IBM WebSphere Development Studio Client for iSeries

IBM

# Using the Remote System Explorer (RSE)

**Version 6.0.1**

IBM WebSphere Development Studio Client for iSeries

IBM

# Using the Remote System Explorer (RSE)
**Version 6.0.1**

# Contents

# Introduction

This tutorial teaches you how to maintain a payroll application written in ILE RPG using the Remote System Explorer.

You will learn how to start the product and open the Remote System Explorer perspective, use tools and views in this perspective to connect to an iSeries™ system and edit, verify, compile and debug a payroll application.

**Prerequisites**

To complete this tutorial, you should be familiar with:
* 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 will also help if you understand:
* ILE RPG
* DDS

The file(s) required for this tutorial are available for download at http://ibm.com/software/awdtools/wdt400/library.

**Time required**

To complete the modules of this tutorial, you will need approximately **3 hours**.

**Learning objectives**

This tutorial is divided into eight modules, each with its own learning objectives. You can choose to skip modules 7 and 8 and you can complete modules 5, 6, 7 and 8 in any order after module 4. Each module contains several exercises that must be completed in order for the tutorial to work properly.

Chapter 1, "Module 1. Starting the product and the Remote System Explorer," on page 1 teaches you about the workbench, a perspective, and specifically the Remote System Explorer perspective. In this module, you will:
* Start the product
* Specify the workspace for your project resources
* Open the iSeries default perspective

Chapter 2, "Module 2. Configuring a connection to an iSeries system and connecting to an iSeries," on page 7 teaches you how to create a connection to an iSeries server and select objects using the Remote System Explorer perspective. In this module, you will:
* Create a connection to an iSeries system
* Connect to an iSeries system
* Add a library to your library list
* View libraries in your job's library list from the Remote Systems view
* Find a source physical file in you library

- View members in a source physical file using the iSeries Table view
- Customize the columns in the iSeries Table view
- Open a member for edit from the iSeries Table view or the Remote Systems view
- Maximize the editor window
- Open another member for edit
- Open multiple views of a source member
- Switch from one edit session to another edit session
- Display a structural outline of items defined in a source member

Chapter 3, "Module 3. Editing source," on page 25 teaches you how to use the Remote Systems LPEX Editor to edit source. In this module, you will:
- Change the default settings of the LPEX Editor Parsers
- Change the color settings and font used by the Editor
- Change the default behavior of the Enter key
- Use SEU commands to edit source
- Undo and redo source changes
- View language sensitive help for the MOVE operation code
- View a list of all help contents
- Limit the search of help to specific documents
- Search the help
- Request a prompt for a specification line
- Display context sensitive help for any field in the iSeries Source Prompter
- View the beginning and ending of constructs in your source
- Use the Find and Replace window to search for an item in your source
- Filter or subset your source
- Filter lines based on line type
- Search through members in a source physical file
- Compare different versions of a program and identify the differences
- Syntax check source by line
- View help on syntax errors

Chapter 4, "Module 4. Verifying and compiling source," on page 63 teaches you how to verify and compile source in the Remote Systems LPEX Editor. In this module, you will:
- Check for semantic errors on your workstation
- Start the Program Verifier tool
- Use the iSeries Error list to locate each error in the source
- Use content-assist to fix an error
- Save your source
- Re-verify source
- Change compile preferences
- Invoke the compile command
- Change the current library using the Command field in the iSeries Table view
- Start an interactive connection
- Invoke the payroll program

Chapter 5, "Module 5. Debugging a program," on page 79 teaches you how to debug your payroll program from the workstation. In this module, you will:
- Invoke the debugger from the Launch Configurations window
- Add a breakpoint
- Add a conditional breakpoint
- Edit a breakpoint
- Monitor a variable through the Monitors view
- Step into your payroll program
- Show a Listing view
- List the call stack entries in the Debug view
- View all breakpoints
- Remove a breakpoint
- Monitor storage
- Set a Watch breakpoint
- Close the Debugger

Chapter 6, "Module 6: Exploring Remote System Explorer," on page 105 teaches you how easy it is to define filters, perform actions and define your own actions. In this module, you will:
- Know the features of Remote System Explorer
- Create a filter to show specific iSeries libraries
- Change the filter to add more iSeries libraries
- Create a filter to show all the source files in a library
- Access members to edit from your filter
- Create a user action that copies a source file with data to a new source file in the same library
- Specify user action parameters
- Specify a restriction on a user action
- Try the user action
- Run an OS/400® command from the iSeries Table view

Chapter 7, "Module 7. Designing screens," on page 123 teaches you how to use CODE Designer to modify a display file. In this module, you will:
- Open a DDS member for edit with CODE Designer
- Show file-level keywords and record-level keywords
- View the details of records, record-level keywords and field-level keywords
- View the design of the payroll application main menu
- Create a group from an existing record format
- Create a new group and add a subfile record and a subfile control record
- Add columns to the subfile record
- Add fields to the subfile control record
- Copy existing fields
- Set indicators to handle field errors
- View and update record and field properties
- View keywords and the properties of a keyword
- Insert a keyword
- View help for a keyword

- Check there are no semantic errors in the DDS source
- View help for an error
- Launch the editor in read mode from the error list
- Launch the editor in write mode to fix the error
- Find a keyword in the source
- Save source changes
- Compile your source changes
- Close the Designer

Chapter 8, "Module 8. Introducing the product and Remote System Explorer (optional)," on page 149 teaches you about IBM® WebSphere® Development Studio Client for iSeries and describes its relationship to IBM WebSphere Development Studio for iSeries. In this module, you will:
- Know the goals of the product
- Know the editions of the product
- Identify the host tools and the client tools
- List and describe the iSeries application development tools

When you are ready, begin Chapter 1, "Module 1. Starting the product and the Remote System Explorer," on page 1.

# Chapter 1. Module 1. Starting the product and the Remote System Explorer

This module teaches you about the workbench, the workspace, a perspective and specifically the Remote System Explorer perspective.

In this module, you will:
- Start the product
- Set the default workspace
- Access unique tools and views targeted towards iSeries application development tasks

**Requirement:** Before beginning this module, you should have the prerequisite knowledge outlined in "Introduction" on page v.

**Exercises**

The exercises within this module must be completed in order:
- "Exercise 1.1: Starting the product"
- "Exercise 1.2: Opening the Remote System Explorer perspective" on page 3

**Time required**

This module will take approximately **10 minutes** to complete.

## Exercise 1.1: Starting the product

If you want to know more about the product before you get started you can read Chapter 8, "Module 8. Introducing the product and Remote System Explorer (optional)," on page 149.

First you must start the product. Follow these steps to start the product:

1. Click **Start** on the task bar of your desktop.
2. Select **Programs > IBM Rational® > IBM WebSphere Development Studio Client for iSeries V6.0 > WebSphere Development Studio Client for iSeries**

   IBM WebSphere Development Studio Client for iSeries V6.0 ► WebSphere Development Studio Client for iSeries

   If you are working with the Advanced Edition of the product you will see the words Advanced Edition in the product name.

   If you are working with the WDSC Lite version you will see the following start menu option.

   IBM WebSphere Development Studio Client Lite technology preview V6.0.1 ► WDSC Lite

3. A dialog will appear. Here you specify the directory of the workspace where your projects and other resources such as folders, subfolders and files that you are developing in the workbench will reside.
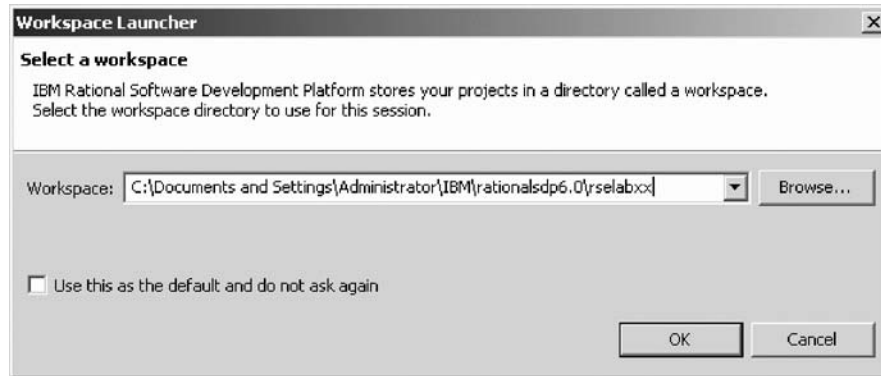
4. (Optional) Change the field in this dialog and use a unique directory name, for example, RSELABxx (where xx is a unique number).

5. Click **OK** to open the workbench.



6. Click the icon in the far right of the Welcome page to go to the Workbench.

7. Click the maximize button to maximize the workbench.



8. Click the **X** in the Welcome tab to close the Welcome view.

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 views and an editor.

You have started the product and now you are ready to begin "Exercise 1.2: Opening the Remote System Explorer perspective."

## Exercise 1.2: Opening the Remote System Explorer perspective

Before you begin, you must complete "Exercise 1.1: Starting the product" on page 1.

In these steps you will open the Remote System Explorer perspective.

1. Check for the name of the 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 a program. Perspectives contain views and editors and control what appears in certain menus and tool bars.

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.

You work in the Remote System Explorer perspective in the workbench. 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 the product 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 use the views from the workbench.

Now you are ready to review your knowledge of this module by taking the quiz. You can also apply what you have learned in this module by completing the practice tasks detailed in More practice.

**Quiz**

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.
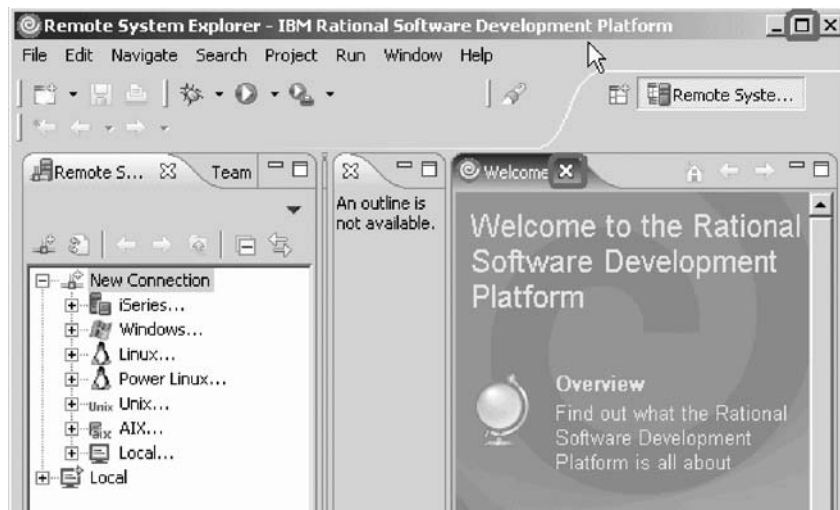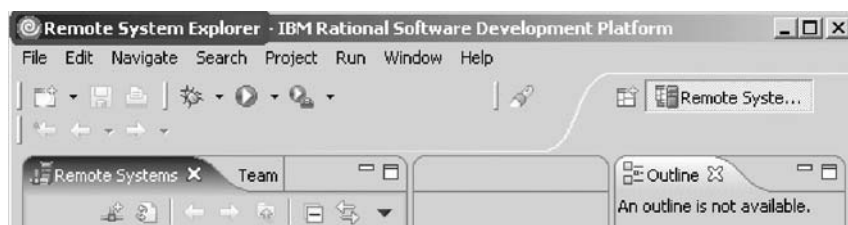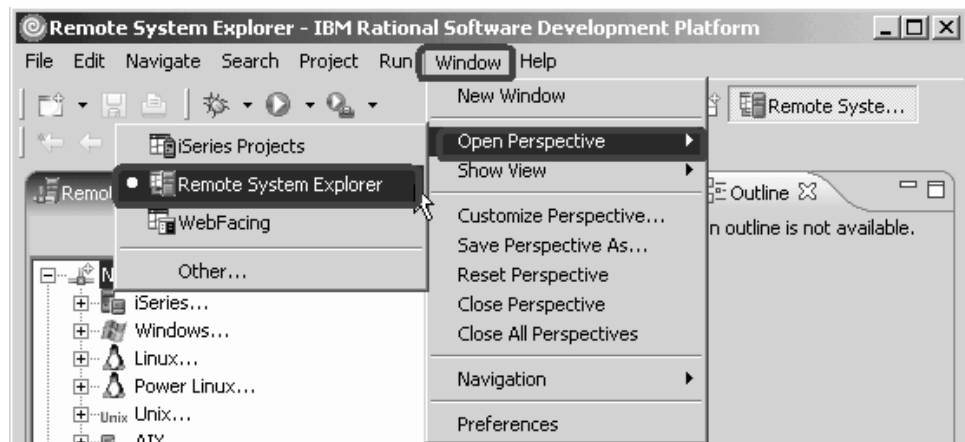
2. A workbench:
   a. Aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workbench resources.
   b. Defines the initial set and layout of views in the Workbench window.
   c. Refers to the desktop development 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
3. 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.
4. Match the perspective with its correct definition:
   a. Combines tools and views that you would commonly use while editing Java source files
   b. Contains tools and views that you would use while debugging programs
   c. Contains tools and views that you would use while developing Web applications
   d. Contains tools and views that you would use while maintaining iSeries applications.
   a. Java perspective
   b. Web perspective
   c. Remote System Explorer perspective
   d. Debug perspective
5. 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**

Given your experience in opening the Remote System Explorer perspective, now open the Web perspective. See the list of tools and views available to the Web developer. Next open the Java perspective. See the 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 top-right frame for the Web perspective icon. Now close both the Java perspective and the Web perspective.

**Module recap**

You have completed Chapter 1, "Module 1. Starting the product and the Remote System Explorer," on page 1. You have learned how to:

- Start the product
- Set the default workspace
- Access unique tools and views targeted towards iSeries application development tasks

Now that you have started the product and have opened the Remote System Explorer perspective, you can move on to getting connected to an iSeries system. Continue to Chapter 2, "Module 2. Configuring a connection to an iSeries system and connecting to an iSeries," on page 7.

# Chapter 2. Module 2. Configuring a connection to an iSeries system and connecting to an iSeries

This module teaches you how to create a connection to an iSeries server, find a library in your library list, select objects from a library and finally 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 this module, you will:
- Create a connection to an iSeries system
- Connect to an iSeries system
- Add a library to your library list
- View libraries in your job's library list from the Remote Systems view
- Find a source physical file in your library
- View members in a source physical file using the iSeries Table view
- Customize the columns in the iSeries Table view
- Open a member for edit from the iSeries Table view or the Remote Systems view
- Maximize the editor window
- Open another member for edit
- Switch from one edit session to another edit session
- Open multiple views of the same source member
- Display a structural outline of items defined in a source member

**Exercises**

The exercises within this module must be completed in order:
- "Exercise 2.1: Configuring a connection to an iSeries system"
- "Exercise 2.2: Connecting to an iSeries system" on page 10
- "Exercise 2.3: Viewing and accessing objects in the Remote System Explorer perspective" on page 14
- "Exercise 2.4: Opening a second source member and multiple views" on page 19
- "Exercise 2.5: Displaying an outline of a source member" on page 21

**Time required**

This module will take approximately **10 minutes** to complete.

## Exercise 2.1: Configuring a connection to an iSeries system

When you first open the Remote System Explorer, 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 specify a profile. All connections, filters, and filter pools belong to profiles. Filters are described in a later exercise. Profiles are discussed 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.

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 System Explorer.
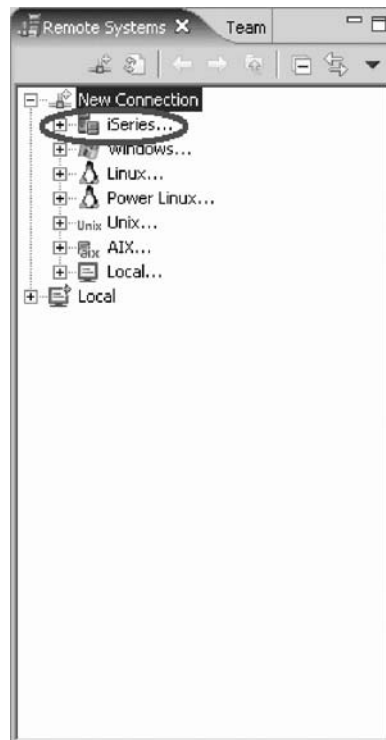


Click the plus sign + beside **iSeries** to configure a connection to an iSeries system.

The Name personal profile page opens.
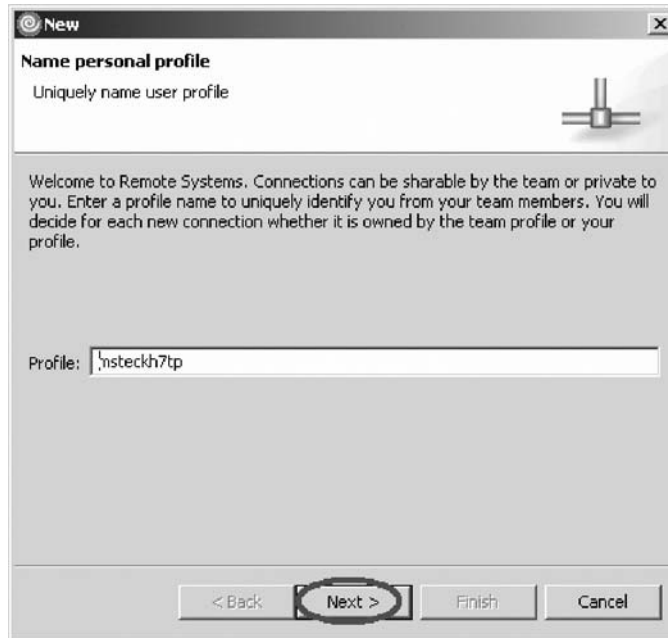
2. Click **Next** to accept the default value.

The profile defaults to the name of the workstation. Your profile will be different from the one shown here.

The Remote iSeries System Connection page opens.
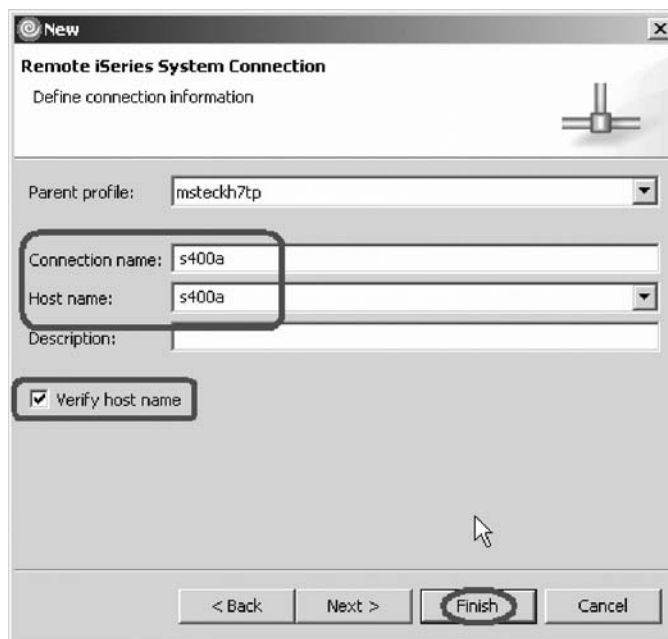


On this second page you specify the information for your connection. The cursor on this page is positioned in the **Host Name** field.

3. In the **Host name** field, type the IP address or 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.

5. Leave the **Verify host name** check box selected.

6. Click **Finish** to define your system.

You have configured a connection and now you are ready to begin "Exercise 2.2: Connecting to an iSeries system."

## Exercise 2.2: Connecting to an iSeries system

Before you begin, you must complete "Exercise 2.1: Configuring a connection to an iSeries system" on page 7.

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

**iSeries Objects**
> A PDM-like group, allowing access to libraries, objects and members.

**iSeries Commands**
> Contains predefined commands and allows you to create 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 iSeries Commands log.

**iSeries Jobs**
> Allow you to see various jobs, subset by job attributes, and to perform a 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.
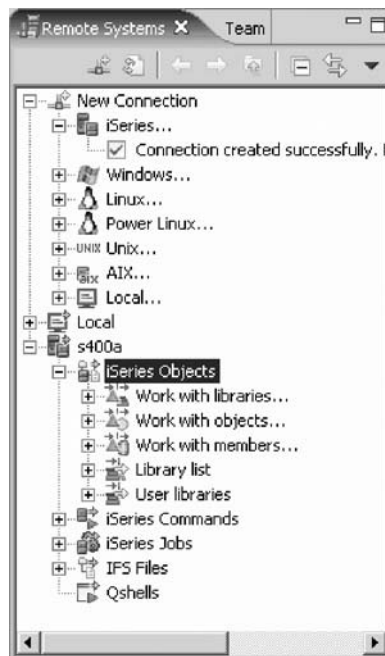


To connect to an iSeries system:
1. In the Remote Systems view, your new connection is expanded to reveal your subsystems. The **iSeries Objects** subsystem is the subsystem you will use most

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 you can start with the pre-defined Library list filter, that when expanded lists all libraries in your library list.)
- **User libraries** (allows you to work with all user libraries you can access on that iSeries server.)

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.
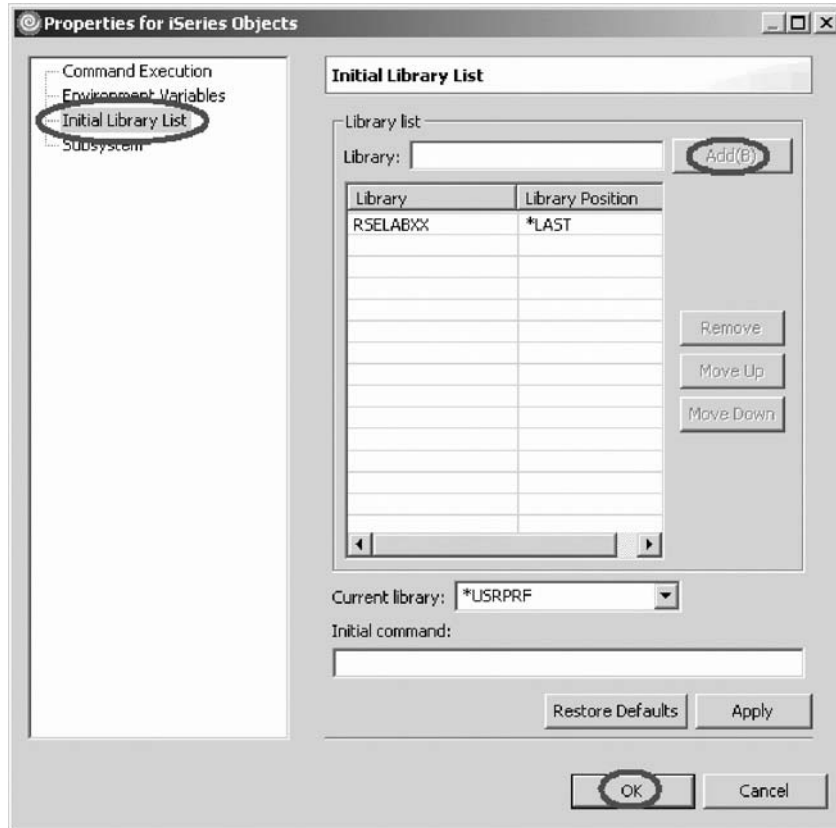
Now let's work with a library in your library list and add the library that you'll be using in this tutorial:

a. Right-click **iSeries Objects** and click **Properties** on the pop-up menu.
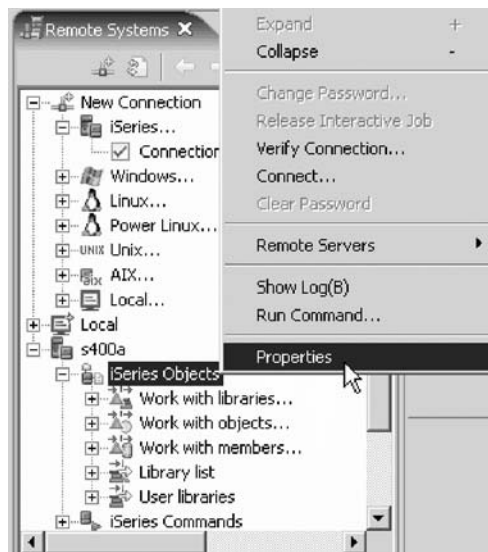


b. Select **Initial Library List** on the left pane.

c. Type RSELABxx where XX is a unique number in the **Library** field and click **Add**.

Properties for iSeries Objects

Command Execution
Environment Variables
Initial Library List
Subsystem

Initial Library List

Library list

Library: [          ]          Add(B)

| Library | Library Position |
|---------|------------------|
| RSELABXX | *LAST |

Remove

Move Up

Move Down

Current library: *USRPRF

Initial command:

[                    ]

Restore Defaults    Apply

OK    Cancel

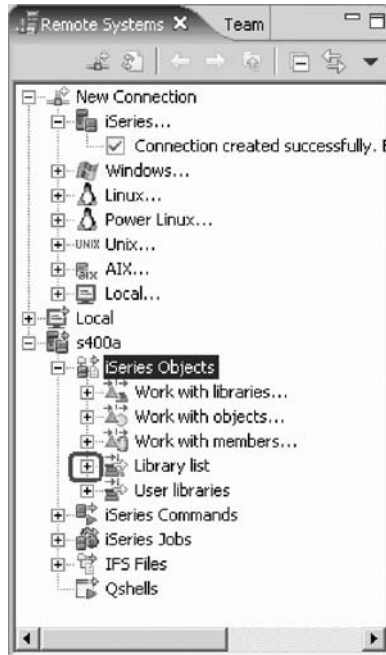d. Click **OK**.

This will add the library RSELABxx to your library list every time you use
this connection.

**Note:** You can also change your library list using the pop-up menu items
**Add Library List Entry** or **Change Current Library** on the **Library
list** folder in the iSeries Objects subsystem. These changes are only
valid until you disconnect.

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 **Save user ID** check box.
6. Select the **Save password** check box.
7. Click **OK**.

   As you know, you can use the properties of any of the subsystems to set connection information such as adding a library to a library list.

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

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

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. Common actions like delete, move, copy, etc. are valid for all kinds of objects.

You have connected to an iSeries system and now you are ready to begin "Exercise 2.3: Viewing and accessing objects in the Remote System Explorer perspective."

## Exercise 2.3: Viewing and accessing objects in the Remote System Explorer perspective

Before you begin, you must complete "Exercise 2.2: Connecting to an iSeries system" on page 10.

Now you are ready to view and access objects in your 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 example, for a source file the pop-up menu has an action to create a new member within the selected file.

2. Drill-down through the files in the Remote Systems view until you find QDDSSRC source file and then expand it.

3. Scroll down through the files in the Remote Systems view until you find QRPGLESRC source file and expand it as well.
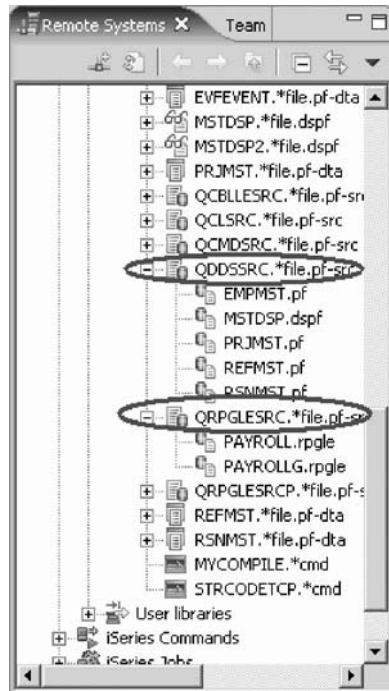
Now you can see and access the members in these two source files. For each member you can right-click and select from a number of actions. The exact list of actions depends on whether the member is a data file or source file and whether you select one or multiple members. For a RPG source member, the pop-up menu actions include:

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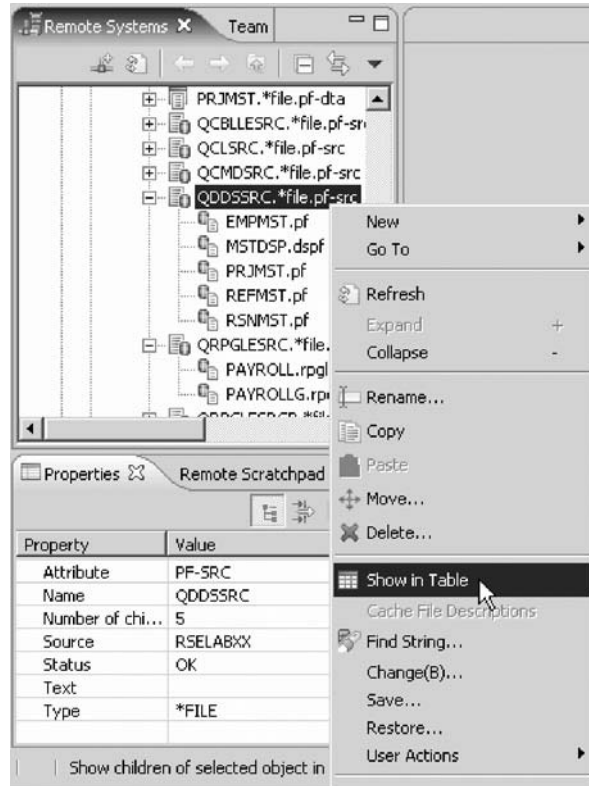
The iSeries Table view takes the selected object in the Remote Systems view as input, and displays the contents in the table. For source physical files, this step displays the members inside, their names, types, attributes, text descriptions, and status.
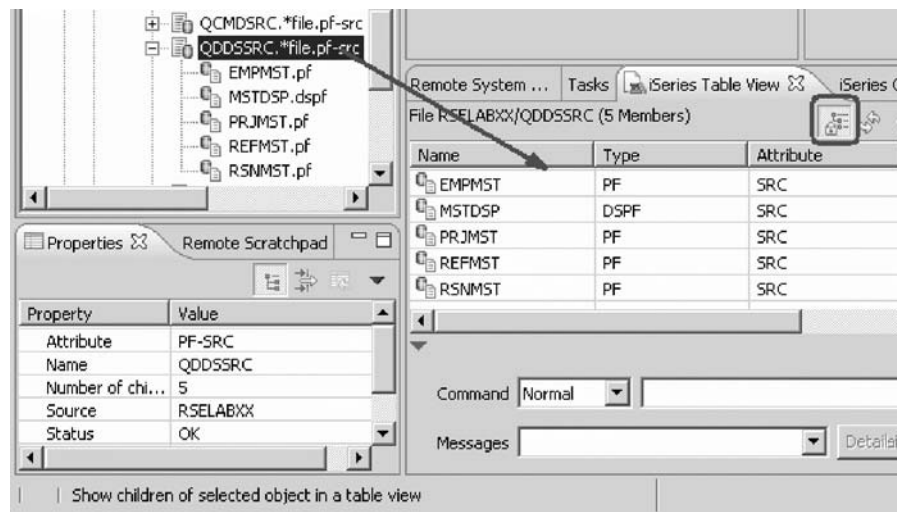


The top of the iSeries Table view contains a lock icon  that controls the correlation between the Remote Systems view and the iSeries Table view. If the lock is disabled then whenever you click an object or library in the Remote Systems view, the associated contents of that item automatically populate the iSeries Table view. If the lock is enabled then when you click on various items in the Remote Systems view, this view does not change the

content of 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 [icon] is in the unlock position. Leave the mouse pointer over the tool button for a second or two to display the flyover help. That way you can check if the view is locked or unlocked.

   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 the pop-up menu for an object in the Remote Systems view and to select Show in Table.

   You can also modify which specific columns you want to see in the iSeries Table view.

6. To modify the iSeries Table properties:

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

      The Preferences Window opens.
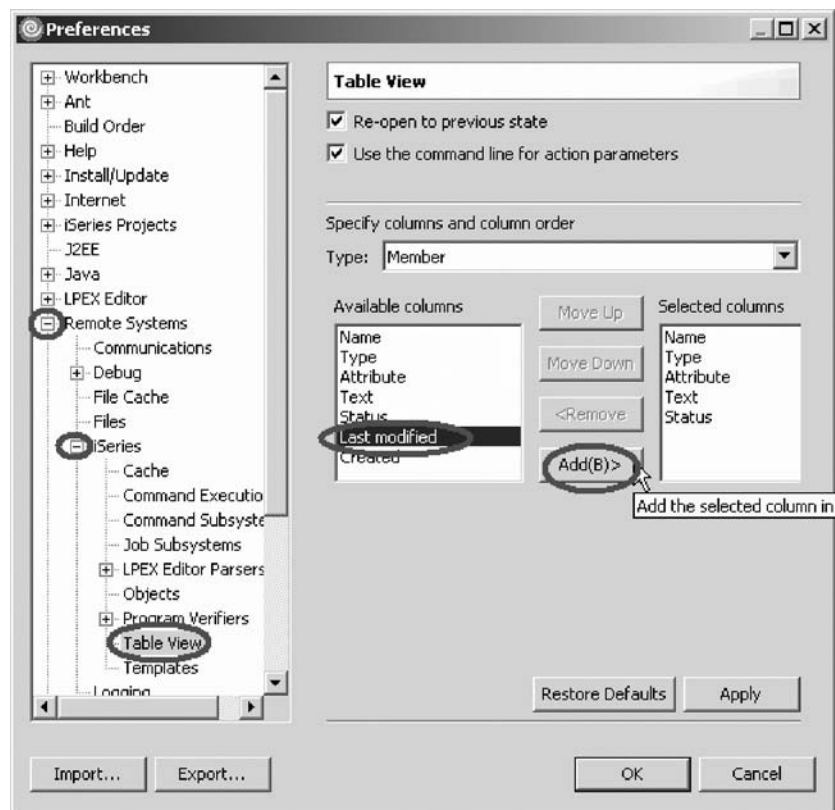


   b. In the left pane of the Preferences window, expand **Remote Systems**.

   c. Expand **iSeries** under Remote Systems.

   d. Click **Table View** under iSeries.

   e. In the right pane of the Preferences window, select **Last modified** in the **Available columns** list.

   f. Click the **Add** button.

   g. Click **OK**.

      Now, let's update the iSeries Table view.

   h. Click the down arrow on the iSeries Table view title bar.
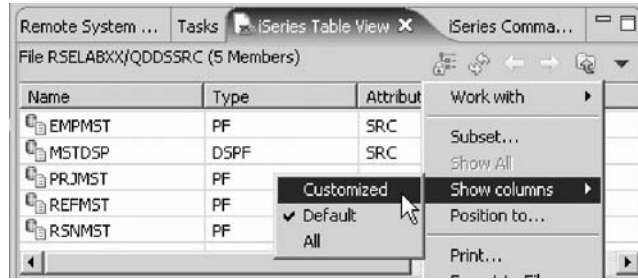
i. Click **Show columns > Customized** in the pop-up menu.

Now you'll see the extra column that you've added.
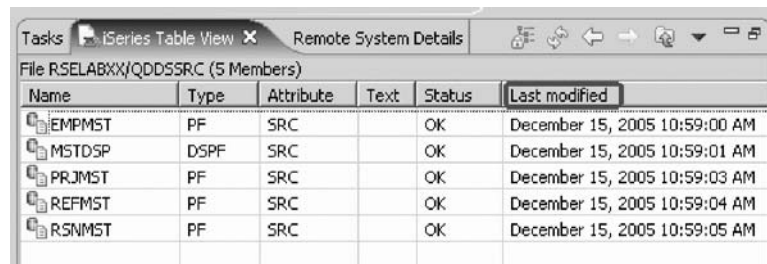


You can also sort the objects in the iSeries Table view by column.

j. To sort the objects in ascending order by Last modified, click on the heading.
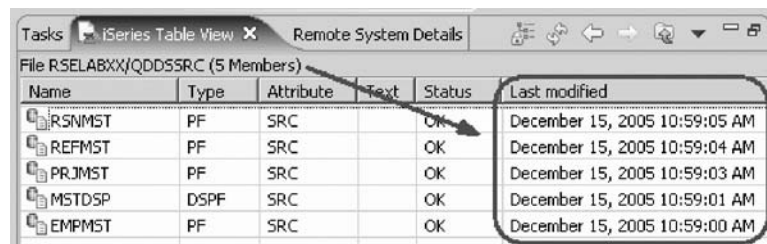


k. If you click the heading the second time, it will sort it in descending order.

7. In the Remote Systems view, select QRPGLESRC.

The table shows the members in QRPGLESRC.
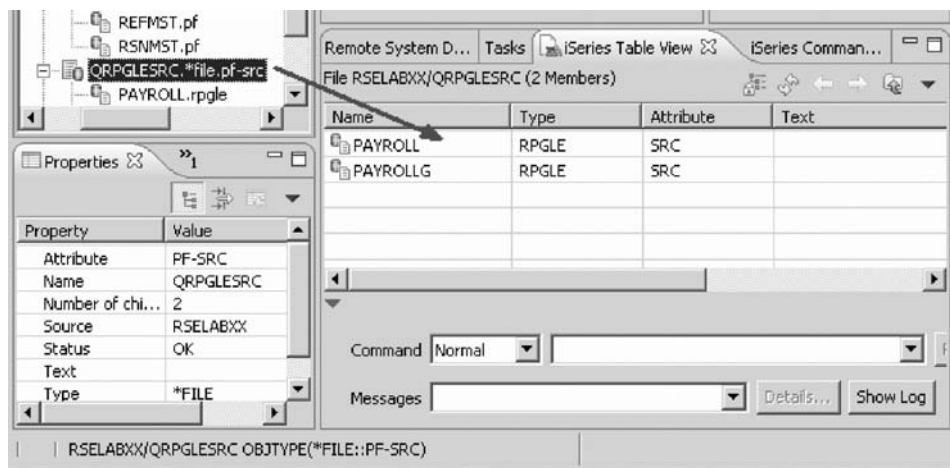


Now you are ready to use the Remote Systems LPEX Editor to edit the member MSTDSP found in QDDSSRC.

8. 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 from SEU, and the reference manuals are built-in and F1 cursor sensitive.

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

10. Double-click the **MSTDSP** tab again to return the view to its original size.



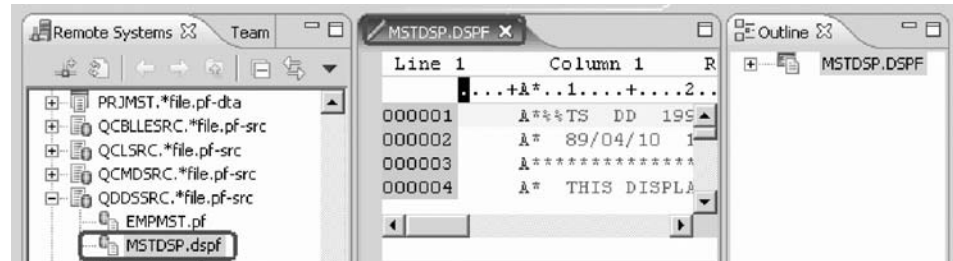You have viewed and accessed objects in your library and now you are ready to begin "Exercise 2.4: Opening a second source member and multiple views."

## Exercise 2.4: Opening a second source member and multiple views

Before you begin, you must complete "Exercise 2.3: Viewing and accessing objects in the Remote System Explorer perspective" on page 14.

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 QRPGLESRC source file.

   This member will be loaded into the editor as well.

   Your Editor window will look something like:
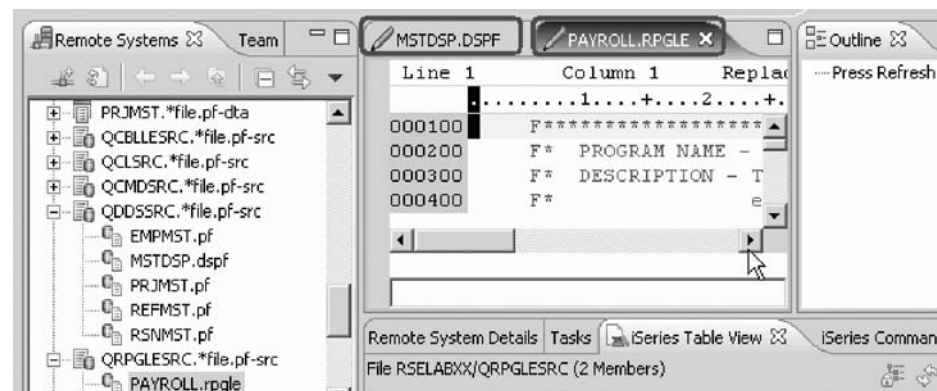


   Notice the two tabs in the Editor window.

2. Click on each tab to switch from one edit session to another edit session.

**Opening multiple views of a source member**

You can open multiple views of the same source member while editing in the Remote Systems LPEX editor.

To see multiple views of your RPG source:

1. Double-click the **PAYROLL** tab in the editor to maximize the Editor window.
2. Right-click this source in the Editor view and click **View > Open new view**.

   **Note:** You can open a maximum of five views of the same source.
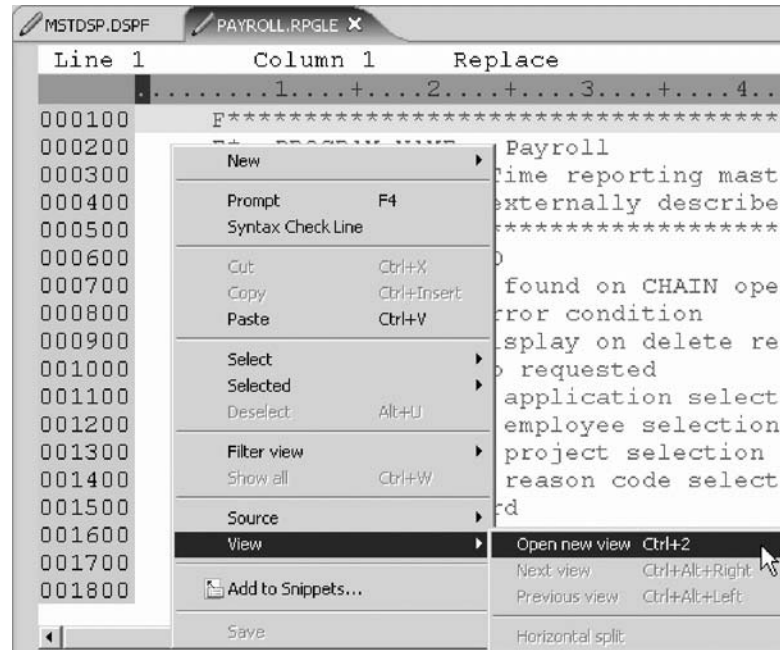


3. Right-click a source view and select **View > Next view** or **View > Previous view** to navigate among the views.
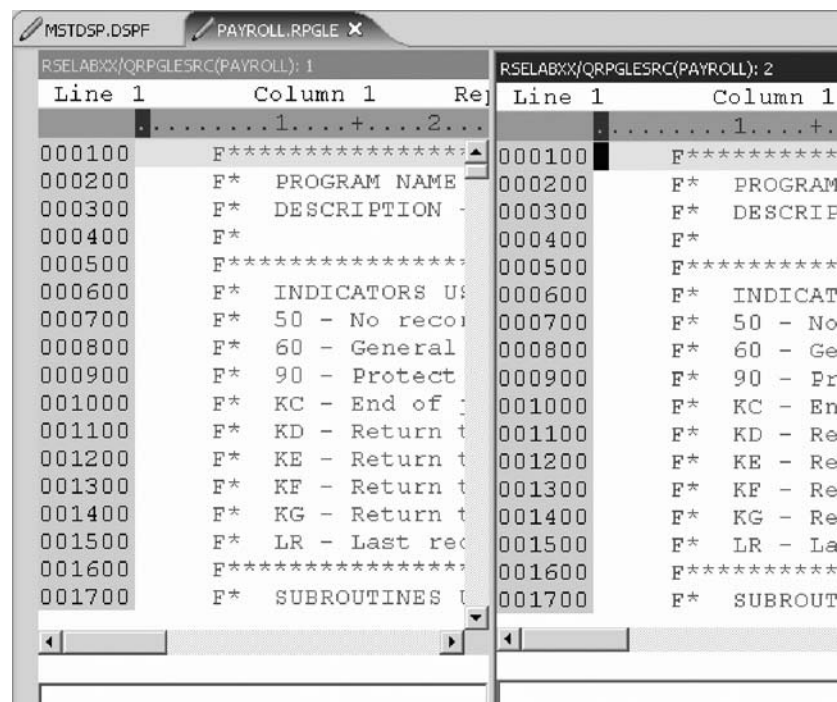
4. Right-click a source view and select **View > Horizontal split** to change from a vertical split of the views to a horizontal split of the views.
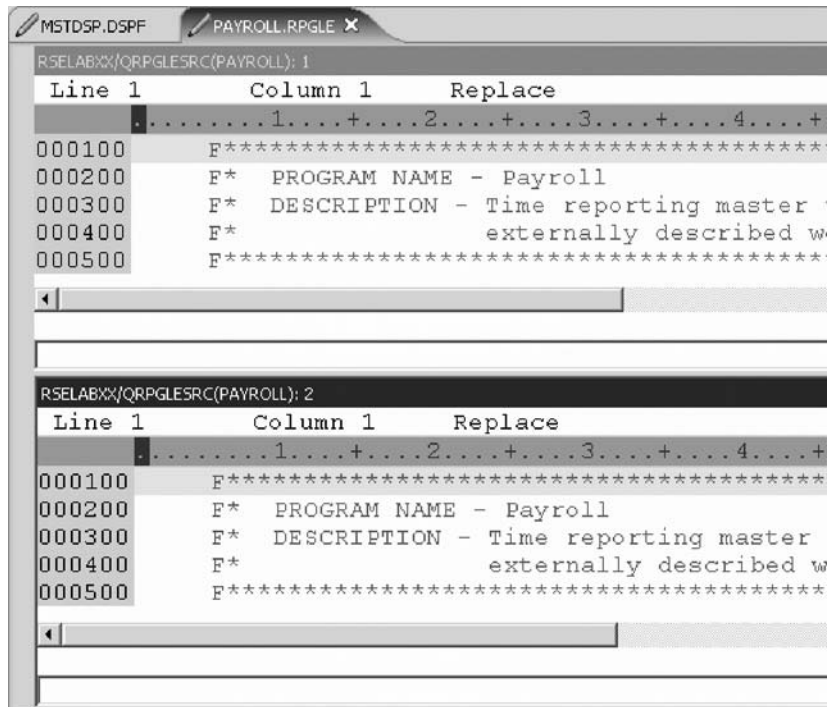


5. Right-click the source view that you want to close (RSELABXX/ QRPGLESRC(PAYROLL): 2) and select **View > Close view**.

   **Note:** This option is not available on the first view.

You have opened another member for edit and seen multiple views of a member and now you are ready to begin "Exercise 2.5: Displaying an outline of a source member."

## Exercise 2.5: Displaying an outline of a source member

Before you begin, you must complete "Exercise 2.4: Opening a second source member and multiple views" on page 19.

The Outline view acts as an excellent resource when you want to edit RPG, COBOL and DDS source in the Remote Systems LPEX editor. The Outline view displays a structural outline of items defined in the file that you currently have open in the Remote Systems LPEX editor window. With the editor active, you can expand the file structure in the Outline view, and click various elements in the view to jump to that location in the source itself.

To see an Outline view of your RPG source:

1. Click the **PAYROLL** tab in the editor and click **Refresh** on the Outline view toolbar.

   **Note:** If you have closed the Outline view, you can reset the perspective by selecting **Window > Reset perspective** from the workbench menu or **Show view > Other** then expand **Basic** and click **Outline** in the Show view dialog.

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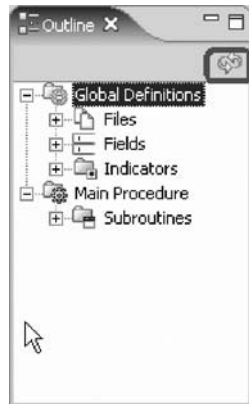


Now you want to see more details of your source member.

2. Expand Files.
3. Expand the MSTDSP workstation file.
4. Expand the EMPSEL record format.
5. Double-click on any of the entries in the Outline view.

   This will position the source editor accordingly.
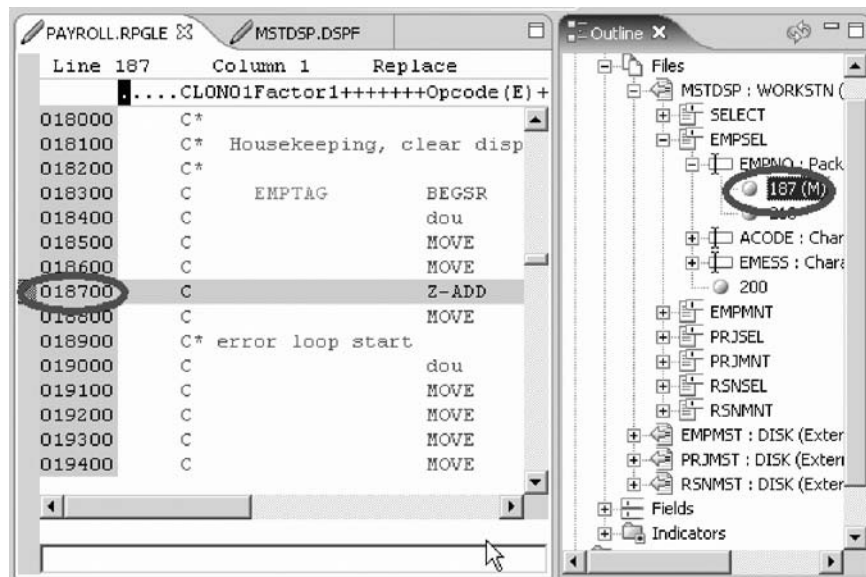


6. Stay in the **PAYROLL** tab to get the PAYROLL Editor window in focus for the next exercise.

Now you are ready to review your knowledge of this module by taking the quiz. You can also apply what you have learned in this module by completing the practice tasks detailed in More practice.

**Quiz**

1. When you first open Remote System Explorer, 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.　Subsystems include:

　a.　iSeries Objects

　b.　iSeries Jobs

　c.　IFS Files

　d.　iSeries Commands

　e.　All of the above

3.　The subsystem iSeries objects includes:

　a.　Work with libraries

　b.　Work with objects

　c.　Work with members

　d.　Library list

　e.　All of the above

4.　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

5.　The lock icon controls the correlation between the Remote Systems view and the iSeries Table view. (T, F)

6.　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).

7.　The Outline view:

　a.　Displays a structural outline of items defined in the file that you currently have open in the editor

　b.　Lists structural elements

　c.　Shows editor specific contents and toolbar

　d.　All of the above

**More practice**

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 properties of the connection. Then use the iSeries objects subsystem to list the libraries in your library list. Use the iSeries Table 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 product online help to assist you in these tasks.

**Module recap**

You have completed Chapter 2, "Module 2. Configuring a connection to an iSeries system and connecting to an iSeries," on page 7. You have learned how to:

• Create a connection to an iSeries system

• Connect to an iSeries system

• Add a library to your library list

• View libraries in your job's library list from the Remote Systems view

- Find a source physical file in your library
- View members in a source physical file using the iSeries Table view
- Customize the columns in the iSeries Table view
- Open a member for edit from the iSeries Table view or the Remote Systems view
- Maximize the editor window
- Open another member for edit
- Open multiple views of the same source member
- Switch from one edit session to another edit session
- Display a structural outline of items defined in a source member

Now that you have a connection to an iSeries server and have opened a source member for edit, you can learn more about the Remote Systems LPEX Editor. Continue to the next module Chapter 3, "Module 3. Editing source," on page 25.

# Chapter 3. Module 3. Editing source

This module teaches you how to edit ILE RPG source member PAYROLL, which should already be open, and learn about some of the Remote Systems LPEX Editor's language features.

In this module, you will:
- Change the default settings of the LPEX Editor Parsers
- Change the color settings and font used by the Editor
- Change the default behavior of the Enter key
- Use SEU commands to edit source
- Undo and redo source changes
- View language sensitive help for the MOVE operation code
- View a list of all help contents
- Limit the search of help to specific documents
- Search the help
- Request a prompt for a specification line
- Display context sensitive help for any field in the iSeries Source Prompter
- View the beginning and ending of constructs in your source
- Use the Find and Replace window to search for an item in your source
- Filter or subset your source
- Filter lines based on line type
- Search through members in a source physical file
- Compare different versions of a program and identify the differences
- Syntax check source by line
- View help on syntax errors

**Exercises**

The exercises within this module must be completed in order:

**Time required**

This module will take approximately **45 minutes** to complete.

**Tip:** You might want to maximize the Editor window during this module.

## Exercise 3.1: 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 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 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 that simplify text insertions and deletions
- On-line language reference

Now you know the features of the editor, you are ready to begin "Exercise 3.2: Changing default editor settings."

## Exercise 3.2: Changing default editor settings

Before you begin, you must complete "Exercise 3.1: Introducing the editor."

The LPEX Editor has predefined settings, but also has an associated preferences page containing settings that you can modify. The name of the category is LPEX Editor and it appears in the left pane of the Preferences window.
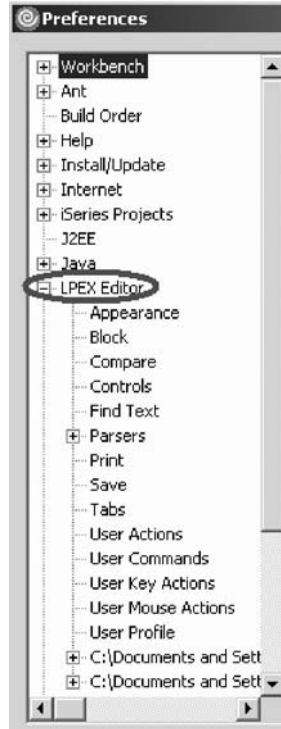
You will change the default settings of LPEX Editor Parsers, Appearance and User Key Actions.

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.

**Setting column sensitive editing**

To set preferences:
1. Click **Window > Preferences** from the workbench menu.
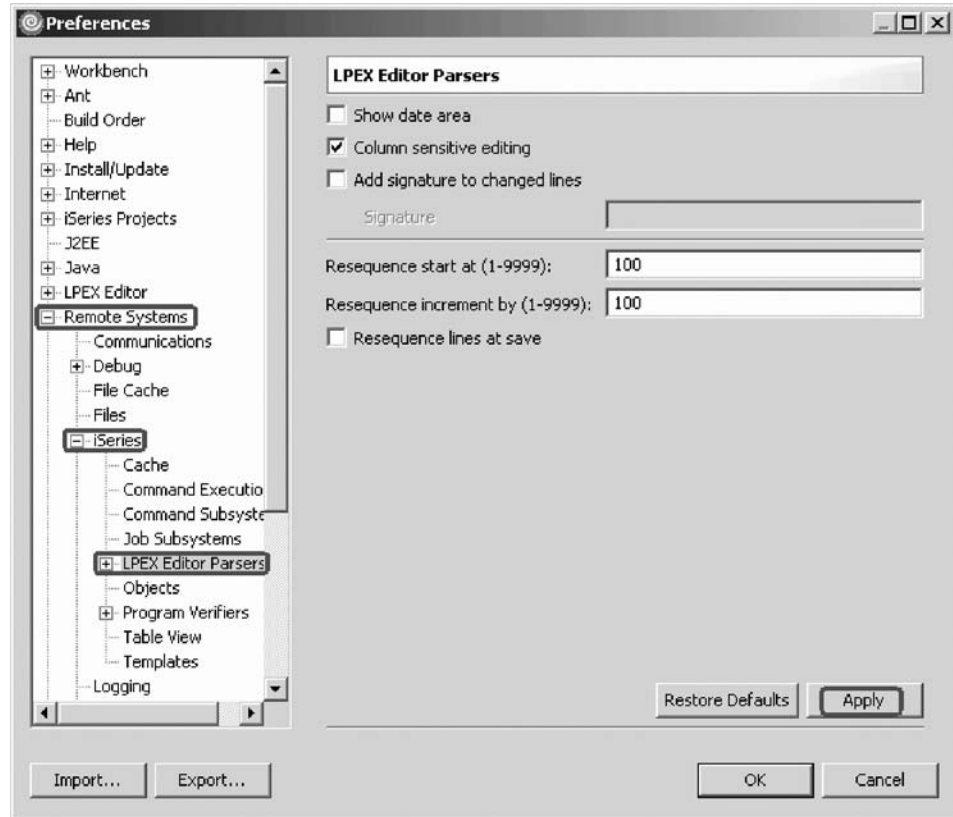
   The Preferences window opens.

2. In the left pane of the Preferences window, expand **Remote Systems**.
3. Expand **iSeries** under Remote Systems.
4. Select **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. Click **Apply**.

   The Preferences window shows.

   Other interesting preference settings are located under LPEX Editor. Appearance allows you to modify color settings and font used by the Editor.

**Changing the editor appearance**
1. In the left pane of the Preferences window, expand **LPEX Editor**.
2. Select **Appearance** under LPEX Editor.
3. Select **formatLine** under the **Styles** list.
4. Change the **Foreground** color to dark green.
5. Change Font to 12.
6. Change **Background** color to light green.

   Notice how your changes are reflected in the sample edit view.
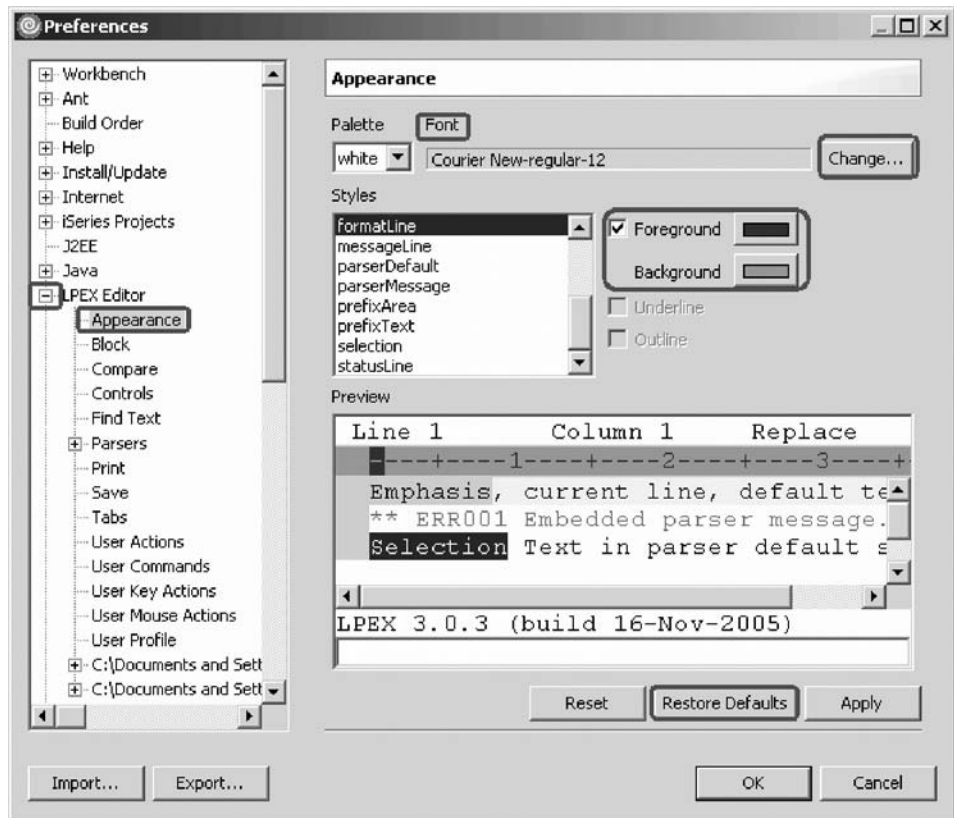
7.  Select **currentLine** under the **Styles** list.

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

8.  Change the **Background** color to light yellow.

9.  If you don't like the changes you made, you can click **Restore Defaults** to return to the original settings.

**Modifying the default behavior of the Enter key**

To modify the default behavior of the Enter key to split the current line in the source instead of creating a new line:

1.  Expand **LPEX Editor** if not already expanded then select **User Key Actions**.
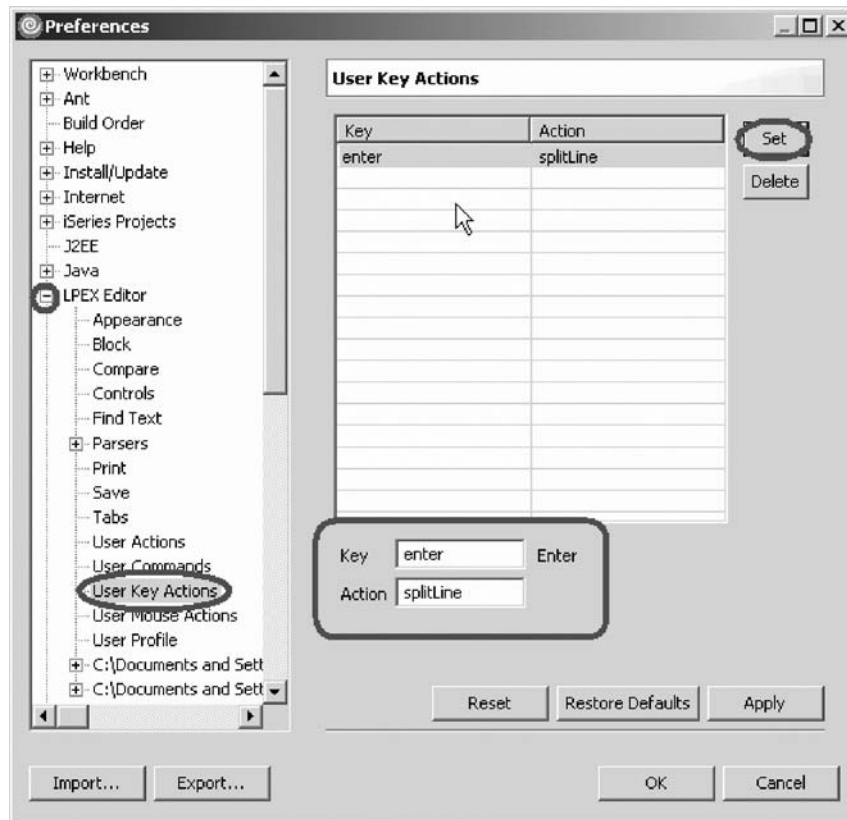
2. Type enter in the **Key** field.

   **Note:** The Key and Action fields are case sensitive. Make sure that the values typed in the Key and Action fields are exactly as shown above.

3. Type splitLine in the **Action** field.

4. Click **Set**.

   The Remote System Explorer has a predefined behavior for the Enter key. For the Enter key to use the new split line action you need to change the default Remote System Explorer behavior for the ILE RPG language.
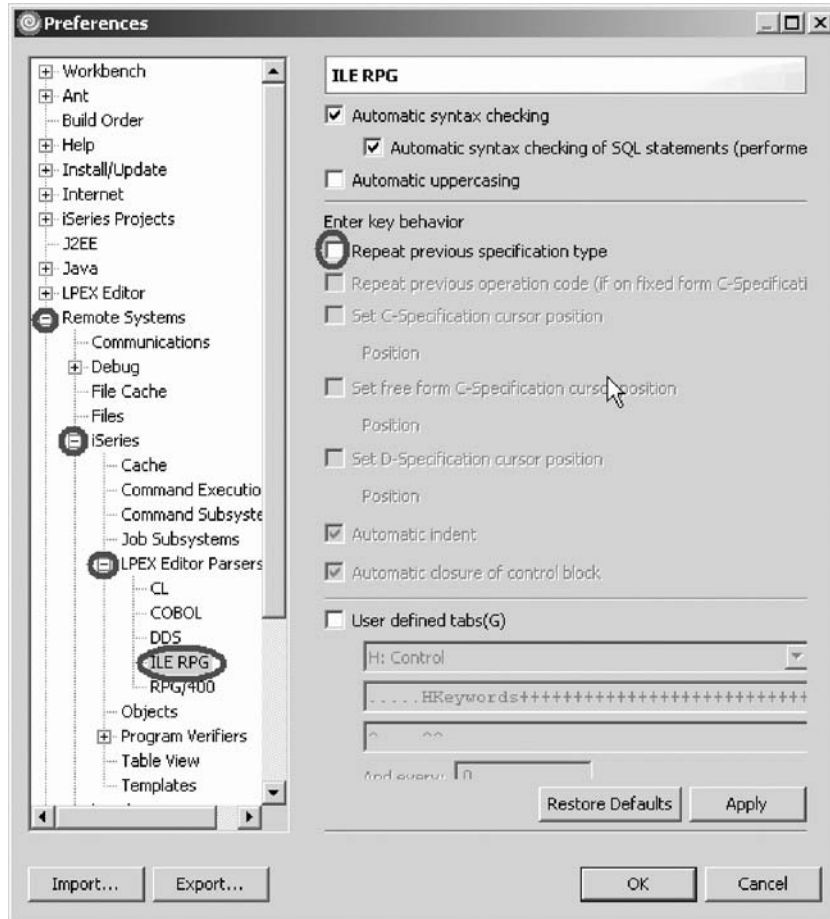
5. Expand **Remote Systems**, then **iSeries** if not already expanded and then **LPEX Editor Parsers**.

6. Select **ILE RPG**.

7. Clear the **Repeat previous specification type** check box.

8. Click **OK** on the Preferences window.

   Return to the Editor window.

**Seeing the results of column sensitive editing**

Notice that the line the cursor is on is now highlighted in the color you selected for it in the preference setting for highlight current line.
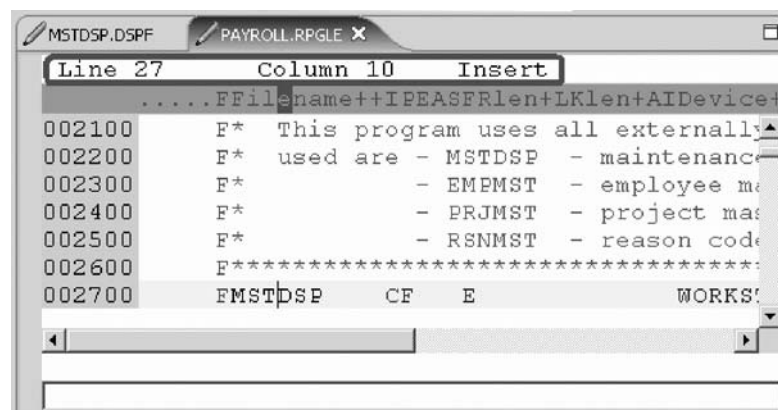
To see what column sensitive editing does:

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

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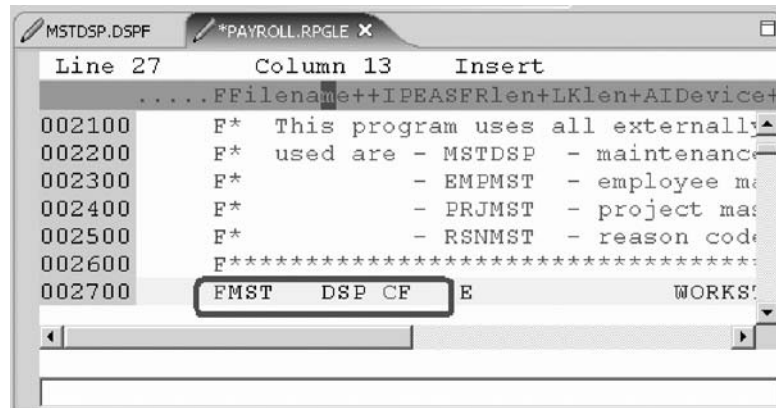


4. Press the **Backspace** 3 times.

   Once again the filename is shifted but no other columns are effected.

**Seeing the results of splitLine**

To see what splitLine does:

1. Place the cursor somewhere on a line and press **Enter**.

   The text to the right of the cursor is moved to the next line.

**Autosaving while editing source**

To enable or disable autosave while editing source in the Remote Systems LPEX Editor, you need to change an editor preference. You use the **Window > Preferences > Remote Systems > LPEX Editor** and click the check box **Autosave**. The default value for the minutes is set to 5. You can specify a value between 1 and 60 minutes.
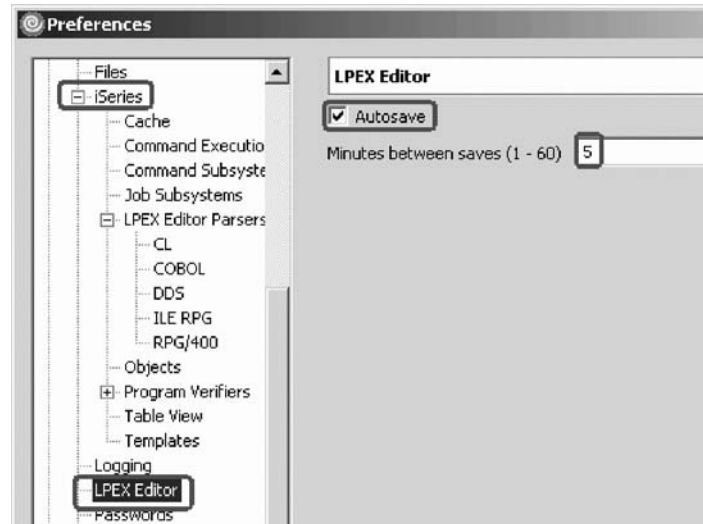
You have changed the default editor settings and now you are ready to begin "Exercise 3.3: Entering SEU commands."

## Exercise 3.3: Entering SEU commands

Before you begin, you must complete "Exercise 3.2: Changing default editor settings" on page 26.

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 a 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.

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.

You have learned how to use SEU commands in the editor and now you are ready to begin "Exercise 3.4: Requesting undo and redo operations."

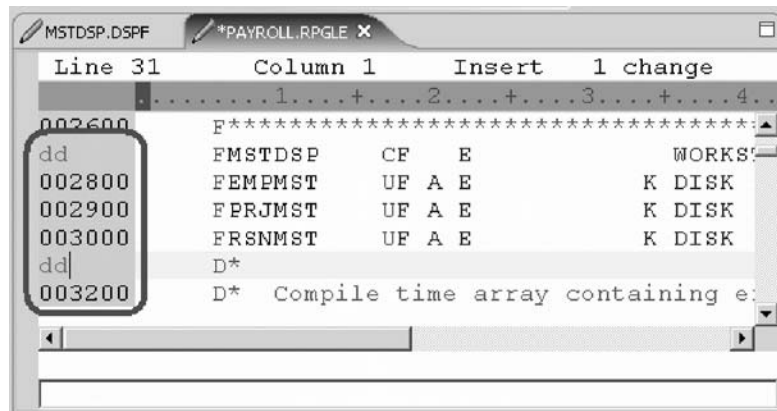## Exercise 3.4: Requesting undo and redo operations

Before you begin, you must complete "Exercise 3.3: Entering SEU commands" on page 33.

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 Undo operation by using the Redo operation. Finally you will reload the source so that it is back to its original content.

To undo and redo edit changes:

1. Click **Edit > Undo** from the workbench menu.

   Notice that the 5 new lines disappear.

2. Press **Ctrl+Z** to undo the last change.

   Notice that the deleted lines reappear.

3. Click **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. Click **File > Close** on the workbench menu.

   **Tip:** You can also click the X on the **PAYROLL** tab.
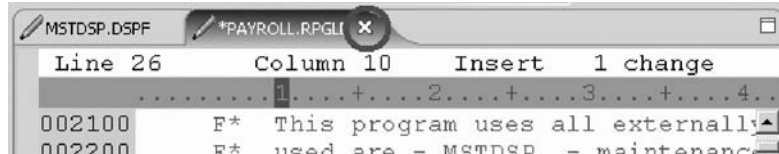
A Save Resource dialog opens asking if you want to save the latest changes.
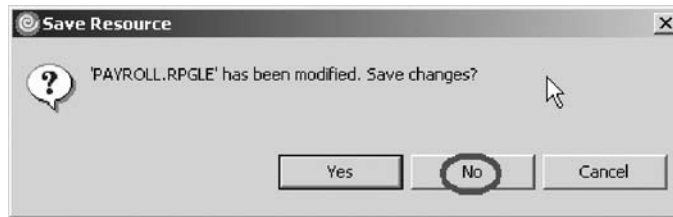


5. Click **No**.
6. Go back to the workbench to the Remote Systems view and open the PAYROLL member in the QRPGLESRC file.

You have learned how to undo and redo changes that you made to a file and now you are ready to begin "Exercise 3.5: Invoking language-sensitive help."

## Exercise 3.5: Invoking language-sensitive help

Before you begin, you must complete "Exercise 3.4: Requesting undo and redo operations" on page 34.

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 allowed values for a variable field. This help is available from the LPEX Editor window.

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

**Accessing language sensitive help**

To access language sensitive help:

1. Position the cursor over the word MOVE in line 112 of the ILE RPG 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.
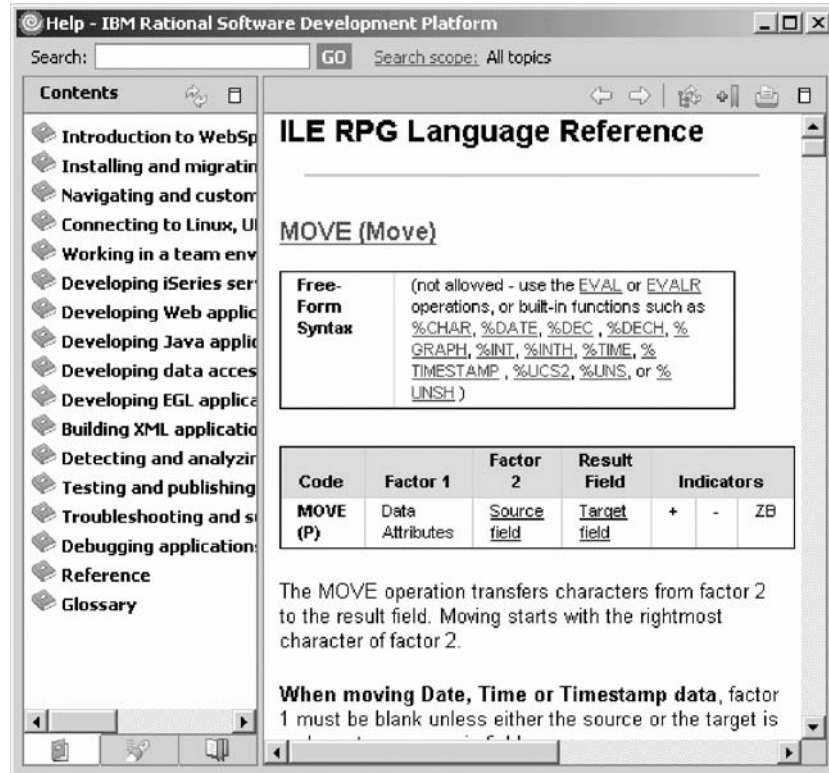
3. Click the link Date Data Type.

   The Help page for Date Data Type is displayed.
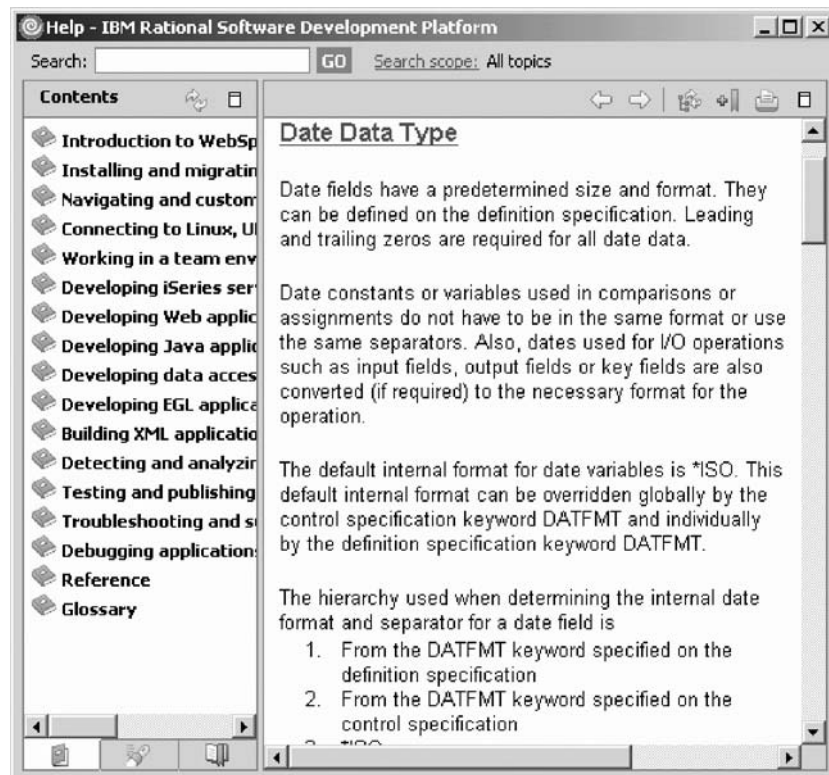


4. Play around in the Help window to see what else is available.
5. Minimize the Help window.

**Note:** If you are using WDSC Lite, the following steps do not apply.

6. Select **Help > Help Contents** on the workbench menu to see a list of all help that is available in the product.
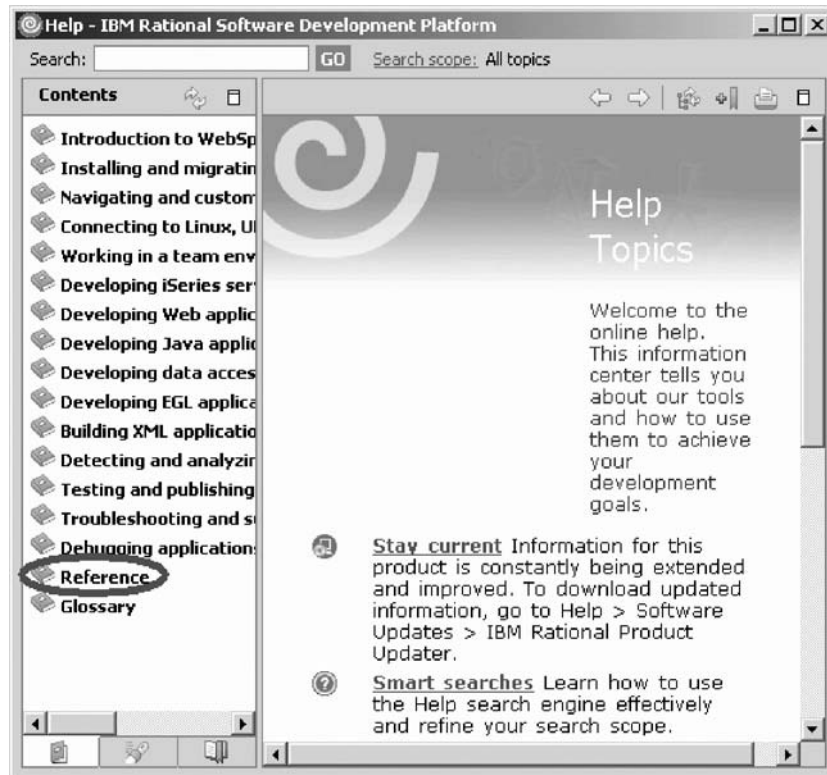


7. In the left pane of the Help window, click Reference.
8. Expand Language Reference.
9. Expand iSeries programming information.
10. Expand High-level languages.
11. Expand ILE RPG.
12. Select Language Reference.
13. Scroll down to Moving Date-Time Data.
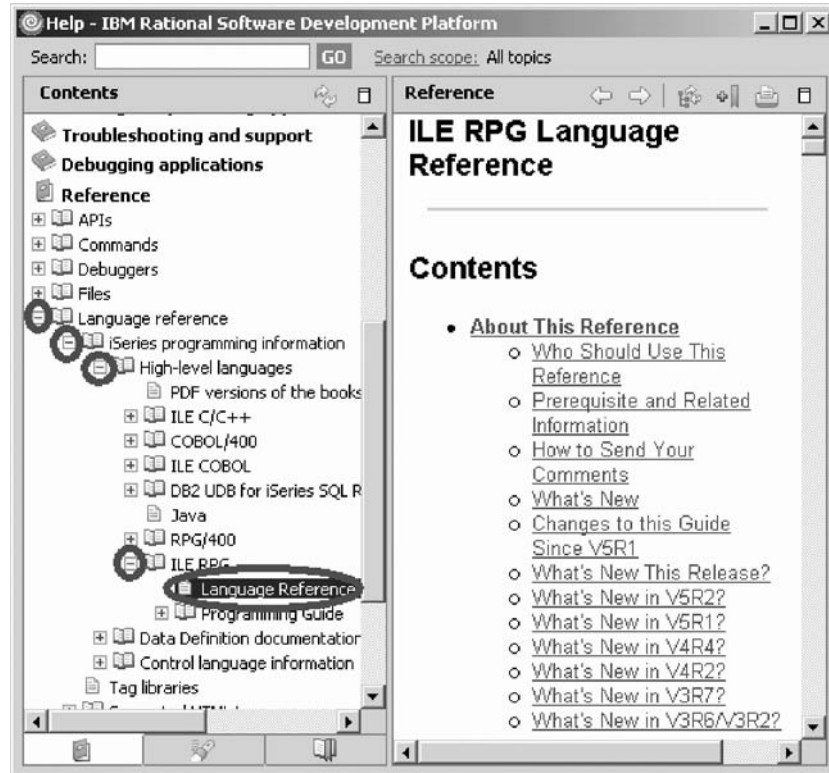
Having the latest version of the manuals at your fingertips will make it easier to find programming information. There is also the option to search the help by specifying a search string. By default, the complete help will be searched.

**Limiting the search scope**

**Note:** If you are using WDSC Lite, you can limit the search scope, however, you must choose another topic instead of the Reference topic shown.

To limit the search to specific documents:
1.  Click **Search scope**.
    The Select Search Scope dialog opens.
2.  Select **Search only the following topics** radio button.
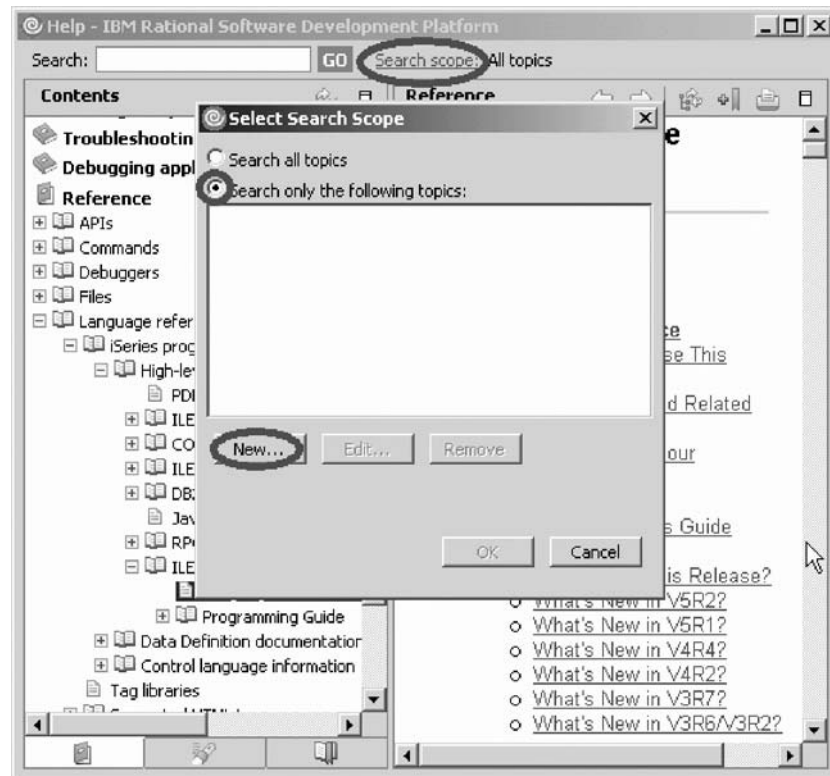
3. Click **New**.

The New Search List dialog opens.
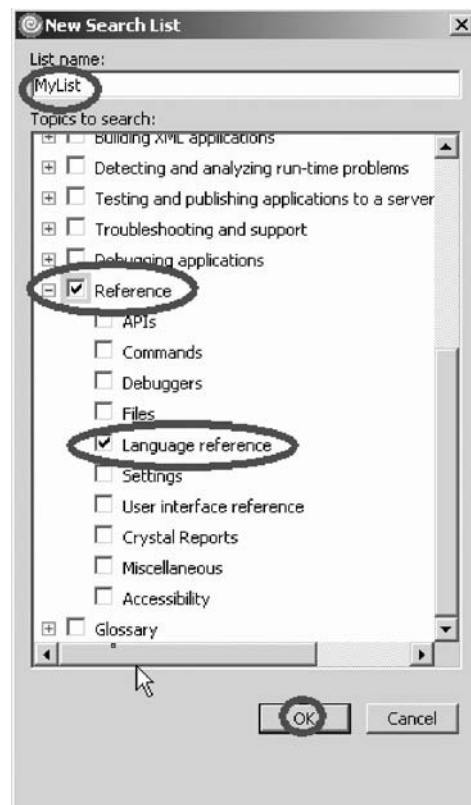


4. In the **List name** field, type `MyList` for example.

5. Expand **Reference**.
6. Select the **Language Reference** check box.
7. Click **OK** on the New Search List dialog.

   The Select Search Scope dialog reopens again with MyList selected in the topic list.



8. Click **OK** on the Select Search Scope dialog.
9. In the **Search** field, type `iSeries and programming` for example.
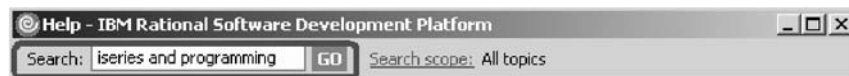


10. Searching requires a help index and it takes approximately 10 minutes to create the index in your workspace. If you want to complete the search query, click **GO**.

    The search results display.
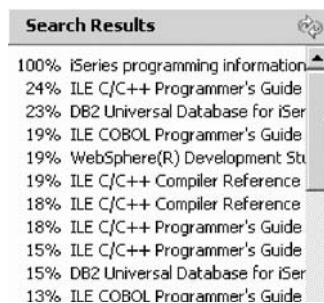


You have accessed language sensitive help and now are ready to begin "Exercise 3.6: Prompting language specifications."

## Exercise 3.6: Prompting language specifications

Before you begin, you must complete "Exercise 3.5: Invoking language-sensitive help" on page 35.

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.

**Accessing language prompts**

To access prompts:

1. Return to the workbench. In the Editor window move your cursor to the D-spec on line 33.

2. Press **F4** (You can also click **Source** from the workbench menu and then click **Prompt**.)

   You 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.
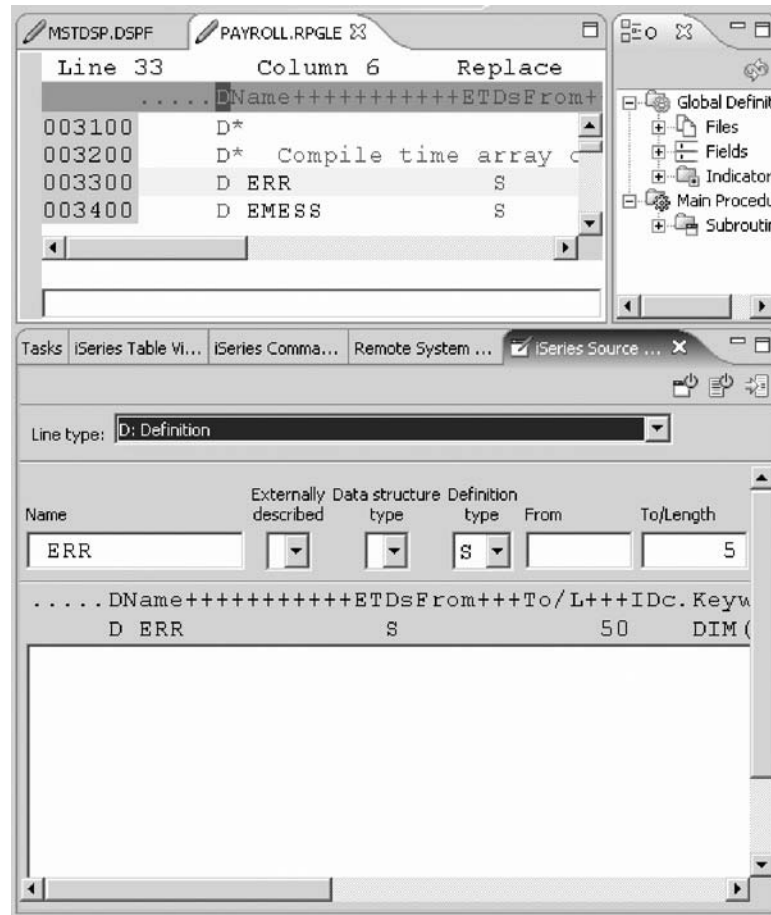


On the iSeries Source Prompter toolbar you can use the three push buttons to: Disable source prompt view, Disable syntax checking, and Change to insert mode.

**Displaying context sensitive help**

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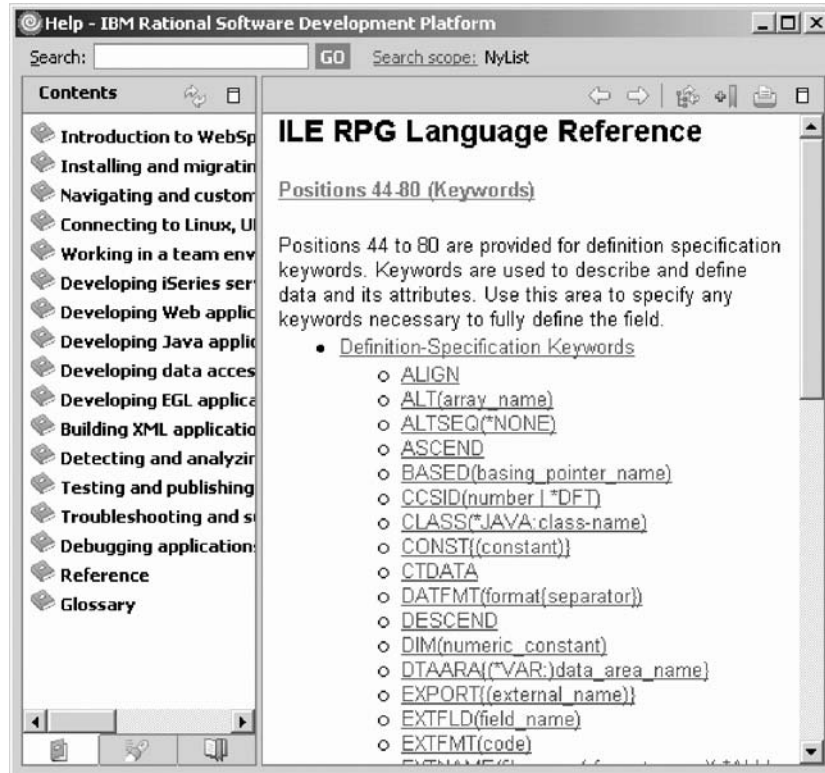
You will see words in the help that appear in a different color than the regular text. These are help links, and they show that there is additional help available on that word or phrase.

3. Click on any link to see specific help for that item.

4. Minimize the Help window.

5. In the workbench, double-click the **PAYROLL** tab to return the Editor window to its original size.

You have used prompts to enter code and now are ready to begin "Exercise 3.7: Indenting source."

## Exercise 3.7: Indenting source

Before you begin, you must complete "Exercise 3.6: Prompting language specifications" on page 40.

When editing ILE RPG source, it can be difficult to determine the beginning and ending of constructs. The iSeries Indent view allows you to see your source with constructs in an indented mode.

To indent source:

1. Click **Source > Show Indentation** from the workbench menu.

   The iSeries Indent view opens at the bottom of the workbench.
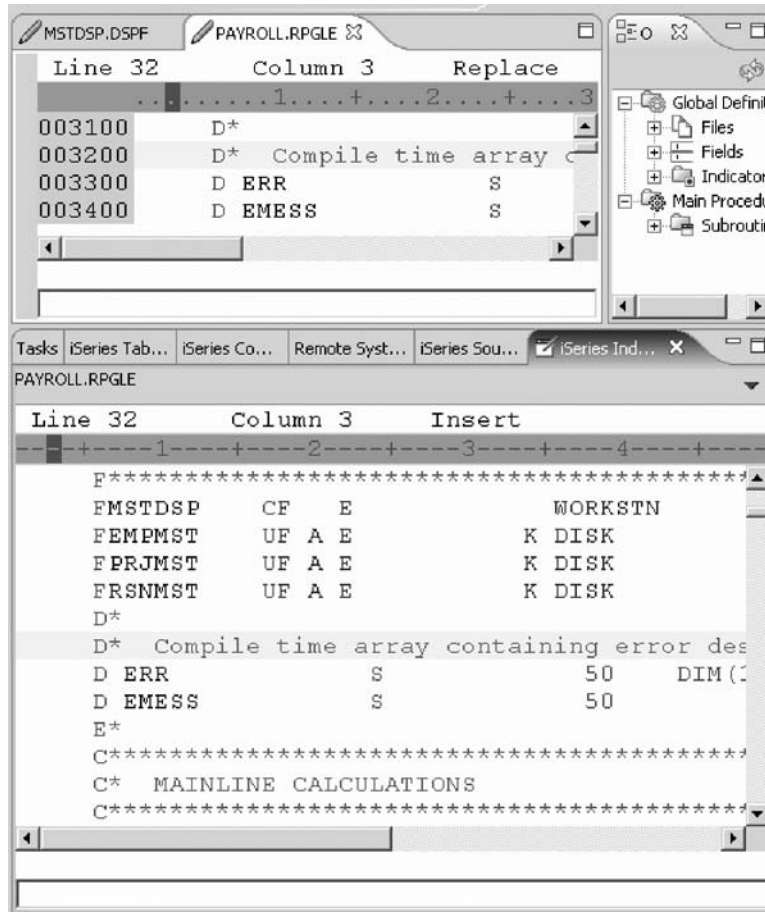
You can display the iSeries Indent view as a full view.

2. Double-click the **iSeries Indent** tab.

```
Tasks   iSeries Table...   iSeries Com...   Remote Syste...   iSeries Sourc...   ☑ iSeries Indent ✕   ▬ ₽
PAYROLL.RPGLE                                                                                          ▼

 Line 150        Column 11     Insert                                                    Br
----+----1----+----2----+----3----+----4----+----5--
        C*                                                                              ▲
        C                        | | | IF          not *INKC
        C      RSCDE             | | | | CHAIN        RSNMST
        C                        | | | | EXSR         ACDESR
        C                        | | | ELSE
        C                        | | | | LEAVESR
        C                        | | | END
        C* end of error loop
        C                        | | ENDDO
        C*
        C*   Display reason code maintenance format
        C*
        C                        | | EXFMT       RSNMNT
        C*
        C                        | | IF          *INKD
        C                        | | | LEAVESR
        C                        | | END
        C                        | | IF          *INKG
        C                        | | | ITER
        C                        | | END
        C*
        C*   Determine update mode and perform record a
        C*
        C        |               | | IF          not *inkc      ▼
◄ ►                                                          ► ►
```

3. Scroll down to line 150. The Row shown on the status bar is the cursor position.

   In the iSeries Indent view you see some nested conditions with indented lines. As you will notice this helps to recognize the beginning and ending of these conditions.

   **Note:** The Indented view is Browse mode only and cannot be edited.

4. Click the X in the top right corner of the view to close the iSeries Indent view.

5. This returns you to the Editor window with the PAYROLL program.

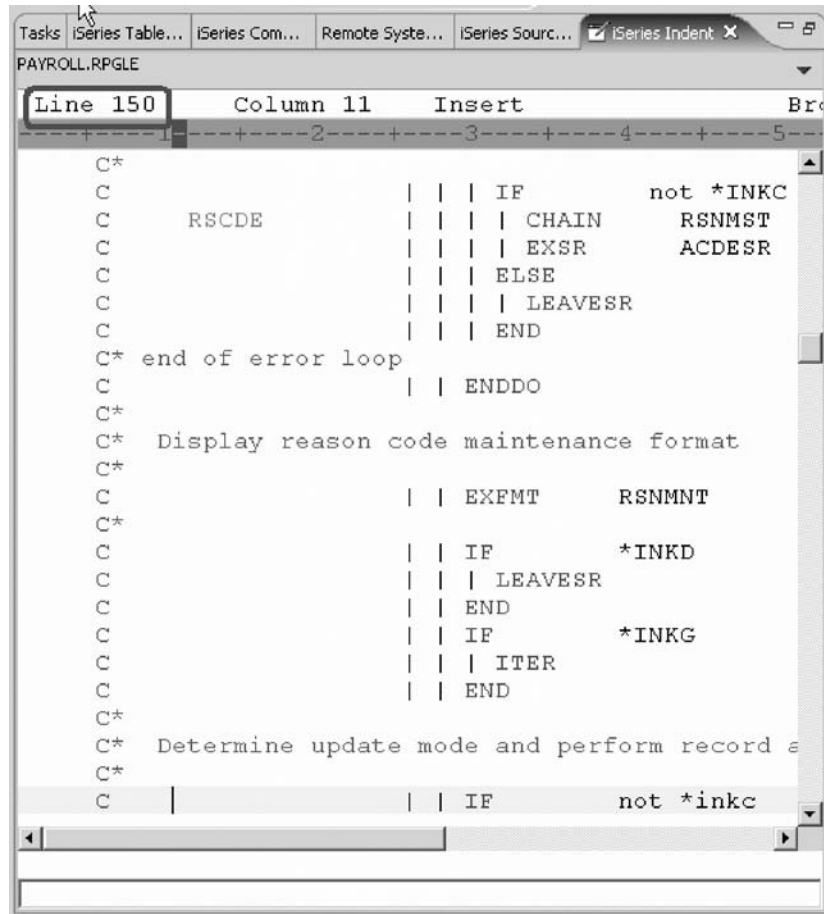You have used the Indent view to see the beginning and ending of constructs and you are ready to begin "Exercise 3.8: Finding and replacing text."

# Exercise 3.8: Finding and replacing text

Before you begin, you must complete "Exercise 3.7: Indenting source" on page 42.

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:

1. Press **Ctrl+Home** to go to the top of the file.

   **Tip:** When you press **Ctrl+Home** to go to the top of a file or **Ctrl+End** to go to the bottom of a file, a quick mark is set at your cursor position. This allows you to return to that line by pressing **Alt+Q**. **Ctrl+Q** will set a quick mark.

2. Click **Edit > Find/Replace** from the workbench menu or press **Ctrl+F**.

   The Find/Replace window opens at the bottom of the Editor window.
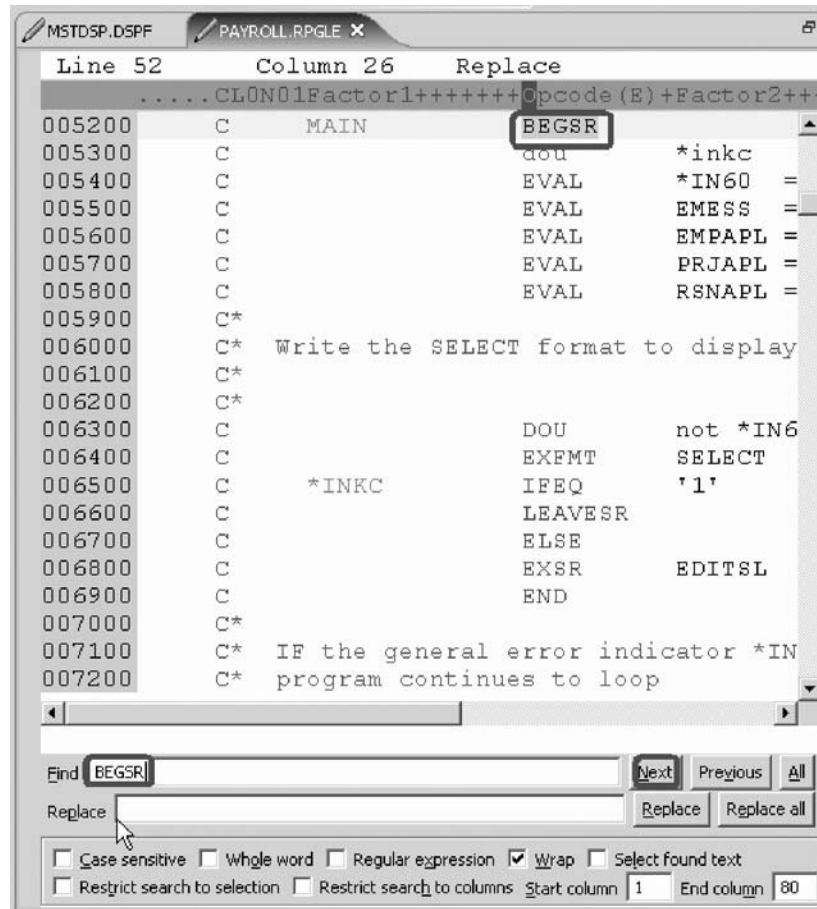


At the bottom of this window, you will notice that you have some options to select from, for example, search only in certain columns, etc. You want to find the first occurrence of BEGSR.

3. In the **Find** field, enter BEGSR to find the start of a subroutine.

4. Make sure the **Replace** field is blank.

   You would use this field for text replacement.

   The Editor moves the active line to line 52, which contains the first BEGSR phrase in the file.

5. Click **Next** to go to the next location of BEGSR in the file.

6. Click in the Editor window to close the Find/Replace window.

You have searched for an item in your source using the Find/Replace window and now you are ready to begin "Exercise 3.9: Filtering lines by string" on page 46.

# Exercise 3.9: Filtering lines by string

Before you begin, you must complete "Exercise 3.8: Finding and replacing text" on page 44.

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.

To filter source by string:
1. Double-click the variable EMPAPL in the Editor window.
2. Select **Edit > Selected > Filter Selection** from the workbench menu.
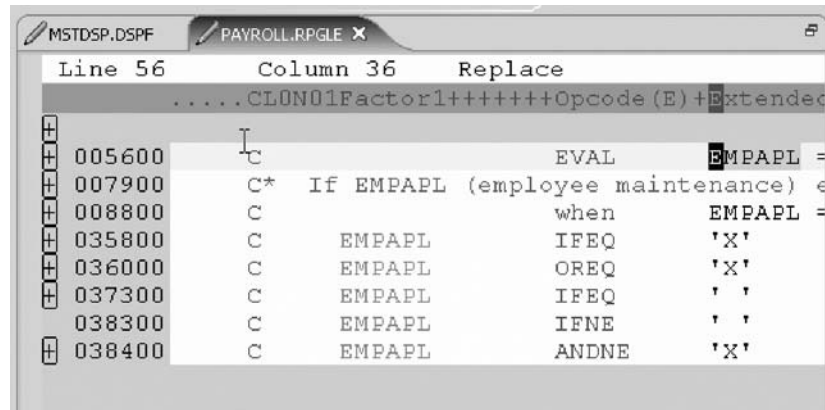


3. Move the cursor down a few lines to line 79.
4. Expand line 79. This expands the section up to the next instance of EMPAPL.
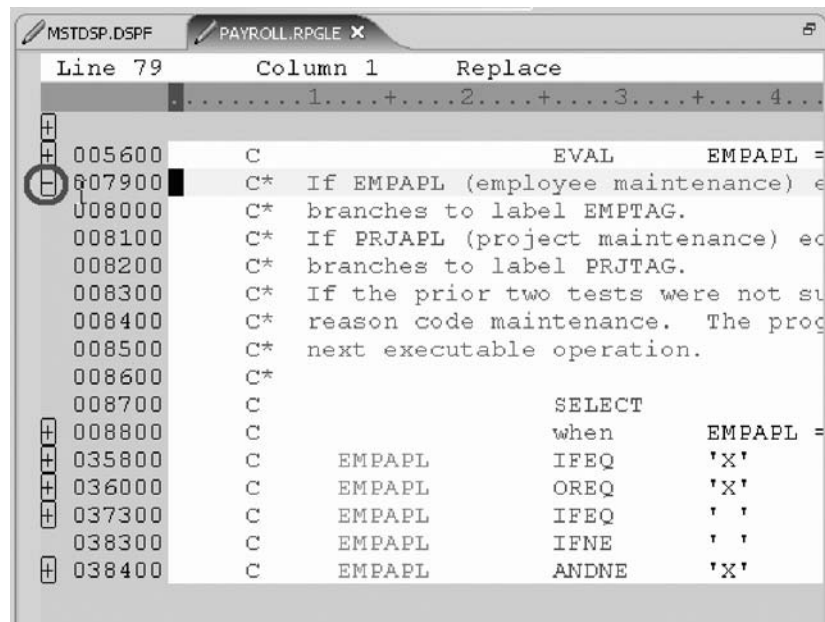


Now you want to show the entire source again.
5. Click **Edit > Show all** from the workbench menu or press **Ctrl+W**.

   Your cursor is still positioned on the same line that you moved the cursor to, even though all lines are now showing.

You have filtered your source so that you see only lines containing a given string and now you are ready to begin "Exercise 3.10: Filtering lines by type."

## Exercise 3.10: Filtering lines by type

Before you begin, you must complete "Exercise 3.9: Filtering lines by string" on page 46.

To help you navigate quickly through your ILE RPG 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 source.
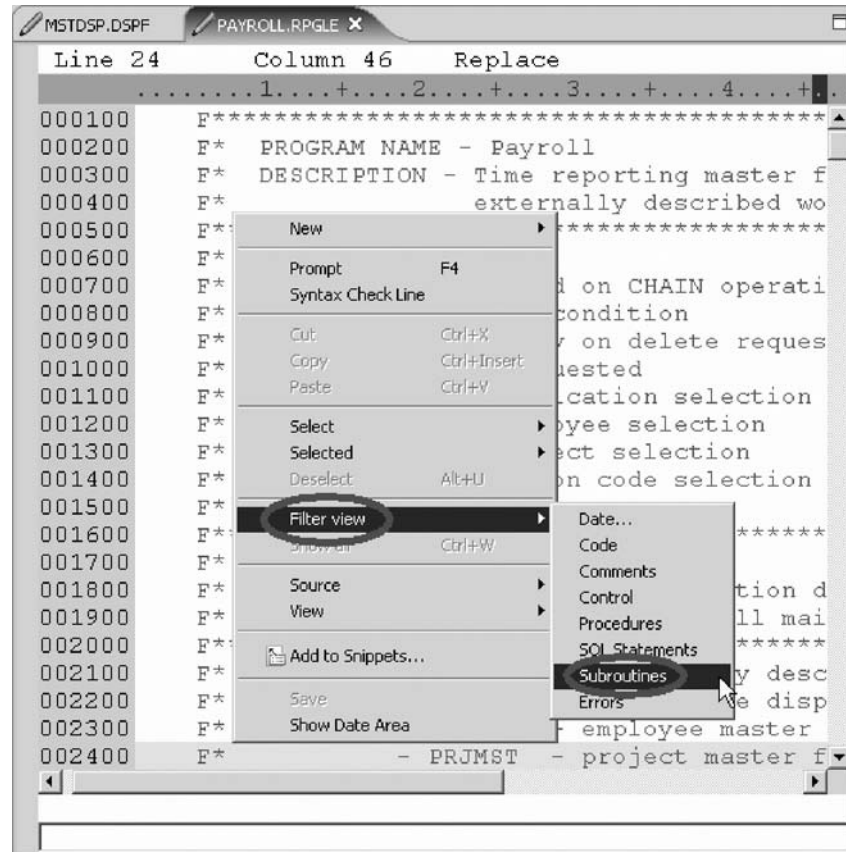


To filter lines by type:
1. Right-click in the Editor window with the PAYROLL program.
2. Click **Filter view > Subroutines** on the pop-up menu.

   All source lines with BEGSR or ENDSR are displayed allowing you to move quickly and easily to the desired subroutine in your file.

```
MSTDSP.DSPF        PAYROLL.RPGLE  ×                                        ⊟
   Line 102       Column 1       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
   ┼ 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;
```

3. Move your cursor to the line with the subroutine declaration CHGCDE (line 444).

4. Expand the declaration to show all lines in this subroutine.

   Now you could work with the source inside this subroutine.

5. Right-click in the Editor window and click **Show all** on the pop-up menu.

You have filter lines in your source by line type and now you are ready to begin "Exercise 3.11: Searching multiple files."

## Exercise 3.11: Searching multiple files

Before you begin, you must complete "Exercise 3.10: Filtering lines by type" on page 47.

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.
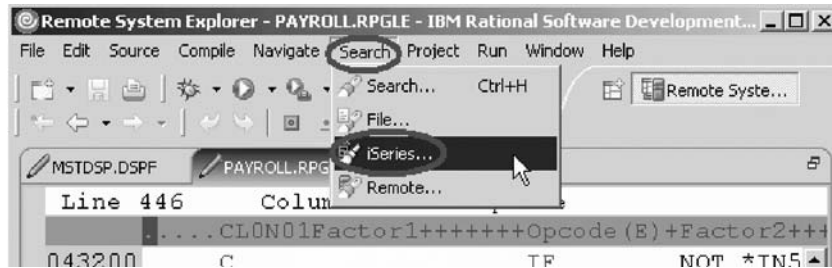
The Search window opens.

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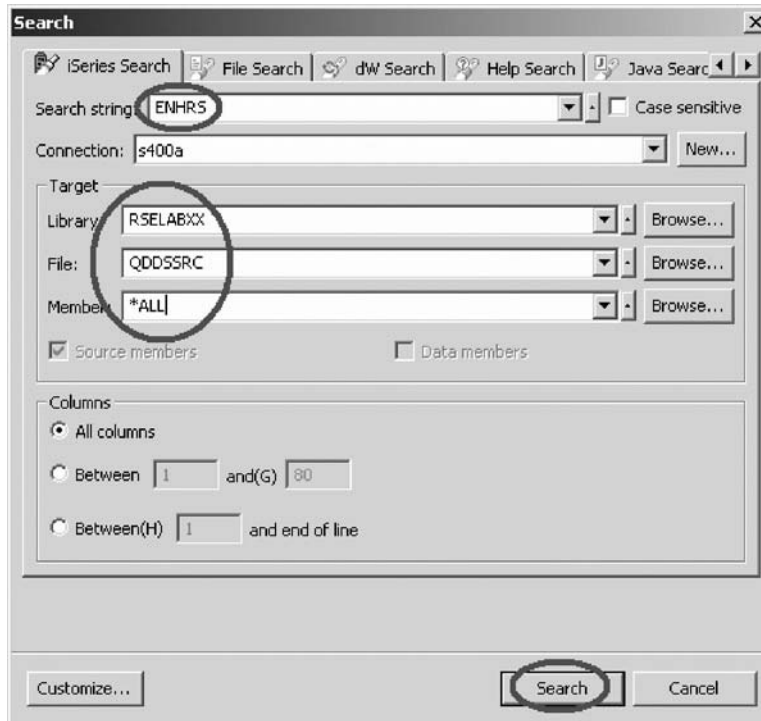


The **Connection** field should contain your iSeries server name, otherwise enter it there.

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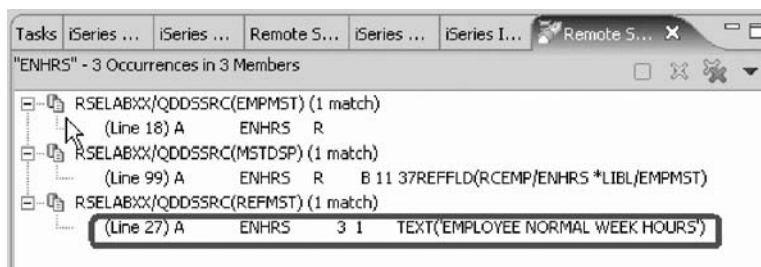
7. Double-click the last line in the list.

```
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.

```
 MSTDSP.DSPF     PAYROLL.RPGLE     REFMST.PF X                          ⊟

   Line 27          Column 1      Replace
                 .....A...........Name+++++RLen++TDpB......F
   000026      A                                              ▲
  ┌000027      A              ENHRS         3   1            ┐
   000028      A
   000029      A              EPHRC         5   1
   000030      A
   000031      A              EPHRY         7   1
   000032      A
   000033      A              EPHRP         7   1
   000034      A
   000035      A              EPNRC         5   1
   000036      A
   000037      A              EPNRY         7   1
   000038      A
   000039      A              EPNRP         7   1
   000040      A
   000041      A              EHWRK         5   1
   000042      A
   000043      A*   PROJECT MASTER RELATED FIELDS
   000044      A              PRCDE         8
   000045      A
   000046      A              PRDSC         50
   000047      A
   000048      A              PRRSP         30
   000049      A
   000050      A            I PRSTR      R                    ▼
 ◄                                                          ►
```
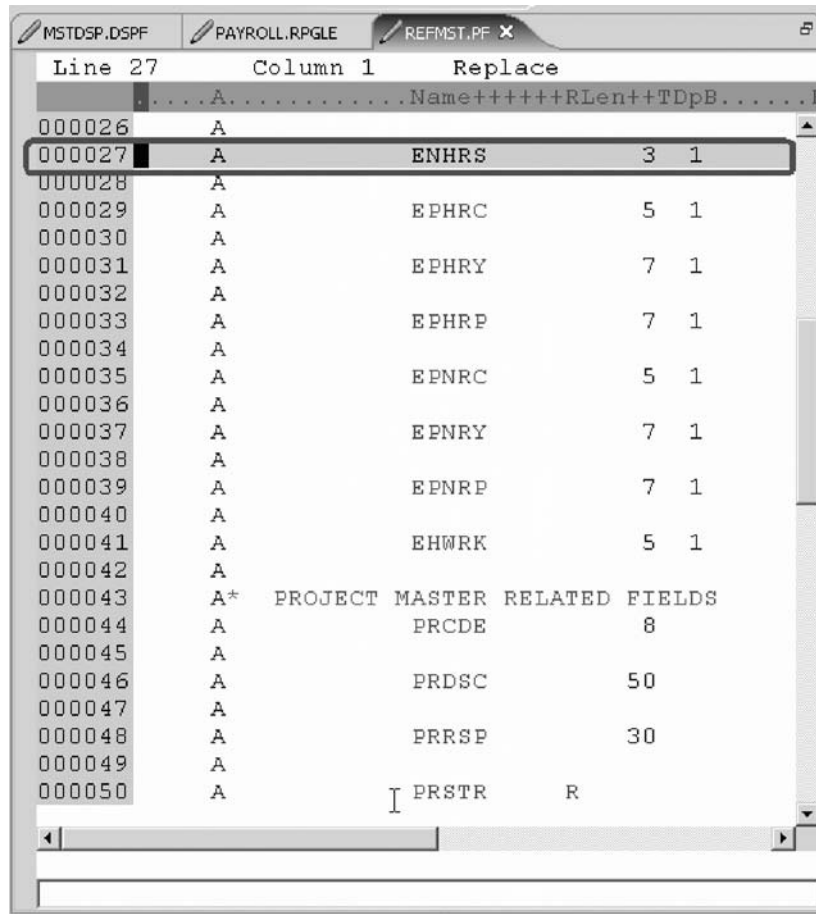
8. Click the X in the **REFMST** tab to close the REFMST file.

You have searched through members in a source physical file and now you are ready to begin "Exercise 3.12: Comparing file differences from the Remote Systems view."

## Exercise 3.12: Comparing file differences from the Remote Systems view

Before you begin, you must complete "Exercise 3.11: Searching multiple files" on page 48.

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.

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.
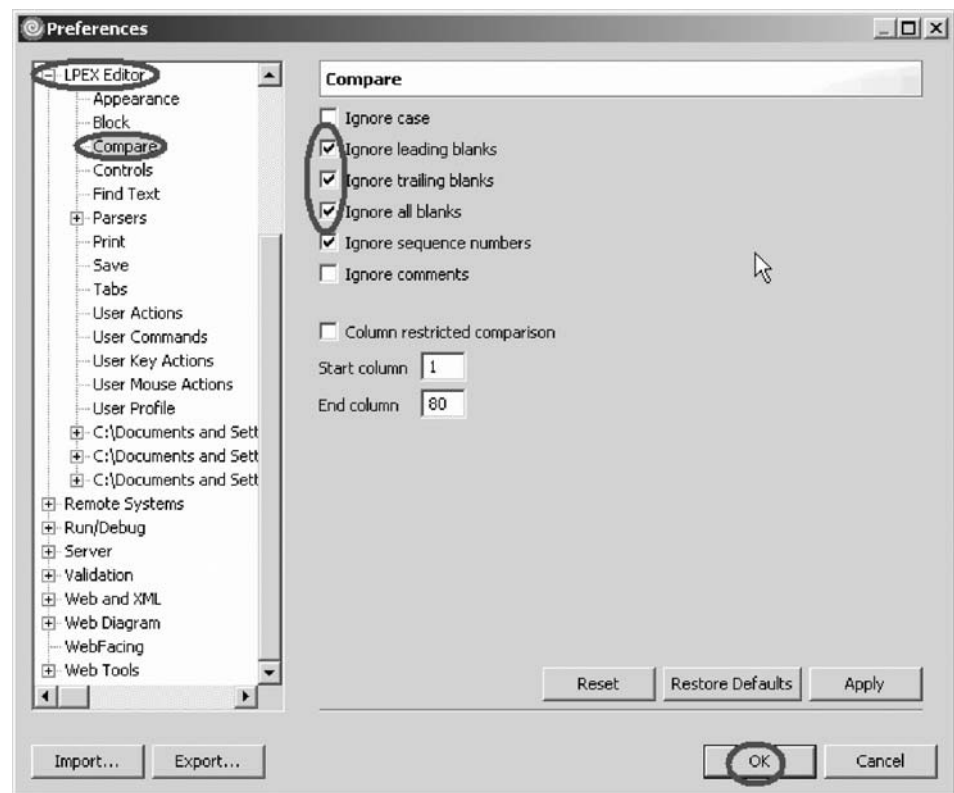
**Note:** Make sure all lines show in the source before starting the Compare tool.

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 Editor window of the PAYROLL member double-click the **PAYROLL** tab.
6. Click **Edit > Compare > Compare to file** on the workbench menu.
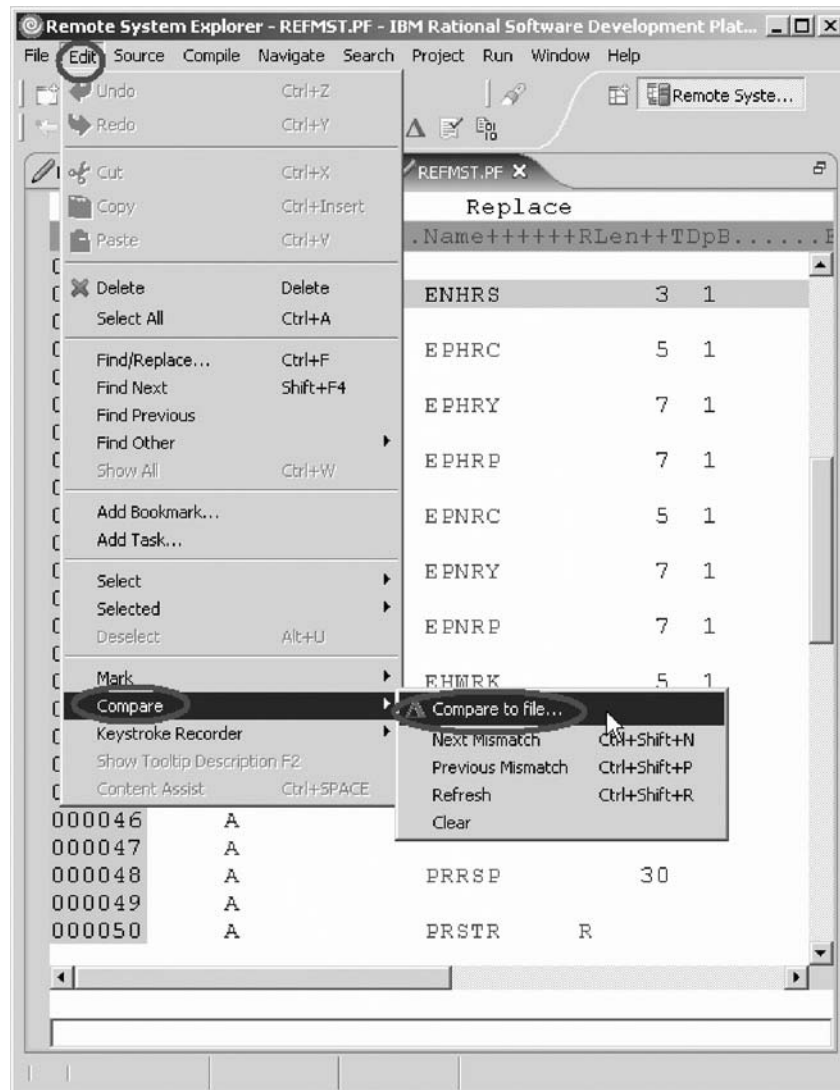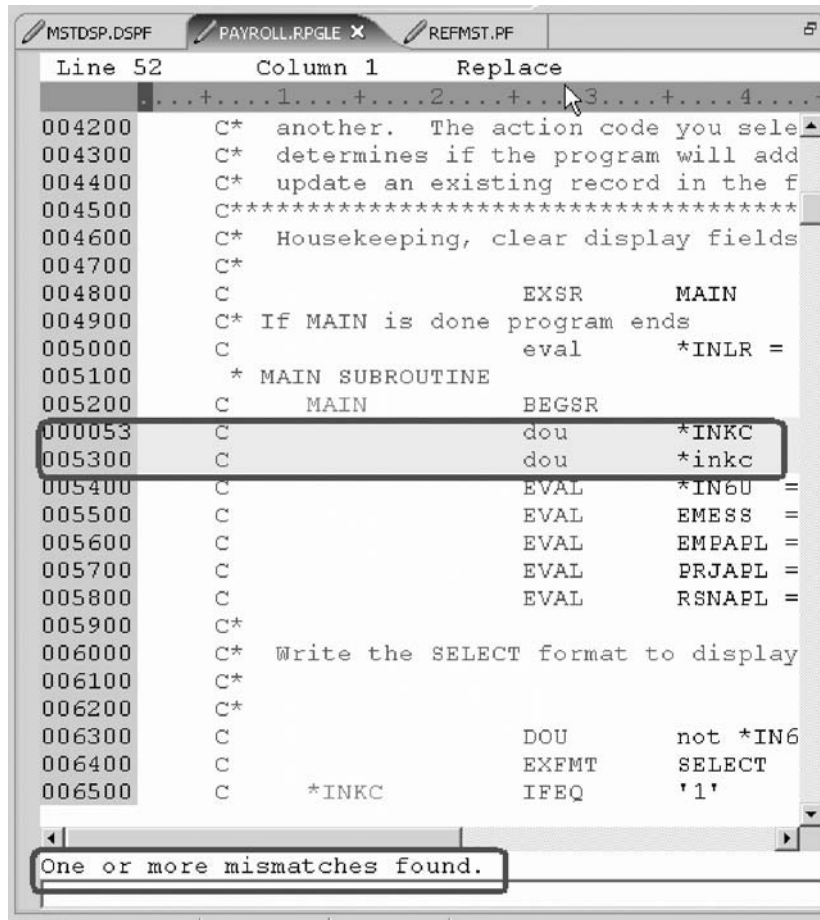


The Compare window opens.
7. Expand your connection.
8. Expand *LIBL.
9. Expand RSELABxx.
10. Expand QRPGLESRC.
11. Select member PAYROLLG.
12. Click **OK**.
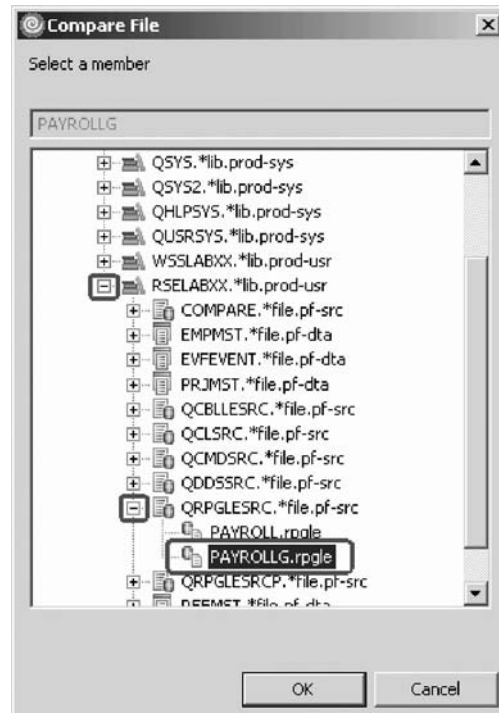
The editor now will show the differences of these two members PAYROLL and PAYROLLG .

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

Mismatches in PAYROLL and PAYROLLG are highlighted in different colors so that you know where the mismatched lines are in each file.

```
MSTDSP.DSPF      PAYROLL.RPGLE ✕      REFMST.PF                        ⊡
  Line 52         Column 1      Replace
       .....+....1....+....2....+....3....+....4....+
004200       C*   another.   The action code you sele ▲
004300       C*   determines if the program will add
004400       C*   update an existing record in the f
004500       C*********************************************
004600       C*   Housekeeping, clear display fields
004700       C*
004800       C                   EXSR      MAIN
004900       C* If MAIN is done program ends
005000       C                   eval      *INLR =
005100        * MAIN SUBROUTINE
005200       C      MAIN         BEGSR
000053       C                   dou       *INKC
005300       C                   dou       *inkc
005400       C                   EVAL      *IN6U   =
005500       C                   EVAL      EMESS   =
005600       C                   EVAL      EMPAPL  =
005700       C                   EVAL      PRJAPL  =
005800       C                   EVAL      RSNAPL  =
005900       C*
006000       C*   Write the SELECT format to display
006100       C*
006200       C*
006300       C                   DOU       not *IN6
006400       C                   EXFMT     SELECT
006500       C      *INKC        IFEQ      '1'              ▼
 ◄                                          ►
One or more mismatches found.
```

13. Click **Ctrl + Shift + N** to find the next mismatch.

    Next, end the compare session.

14. Click **Edit > Compare > Clear** from the workbench menu.

15. Double-click the **PAYROLL** tab to return the Editor window to its original size.

You have compared different versions of the program and found the differences and now you are ready to begin "Exercise 3.13: Comparing files in the CODE Editor (optional)."

## Exercise 3.13: Comparing files in the CODE Editor (optional)

Before you begin, you must complete "Exercise 3.12: Comparing file differences from the Remote Systems view" on page 50.

The CODE tool provides a side-by-side view of the members being compared. If you prefer this type of view follow the steps described next. You might want to skip this section if the Compare Tool you just used provides sufficient information for you. Now you will open a couple of files and edit them and use the CODE compare utility.

To compare files in the CODE Editor:
1. In the Remote Systems view, right-click member PAYROLL in QRPGLESRC.
2. Click **Open With > CODE Editor** on the pop-up menu.

This opens the CODE Editor window with the member. It will open in browse mode since it is locked by the LPEX Editor session.
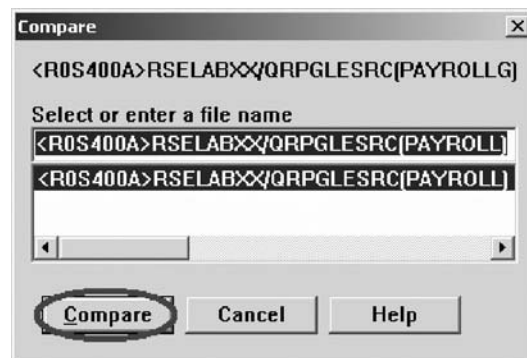
3. In the CODE Editor, open the PAYROLLG member.

4. Click **File > Open** from the CODE Editor menu.

   The Select file - Open for Edit dialog opens.
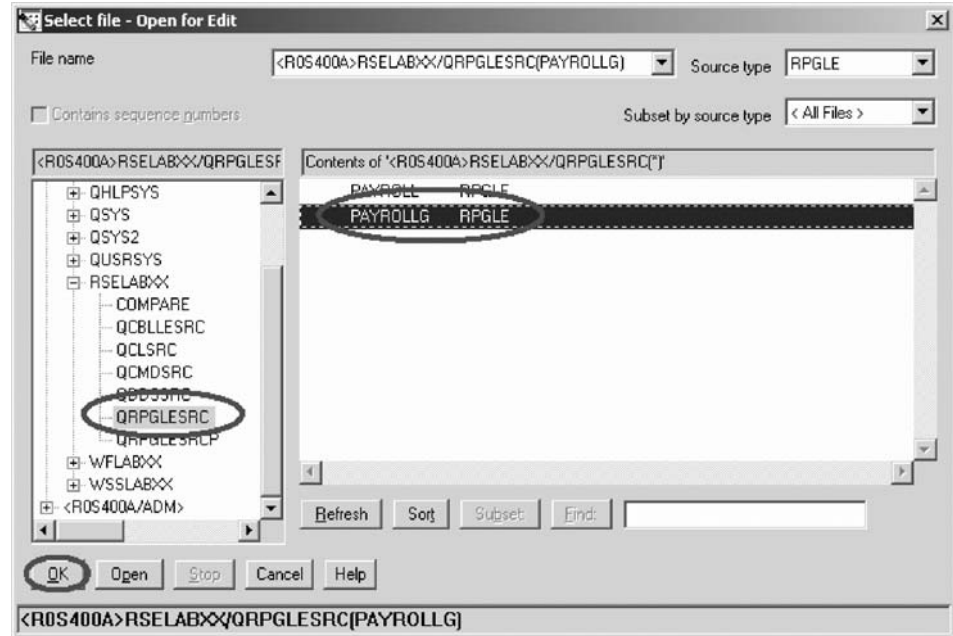


5. In the left pane of the Select file - Open for Edit dialog expand the **R0S400A** connection (the Remote System Explorer connection).

6. Expand library RSELABxx.

7. Select file QRPGLESRC.

8. In the right pane of the Select file-Open for Edit dialog, select member PAYROLLG from the member list.

9. Click **OK**.

10. Click **Actions > Compare** from the CODE Editor menu.

    The Compare dialog opens:



    All entries are preloaded.

11. Click **Compare**.

    The editor now has the PAYROLLG member loaded on the left and member PAYROLL loaded on the right. In between the two members are two long vertical blue lines with horizontal yellow and red bars highlighting the

differences in these members.
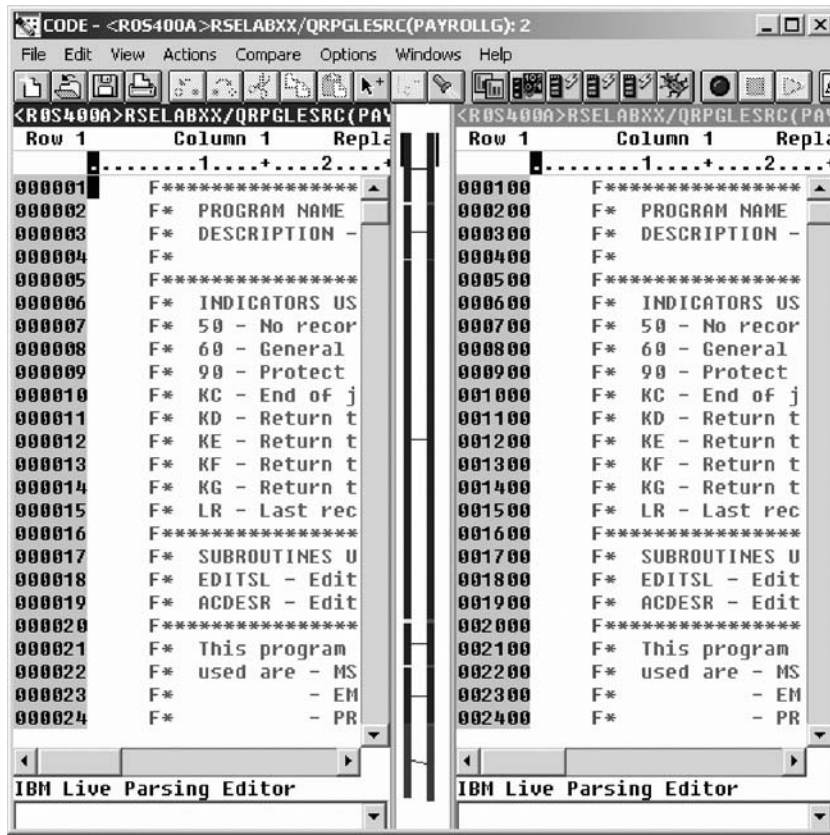


12. Use the vertical scroll bars to move within the files.

As you scroll, you will see where the differences are in the members.

RPG experts will notice that PAYROLL has some errors in it. You will fix these in a few moments.

13. Click **Ctrl + Shift + N** to find the next mismatch.

14. Click **Compare > Exit compare** from the CODE Editor menu (which was inserted while performing this action).

The CODE Editor window re-opens.

Close the CODE Editor.

15. Click **File > Exit** from the CODE Editor menu. The workbench re-opens. Continue working in the workbench.

You have compared files using the CODE Editor and now you are ready to begin "Exercise 3.14: Checking syntax."

## Exercise 3.14: Checking syntax

Before you begin, you must complete "Exercise 3.13: Comparing files in the CODE Editor (optional)" on page 54.

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 source member.

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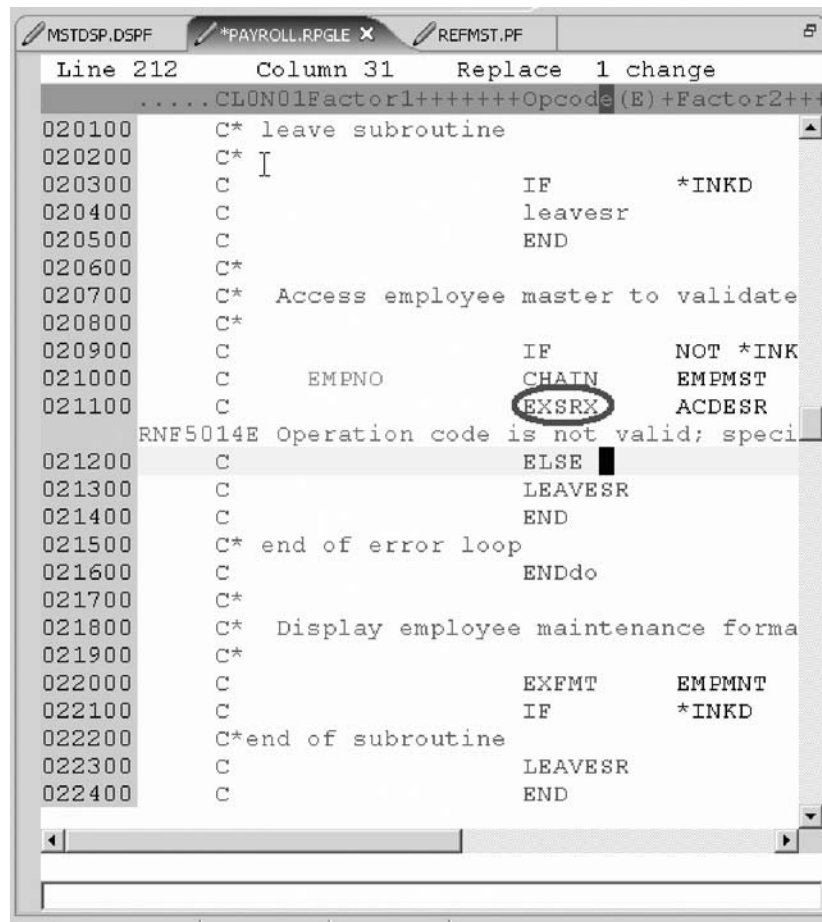
To syntax check the file:

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.

```
MSTDSP.DSPF      *PAYROLL.RPGLE ×      REFMST.PF

   Line 212        Column 31      Replace   1 change
          .....CLON01Factor1++++++Opcode(E)+Factor2+++
   020100        C* leave subroutine
   020200        C*
   020300        C                          IF        *INKD
   020400        C                          leavesr
   020500        C                          END
   020600        C*
   020700        C*   Access employee master to validate
   020800        C*
   020900        C                          IF        NOT *INK
   021000        C       EMPNO              CHAIN     EMPMST
   021100        C                          EXSRX     ACDESR
          RNF5014E Operation code is not valid; speci
   021200        C                          ELSE
   021300        C                          LEAVESR
   021400        C                          END
   021500        C* end of error loop
   021600        C                          ENDdo
   021700        C*
   021800        C*   Display employee maintenance forma
   021900        C*
   022000        C                          EXFMT     EMPMNT
   022100        C                          IF        *INKD
   022200        C*end of subroutine
   022300        C                          LEAVESR
   022400        C                          END
```

4. Move the cursor onto the pink error message.

5. Press **F1**.
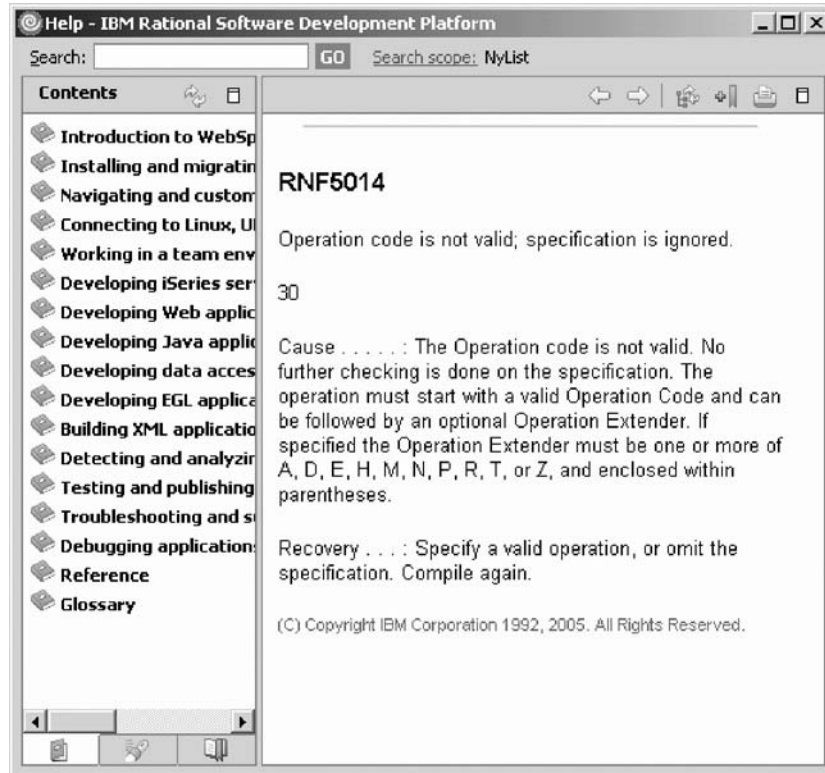
This opens a window with second level help for the error.

6. Minimize the Help window.

7. Change EXSRX to EXSR to correct the error. (Similarly, you can use **Edit > Undo** to correct this).

8. Move the cursor off the line you just fixed.

   The error message is automatically removed from the editor.

   **Tip:** You can toggle automatic syntax checking. Click **Window > Preferences** from the workbench menu and then expand **Remote Systems, iSeries, LPEX Editor Parsers** and select the language you want to change the settings for in the left pane of the Preferences window, select the **Automatic syntax checking** check box and then click **OK**.

   **Tip:** You can syntax check the whole source member currently in the editor by clicking **Source > Syntax Check All**.
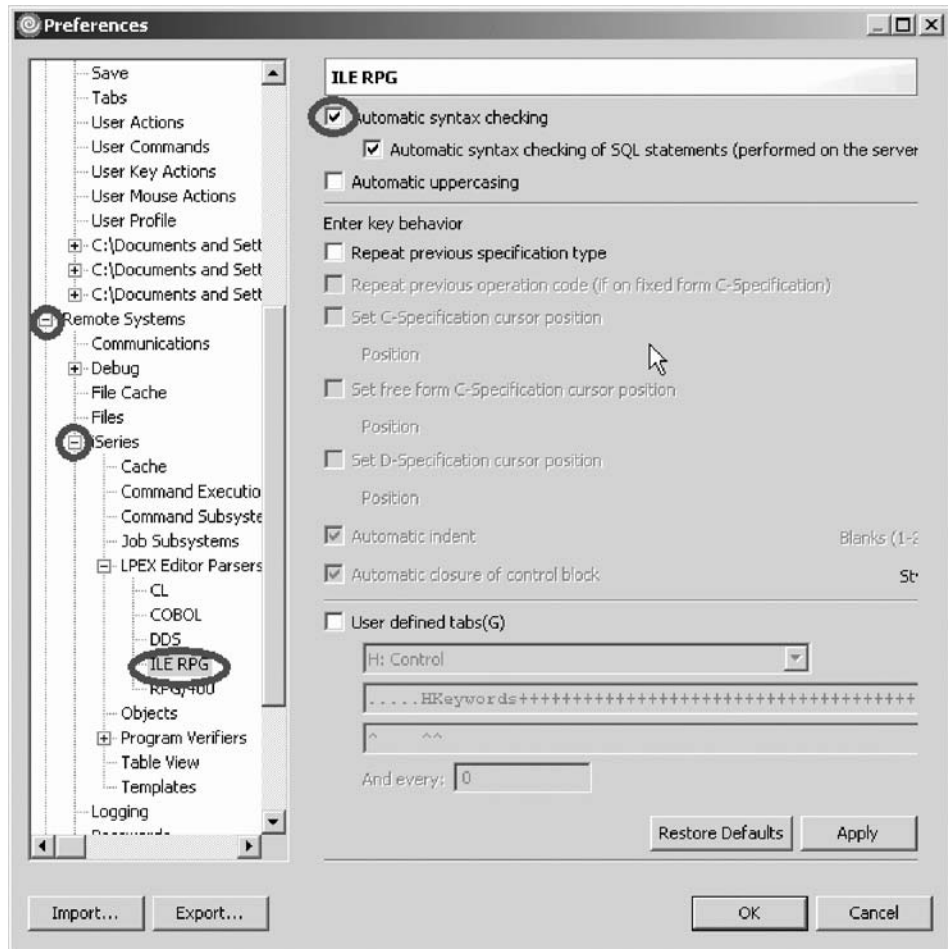
Now you are ready to review your knowledge of this module by taking the quiz. You can also apply what you have learned in this module by completing the practice tasks detailed in More practice.

**Quiz**

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 preferences 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 allowed 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 the ____ key in an Editor window. Name the key:

   a. F2

   b. F3

   c. F1

   d. F4

9. If the cursor is _____ an operation code, you receive help for that operation code; otherwise, you receive help for the current specification.

   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. The _____ view allows you to see your source with constructs in an indented mode. By default, the _____ view will show the indented view in the bottom pane.

   a. Outdent

   b. iSeries Indent

   c. Edit

12. The iSeries Indent view is browse mode only and cannot be edited. (T, F)

13. You use the _____ window to search for an item in the current source. Choose the best answer.

   a. Search

   b. Find

   c. Edit

   d. Find/Replace

14. 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

15. 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

16. 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. both

17. 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

18. 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

19. There are two ways to compare files. They are Compare tool in the workbench and the Compare tool in the CODE Editor. (T, F)

20. 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 source member. (T, F)

**More practice**

Given your experience 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 product online help to assist you in these tasks.

**Module recap**

You have completed Chapter 3, "Module 3. Editing source," on page 25. You have learned how to:

- Change the default settings of the LPEX Editor Parsers
- Change the color settings and font used by the Editor
- Change the default behavior of the Enter key
- Use SEU commands to edit source
- Undo and redo source changes
- View language sensitive help for the MOVE operation code
- View a list of all help contents
- Limit the search of help to specific documents
- Search the help
- Request a prompt for a specification line
- Display context sensitive help for any field in the iSeries Source Prompter
- View the beginning and ending of constructs in your source
- Use the Find and Replace window to search for an item in your source
- Filter or subset your source
- Filter lines based on line type
- Search through members in a source physical file
- Compare different versions of a program and identify the differences
- Syntax check source by line
- View help on syntax errors

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! And you perform both verify and compile from the Remote Systems view! Continue to Chapter 4, "Module 4. Verifying and compiling source," on page 63.

# Chapter 4. Module 4. Verifying and compiling source

This module teaches you how to verify and compile RPG in the Remote Systems LPEX Editor. When errors are found by either the verify or the compile step, the iSeries Error List appears. The iSeries Error List is a powerful tool that manages errors found by verify and compile utilities. You will become familiar with these tools, the various capabilities of the iSeries Error List and the RPG program that you have created.

In this module, you will:
- Check for semantic errors on your workstation
- Start the Program Verifier tool
- Use the iSeries Error list to locate each error in the source
- Use content-assist to fix an error
- Save your source
- Re-verify source
- Change compile preferences
- Invoke the compile command
- Change the current library using the Command field in the iSeries Table view
- Start an interactive connection
- Invoke the payroll program

**Exercises**

The exercises within this module must be completed in order:
- "Exercise 4.1: Verifying the source"
- "Exercise 4.2: Compiling source remotely" on page 67
- "Exercise 4.3: Submitting iSeries commands in the iSeries Table view" on page 71
- "Exercise 4.4: Running commands and programs" on page 73

**Time required**

This module will take approximately **20 minutes** to complete.

## Exercise 4.1: Verifying the source

Now you get to play with one of the most powerful and unique features of the Remote System Explorer – 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 Remote System Explorer ported the parsing and checking code from the iSeries system compilers to the workstation. The iSeries Error List view 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.

To invoke the verifier:

1. Click **Source > Verify** from the workbench menu. (Similarly, you can also use the pop-up menu for the source member in the Remote Systems View or the Verify tool button — you need the source in the editor for the button to appear.)

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



The error list shows you:
- The error message itself
- The severity
- The line number
- The source location
- The connection name

**Fixing errors**

Next you will fix the errors in your source.

To fix an error in your source go to 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 should be EMESS. One good way of finding the correct name is the content-assist.
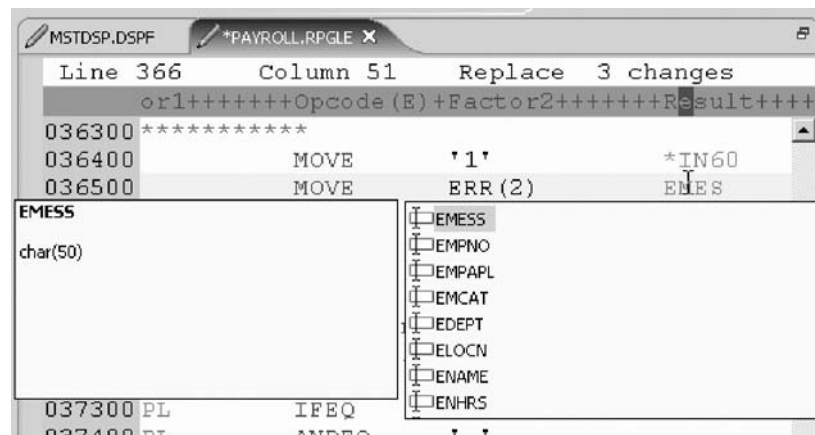
**Note:** The Outline view must be populated to use content-assist.

2. Select the misspelled variable and press **CRTL+spacebar.**

   If the variable starts correctly the selection presented contains the correct name.

3. 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.
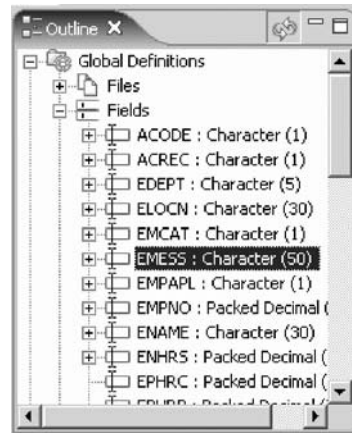


   The next error is a RNF7030 as well.

4. Double-click RNF7030. Fix it in the editor.

5. RSNTAX should really be RSNTAG.

   Make the appropriate change.

6. Go to the next error RNF5184.

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

7. Double-click RNF5184.

8. 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 RFN5178. 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. You can use the Indent view to determine where the ENDSR is missing.

9. Move to line 398 in the editor.

```
MSTDSP.DSPF        *PAYROLL.RPGLE X       REFMST.PF
  Line 398         Column 8       Replace   5 changes
        .......1....+....2....+....3....+....4....+
  039700       C                        END
  039800       C*                       ENDSR
  039900       C*
  040000       C****************************************
  040100       C*   ACDESR subroutine verifies the tim
  040200       C*   all maintenance selections.
  040300       C****************************************
  040400       C     ACDESR          BEGSR
  040500       C*
  040600       C*   Housekeeping, clear display fields
  040700       C*
  040800       C                        EVAL       EMESS =
  040900       C                        EVAL       *IN60 =
  041000       C                        EVAL       *IN90 =
  041100       C*
  041200       C*   The following   statements perform
  041300       C*   two functions.   First they determi
  041400       C*   requested and branche to the appro
  041500       C*   they determine if the maintenance
  041600       C*
  041700       C                        SELECT
  041800       C                        WHEN       ACODE =
  041900       C                        EXSR       ADDCDE
  042000       C                        WHEN       ACODE =
  042100       C                        EXSR       CHGCDE
```

10. 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 messages according to these severities by using the filter menu.
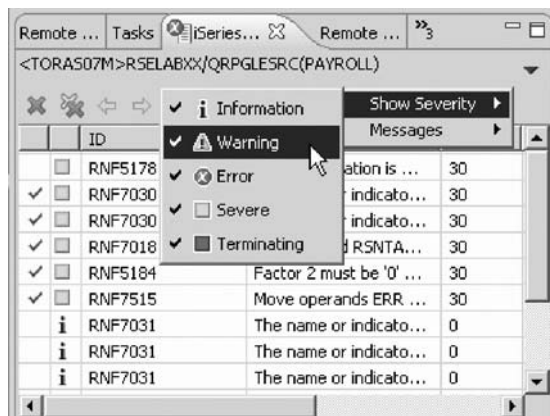


11. Click the arrow in the iSeries Error List title bar.
12. Click **Show Severity** on the pop-up menu.
13. Clear the severities you don't want to see in the list (**Warning** for example).

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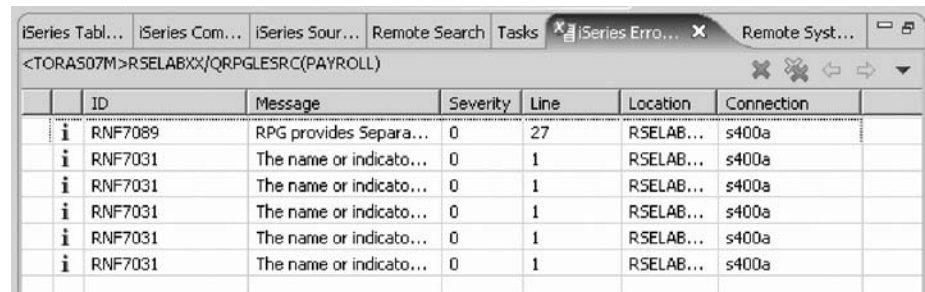
Now before you loose any of your changes, it's a good idea to save them. Make sure the member is selected. You then verify the source again to make sure that all the errors are fixed.

You can save the member using one of these ways:

1. Click **File > Save** from the workbench menu.

2. Click the Save icon 🖫 in the workbench toolbar.

3. Press **Ctrl+S**.

   Changes are uploaded to the iSeries.

4. Verify your source again.

| | | ID | Message | Severity | Line | Location | Connection | |
|---|---|---|---|---|---|---|---|---|
| | i | RNF7089 | RPG provides Separa... | 0 | 27 | RSELAB... | s400a | |
| | i | RNF7031 | The name or indicato... | 0 | 1 | RSELAB... | s400a | |
| | i | RNF7031 | The name or indicato... | 0 | 1 | RSELAB... | s400a | |
| | i | RNF7031 | The name or indicato... | 0 | 1 | RSELAB... | s400a | |
| | i | RNF7031 | The name or indicato... | 0 | 1 | RSELAB... | s400a | |
| | i | RNF7031 | The name or indicato... | 0 | 1 | RSELAB... | s400a | |

iSeries Tabl... | iSeries Com... | iSeries Sour... | Remote Search | Tasks | iSeries Erro... ✕ | Remote Syst...

&lt;TORAS07M&gt;RSELABXX/QRPGLESRC(PAYROLL)

Everything should be ok. You should see only severity 0 messages. You are ready to compile the program.

You have verified your source and fixed any errors and now you are ready to begin "Exercise 4.2: Compiling source remotely."

# Exercise 4.2: Compiling source remotely

Before you begin, you must complete "Exercise 4.1: Verifying the source" on page 63.

The remote compile capability is part of the Remote System Explorer. It gives you a workstation interface to submit requests to compile, bind, or build objects on the iSeries 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 view, 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.

**Changing compile preferences**

To change the preferences to run the compile interactive:

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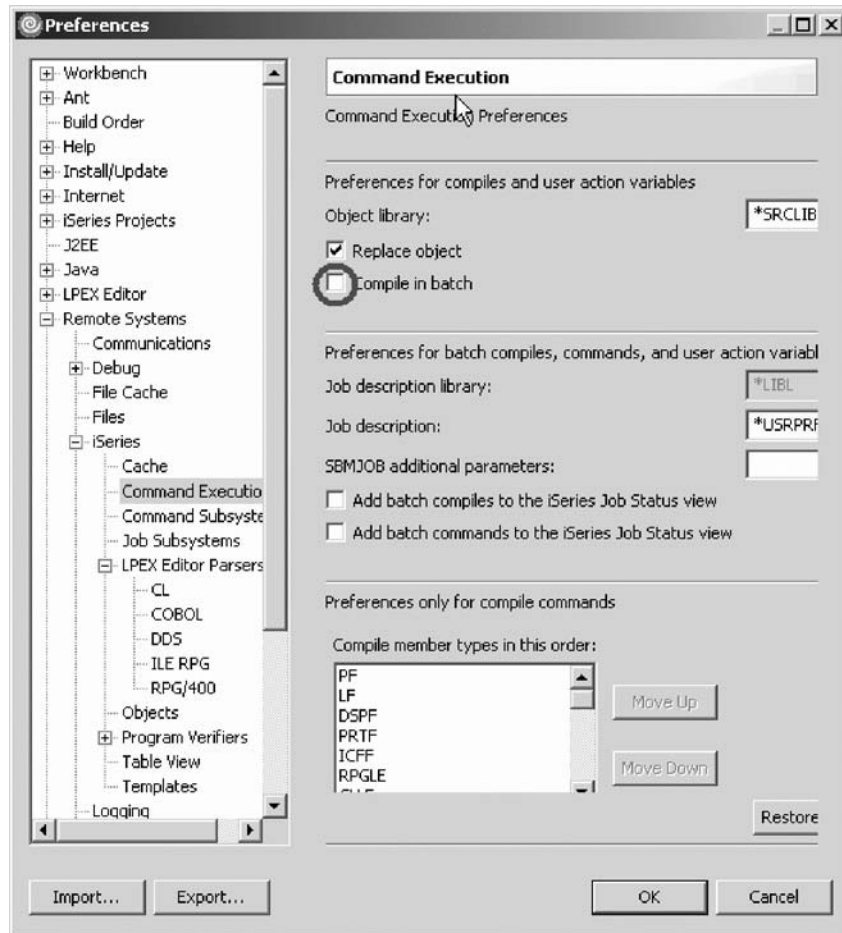
2. In the left pane of the Preferences window, expand **Remote Systems**.
3. Expand **iSeries** under Remote Systems.
4. Click **Command Execution** under iSeries.
5. In the right pane of the Preferences window, clear the **Compile in batch** check box.
6. Click **OK** to return to the Remote System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

**Invoking the compile command**

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 QRPGLESRC.
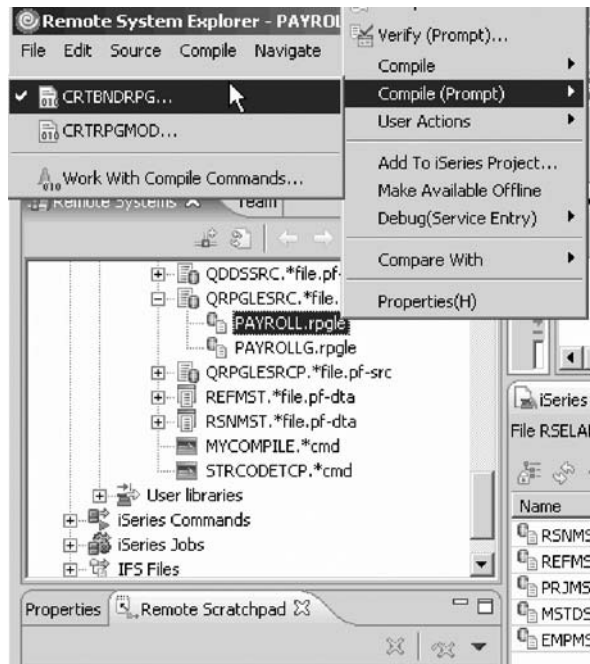
2. Click **Compile (Prompt) > CRTBNDRPG** on the pop-up menu.
   The Create Bound RPG Program (CRTBNDRPG) dialog opens.
3. In the **Debugging views** list, select the **\*ALL** parameter.
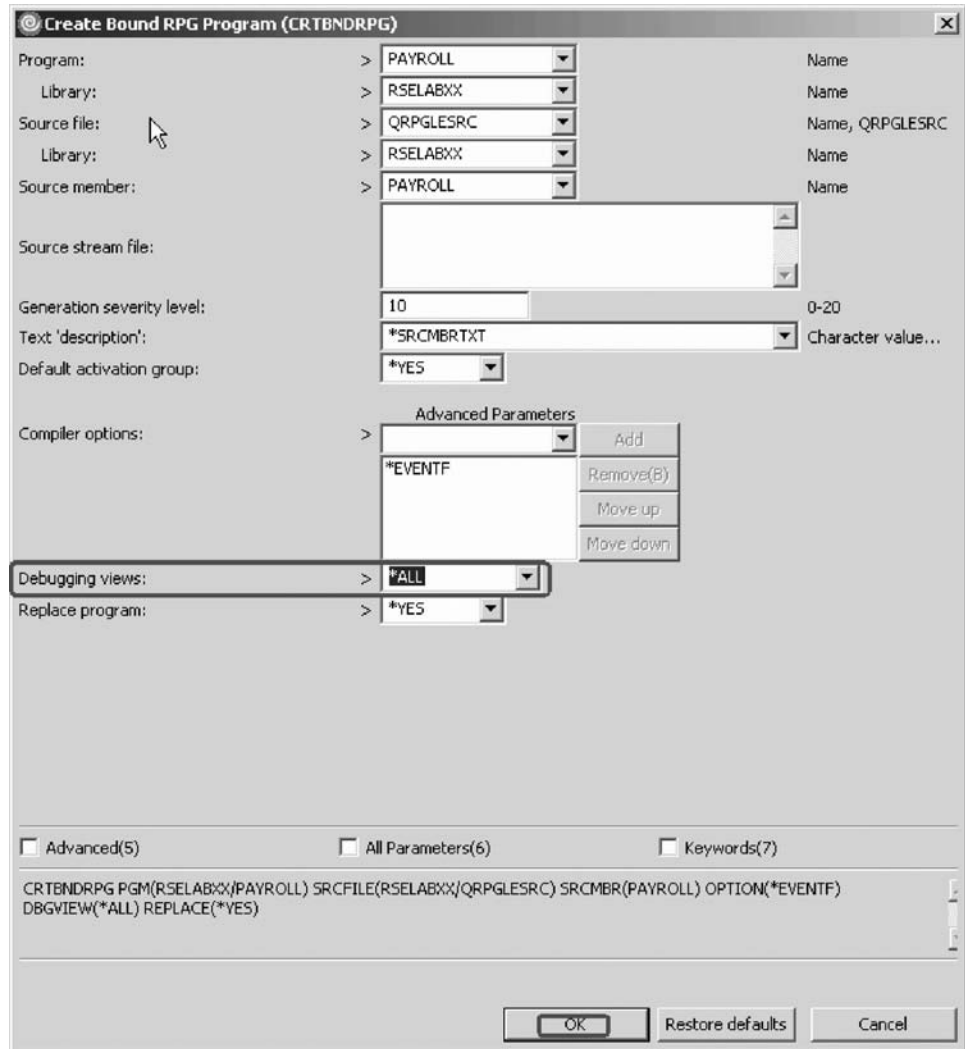
If you want to see the other parameters available, click **Advanced**.

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

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.

```
CRTBNDRPG PGM(RSELABXX/PAYROLL)
SRCFILE(RSELABXX/QRPGLESRC)
SRCMBR(PAYROLL) OPTION(*EVENTF)
DBGVIEW(*ALL) REPLACE(*YES)
Program PAYROLL placed in library RSELABXX. 00
highest severity. Created on 01/04/05 at
14:03:39.
Cause . . . . . :  Program PAYROLL was
successfully created in library RSELABXX. The
highest message severity that resulted was 00.
The program creation date and time are 01/04/05
and 14:03:39.
```

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.

You have set compile preferences, invoked the compile command, checked for a successful compile and now you are ready to begin "Exercise 4.3: Submitting iSeries commands in the iSeries Table view."

## Exercise 4.3: Submitting iSeries commands in the iSeries Table view

Before you begin, you must complete "Exercise 4.2: Compiling source remotely" on page 67.

You can use the iSeries Table view inside the Remote System Explorer to submit commands to the iSeries. You can run commands from 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 click either **Prompt** to specify parameters and then **Run** or just click **Run**. When you run a command, the **Messages** field is populated with the messages from the command. When you select a message, the **Details** button is enabled. When you click this button, the message and its help is displayed.

Also note that besides the iSeries Table view, you can also use the Remote Systems view to run commands and programs. Which one you choose depends on your personal preference. 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.

In the **Commands** field, you select where you want to run the command. The choices are Normal, which means that the command will run in the RSE communication server job, Batch or Interactive.

To change the library list:
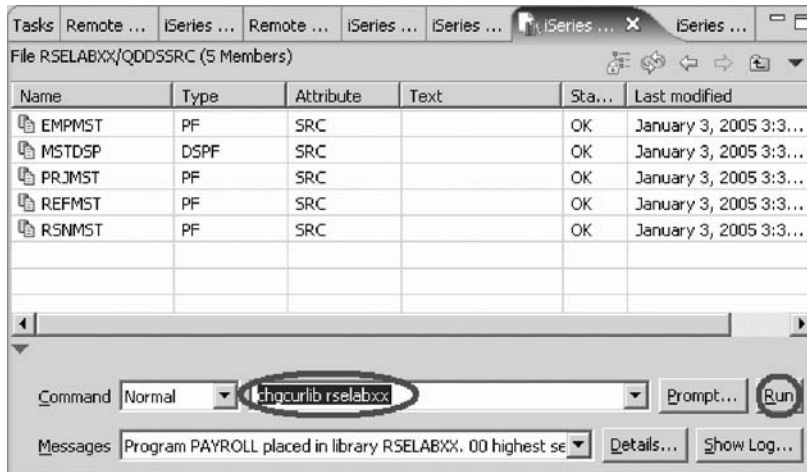
1. Click the **iSeries Table View** tab from the views at the bottom of the workbench.
2. In the **Command** field type, `CHGCURLIB RSELABxx` for example.

   **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 see an error message because the iSeries Table view is not linked to an active connection.

   If you see this message, click **OK**.

   a. In the Remote Systems view, right-click QRPGLESRC.
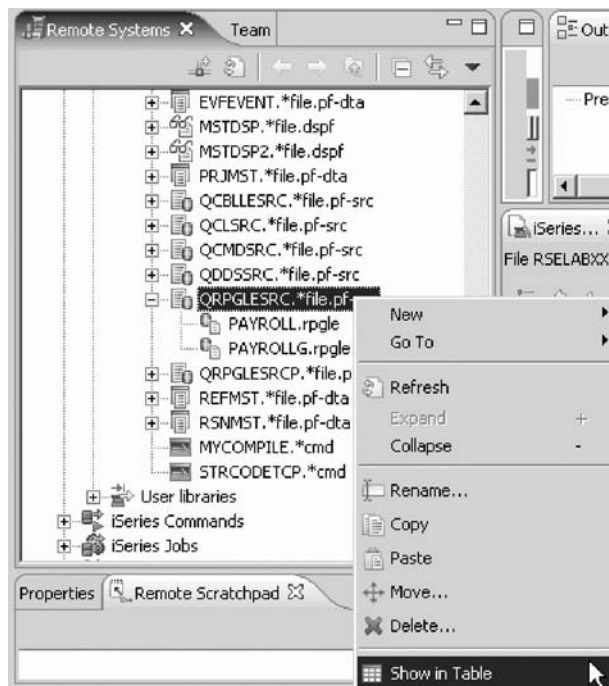


   b. Click **Show in Table** on the pop-up menu.

      The iSeries Table view is now populated with the member in the selected source file.

   c. 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.
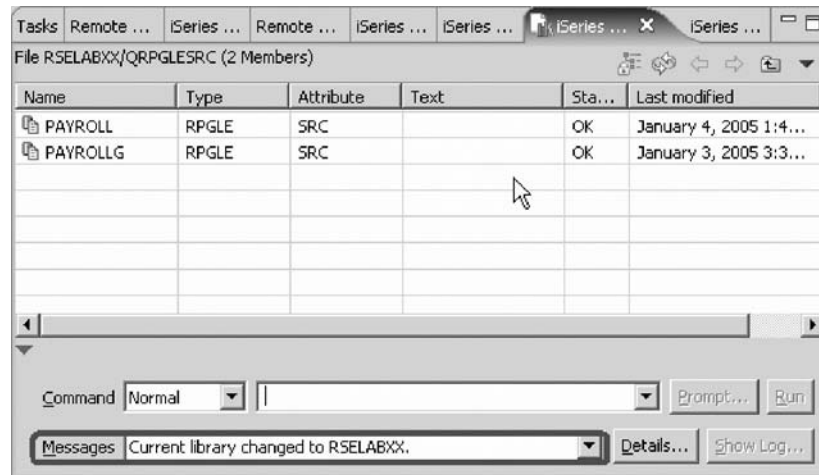


4. Back in the workbench in the Remote Systems view, right-click **Library list** and click **Refresh**.
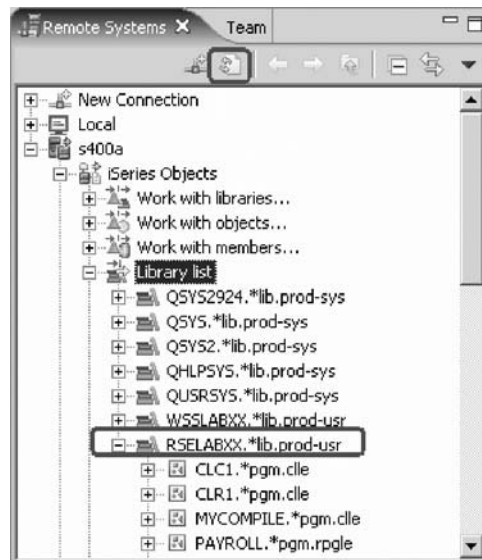


5. You will see a small green asterisk beside the RSELABxx library to indicate it as your current library.

6. 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™.

You have submitted a command to change the current library in the command line of the iSeries Table view now you are ready to begin "Exercise 4.4: Running commands and programs."

## Exercise 4.4: Running commands and programs

Before you begin, you must complete "Exercise 4.3: Submitting iSeries commands in the iSeries Table view" on page 71.

As you know already, you can run programs and commands from the Remote Systems view or the iSeries Table view in three ways:

1. In the Remote System Explorer communications server job.

   This is the one you are using currently.
2. In a batch job.
3. In an interactive job (to test 5250 applications).
4. In a server job

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, and you need to first run a STRRSESVR connectionName command to associate the emulator with a particular connection in the Remote System Explorer communications server. A multi-threaded debug session creates a new server job and this way keeps the RSE communications server job free for other tasks.

**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.
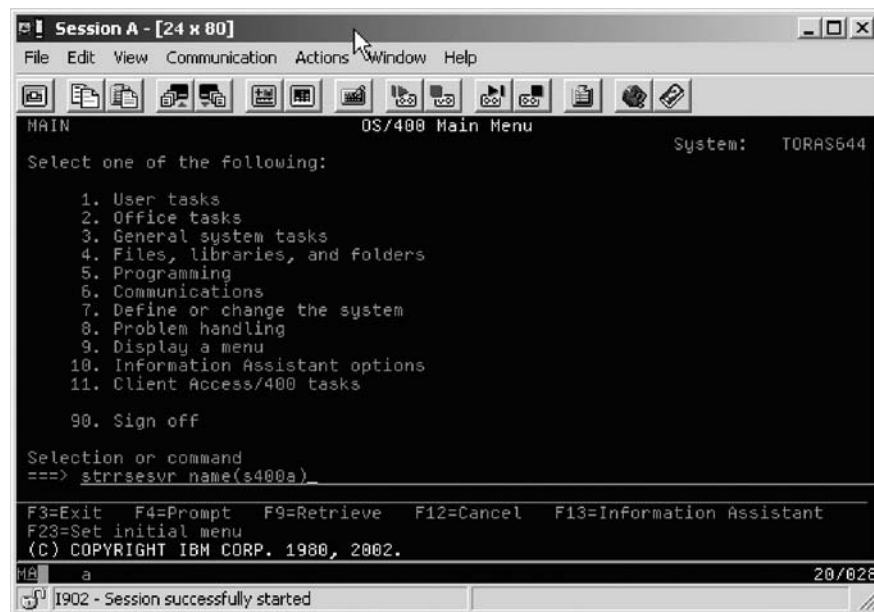


3. In the command line, type the command

   `STRRSESVR connectionName`
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.

**Running the payroll program**

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

Return to the workbench.

To run the program:
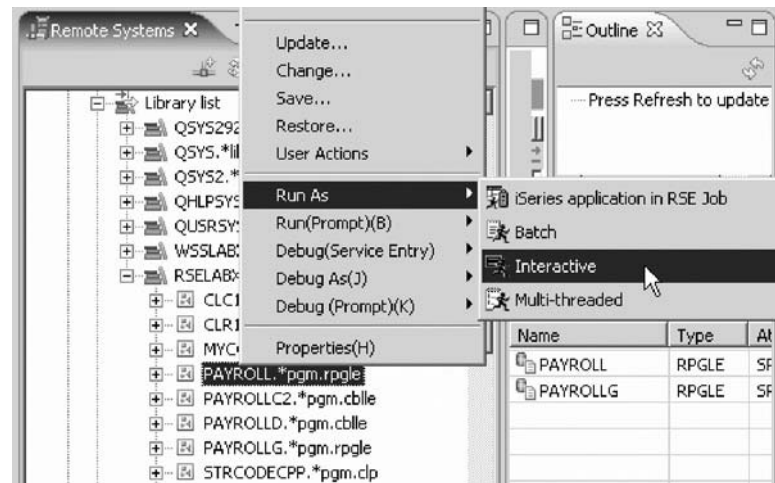


1. In the Remote Systems view, locate the PAYROLL program that you created.
2. Right-click the PAYROLL program.
3. Click **Run As > Interactive** on the pop-up menu.
4. Switch to your 5250-emulation session.

   You will see the payroll program menu.



5. Type x beside **Employee Master Maintenance**.
6. Press **Enter**.
7. Type 234 for the Employee Number.
8. Type A for the Action Code to add employee **234**.
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 applications.

14. To get control of the interactive job, right-click **iSeries Objects** and click **Release Interactive Job** on the pop-up menu.

    You can also choose to disconnect a session. You would right-click the connection and click **Disconnect** on the pop-up menu.

Now you are ready to review your knowledge of this module by taking the quiz. You can also apply what you have learned in this module by completing the practice tasks detailed in More practice.

**Quiz**

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 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)

5. 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)

6. From the iSeries Table view you can:
   a. List and sort libraries, objects, and members
   b. Copy, rename, delete, edit, compile and run (for programs), open with and browse (for members) in the view from the pop-up menu
   c. Transfer files from one system to another
   d. All of the above

7. 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. Multi-threaded

    e. All of the above

8. Interactive programs require a 5250 emulator, and you need to first run a STRRSESVR connectionName command to associate the emulator with a particular connection in the Remote System Explorer. (T, F)

**More practice**

Given your experience using the Program Verifier and 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 verify your source, choose insert all error messages into the editor from the Error list. Use the product online help to assist you in these tasks.

**Module recap**

You have completed Chapter 4, "Module 4. Verifying and compiling source," on page 63. You have learned how to:

- Check for semantic errors on your workstation
- Start the Program Verifier tool
- Use the iSeries Error list to locate each error in the source
- Use content-assist to fix an error
- Save your source
- Re-verify source
- Change compile preferences
- Invoke the compile command
- Change the current library using the **Command** field in the iSeries Table view
- Start an interactive connection
- Invoke the payroll program

Now that you know how to verify and compile source from the Remote Systems view and run the payroll program interactively, you can move onto debugging your payroll program. Continue to Chapter 5, "Module 5. Debugging a program," on page 79.

# Chapter 5. Module 5. Debugging a program

This modules teaches you how to debug a CL and ILE RPG program.

You will 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 this module, you will:
- Invoke the debugger from the Launch Configurations window
- Add a breakpoint
- Add a conditional breakpoint
- Edit a breakpoint
- Monitor a variable through the Monitors view
- Step into your payroll program
- Show a Listing view
- List the call stack entries in the Debug view
- View all breakpoints
- Remove a breakpoint
- Monitor storage
- Set a Watch breakpoint
- Close the debugger

**Exercises**

The exercises within this module must be completed in order:

**Time required**

This module will take approximately **30 minutes** to complete.

# Exercise 5.1: Introducing the Integrated iSeries Debugger

The Integrated iSeries 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 multi-threaded 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® and ILE RPG, COBOL and ILE COBOL, C, C++ and CL.

In the following exercises you will be given the opportunity to learn about some of the basic features of the Debugger. For the purpose of these exercises you will debug a CL and an ILE RPG program. Don't worry if you don't know RPG.

You know the basic features of the debugger and now you are ready to begin "Exercise 5.2: Starting the integrated debugger."

# Exercise 5.2: Starting the integrated debugger

Before you begin, you must complete "Exercise 5.1: Introducing the Integrated iSeries Debugger."

You will be working with the ILE RPG program PAYROLLG.

**Note:** 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 exercises in Chapter 3, "Module 3. Editing source," on page 25.

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 a 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.

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.
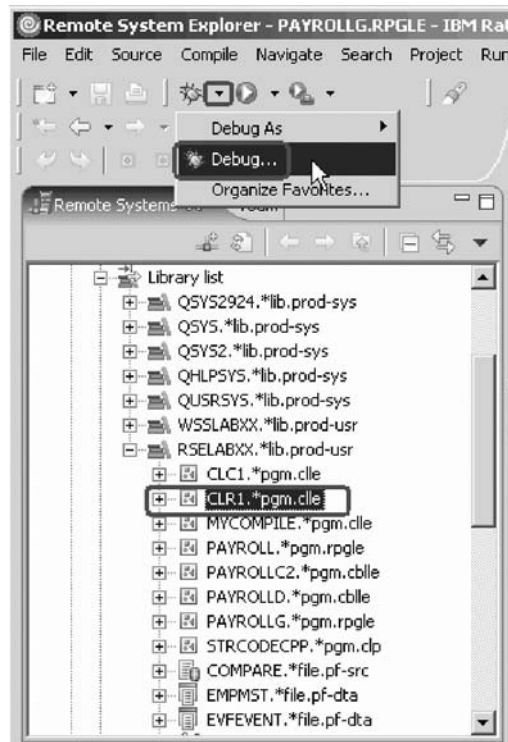


2. Expand library RSELABxx, if it isn't expanded already.
3. Select program CLR1 in library RSELABxx.
4. Click the arrow beside the **DEBUG** icon 🐞 ▾ on the workbench toolbar.
5. Select **Debug** from the list.
   The Debug Launch Configurations window opens.
6. Select **iSeries: Debug Interactive Application** under the **Configurations** list.
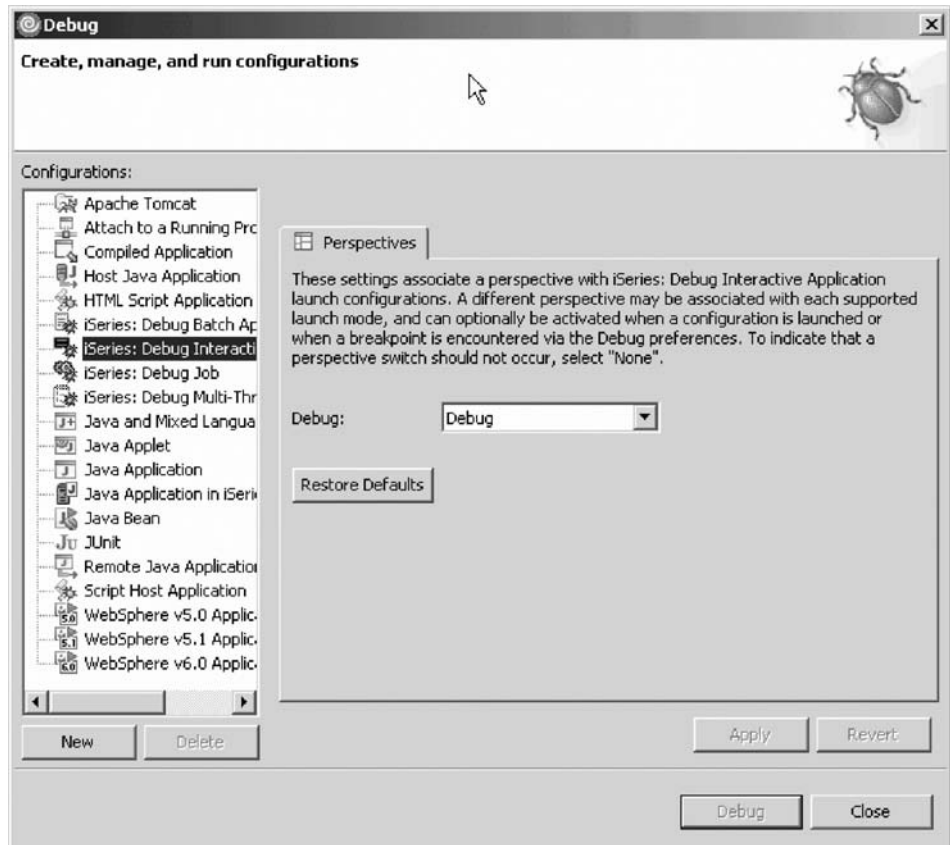
7. Click **New**.

> **Note:** You could also use **Debug(prompt) > Interactive** from the pop-up menu.

The right pane of the Debug Launch Configurations window opens.
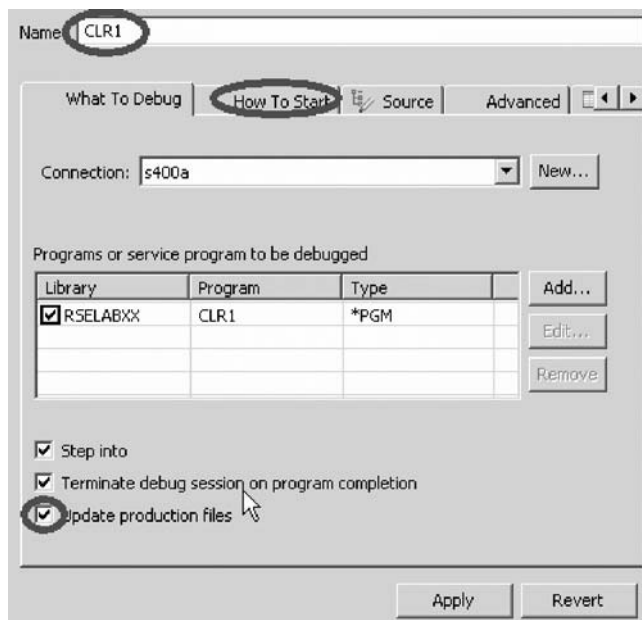


8. In the **Name** field, type the program name CLR1.
9. Select the **Update production files** check box.
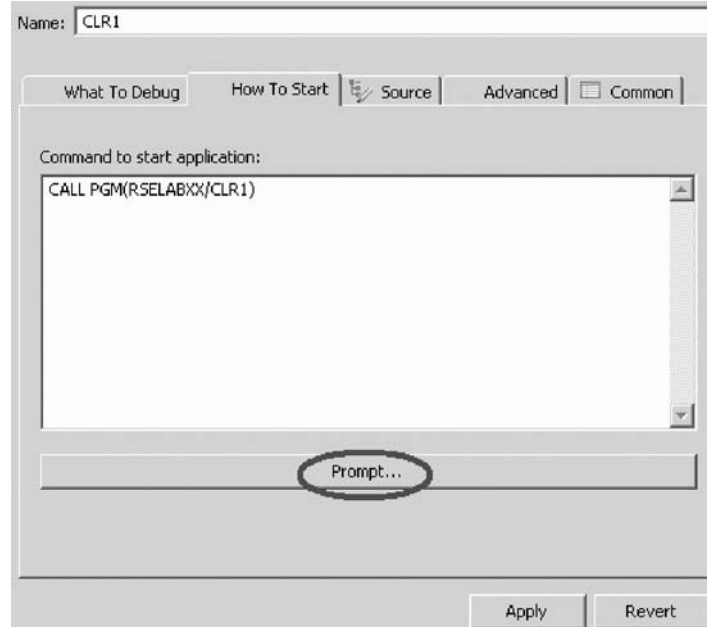
10. Click the **How To Start** tab.

    The How To Start page opens.



    By default, the page contains a call for the program selected in the Remote
    Systems view.

11. Click **Prompt**.

    The Call Program (CALL) window opens.
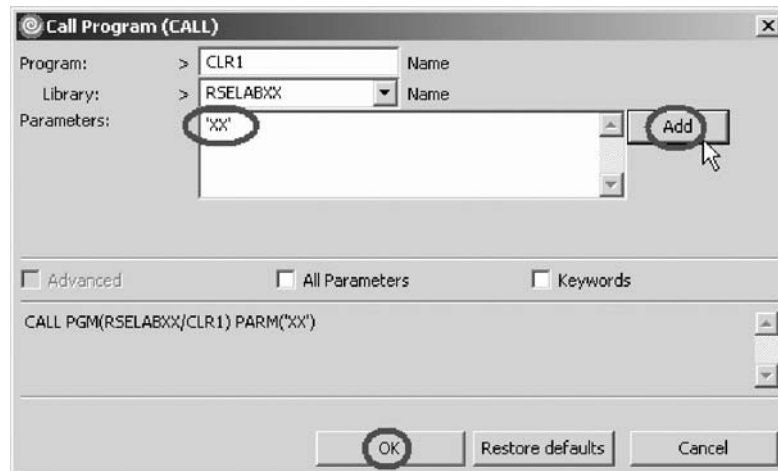


12. In the **Parameters** field, type `'XX'` where 'XX' is your workstation number.
    Click **Add**.

    The parameter value will appear in the lower list.

    If you forget to click **Add**, the parameter will be missing from the CALL
    program command and you will receive an error when the program tries to
    call the payroll program.

13. Click **OK**.

    The complete start command for the program appears.
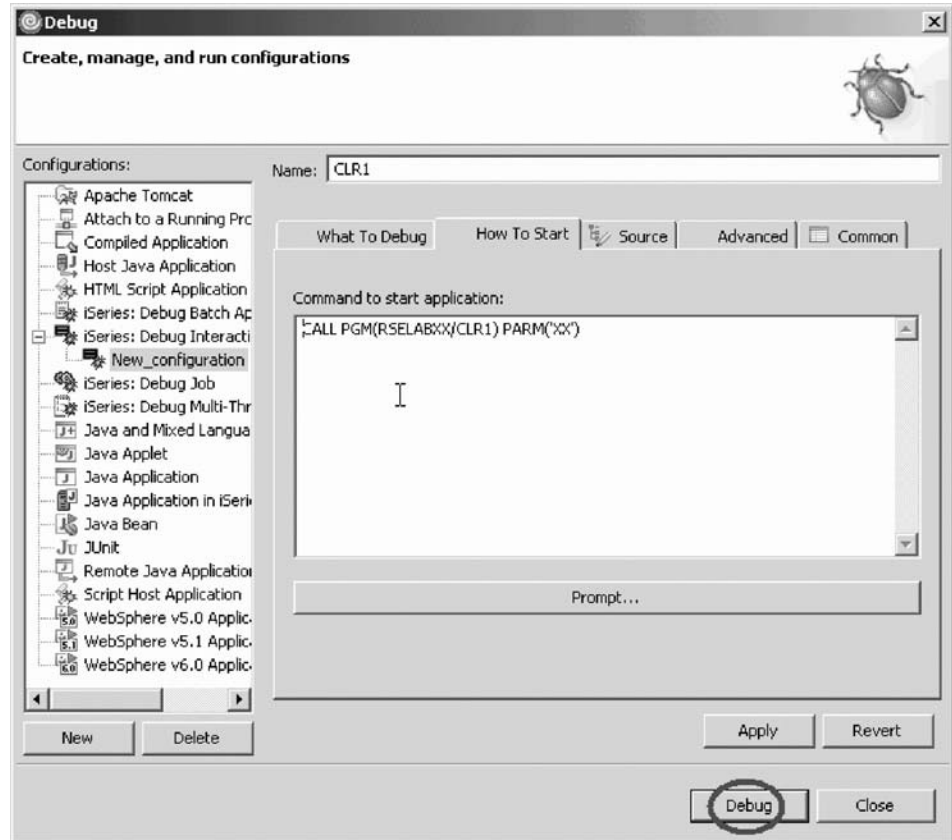
14. Click **Debug**.

    The Debug perspective opens.

    If not, you may see this error message.
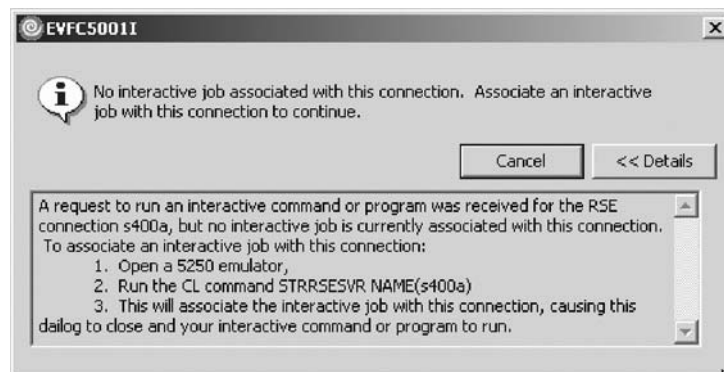


    The Remote System Explorer communications server has been shut down in the meantime. Go to your 5250 emulator and restart the Remote System Explorer communications server following the instructions in the message. You don't have to cancel the message. It will be removed as soon as the connection between 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.
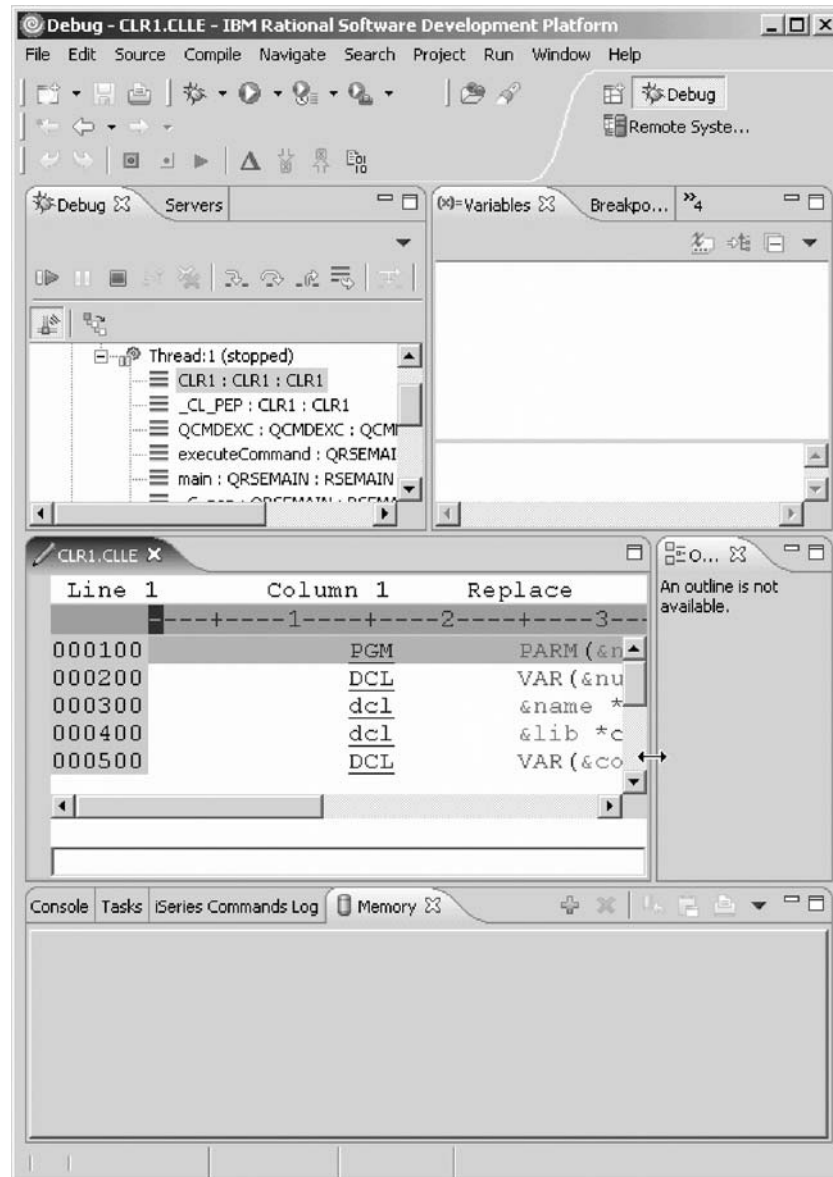
You have started an interactive debug session and now you are ready to begin
"Exercise 5.3: Setting breakpoints."

## Exercise 5.3: Setting breakpoints

Before you begin, you must complete "Exercise 5.2: Starting the integrated
debugger" on page 80.

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

To set a breakpoint:
1. Position the cursor on line 11.
2. Right-click anywhere on line 11.
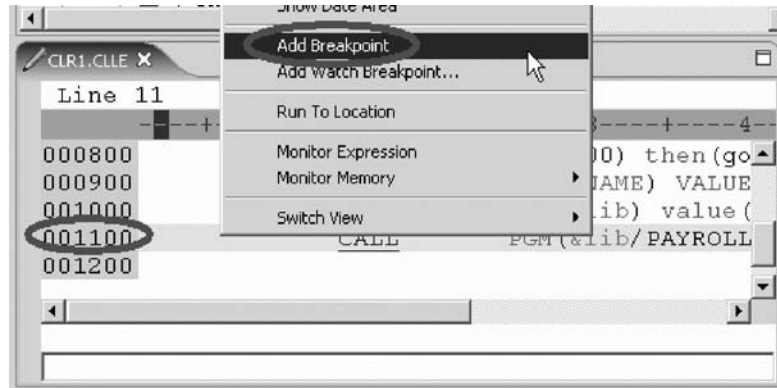
3. Click **Add breakpoint** on the pop-up menu.

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 to stop in the loop when it loops the 99th time.

**Adding a conditional breakpoint**

To add a conditional breakpoint:

1. Position the cursor on line 8.
2. Right-click on line 8.
3. Click **Add breakpoint** on the pop-up menu.
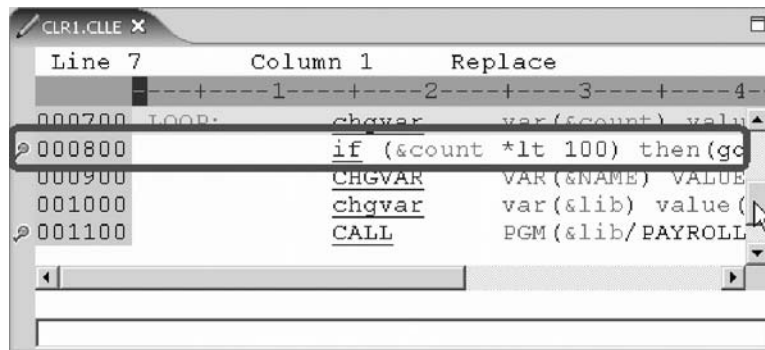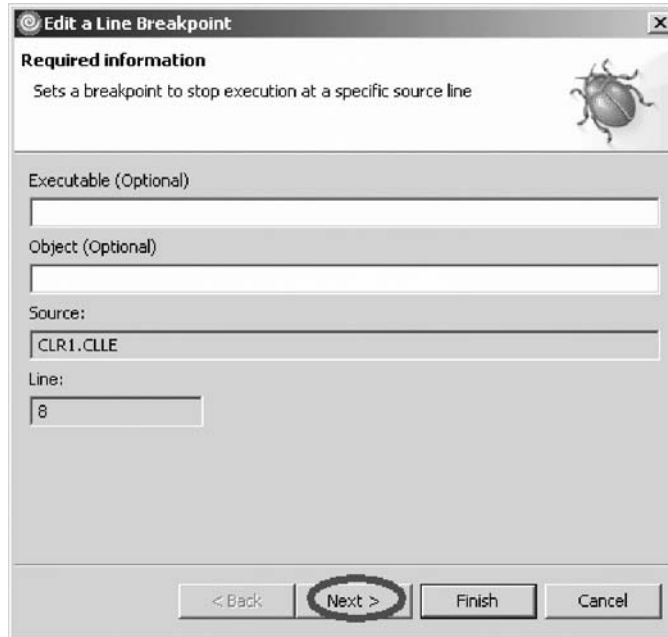


4. Click the **Breakpoints** tab in the upper right pane of the Debug perspective.

The Breakpoints view opens.
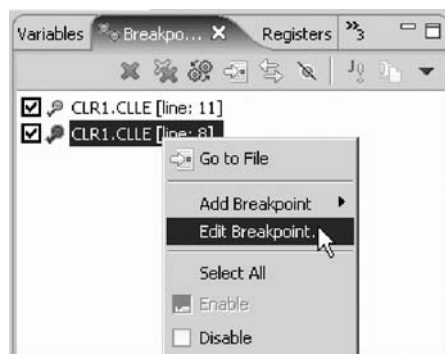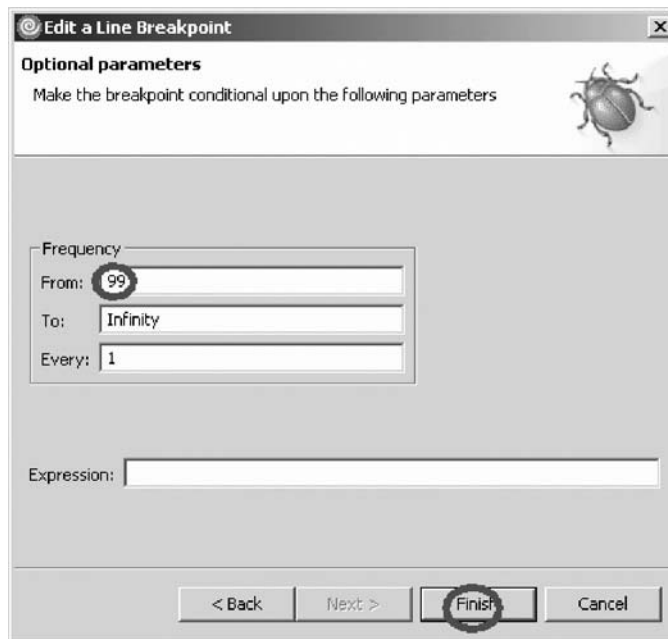


5. In the **Breakpoints** view right-click the breakpoint for line 8.

6. Click **Edit Breakpoint** on the pop-up menu.

   The Edit a Line Breakpoint window opens.



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**.

You have added a breakpoint including a conditional breakpoint to your source and now you are ready to begin "Exercise 5.4: Monitoring variables" on page 88.

# Exercise 5.4: Monitoring variables

Before you begin, you must complete "Exercise 5.3: Setting breakpoints" on page 85.

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.
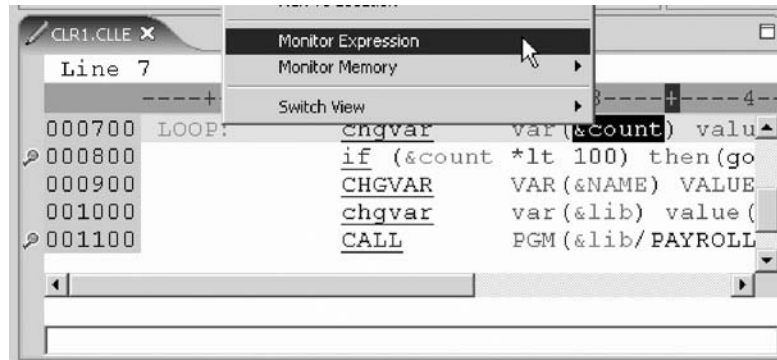


2. Right-click &count.
3. Click **Monitor Expression** on the pop-up menu.

   The Monitors view opens.
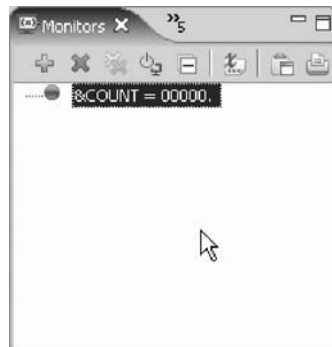


   The variable appears in the Monitors view. Its current value is zero.

   **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.
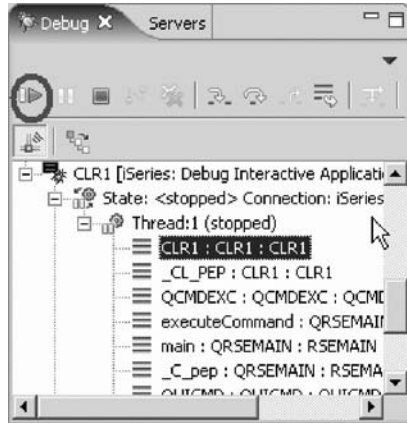
4. Click the **Resume** [▷] icon from the Debug toolbar.

   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.

   **If you do not see the error message below, go to** "Exercise 5.5: Stepping into a program" on page 93.

   **Error Handling**

   If you forget to add the parameter to the CALL program command when you debug the program, you will see this error message.
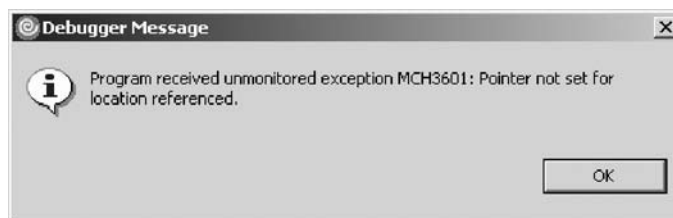


   a. Click **OK**.

   b. Click the **Terminate** [■] icon on the Debug toolbar.

      The debug session terminates on the workstation but the exception waits for input from the 5250 emulation session.

      If you closed the Debug view by mistake, you will need to re-open the Debug view and then terminate the debug session on the workstation.
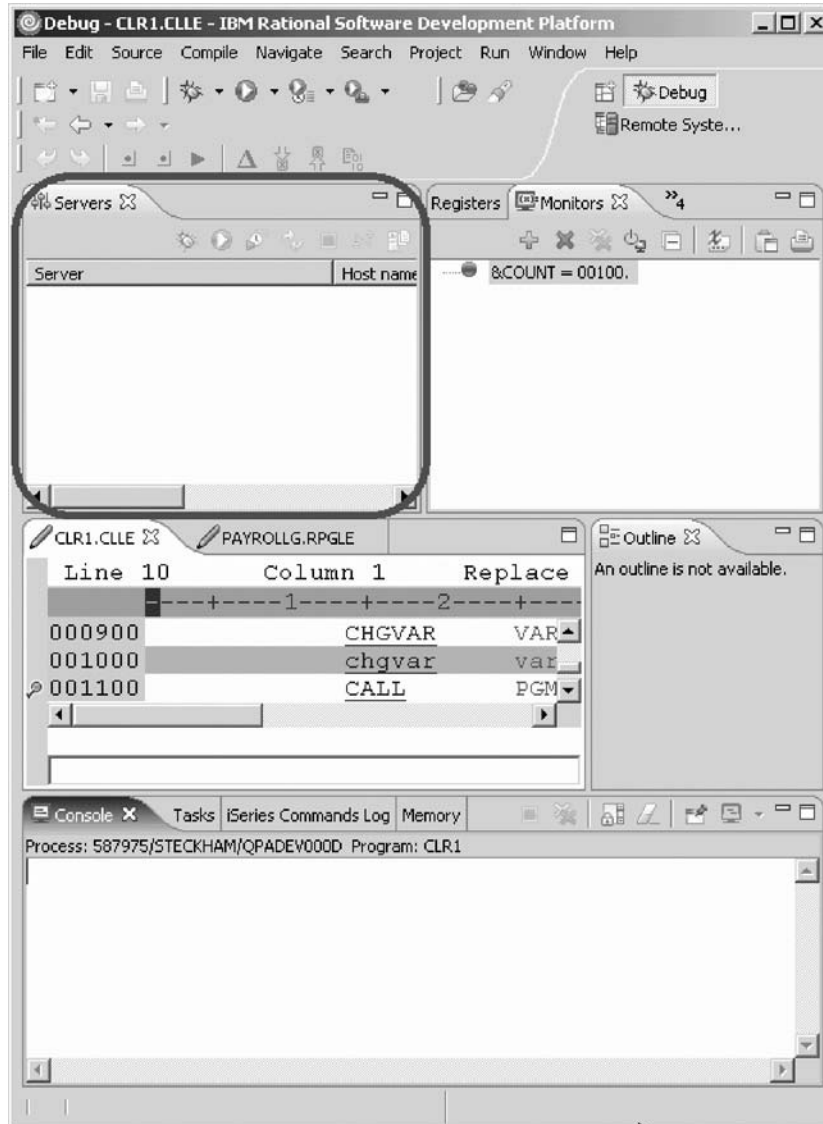
Click **Window > Show View > Debug**.
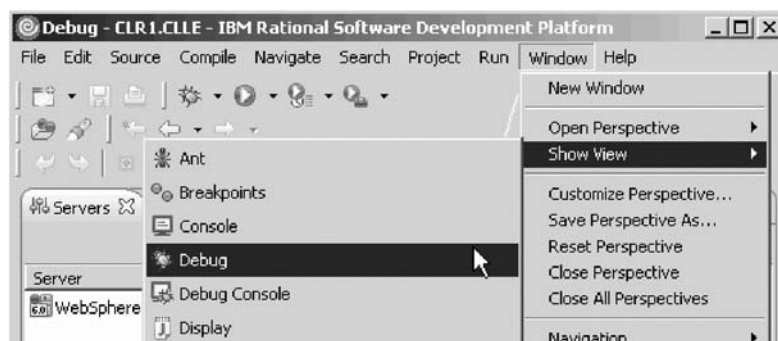


Remember to terminate the Debug view if you haven't done so already.

c. Go to your 5250 emulator and press **Enter** until the program messages complete and the Remote System Explorer communications server screen appears again.
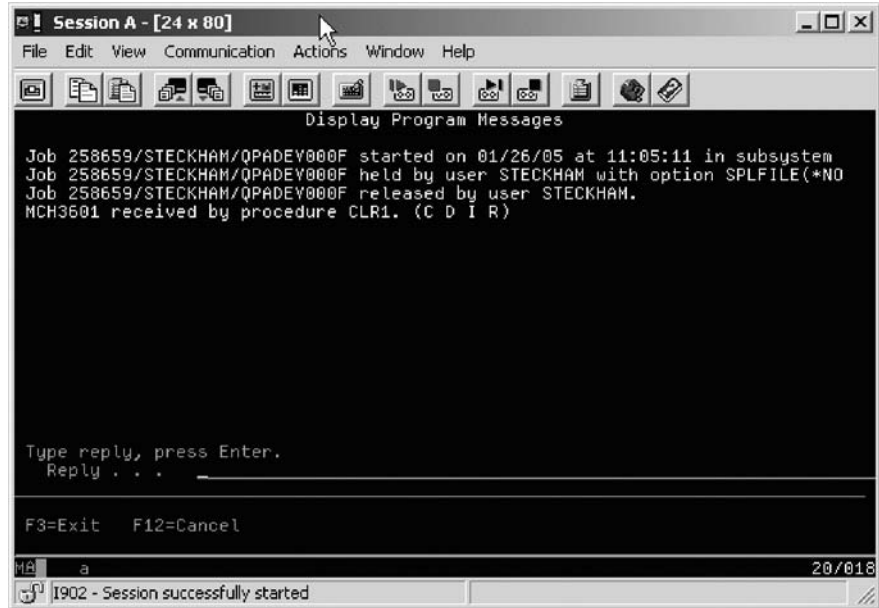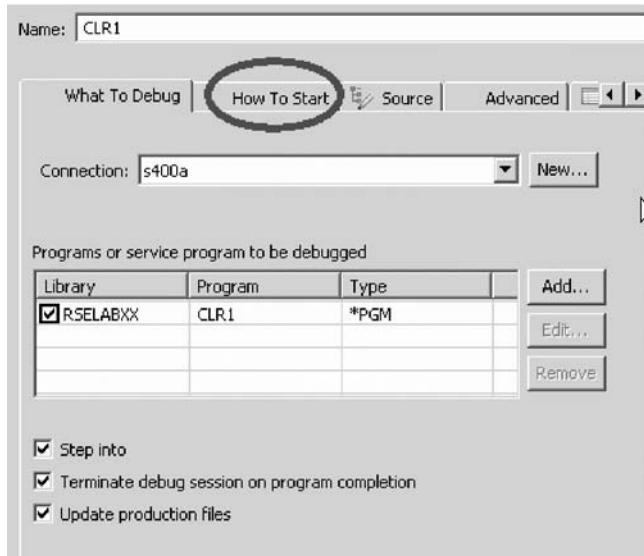
d. In the workbench, click the **Remove all terminated launches** icon on the Debug toolbar to clean up the Debug view.
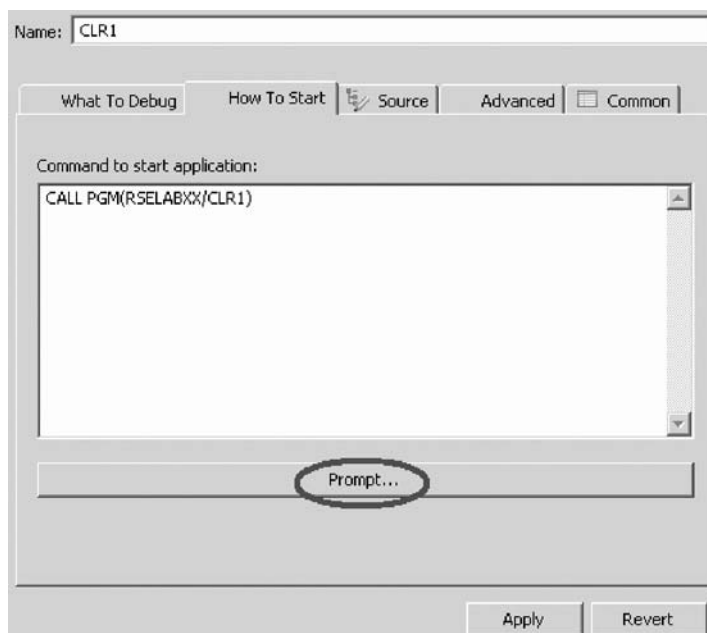


e. Click the arrow beside the Debug icon on the workbench toolbar to start a new debug session.

f. Select **Debug** from the list.
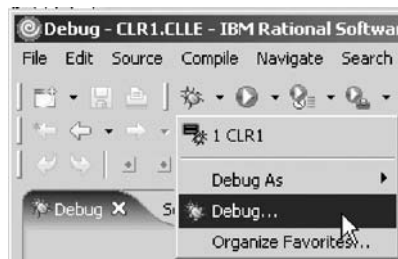


The Debug Launch Configurations window opens.

g. Click the **How To Start** tab.

The How To Start page opens.



By default, the page contains a call for the program selected in the Remote Systems view.

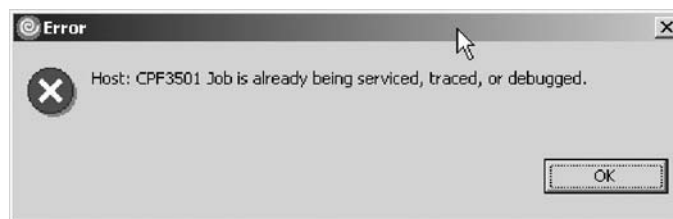h. Click **Prompt**.

The Call Program (CALL) window opens.
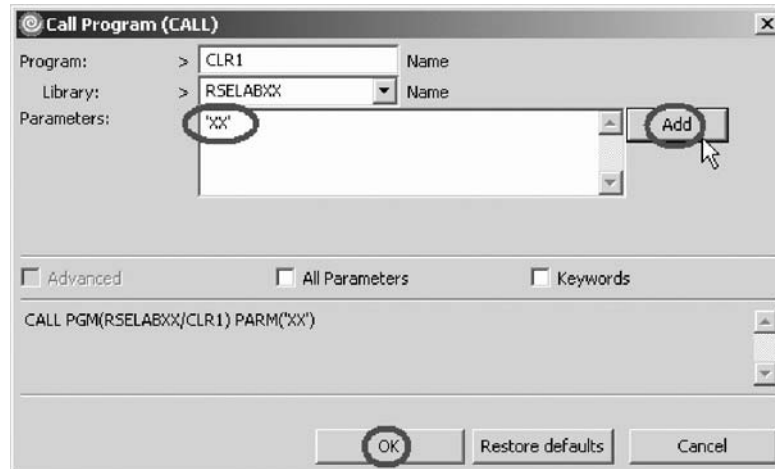
i. In the **Parameters** field, type `'XX'` where 'XX' is your workstation number. Click **Add**.

The parameter value will appear in the lower list.

j. Click **OK**.

k. Click **Debug**.

If you did not terminate the debug session, you will see this error message.



Complete the steps shown earlier in the section called **Error Handling** to terminate the debug session then start the debug session again by clicking the arrow beside the Debug icon on the workbench toolbar and selecting **CLR1** from the list. Breakpoints and monitors are restored.

You have monitored the variable &count and now you are ready to begin "Exercise 5.5: Stepping into a program."

## Exercise 5.5: Stepping into a program

Before you begin, you must complete "Exercise 5.4: Monitoring variables" on page 88.

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 the PAYROLLG program.

To step into a program:

1. Click the **Step into** icon on the Debug toolbar.
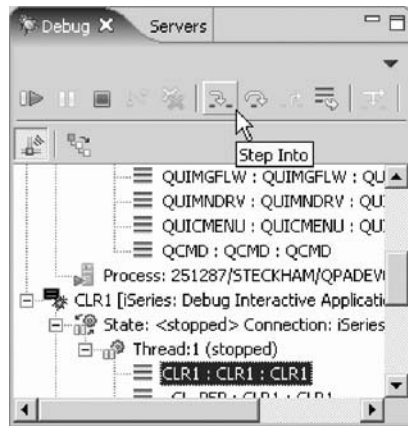
The source of PAYROLLG is displayed. Depending on the option you used to compile the program (*SRCDBG or *LSTDBG for RPG, or *SOURCE, *LIST, or *ALL for ILE RPG), this window displays either the Source or Listing View.

If you specified an incorrect parameter for the CALL program command, you will see this error message.



Complete the same steps as covered in the section called **Error Handling** in "Exercise 5.4: Monitoring variables" on page 88.

2. Right-click anywhere in the Source view and click **Switch view > Show *LISTING** on the pop-up menu.
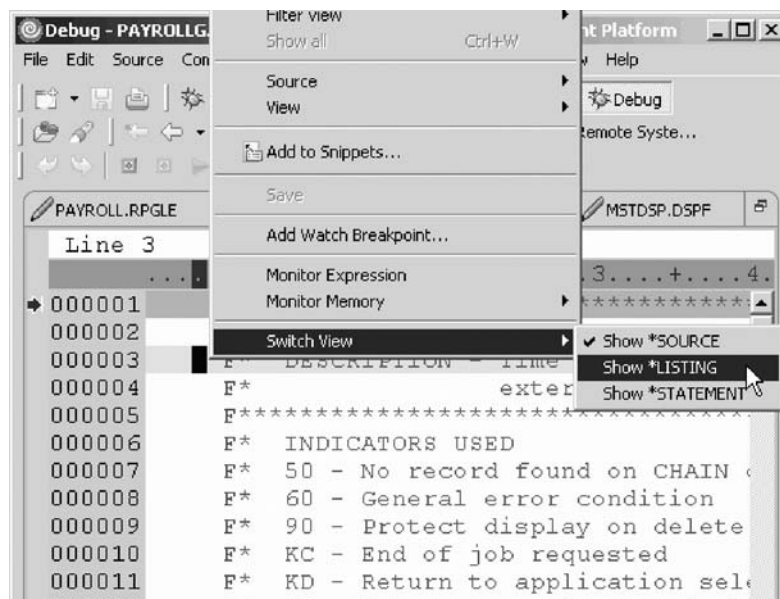


3. Page down in the source and take a look at the expanded file descriptions.

   You don't have any /Copy member in your PAYROLLG program but these would also be shown in a Listing view. Switch back to the Source view.

4. Right-click anywhere in the Source view.

5. Click **Switch view > Show *SOURCE** on the pop-up menu.

You have stepped into PAYROLLG program, switched the view from source to listing and back to source and now you are ready to begin "Exercise 5.6: Listing call stack entries."

## Exercise 5.6: Listing call stack entries

Before you begin, you must complete "Exercise 5.5: Stepping into a program" on page 93.

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, and procedure 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.
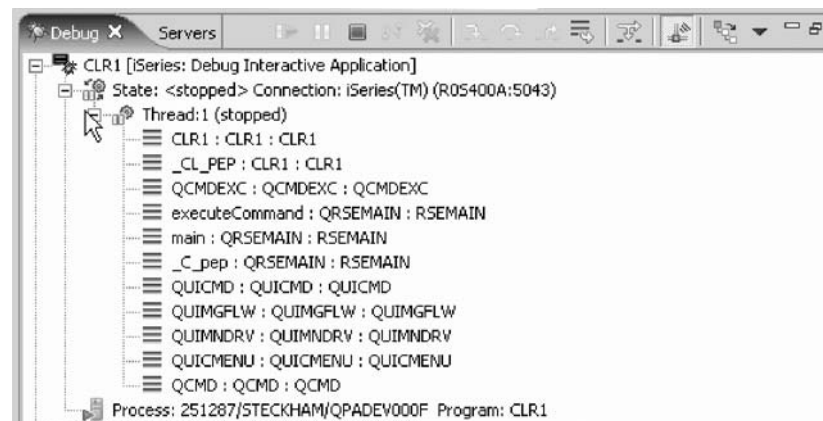


The stack entry allows you to work with and switch between different programs and/or ILE modules.

You have viewed the call stack entries of your program and now you are ready to begin "Exercise 5.7: Setting breakpoints in PAYROLLG."

## Exercise 5.7: Setting breakpoints in PAYROLLG

Before you begin, you must complete "Exercise 5.6: Listing call stack entries."

Now you add some breakpoints in PAYROLLG.

To add breakpoints:
1. Select PAYROLLG in Thread1.
2. In the source view (also called the iSeries default editor) scroll to line 57.
3. Double-click the prefix area of line 57.

   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.
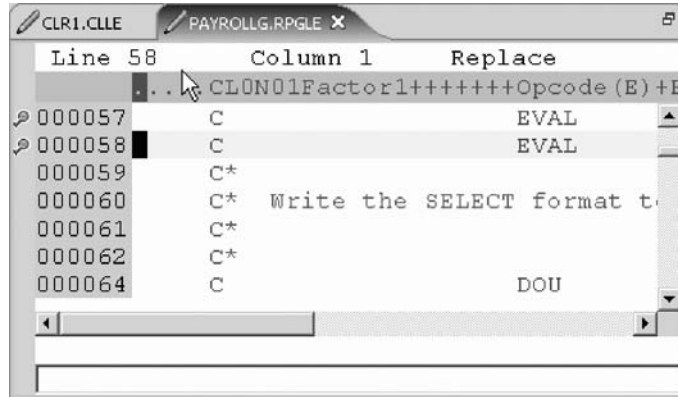
5. Right-click in the prefix area of line 87 and click **Add Breakpoint** on the pop-up menu.
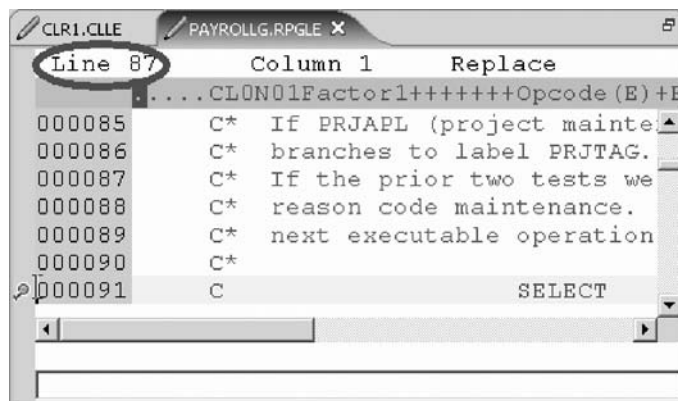


To view all breakpoints, select the **Breakpoints** tab from the top left pane.
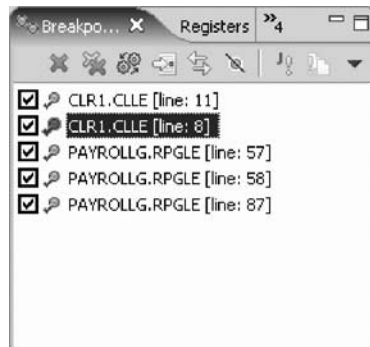


This view shows all breakpoints currently set in your Debug session. This is a convenient place to work with breakpoints. You can remove, disable/enable, add, or edit a breakpoint. These tasks are available from the pop-up menu when you right-click in the view area. Double-click any entry to show the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

You have added several breakpoints to PAYROLLG and now you are ready to begin "Exercise 5.8: Removing a breakpoint in PAYROLLG" on page 97.
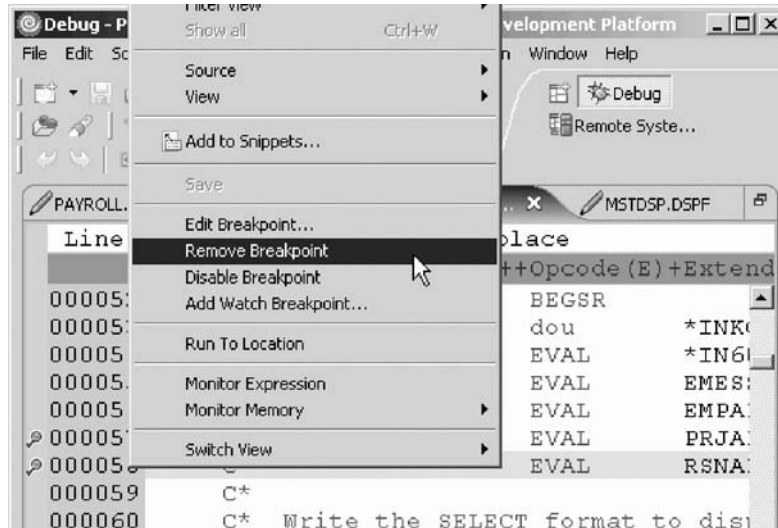
## Exercise 5.8: Removing a breakpoint in PAYROLLG

Before you begin, you must complete "Exercise 5.7: Setting breakpoints in PAYROLLG" on page 95.

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

To remove a breakpoint:
1.  Right-click the prefix area of line 58.
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.

3.  Click the **Resume** [▷] icon from the Debug toolbar.
4.  The program starts running and runs to the breakpoint at line 57.
5.  Click the **Resume** icon again.

The program waits for input from the 5250-emulation session.
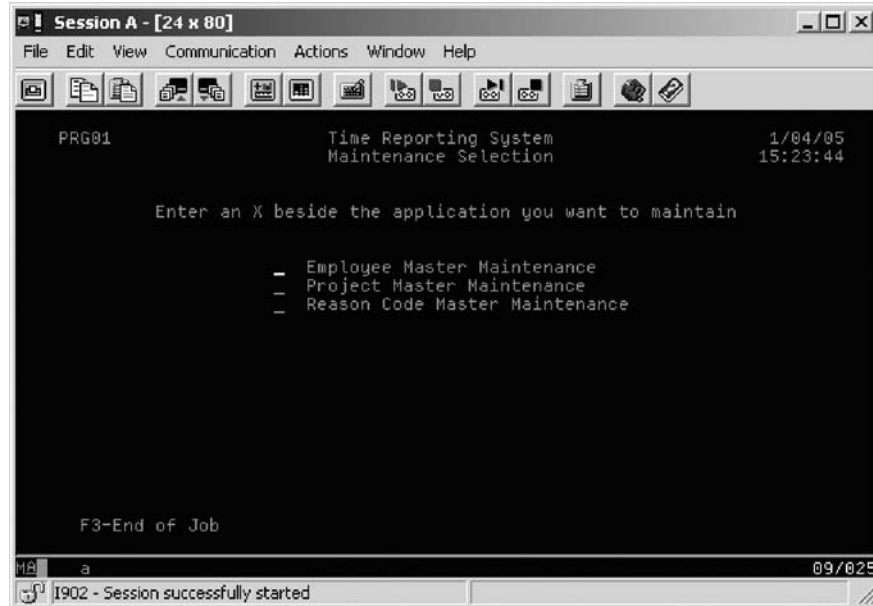
6. Type an X beside the **Project Master Maintenance** option.

7. Press **Enter** in the emulation session.

   The program runs to the breakpoint at line 87.

You have removed a breakpoint from PAYROLLG and now you are ready to begin "Exercise 5.9: Monitoring variables in PAYROLLG."
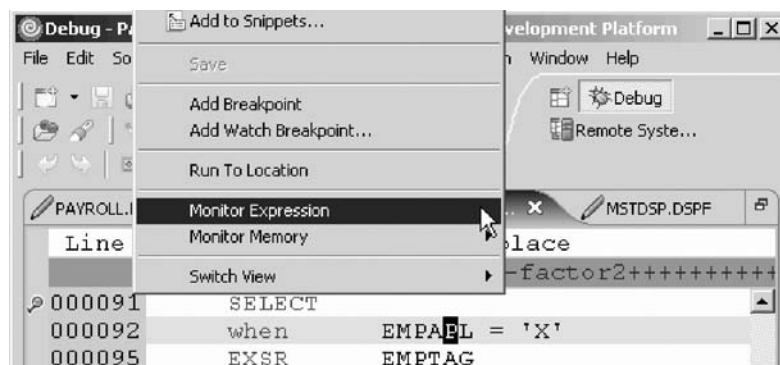
## Exercise 5.9: Monitoring variables in PAYROLLG

Before you begin, you must complete "Exercise 5.8: Removing a breakpoint in PAYROLLG" on page 97.

Now lets monitor variables and change them in PAYROLLG.

To monitor variables:

1. In the source view, double-click the variable EMPAPL on line 88.

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.

Variable PRJAPL equals X because you did select the **Project Master Maintenance** option.

6. In the Monitors view, double-click the variable RSCDE.

The value changes into an entry field.

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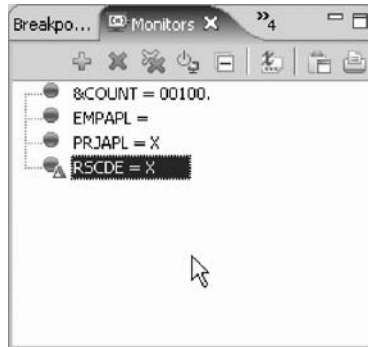


8. Press **Enter**.

The variable is successfully changed.

You have monitored several variables in PAYROLLG and now you are ready to begin "Exercise 5.10: Adding a memory monitor."

## Exercise 5.10: Adding a memory monitor

Before you begin, you must complete "Exercise 5.9: Monitoring variables in PAYROLLG" on page 98.

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

To add a memory monitor:
1. In the Source view, double-click the variable ERR in line 33.
2. Right-click and select **Monitor Memory > Memory** on the pop-up menu.
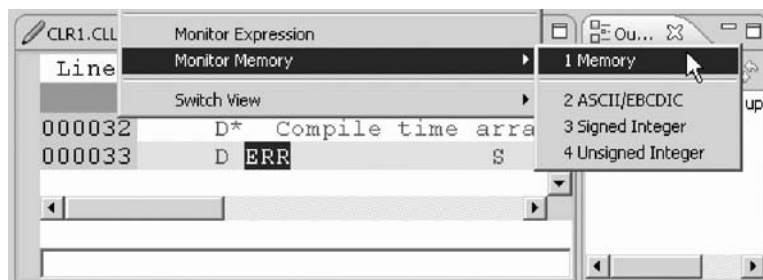


A new page is added to the Memory view. The tab shows the name of the variable.
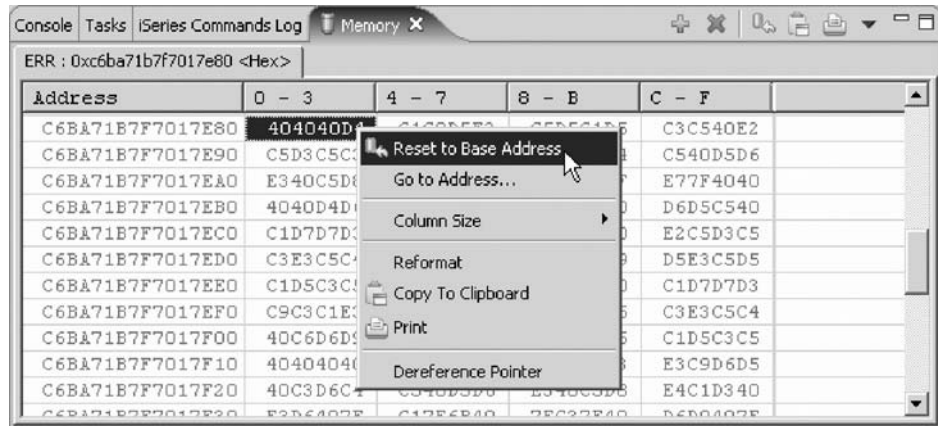
3. Use the scroll bar on the right of the Memory view to scroll down.

    You can see the current content of the memory.

4. Right-click in the view area.

5. Click **Reset to Base Address** on the pop-up menu to return to the starting address.

6. Click the X on the Memory title bar to remove the memory monitor.



You have added a memory monitor for the variable ERR and now you are ready to begin "Exercise 5.11: Setting Watch breakpoints."

## Exercise 5.11: Setting Watch breakpoints

Before you begin, you must complete "Exercise 5.10: Adding a memory monitor" on page 99.

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.
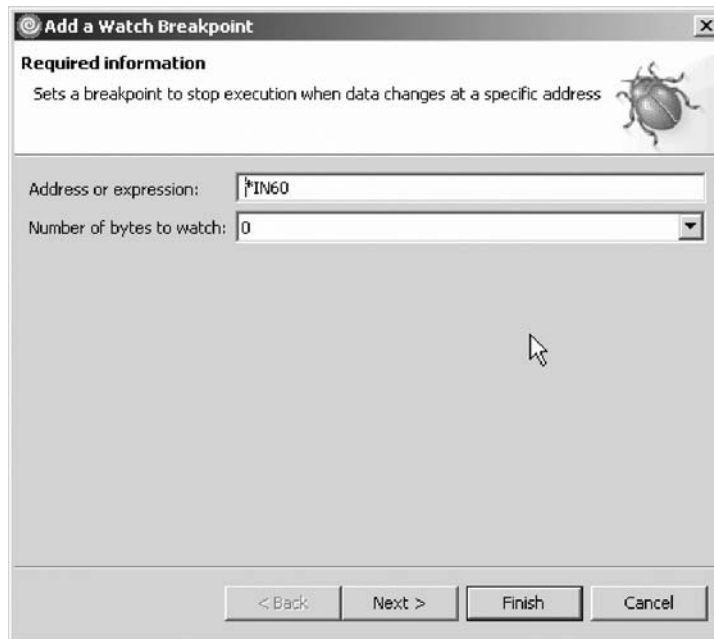
To set a Watch breakpoint:

1. Go to the **Line number** field at the bottom of the source area. In this field enter 116 to go to that line.
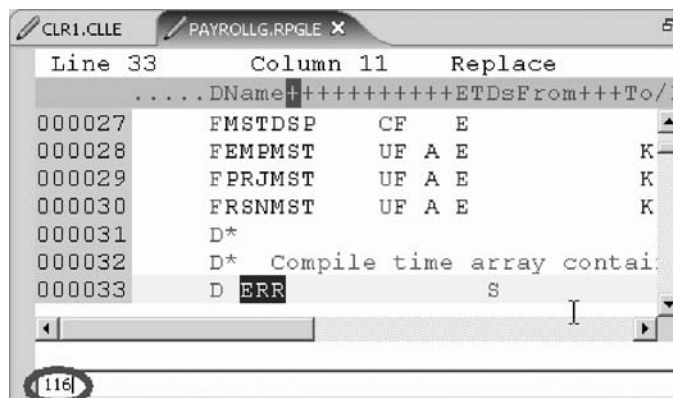


2. Double-click variable *IN60 to highlight it.

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.

By default the **Number of bytes to watch** field is set to zero, which means the variable will be watched in its defined length.
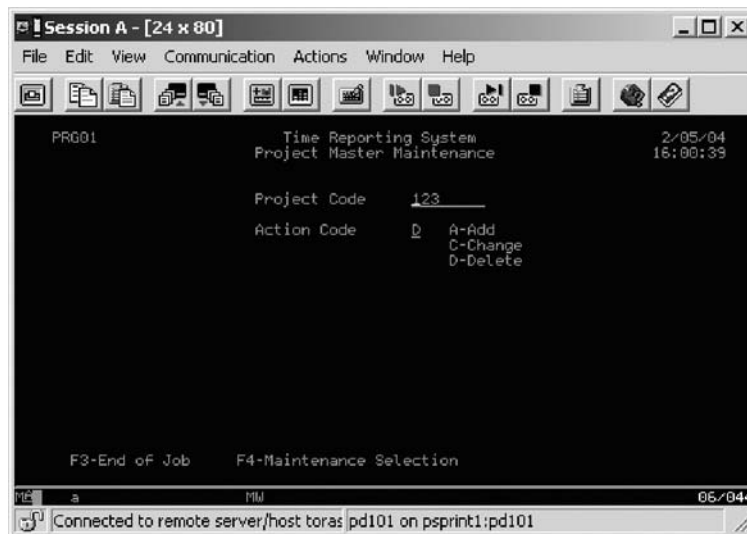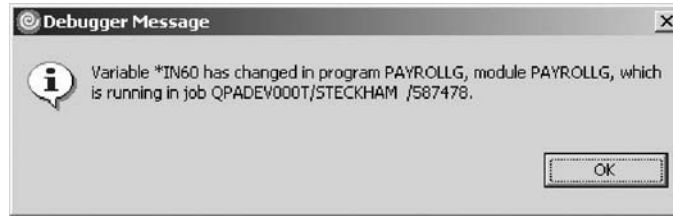


4. Click **Finish**.

   The Watch breakpoint is now set.

5. Click the **Resume** button 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 has changed.

8. Click **OK**. The program stops at line **465**. This line is located immediately after the statement which caused the variable *IN60 to change.

You have added a Watch breakpoint for the variable *IN60 and run the program to see the notification that the variable has changed and you are ready to begin "Exercise 5.12: Closing the debug session."

## Exercise 5.12: Closing the debug session

Before you begin, you must complete "Exercise 5.11: Setting Watch breakpoints" on page 100.

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.
4. A message **Program terminated** appears:



   Click **OK**.
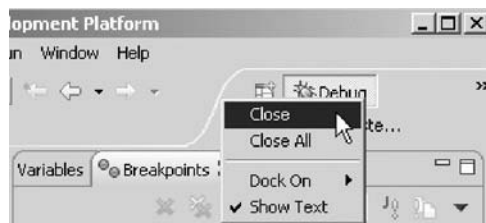


5. Right-click the **Debug icon** on the top right of the workbench.
6. Click **Close** on the pop-up menu to close the Debug perspective.

Now you are ready to review your knowledge of this module by taking the quiz. You can also apply what you have learned in this module by completing the practice tasks detailed in More practice.

**Quiz**
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. To set a breakpoint you:

　　　a. Right-click on the line and click Add Breakpoint on the pop-up menu.

　　　b. Double-click in the prefix area.

　　　c. Right-click in the Breakpoints view and click Add Breakpoint on the pop-up menu.

　　　d. Right-click in the prefix area and click Add Breakpoint on the pop-up menu.

　　　e. All of the above

　4. You can change variables and indicators in the:

　　　a. Remote Systems view

　　　b. Debug view

　　　c. Monitors view

　　　d. Source view

　　　e. All of the above

　5. The debugger allows you to:

　　　a. Step over a program call

　　　b. Step into a called program

　　　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/Enable

　　　d. Edit

　　　e. 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. The storage monitor supports these display formats:

　　　a. Hexadecimal and text

　　　b. Character

　　　c. Text only

　　　d. Decimal

　　　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. Line

　　　c. Conditional

**More practice**

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 product online help to assist you in these tasks.

**Module recap**

You have completed Chapter 5, "Module 5. Debugging a program," on page 79. You have learned how to:

- Invoke the debugger from the Launch Configurations window
- Add a breakpoint
- Add a conditional breakpoint
- Edit a breakpoint
- Monitor a variable through the Monitors view
- Step into your payroll program
- Show a Listing view
- List the call stack entries in the Debug view
- View all breakpoints
- Remove a breakpoint
- Monitor storage
- Set a Watch breakpoint
- Close the debugger

You have mastered debugging your program and now you learn how to define filters and perform actions. Continue with Chapter 6, "Module 6: Exploring Remote System Explorer," on page 105.

# Chapter 6. Module 6: Exploring Remote System Explorer

This module teaches you how to use 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. In short, you'll see how Remote System Explorer can organize and integrate your work and make that work easier.

In this module, you will:
- Know the features of Remote System Explorer
- Create a filter to show specific iSeries libraries
- Change the filter to add more iSeries libraries
- Create a filter to show all the source files in a library
- Access members to edit from your filter
- Create a user action that copies a source file with data to a new source file in the same library
- Specify user action parameters
- Specify a restriction on a user action
- Try the user action
- Run an OS/400 command from the iSeries Table view

**Exercises**
- "Exercise 6.1: More about the Remote System Explorer"
- "Exercise 6.2: Creating a library filter" on page 106
- "Exercise 6.3: Creating an object filter" on page 109
- "Exercise 6.4: Creating a user action" on page 113
- "Exercise 6.5: Running commands from the Remote System Explorer" on page 117

**Time required**

This module will take approximately **15 minutes** to complete.

## Exercise 6.1: More about the Remote System Explorer

Most of the functions of CODE Project Organizer have been replaced by WebSphere Studio functions 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 over time eventually 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.

You have read the list of Remote System Explorer capabilities and you are ready to begin "Exercise 6.2: Creating a library filter."

## Exercise 6.2: Creating a library filter

Before you begin, you must complete "Exercise 6.1: More about the Remote System Explorer" on page 105.

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

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.

Filters allow you to easily organize elements within your system. You use the filter function to list iSeries native file system objects (such as libraries, objects, or members).
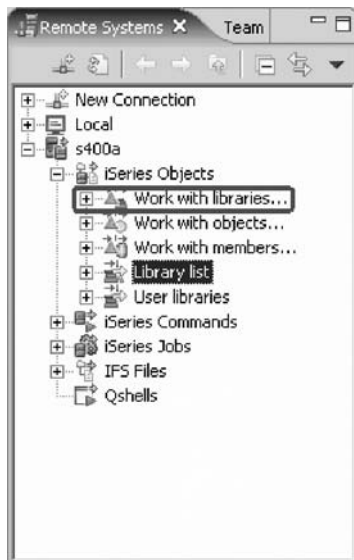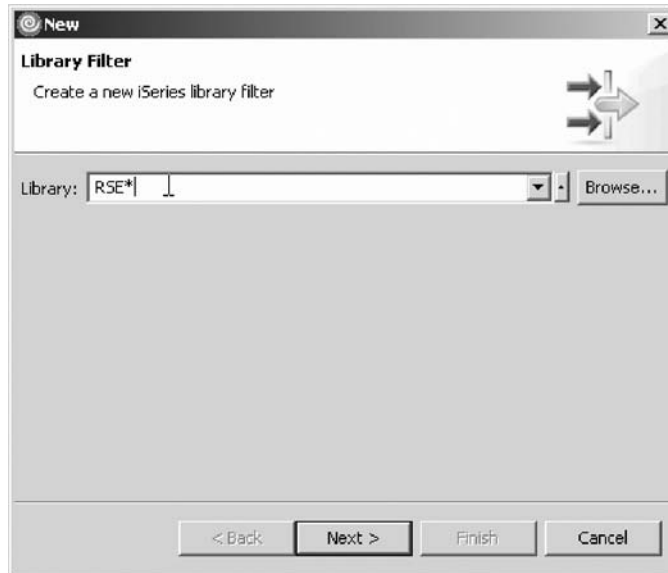


To create a library filter:

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 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 **RSExxxxx** 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 RSE.

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

   The Name the new filter page opens.
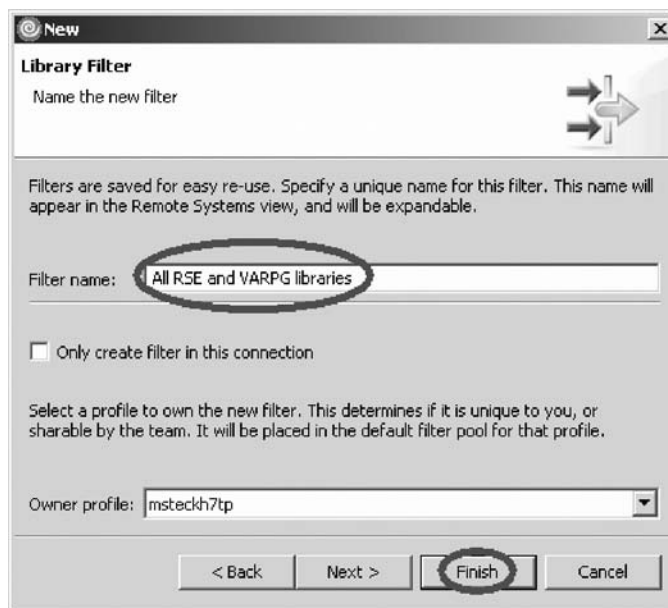


> **Note:** You can choose between creating the filter for all connections or for this specific one only.

6. In the **Filter name** field, type All RSE and VARPG libraries.

You give your filters a name because the Remote System Explorer saves them for future use, unlike PDM, which does not save filters.
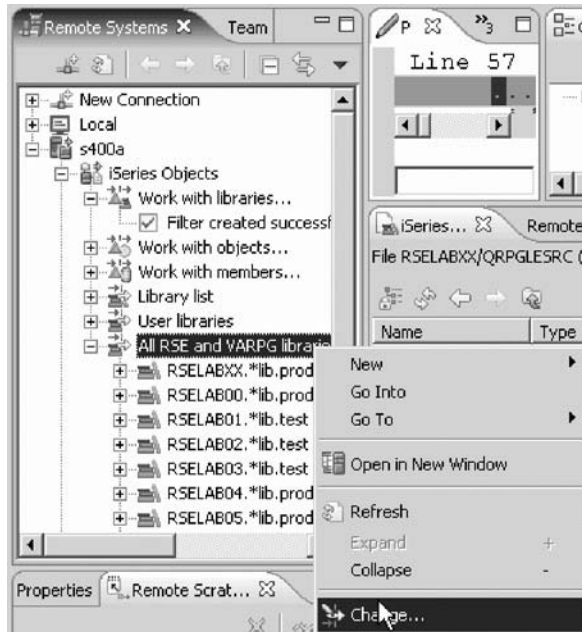
7. Click **Finish**.

Back in the Remote Systems view under iSeries Objects you will see the new filter expanded, listing all RSE* libraries.
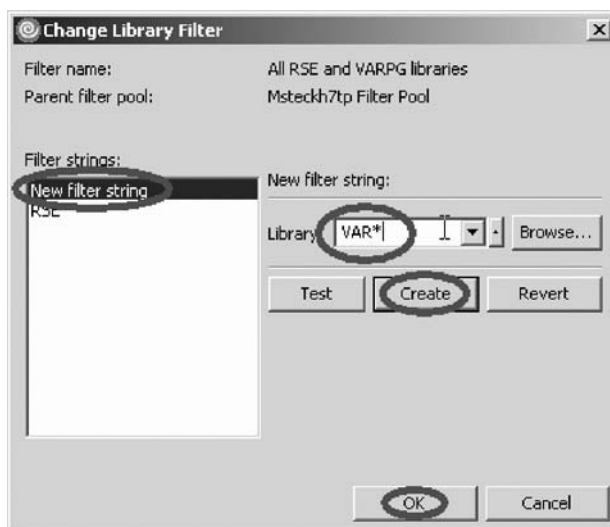
Now you need to add the VARPG libraries.

**Changing the library filter**

To change the library filter:

1. Right-click the filter **All RSE and VARPG libraries** and click **Change**.
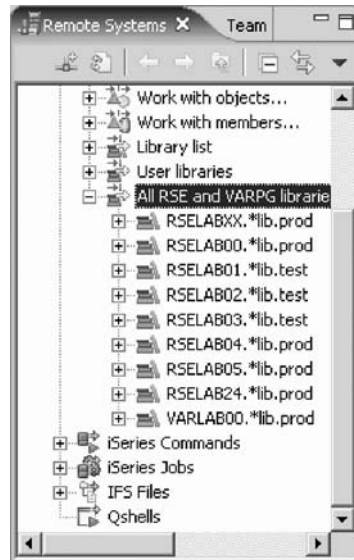


The **Change Library Filter** window opens.



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.

You have created a filter to show a specific iSeries library and changed that filter to add more iSeries libraries and you are ready to begin "Exercise 6.3: Creating an object filter."

## Exercise 6.3: Creating an object filter

Before you begin, you must complete "Exercise 6.2: Creating a library filter" on page 106.

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.
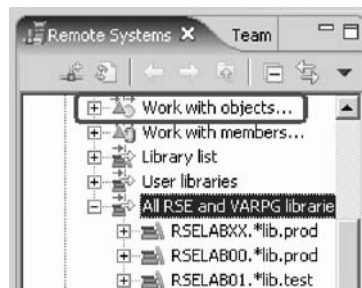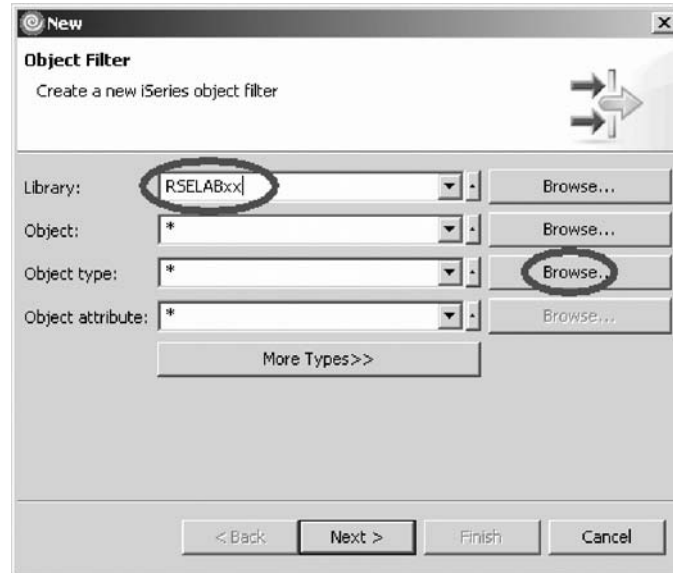


2. Expand **Work with objects**. You can also right-click **iSeries Objects** and click **New > Object filter** on the pop-up menu.

   **Note:** Expanding Work with objects corresponds to the WRKOBJPDM command.

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**.
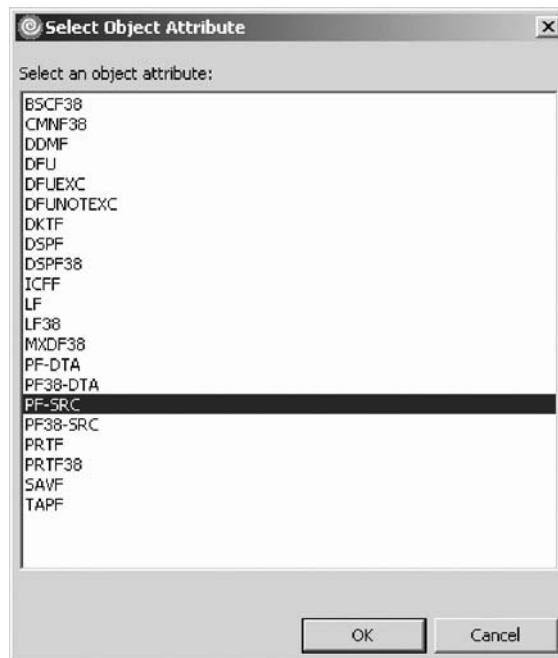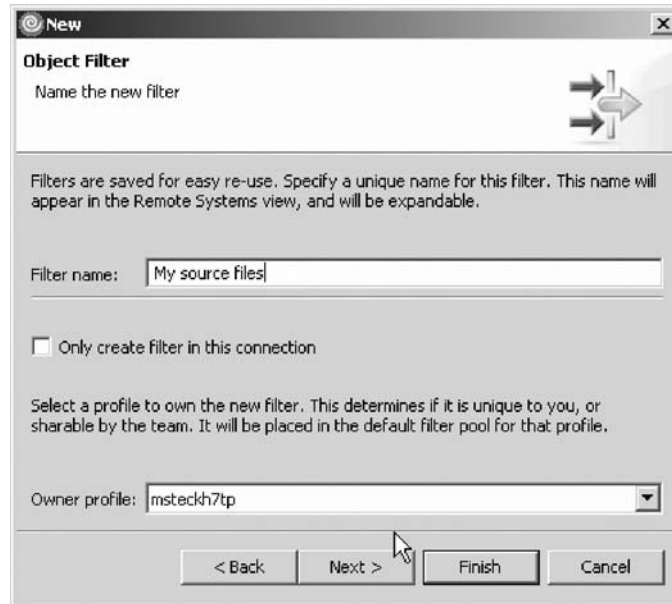
   The Create a new iSeries object filter page displays with the object type updated.
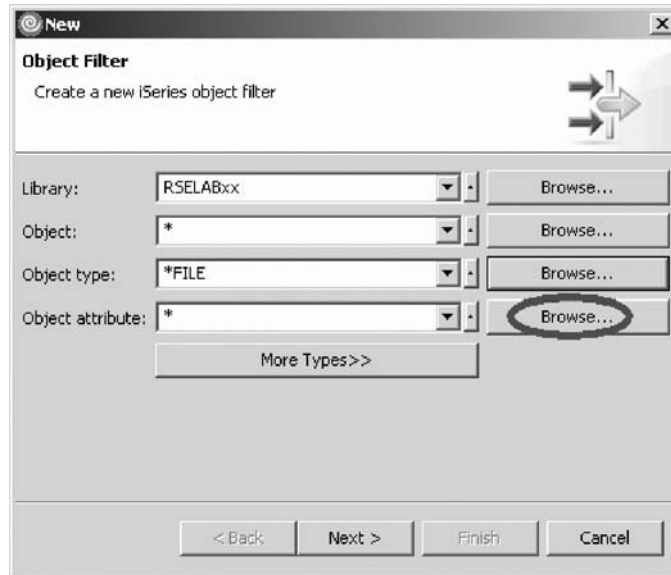
7. Click **Browse** beside the **Object attribute** field.

   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**.

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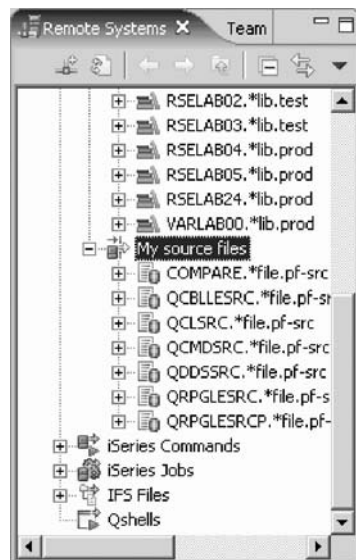


> **Note:** If you end up with too many filters, you can create filter pools. They allow you to group filters.

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 a subset of libraries, objects, or members.

Let's assume you want to edit the member PAYROLL in the source file QRPGLESRC using this object filter.

**Editing a member from your own object filter**

To edit a member:
1. Expand QRPGLESRC.
2. Right-click member PAYROLL.
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.

You have created a filter to show all the source files in your library and accessed members to edit from your filter and you are ready to begin "Exercise 6.4: Creating a user action."

## Exercise 6.4: Creating a user action

Before you begin, you must complete "Exercise 6.3: Creating an object filter" on page 109.

In PDM you can create user actions in addition to using the pre-supplied system actions. In Remote System Explorer you can do the same. You define user actions through the Work With User Actions window. User actions can be defined for iSeries libraries, objects, members and jobs as well as 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.
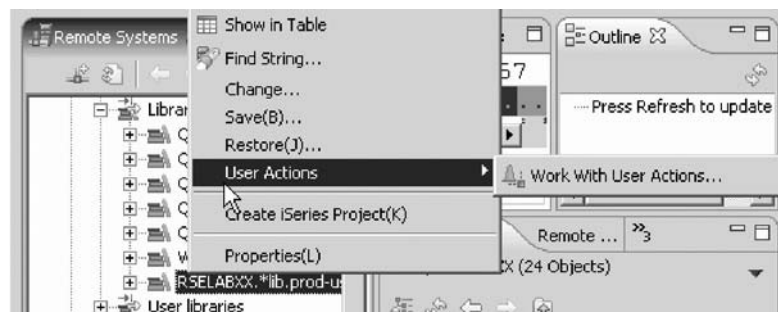


2. Expand the Library list filter if not already expanded.
3. Right-click RSELABxx.
4. Click **User Actions > Work with User Actions** on the pop-up menu.

    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. Select **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.
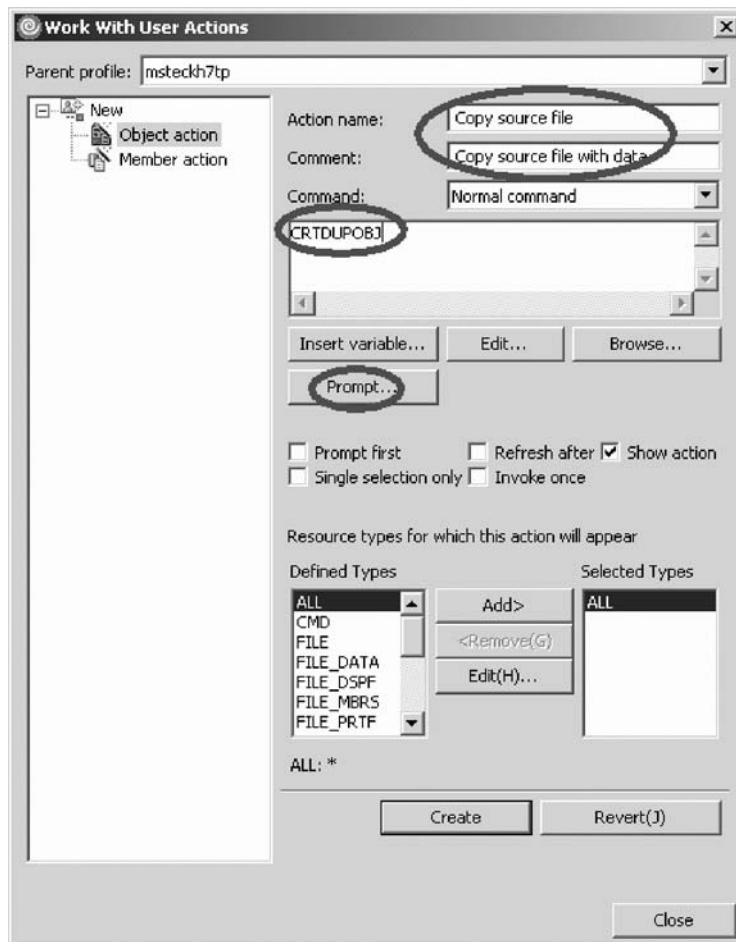


7. In the **Action** field, type Copy source file for the user action name.

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 prompter for this command.
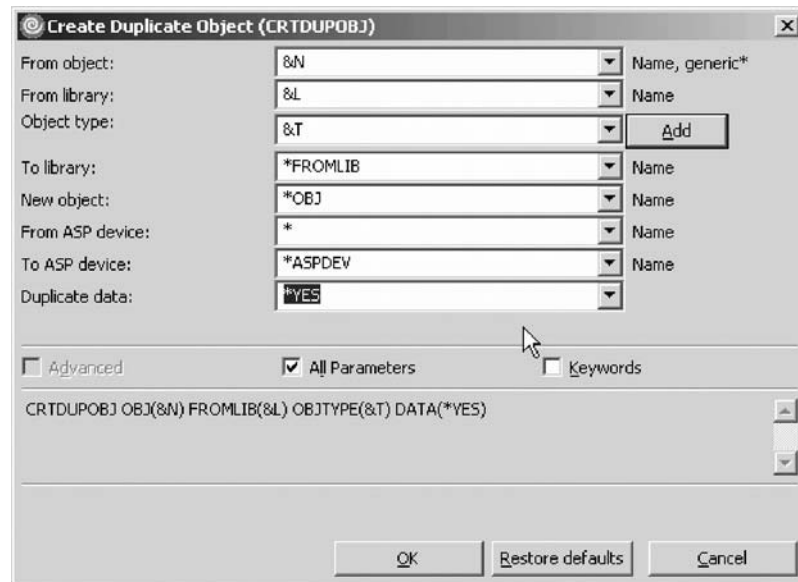
This is the command you will be running:

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

**Specifying user action parameters**

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 the **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.
8. Select the **Refresh after** check box, so that the Remote Systems view gets refreshed after the action has been run.
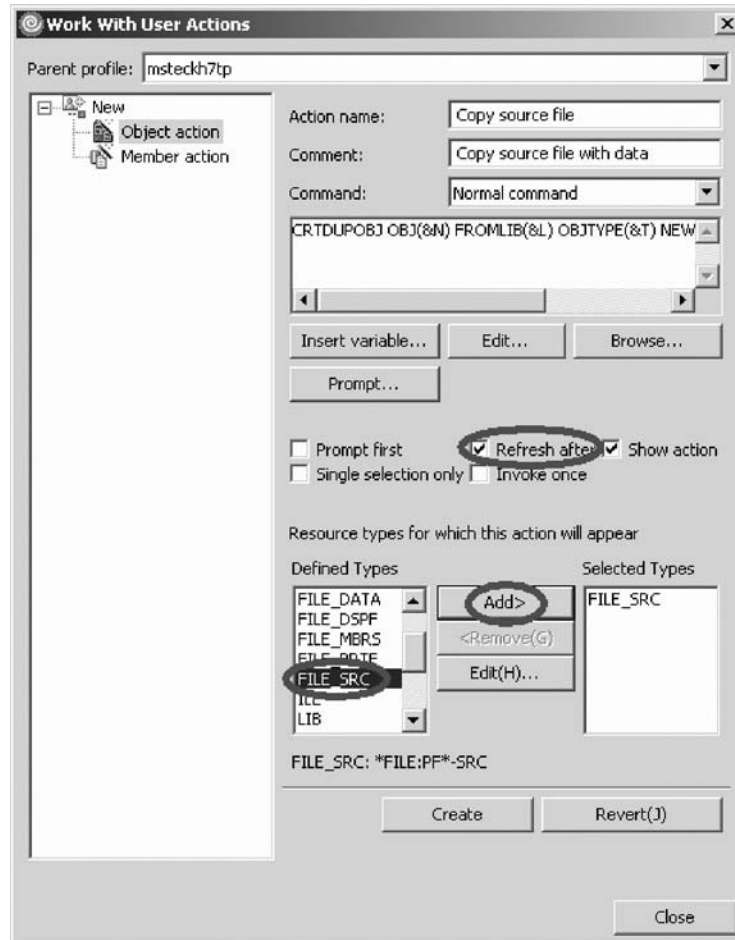
**Note:** Clicking the **Insert variable** push button displays a list of valid replacement variables with the explanation of what they do.

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.

**Specifying a restriction on a user action**

To specify this restriction:
1. Under the **Defined Types** list box, click **FILE_SRC**.
2. Click **Add** beside the **Defined Types** list box.

   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. For any other object type it will not appear. Back in the workbench and the Remote System view, give it a try.

   **Note:** Remember to close the payroll source member if you opened it earlier.

**Trying the user action**

To try your user action:

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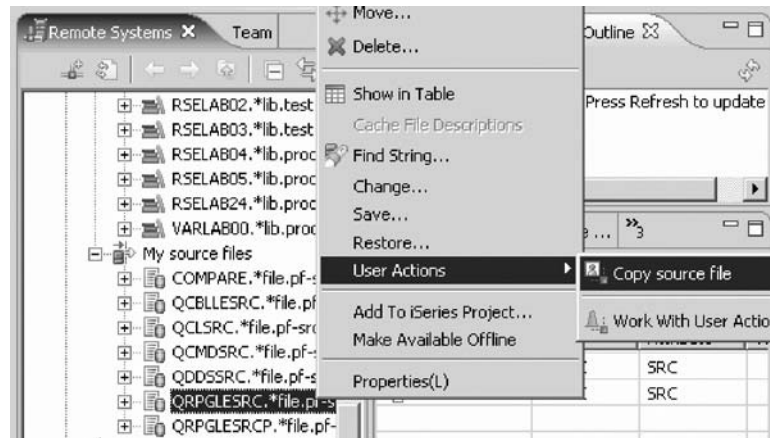


2. Expand the filter **My source files**, if it is not already expanded.
3. Right-click the QRPGLESRC file.
4. Click **User Actions > Copy source file** on the pop-up menu.

   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 RSE communications server job by looking at the iSeries Commands log in the right hand side bottom pane of the workbench.
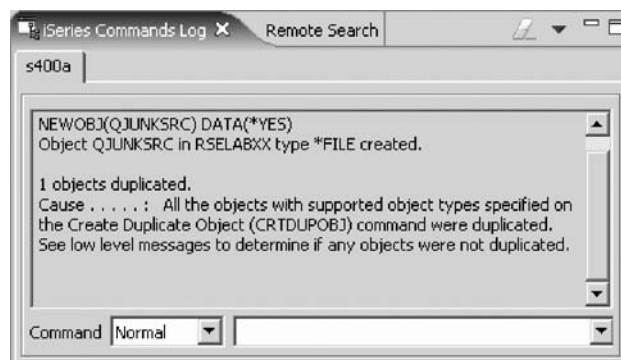


5. To delete the source file QJUNKSRC that you just created, right-click **QJUNKSRC**.
6. Click **Delete** on the pop-up menu.

   The Delete Confirmation dialog opens.
7. Click **Delete**.

You have created a user action that copies a source file with data to a new source file, specified user action parameters, specified restrictions on the user action and tried the user action and you are ready to begin "Exercise 6.2: Creating a library filter" on page 106.

## Exercise 6.5: Running commands from the Remote System Explorer

Before you begin, you must complete "Exercise 6.4: Creating a user action" on page 113.

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 commands 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.

   As you have seen already you can now run commands on the iSeries server that the table is connected to. You can enter commands in the Commands field beneath the iSeries Table view, and view messages in the Messages field. You can enter a command and click either Prompt to specify parameters and then Run, or just click Run. When you run a command, the Messages field is populated with the messages from the command. When you select a message, the Details button is enabled. When you click this button, the message and its help is displayed.
4. Type an iSeries command, for example ?ADDLIBLE.
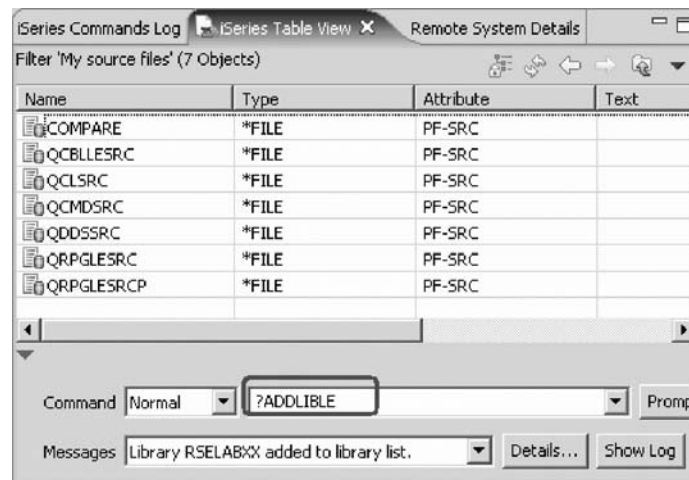


   The question mark is there to display a prompt screen.

   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.
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 (down arrow beside the **Messages** field.
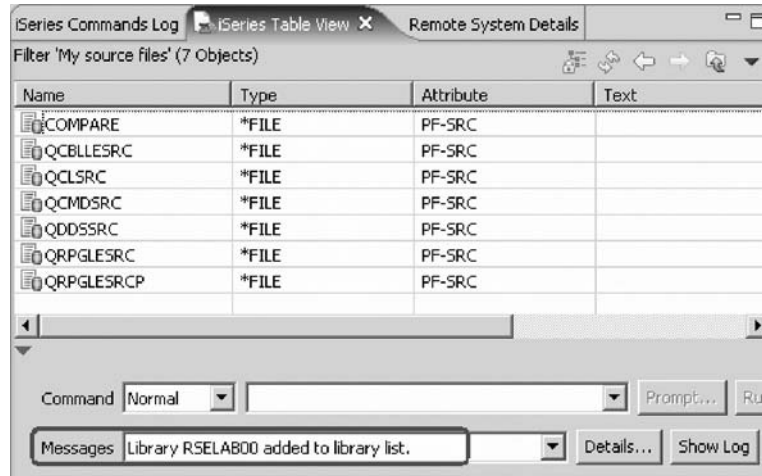


You could also use the iSeries Commands subsystem in the Remote Systems view underneath the iSeries Objects subsystem and run predefined commands or define your own commands.

You have run an OS/400 command from the iSeries Table view.

Now you are ready to review your knowledge of this module by taking the quiz. You can also apply what you have learned in this module by completing the practice tasks detailed in More practice.

**Quiz**
1. The Remote System Explorer is replacing what ADTS tool:
    a. Page Designer
    b. CODE Designer
    c. Screen Design Aid
    d. Program Development Manager
2. _____ filters list a set of libraries from your iSeries system in the Remote Systems view.
    a. Object
    b. Member
    c. Library
    d. Job
3. Filters allow you to easily organize elements within your system. You use the filter function to list iSeries native file system objects (such as libraries, objects, or members). (T, F)
4. Expanding Work with Libraries corresponds to the _____ command.
    a. WRKOBJPDM
    b. WRKLIBPDM
    c. WRKMEMPDM
    d. All of the above

5. You can create filters for all connections or for a specific connection. (T, F)
6. You give your filters a specific name because the Remote System Explorer saves them for future use, unlike PDM, which does not save filters. (T, F)
7. _____ filters list a set of objects from your iSeries host in the Remote Systems view.
   a. Job
   b. IFS
   c. Object
   d. Member
   e. Library
8. If you end up with too many filters, you can create filter pools. (T F)
9. Filters can be specified for non iSeries servers and your local system. (T, F)
10. User actions can be defined for:
    a. iSeries libraries
    b. iSeries objects
    c. iSeries members
    d. iSeries jobs
    e. files and folders in any remote UNIX, Windows, Linux, Local or IFS system
    f. All of the above
11. You can specify a restriction on a user action. (T, F)
12. You can run OS/400 commands from the:
    a. iSeries Table view
    b. iSeries Commands log
    c. Tasks view
    d. iSeries Error List
    e. a and b
    f. All of the above

**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**

You have completed Chapter 6, "Module 6: Exploring Remote System Explorer," on page 105. You have learned how to:
• Know the features of Remote System Explorer
• Create a filter to show specific iSeries libraries
• Change the filter to add more iSeries libraries
• Create a filter to show all the source files in a library
• Access members to edit from your filter
• Create a user action that copies a source file with data to a new source file in the same library
• Specify user action parameters

- Specify a restriction on a user action
- Try the user action
- Run an OS/400 command from the iSeries Table view

Now that you know how to create filters and actions to manage your iSeries objects you can learn how to visually design screens. Continue with Chapter 7, "Module 7. Designing screens," on page 123.

# Chapter 7. Module 7. Designing screens

This module teaches you about 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 this module, you will:
- Open a DDS member for edit with CODE Designer
- Show file-level keywords and record keywords
- View the details of records, record-level keywords and field-level keywords
- View the design of the payroll application main menu
- Create a group from an existing record format
- Create a new group and add a subfile record and a subfile control record
- Add columns to the subfile record
- Add fields to the subfile control record
- Copy existing fields
- Set indicators to handle field errors
- View and update record and field properties
- View keywords and the properties of a keyword
- Insert a keyword
- View help for a keyword
- Check there are no semantic errors in the DDS source
- View help for an error
- Launch the editor in read mode from the error list
- Launch the editor in write mode to fix the error
- Find a keyword in the source
- Save source changes
- Compile your source changes
- Close the Designer

**Exercises**

The exercises within this module must be completed in order:

- "Exercise 7.10: Accessing field properties" on page 139
- "Exercise 7.11: Adding new keywords" on page 141
- "Exercise 7.12: Verifying the source changes" on page 143
- "Exercise 7.13: Switching between designing and editing the screen" on page 145
- "Exercise 7.14: Compiling your source changes and closing the Designer" on page 146

**Time required**

This module will take approximately **30 minutes** to complete.

# Exercise 7.1: Opening a DDS member in the Remote Systems view

Using an editor to create and maintain DDS source for your display, printer and physical 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 the previous LPEX Editor exercises.

To open a DDS file member from the Remote Systems view:
1. Expand the Library list filter if it is not already expanded.
2. Expand the QDDSSRC file in library RSELABxx.
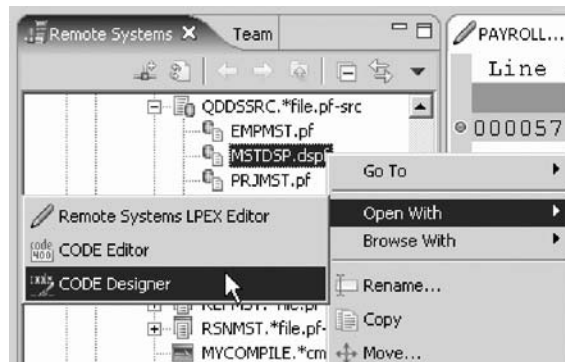


3. Right-click the MSTDSP member.
4. Click **Open with > CODE Designer** on the pop-up menu.

   The member MSTDSP will be downloaded to the workstation and loaded into CODE Designer. CODE Designer is a separate tool not integrated into the workbench.

You have opened the DDS source member MSTDSP in the Remote Systems view using CODE Designer and you are ready to begin "Exercise 7.2: Viewing the DDS tree" on page 125.

## Exercise 7.2: Viewing the DDS tree

Before you begin, you must complete "Exercise 7.1: Opening a DDS member in the Remote Systems view" on page 124.

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, everything will be explained to you.

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 show file-level keywords and the record SELECT in the DDS tree:

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.
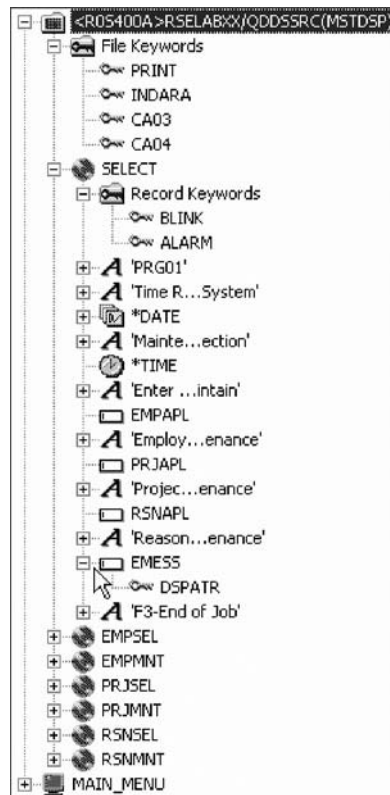


The DDS tree now shows you a summary of the file-level keywords and of the record SELECT.

You have seen the file-level keywords and the record SELECT in the DDS tree and you are ready to begin "Exercise 7.3: Selecting the DDS object."

## Exercise 7.3: Selecting the DDS object

Before you begin, you must complete "Exercise 7.2: Viewing the DDS tree" on page 125.

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, printer or physical 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 clicking View > List from the CODE Designer menu.

In the Details page columns display information about the selected DDS object. You can use this page to display for example, details of all the fields in the record SELECT or keywords and conditions of a field or record.

The Listing page is a listing of the source statements generated by the Program Verifier.

In the bottom right-hand side of the Designer is the Utility notebook. This notebook contains several pages: Selected DDS, Web Settings, Comments and Error List. The Selected DDS page in the notebook shows the actual DDS source for the currently selected item.

**Note:** The Web Settings page allows you to specify attributes that are used by the WebFacing tool.

To work with the DDS record SELECT:

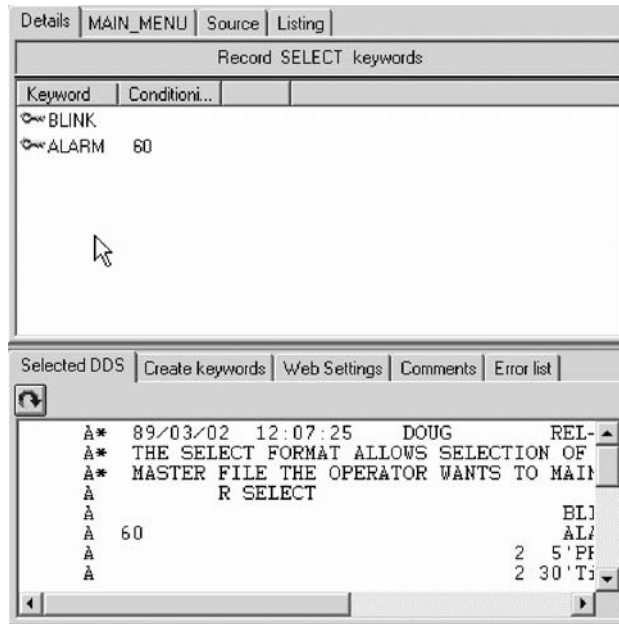1. In the DDS tree click the SELECT record.

   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.
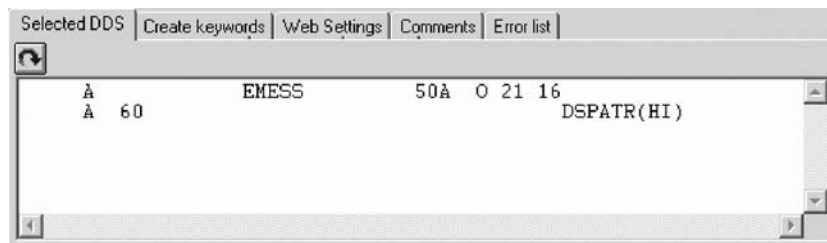


2. In the DDS tree click Record keywords immediately below SELECT.

   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.



   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 independent of the DDS required to make things appear the way they do. You need the Design page.

You have seen the details of the record SELECT, the record-level keywords and the field-level keywords and you are ready to begin "Exercise 7.4: Designing the DDS screen."

## Exercise 7.4: Designing the DDS screen

Before you begin, you must complete "Exercise 7.3: Selecting the DDS object" on page 126.

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. You can 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.

1. Click the **MAIN-MENU** tab in the workbook.



In order to understand where MAIN_MENU came from, you need to understand 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 this has been done for the screen you see in the Design page. A group MAIN_MENU has been created for you and CODE Designer has 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.

You have seen the concept of a group, specifically the MAIN_MENU group and you are ready to begin "Exercise 7.5: Creating groups from existing records."

## Exercise 7.5: Creating groups from existing records

Before you begin, you must complete "Exercise 7.4: Designing the DDS screen" on page 127.

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 the 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.
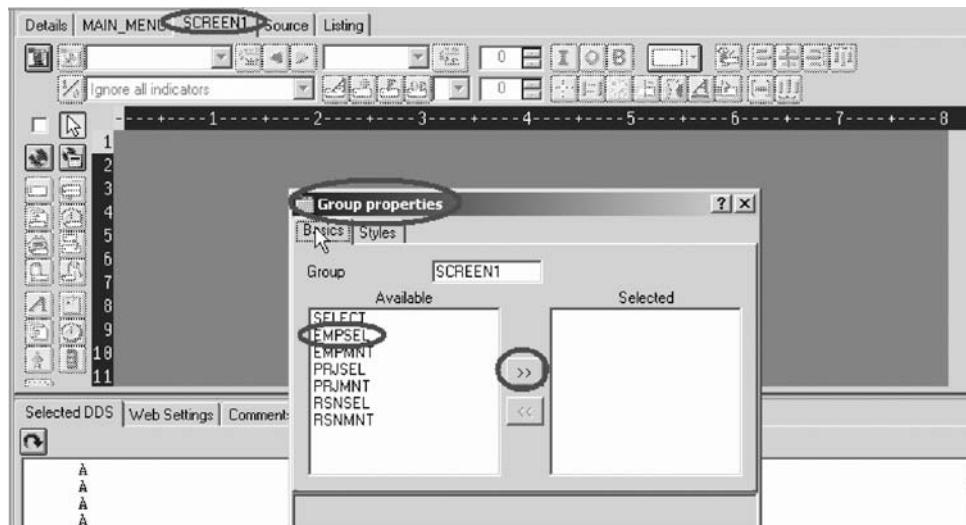


   The Properties notebook lets you view and update the properties of the currently selected DDS object. You can open this notebook from any view, pop-up menu, or menu of the CODE Designer. The Properties notebook is modeless. When you change an object's properties, the selected object changes immediately.

4. In the Group Properties notebook, select the 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 you what the record EMPSEL looks like.



5. In the group field, type EMPL_SELCT over SCREEN1 to rename the group.
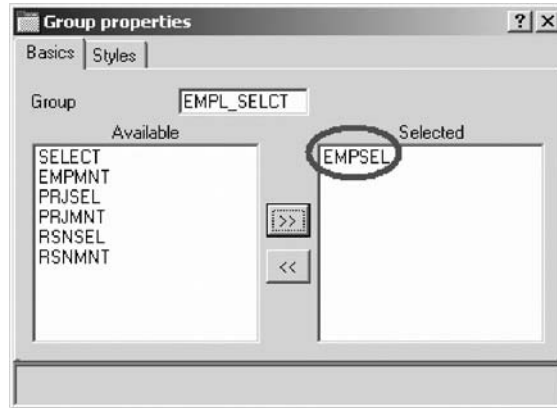
6. Close the Group Properties notebook. Click the **X** in the top right corner of the Group Properties notebook.

   You have finished creating a group. You could now work in the Design page with the record formats contained in this group. Instead 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.

You have inserted a new group using the existing record EMPSEL and you are ready to begin "Exercise 7.6: Creating new screens."

## Exercise 7.6: Creating new screens

Before you begin, you must complete "Exercise 7.5: Creating groups from existing records" on page 128.

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.
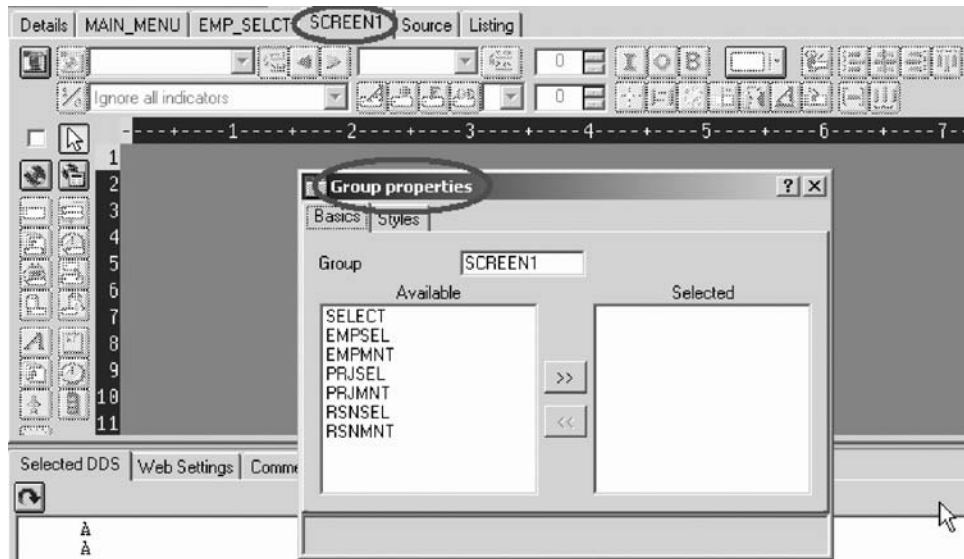
3. Rename the group to EMPL_LIST and close the Group Properties notebook.

4. 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 buttons 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.
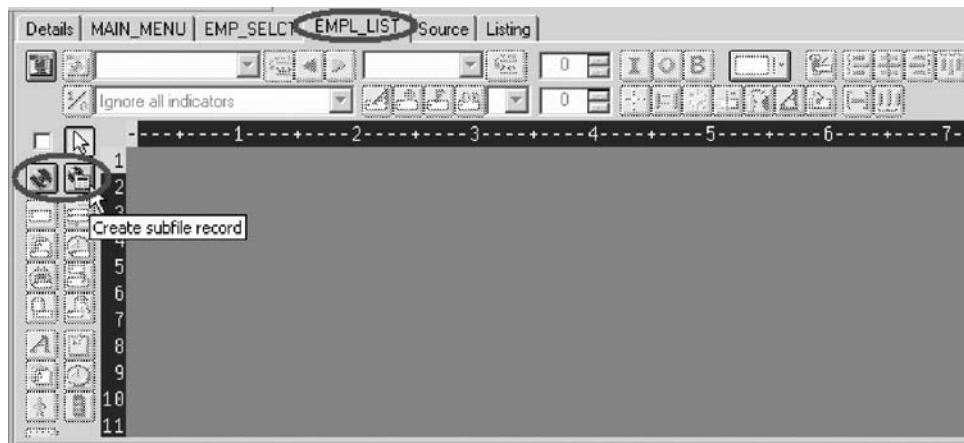


5. Click the **Create subfile** record button and then click in the dark gray area.

   A subfile and a subfile control record pair are created.
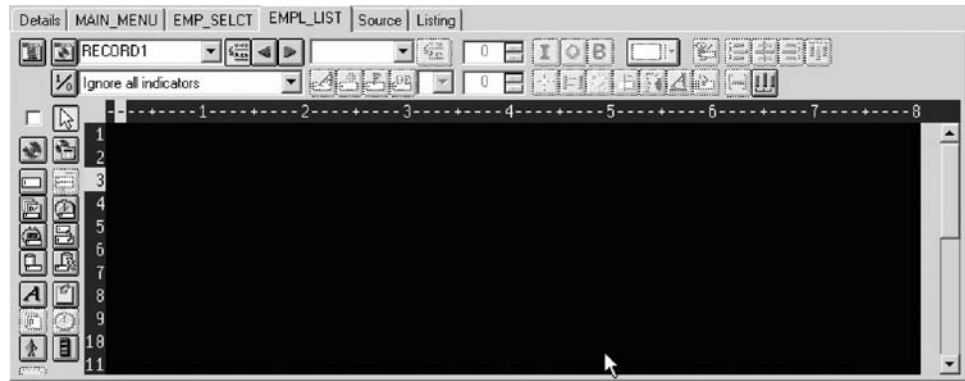
You have created a new record screen and you are ready to begin "Exercise 7.7: Adding fields to the subfile record."

## Exercise 7.7: Adding fields to the subfile record

Before you begin, you must complete "Exercise 7.6: Creating new screens" on page 130.

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:

1. Click the **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.

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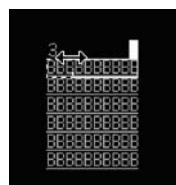
The toolbar at the top of the Design page is a very convenient place to monitor and manipulate the currently selected parts.

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.

6. Click the  button to change the usage of the field to input.



Now you will create an additional column in the subfile.

**Creating an additional output column**

Now you want to create an output column.

To create an additional output column:

1. 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.

2. If you are creating a long field with an exact length, the SDA syntax can be easier. Type:

   ```
   +O(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 +O(30).

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





4. Rename the new field to ENAME using the toolbar. This will show the name of the employee.
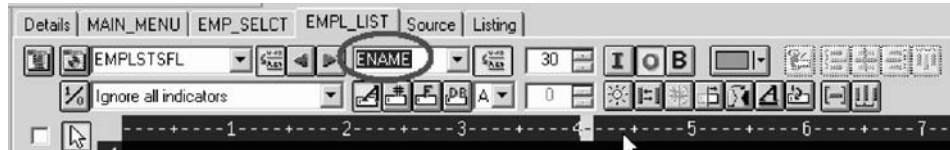
5. Position the cursor to **8, 41**.

6. 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?

7. 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!
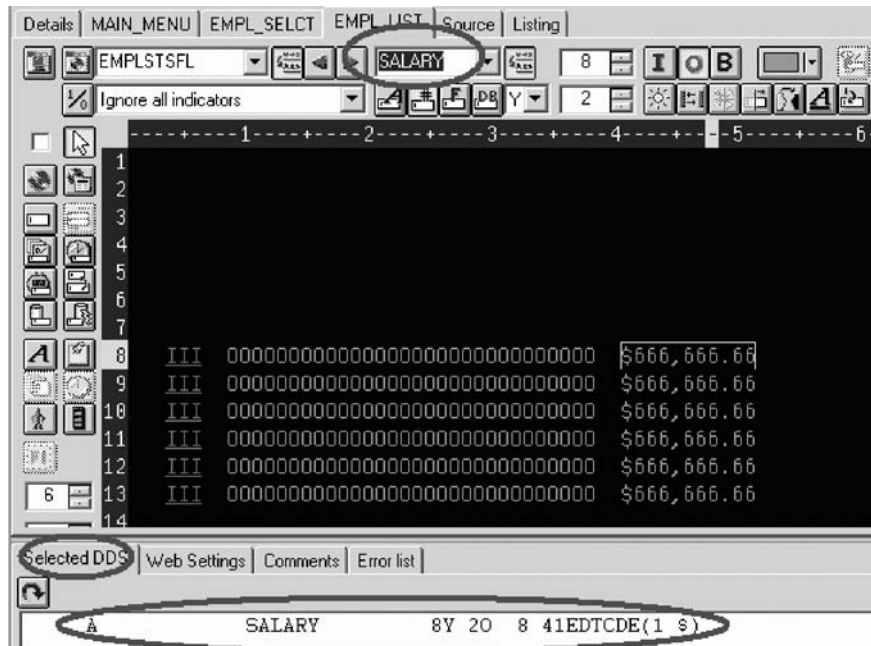


8. Rename the field to SALARY and change its color to yellow, using the toolbar.

9. The subfile seems compacted to the left. It would be better to space it out evenly. Just select a 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.
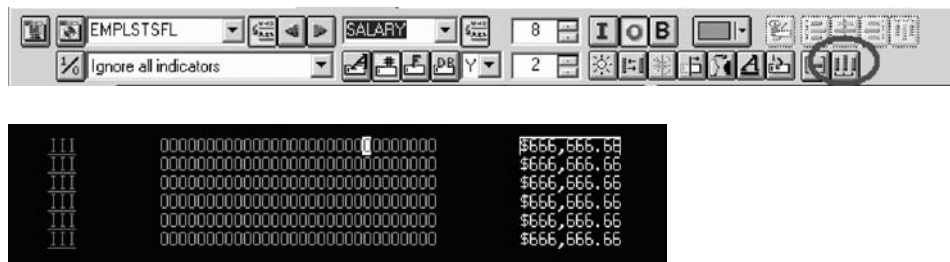
Just below the palette on the left there are three spin buttons. The top one, **Subfile size**, specifies the total number of entries in the list that will be filled in by the application. The second one, **Subfile page size**, is how many entries appear on the screen.

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

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

12. Click in the Design page.
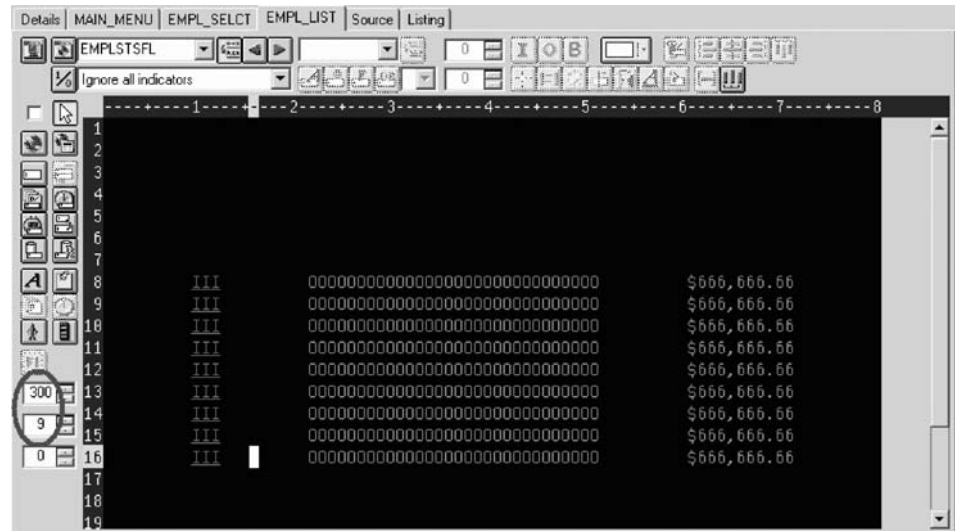


The Design page is updated accordingly.

You have added some columns to the new subfile record screen and now you are ready to begin "Exercise 7.8: Switching between multiple records."

## Exercise 7.8: Switching between multiple records

Before you begin, you must complete "Exercise 7.7: Adding fields to the subfile record" on page 132.

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:



1. 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.
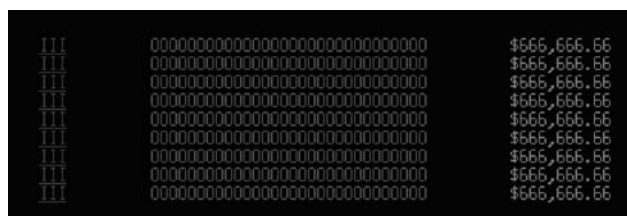
2. Rename the record to `EMPLSTCTL` using the toolbar. Let's provide a 'Position to' entry in the subfile control header.
3. Position the cursor at **4, 9** and type:

   ```
   Position to:
   ```
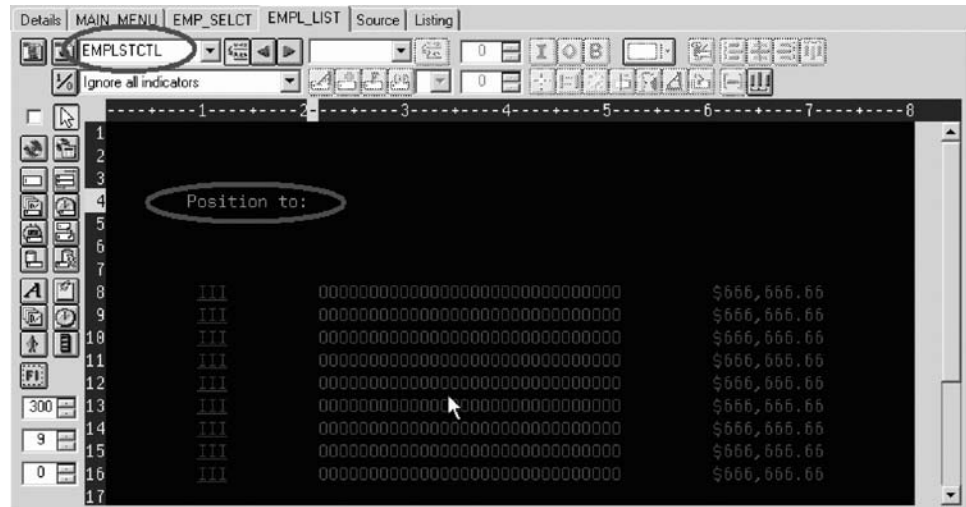


Now you need an employee name field.

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 either the Create database reference field button or the 🔲 Create database reference field(s) by selection 🔳 button. But there is an even simpler way. Use copy and paste!

**Copying the employee name field**

To copy the employee name field:
1. In the DDS tree expand the EMPMNT record.
2. Click the ENAME field and press **Ctrl+C**.
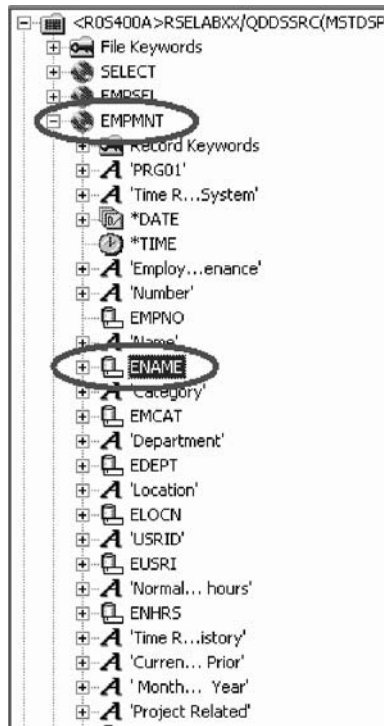
(The pop-up menu or Edit menu shows the Copy menu item as well).

3. Position the cursor to **4, 23** and press **Ctrl+V**. Now that was easy!

4. Click the field and change the name from ENAME to POS_TO.
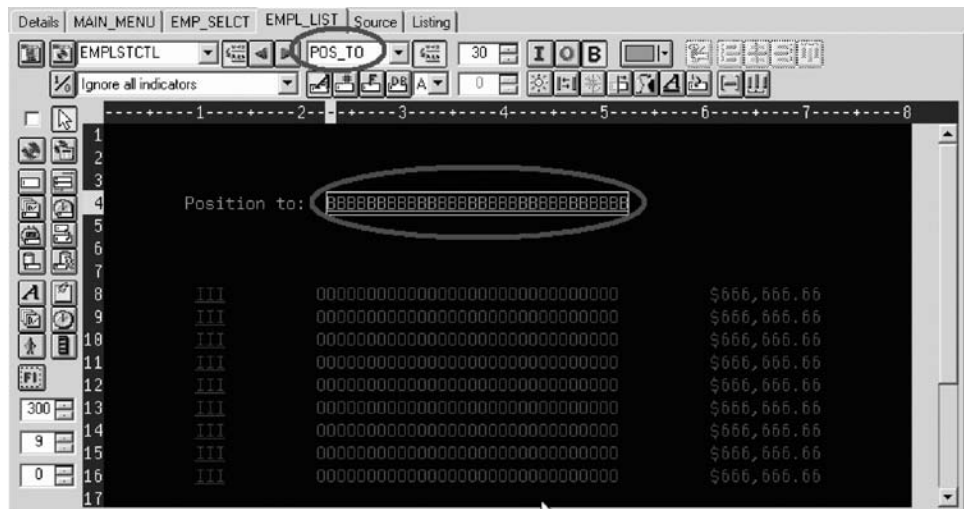


You have fixed up the subfile records to add an employee name field with a position to entry opposite this field and you are ready to begin "Exercise 7.9: Adding field error handling."

## Exercise 7.9: Adding field error handling

Before you begin, you must complete "Exercise 7.8: Switching between multiple records" on page 135.

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 set on indicator 60.

In this case the field should turn red, reverse image and position the cursor to it. Now wouldn't it be better if you had something easier to remember than some arbitrary number from 1 to 99.

To set indicators:

1. Click the Change named indicator sets ![icon] button on the Design page toolbar (or press **F7**.)

   The Named indicator sets window opens.

2. In the **Settings name** field, type: Not Found.

3. Click **Create**.
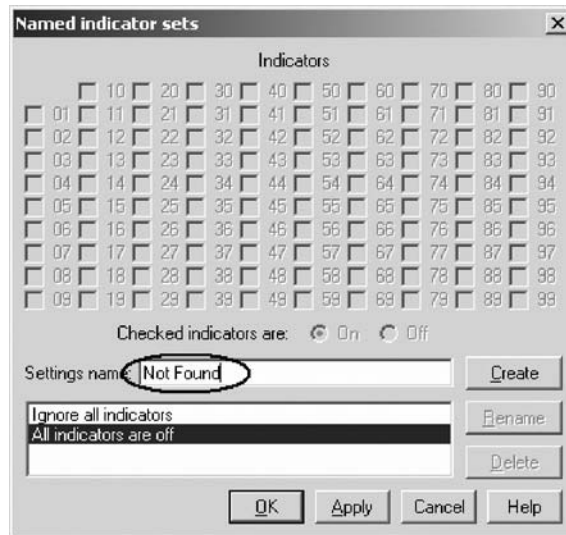


4. Select the check box next to **60** and click **OK**.

   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.



5. Now select the **POS_TO** field.

6. 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:



7. Examine the DDS generated in the Selected DDS page.
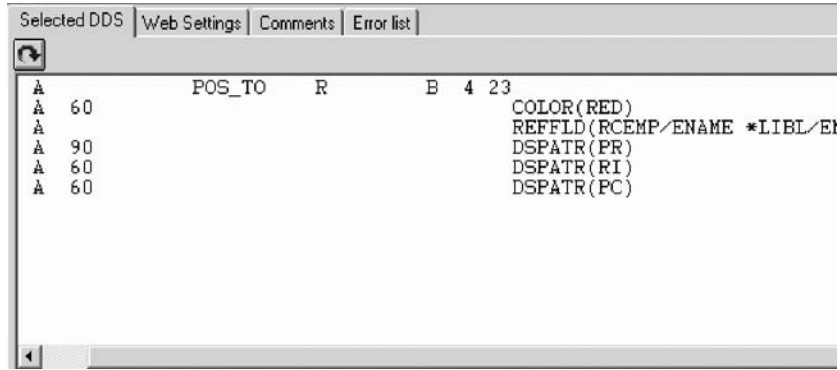
Notice that all the new keywords were created with a condition of 60. (The DSPATR(PR) was pasted with the field originally).

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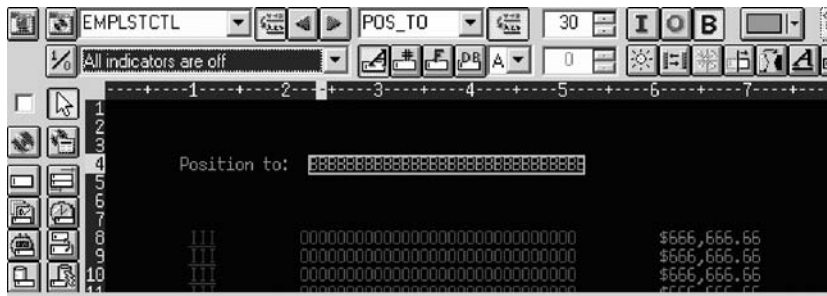


Wow! This really works!

9. From the **Select named indicator sets** list, select **Not found**.

   The field appears red and reverse imaged.

You have added error handling to the position to employee name field and you are ready to begin "Exercise 7.10: Accessing field properties."

## Exercise 7.10: Accessing field properties

Before you begin, you must complete "Exercise 7.9: Adding field error handling" on page 137.

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 the 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:

1. 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.

2. 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.
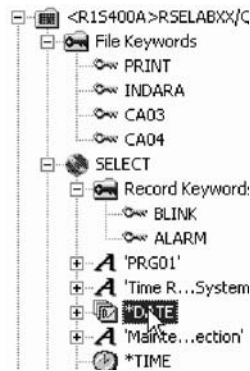


3. Change the year from 2 to 4 digits. Select the **Length of year** check box.
4. Select **4** digits from the list.
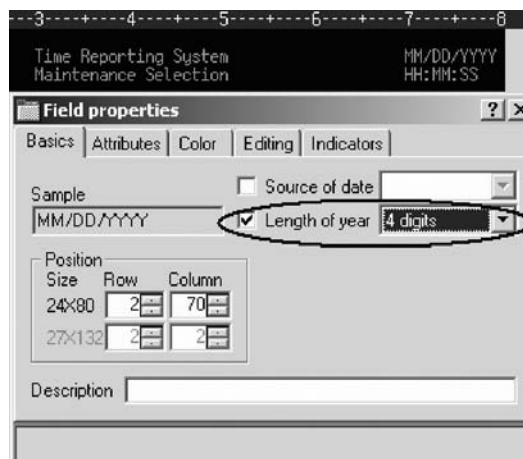


   Notice how the sample is updated on the Properties notebook.

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 of this field.
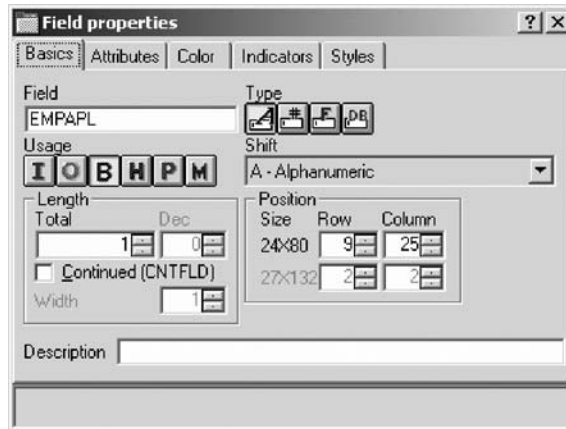


You have seen the record properties for the record SELECT and changed the length of year to 4 digits and you are ready to begin "Exercise 7.11: Adding new keywords."

## Exercise 7.11: Adding new keywords

Before you begin, you must complete "Exercise 7.10: Accessing field properties" on page 139.

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 the EMPAPL field in the DDS tree.
2. Press **F5** or right-click and click **Insert keywords** on the pop-up menu.
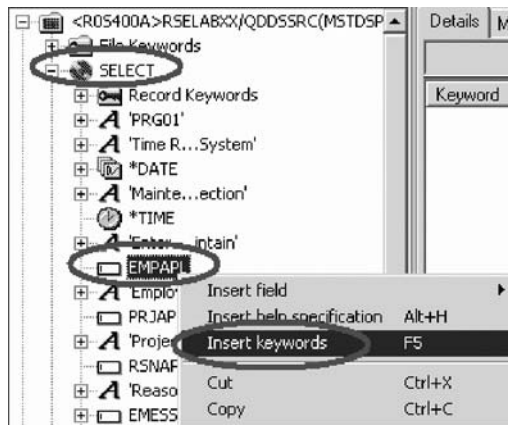


You see the Details page for the EMPAPL field and the **Create keywords** tab is added to the Utility notebook. This page shows you the subset of keywords that are allowed for the selected file, record or field and it takes into account the field's type, usage, shift and what record it is in. It is very powerful to know exactly what your options are. This information cannot be quickly ascertained from the Reference manual.
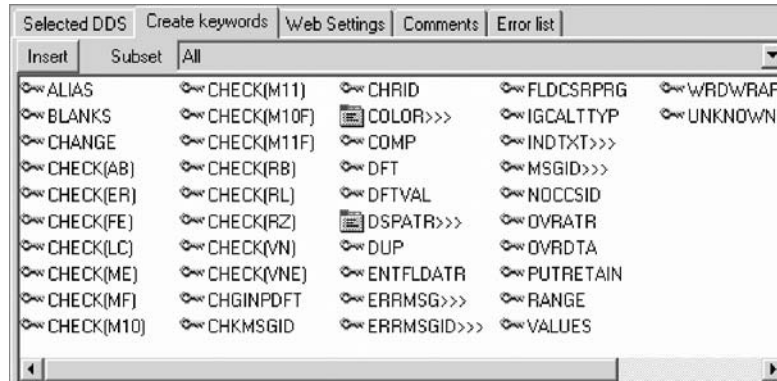
3. With the EMPAPL Properties notebook at the **Basics** page, click the numeric field ![icon] button to change the field to numeric type.
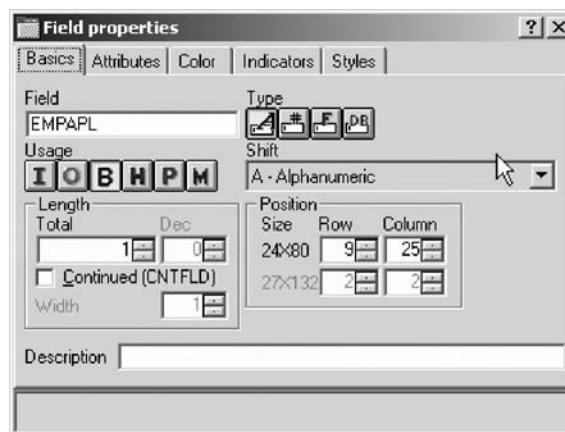


Notice that the list of keywords in the Create keywords page has changed.

4. Click the ![icon] button to change the field back to alphanumeric.

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

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 see help relevant to what you are currently trying to do. From the Help menu you have quick access to the DDS Language Reference as well as several other useful sources of information.
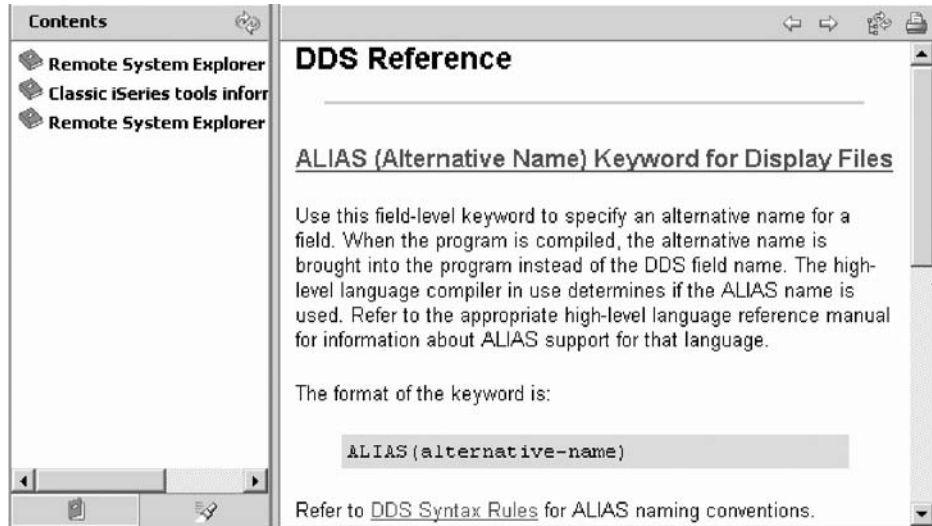
6. Minimize the Help window.

7. In the Create keywords page, 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.

You have seen the help on the ALIAS keyword, added INDTXT keywords and you are ready to begin "Exercise 7.12: Verifying the source changes."

## Exercise 7.12: Verifying the source changes

Before you begin, you must complete "Exercise 7.11: Adding new keywords" on page 141.

You have just added a new record and some new fields to your 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 exercise 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.

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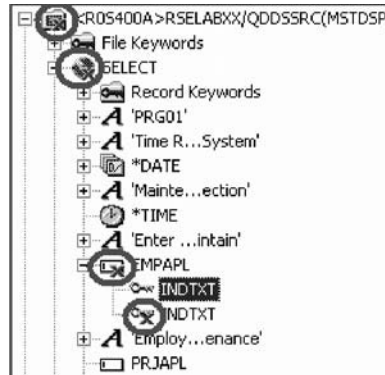


The file icon has a red X, as does the SELECT record, the EMPAPL field and finally the second INDTXT keyword.

3.  Click the **MAIN_MENU** tab in the workbook.

    The EMPAPL field is highlighted in red.

4.  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.

5.  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.

6.  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.
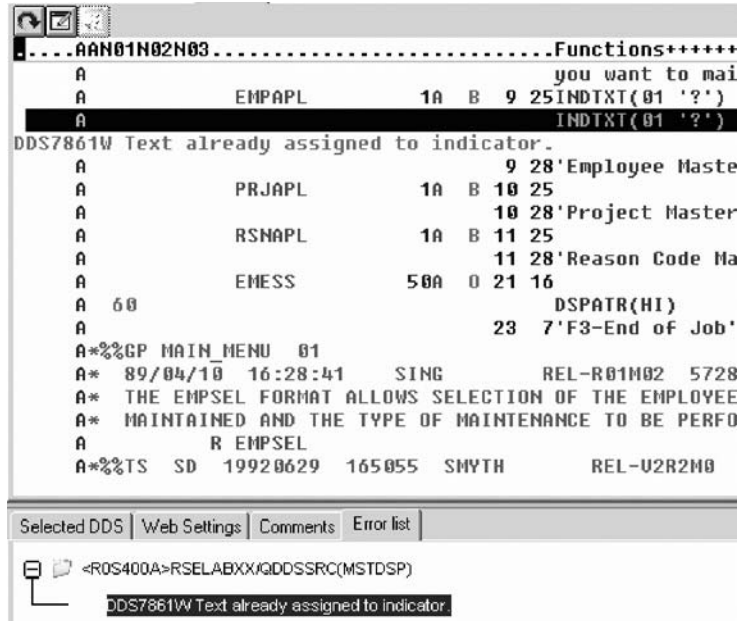
You have verified your DDS source and have identified an error in the source and you are ready to begin "Exercise 7.13: Switching between designing and editing the screen."

## Exercise 7.13: Switching between designing and editing the screen

Before you begin, you must complete "Exercise 7.12: Verifying the source changes" on page 143.

To switch between the Design mode and the Edit mode:

1. Click the Edit DDS source  button or click **File > Edit DDS source** from the Editor menu.

   You now have access to the full power of the editor.
2. Explore the **Edit** and **View** menu items.
3. Press **Ctrl-F** to open the Find/Replace window.
4. In the **Find** field, type INDTXT and click **Find**.
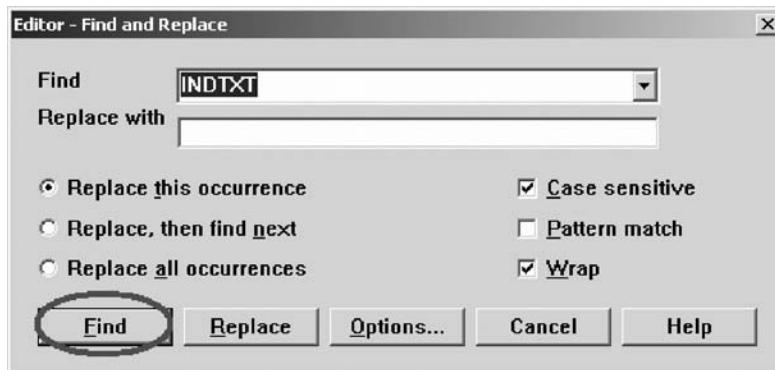


5. Press **Ctrl-N** to find the next occurrence.
6. Delete the second INDTXT line. Type D in the number column and press **Enter**.

You have edited the source to fix the error by switching to the editor and you are ready to begin "Exercise 7.14: Compiling your source changes and closing the Designer."

## Exercise 7.14: Compiling your source changes and closing the Designer

Before you begin, you must complete "Exercise 7.13: Switching between designing and editing the screen" on page 145.

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

   **Note:** You will also see the message CPD7886W Field overlaps another field with no conditions specified. You can ignore this message.

To close the Designer:
1. Click **File > Exit** from the Designer menu.

Now you are ready to review your knowledge of this module by taking the quiz. You can also apply what you have learned in this module by completing the practice tasks detailed in More practice.

**Quiz**
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. Utility notebook
   b. Design page
   c. Properties notebook
   d. Group
   a. Contains several different tabbed pages
   b. For visually designing screens
   c. A collection of one or more DDS records that make up a single screen or report
   d. Access properties of a field, record, or entire file
3. The Utility notebook contains:
   a. Selected DDS page

b. Error list page

c. Web Settings 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 source on the iSeries just as you do in the LPEX Editor. (T, F)

**More practice**

Given your experience in using the CODE Designer from the Remote Systems view, 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 the exercises in this module as you create your report. When you are familiar with the file REFMST, you can create your printer file. Use the product online help to assist you in this task.

**Module recap**

You have completed Chapter 7, "Module 7. Designing screens," on page 123. You have learned how to:

- Open a DDS member for edit with CODE Designer
- Show file-level keywords and record-level keywords
- View the details of records, record-level keywords and field-level keywords
- View the design of the payroll application main menu
- Create a group from an existing record format
- Create a new group and add a subfile record and a subfile control record
- Add columns to the subfile record
- Add fields to the subfile control record
- Copy existing fields
- Set indicators to handle field errors
- View and update record and field properties
- View keywords and the properties of a keyword

- Insert a keyword
- View help for a keyword
- Check there are no semantic errors in the DDS source
- View help for an error
- Launch the editor in read mode from the error list
- Launch the editor in write mode to fix the error
- Find a keyword in the source
- Save source changes
- Compile your source changes
- Close the Designer

You have learned about various aspects of the CODE Designer while modifying a display file and now can learn more about the product and how it is packaged. This is an optional module. Continue to Chapter 8, "Module 8. Introducing the product and Remote System Explorer (optional)," on page 149.

# Chapter 8. Module 8. Introducing the product and Remote System Explorer (optional)

This modules teaches you about IBM WebSphere Development Studio for iSeries 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 this module, you will:
- Know the goals of the product
- Know the editions of the product
- Identify the host tools and the client tools
- List and describe the iSeries application development tools

**Exercises**

The exercises within this module must be completed in order:
- "Introducing Development Studio and Development Studio Client"
- "Introducing iSeries Application Development Tools" on page 150

**Time required**

This module will take approximately **15 minutes** to complete.

## Introducing Development Studio and Development Studio Client

Development Studio Client is the ideal set of workstation development tools for creating, testing, deploying and maintaining traditional and e-business applications for your iSeries server. Development Studio Client is included with the compiler-based server product, WebSphere Development Studio. The following diagram illustrates the interaction between host and client tools:
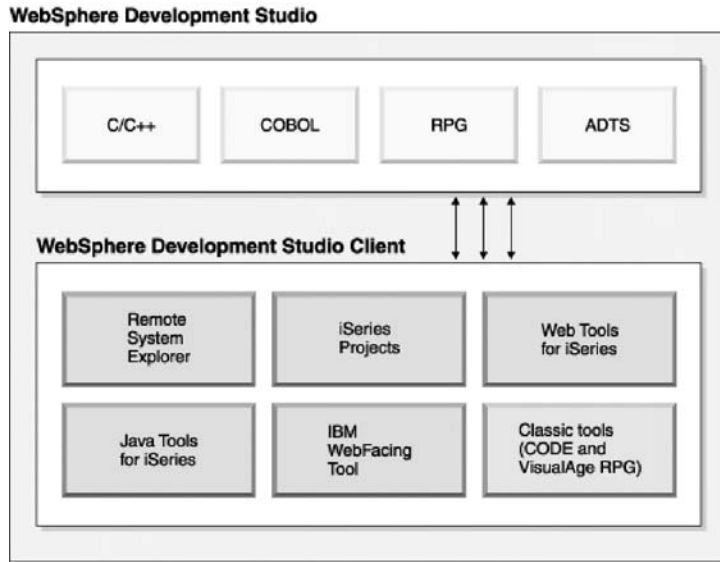
Development Studio Client is designed to help you:

1. Develop and maintain iSeries applications using the Remote System Explorer.
2. Develop Web-enabled front-ends to iSeries business logic.
3. Create GUI front-ends to iSeries business logic.

Both Development Studio Client and Development Studio Client Advanced Edition are built on the Rational Software Development Platform. This platform offers a fresh look and feel for the Eclipse workbench, and helps make it easier for you to build, integrate, and extend your applications. The Rational Software Development Platform offers a tutorials gallery and a samples gallery, to help you get up and running with the product as quickly as possible. The platform also offers user roles, which you can select from bottom-right corner of the Welcome view, (and from Window > Preferences > Workbench > Capabilities) that customize and simplify the user interface according to your programming role. These are just some of the new features provided in the Rational Software Development Platform.

The product comes in two editions for iSeries programmers. Both editions of the product are packaged with an additional base Rational product:

• Development Studio Client is packaged with Rational Web Developer. Rational Web Developer includes support for Web services, XML development tools, and core support for Java and Web development tools.

• Development Studio Client Advanced Edition is packaged with Rational 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 JavaBeans™, EGL, and for Java messaging services

You have reviewed the goals of the product and the product editions and you are ready to begin "Introducing iSeries Application Development Tools."

## Introducing iSeries Application Development Tools

Before you begin, you must complete "Introducing Development Studio and Development Studio Client" on page 149.

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. What are they and what do they do?

**Remote System Explorer**

You can manage your development 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 without errors 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 Commands 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, you 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.

**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.

**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, but you can launch it as a separate tool from the workbench.

You have identified the host tools and the client tools as well as the iSeries application development tools and now you are ready to review your knowledge of this module by taking the quiz.

**Quiz**
1. WebSphere Development Studio for iSeries:
   a. Includes all four host compilers
   b. Includes all four host compilers and the workstation-based tools named Development Studio Client
   c. Includes only the workstation-based tools named Development Studio Client
2. WebSphere Development Studio Client for iSeries includes:
   a. Rational Web Developer
   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 includes:

a. Rational Application Developer

b. Cooperative development environment (CODE)

c. VisualAge RPG

d. Java tools

e. Web Tools

f. WebFacing Tool

g. All of the above

4. Rational 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. Rational Web Developer includes support for:

a. Web services

b. XML development tools

c. Java tools

d. Web tools

e. All of the above

6. The editor can access source files on your workstation or on your iSeries system directly. When a compilation or verify action 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)

7. You can debug your program running on the iSeries system from your workstation using the Integrated iSeries Debugger. (T, F)

8. 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

9. 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

10. 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)

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

**Module recap**

You have completed Chapter 8, "Module 8. Introducing the product and Remote System Explorer (optional)," on page 149. You should now understand:

- What components make up Development Studio and Development Studio Client
- What components are included in Rational Application Developer and Rational Web Developer
- What tools you use to manage your development cycle tasks
- What the benefits are of Remote System Explorer

Finish your tutorial by reviewing the materials in Chapter 9, "Summary," on page 155.

# Chapter 9. Summary

This tutorial has taught you how to maintain a payroll application using the Remote System Explorer. You learned how to start the product and open the Remote System Explorer perspective and how to use tools and views in this perspective to connect to an iSeries system and edit, verify, compile and debug the payroll application.

**Completed Learning Objectives**

If you have completed all of the modules, you should now be able to:
- Start the product and open the Remote System Explorer perspective
- Create a connection to an iSeries and select iSeries objects from this connection
- Use the Remote Systems LPEX Editor to edit source
- Verify and compile source in the Remote Systems LPEX Editor
- Debug your interactive payroll application from the workstation
- Modify a display file
- Recognize the product features and packaging

**More information**

For more information on the product and the Remote System Explorer, see http://ibm.com/software/awdtools/iseries.

**Quiz answers**

Here are the answers to each module quiz.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 1d | 1c | 1d | 1d | 1c | 1d | 1b | 1b |
| 2e | 2e | 2T | 2d | 2T | 2c | 2 aa, bb, cd, dc | 2g |
| 3b | 3e | 3a | 3g | 3e | 3T | 3e | 3g |
| 4 aa, bd, cb, dc | 4c | 4T | 4T | 4c | 4b | 4h | 4e |
| 5e | 5T | 5d | 5T | 5c | 5T | 5T | 5e |
| | 6T | 6c | 6d | 6T | 6T | 6T | 6T |
| | 7d | 7b | 7e | 7e | 7c | 7T | 7T |
| | | 8c | 8T | 8T | 8T | 8T | 8f |
| | | 9c | | 9a | 9T | 9T | 9e |
| | | 10a | | 10a | 10f | | 10T |
| | | 11b | | | 11T | | 11b |
| | | 12T | | | 12e | | |
| | | 13d | | | | | |
| | | 14e | | | | | |
| | | 15d | | | | | |

| | | 16c | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| | | 17b | | | | | |
| | | 18c | | | | | |
| | | 19T | | | | | |
| | | 20T | | | | | |

# 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, 2005. 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
Rational
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®, 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.

Linux is a registered trademark of Linus Torvalds.

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

**IBM** ®

Printed in USA