you select different items, the Properties notebook will continuously update itself to show you the properties of the selected item.

- Click the **\*DATE** field in the **SELECT** record. (You may have to move the Properties notebook out of the way.) This field has a different set of pages describing its properties.
- Change the year from 2 to 4 digits. Select the **Length of year** check box.

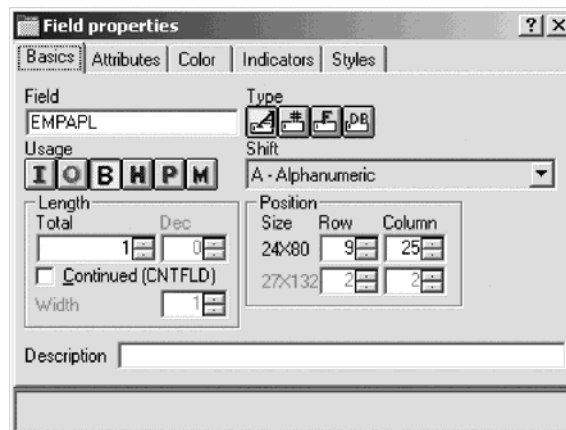
4. Select 4 digits from the list.



Notice how the sample is updated on the Properties notebook.

5. To test the Design page, click the **MAIN\_MENU** tab in the workbook and look at the upper right corner of the screen. The year now has 4 digits.
6. Click the EMPAPL field in the SELECT record. On the Field properties notebook click the **Basics** tab.

On this page you can change the field's name, usage, length, type, and screen position. The other pages give you quick access to other properties for this field.



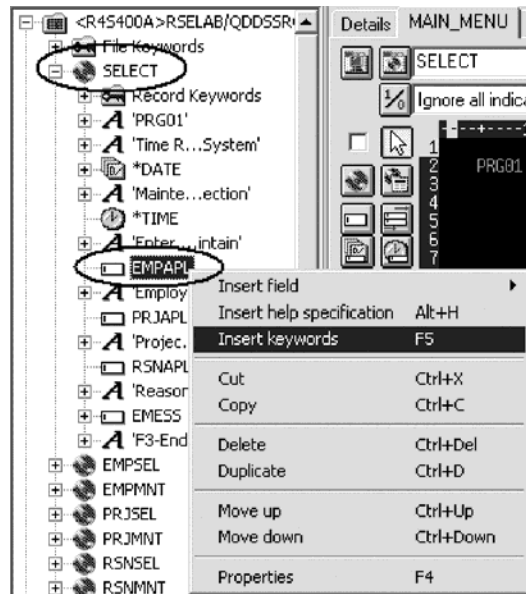
## Adding new keywords

CODE Designer helps you manage the visual aspects of your displays and reports. But you also need to access the full power of DDS. You need to access keywords.

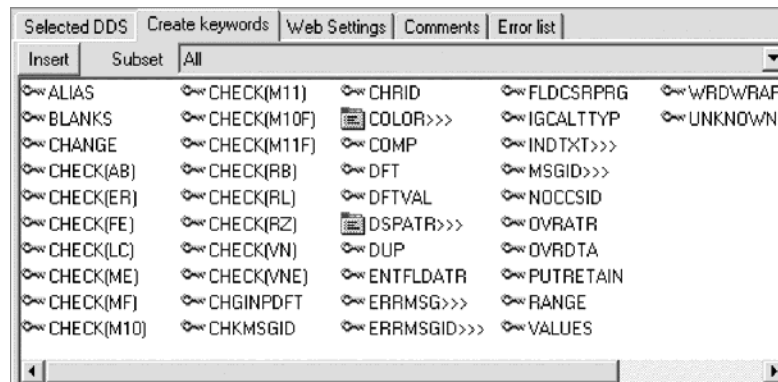
To add keywords:


1. Click EMPAPL field in the DDS tree.

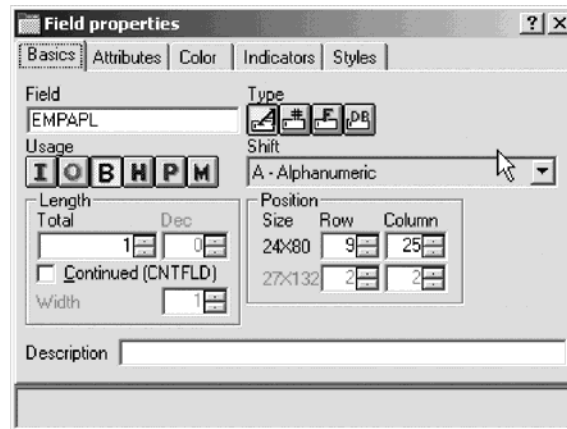
2. Press F5 or right-click and click **Insert keywords** on the pop-up menu.




You see the Details page for the EMPAPL field and the **Create keywords** tab is selected in the Utilities notebook. This page shows you the subset of keywords that are allowed for the selected file, record or field and it takes into account the field's type, usage, shift and what record it is in. It is very powerful to know exactly what your options are. This information cannot be quickly ascertained from the Reference manual.



3. With the EMPAPL Properties notebook at the **Basics** page, click the numeric field  button to change the field to numeric type.




Notice that the list of keywords in the Create keywords page has changed.

4. Click the  button to change the field back to alphanumeric.  
Notice that the list of keywords in the Create keywords page has changed.
5. Click the **ALIAS** keyword and press **F1**.  
The DDS Reference help for the ALIAS keyword appears.  
Tip: CODE Designer has lots of on-line help. Press F1 anywhere you want to see help for an item, icon or notebook. You will get help relevant to what you are currently trying to do. From the Help menu you can get quick access to the DDS Language Reference as well as several other useful sources of information.
6. Minimize the Help window.
7. Double-click the **INDTXT** keyword. (You may have to scroll to the right to find it).  
The keyword is created with default values which can be changed when you want.
8. Double-click the **INDTXT** keyword again.  
The keyword is created with the same default values creating a conflict.
9. Close the Keyword Properties dialog.

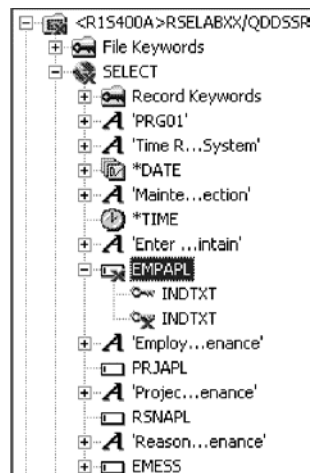
## Verifying the source changes

You have just added a new record and some new fields to our DDS source. Everything that the CODE Designer adds to your DDS source is certain to have the correct syntax. Now you need to make sure that there are no semantic errors. You just introduced one in the last section by creating two **INDTXT** keywords describing the same indicator.

To verify your source:

1. Click **Tools > Verify file** (or click the verify  button on the main toolbar) on the CODE Designer menu.  
The DDS source is checked using the same verifier that the CODE Editor or LPEX Editor uses. A message appears on the status line at the bottom of the Designer stating that the verify process completed with errors.

- In the DDS tree, there is a trail of red x's leading to the problem.



The file icon has a red x, as does the SELECT record, the EMPAPL field and finally the second INDTXT keyword.

- Click the **MAIN\_MENU** tab in the workbook.

The EMPAPL field is highlighted in red.

- Click the **Listing** tab in the workbook.

This page shows you the listing generated by the most recent program verify. A warning message is buried somewhere in the listing but it's not easy to find.


- If there are problems, they will show up in the Error list page in the Utility notebook. It behaves exactly like the Error list in the CODE Editor or LPEX Editor. Click the **Error list** tab.
- Double-click the warning **DDS7861** in the Error list. (Press **F1** to see detailed help on the message).

The Source page appears and the cursor is placed exactly where the error is in the source. The Source page is a tokenized read-only view of the current state of the DDS source. Read-only? Wouldn't it be great if you could just clear the error right here. There are some things that are just plain faster in the editor and many others that are faster in the visual environment. It would be great to switch between the two modes at the push of a button. Well, let's just do that.

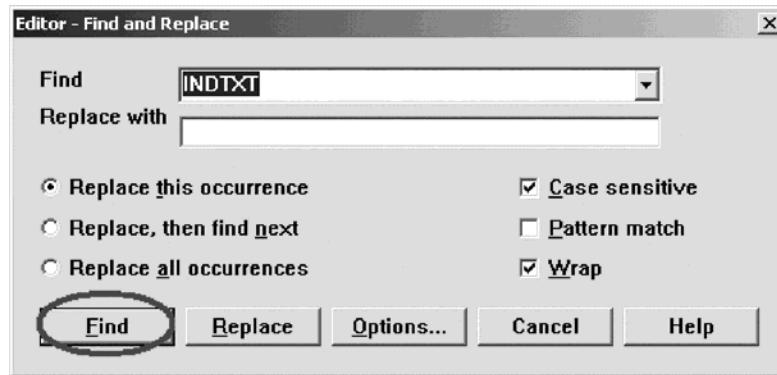
---

## Switching between designing and editing the screen

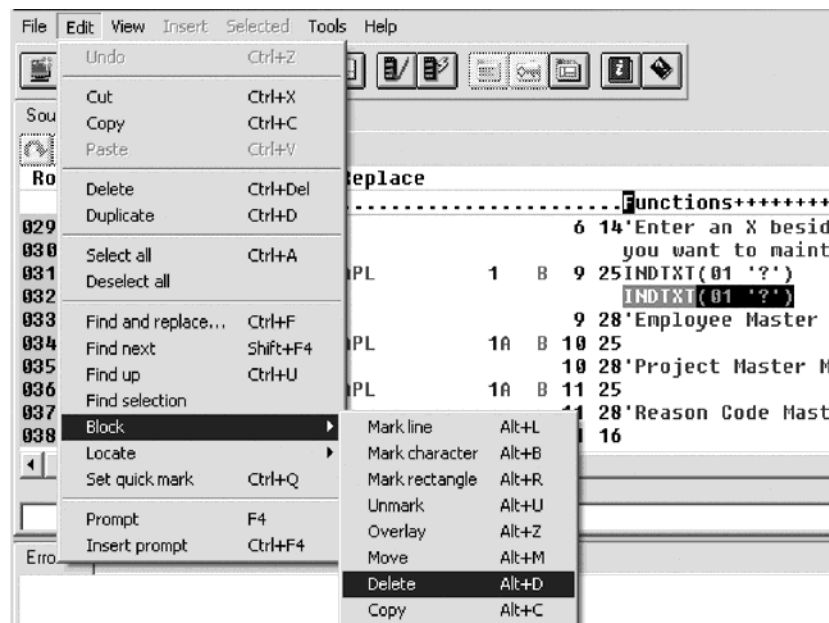
To switch between the Design mode and the Edit mode:

- Click the Edit DDS source  button or select **File > Edit DDS Source** from the Editor menu.  
You now have access to the full power of the editor.
- Explore the **Edit** and **View** menu items.
- Press **Ctrl-F** to open the Find/Replace window.

- In the **Find** field, type **INDTXT** and click **Find**.




- Press **Ctrl-N** to find the next occurrence.
- Delete the second **INDTXT** line.



## Compiling your source changes and closing the Designer

Now you will compile the source on the iSeries just as you did in the Remote Systems LPEX Editor and then close the designer.

To compile your source:

- Click **File > Save** from the Designer menu to save your source to the iSeries.
- Click **Tools > Compile** from the Designer menu and then click **No prompt** (or click the compile  button on the CODE Designer toolbar).
- A message indicates when the compile is complete. Click **OK** in the Message window. If you run the **PAYROLL** program, you will see the 4 digit year change you made to the opening screen of the program.

To close the Designer:

- Click **File > Exit** from the Designer menu.



---

## Checkpoint

Complete the checkpoint below to determine if you are ready to move on to the next module.

1. There is a graphical design tool that let's you design your screens and reports visually and then generate the DDS source for you. This tool is named:
  - a. Page Designer
  - b. CODE Designer
  - c. Screen Design Aid
  - d. None of the above
2. Match the item with its correct description.
  - a. Workbook
  - b. Details page
  - c. Utility notebook
  - d. Design page
  - a. View specific elements of the DDS source such as errors and selected objects
  - b. Lists the contents of the currently selected item in the DDS tree
  - c. Contains several different tabbed pages
  - d. For visually designing screens and reports
3. The Utility notebook contains:
  - a. Selected DDS page
  - b. Error List page
  - c. Create keywords page
  - d. Comments page
  - e. All of the above
4. On the Design page, you can easily:
  - a. Create DDS objects
  - b. Edit DDS objects
  - c. Resize DDS objects
  - d. Move DDS objects
  - e. Create records
  - f. Create fields
  - g. Create constants
  - h. All of the above
5. Grouping records together allows you to work on one record while still seeing the related records in the background. (T, F)
6. The Properties notebook is modeless. When you change an object's properties, the selected object changes immediately. (T, F)
7. The DDS source is checked using the same verifier that the LPEX Editor or CODE Editor uses. (T, F)
8. You can switch between design mode and edit mode in CODE Designer. (T, F)
9. You can compile sources on the iSeries just as you can in the LPEX Editor. (T, F)

---

## More practice

Want more practice? Try this!

Given your experience in this exercise using the CODE Designer, try creating a printer report. You can use the physical file specification REFMST in member QDDSSRC in library RSELABXX. Take time to explore the fields and information for this physical file. You may want to refer back to this information as you create your report. When you are familiar with the file REFMST, you can create your printer file. Use the Development Studio Client for iSeries online help to assist you in this task.

---

## Recap

Congratulations! You have completed this module. You should now understand:

- The purpose of a workbook, details page, utility notebook, design page, properties notebook, group and indicator
- How to modify a display file to add a screen
- How to add groups, records and fields
- How to save your DDS source
- How to verify your DDS source changes
- How to compile your DDS source changes

Now that you know how to design screens, you can learn how to debug and test the payroll application on an iSeries system. Continue with the next module.

---

## Chapter 10. Debugging a program

In this module, you will review the Debugger features. You then learn how to start the debugger, set breakpoints, monitor variables, run and step into a program, view the call stack in the Debug view, remove a breakpoint, add a storage monitor, and set watch breakpoints and all from the Debug perspective.

In order to debug a program, there are several steps you will need to learn about and follow:

- Describe some of the features of the Debugger
- Start a Debug session
- Add and delete line breakpoints
- Display a variable
- Change a variable
- View the call stack
- Run, step and step into a program
- Switch between source and listing view
- Add a storage monitor
- Add a watch breakpoint

In order to accomplish these learning objectives, there are several steps that are involved, including:

- Debugging iSeries programs from the Debug perspective
- Starting the Integrated Debugger
- Setting breakpoints
- Monitoring variables
- Stepping into a program
- Listing call stack entries
- Setting breakpoints in PAYROLLG or PAYROLLD
- Removing a breakpoint in PAYROLLG or PAYROLLD
- Monitoring variables in PAYROLLG or PAYROLLD
- Adding a storage monitor
- Setting watch breakpoints
- Closing the debug session

The exercises within this module must be completed in order. Start with the first exercise when you are ready to begin.

**Length of time:**

This module will take approximately 20 minutes to complete.

---

### Introducing the iSeries Integrated Debugger

The Integrated Debugger is a source-level debugger that enables you to debug and test an application that is running on an iSeries system. It provides a functionally rich interactive graphical interface that allows you to:

- View source code or compiler listings, while the program is running on the iSeries system.
- Set, change, delete, enable and disable line breakpoints in the application program. You can easily manage all your breakpoints using the Breakpoints view.
- Set watch breakpoints to make the program stop whenever a specified variable changes.
- View the call stack of your program in the Debug view. As you debug, the call stack gets updated dynamically. You can view the source of any debug program by clicking on its call stack entry.
- Step through your code one line at a time.
- Step into or step over program calls and ILE procedure calls.
- Display a variable and its value in the Monitors view. The value can easily be changed to see the effect on the program's execution.
- Locate procedure calls in a large program quickly and easily using the Modules/Programs view.
- Debug multithreaded applications, maintaining separate stacks for each thread with the ability to enable and disable any individual thread.
- Load source from the workstation instead of the iSeries – useful if you don't want the source code on a production machine.
- Debug client/server and distributed applications.

The Debugger supports RPG/400<sup>®</sup> and ILE RPG, COBOL and ILE COBOL, C, C++ and CL.

In the following exercise you will be given the opportunity to learn about some of the basic features of the Debugger. For the purpose of this exercise you will debug an ILE RPG/400 program. Don't worry if you don't know RPG.

---

## Starting the integrated debugger

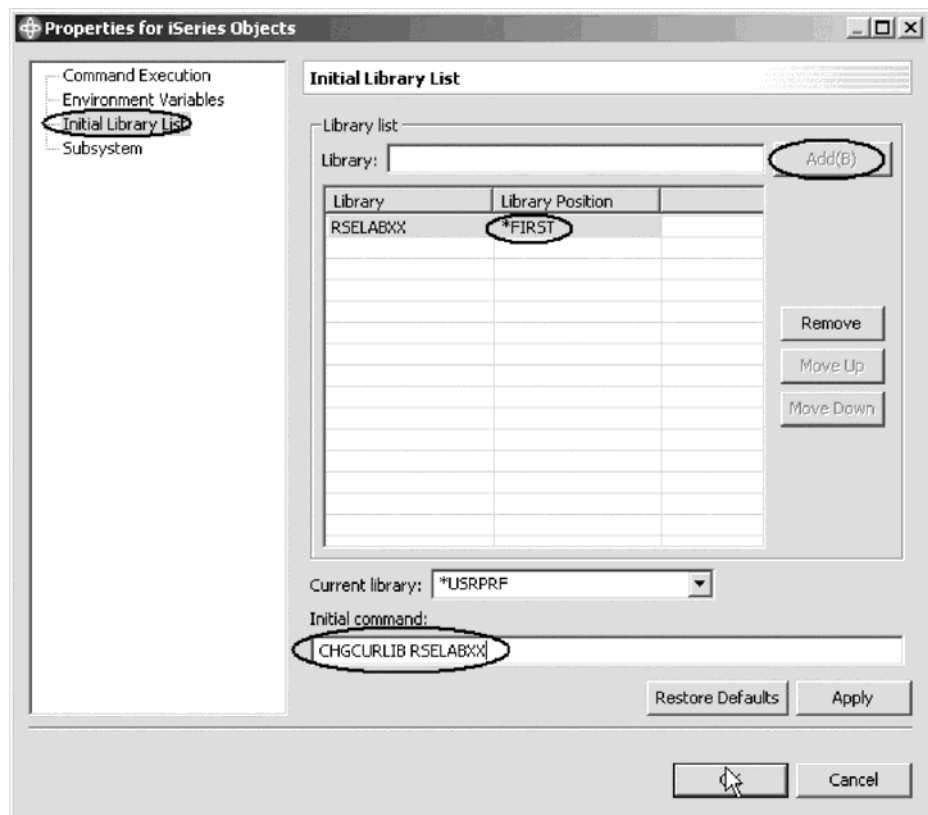
You will be working with the ILE RPG program PAYROLLG.

**Note:** For RPG programmers: PAYROLLG is the same RPG program as PAYROLL but without compile errors. You are using it instead of PAYROLL in this exercise, to accommodate anyone who decided to skip right to this exercise without completing the editor exercise.

**Note:** For COBOL programmers: PAYROLLD is the same COBOL program as PAYROLLC but without compile errors. You are using it instead of PAYROLLC in this exercise, to accommodate anyone who decided to skip right to this exercise without completing the editor exercise.

**Note:** Make sure that your user ID has been setup, so that your library has been added to the library list automatically. You can use the properties for iSeries Objects dialog to set connection information such as adding a library to a library list and changing the current library. You can right-click on iSeries Objects then click Properties to see this dialog. You will need this setup to

run jobs interactively from your workstation.

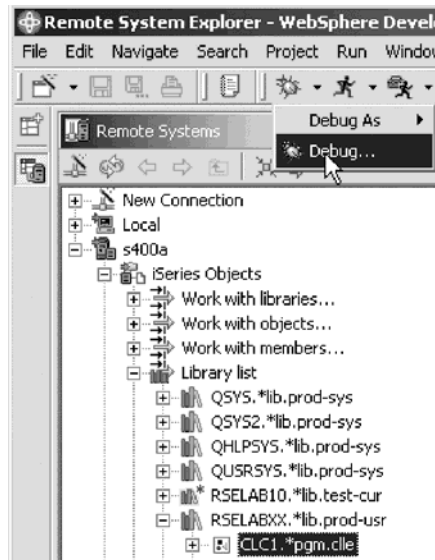
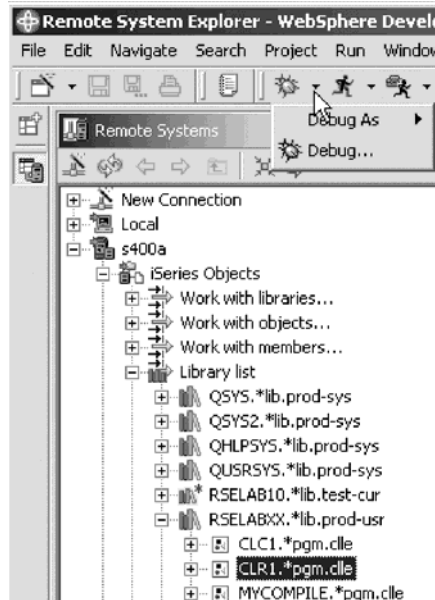



You can start the Debugger in several ways: directly from the pop-up menu of a program or service program in the Remote Systems view, or from the Launch Configurations window. Starting directly from the Remote Systems view doesn't allow you to specify parameters to be passed to the program. The Launch Configurations window allows you to modify how the program is invoked and to specify parameters.

To make the exercise interesting you will use CL program CLR1 to call PAYROLLG and you will pass one parameter to CLR1 or you will use CL program CLC1 to call PAYROLLD and you will pass one parameter to CLC1. This means you will use the Launch Configurations window.

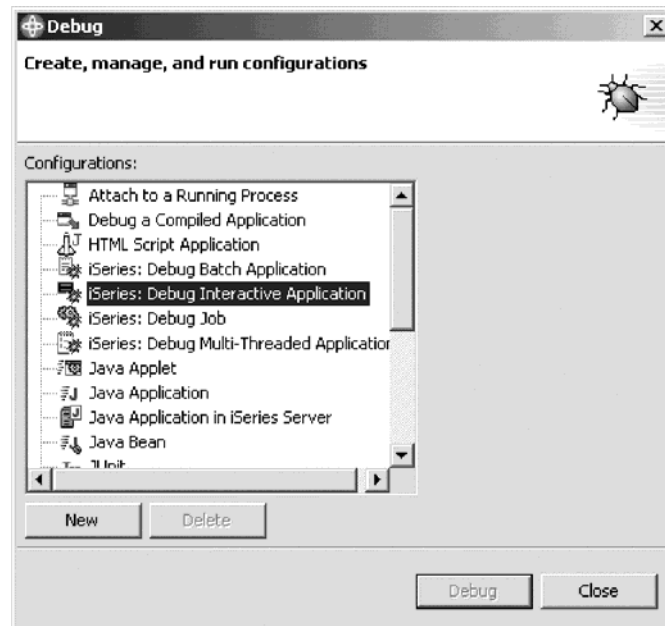
To start the debugger:

1. In the Remote Systems view expand the Library list filter, if it isn't expanded already.

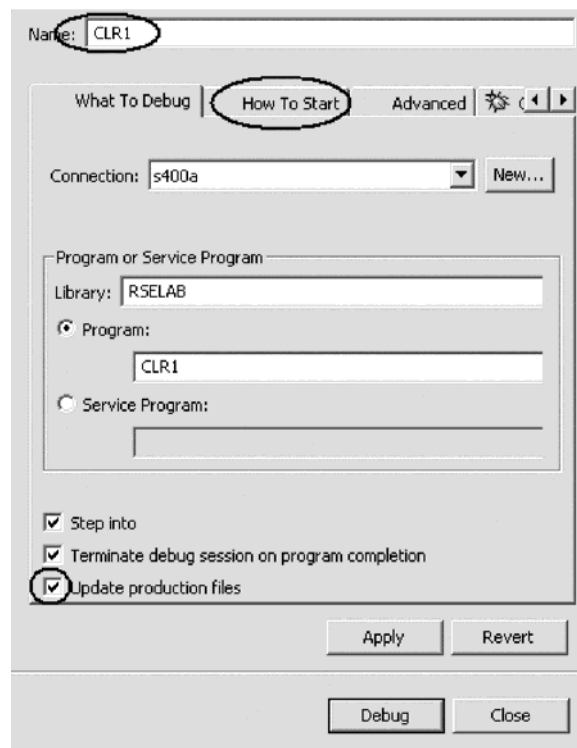


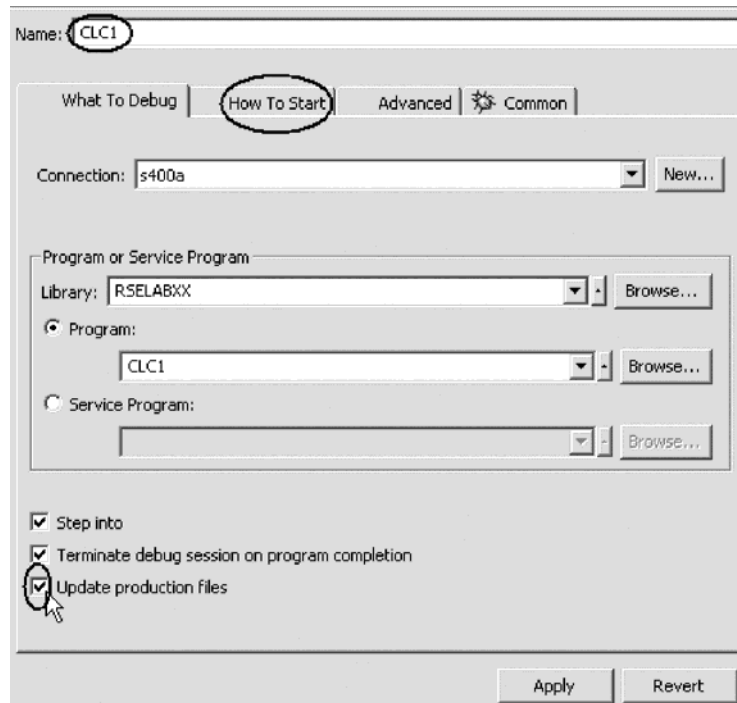
2. Expand library RSELABXX, if it isn't expanded already.
  3. Select program CLR1 or CLC1.
  4. Click the arrow beside the **DEBUG** icon  on the workbench toolbar.
  5. Select **Debug** from the list.
- The Debug Launch Configurations window opens.

6. Select **iSeries: Debug Interactive Application** under the **Configurations** list.

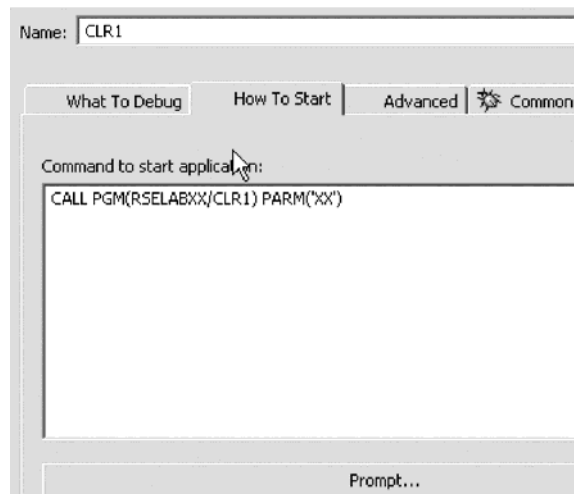


7. Click **New**.  
The right pane of the Debug Launch Configurations window opens.

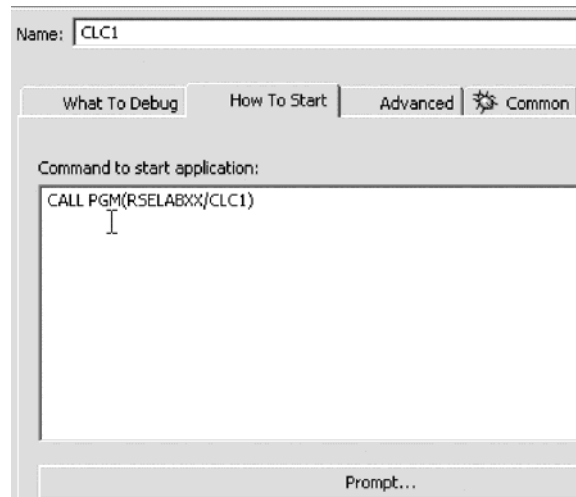




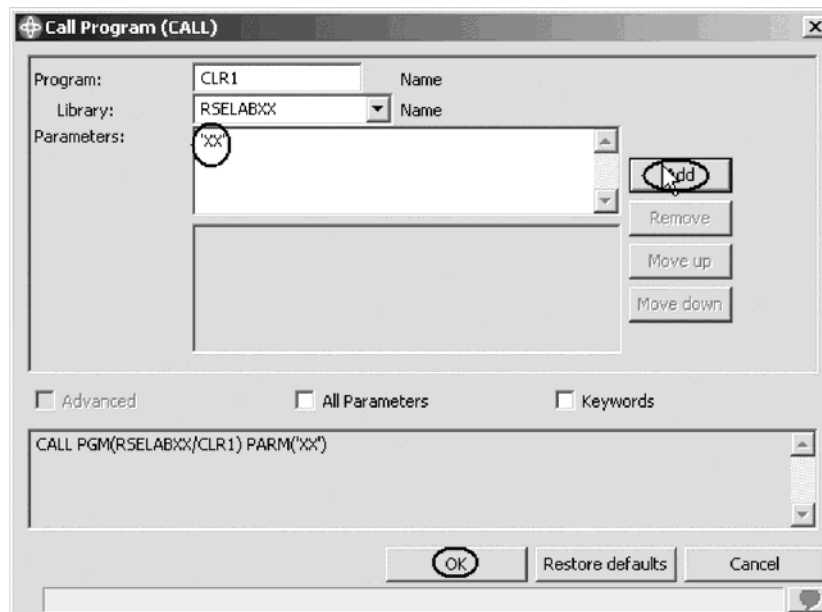
8. In the **Name** field, type the program name CLR1 or CLC1.
9. Select the **Update production files** check box.
10. Click the **How To Start** tab.  
The How To Start page opens.

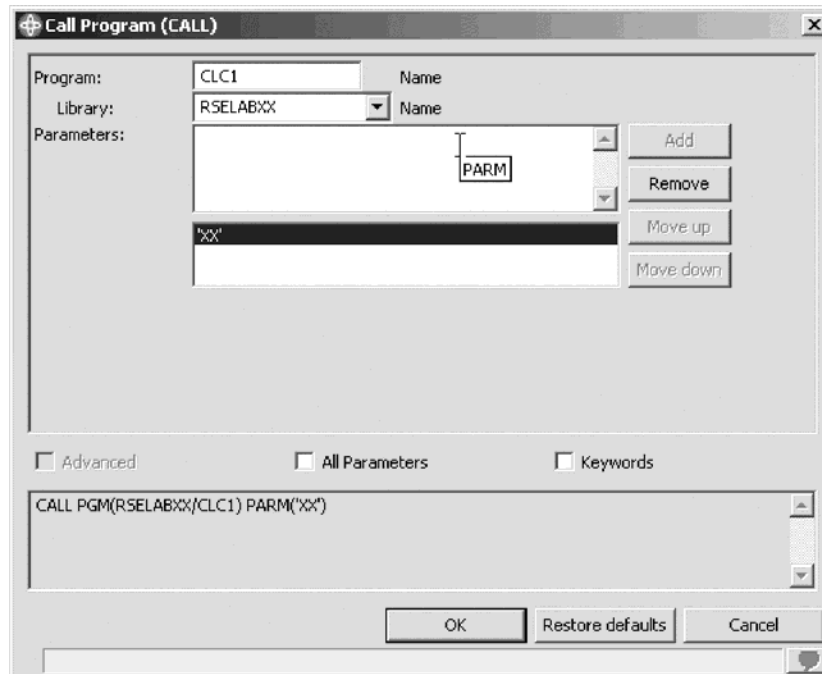






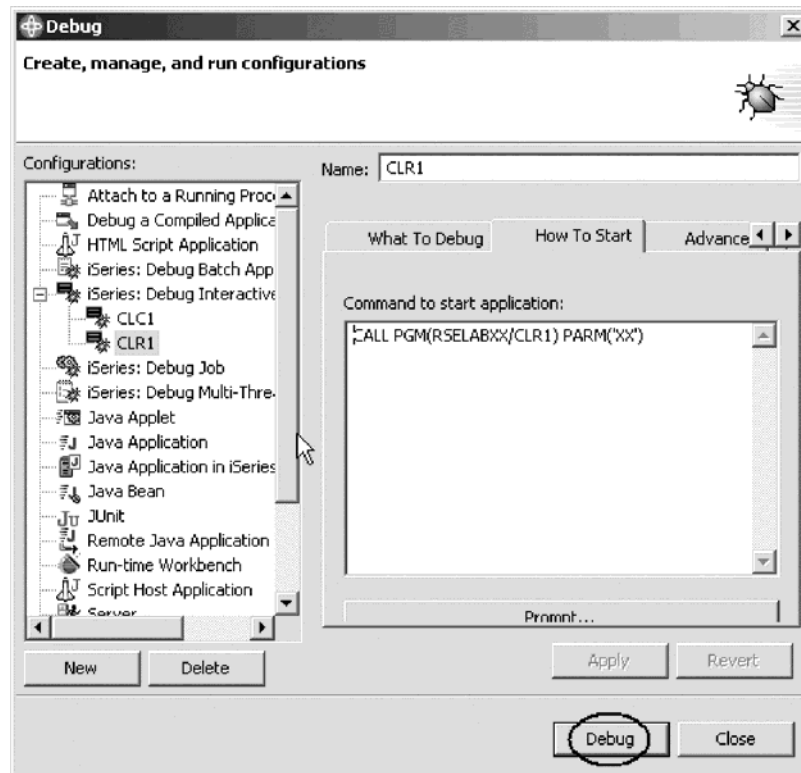
11. Click **Prompt**.  
The Call Program (CALL) window opens.

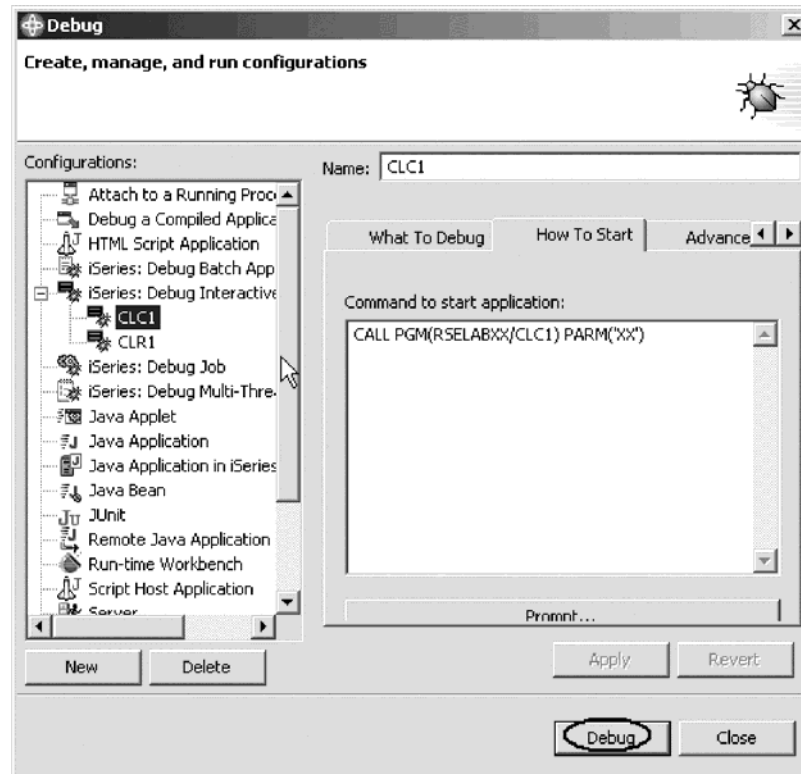




12. In the **Parameters** field, type 'XX'. Click **Add**.  
The parameter value will appear in the lower list.
13. Click **OK**.

The complete start command for the program appears:

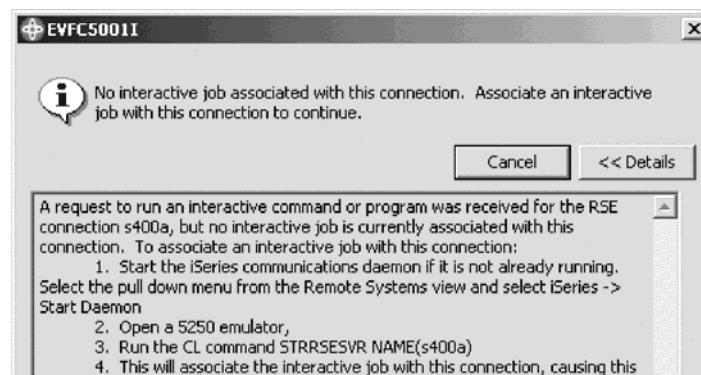




14. Click **Debug**.

The Debug perspective opens.

If not, you may see this error message:



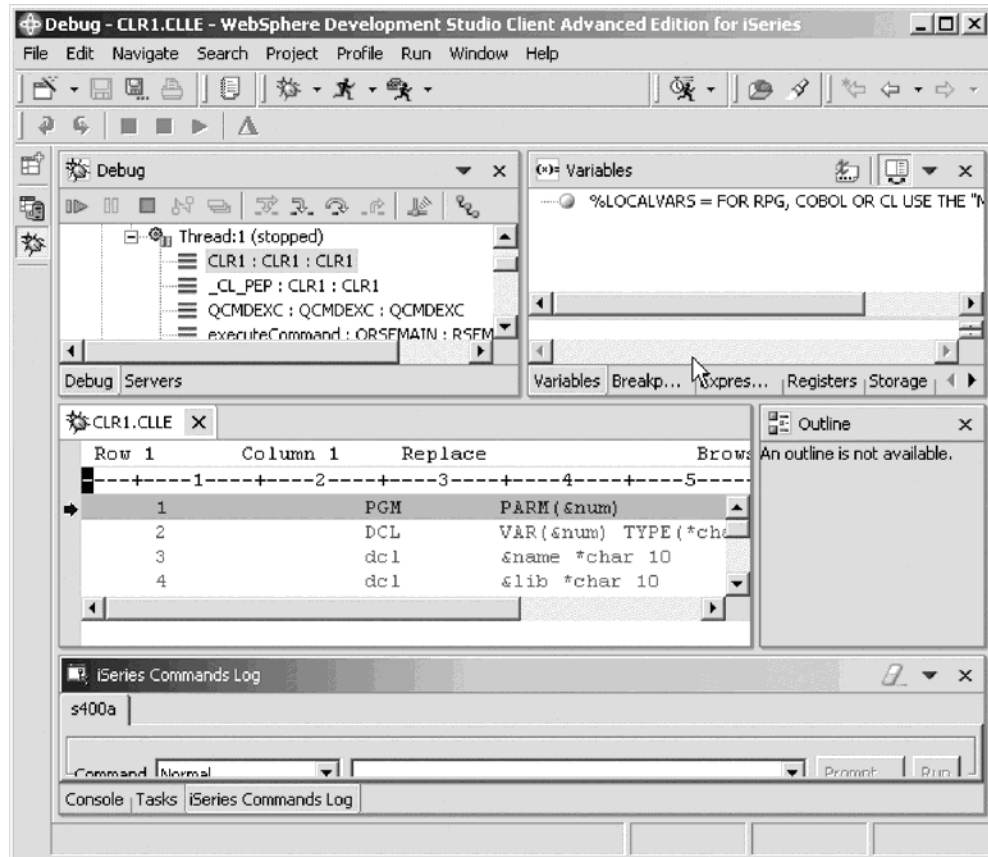
The Remote System Explorer communications server has been shut down in the meantime. Go to your 5250 emulator and restart the Remote System Explorer communications server following the instructions in the message. You don't have to cancel the message. It will be removed as soon as the connection between the Remote System Explorer communications server and the interactive session has been established. Now the Debug perspective is loaded in the workbench.

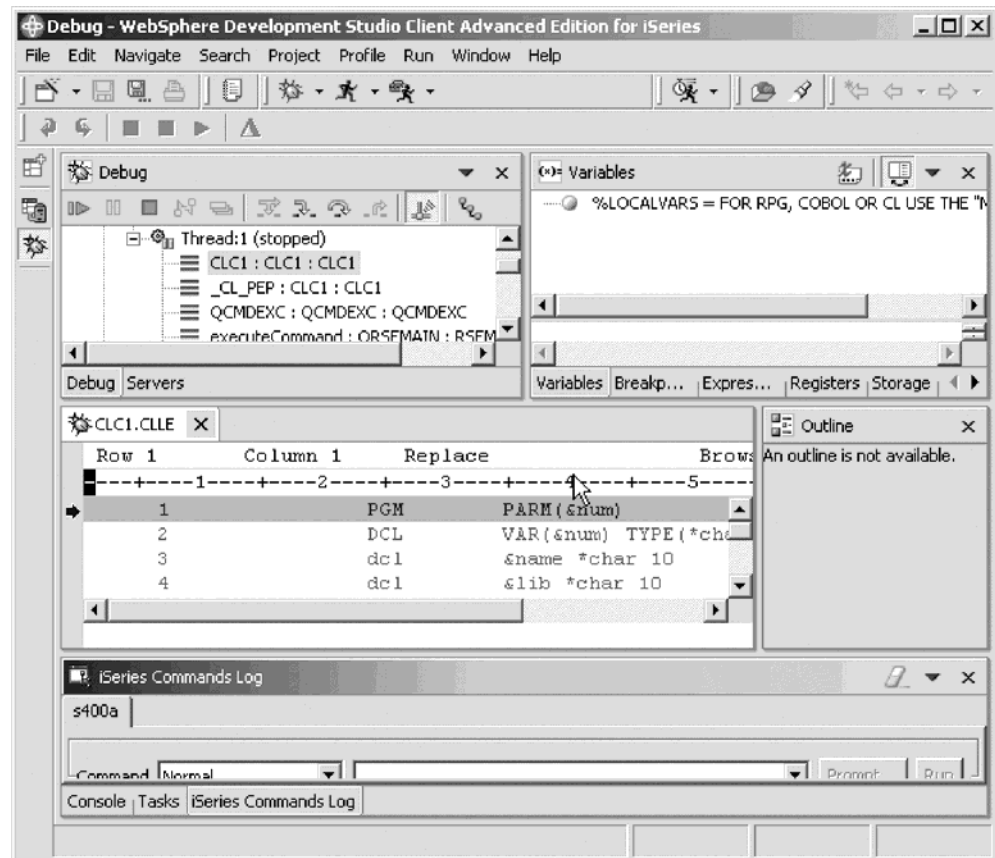
Now that the program is active on the iSeries and stopped at the first executable statement, the debugger displays the source.

## Setting breakpoints

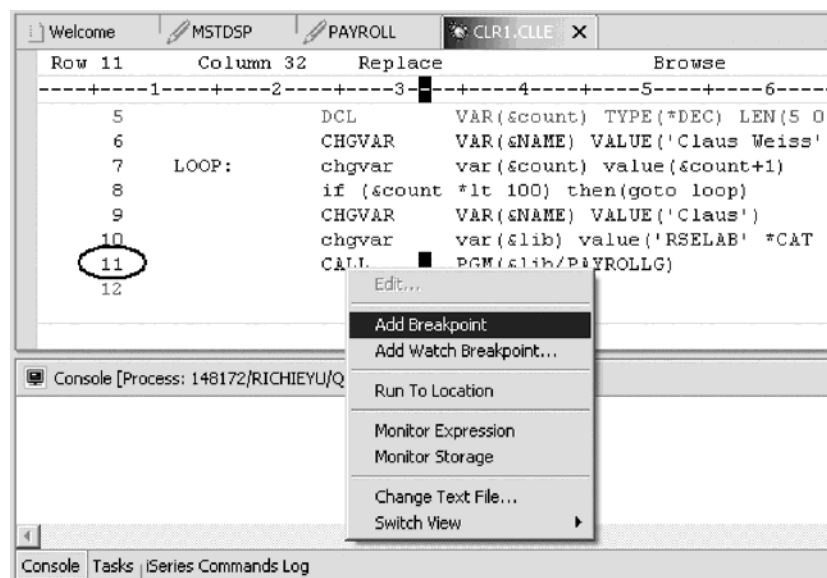
You can only set breakpoints at executable lines. All executable lines are displayed in blue. The easiest way to set a breakpoint is to right-click on the line in the Source view.

To set a breakpoint:





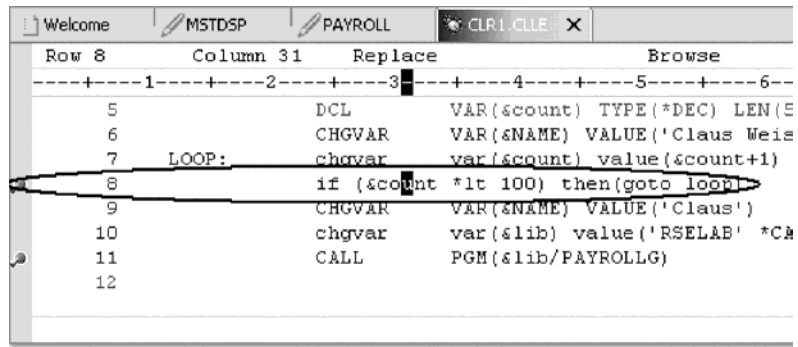
1. Scroll to line 11 then right-click anywhere on line 11.



2. Click **Add breakpoint** on the pop-up menu.

A dot with a checkmark in the prefix area indicates that a breakpoint has been set for that line. The prefix area is the small grey margin to the left of the source lines.

Now you add a conditional breakpoint in the loop when it loops the 99th time.



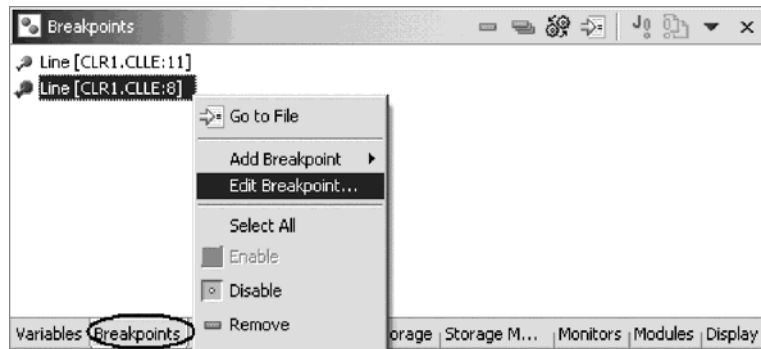
The screenshot shows a code editor window titled 'CLR1.CLE'. The code is as follows:

```
Row 8      Column 31  Replace      Browse
-----+-----+-----+-----+-----+-----
5          DCL      VAR(%count) TYPE(*DEC) LEN(8)
6          CHGVAR  VAR(%NAME) VALUE('Claus Weie
7          LOOP:   chgvar  var(%count) value(%count+1)
8          if (%count *lt 100) then(goto LOOP)
9          CHGVAR  VAR(%NAME) VALUE('Claus')
10         chgvar  var(%lib) value('RSELAB' *Ca
11         CALL    PGM(%lib/PAYROLLG)
12
```

To add a conditional breakpoint:

1. Scroll to line 8 then right-click on line 8.
2. Click **Add breakpoint** on the pop-up menu.
3. Click the **Breakpoints** tab.

The Breakpoints view opens:



4. In the **Breakpoints** view right-click the breakpoint for line 8.
5. Click **Edit Breakpoint** on the pop-up menu.

The Edit a Line Breakpoint window opens.

6. Under **Views**, click the **\*LISTING** radio button.

**Edit a Line Breakpoint**

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

Program: \*PGM RSELABXX/CLR1

Module: CLR1

Views:  
 \*SOURCE  
 \*LISTING  
 \*STATEMENT

Source(optional): CLR1.LISTING

Line: 31

< Back   Next >   Finish   Cancel

**Edit a Line Breakpoint**

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

Program: \*PGM RSELABXX/CLC1

Module: CLC1

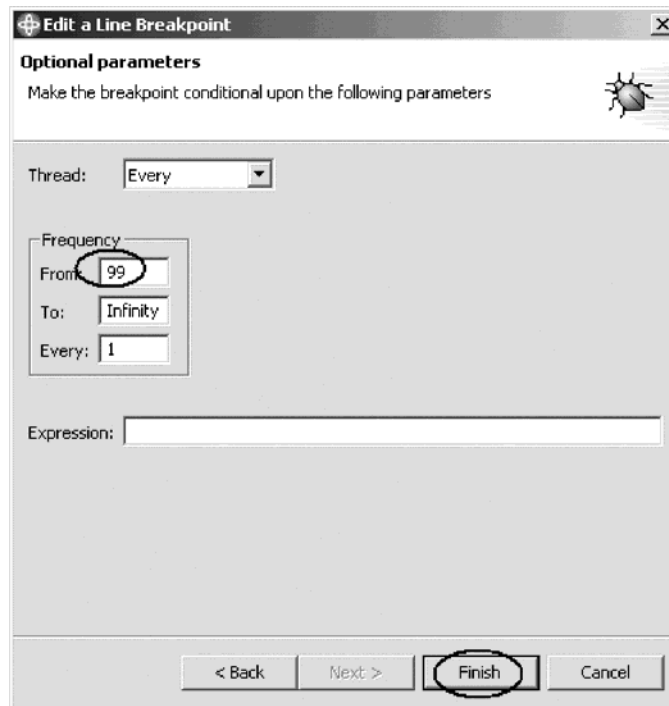
Views:  
 \*SOURCE  
 \*LISTING  
 \*STATEMENT

Source(optional): CLC1.LISTING

Line: 31

< Back   Next >   Finish   Cancel

7. Click Next.



You only want to stop in the loop when it executes for the 99th time or more. You can do that by setting the From field of the Frequency group to 99.

8. Under **Frequency**, in the **From** field, type 99.
9. Click **Finish**.

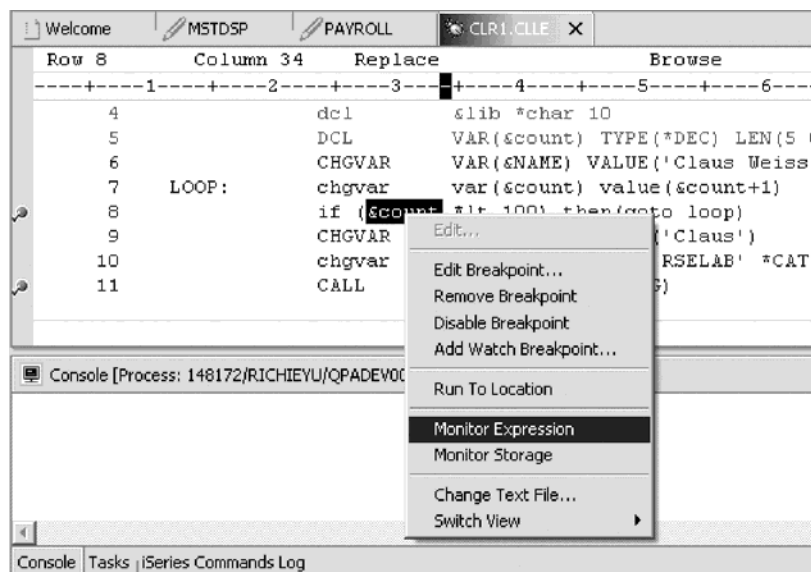
---

## Monitoring variables

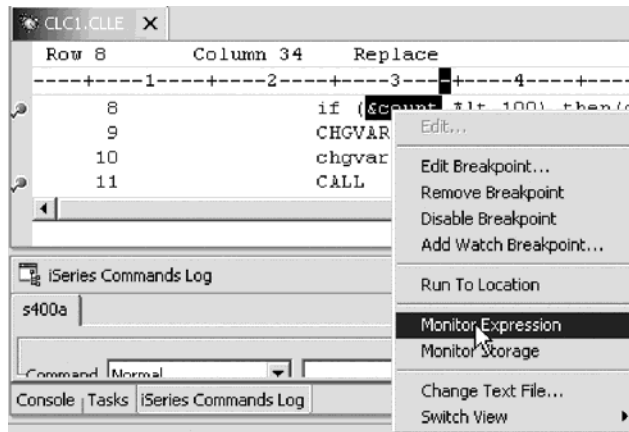
You can monitor variables in the Monitors view. Now you will monitor the variable &count.

To monitor a variable:

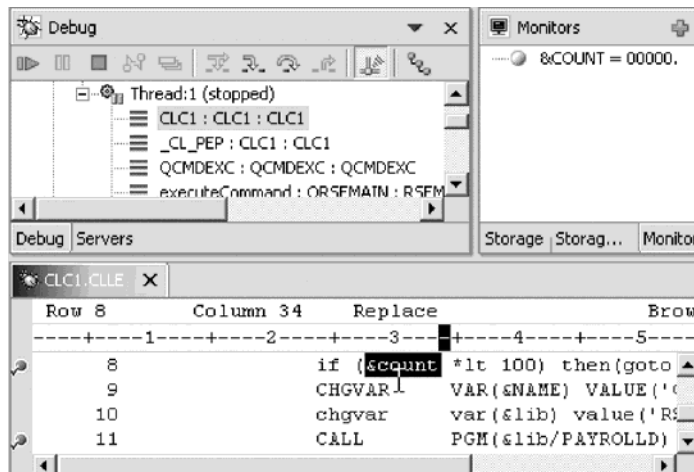
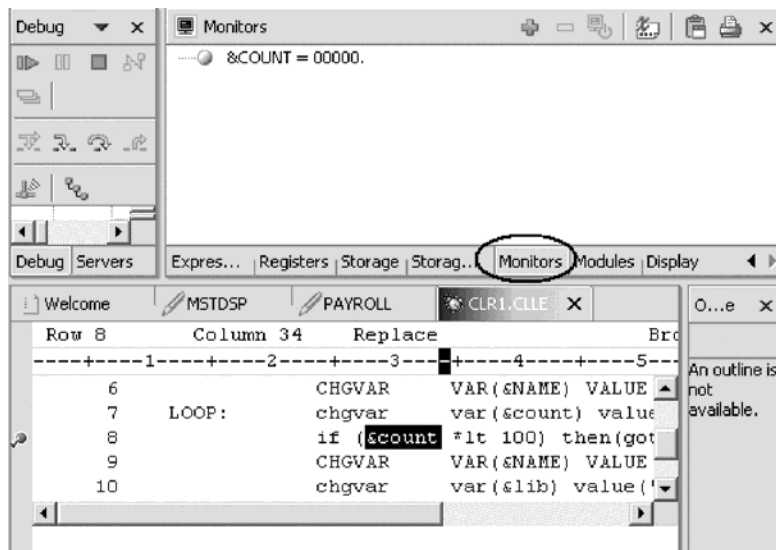
1. In the Source view, double-click the variable &COUNT.







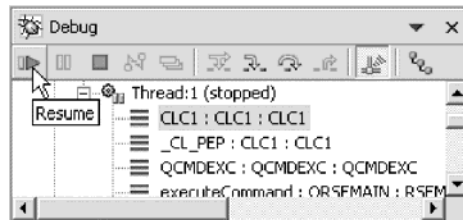
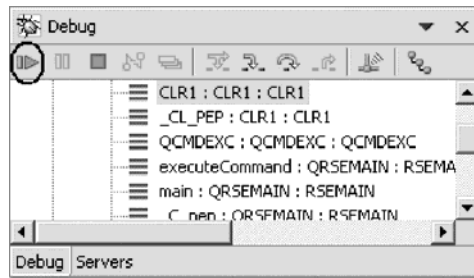
2. Right-click `&COUNT`.
  3. Click **Monitor Expression** on the pop-up menu.
- The Monitors view opens.



The variable appears in the Monitors view. Its current value is zero.

Tip: If you quickly want to see the value of a variable without adding it to the Monitor, leaving the mouse pointer on a variable for a second or so will display its value in a pop-up window.

Now that some breakpoints are set, you can start to run the application.



4. Click the **Resume**  icon from the Debug toolbar.

The program starts running and stops at the breakpoint at line 8. (Be patient, the Debugger has to stop 98 times but because of the condition continues to run until the 99th time.) Notice in the Monitors view, that `&count` now has the value 99.

5. Click the **Resume** icon again.

The program stops at the breakpoint at line 8 again and `&count` now has the value 100.

6. Click the **Resume** icon once more so that the program runs to the breakpoint at line 11.

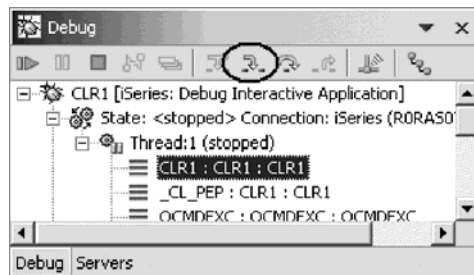
---

## Stepping into a program

The Debugger allows you to step over a program call or step into it. When you step over a program call, the called program runs and the Debugger stops at the next executable statement in the calling program. You are going to step into (Step into) the Payroll program.

To step into a program:

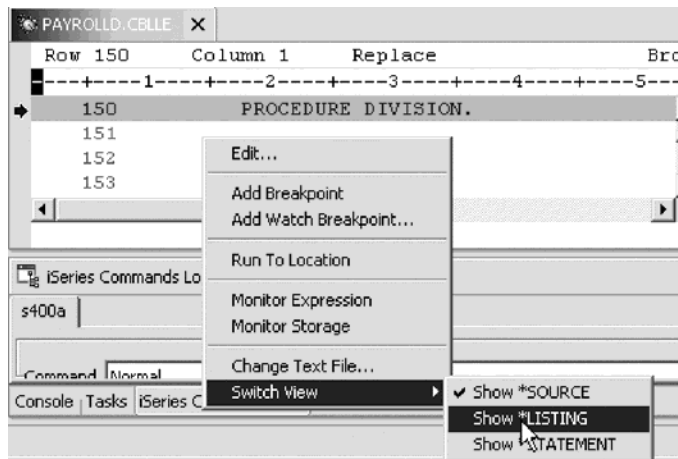
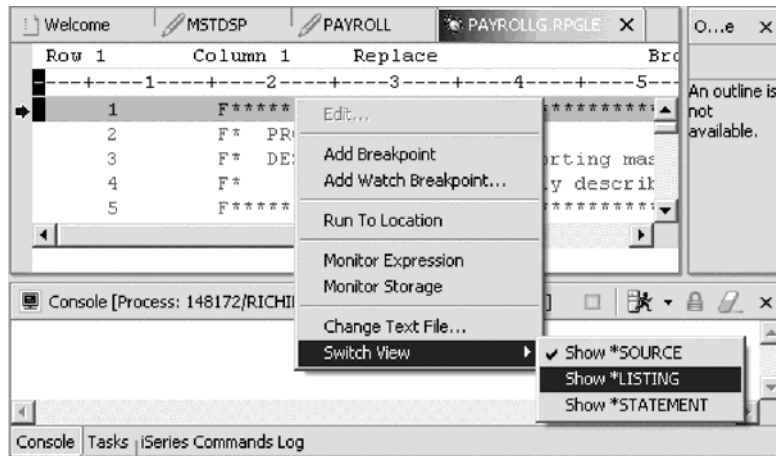
1. Click the **Step into** icon on the Debug toolbar.



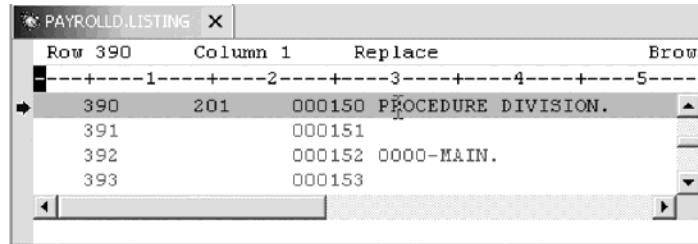
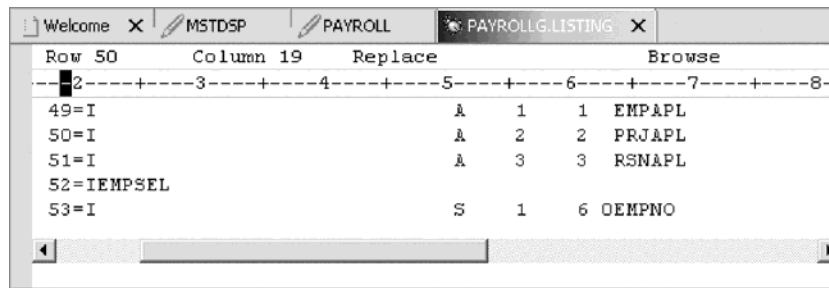


The source of PAYROLLG is displayed if you stepped into RPG or PAYROLLD if you stepped into COBOL. Depending on the option you used to compile the program (\*SRCDBG or \*LSTDBG for RPG or COBOL, or \*SOURCE, \*LIST, or \*ALL for ILE RPG or ILE COBOL), this window displays either the Source or Listing View.

2. Right-click anywhere in the Source view.



- Click **Switch view > Show \*LISTING** on the pop-up menu.

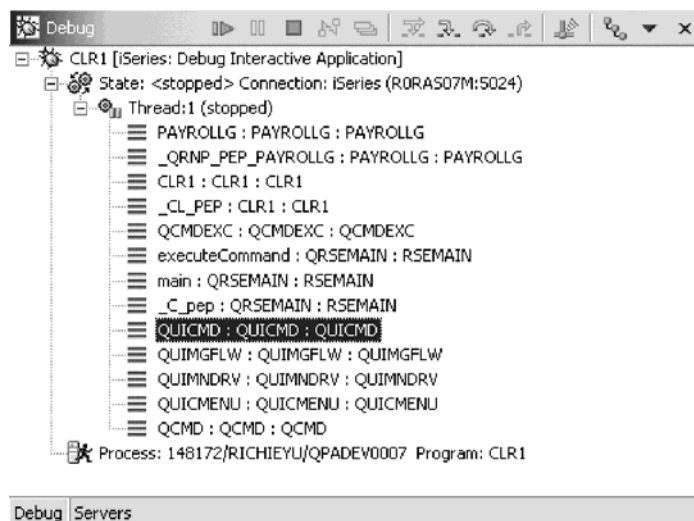


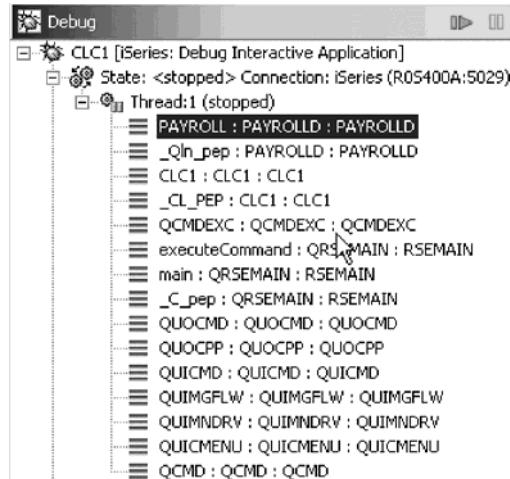
- Page down in the source and take a look at the expanded file descriptions. You don't have any /Copy member in your PAYROLL program but these would also be shown in a Listing view. Switch back to the Source view.
- Right-click anywhere in the Source view.
- Click **Switch view > Show \*SOURCE** on the pop-up menu.

## Listing call stack entries

The Debug view in the upper left pane, lists all call stack entries. It contains a tree view for each thread. The thread can be expanded to show every program, module, procedure and method that is on the stack at the current execution point. If you double click on a stack entry you will display the corresponding source if it is available. Otherwise the message No Debug data available appears in the Source view.

In the Debug view, expand the stack entry of Thread1 if it is not expanded already.





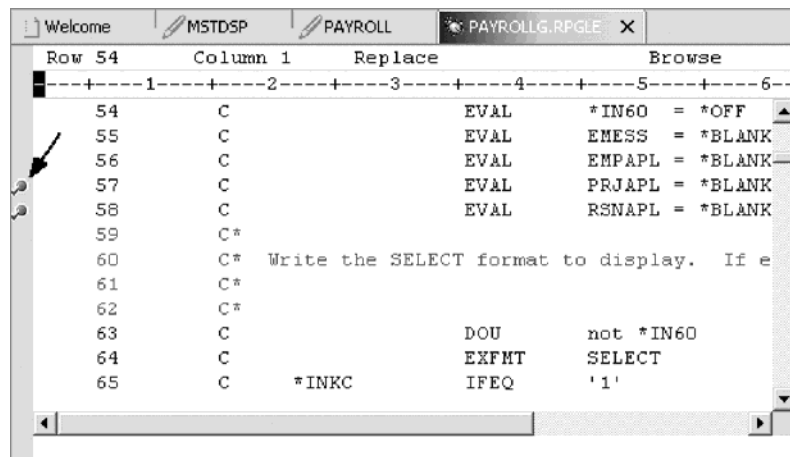
The stack entry allows you to work with and switch between different programs and/or ILE modules.

## Setting breakpoints in PAYROLLG or PAYROLLD

Now you add some breakpoints in PAYROLLG or PAYROLLD.

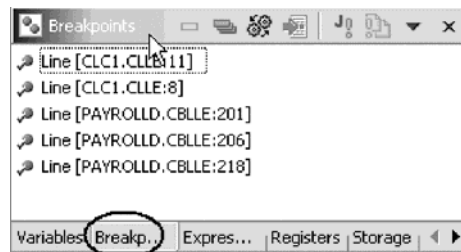
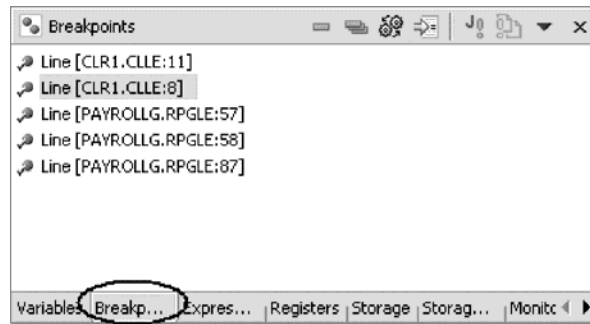
To add breakpoints:

1. Select PAYROLLG or PAYROLLD in Thread1.
2. Scroll to line 57 in PAYROLLG or 201 in PAYROLLD.
3. Double-click the prefix area of line 57 or double-click the prefix area of line 201.  
A breakpoint icon is added to the prefix area of this line to indicate that a breakpoint is set.
4. Repeat the above step for line 58 in PAYROLLG or 206 in PAYROLLD.



5. Repeat the above step for line 87 in PAYROLLG or 218 in PAYROLLD.

To view all breakpoints, select the **Breakpoints** tab from the top left pane.



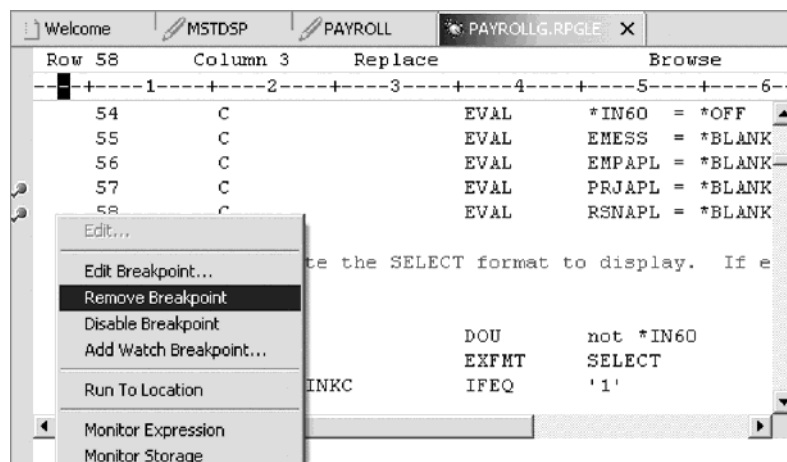
This view shows all breakpoints currently set in your Debug session. This is a convenient place to work with breakpoints. You can remove, disable/enable, add, or edit a breakpoint. These tasks are available from the pop-up menu when you right mouse click in the view area. Double-click any entry to show the source where the breakpoint is set.

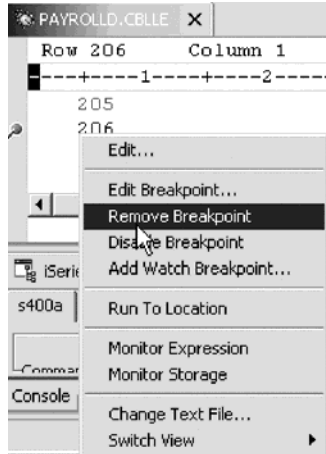
## Removing a breakpoint in PAYROLLG or PAYROLLD

It is also easy to remove breakpoints from the Source view.

To remove a breakpoint:


1. Scroll to line 58 and then right-click line 58 in PAYROLLG or 206 in PAYROLLD.
2. Click **Remove Breakpoint** on the pop-up menu.





The icon is removed from the prefix area indicating that no breakpoint is set on that line.

Now you are ready to run the PAYROLLG program or PAYROLLD program.

3. Click the **Resume**  icon from the Debug toolbar.
4. The RPG program PAYROLLG starts running and runs to the breakpoint at line 57. The COBOL program PAYROLLD waits for 5250 input.
5. Click the **Resume** icon again for PAYROLLG.

The program waits for input from the 5250-emulation session.



6. Type an X beside the **Project Master Maintenance** option.
7. Press **Enter** in the emulation session.

The program runs to the breakpoint at line 87 for PAYROLLG or 201 for PAYROLLD.

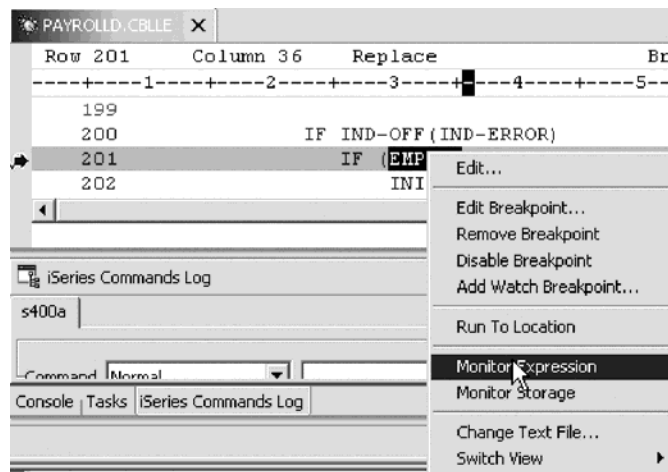
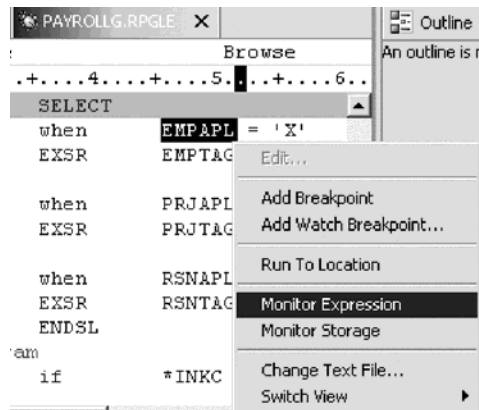
## Monitoring variables in PAYROLLG or PAYROLLD

Now lets monitor variables and change them in PAYROLLG or PAYROLLD.

To monitor variables:

1. In the source view, double-click the variable EMPAPL on line 88 in PAYROLLG or 201 in PAYROLLD.

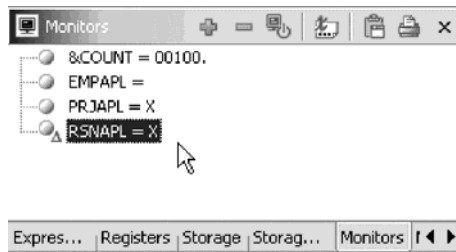
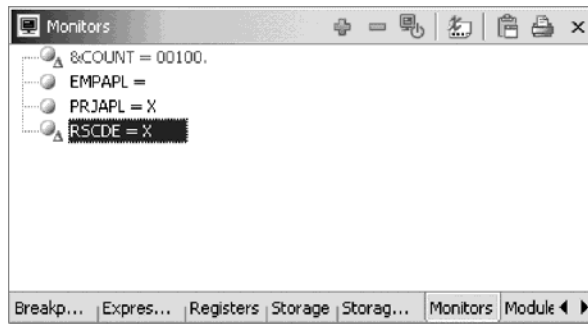
2. Right-click the variable.
3. Click **Monitor Expression** on the pop-up menu.



4. Click the **Monitors** tab in the upper right pane.  
The variable appears in the **Monitors** view. Its value is blank because you did not select the **Employee Master Maintenance** option.
5. In the same way add the variables PRJAPL on line 91 and RSCDE on line 113 to the monitor for PAYROLLG or line 206 for variable PRJAPL and line 244 for variable RSNAPL for PAYROLLD.  
Variable PRJAPL equals 'X' because you did select the **Project Master Maintenance** option.
6. In the Monitors view, double-click the variable RSCDE for RPG or RSNAPL for COBOL.  
The value changes into an entry field.



- In the value field, type in the new value X for the variable.



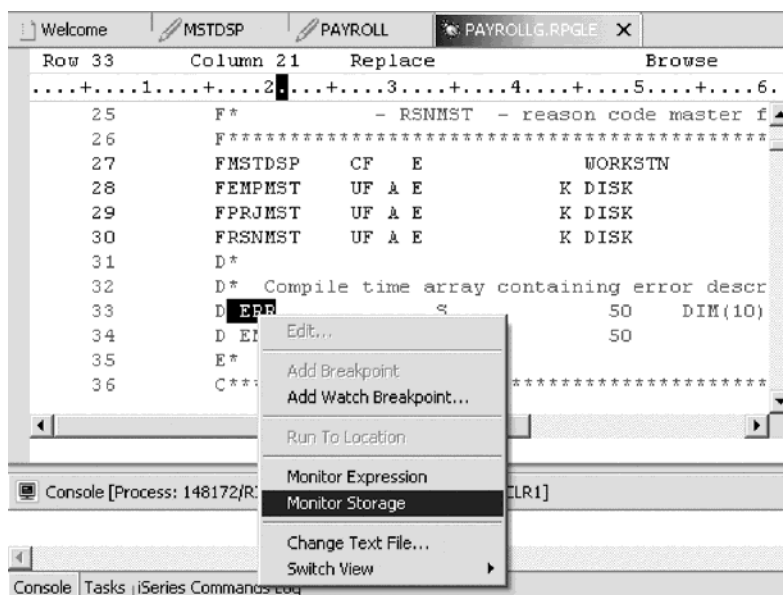
- Press **Enter**.  
The variable is successfully changed.

## Adding a storage monitor

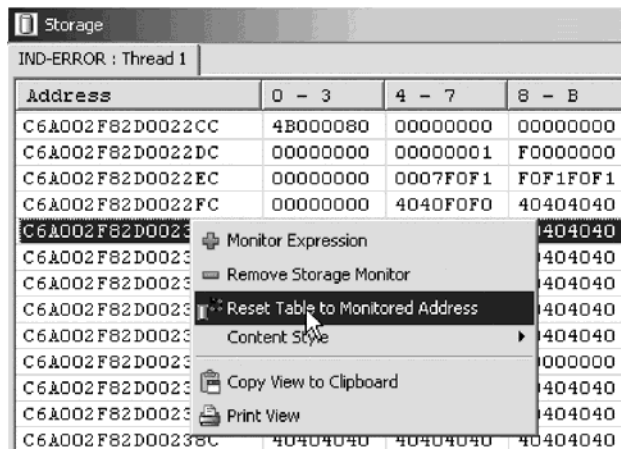
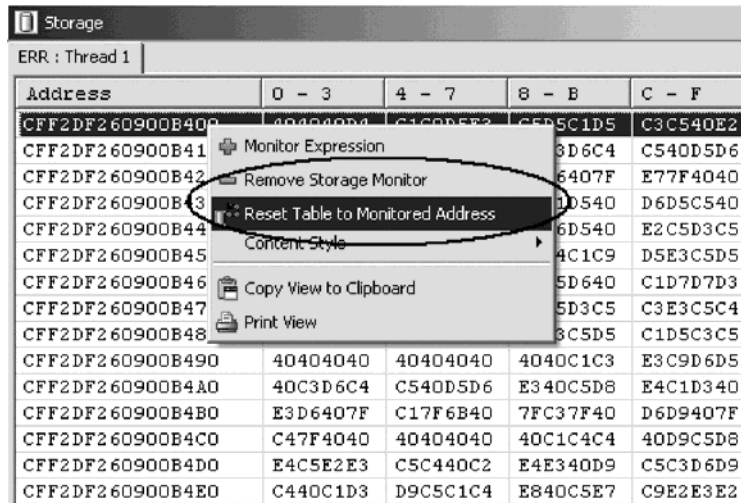
Adding a storage monitor for a variable allows you to view the storage starting with the address where the variable is located. The storage is displayed in hexadecimal and text format.

To add a storage monitor:

- In the Source view, double-click the variable ERR in line 33 for PAYROLLG or IND-ERROR in line 219 for PAYROLLD.
- Right-click and select **Monitor Storage** on the pop-up menu.



A new page is added to the Storage view. The tab shows the name of the variable.



- Use the scroll bar on the right of the Storage view to scroll down. You can see the current content of the memory.
- Right-click in the view area.
- Click **Reset Table to Monitored Address** on the pop-up menu to return to the starting address.
- Right-click the view area.
- Click **Remove Storage Monitor** on the pop-up menu to remove the storage monitor.

## Setting watch breakpoints

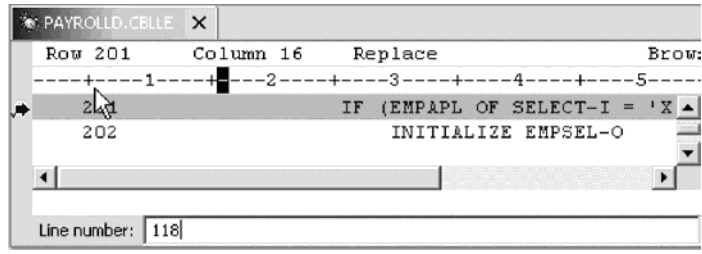
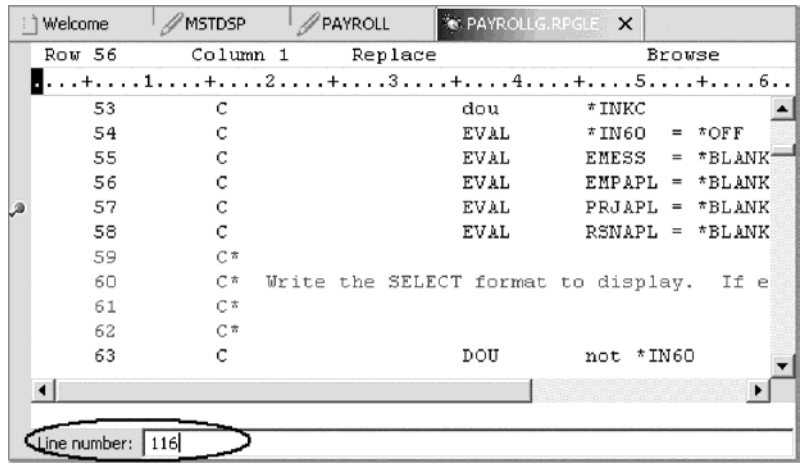
A watch breakpoint provides a notification to the user when a variable changes. It will suspend the execution of the program until an action is taken.

To set a watch breakpoint:

- Click somewhere in the Source view and press **Ctrl+L**.

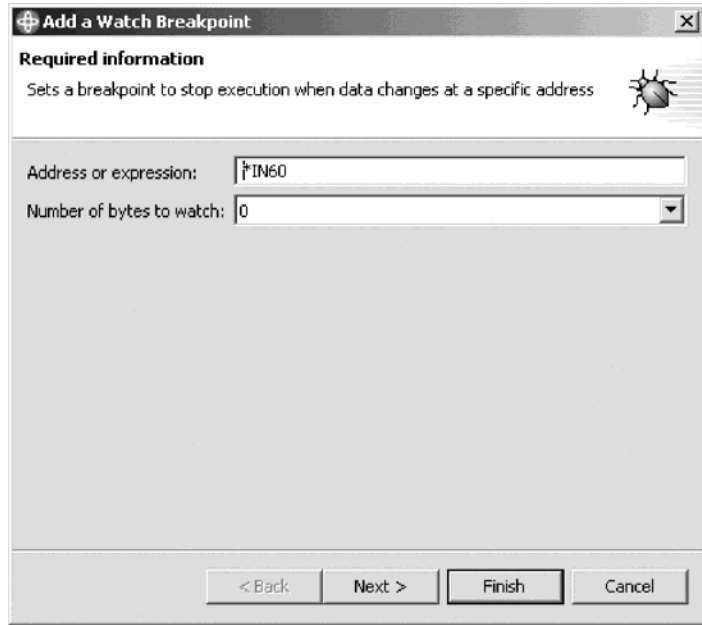
A **Line number** field is added to the bottom of the source area. In this field type 116 to go to that line in PAYROLLG or 118 to go to that line in

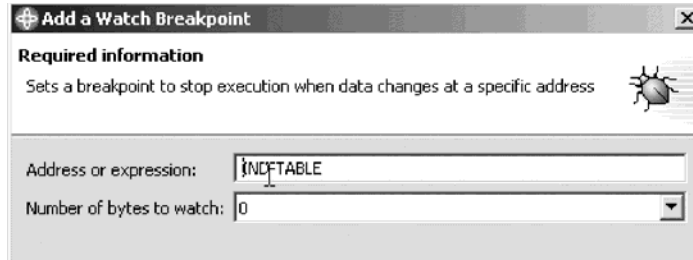
PAYROLLD.



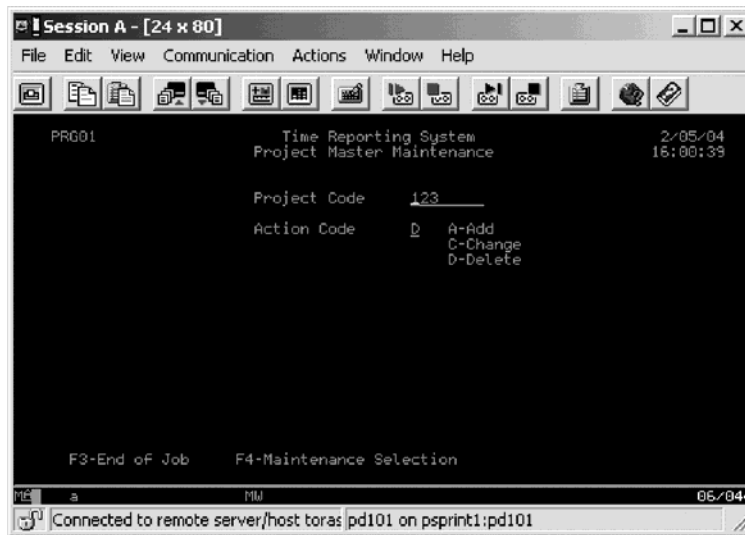
2. Double-click variable \*IN60 in PAYROLLG or IND-TABLE in PAYROLLD to highlight it.
3. Right-click and click **Add Watch Breakpoint** on the pop-up menu.

The Add a Watch Breakpoint window opens. The **Expression** field is pre-filled with the highlighted variable \*IN60 or IND-TABLE. By default the **Number of bytes to watch** field is set to zero, which means the variable will be watched in its defined length.

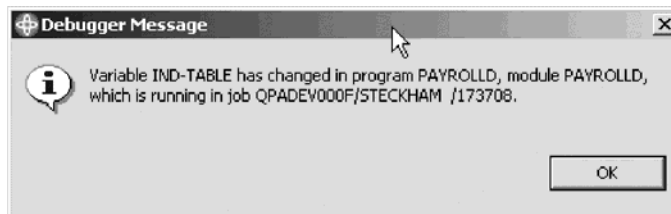
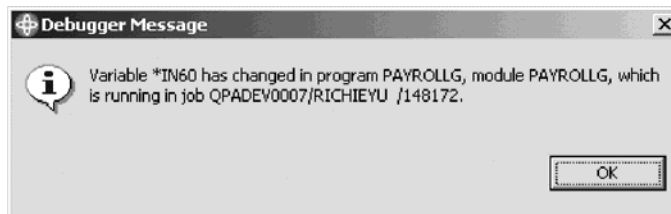




4. Click **Finish**.  
The watch breakpoint is now set.
5. Click the **Resume** icon on the Debug toolbar.  
The application waits for input from the 5250-emulation session.



6. In the 5250 emulation session, type 123 for **Project Code** and D (for delete) in the **Action Code** field.
7. Press **Enter**. A message is displayed indicating that the variable \*IN60 or IND-TABLE has changed.



8. Click **OK**. The program stops at line 465 in PAYROLLG or 407 for PAYROLLD. This line is located immediately after the statement which caused the variable \*IN60 or IND-TABLE to change.

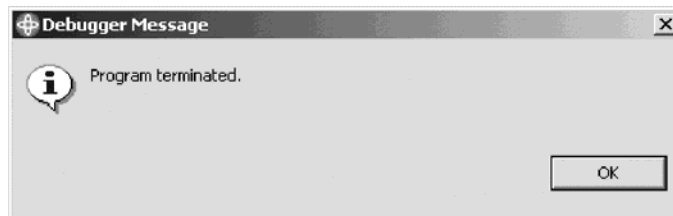
---

## Closing the debug session

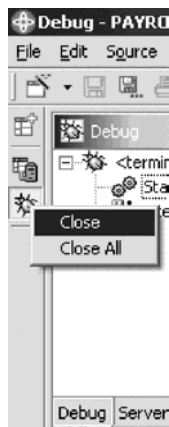
To close the debugger:

1. Click the **Resume** icon on the Debug toolbar.  
The application waits for input from the 5250-emulation session.
2. Switch to the 5250 emulation session.
3. Press **F3** to end the job.

A message **Program terminated** appears:



4. Click **OK**.



5. Right-click the **Debug** icon on the left task bar of the workbench.
6. Click **Close** on the pop-up menu.

---

## Checkpoint

Complete the checkpoint below to determine if you are ready to move on to the next module.

1. You can start the debugger:
  - a. From the Remote Systems view
  - b. Launch Configurations window
  - c. Both
2. You can only set breakpoints at executable lines. (T, F)
3. The easiest way to set a breakpoint is to:
  - a. Right-click on the line
  - b. Right-click after the line.
  - c. Right-click before the line
  - d. All of the above
4. You can change variables and indicators in the:
  - a. Remote Systems view

- b. Debug view
  - c. Monitors view
  - d. Storage view
  - e. All of the above
5. The debugger allows you to:
  - a. Step over a program call
  - b. Step into a program call C
  - c. Both
6. The Debug view lists all call stack entries. It contains a tree view for each thread. (T, F)
7. You can perform which actions on breakpoints:
  - a. Delete
  - b. Add
  - c. Disable
  - d. Enable
  - e. Edit
  - f. All of the above
8. Adding a storage monitor for a variable allows you to view the storage starting with the address where the variable is located. (T, F) 9.
9. The Storage monitor supports these display formats:
  - a. Hexadecimal and character
  - b. Character only
  - c. Decimal
  - d. All of the above
  - e. A and B
10. A \_\_\_\_\_ breakpoint provides a notification to a user when a variable changes. It will suspend the execution of the program until an action is taken.
  - a. Watch
  - b. Support
  - c. Java Exception
  - d. Type

---

## More practice

Want more practice? Try this!

Given your experience in working with the debugger features, in your own source, try setting, changing, deleting, enabling, disabling line breakpoints, setting watch breakpoints, displaying and changing variables and viewing the call stack as you debug your program. Use the Development Studio Client online help to assist you in these tasks.

---

## Recap

Congratulations! You have completed this module. You should now understand:

- The purpose of the Debug view, the Monitors view, the Storage Monitor, the Call Stack and the roles of different types of breakpoints
- The actions you can perform on breakpoints

- How to start the debugger
- How to set breakpoints
- How to monitor variables
- How to run and step into a program
- How to view the call stack in the Debug view
- How to remove a breakpoint
- How to add a storage monitor
- How to set watch breakpoints





---

## Chapter 11. Exploring Remote System Explorer

In this module, you will use the Remote System Explorer perspective to work with the iSeries objects that you used in the previous module. You will learn how easy it is to define filters, perform actions and define your own actions. In short, you'll see how Remote System Explorer can organize and integrate your work and make that work easier.

In order to explore the Remote System Explorer, there are several steps you will need to learn about and follow:

1. Describe Remote System Explorer, filters, user actions and running commands
2. Open Remote System Explorer
3. Create filters (library, object)
4. Change the library filter
5. Create a user-defined action
6. View properties
7. Run commands from iSeries Table view

In order to accomplish these learning objectives, there are several steps that are involved, including:

- Introducing the Remote System Explorer
- Creating a library filter
- Creating an object filter
- Creating a user action
- Running commands from the Remote System Explorer

The exercise within this module must be completed in order. Start with the first exercise when you ready to begin.

**Length of time:**

This module will take approximately 15 minutes to complete.

---

### Introducing the Remote System Explorer

Most of the function of the CODE Project Organizer has been replaced by WebSphere Studio function with the exception of accessing ADM parts. The Remote System Explorer is replacing PDM (Program Development Manager) on the workstation. It currently doesn't have all the function of PDM but will be a full replacement for PDM.

Remote System Explorer allows you to:

1. Simplify your work by giving you quick access to lists of iSeries libraries, objects, members, IFS files, UNIX files, and local files.
2. Use the context-sensitive pop-up menus on these lists to perform actions such as start the Remote Systems LPEX Editor, CODE Designer, or Integrated Debugger or other common iSeries actions.
3. Use the Work with User Actions option to create and manage your own user-defined actions and have them appear in the pop-up menus.

4. Use the command support to increase your productivity by allowing you to enter and repeat iSeries or local commands without switching to an emulator session.

---

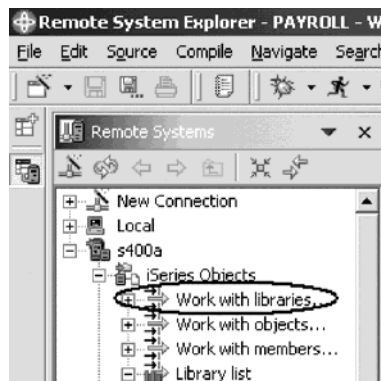
## Creating a library filter

In the Remote System Explorer perspective, you now want to work with specific iSeries objects.

In the previous modules you have worked with the Library list. Now you will create your own library filter. Library filters list a set of libraries from your iSeries system in the Remote Systems view. But first let's understand what filters are all about.

As a user, you want a flexible user interface that allows for integration between systems. As a developer, you often need to create libraries, source files, and source members. The Remote Systems view exposes subsystems, filters, and items specified by each filter. This view lets you organize your filtered information in an easy-to-understand tree view. You can create filters for collections of libraries, objects, and source members.

The Remote System Explorer provides iSeries native file system (QSYS) support. The iSeries native file system lets you query what objects are on your iSeries host and perform actions against those objects. Filters allow you to easily organize elements within your system. Use the filter function to list iSeries native file system objects (such as libraries, objects, or members). An initial library list filter also displays by default. It shows the initial libraries defined in your user profile on the iSeries host. You can manipulate this list to add or move libraries within the list.

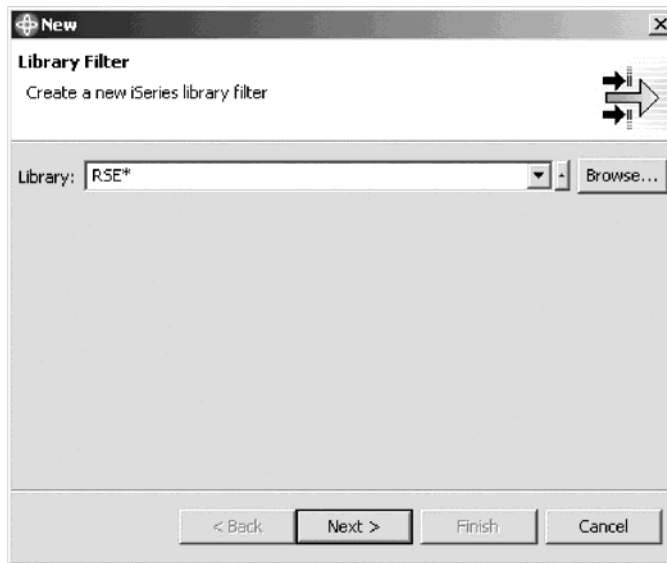


To create a library filter:

1. In the Remote Systems view expand the connection that connects to your iSeries system if its not already expanded.
2. Expand iSeries Objects if its not already expanded.
3. Expand Work with libraries. (You can also right-click iSeries Objects and click **New > Library Filter**) on the pop-up menu.

Expanding Work with libraries corresponds to the WRKLIBPDM command, plus creates and expands the filter in the Remote Systems view.

The Create a new iSeries library filter page opens:



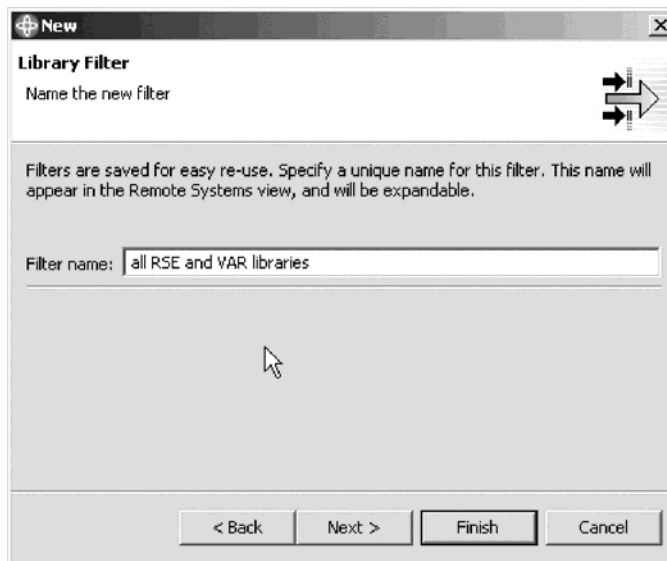
You are going to create a filter to specify the libraries you want to work with, so they will show in iSeries Objects. You want to create a filter that shows all libraries on the iSeries with the name **RSExxxxxx** and **VARxxxxxxx**, xxx being any character.

**Note:** You may need to select different libraries that appear on your system if libraries with the above names do not exist.

You specify the first filter string that selects the libraries starting with Remote System Explorer.

4. Type RSE\* into the **Library** field, using the \* wild card character.
5. Click **Next**.

The Name the new filter page opens:



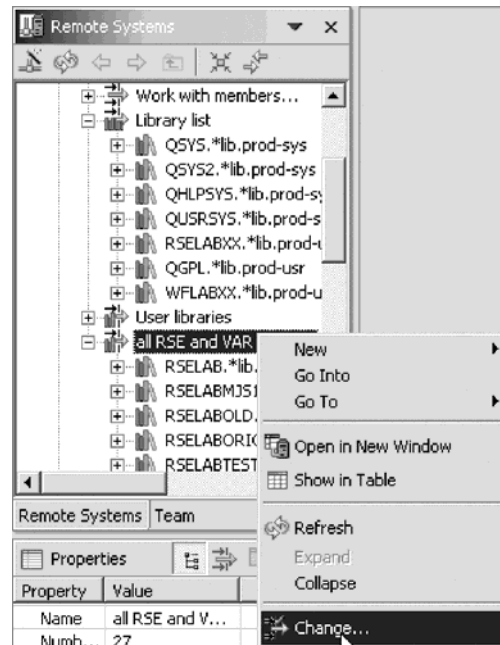
6. In the **Filter name** field, type all RSE and VAR libraries.  
You give your filters a name because the Remote System Explorer saves them for future use, in opposition to PDM, which does not save filters.
7. Click **Finish**.

Back in the Remote Systems view under iSeries Objects you will see the new filter expanded, listing all RSE\* libraries.

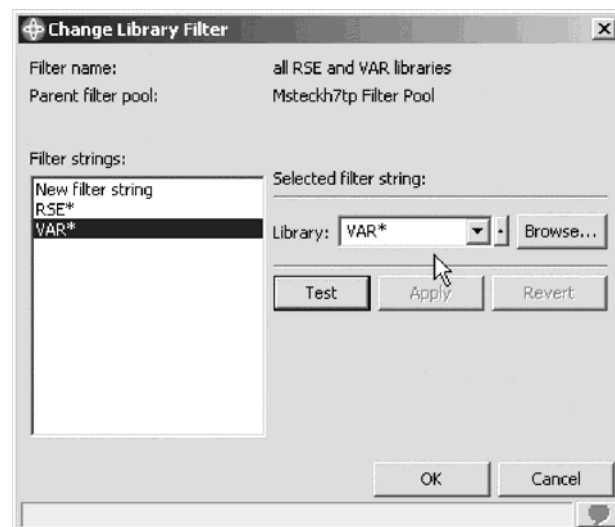
Now you need to add the VAR libraries.

To change the library filter:

1. Right-click the filter **all RSE and VAR libraries** and click **Change**.

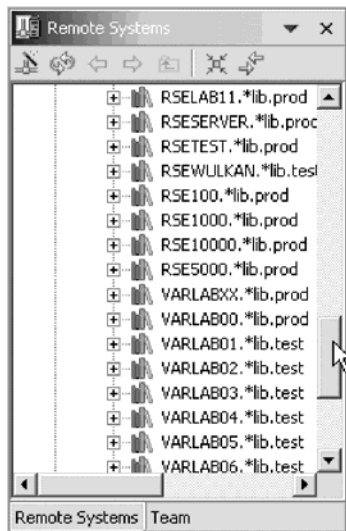


The **Change Library Filter** window opens:



2. Select **New filter string** from the **Filter strings** list.
3. In the **Library** field, type VAR\*.
4. Click **Create**.  
The **VAR\*** filter string is added to the list.
5. Click **OK**.

You are now back in the Remote Systems view.



You will see the list expanded to include your filter. Now you can work with the libraries directly and can drill down to the object you want to work with directly.

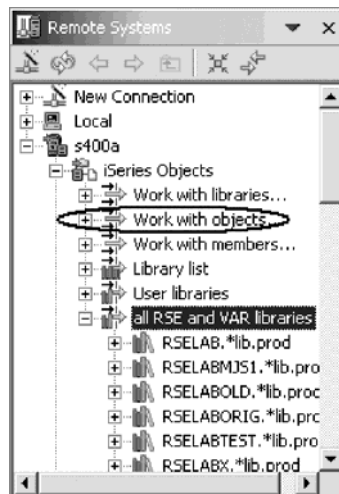
---

## Creating an object filter

Now create an object filter. Object filters list a set of objects from your iSeries host in the Remote Systems view.

To create an object filter:

1. In the Remote Systems view, expand your connection and then expand iSeries Objects if not already expanded.

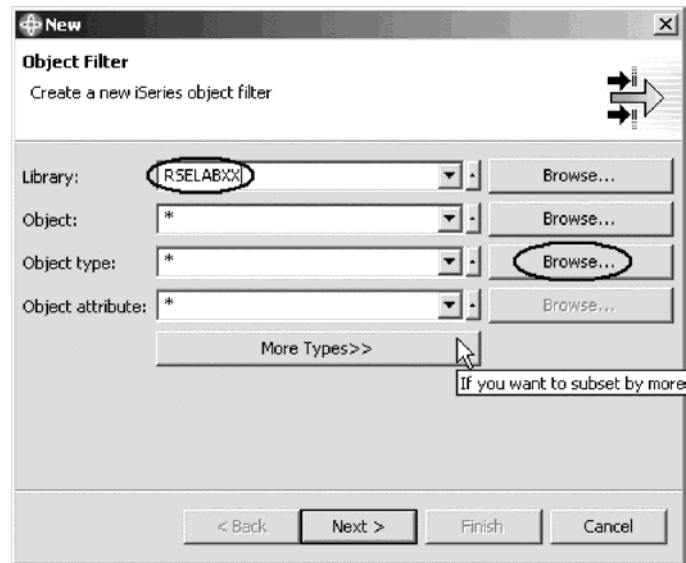


2. Expand **Work with objects**. You can also right-click **iSeries Objects** and click **New > Object filter** on the pop-up menu.

**Note:** Expanding Work with objects corresponds to the WRKOBJPDM command. This will show the New Object Filter window.

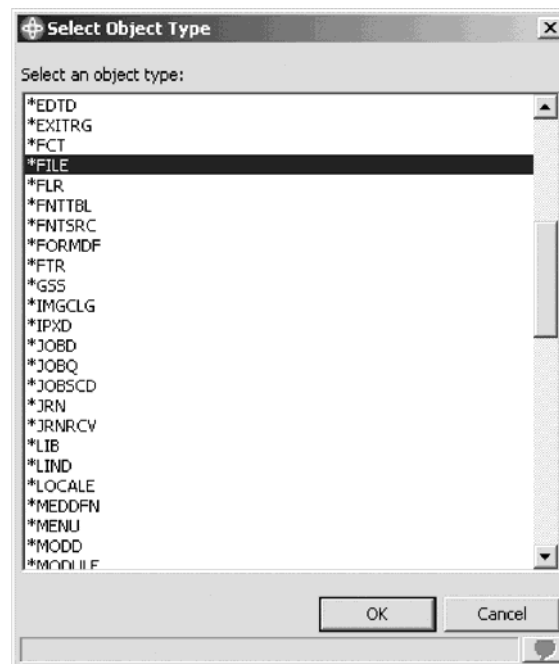
The Create a new iSeries object filter page opens:

Now create a filter to show all your source files in your RSELABXX library.



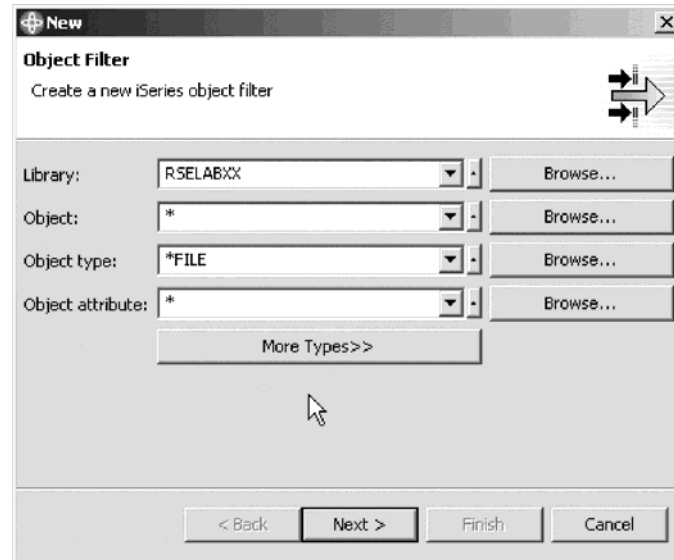
3. In the **Library** field, type RSELABXX.
4. Click **Browse** beside the **Object type** field.

The Select Object Type window opens:

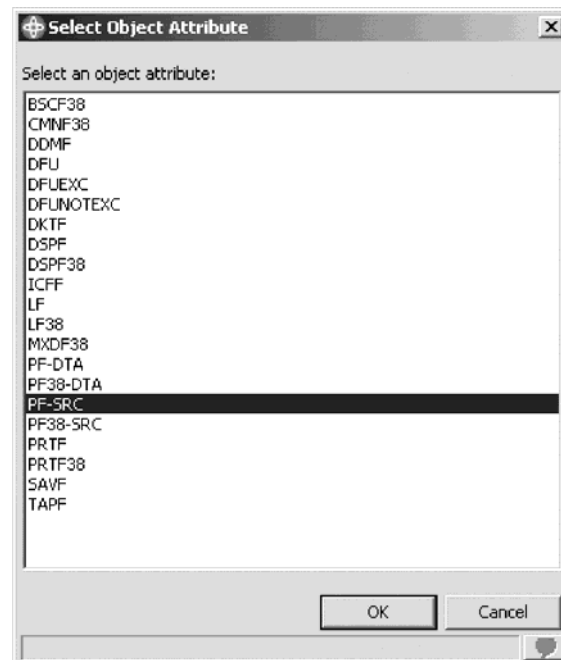


5. Select **\*File** under the **Select an object type** list.
6. Click **OK**.

The Create a new iSeries object filter page displays with the object type updated.

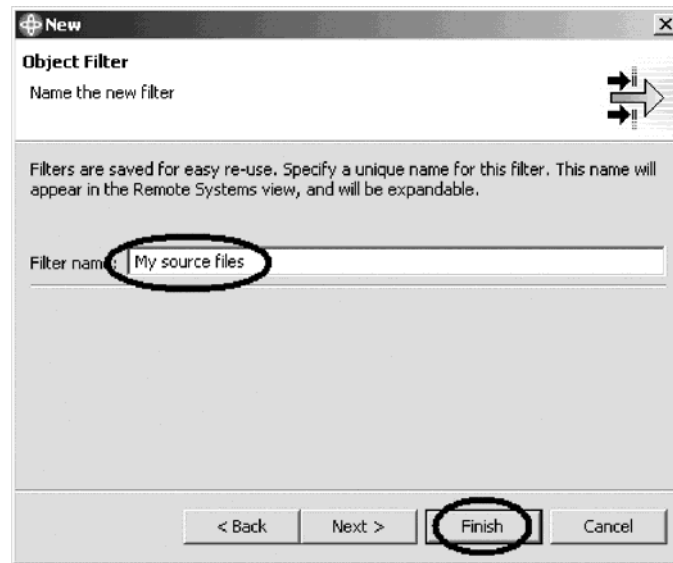


7. Click **Browse** beside the **Object Attribute** field.  
The Select Object Attribute window opens.
8. Select **PF-SRC** under the **Select an object attribute** list.
9. Click **OK**.



10. Click **Next**.

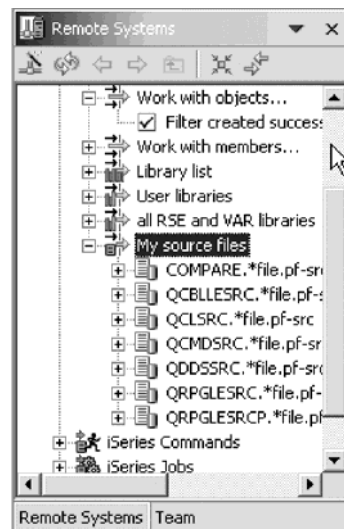
The Name the new filter page opens:



11. In the **Filter name** field, type My source files.
12. Click **Finish**.

**Note:** You give your filters a name because the Remote System Explorer saves them for future use, in opposition to PDM, which does not save filters.

The new object filter displays in the Remote Systems view under iSeries Objects:



Now you know how to create filters and tailor your development environment. Filters can also be specified for non iSeries servers and your local system.

Now you can work with the objects you have in your Remote Systems view like you worked in PDM with libraries, objects, or members.

Let's assume you want to edit the member PAYROLL in the source file QRPGLSRC or PAYROLLC in QCBLESRC using this object filter.

To edit a member:



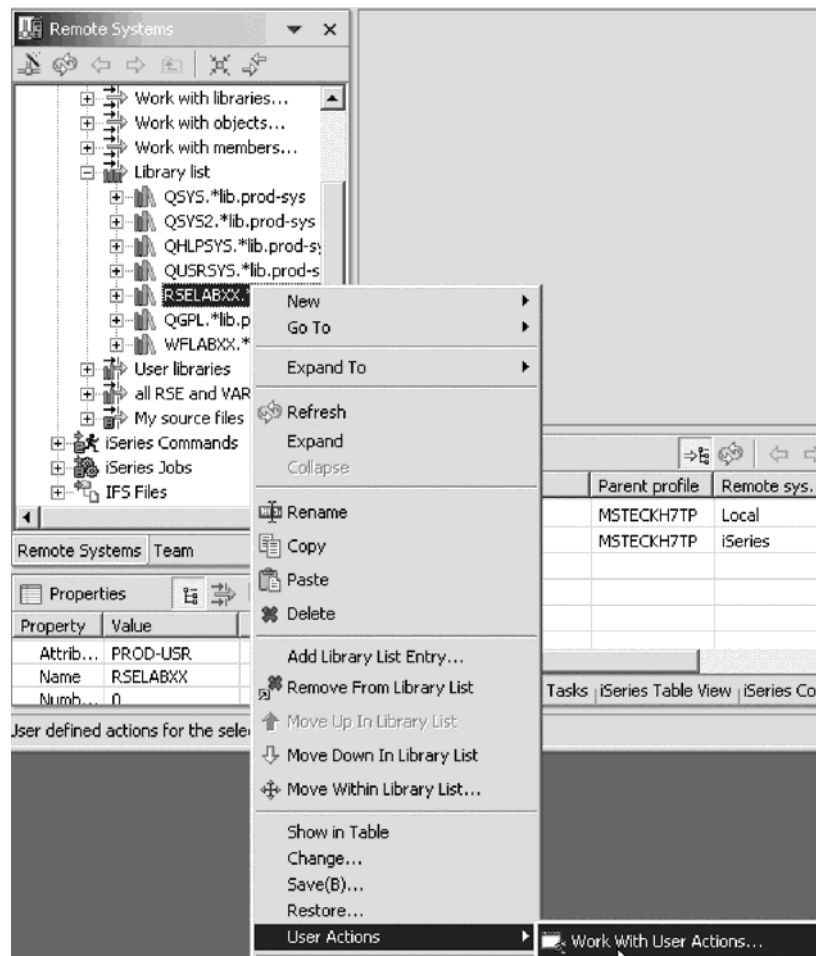
1. Expand QRPGLSRC or QCBLLESRC.
2. Right-click member PAYROLL or PAYROLLC.
3. Click **Open With > Remote Systems LPEX Editor** on the pop-up menu.  
This will download the source member and open the editor with this member. After you have edited the member you could save it and then compile it from the Remote Systems view by using the pop-up menu options on this member. You can also create your own actions in addition to the default actions.
4. Close PAYROLL or PAYROLLC.

## Creating a user action

In PDM you can create user actions in addition to the pre-supplied system actions. In Remote System Explorer you can do the same. So what are actions? Actions are host commands that you define on the Work With User Actions window, and will run against iSeries libraries, objects, jobs, and members. They can also be defined for folders and files in any remote UNIX, Windows, Linux, Local, or IFS system.

To open the Work with User Actions wizard:

1. Expand your iSeries connection and expand iSeries Objects if not already expanded.



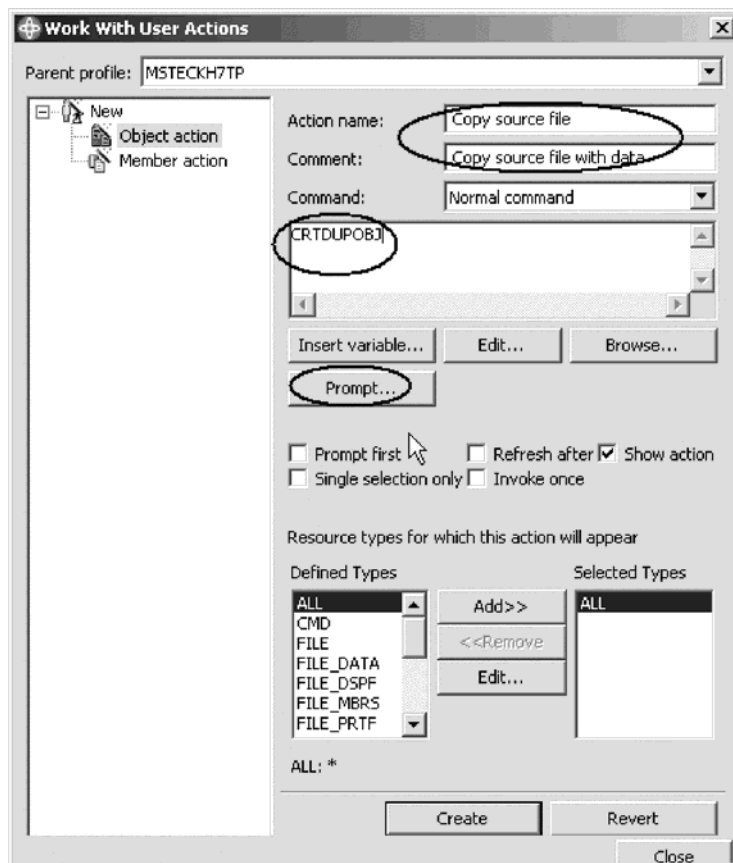
2. Expand the Library list filter if not already expanded.
3. Right-click RSELABXX.
4. Click **User Actions > Work with User Actions** on the pop-up menu.

The Work with User Actions window opens:



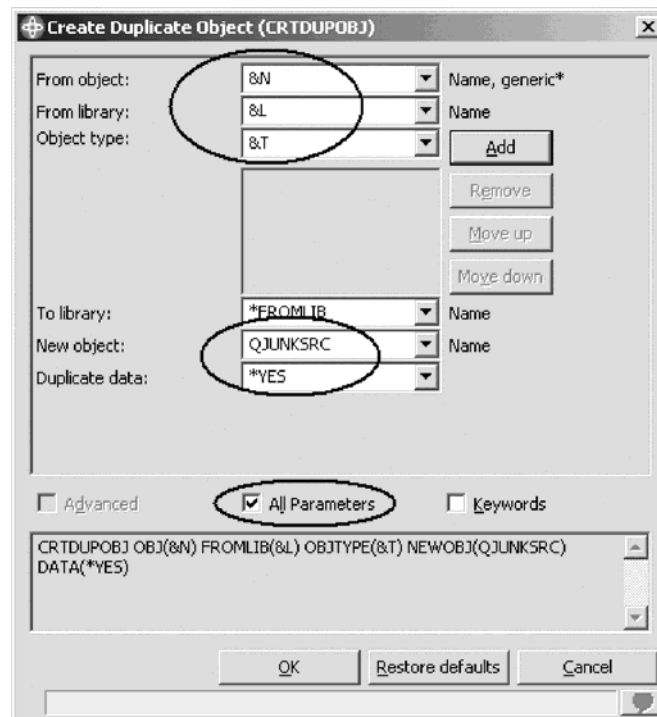
5. In the right pane of the Work with User Actions window, expand **New** in the list, if it is not expanded already,
6. Click **Object action**.

You want to create a user action that copies a source file with data to a new source file called QJUNKSRC in the same library.



7. In the **Action** field, type Copy source file for the user action name.
8. In the **Comment** field, type Copy source files with data.
9. In the **Command** field, type CRTDUPOBJ for the command to execute.

10. Click **Prompt** to open the command prompt for this command.



This is the command you will be running:

```
CRTDUPOBJ OBJ(&N) FROMLIB(&L) OBJTYPE(&T) NEWOBJ(QJUNKSRC) DATA(*YES)
```

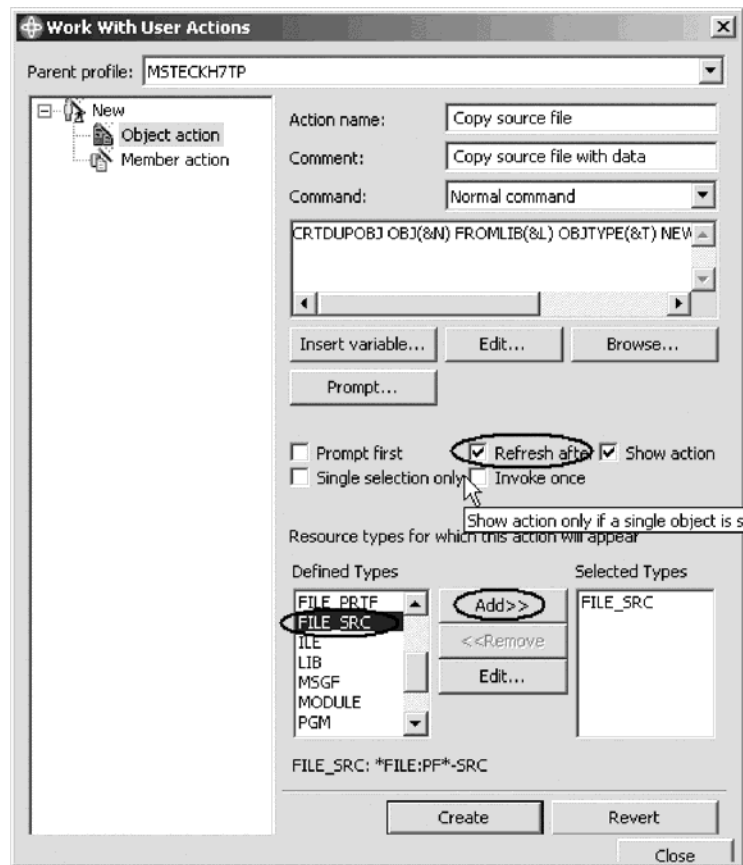
To specify user action parameters:

1. In the **From Object** field, type &N to indicate to use the name of the selected object in the Remote Systems view.
2. In the **From Library** field, type &L to pick up the library name from the selected object.
3. In the **Object Type** field, type &T to pick up the object type from the selected object.
4. In the **New Object** field, type QJUNKSRC.
5. Select **All parameters** check box to see the additional Duplicate data parameter. Now the Duplicate data parameter is also shown on the prompt window.
6. Select **\*YES** from the **Duplicate data** list.
7. Click **OK**.

You return to the Work with User Actions window.

Now you want to refresh the Remote Systems view after the action has been run.

8. Select the **Refresh after** check box.



Click **Insert variable** to see a list of valid replacement variables with the explanation of what they do.

This user action is only valid for Source physical files. You need to specify this restriction so this user action will only show in pop-up menus when you right-click on a source physical file.

To specify this restriction:

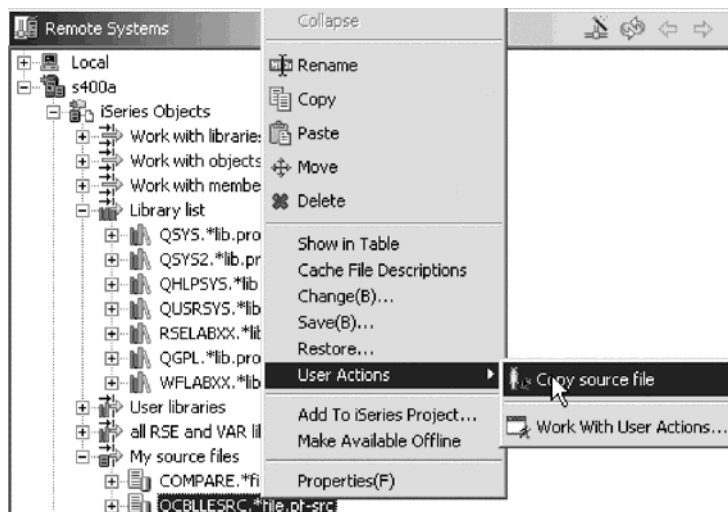
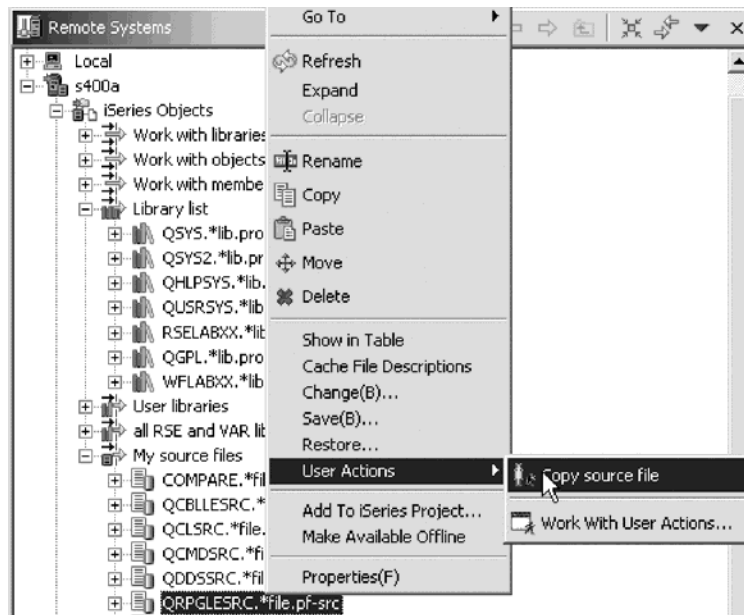
1. Under the **Defined Types** list, select **FILE\_SRC**.
2. Click **Add** beside the **Defined Types** list.  
FILE\_SRC is now one of the selected types. Actually since you only selected this one it is the only one.
3. Click **Create** then **Close**.

Now, only when you right-click on a source file, will this user action appear on the pop-up menu selected. For any other object type it will not appear. Back in the workbench and the Remote System Explorer perspective, give it a try.

**Note:** Remember to close the PAYROLL or PAYROLLC member if you opened it earlier.

To try your filter:

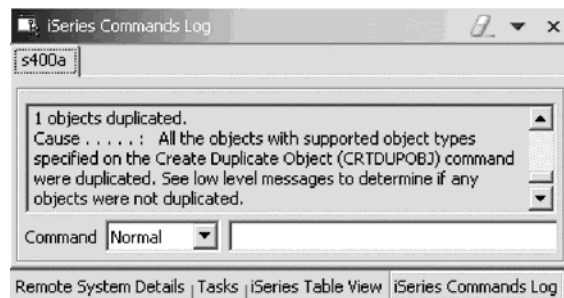
1. Locate your filter **My Source files**.



2. Expand the filter, if it is not already expanded.
3. Right-click the QRPGLSRC or QCBLLESRC file.
4. Click **User Actions > Copy source file** on the pop-up menu.

The file gets duplicated and the list gets refreshed. Your new source file will show in the list. You can check the messages of the CL commands you are running in the Remote System Explorer job by looking at the Commands log in

the right hand side bottom pane of the workbench.



Now you want to delete the source file QJUNKSRC that you just created.

5. Right-click QJUNKSRC.
6. Click **Delete** on the pop-up menu.  
The Delete Confirmation dialog opens.
7. Click **Delete**.

---

## Running commands from the Remote System Explorer

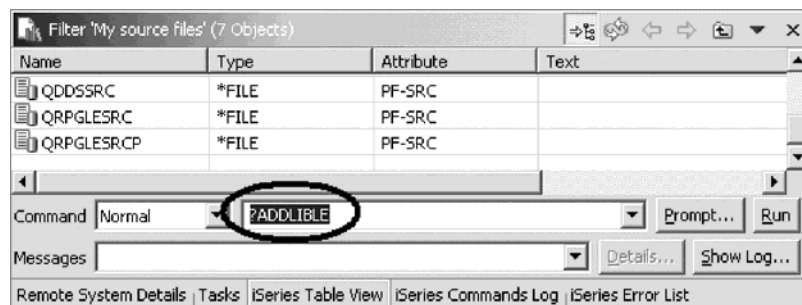
The Command entry is part of the Remote Systems view.

To run a command:

1. Check if you have an iSeries Table View tab in the bottom right pane where your command log appeared in the previous exercise.
2. If you have it click on it.
3. If you don't have it:
  - a. In the Remote Systems view, right-click **My source files filter**
  - b. Click **Show in table** on the pop-up menu.

Now in the iSeries Table view you can run commands on the iSeries server that the table is connected to. You can run commands in the Commands field beneath the iSeries Table view, and view messages in the Messages field. After you populate the table, you can enter a command and select either Prompt to specify parameters and then Run, or just select Run (which prompts and runs the command as normal). When you run a command, the Messages drop-down field is populated with the messages from the command. When you select a message, the Details button is enabled. When you press this button, the message and its help is displayed.

4. Type an iSeries command, for example ?ADDLIBLE.



The question mark is there to display a prompt screen.

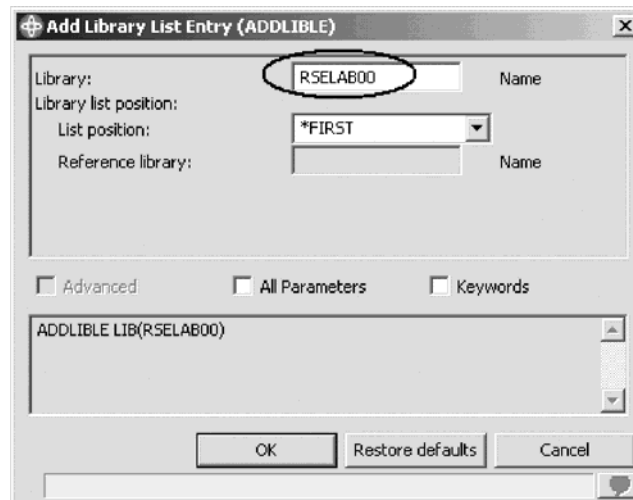
Instead of specifying a question mark you could use the Prompt push button.

5. Click **Run**.

The Command Prompt window for the ADDLIBLE command opens.

If you receive a connection message instead then you need to populate the table view with source members. In the Remote Systems view, right-click QRPGLSRC or QCBLLSRC and click Show in Table from the pop-up menu. Click Run again.

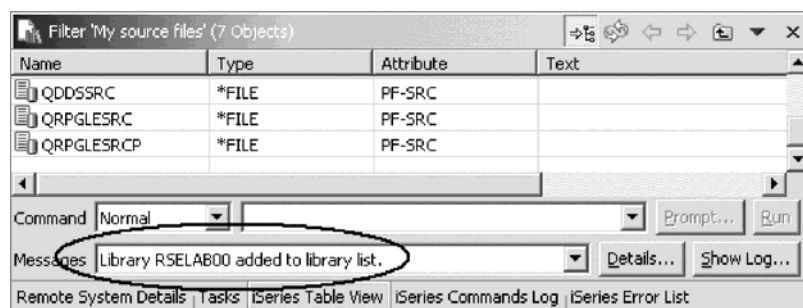
6. In the **Library** field, type RSELAB00 and click **OK**.



That will add this library to the library list of your Remote System Explorer job on the iSeries server.

**Note:** You may need to add a different library if the library RSELAB00 does not exist on your iSeries system.

The messages field will confirm the successful completion of this command. To get to the command history, similar to F9 on the 5250 screen, select the appropriate value from the **Command** list.



You could also use the iSeries Commands subsystem in the Remote Systems view underneath the iSeries Objects subsystem and run predefined commands or define your own commands.

---

## More practice

Given your own libraries on your iSeries system, create a member filter and a job filter. Then move libraries up, down and within your library list. Finally create a filter pool. Use the Development Studio Client for iSeries online help to assist you in these tasks.

---

## Recap

Congratulations! You have completed this module. You should now understand:

- The concepts of
- How to use the Remote System Explorer perspective to work with the iSeries objects that you used in the previous modules.
- How easy it is to perform actions and define your own actions
- How Remote System Explorer can organize and integrate your work and make that work easier

See how easy it is to be more productive building new or traditional iSeries applications. The Remote System Explorer gives you PDM-like access to OS/400® resources. The PDM like table view meant you could right-click on objects to perform actions that are the same as PDM's and use a command line at the bottom of the table just like PDM. There were also user-defined actions that supported all PDM substitution variables and of course the editing, verifying, compiling, running and debugging tasks which were tightly integrated with the Remote System Explorer.

You have been introduced to a highly integrated and productive environment, offering a consistent experience for all development work from RPG to Java. This new generation of tools offered rich support for exploring the file system, compiling, building, editing, running and debugging. The new tools allowed you to leverage the classic tools for extremely rich editing and DDS design support. The Remote System Explorer tools offered a superset of functions over the classic, and feature frozen ADTS tools. It offered significant productivity and usability gains, support for disconnected and team development and a common harness for the tight integration of IBM and partner-supplied tools for server application development. Using these new generation tools also implicitly increases your skills and will make the transition into new programming models easier, such as the IBM WebFacing Tool, Web, Web Services, Java and XML.

Step aside ADTS the traditional method for developing and maintaining server-side iSeries applications. Here comes Development Studio Client which includes new highly integrated and highly extendible tools for iSeries RPG, COBOL, C, C++, CL and DDS development. These new tools offered you a development experience that is consistent with the experience for developing Java, Web, Web Services, and XML applications, lowering the learning curve for all and making your transition to these e-business application programming models that much easier.



## Chapter 12. Module checkpoint answers

1	2	3	4	5	6	7	8
1b	1d	1c	1d	1d	1b	1c	1a
2g	2e	2a	2T	2d	2d	2T	2e
3g	3b	3f	3a	3g	3ac, bb, ca, dd, ef, fe, gg	3a	3a
4e	4aa, bd, cb, dc	4c	4T	4T	4e	4c	4T
5e	5e	5T	5d	5T	5T	5c	5f
6a		6e	6c	6T	6h	6T	6T
7T		7e	7b	7f	7T	7f	7T
8a		8c	8c	8a	8T	8T	
9T		9T	9c	9d	9T	9e	
10e		10c	10a	10d	10T	10a	
11b		11b	11d	11d	11T		
12f		12T	12c	12T			
13e		13d	13T				
14T			14d				
15T			15e				
16T			16d				
			17d				
			18b				
			19c				
			20T				
			21T				



---

## Appendix. Notices

Note to U.S. Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Lab Director  
IBM Canada Ltd. Laboratory  
8200 Warden Avenue  
Markham, Ontario  
Canada  
L6G 1C7

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 1992, 2004. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## Programming interface information

Programming interface information is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

---

## Trademarks

IBM  
iSeries  
OS/400  
RPG/400  
VisualAge  
WebSphere

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

ActiveX, Microsoft, SourceSafe, Visual C++, Visual SourceSafe, Windows, Windows NT<sup>®</sup>, Win32, Win32s and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group.

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







Printed in USA