Tutorial: Maintaining an ILE COBOL application: Introducing a new level of server tool integration for iSeries application development

About the tutorial

This tutorial through a series of exercises, checkpoints and practices, introduces you to a new level of tool integration for iSeries application development. The iSeries server development capabilities of the new integrated development environment (IDE) provides many compelling reasons for you to upgrade from Application Development Toolset (ADTS) or CoOperative Development Environment (CODE).

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

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

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

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

Completing this tutorial should take you about 2 hours.

Work on the exercises in sequence. The pictures in the exercises show similar tasks. Some of the names and icons may be different from the environment you will be working with when you complete the exercises.

Following each exercise it is recommended that you take the checkpoint quiz. Taking each quiz will help you to determine if you have mastered the exercise content and are ready to move on to the next exercise. The quiz answers are included at the end of this tutorial so you can check your answers.

Following each exercise checkpoint is a practice section. This section gives you the opportunity to apply what you have learned in the exercise. Often you will complete different but similar tasks to

what you have just learned in the exercise. The Development Studio Client for iSeries online help can assist you in completing the practice tasks.

The tutorial is available in PDF format. You can view the PDF version from the WebSphere Development Studio Client for iSeries Version 5.1 product CD or the product Web site (ibm.com/software/adwtools/iseries).

Tutorial business problem

You are an iSeries COBOL application developer but you are 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. You are currently using the iSeries Application Development Tools (ADTS) products (PDM, SEU, SDA) or CoOperative Development Environment (CODE). ADTS has been the traditional method for developing and maintaining server-side iSeries applications. But now there is a new set of highly integrated and highly extendible tools for iSeries RPG, COBOL, C, C++, CL and DDS development. These new tools offer you a development experience that is consistent with the experience for developing Java, Web, Web Services, and XML applications. The new tools also allow you to leverage the classic CODE tools for extremely rich editing and DDS design support. These new generation tools offer rich support for exploring the file system, compiling/building, editing, running, and debugging. They offer significant productivity and usability gains, support for disconnected and team development, and a common harness for the tight integration of IBM and partner-supplied tools for server development.

You know this is the place you and your company want to be especially as you both transition to delivering iSeries applications through the Web. Your first step in modernizing your iSeries applications is to move to the next generation of iSeries server application development tools. Here is what you and your company know about these next generation tools.

The first server application development tool is the Remote System Explorer which has its own tools and views. Remote System Explorer is similar to Programming Development Manager (PDM) in that it allows you to drill down to the QSYS file system, or use filters to list specific objects within the QSYS file system.

The Remote Systems Explorer goes well beyond PDM however! It also allows you to explore iSeries jobs and commands, and the IFS file system. Further, you can also use it to explore the file system of remote Linux, UNIX and Windows systems.

The Remote System view is the primary drill-down view, similar to PDM. Double-click a member and you open the Remote Systems LPEX Editor, built right into the IDE and with rich editing functions. This editor goes well beyond SEU's function; it is a superset of SEU! and is getting close to all of the CODE Editor's functions. The syntax checker is ported down from SEU, the compilers are imbedded for verifying errors and the reference manuals are built-in and F1 cursor sensitive. You can see the program hierarchy in the Outline view. The content assist is a popular feature. There is explicit and rich iSeries specific support for not only edit, but verify, compile, run and debug of RPG, COBOL, C, C++, CL and DDS not just from the Remote Systems LPEX Editor but also from the IDE in your very own Remote System Explorer perspective; the place for views and tools specific for iSeries server application development.

Instead of the Remote Systems view which is a tree view, PDM users are use to a table, so there is an iSeries Table view that shows what the tree shows but in table format. You can sort the columns when you click on a column heading. The contents of the table are easily replaced when you click on a file in the Remote Systems view. There are right-click actions that are the same as PDM's and a command line at the bottom of the table just like PDM.

There is no built-in tool for display file and printer file development yet, but you can easily launch the CODE Designer from the Remote Systems view.

COBOL program debug is easy from the Remote System Explorer interactive debug perspective. Through this perspective you can debug Java, OPM/ILE, RPG, COBOL, CL, ILE C and C++ programs.

With this information in hand you are ready to move on.

Before you begin

Before you begin, you must install the following:

On iSeries:

- OS/400 Version 5 Release 1 or later. The Remote System Explorer component will need additional PTFs. The PTF information is available at ibm.com/software/adwtools/iseries. OS/400 is needed since this tutorial includes programming objects located on the iSeries server and the applications contain backend code on the iSeries server.
- IBM WebSphere Development Studio for iSeries (5722-WDS)
- IBM WebSphere Development Studio for iSeries requires V5R2M0 OS/400 (5722-SS1)
- RSELAB savf restored on iSeries
- iSeries system servers started with the command STRHOSTSVR *ALL and STRTCPSVR *DDM.

On the workstation for iSeries server application development:

- Intel Pentium II processor
- 256 MB RAM minimum
- Required hard drive space: 1650 MB
- Additional 700MB of temporary hard drive space is needed during product installation
- Windows: VGA graphics card (800 x 600, or higher, recommended, 256 colors)
- CD-ROM/DVD drive
- Mouse or pointing device

Software

- IBM WebSphere Development Studio Client for iSeries, Version 5 (5724-A81)

- Any PTFs available. PTFs are available at ibm.com/software/adwtools/iseries

- eNetwork Personal Communications Version 5.5 or later

- TCP/IP access to an iSeries system.

- For Windows NT:

  o Windows NT Server or Workgroup, Version 4.0, Service Pack 6a or higher

  o Microsoft Internet Explorer 5.5 Service Pack 1 or higher

- For Windows 2000:

  o Windows 2000 Professional, Service Pack 2 or higher

  o Microsoft Internet Explorer 5.5 Service Pack 1 or higher

- For Windows XP

  o Windows XP Professional

To use the library for this tutorial, you must restore it to your iSeries.

On your iSeries, create a save file:
       CRTSAVF <library_name>/RSELAB

On your workstation, open a Command Prompt window and go to the directory where
you downloaded the save file and do the following:

ftp <iSeries_name>
<user_name>
<password>
binary
put rselab.sav <library_name>/RSELAB
quit

On your iSeries, restore the RSELAB library:
       RSTLIB SAVLIB(RSELAB) DEV(*SAVF) SAVF(<library_name>/RSELAB)

Once you have restored the library, you can delete the save file <library_name>/RSELAB.

Conventions that are used in this tutorial

This tutorial uses typographical conventions in the text to help you distinguish between the names of
controls and text that you type. For example:

- Menu items are in boldface font:

  Click **Menu → Menu choice**

- The names of fields, check boxes, and buttons are also in boldface font:

  Type text in the **Field** field.

- Text that you type is in example font on a new line:

  ```
  This is the text that you type.
  ```

Related information

This tutorial covers the most common tasks that you can accomplish with the Remote System Explorer. For more information about related tasks, see the following documents:
WebSphere Development Studio Client for iSeries
- Maintaining and Developing iSeries applications online help
- Remote System Explorer presentation on the Library page (ibm.com/software/adwtools/iseries)
- Remote System Explorer demo on the Library page (ibm.com/software/adwtools/iseries)
- iSeries Server Application Development Quick Tour on the Library page (ibm.com/software/adwtools/iseries)

Exercise 1: Introducing to WebSphere Development Studio and Remote System Explorer

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

At the end of the exercise, you should be able to:
- Describe the Development Studio product
- Describe the Development Studio Client product
- Describe how Development Studio Client fits into the WebSphere Studio family of products
- Explain the tools available to iSeries programmers for iSeries application development

Let's get started by first looking at what makes up the Development Studio and Development Studio Client products.

Development Studio and Development Studio Client

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

WebSphere Development Studio

C/C++	COBOL	RPG	ADTS

WebSphere Development Studio Client

| Remote System Explorer	iSeries Projects	Web Tools for iSeries
Java Tools for iSeries	IBM WebFacing Tool	Classic tools (CODE and VisualAge RPG)
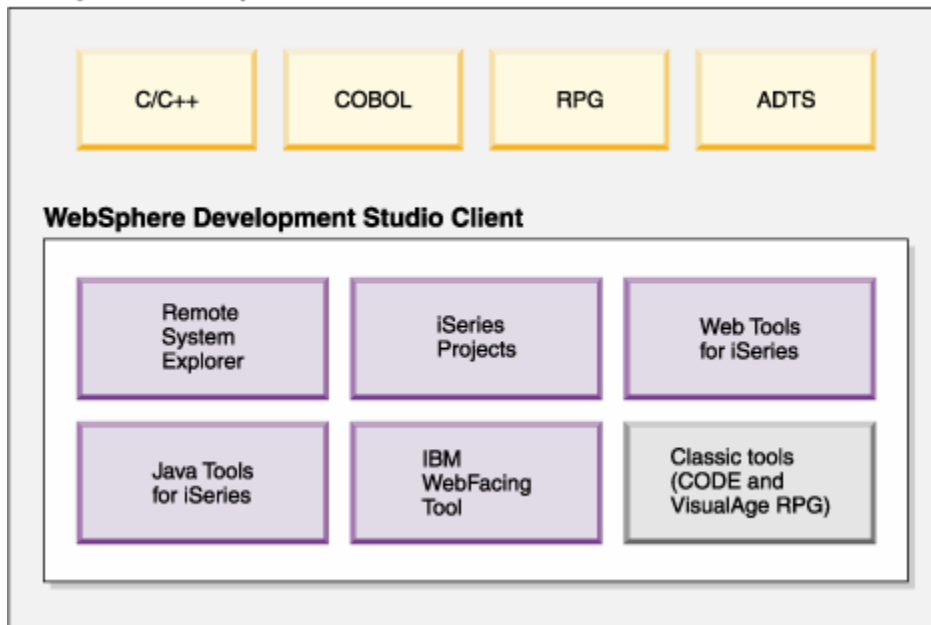
Development Studio Client is designed to help you:
1. Develop and maintain iSeries applications using the Remote System Explorer
2. Develop Web GUIs for iSeries classic applications using the IBM WebFacing Tool and other Web Tools
3. Develop client and server applications for iSeries using Java Tools
4. Work with other Integrated Site Developer Tools (XML, Web Services, SQL, Relational Databases)

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

- IBM WebSphere Development Studio Client for iSeries includes WebSphere Studio Site Developer. Site Developer includes support for Web services, XML development tools, and core support for Java and Web development tools.

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

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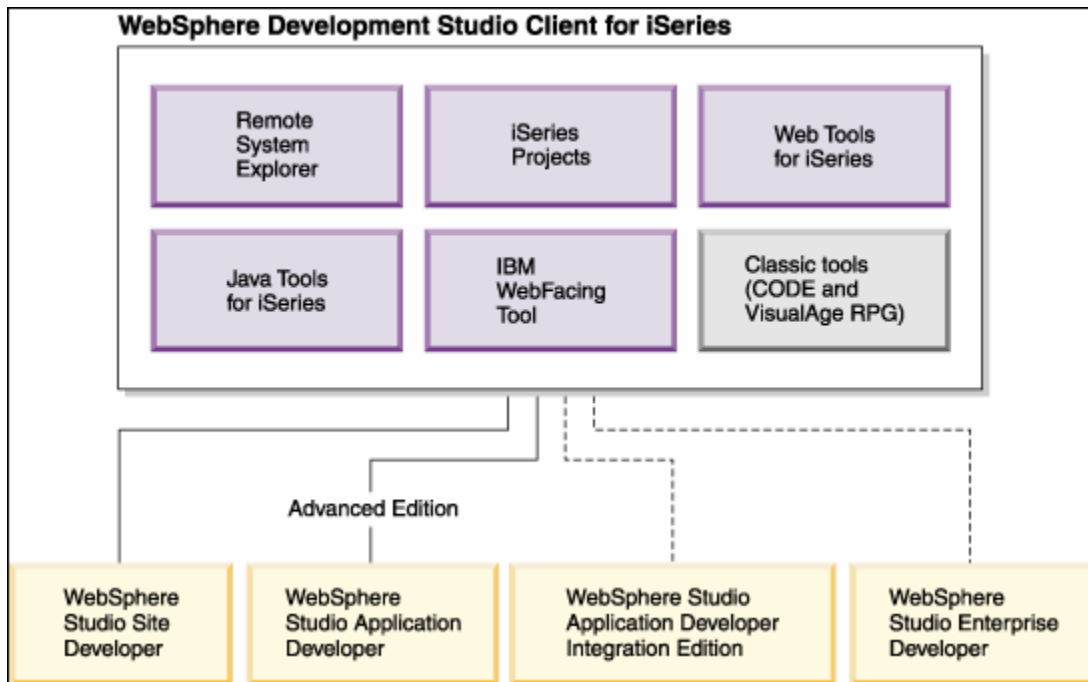
iSeries Server Application Development Tools

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

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

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

language references, and context-sensitive help make finding the information you need just a keystroke away.

Shells and Commands in the Remote Command view

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

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

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

Program Verifier

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

CODE Designer

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

The CODE Designer interface was designed to help the novice DDS programmer create screens, reports and databases quickly and easily without worrying about the details of the DDS language, while at the same time letting the expert DDS programmer get access to all the features and power of the language. Let's now step through each part of the interface and update some DDS as well.

iSeries Debugger

With the integrated iSeries debugger you can debug your program running on the iSeries system from a graphical user interface on your workstation. 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.

In this tutorial you are going to learn some of the basic features and functions of each of these tools. We are confident that Development Studio Client will save you lots of time and effort in your day-to-day programming tasks. It will make you a more efficient and effective programmer. At the same time, it will save cycles on your iSeries and better yet it will get you ready for the next step and that is moving your iSeries applications to the Web.

So, let's get started! Complete the checkpoint below to determine if you are ready to move on to the next exercise.

Checkpoint

1. WebSphere Development Studio for iSeries:
    - A. Includes all four host compilers and all traditional tools (ADTS)
    - B. Includes all four host compilers, all traditional tools (ADTS) and unlimited licenses of the workstation-based tools named Development Studio Client
    - C. Includes only the workstation-based tools named Development Studio Client
    - D. Includes only the four host compilers
2. WebSphere Development Studio Client for iSeries Version 5 includes:
    - A. WebSphere Studio Site Developer Version 5 for e-business development
    - B. Cooperative development environment (CODE)
    - C. VisualAge RPG
    - D. Java tools
    - E. Web tools
    - F. WebFacing tool
    - G. All of the above
3. WebSphere Development Studio Client for iSeries Advanced Version 5 includes:
    - A. WebSphere Studio Application Developer Version 5 for e-business development
    - B. Cooperative development environment (CODE)
    - C. VisualAge RPG
    - D. Java tools
    - E. Web Tools
    - F. WebFacing Tool
    - G. All of the above
4. WebSphere Studio Application Developer includes support for:
    - A. Creation and maintenance of J2EE applications
    - B. Creation and maintenance of Web services
    - C. Enterprise Java Beans
    - D. Java Messaging Services
    - E. All of the above
5. WebSphere Studio Site Developer includes support for:
    - A. Web services
    - B. XML development tools
    - C. Java tools
    - D. Web tools
    - E. All of the above
6. You can manage your development-cycle tasks in:

A. The Remote System Explorer
B. iSeries Projects
C. The IBM WebFacing tool
D. All of the above

7. With the Remote System Explorer and iSeries Projects you can view iSeries libraries, files and members. You can also launch the host compilers, the workstation editor, and various debuggers. (T, F)

8. Your program editing tasks are simplified with the:
   A. The Remote System Explorer
   B. iSeries Projects
   C. The IBM WebFacing tool
   D. The LPEX Editor
   E. All of the above

9. The editor can access source files on your workstation or on your iSeries system directly. When a compilation results in errors, you can jump from the compiler messages to an editor containing the source. The editor opens with the cursor positioned at the offending source statements so that you can correct them. (T, F)

10. You can debug your program running on the iSeries system from your workstation using:
    A. The Remote System Explorer
    B. iSeries Projects
    C. The IBM WebFacing tool
    D. The LPEX Editor
    E. The Integrated iSeries Debugger
    F. All of the above

11. The graphical design tool that lets you design your screens and reports visually and then generates DDS source for you is:
    A. The Remote System Explorer
    B. CODE Designer
    C. The IBM WebFacing tool
    D. The LPEX Editor
    E. The Integrated iSeries Debugger

12. Before you compile your code on an iSeries system, you can ensure that there are no errors by invoking the:
    A. The Remote System Explorer
    B. CODE Designer
    C. The IBM WebFacing tool
    D. The LPEX Editor
    E. The Integrated iSeries Debugger
    F. Program Verifier

13. You can use the Remote Commands view to:
    A. Run commands in a command shell
    B. Display and interpret the output of a program
    C. Enter input to a program
    D. Display and manage different commands and shells from the same view
    E. All of the above

14. The integrated iSeries Debugger enables you to run your program, set line, watch, and service entry point breakpoints, step through program instructions, examine variables, and examine the call stack. (T, F)
15. If the Advanced Version of the product is not installed on your workstation then you will not see the word Advanced in the Start menu. (T, F)

What you just did

In this exercise you were introduced to Development Studio and Development Studio Client and how Development Studio Client fits in the WebSphere Studio family of products, the difference between Development Studio Client and Development Studio Client Advanced and what iSeries server application development tools the Development Studio Client workbench provides.

In the next exercise you will start Development Studio Client and open the Remote System Explorer.

Exercise 2: Starting Development Studio Client and Opening the Remote System Explorer

In this exercise you are introduced to workbench, the workspace, a perspective and specifically the Remote System Explorer perspective. You then learn how to start Development Studio Client and how to open the Remote System Explorer perspective.

At the end of the exercise, you should be able to:
- Explain workspace, workbench and perspectives
- Describe the Remote System Explorer perspective
- Start Development Studio Client
- Open the Remote System Explorer perspective

First let's start Development Studio Client.

Starting Development Studio

To start Development Studio Client:

1. Click **Start** on the task bar of your desktop

2. Select **Start ->Programs-> IBM WebSphere Studio -> Development Studio Client Advanced Edition for iSeries**

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

A window may appear. Here you specify the name of the workspace where your projects and other resources such as folders, subfolders and files that you are developing in the workbench will reside.
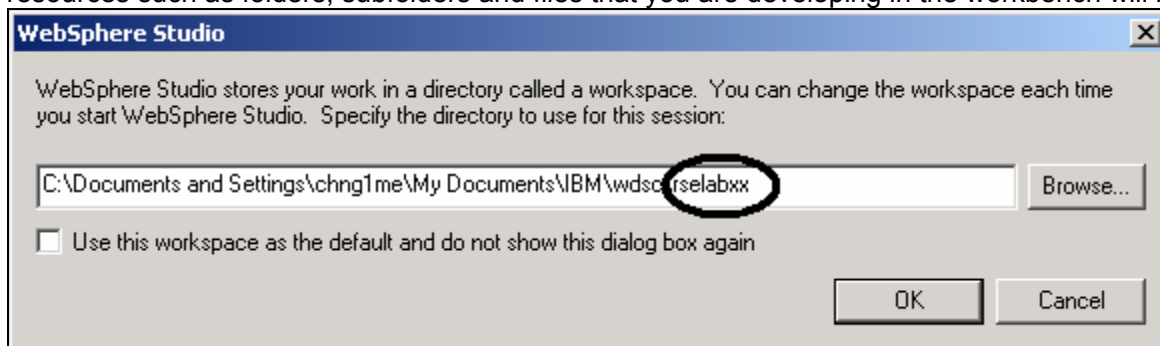
Figure 1: WebSphere Studio window for specifying workspace directory name

3. (Optional) Change the field in this window and use a unique directory name, for example, rselabXX (where XX is your unique number). If you use 01 use directory name rselab01 as shown in Figure 1. After a few moments of loading, the workbench appears.
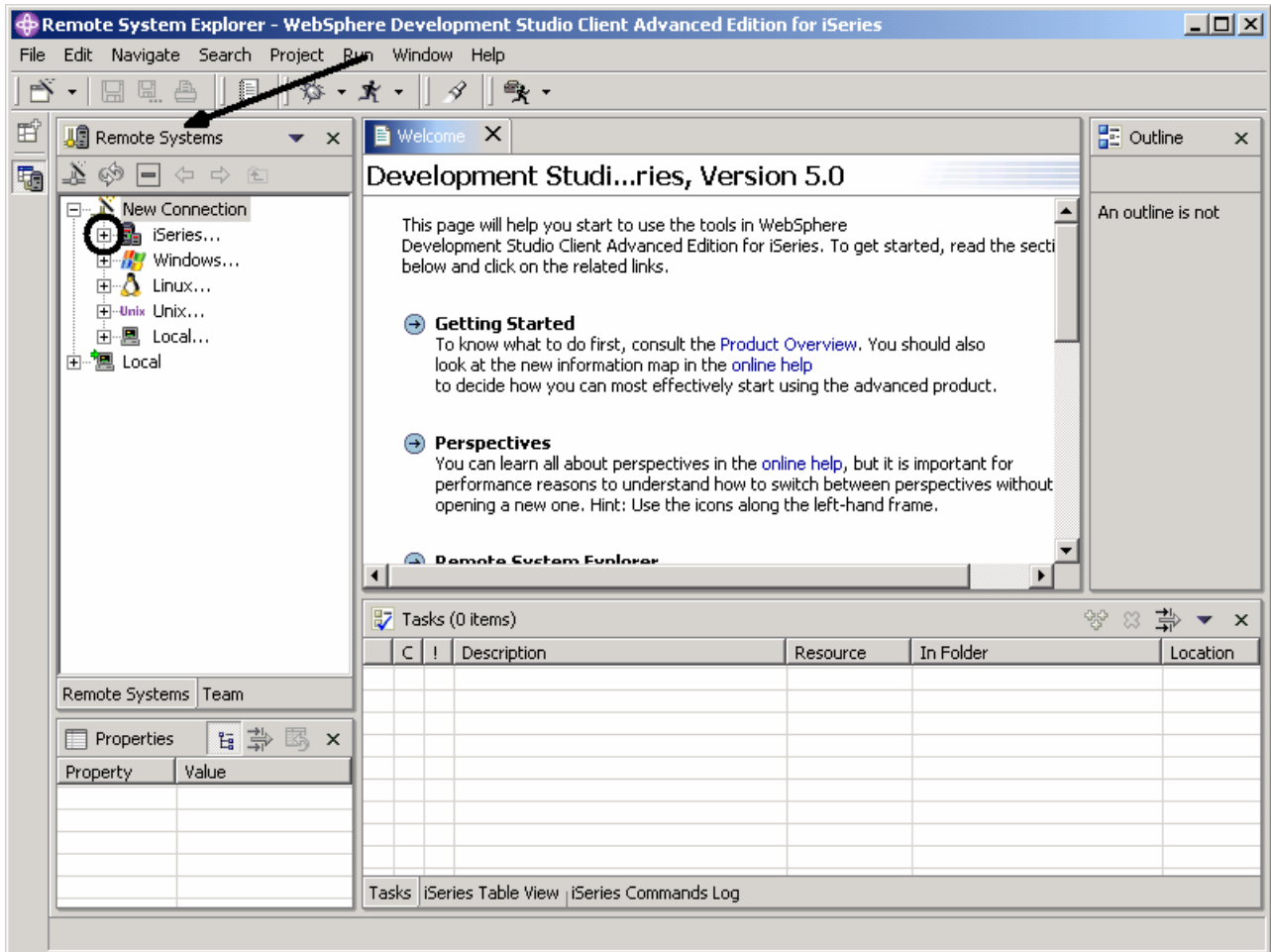
Figure 2: Workbench with Remote System Explorer

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

Open Remote System Explorer perspective

1. Check for the name of the perspective, the arrow in Figure 2 indicates where to look for the

   perspective name.

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

debugging Java programs. Perspectives contain views and editors and control what appears in certain menus and tool bars.

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

2. Select **Window** from the workbench Menu bar
3. Select **Open Perspective**
4. Select **Remote System Explorer**

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

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

Checkpoint

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 views that you would commonly use while editing Java source files
      B. Contains views that you would use while debugging Java programs
      C. Contains views that you would use while developing Web applications
      D. Contains views that you would use while maintaining iSeries applications.

      A. Java perspective
      B. Web perspective
      C. Remote System Explorer perspective
      D. Debug perspective

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

Practice

Given your experience in opening the Remote Systems Explorer perspective, open the Web perspective. Explore the tools and views available to the Web developer. Next open the Java perspective. Explore the tools and views available to the Java developer. Now since you are supposedly in the Java perspective, open the Web perspective. Be careful not to open another Web perspective.

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

What you just did

In this exercise you were introduced to workbench, the workspace, a perspective and specifically the Remote System Explorer perspective. You then learned how to start Development Studio Client and how to open the Remote System Explorer perspective.

In the next exercise you will configure a connection to an iSeries system and then connect to your iSeries system.

Exercise 3: Configure a connection to an iSeries system and Connect to your iSeries

In this exercise you create a connection to an iSeries server and select objects using the Remote System Explorer perspective. You will learn the steps to create a connection. You will then learn how to find a library in your library list. Finally you open a member in the Remote Systems LPEX Editor. You also learn about several views such as the Remote Systems view, iSeries Table view, and the Outline view.

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

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

Ok, let's get started. Remember you have already opened the Remote System Explorer perspective in the previous exercise.
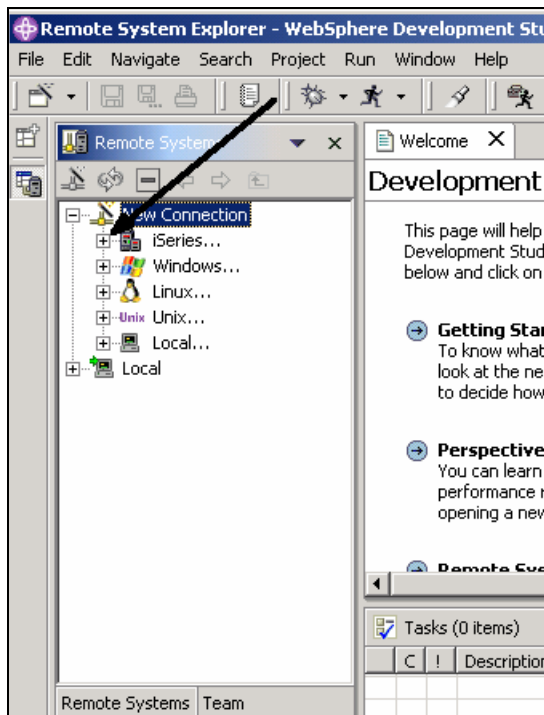
Figure 3: Configure a connection

1. Switch to the Remote System Explorer perspective.
2. 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.
Expand **iSeries** to configure a connection to an iSeries system

The **New – Name personal profile** page of the New Connection wizard appears:
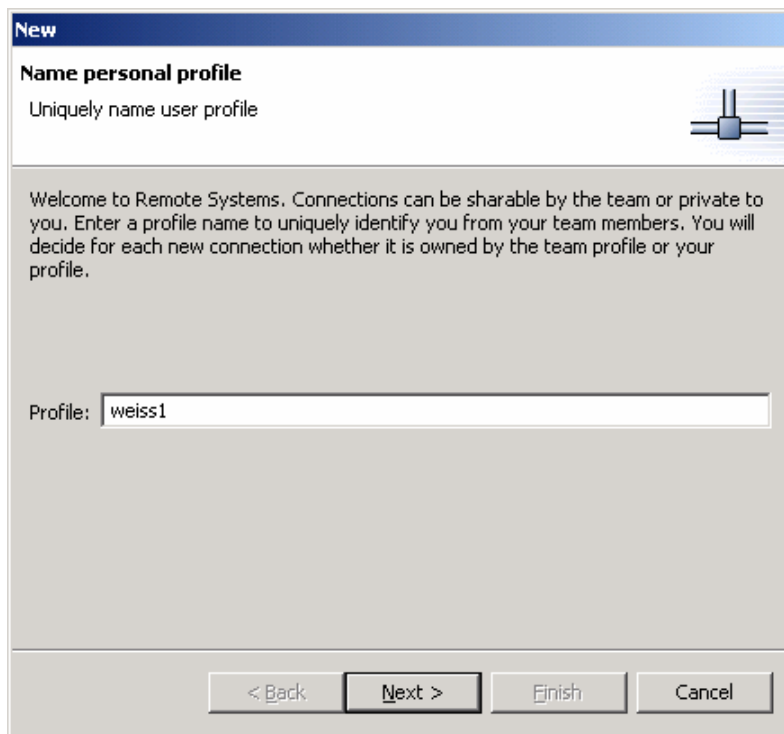
Figure 4: New – Name personal profile page

On this page of the New connection wizard and since it is the first time you connect to an iSeries system, you need to define a profile. All connections, filters, and filter pools belong to profiles. Profiles offer a way to group connections, share connections, or keep them private, and they help you partition data if you have a lot of connections or filter pools. Your first profile will be for your local workstation. This is your personal profile. You use this profile as you want to keep your connection private. You do not want to share resources and information with other people.

3.   Leave the **Profile** information as is. Click **Next**

The second page of the **New connection** wizard shows:
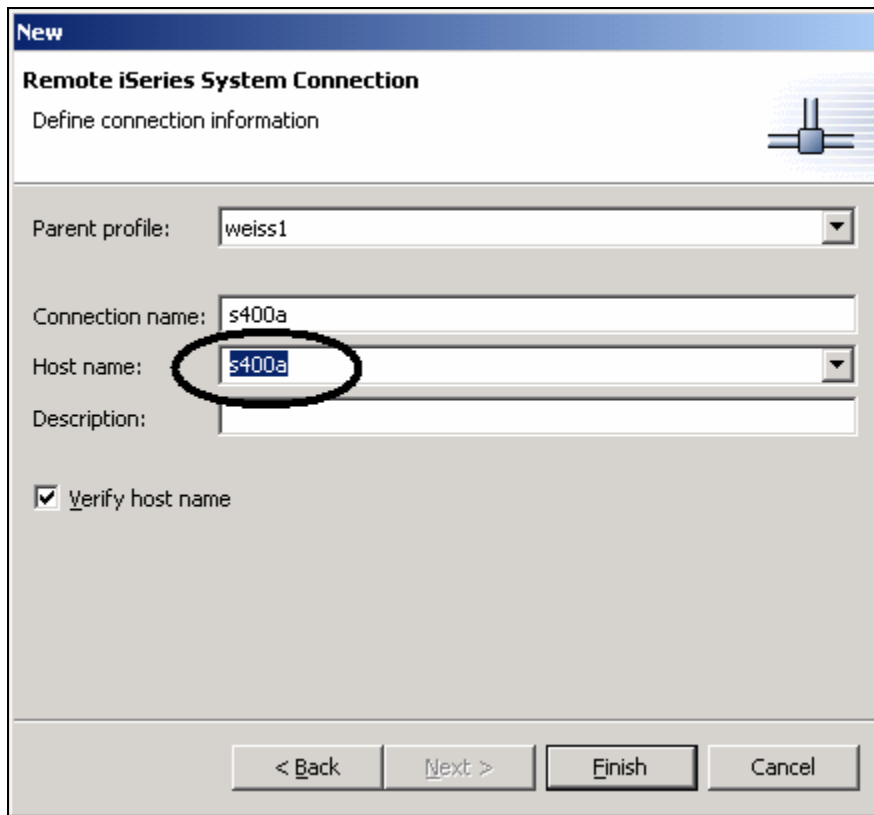
Figure 5: The Remote iSeries Sysem Connection page

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

4. Enter the name of your host system in the **Host name** field

   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.

5. Leave the **Parent profile** as is. You don't need to change it. As the files will be kept private the parent profile defaults to the default profile shown on the previous page of the wizard. If you were sharing resources you would select a team profile that you would have created earlier.

6. Leave the **Verify host name** check box checked

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

Connecting to your iSeries system

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

- iSeries Objects is a PDM-like group, allowing access to libraries, objects and members

- iSeries Commands allow you to predefine command sets each of which contain one or more

    often used commands. When run, all commands in a command set are sent to the remote

    system and executed, and the results are logged in the Commands view

- iSeries Jobs allow you to see various jobs, subset by job attributes, and to perform a limited

    number of operations on those jobs.

- IFS Files allow you to explore folders and files in the Integrated File System of the remote

    iSeries system
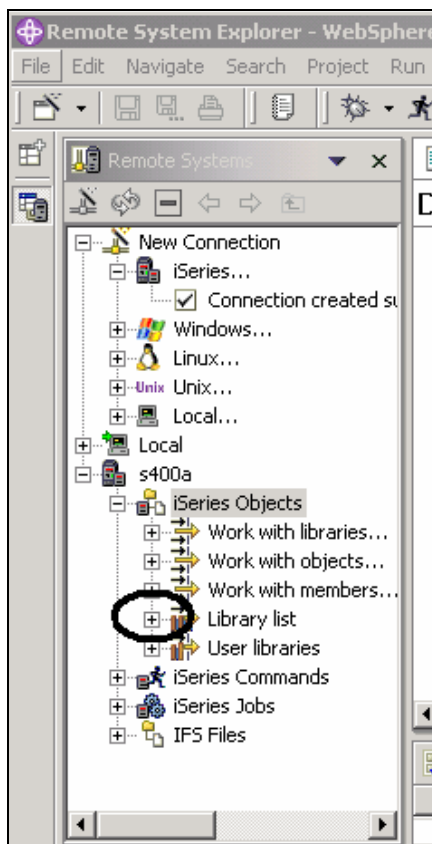
Figure 6: Remote Systems view with iSeries connection

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 STRPDM option 12 you can start with the pre-defined Library list filter, that when expanded lists all libraries in your library list.)
    *   User libraries

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

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

3. Click the plus sign beside **Library list**.

    Now the connection will be activated and you will be prompted for a user ID and password.
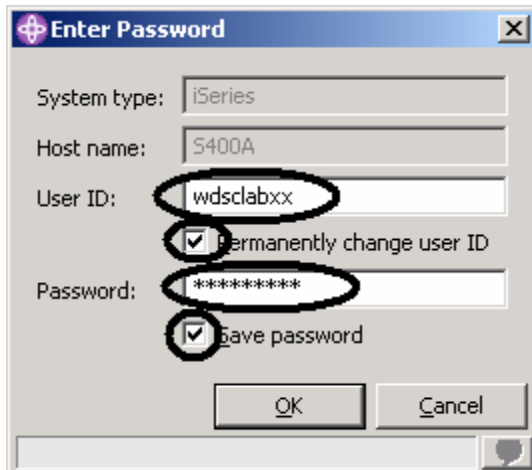
Figure 7: Enter password window

4. Enter your user ID and password.

5. Check the check box **Permanently change user id**

6. Check the check box **Save password**

7. Click **OK**.

Make sure that your user ID has been setup, so that your library has been added to the library list automatically. You can use the properties of iSeries Commands to set connection information such as adding a library to a library list.

Note: if you do not see RSELABxx in your library list you can right click on the Library list and use the Library List Entry window to add RSELABxx to your library list.

Back in the workbench in the Remote System Explorer perspective you will see the libraries in your job's library list.
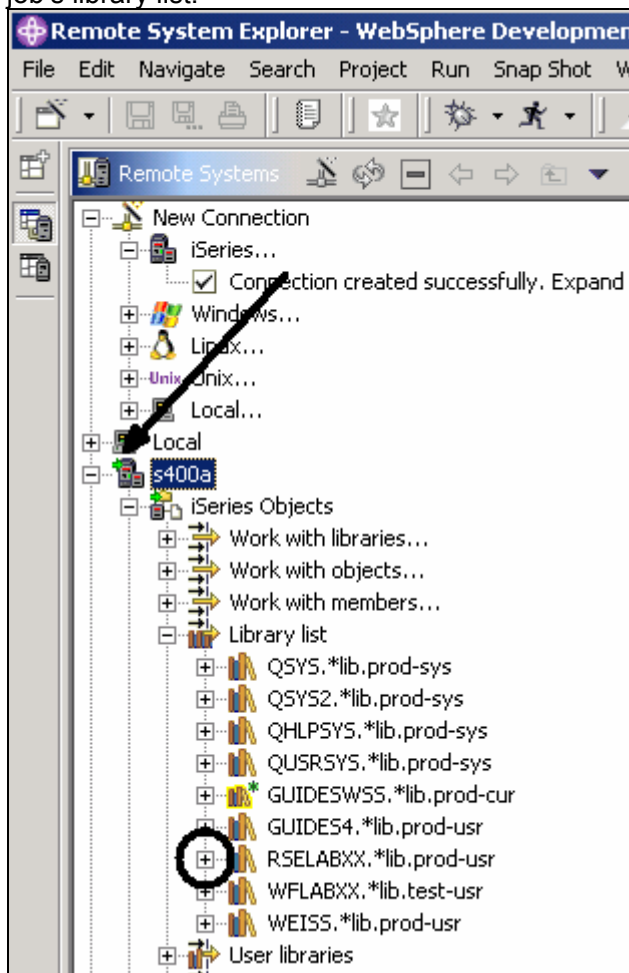
Figure 8: Remote Systems view with libraries in library list

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

Also your **RSELABxx** library should have a small green star that indicates it is the current library. In Figure 8 the library **GUIDESWSS** is the current library, if you have sharp eyes you can spot the star in the icon.

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

Viewing and accessing objects in the Remote System Explorer perspective

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

To view 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 also. The list of actions depends on the object selected and whether you selected one or multiple objects. For a source file the pop-up menu has an action to create a new member within the selected file, to refresh the contents of a file if it is expanded, to rename the file, copy the file and delete the file. These actions remotely run the appropriate iSeries command and you will see it logged in the Commands view.

2. Drill-down through the files in the Remote Systems view until you find **QCBLLESRC** source file and then expand it.
3. Drill-down through the files in QCBLLESRC until you find **QDDSSRC** source file and expand it as well.
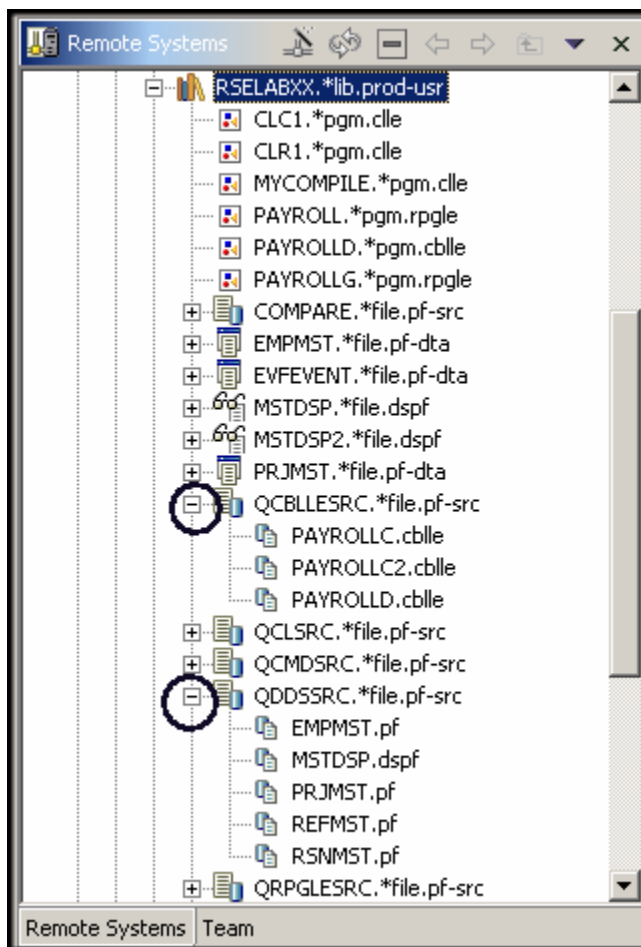
Figure 9: Remote Systems view with expanded source files

Now you can see and access the members in these two source files. For each member you can right-click and select from a number of actions. The exact list of actions depends on whether the member is a data file or source file and whether you select one or multiple members. For a member, the popup menu actions are edit, rename, copy, move, delete, and compile.
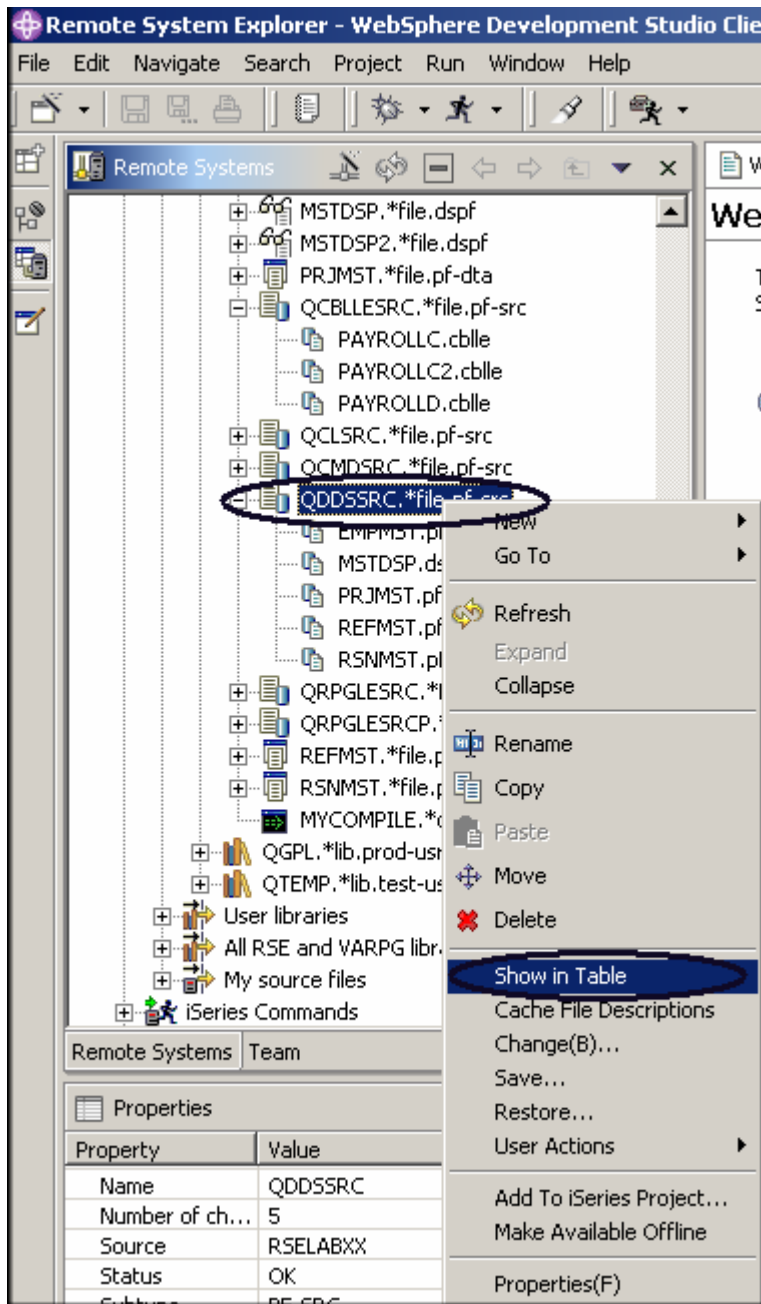
Figure 10: Select Show in table for QDDSSRC file

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

also perform actions against these items such as editing and compiling.

4. Right-click the **QDDSSRC** file and select **Show in Table**.

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.
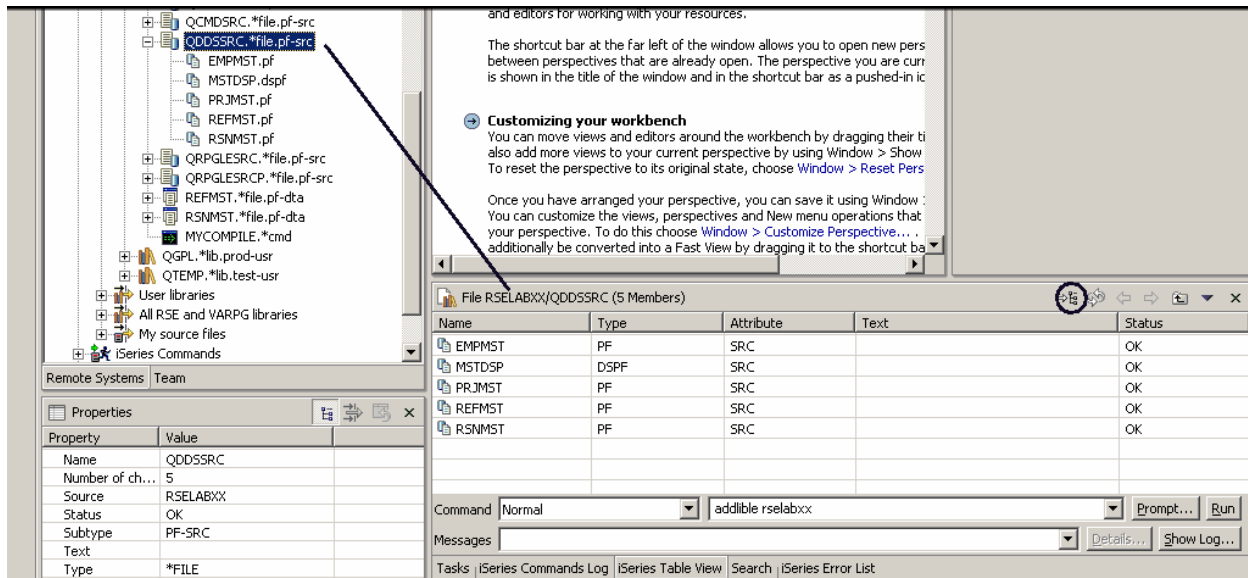
Figure 11: Table view with lock/unlock button

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.
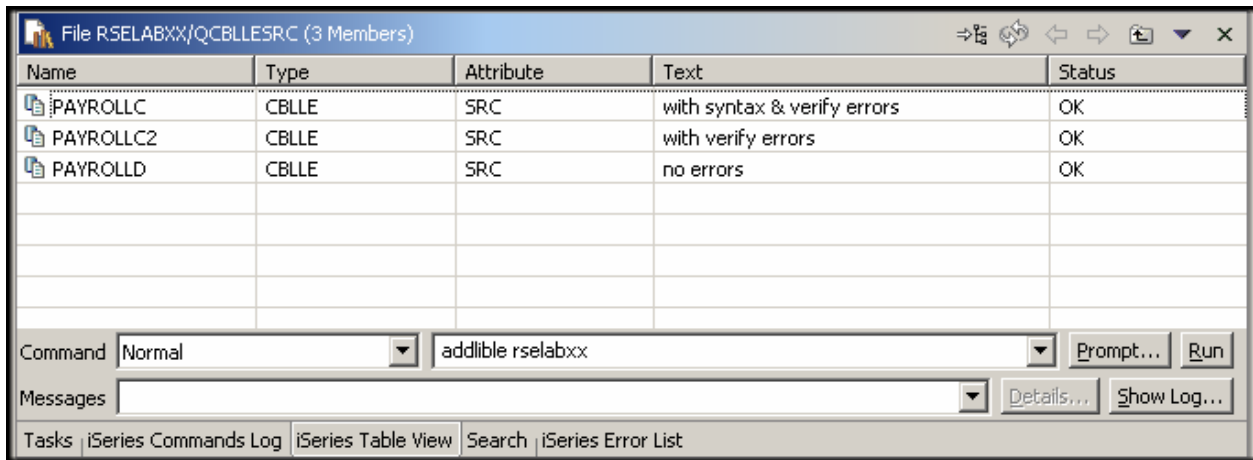
If the lock is enabled then when you click on various items in the Remote Systems view, this view does not change the input to the iSeries Table view.

To enable or disable the lock, you can click it once to change its state.

Tip: You click any of the column titles to sort by that attribute, for example, click **Type** to sort the items by type, which sorts them from A to Z. Click the same attribute again to reverse the order that the items are sorted, for example, click **Type** again to sort from Z to A.

You can unlock this iSeries Table view. It then switches to the selected object in the Remote Systems view, as you select objects in the Remote Systems view.

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

    Since it is in the unlock position by default it is now unlocked from the current QDDSSRC object.

    Meaning it is not locked to QDDSSRC.

6. In the Remote Systems view, select QCBLLESRC. The table shows the members in

    QCBLLESRC

Name	Type	Attribute	Text	Status
PAYROLLC	CBLLE	SRC	with syntax & verify errors	OK
PAYROLLC2	CBLLE	SRC	with verify errors	OK
PAYROLLD	CBLLE	SRC	no errors	OK

File RSELABXX/QCBLLESRC (3 Members)

Command Normal ▼ addlible rselabxx ▼ Prompt... Run
Messages ▼ Details... Show Log...
Tasks | iSeries Commands Log | iSeries Table View | Search | iSeries Error List

Figure 12: iSeries Table view for QCBLLEESRC

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

7. Right-click on the iSeries Table view window heading

8. Select **Fast view** from the pop-up menu

    You will see the iSeries Table view icon on the task bar on the left hand side of the workbench.

9. Maximize the Editor window

10. Click the iSeries Table view icon on the task bar to see this view. Click the icon again to close this view.
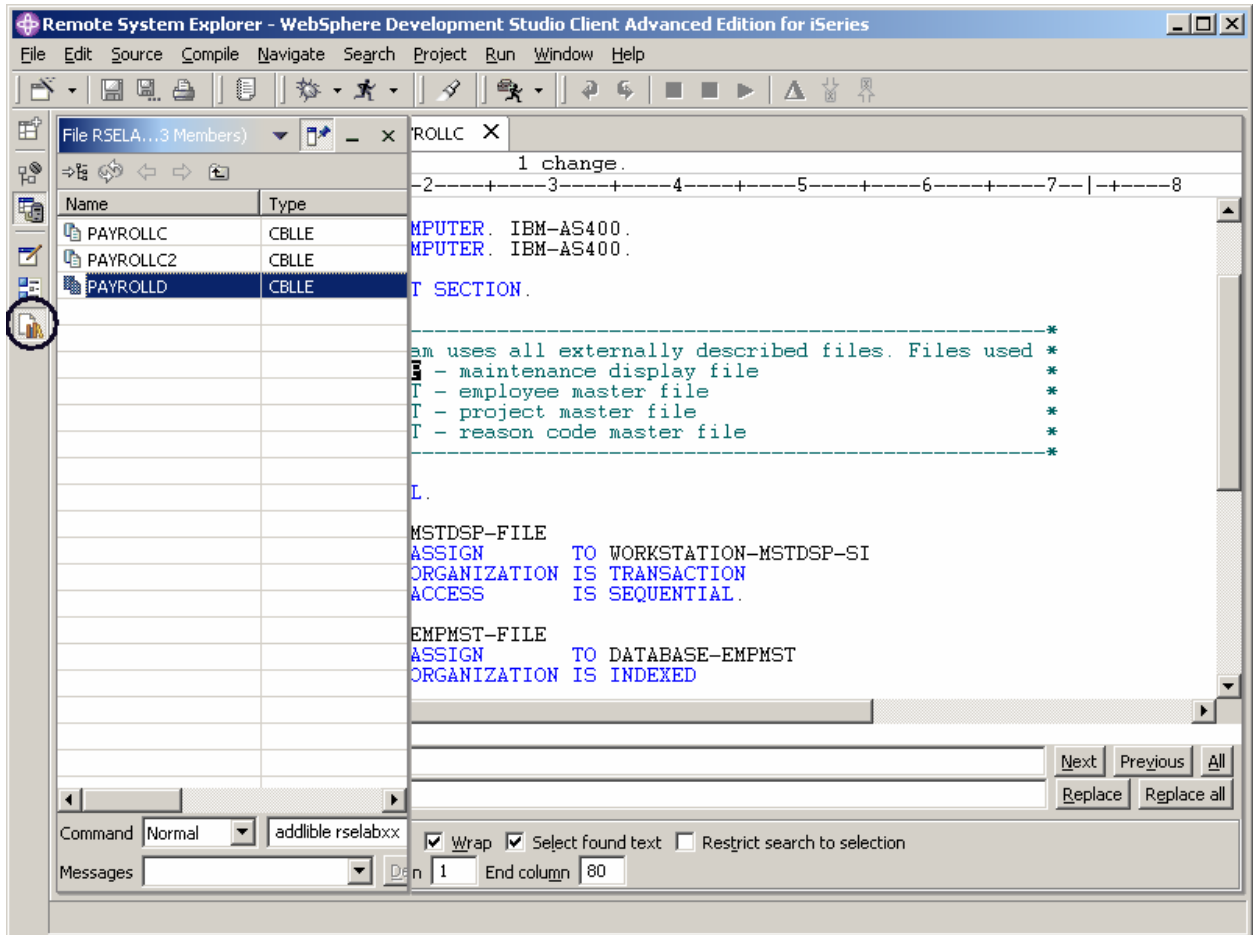
Now you are ready to use the Remote Systems LPEX Editor to edit the member MSTDSP found in QDDSSRC.

11. Double click the PAYROLLC Editor window.

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

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

The Remote Systems LPEX Editor opens, built right into the workbench, with rich editing functions and is iSeries aware! It is a superset of SEU! The syntax checker is ported down from SEU, the compilers embedded for verifying errors and the reference manuals are built-in and F1 cursor sensitive. The Outline view will show us the program hierarchy. There is explicit and rich iSeries support for verify, compile, run and debug of RPG, COBOL, C, C++, CL and DDS from the Remote Systems LPEX Editor.

Many of the editing features offered in the CODE Editor for RPG, COBOL, CL, and DDS source, such as syntax checking, automatic uppercasing, program verification, and so on, are now available in the Remote Systems LPEX Editor.
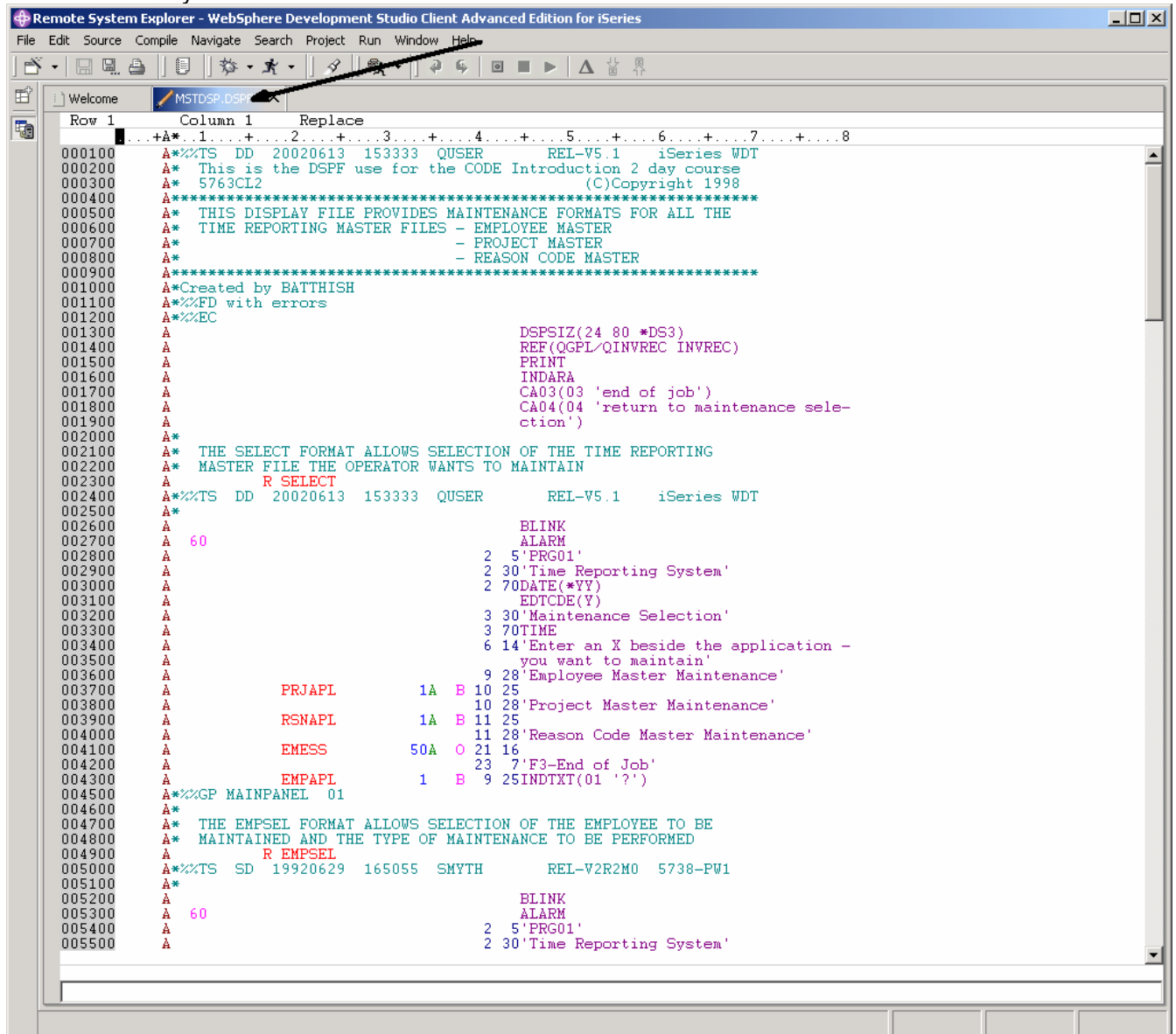
Figure 13: Source Editor with DDS member

13. Double click the **MSTDSP** tab to maximize the Editor window, as shown by the arrow in Figure 13. Double click the **MSTDSP** tab again to return the window to its original size.

Opening a second source member

Next let's open a second member in the editor. Back in the Remote Systems view:

1. Double-click member **PAYROLLC** in the QCBLLESRC source file.

   This member will be loaded into the editor as well. Your Editor window will look something like Figure 14. Notice the two tabs in the editor pane. Click on each tab to switch from one edit session to another edit session.
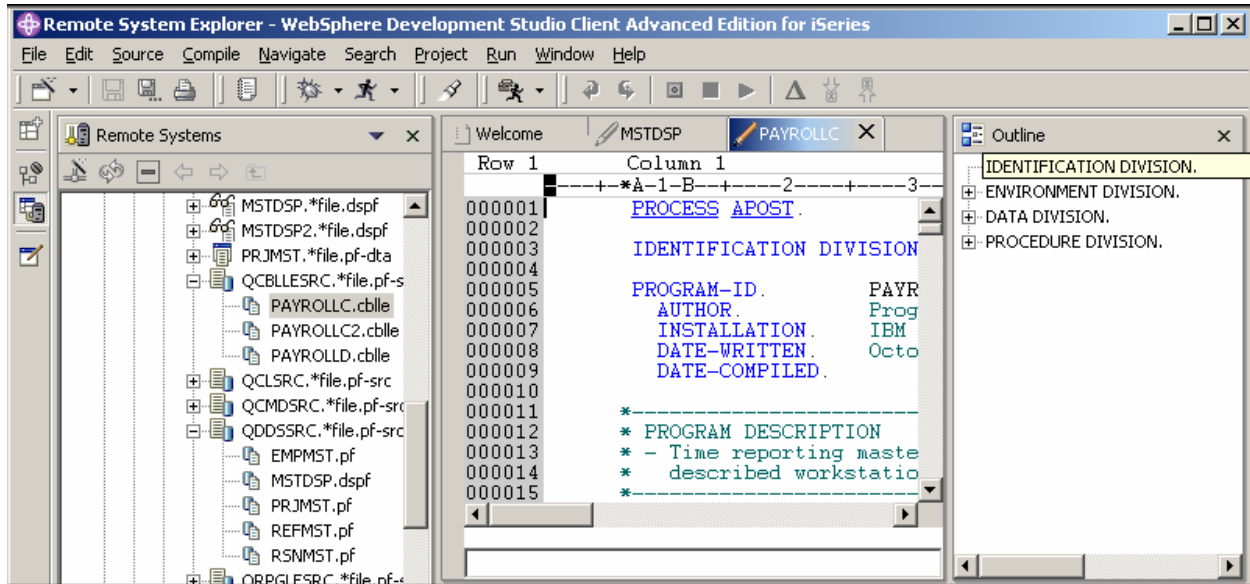
Figure 14: Editor with COBOL source member and Outline view

Displaying an outline of a structured file

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

You can see the Outline view contains your source program in a tree view without the lines

containing logic.
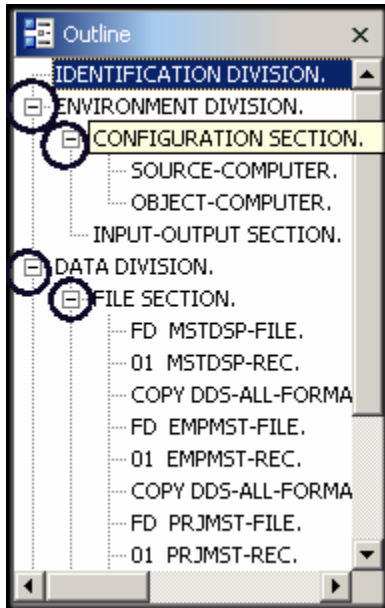
Figure 15: Outline view with file and record format expanded

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

1. Expand **ENVIRONMENT DIVISION**

2. Expand the **CONFIGURATION SECTION**

3. Expand the **DATA DIVISION**

4. Expand the **FILE SECTION**

Double click on any of the icons in the Outline view. This will position the source editor accordingly.

Now if you want to switch back to a different member loaded in the source editor just click on its tab and the Source Editor window with the correct member comes into focus.
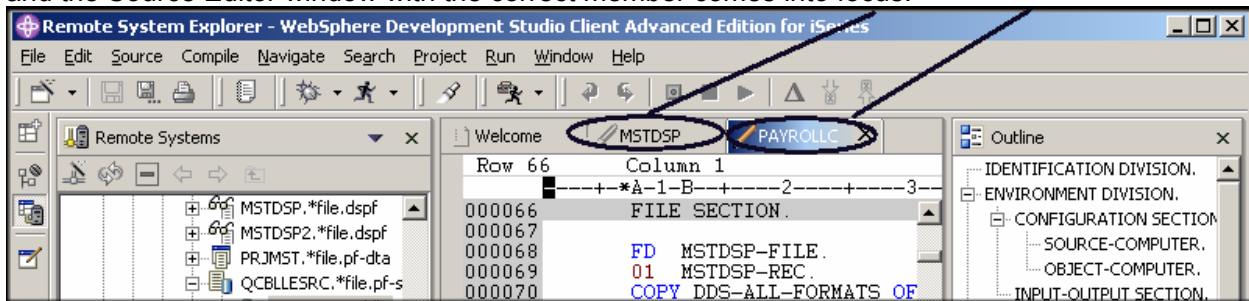
Figure 16: Click the tab to give focus to the edit session

Now you want to switch back to the DDS source member.

5. Click the **MSTDSP** tab to get the **MSTDSP** Editor window in focus.

6. Click the **PAYROLLC** tab to get the **PAYROLLC** Editor window in focus for the next exercise.

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

Checkpoint

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. The first time you connect to an iSeries system, you need to define a:
    A. Profile
    B. Filter
    C. Filter pool
    D. Connection
    E. All of the above
3. Profiles:
    A. Help partition data when you have a lot of connections or filter pools
    B. Include all connections, filters, and filter pools
    C. Group connections
    D. Share connections
    E. Keep connections private
    F. All of the above
4. There are several type of profiles:
    A. Team
    B. Personal
    C. Both
5. A team profile is used to share resources and information with other people. (T, F)
6. Subsystems include:
    A. iSeries Objects
    B. iSeries Jobs
    C. IFS Files
    D. iSeries Commands
    E. All of the above
7. iSeries objects include:
    A. Work with libraries
    B. Work with objects
    C. Work with members
    D. Library list
    E. All of the above
8. The iSeries Table view is used to:
    A. Display a list of items, for example, members or objects in a table format similar to PDM
    B. Perform actions against a list of items, such as editing and compiling
    C. Both

9. The lock icon controls the correlation between the Remote Systems view and the iSeries Table view. (T, F)
10. A disabled lock means when you click on an object or library in the Remote Systems view
    A. The associated contents of that item automatically populate the iSeries Table view
    B. Only the Remote Systems view is updated
    C. Only the iSeries Table view is updated
    D. None of the above
11. An enabled lock means when you click on an various items in the Remote Systems view
    A. The associated contents of that item automatically populate the iSeries Table view
    B. The iSeries Table view input does not change
    C. Only the Remote Systems view is updated
    D. Only the iSeries Table view is updated
    E. None of the above
12. You can maximize the Editor window by double clicking on its window heading. You can get back to its original size, by double-clicking on the heading again (T, F).
13. The outline view:
    A. Displays an outline of a structured file currently open in the editor
    B. Lists structural elements
    C. Contents and toolbar are editor specific
    D. All of the above
14. _____ views are hidden views, which can be quickly made visible. They work identical to normal views, the only difference being that when hidden they do not take up screen space on your workbench window.
    A. Remote Systems
    B. Navigator
    C. Outline
    D. Fast

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 attributes of the connection. Then use the iSeries objects subsystem to list the libraries in your library list. Use the iSeries Tables view to see the objects in your library. Subset objects in the iSeries Table view. Open the Remote Systems LPEX Editor from the iSeries Table view. Use the Development Studio Client for iSeries online help to assist you in these tasks.

What you just did

In this exercise you were introduced to profiles, subsystems, and the iSeries Tables view. You defined connection information for an iSeries server, selected the iSeries Objects subsystem and viewed the libraries in the library list. You then expanded the current library and looked at the iSeries Table view of QDDSSRC. You locked and unlocked the iSeries Table view. You then opened several source members with the Remote System LPEX Editor. You maximized the Editor window, opened an Outline view and switched from the Outline view to the member ready for edit.

In the next exercise you will continue to work with the source members you just opened and learn how to simplify your editing tasks with the Remote Systems LPEX Editor. Remember you are editing

right from the Remote Systems view! And keep in mind you can use SEU like format line rules, SEU specification prompting and SEU style commands in the Remote Systems LPEX Editor.

Exercise 4: Editing ILE COBOL

In this exercise you edit ILE COBOL source and using some of the Remote Systems LPEX Editor's language features right from the Remote Systems view.

At the end of the exercise, you should be able to:
- Describe the features of the Remote Systems LPEX Editor
- Invoke SEU commands in the prefix area
- Undo and redo edit changes
- Display COBOL types of lines
- Filter source
- Invoke program language help
- Find a string in multiple files with the search utility
- Compare two files with the compare utility
- Identify lines with syntax errors
- Correct syntax errors
- Switch automatic syntax checking off

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

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

- Cut, copy, and paste

- Block marking of lines, characters, or rectangles with copy, move, and delete operations

- Powerful find and replace function

- Unlimited undo and redo

- Automatic backup and recovery

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

- Token highlighting where different language constructs are highlighted using different colors and fonts to help identify them in a program

- SEU-like format-line rulers to show the purpose of each column for column-sensitive languages like RPG and DDS. These rulers can automatically update themselves to reflect the current specification.

- SEU-like specification prompting for RPG and DDS

- Sequence numbers, which allow SEU-style commands in the prefix area

- Intelligent tabbing between columns for column-sensitive languages

- Automatic uppercasing for languages that expect uppercase

- For column-sensitive languages there is a command that simplifies text insertions and deletions

- On-line language reference help

Now let's take a minute to try a few of these features.

Working with the COBOL source member PAYROLLC, which should be already open, try some of these features by following the instructions below. You might want to maximize the Editor window during this exercise.

Entering SEU Commands

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

- ispf
- seu
- xedit

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

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

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.

Requesting Undo and Redo operations

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

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

To undo and redo edit changes:

1. Select **Edit > Undo** from the workbench menu bar.

   Notice that the 5 new lines disappear.

2. Do another undo action by pressing **Ctrl+Z**.

   Notice that the deleted lines reappear.

3. Select **Edit > Redo** from the workbench menu bar

   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. To do this:

4. Select **File > Close** option on the workbench menu bar

   `A Save Resources window will appear asking you to save the latest changes.`

5. Click **No**.

6. Go back to the workbench to the Remote System Explorer perspective and reload the

   PAYROLLC member in the QCBLLESRC file.

Invoking Language-Sensitive Help

Inside the editor, there is cursor-sensitive language-reference help available. This help is invaluable if you cannot remember the syntax of certain statements. This help is available from the LPEX Editor window.

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

To access language sensitive help:

1. Position the cursor over the word `MOVE` in line **231** of the ILE COBOL source.

2. Press **F1**.

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

Figure 17: Help for MOVE and synchronize button

3. Click the **Synchronize Navigation** icon on the Help window toolbar

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

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

5. Minimize the Help window.

6. Select **Help > Help contents** on the workbench menu bar.
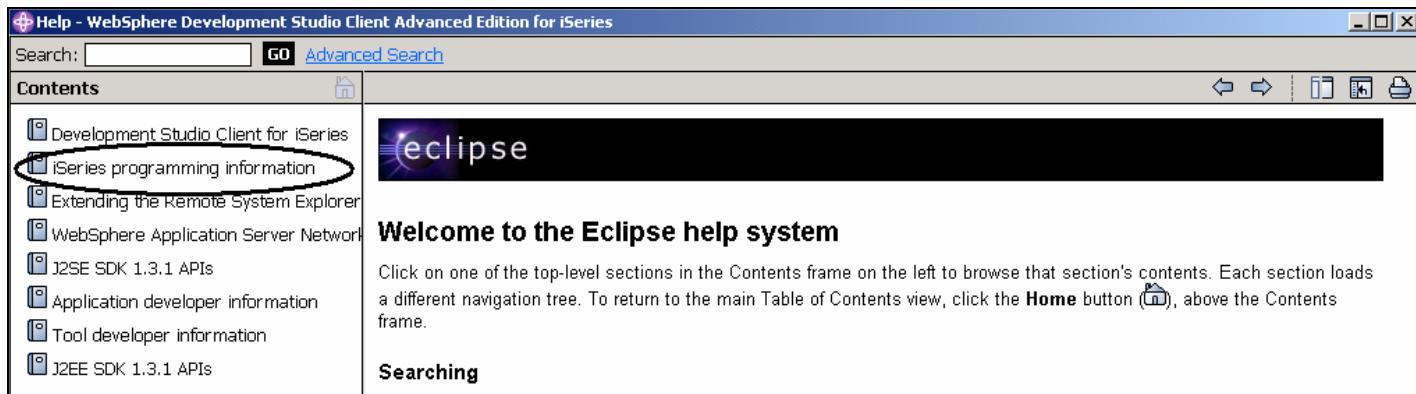
Figure 18: Main Help window

7. Select i**Series programming information** from the topic list in the left pane
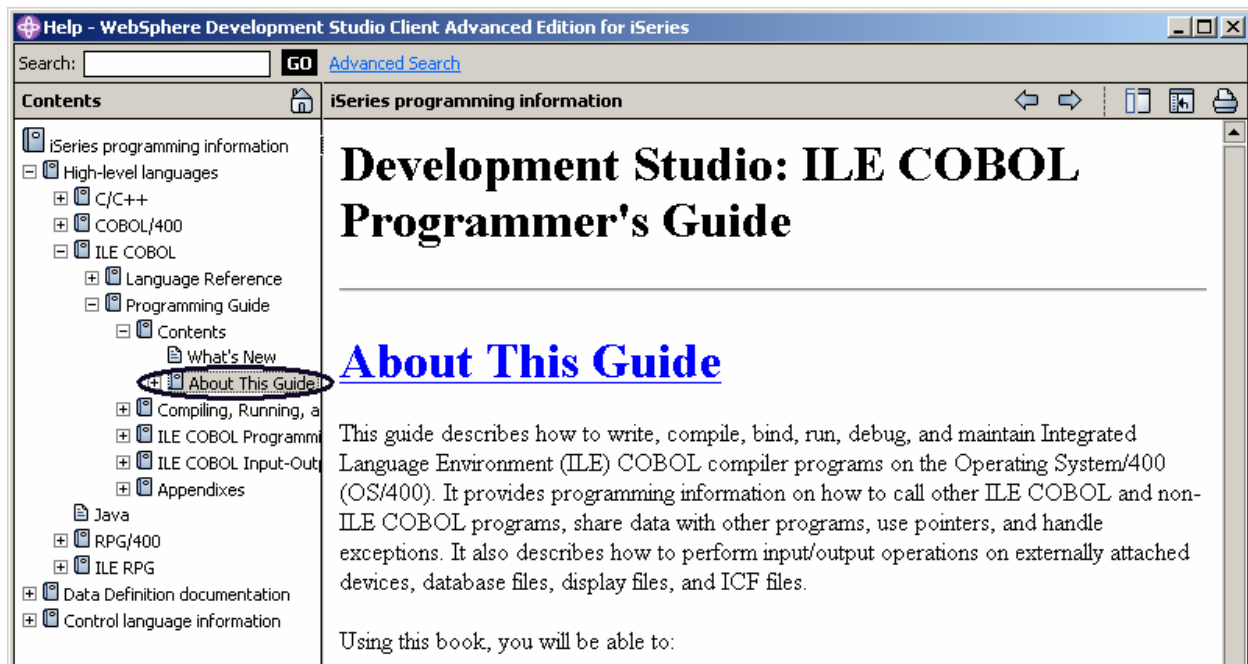
Figure 19: Finding the COBOL manuals

8. Expand **High level languages**

9. Expand **ILE COBOL**

10. Expand **Programming Guide**

Here you find the table of contents and can select your topic of interest.
Having the latest version of the manuals at your fingertips makes it easier to find programming information.

Finding and Replacing text

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

To find and replace text in the Payroll Editor window:

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

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

The Find /Replace window appears at the bottom of the Editor window.
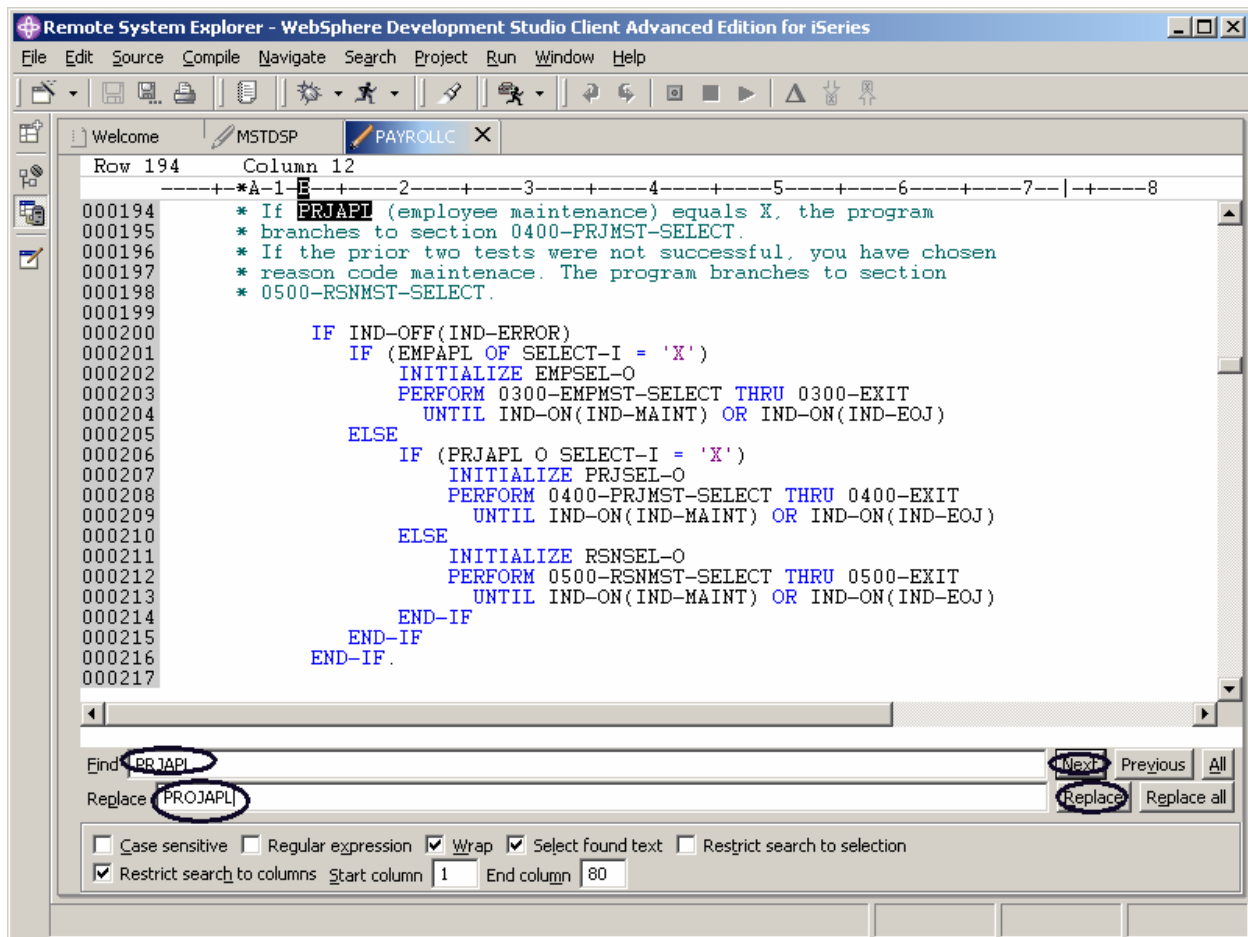
Figure 20: Find/Replace window

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

3. In the **Find** field, enter PRJAPL.

4. Press the tab key to move to the **Replace** field and type PROJAPL.

5. Uncheck the **Case sensitive** check box.

6. Click **Next** (or Press **Ctrl+F)**

    You will notice that the Editor moves the active line, to line 194, which contains the first occurrence of PRJAPL in the file.

7. Click **Replace** to replace PRJAPL with PROJAPL then click **Next** (or Press **Ctrl+F)** to go to the next location of PRJAPL in the file.

8. Click **Replace all** to replace all occurrences of PRJAPL with PROJAPL.

9. Double-click member **PAYROLLC** tab in the Editor window to return the PAYROLLC Editor window to its original size.

10. Close the PAYROLLC Editor window.

    A window will appear asking you to save the latest changes.

11. Click the **No** button.

12. From the Remote Systems view, double-click **PAYROLLC** to reload the COBOL original source.

Filtering Lines by string

The LPEX 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. Select the **MSTDSP** in the Edit window

2. Right-click in the Editor window and select **Filter Selection** from the pop-up menu
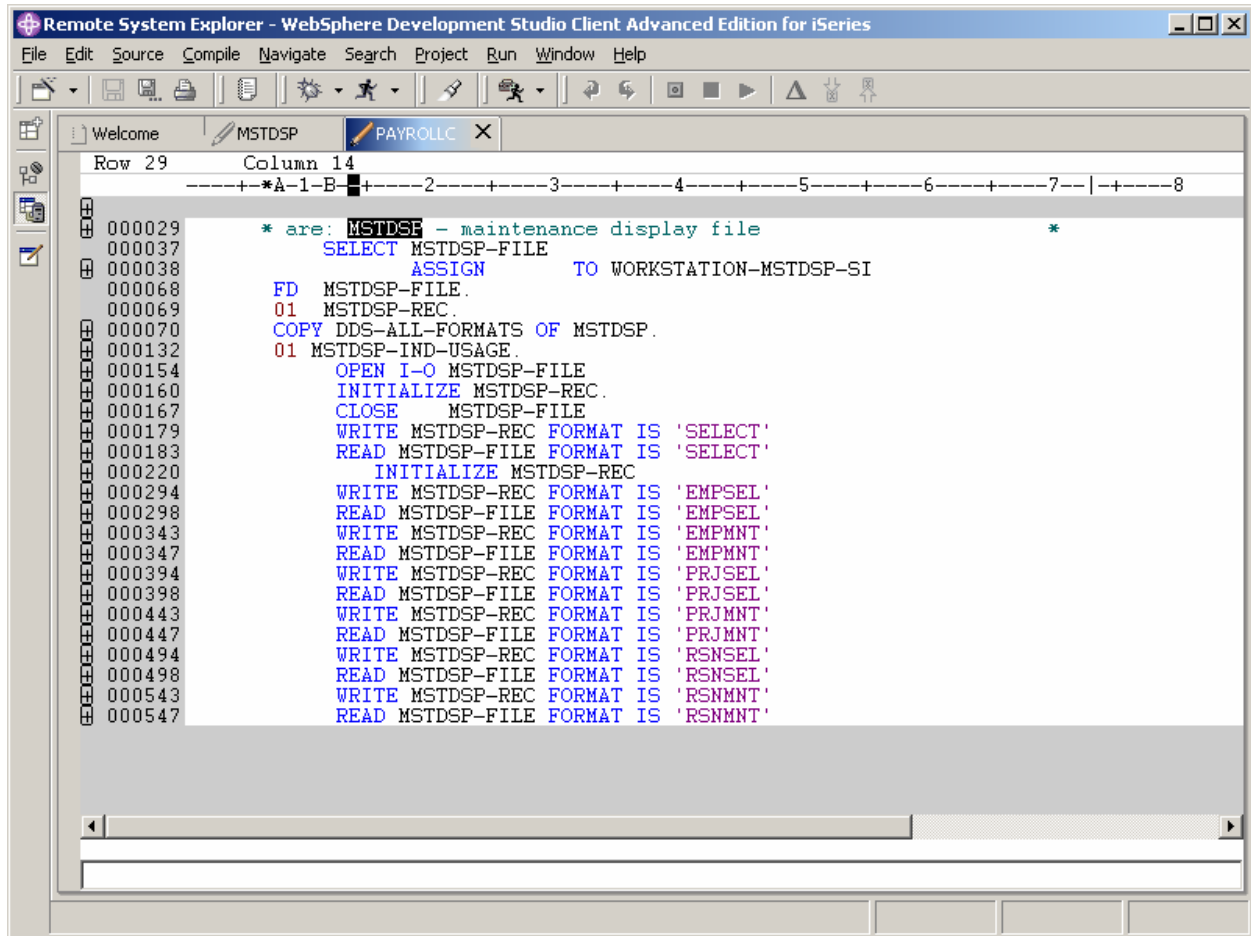
Figure 21: Editor window showing all lines with MSTDSP

3. Move the cursor down a couple of lines, to line **294**
4. Click the plus beside the line to expand this section

Figure 22: Section expanded
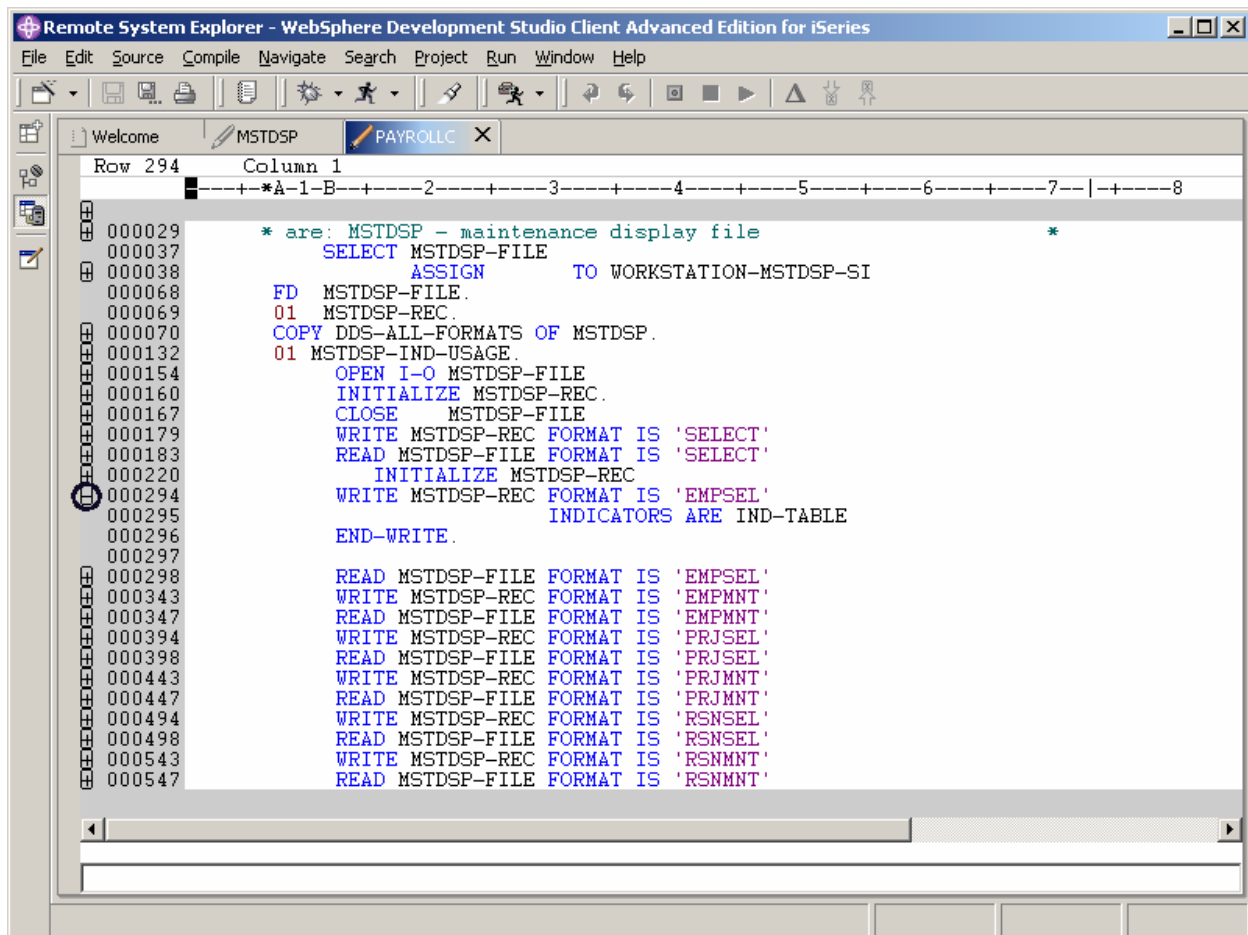
Now you want to show the entire source again.

5. Right-click in the Editor window and select **Show all** from the pop-up 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.

Filter lines based on type

To help you navigate quickly through your ILE COBOL source the Editor allows you to filter lines based on the line type. Imagine you want to see where all the divisions are defined in your source:
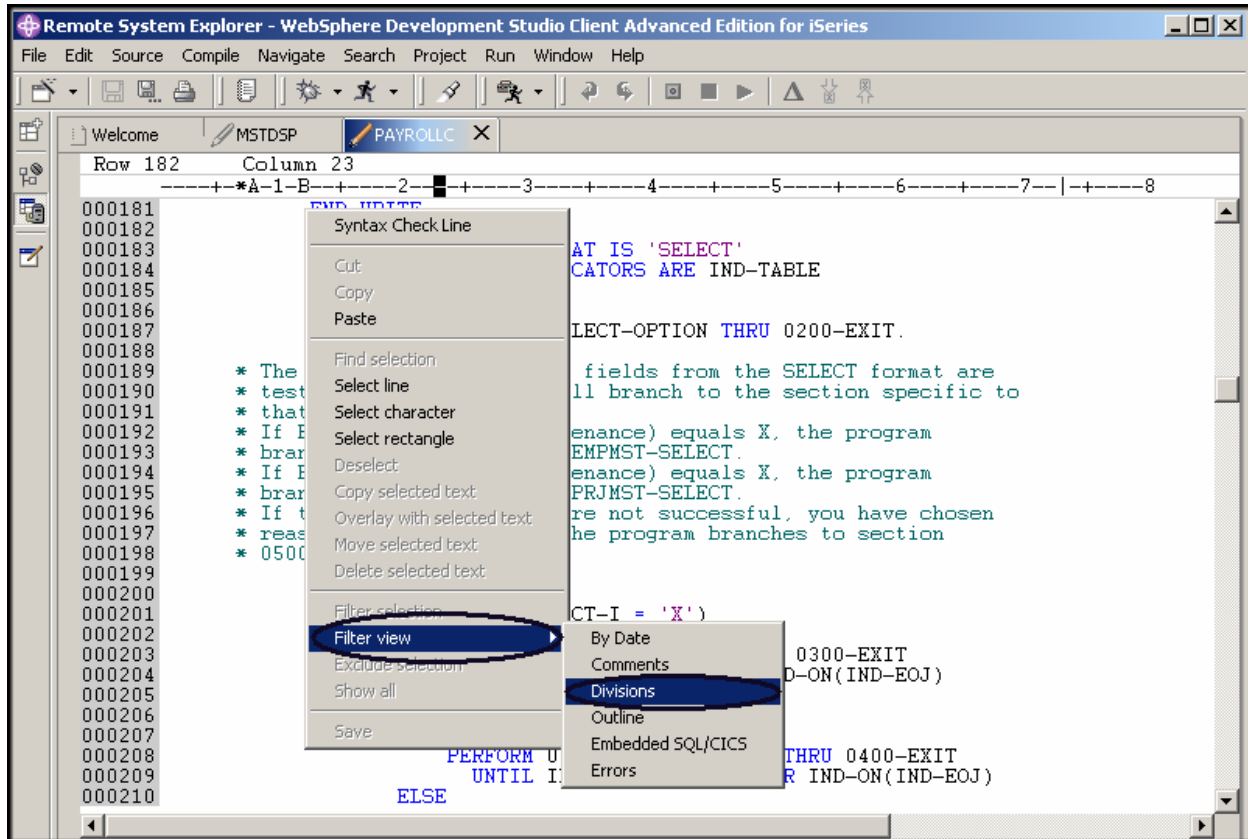
Figure 23: Editor pop-up menu with Filter sub menu

1. Right click in the **Editor window**

2. Select **Filter view > Divisions**.

All divisions are displayed allowing you to move quickly and easily to the area in your file where the desired division is.
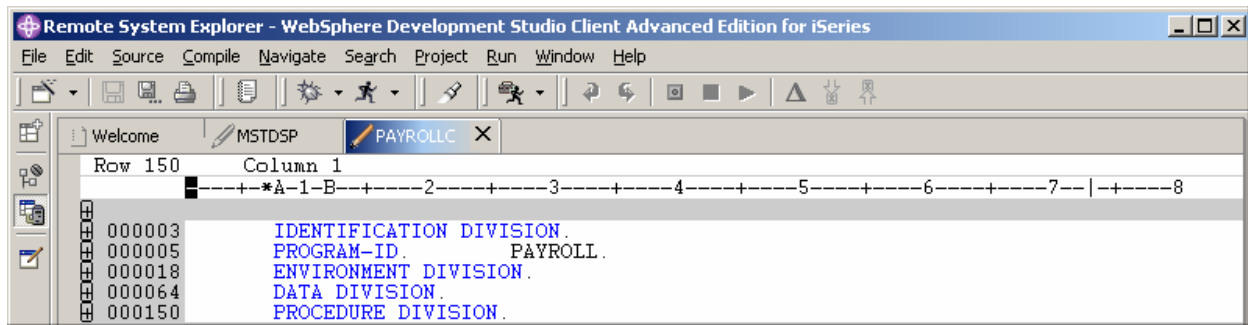
Figure 24: Division filter in action

3. Move your cursor to the line with the procedure division. (line 150)

4. Click the plus sign beside the declaration to show all lines in this division

Now you can work with the source inside this division.

Searching multiple files

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

To search multiple files:

1. Select **Search** from the workbench menu bar.
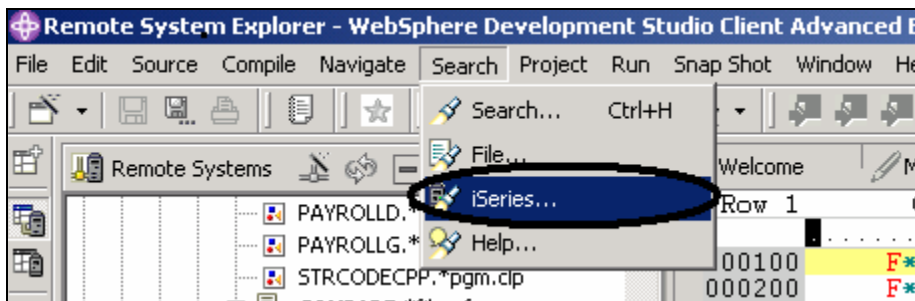
The Search window appears.

2. In the **Search string** field type `ENHRS`
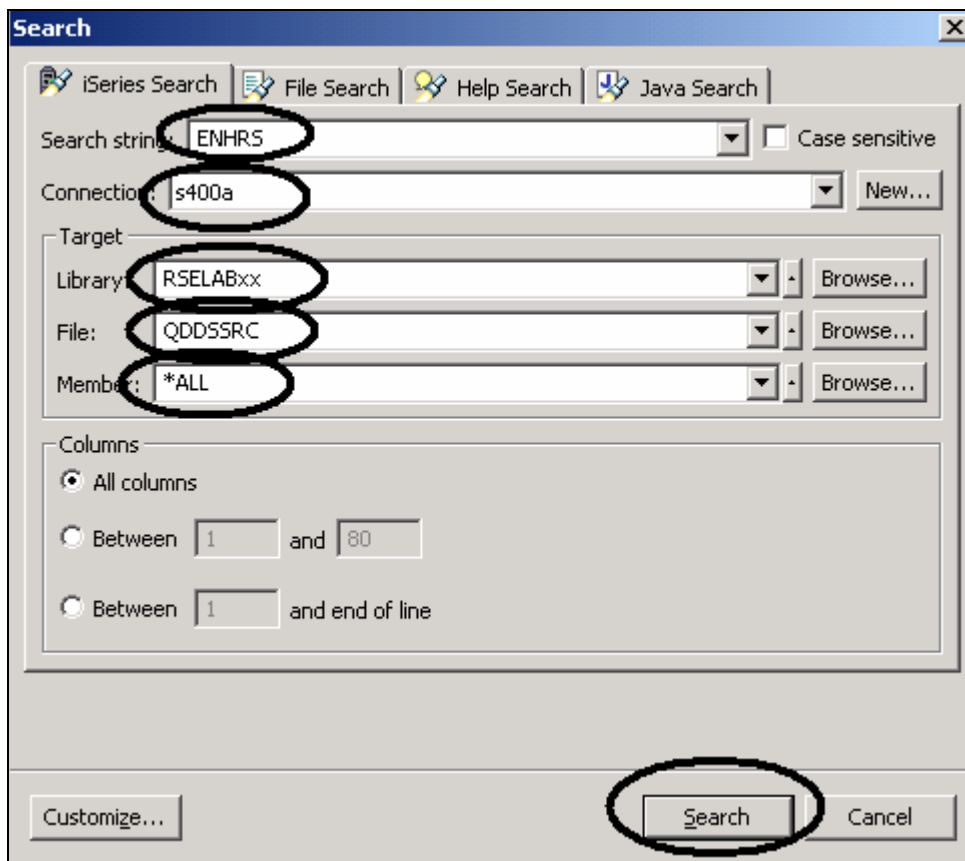
Figure 25: File Search window

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

In the Target area:

3. Enter RSELABxx in the **Library** field.
4. Enter QDDSSRC in the **File** field, to search all members in this source physical file.
5. Enter *ALL in the **member** field
6. Click **Search**.

The Multi-File Search window lists all the lines in all the files that reference ENHRS.
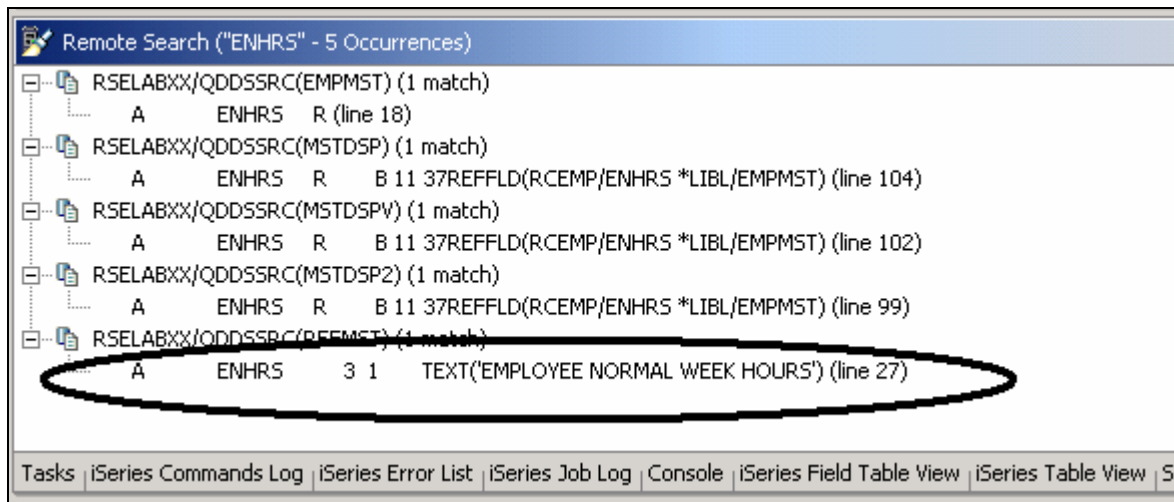
Figure 26: Search result

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. Wow !

Comparing file differences from the Remote Systems view

If your product undergoes many changes, you will find the Compare utility useful. It allows you to compare different versions of a program and find the differences. There are 2 ways in doing the compare: use the Compare utility in the workbench or use the Compare utility in the CODE tool. The compare in the CODE tool is more intuitive but requires you to start the CODE Editor outside of the workbench.

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

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

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

To compare files in the workbench:

1. Select **Window > Preferences** from the workbench menu bar
2. Expand **LPEX Editor** from the tree view in the Preferences window
3. Select **Compare** under LPEX Editor
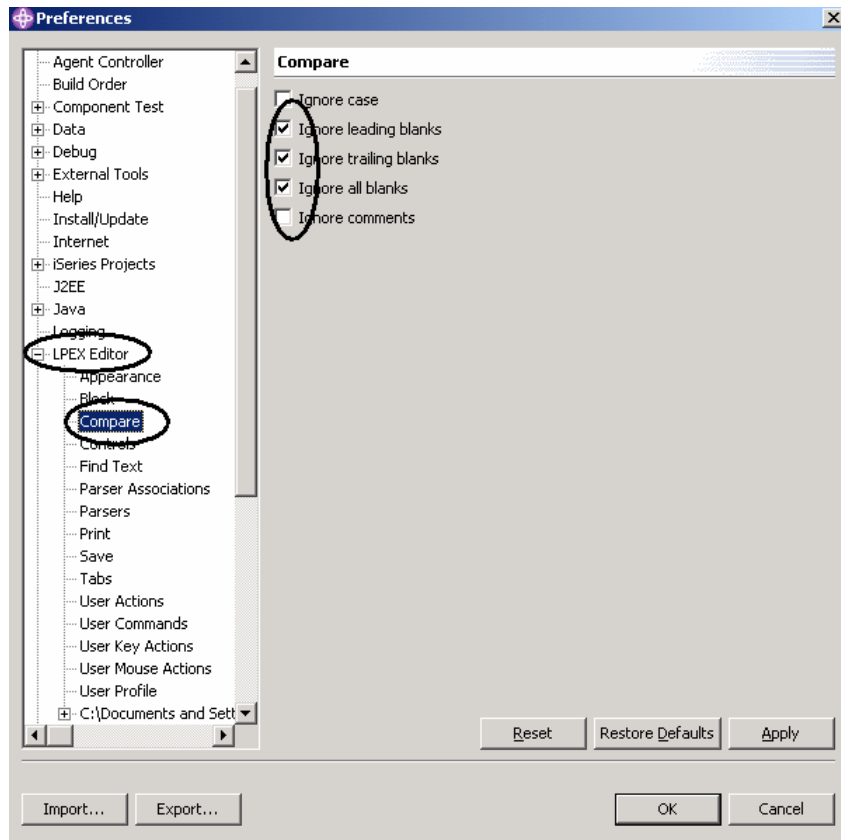
4. In the **Compare** window make sure that the **Ignore blanks** check boxes are selected

5. Click **OK** in the **Preferences** window

Back in the Edit window of the PayrollC member:

6. Make sure all lines show in PayrollC (Right-click and select **Show All** from the pop-up menu).
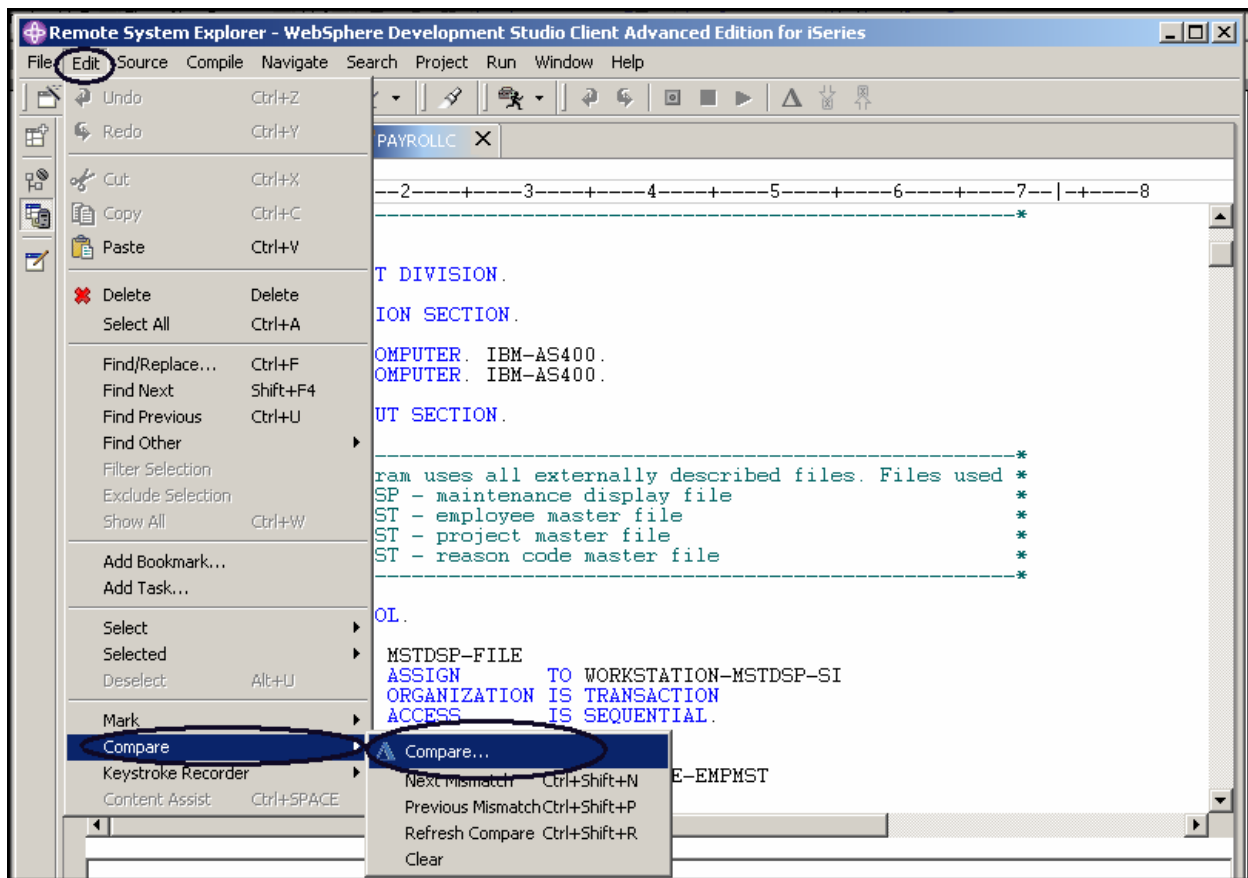
7. Select **Edit > Compare > Compare** from the workbench menu bar
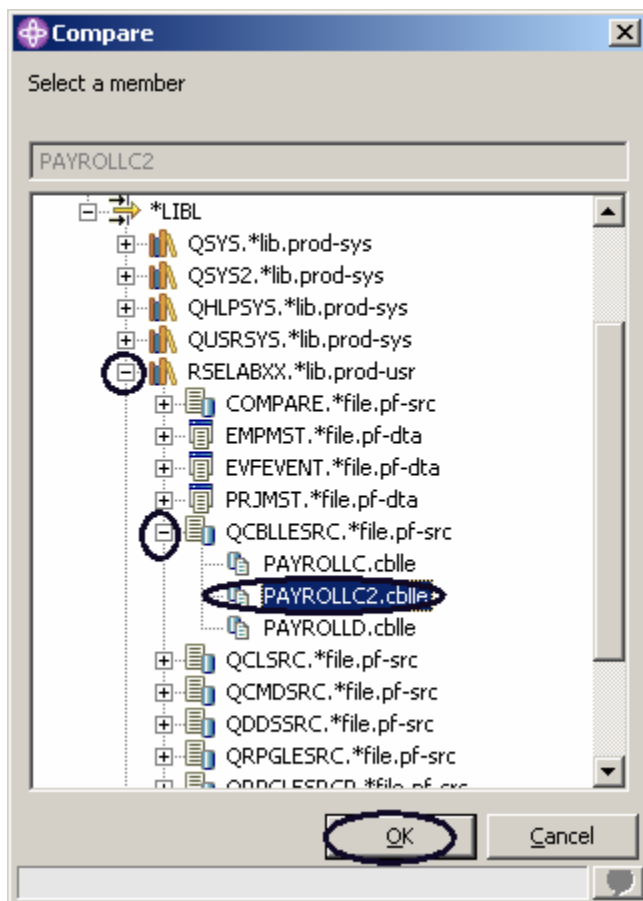
In the Compare window:

8. Expand your connection

9. Expand ***LIBL**

10. Expand **RSELABxx**

11. Expand **QCBLLESRC**

12. Select member **PAYROLLC2**

13. Click **OK** on the Compare window

The editor now will show the differences of these two members PAYROLLC and PAYROLLC2
You can move from mismatch to mismatch by going back to the Compare menu option under the
Edit menu option or by using short cut keys CTRL+Shift+N .

The color of PAYROLLC and PAYROLLC2 will be different so you have an indication where lines
are different from each other.

Next, end the compare session.

14. Select **Edit > Compare > Clear** from the workbench menu bar

Comparing file changes from the CODE Editor (optional)

The CODE tool provides a side by side view of the members being compared. If you prefer this kind 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.
In the Remote systems view:

1. Right-click member **PAYROLLC** in **QCBLLESRC**

2. Select **Open with > CODE Editor** from the pop-up menu

   This opens the CODE Editor window with the PAYROLLC member. It will come up in browse mode since it is locked by the LPEX Editor session.

In the CODE Editor, open the PAYROLLG member:

3. Select **File > Open** option from the CODE Editor menu bar
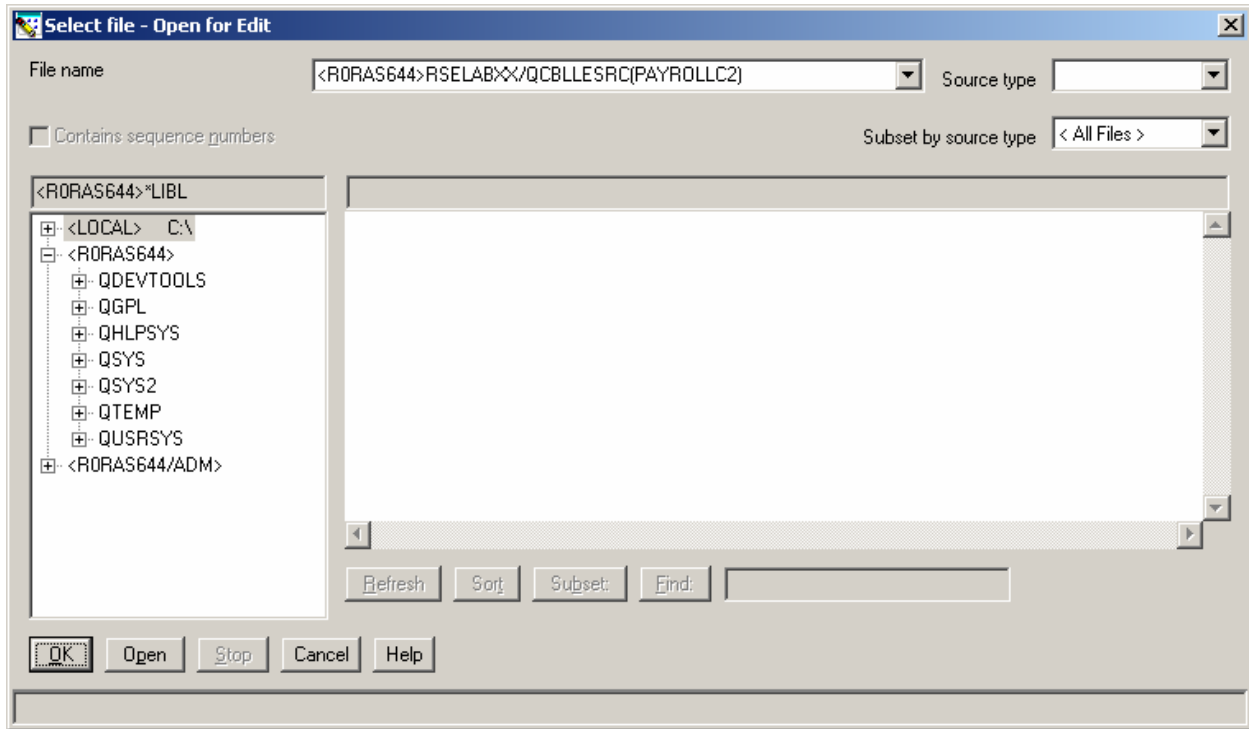
In the Open window:

4. Change PAYROLLC to PAYROLLC2 in the **File name** field.

5. Click **OK** on the Open window.

6. From the **Actions** menu, select **Compare**. The Compare window appears.
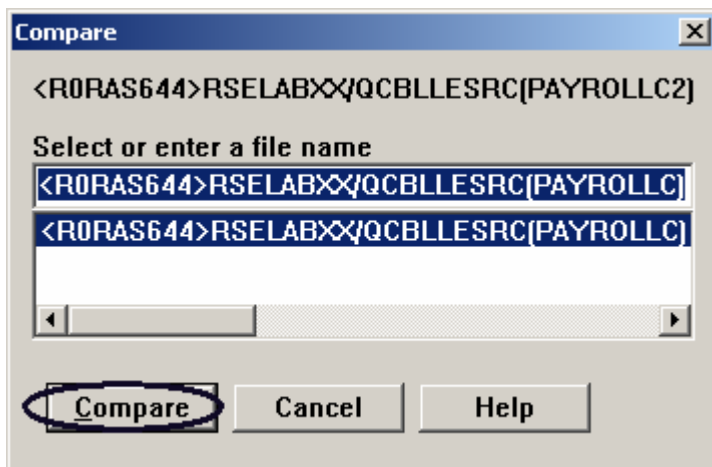
All entries are preloaded

7. Click **Compare**.

The editor now has the PAYROLLC2 member loaded on the left and member PAYROLLC 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.

8. Use the vertical scroll bars to move within the files. As you scroll, you will see where the differences are in the members. COBOL experts will notice that PAYROLLC has some errors in it. We will fix these in a few moments.

9. From the **Compare** menu (which was inserted while performing this action), select **Exit Compare** to go back to the original view.

10. Close the CODE Editor and continue working in the workbench.

Checking Syntax

One of the powerful features that the LPEX Editor shares with SEU is its ability to syntax check your source. Syntax checking can be done either when the cursor leaves each line of source or all at once on either the currently selected source or on the entire file.
Now create a syntax error and watch for the prompt to correct it:

1. In the PAYROLLC Editor window select **Source > Syntax Check All**.

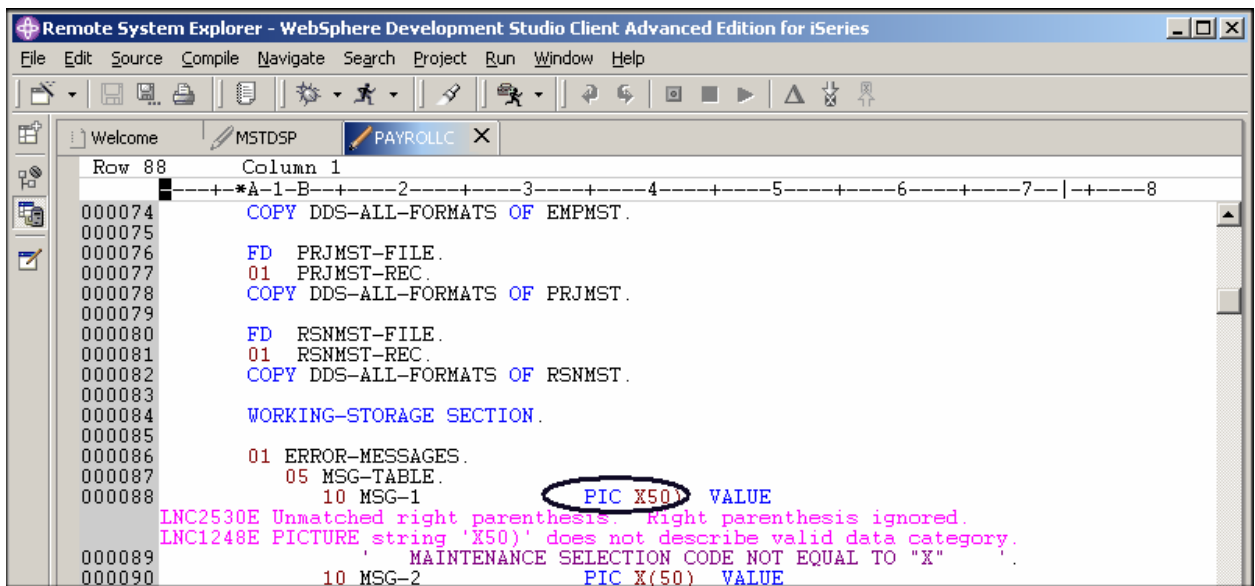   An error message appears to draw attention to the error.

Figure 27: Editor window with syntax error

2. Move the **cursor** onto the pink error message.
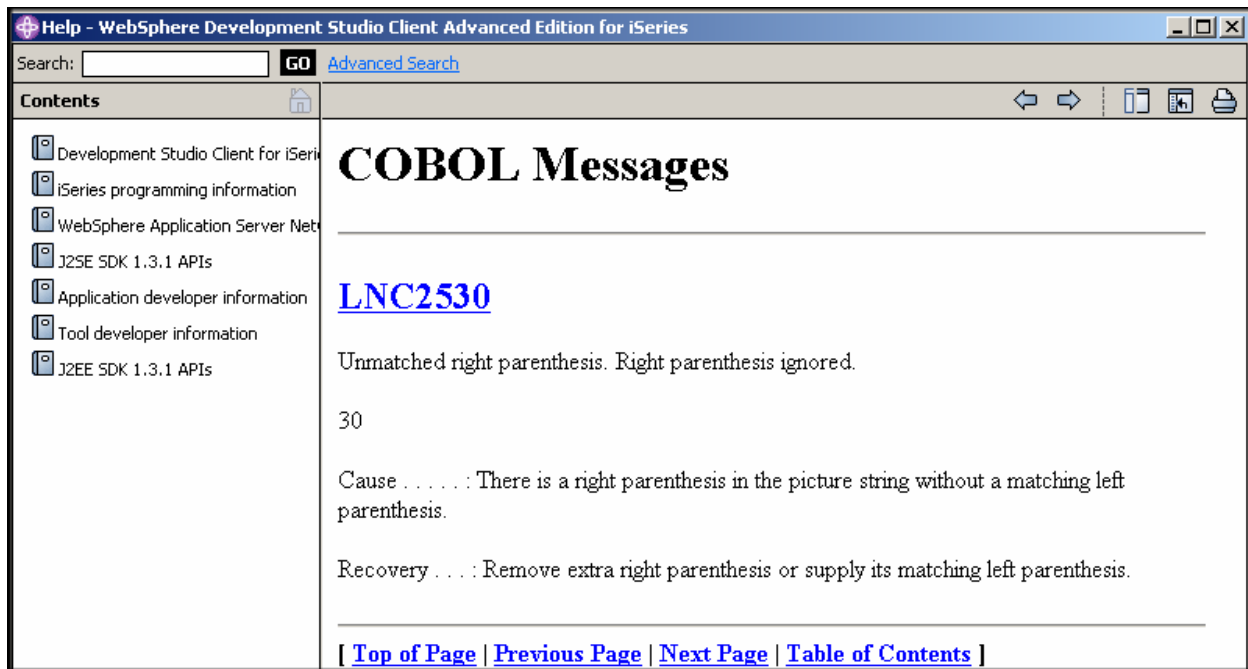
3. Press **F1**.

Figure 28: Second level help for syntax error

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

4. Minimize the Help window.

5. Add the required left parenthesis to correct the error.

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

The error message is automatically removed from the editor.

Tip: You can toggle automatic syntax checking by using the **Windows > Preferences** option on the workbench menu bar and **Remote Systems** then **iSeries** then **LPEX Editor Parsers** and then **ILE COBOL** in the Preferences tree view.

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

Checkpoint

1. The LPEX Editor has predefined settings, but also has an associated preference page containing settings that you can define. (T, F)
2. LPEX Editor preferences are set in the:
     A. Preferences window
     B. Editor window
     C. New wizard
     D. Remote Systems view
 3.   You can configure the LPEX Editor to adopt the keyboard and command personalities of many popular editors. (T, F)

4. 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
5. 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
6. This help is invaluable if you cannot remember the syntax of certain statements. Name the help. Choose the best answer.
    A. Context sensitive help
    B. Language sensitive help
    C. Display help
    D. Field help
7. To receive language sensitive help, press _____ key in an Edit window. Name the key:
    A. F2
    B. F3
    C. F1
    D. F4
8. 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
9. You use the _____ window to search for an item. Choose the best answer.
    A. Search
    B. Find
    C. Edit
    D. Find and Replace
10. 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
11. 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

12. To help you navigate quickly through your ILE COBOL source the editor allows you to filter lines based on the _____. Choose the best answer.
    A. String
    B. Line type
    C. Line number
    D. All of the above
13. 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
14. 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
15. There are two ways to compare files. They are Compare tool in the workbench and the Compare tool in the CODE Editor. (T, F)
16. Syntax checking can be done either when the cursor leaves each line of source or all at once on either the currently selected source or on the entire file. (T, F)

Practice

Given your experience in this exercise using the Remote Systems LPEX Editor, try these new tasks: Check out how copy, cut and paste your ILE COBOL source. Block mark lines, and try move and delete operations in your source. Filter source by comments. Set auto-uppercase on. Use the Development Studio Client for iSeries online help to assist you in these tasks.

What you just did

In this exercise you opened the Remote Systems LPEX Editor and learned about these editor features: SEU commands, undo and redo operations, language sensitive help, find and replace option, filter lines by string, filter lines by line type, search files, compare files, and syntax checking.

In the next exercise you will verify 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!

Exercise 5: Verify and Compile COBOL

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

At the end of this exercise, you should be able to:
- Describe the purpose of the Program Verifier
- Invoke the Program Verifier
- Use the iSeries Error List to insert messages and to check that errors were fixed
- Use the iSeries Table view in Remote System Explorer to submit iSeries commands
- Use the Compile utility to select compile options and compile ILE COBOL source
- Explain the generated ILE COBOL program called PAYROLLC2

Invoking the Program Verifier

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 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.

To invoke the verifier:

1. Select **Source > Verify** from the workbench menu bar

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

Figure 29: Verifier error list

The error list will show you:

1. The error message itself

2. The severity

3. The line number

4. The source location

5. The connection name

Fixing errors
Next you fix the errors in your source.

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

1. Double-click on the error **LNC1463**
    You are automatically brought back into the Editor window to the line where the error occurred.
    The error is caused by EMPNO not being uniquely defined. Go to line 302 where EMPNO is
    defined.

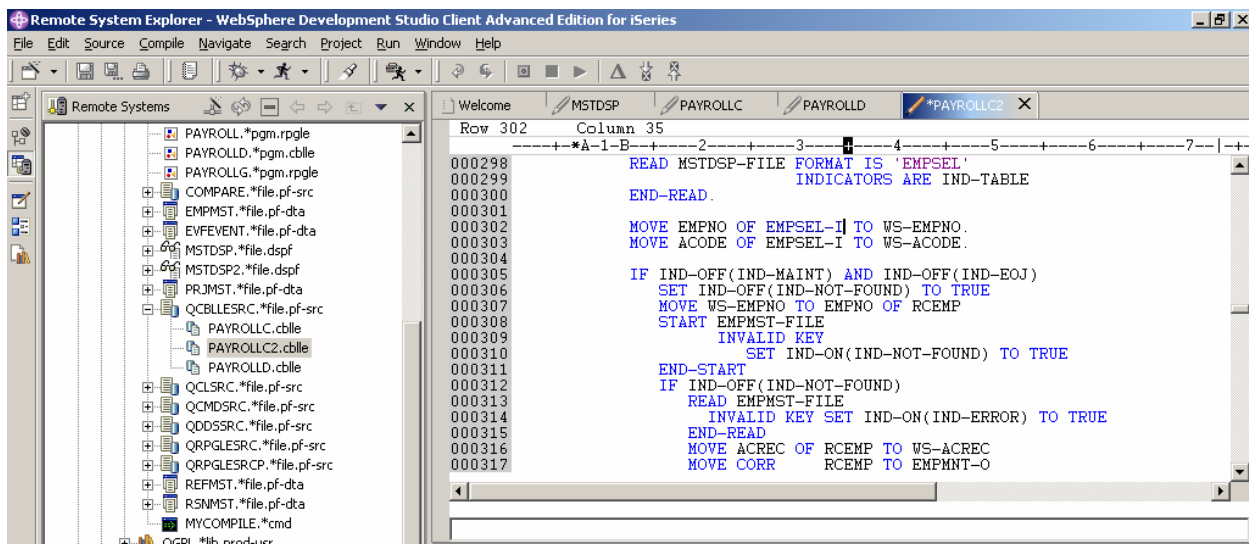   EMPNO should really be EMPNO OF EMPSEL-I. Make the change.

Figure 30: Fixed EMPNO error

The next error is a **LNC1326**
2. Double click on it. Fix it in the editor.
3. **PRCD** should really be **PRCDE**. Make the appropriate change on line 402.
4. Double click on **LNC1329**. This error is because the array was declared with 4 elements. Go to line 606.
5. Change the index of the array from 14 to 4.

6. The next errors are LNC0407. You can ignore these errors as they are severity 20.

You are now done fixing all needed error messages.

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

1. The **File** option on the workbench menu bar

2. The workbench toolbar (click on 🖫)

3. Press **Ctrl+S**.

Changes are uploaded to the iSeries.

4. Verify your source again.

Figure 31: Error list, only severity 10 and 20 are left and can be ignored

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

Invoking a remote compile

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

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

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

To change the preferences to compile in interactive:

1. Select the **Windows > Preferences** from the workbench menu bar
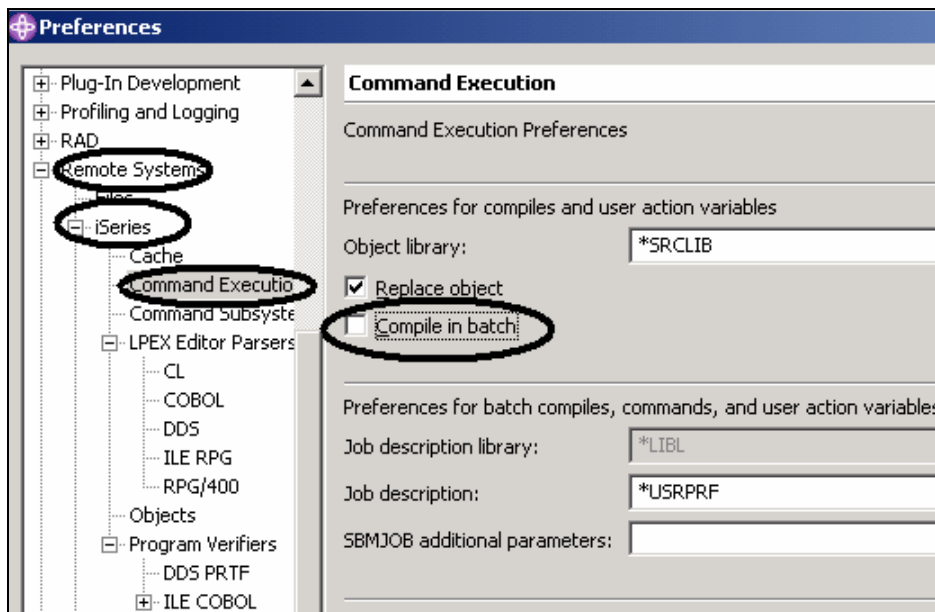
Figure 32: Change Compile command default

2. Expand **Remote Systems**
3. Expand **iSeries**
4. Select **Command Execution**
5. De-select the **Compile in batch** check box
6. Click **OK** to get back to the Remote System Explorer perspective

Starting the compile

Starting a compile is like using any other command in the Remote System Explorer. The pop-up menu for source members includes compile submenus. The compile submenus contain several compile commands for various member types. A compile name is an identifier for a compile command to run on a remote system. For example, the compile label CRTBNDCBL tells the iSeries system to create a bound COBOL program. Compile names are only associated with source member types: members (*MBR) and modules (*MODULE). For this reason, you will only see the options to compile when you right-click these types of elements. There are two ways to compile: prompted and non-prompted. Both compile actions invoke a submenu with a list of compile names. In the Remote Systems view that shows your expanded library RSELABxx:
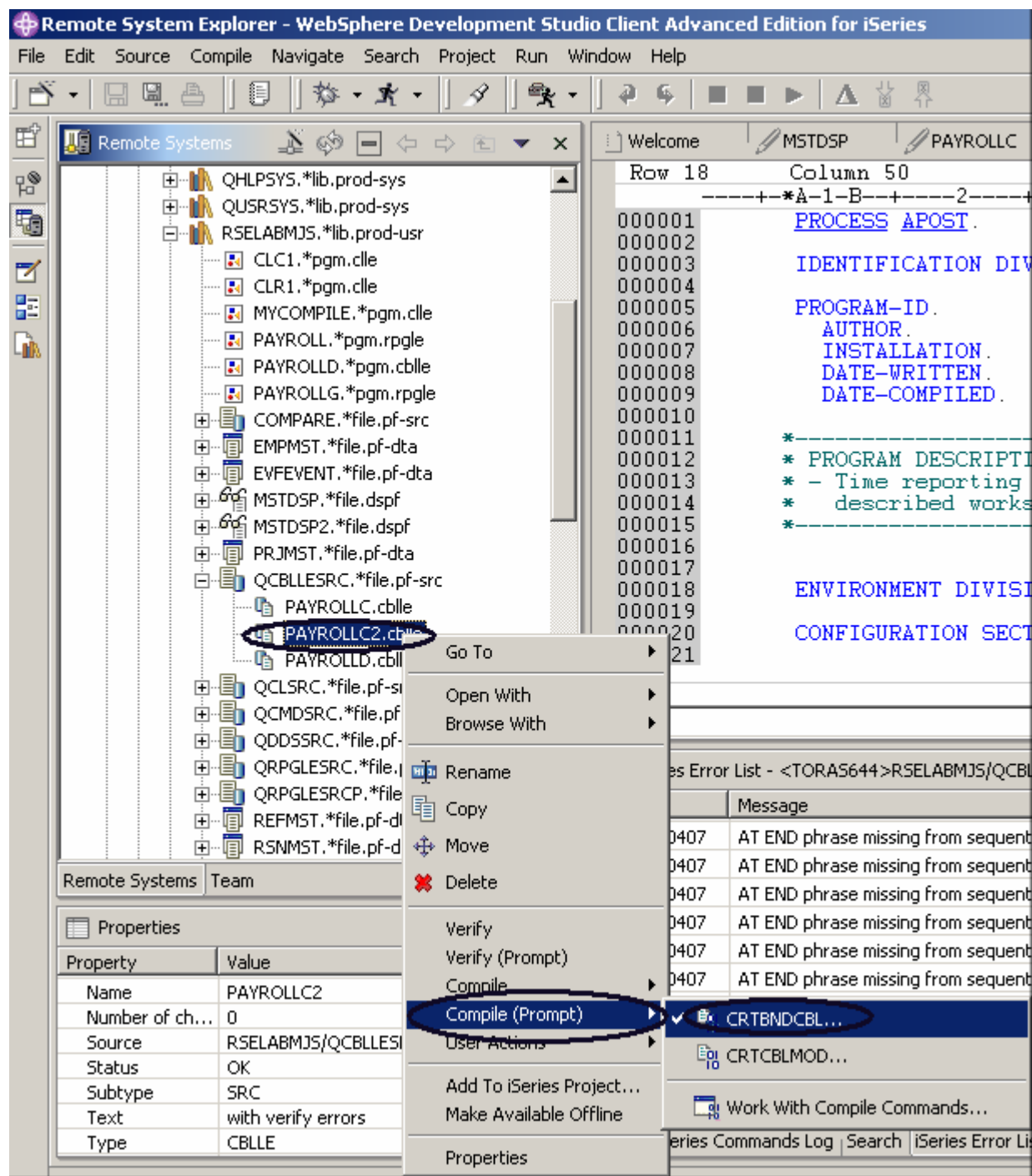
Figure 33: Select member to compile

1. Right-click the **PAYROLLC2** member in QCBLLESRC

2. Select **Compile (Prompt) > CRTBNDCBL** option from the pop-up menu

Compiling Your Source

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.

Figure 34: Prompt for CRTBNDCBL

1. Change the **Debugging views** parameter to ***ALL**.

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

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

A window box will show that the compile started.

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 informational messages. Make sure you have Information Messages selected as a type of message to show. You may have unchecked this selection in a previous exercise.

If you are not sure that the compile was successful:
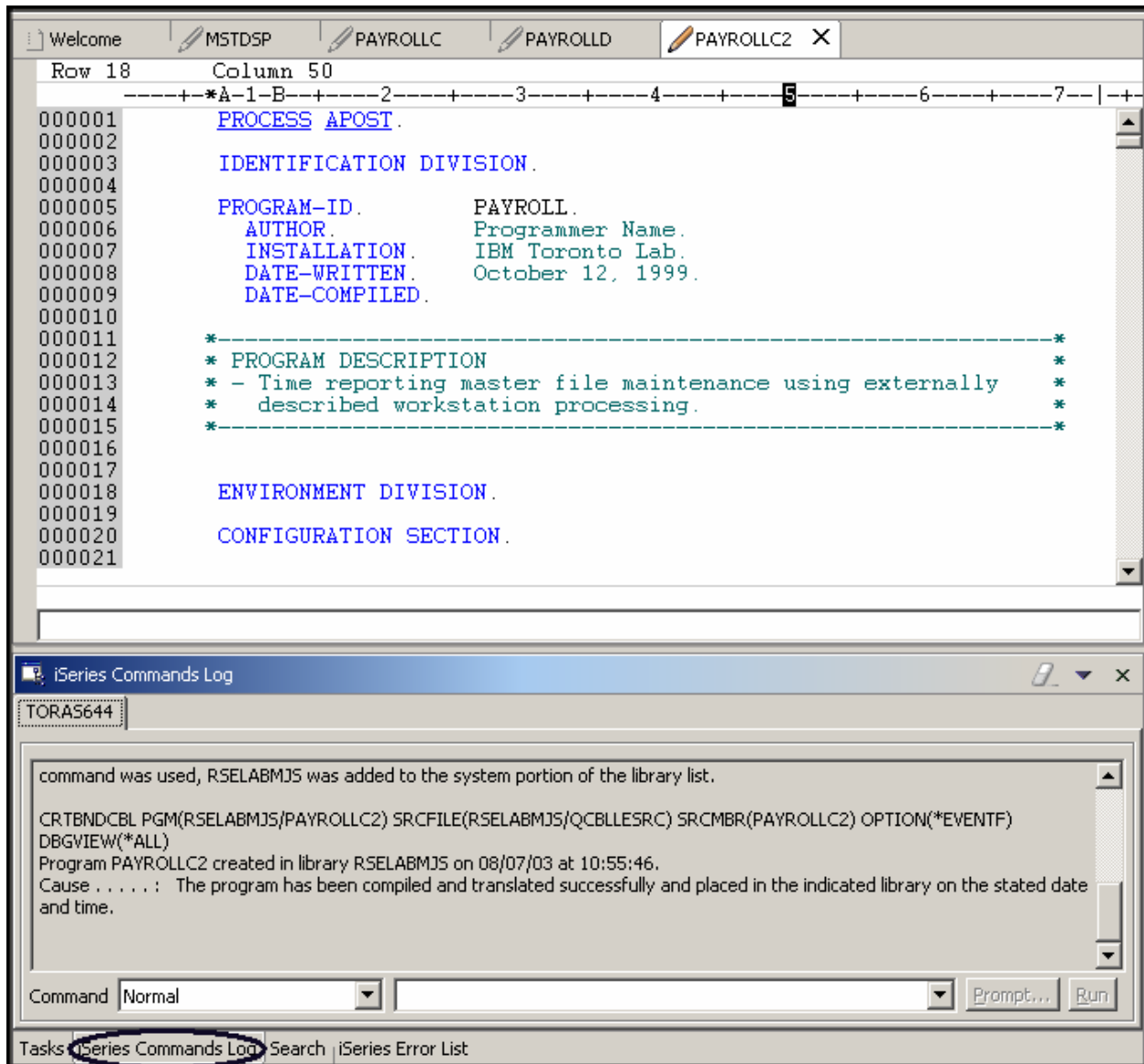
Figure 35: iSeries Commands Log

3. 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 that have been returned for each command.

Submitting iSeries Commands in the iSeries Table view

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

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

To change your library list:
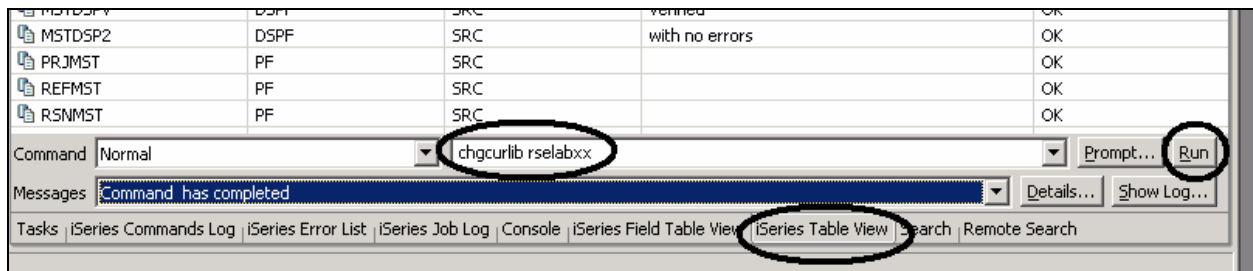
Figure 36: iSeries Table view with command entry

1. Select the **iSeries Table View** tab from the views at the bottom of the workbench

2. Enter in the **Command** field: CHGCURLIB RSELABxx

3. Click **Run**.

   If you haven't used the iSeries Table view to show iSeries objects in this view you will get this error message because the table view is not linked to an active connection.
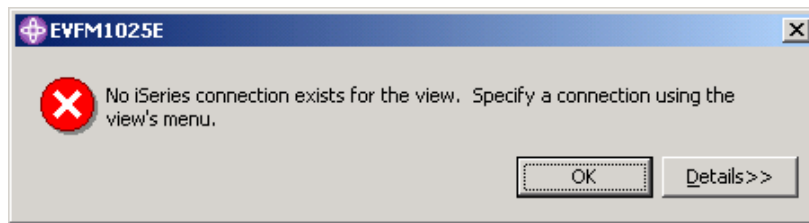
Figure 37: Error message when using iSeries Table view without active connection

If you get this message, click **OK** and go to the Remote Systems view
 4.   Right-click QCBLLESRC
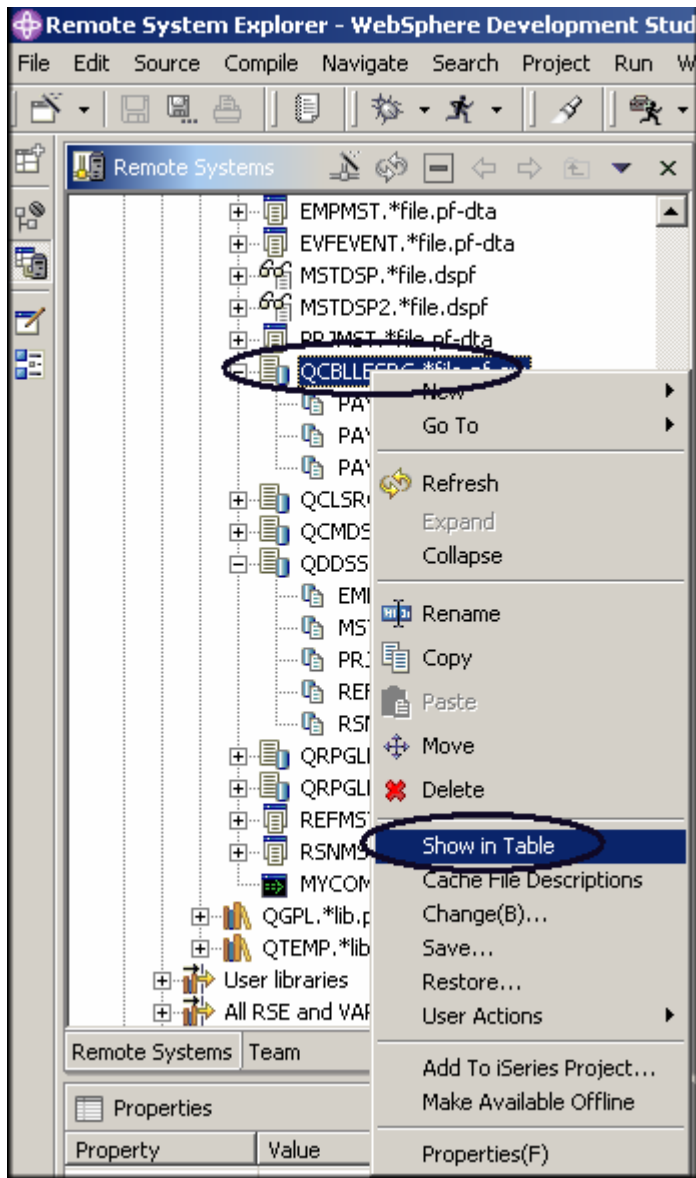
Figure 38: Select Table view to connect it to iSeries

5. Select the **Show in Table** option from the pop-up menu
    The table view is now populated with the member in the selected source file.

6. 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.

Name	Type	Attribute	Text	Status
PAYROLLC	CBLLE	SRC	with syntax & verify errors	OK
PAYROLLC2	CBLLE	SRC	with verify errors	OK
PAYROLLD	CBLLE	SRC	no errors	OK
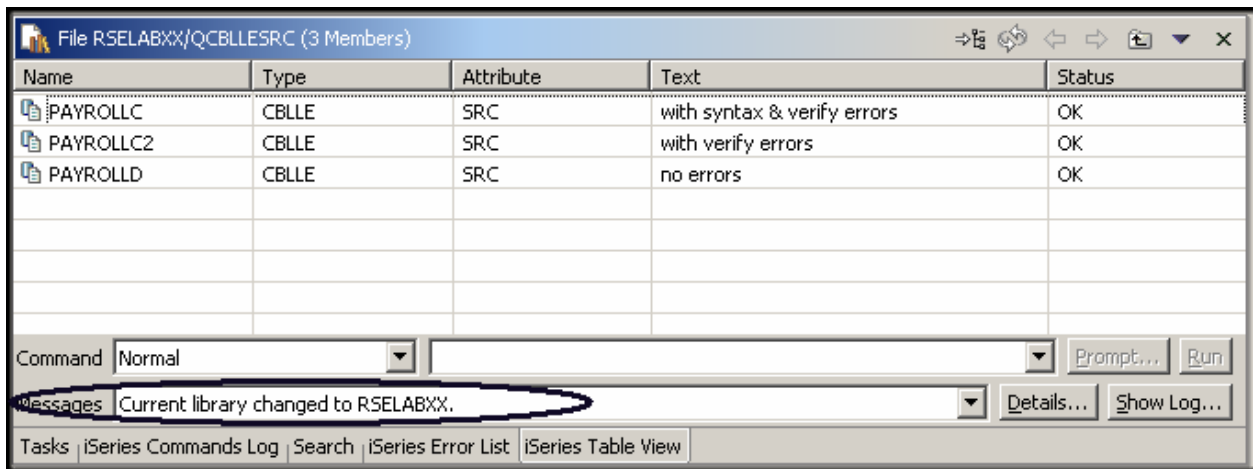
Figure 39: Completion message for command

Tip: You can also connect to other than iSeries systems with the Remote System Explorer and

Launch commands for these systems as well, for example, your local system, or LINUX.

Running Commands and Programs

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

1. In the Remote System Explorer communications server job
2. In a batch job
3. In an interactive 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, so you need to first run a STRRSESVR <connectionName> command to associate the emulator with a particular connection in the Remote System Explorer communications server.

Starting an interactive connection

To start an interactive connection:
1. Start a 5250-emulation session.

2. Sign on to the iSeries with your User ID and password

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

   session.

```
Session A - [24 x 80]                                                  _ |□| ×|
File  Edit  View  Communication  Actions  Window  Help

 MAIN                          OS/400 Main Menu
                                                   System:    TORAS07M

 Select one of the following:


       1. User tasks
       2. Office tasks
       3. General system tasks
       4. Files, libraries, and folders
       5. Programming
       6. Communications
       7. Define or change the system
       8. Problem handling
       9. Display a menu
      10. Information Assistant options
      11. Client Access/400 tasks


      90. Sign off


 Selection or command
 ===> STRRSESVR NAME(s400a) RMTLOCNAME(*RESOLVE)


 F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel    F13=Information Assistant
 F23=Set initial menu


MA     a                                                              20/007
 Connected to remote server/host toras07m using port 23
```

Figure 40: Start Remote System Explorer interactive connection

3. In the emulator, enter the command

    `STRRSESVR <connectionName> RMTLOCNAME(*RESOLVE)`

    The `connectionName` parameter is the name of your connection defined in the Remote

    Systems view. This associates the interactive job with the Remote System Explorer

    communications server. The ***RESOLVE** keyword will get the IP address of your workstation

    and with this information the Remote System Explorer server will communicate with the Remote

    System Explorer daemon that runs on your workstation.

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

You should see a screen as shown in Figure 41.
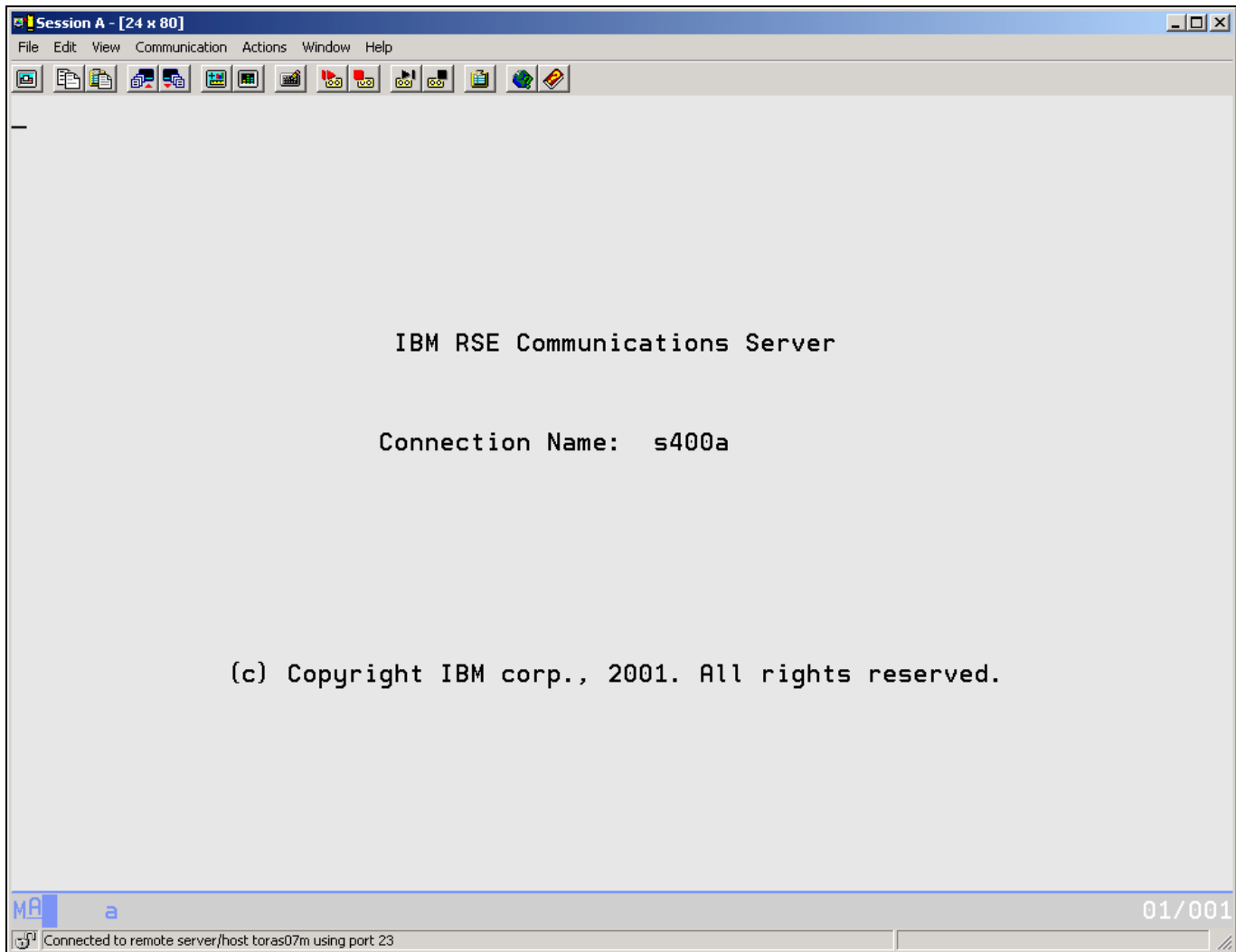
Figure 41: Remote System Explorer connection screen

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

Running the PAYROLLC2 Program

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

Return to the workbench, and in the Remote Systems view:
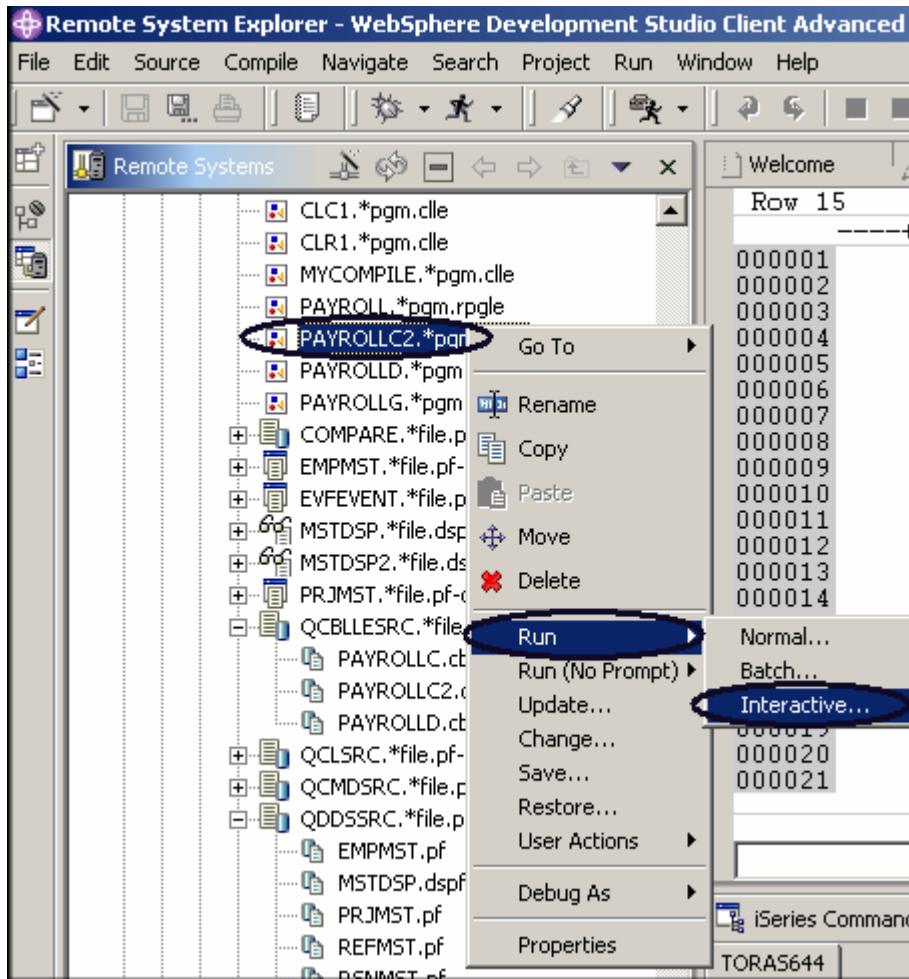
Figure 42: Run program in interactive job

1. Locate the **PAYROLLC2** program that you created

2. Right-click the **PAYROLLC2** program in the Remote Systems view

3. Select **Run As > Interactive** from the pop-up menu

4. Switch to your 5250-emulation session.

You will see the **Start Menu** of the payroll program.
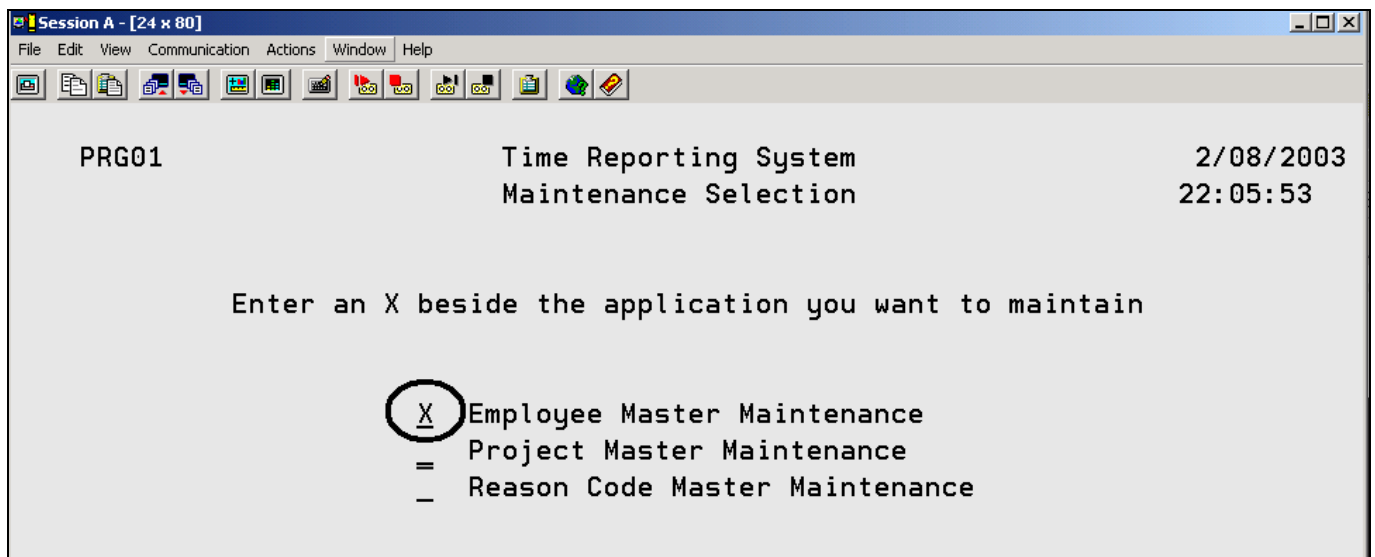
Figure 43: Payroll Start Menu

5. Place an 'x' beside **Employee Master Maintenance**.

6. Press **Enter**.

7. Type **123** for the Employee Number.

8. Type **A** for the Action Code to add employee **123**.

9. Press **Enter**.

10. Type any information you like about the employee.

11. Press **Enter**.

12. Play in the application as much as you like.

13. Press **F3** to end the PAYROLL program.

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

Checkpoint

1. The _____ tool checks for semantic (compile) errors on your workstation so that you can guarantee a clean compile on the iSeries.
   A. Compile
   B. LPEX Editor
   C. Program Generator
   D. Program Verifier
2. The _____ view identifies each error with an icon that identifies the severity level of the error, the ID of the error, the message, the severity, the line in the source member that caused the error, the location of the source member that produced the error, and the connection name.

    A. Remote Systems
    B. Outline
    C. Navigator
    D. iSeries Error List

3. You can sort the entries in the iSeries Error List view by:
    A. ID
    B. Message
    C. Severity
    D. Line
    E. Location
    F. Connection
    G. All of the above

4. If you used the local program verifier, then your host compiles should be successful; no wasted iSeries cycles. (T, F)

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. Each profile in the Remote System Explorer has a set of source member types, and each source type has a set of compile names associated with them. (T, F)

7. There are several ways to compile. They are:
    A. Prompted
    B. Non-prompted
    C. Interactive
    D. Batch
    E. Remote System Explorer communications server
    F. All of the above

8. The iSeries Table view displays the same information as the _____ view, with the ability to sort items and perform numerous actions.
    A. Remote Systems
    B. Outline
    C. Navigator
    D. iSeries Error List

9. From the iSeries Table view you can:
    A. List and sort libraries, objects, and members
    B. Copy, rename, delete, edit, compile and run any item in the view from the pop-up menu
    C. Transfer files from one system to another
    D. All of the above

10. You can run programs and commands from the Remote Systems view or the iSeries Table view in:
    A. A Remote System Explorer communications server job
    B. A batch job
    C. An interactive job
    D. All of the above

11. The _____ view identifies each error with an icon that identifies the severity level of the error, the ID of the error, the message, the severity, the line in the source member that caused the error, the location of the source member that produced the error, and the connection name.
    A. Remote Systems
    B. Outline

      C. Navigator
      D. iSeries Error List
12. Interactive programs require a 5250 emulator, so you need to first run a STRRSESVR <connectionName> command to associate the emulator with a particular connection in the Remote System Explorer Communications server. (T, F)

Practice

Given your experience in this exercise using the Program Verifier and Compile command, and that you have your own COBOL 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 Development Studio Client for iSeries online help to assist you in these tasks.

What you just did

In this exercise you verified your source by invoking the Program Verifier from the Remote Systems view. You then reviewed the Error List, fixed some errors and saved the source. Next you ran a remote compile from the Remote Systems view. You used the iSeries Table view to enter a command. Finally you started an interactive connection to the iSeries and ran your program all from the Remote Systems view.

In the next exercise you will use the CODE Designer tool to visually work with DDS source to design a screen for the COBOL payroll program.

Exercise 6: Design Screens and Reports

In this exercise you will become familiar with the various aspects of the CODE Designer while modifying a display file to add a screen.

At the end of the exercise, you should be able to:
* Describe the features of the CODE Designer
* Start the CODE Designer
* Open, modify and save a display file
* Add a group
* Add a record
* Add, delete and modify various DDS fields
* Verify DDS
* Compile DDS

Using an editor to create and maintain DDS source for your display and printer files can be a frustrating and difficult task. What would be great is a graphical design tool that let's you design your screens and reports visually and then generate the DDS source for you. Well, that's exactly what 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. Let's now step through each part of the interface and update some DDS as well.

In the workbench, in the Remote System Explorer perspective use the connection that you used in the exercise before.

Opening a DDS member in the Remote Systems view

To open a DDS file member in the Remote Systems view:

1. Expand the **Library List** filter or if it is still expanded use it as is

2. Expand the **QDDSSRC** file in library **RSELABxx**

3. Right-click **MSTDSP** member

4. Select **Open with > CODE Designer**

The member *MSTDSP* will be downloaded to the workstation and loaded into CODE Designer.
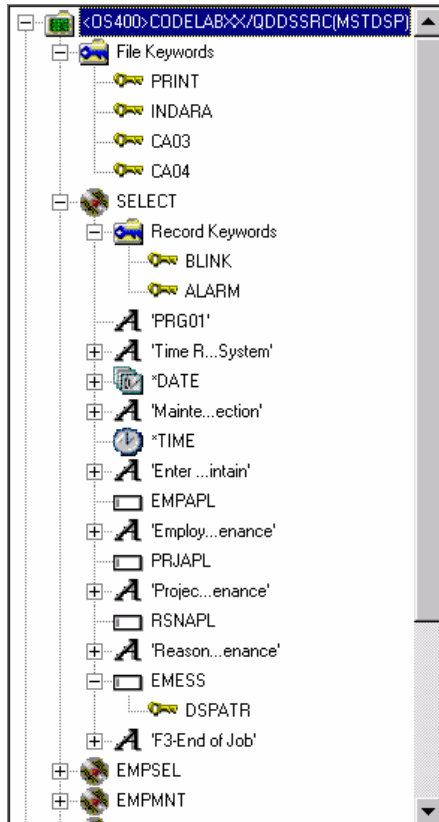
Viewing the DDS Tree

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

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

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

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

1. Click the plus sign beside the folder **<Servername>RSELABxx/QDDSSRC(MSTDSP).**
2. Click the plus sign beside the folder **File Keywords**.
3. Click the plus sign beside the record **SELECT**.
4. Click the plus sign beside the folder **Record Keywords**.
5. Click the plus sign beside the field **EMESS**.

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

Selecting the DDS object

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

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

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

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

To work with the DDS record **SELECT**:

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

Field	Position	Length	Type	Shift	Usage	Sample
A 'PRG01'	2, 5	5	Text constant			PRG01
A 'Time R...System'	2, 30	21	Text constant			Time Reporting System
*DATE	2, 70	6	Date constant			YY/MM/DD
A 'Mainte...ection'	3, 30	21	Text constant			Maintenance Selection
*TIME	3, 70	6	Time consta...			HH:MM:SS
A 'Enter ...intain'	6, 14	54	Text constant			Enter an X beside the application you
EMPAPL	9, 25	1	Alphanumeric	A - Alphanumeric	Both	B
A 'Employ...enance'	9, 28	27	Text constant			Employee Master Maintenance
PRJAPL	10, 25	1	Alphanumeric	A - Alphanumeric	Both	B
A 'Projec...enance'	10, 28	26	Text constant			Project Master Maintenance
RSNAPL	11, 25	1	Alphanumeric	A - Alphanumeric	Both	B
A 'Reason...enance'	11, 28	30	Text constant			Reason Code Master Maintenance
EMESS	21, 16	50	Alphanumeric	A - Alphanumeric	Output	00000000000000000000000
A 'F3-End of Job'	23, 7	13	Text constant			F3-End of Job

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

```
Selected DDS | Create keywords | Web Settings | Comments | Error list |

    A              EMESS          50A  O 21 16
    A   60                                       DSPATR(HI)
```

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

Designing your DDS Screen

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

On the Design page, you can easily create, edit, resize, and move DDS objects graphically. Create new records, fields, and constants directly on the Design page by using the palette push buttons to the left of the Design area or from the pop-up menus. The toolbar above the Design area allows quick access to many of the editing features as well as information about the currently selected object.

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

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

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

Creating Groups from Existing Records

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

format EMPSEL.

To create a group:

1. Scroll to the bottom of the **DDS Tree** and click the plus sign beside the **MAIN_MENU** group.
   The **SELECT** record appears as the only record in this group.

2. Right-click on the **MAIN_MENU** group.

3. From the pop-up menu, select **Insert group**. A Group Properties notebook appears and a blank
   Design page for the group **SCREEN1** also appears.
   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 bar of the CODE
   Designer. The properties notebook is modeless. When you change an object's properties, the
   selected object changes immediately.

4. On the Group Properties notebook, click the record **EMPSEL** in the **Available** list box and click
   the ⟩⟩ button. For simplicity this is the only record you will add for now. The Design page now
   shows us what the record **EMPSEL** looks like.

5. Name the group by typing over SCREEN1 with EMPL_SELCT
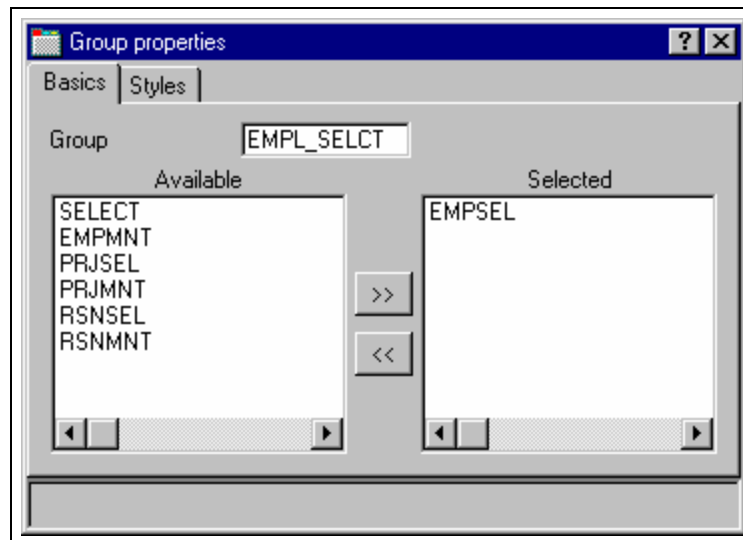
6. Click the **X** in the top right corner to close the Group Properties notebook.

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

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

Creating New Screens

To create a new 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. From the pop-up menu, select **Insert group**. 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 tools available are Create standard record and Create subfile record. If you leave the mouse over a button for a second or two, flyover help will appear describing the indicated button.
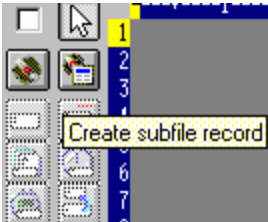
5. Click the **Create subfile record** button and then click in the dark gray area. A subfile and a subfile control record pair are created.

Adding fields to the subfile record

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

1. Now 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 mouse 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.

4. The toolbar at the top of the Design page is a very convenient place to monitor and manipulate the currently selected part.
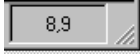
5. Rename the record from RECORD1 to EMPLSTSFL and the field from FIELD1 to OPCODE by typing over the text in the drop-down list box.

6. Click the **Color palette** button and select pink to change the color of the field. Click the **I** button to change the usage of the field to input.

   Now you will create an additional column in the subfile:

7. Position the cursor at **row 8, column 9**. Note that the bottom right of the Designer frame shows

   the current cursor position ⌷8,9⌷ If you can't see the field with the cursor position on your

   screen, click the **Maximize** button in the top right corner. You can use the cursor keys or the

   mouse to move the cursor.

8. 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)'.

9. Click **Convert string to field** 📲 button on the toolbar or press **F11** to convert the SDA syntax

   into a character output field of length 30.

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

11. Position the cursor to **8, 41**.

12. Now you will add a field for the employee's salary. 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? Type
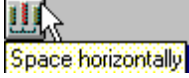
    ```
    $666,666.66
    ```

    and then press the back arrow.

13. 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!

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

15. Your subfile seems compacted to the left. It would be better to space it out evenly. Just select

    the field and click Space horizontally on the far right side of the toolbar. The other alignment buttons

    will align fields, left, right, center and top.

16. Just below the palette there are three spin buttons. The top one, **Subfile size**, specifies the total

    number of entries in the list that will be filled in by the application. The second one, **Subfile**

page size, is how many entries appear on the screen. Set the **Subfile size** to `300` (by over typing) and the **Subfile page size** to `9`. The Design page is updated accordingly.

Switching between Multiple Records

Now let's fix up the Subfile control record. The group you created contains 2 records. You can verify this by dropping down the record drop-down list box in the toolbar:

1. Change the current record by selecting RECORD1CTL from the drop-down list box or click

    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:

4. Now you need an employee name field. You could create a named field with the right

    characteristics like you did in the subfile, or you could create a source reference using the

    button in the palette or you could reference the original database field using one of the

    buttons. But there is an even simpler way. Use copy and paste! In the DDS Tree expand (click

    on the plus sign) the **EMPMNT** record.

5. Click the **ENAME** field and press **Ctrl+C.** (The pop-up menu or Edit pull down would give the

    Copy menu item as well).

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

7. Rename the field to `POS_TO`.

Adding field error handling

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

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

To set indicators:

1. Click on the ![button] button on the Design page toolbar (or press **F7**). The **Named indicator sets** window appears.

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

3. Click **Create**.
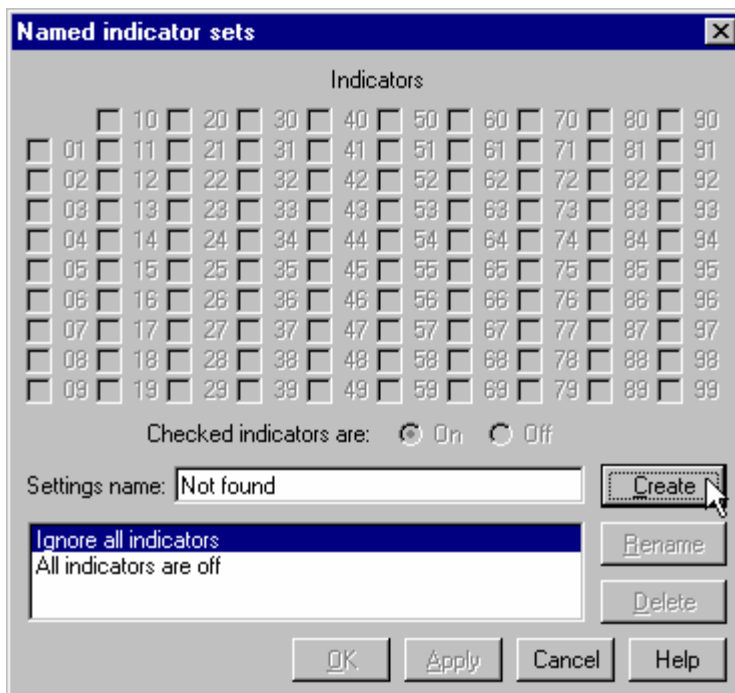
4. Click the check box next to **60** and click **OK.** The **Not found** indicator set is now in effect. The Design area is shown as if indicator 60 was on and all other indicators were off. The Design page toolbar shows the current indicator set in the drop-down list box on the bottom left.
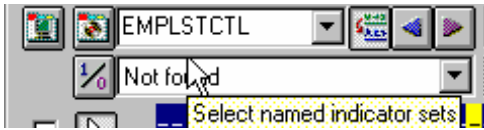
5. Now select the **POS_TO** field.

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.


```
     A                    POS_TO     R          B   4 23
     A   60                                         COLOR(RED)
     A                                              REFFLD(RCEMP/ENAME *LIBL/EMPMST)
     A   90                                         DSPATR(PR)
     A   60                                         DSPATR(RI)
     A   60                                         DSPATR(PC)
```
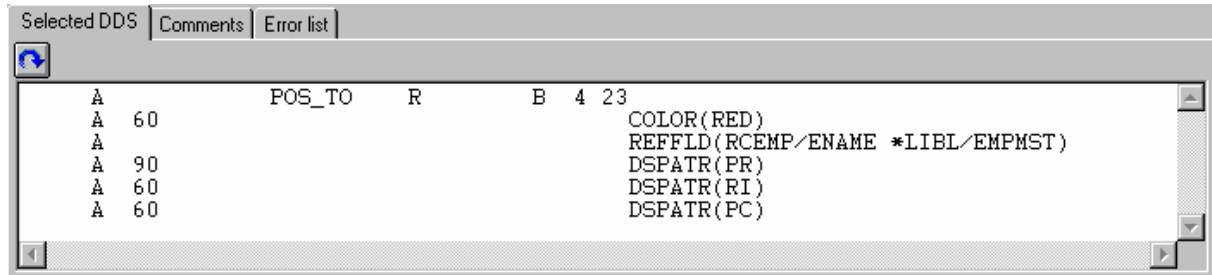
Notice that all the new keywords were created with a condition of 60. (The DSPATR(PR) was pasted in with the field originally).

8. Now let's try it out! From the **Select named indicator sets** drop-down list box, select **All indicators are off**.
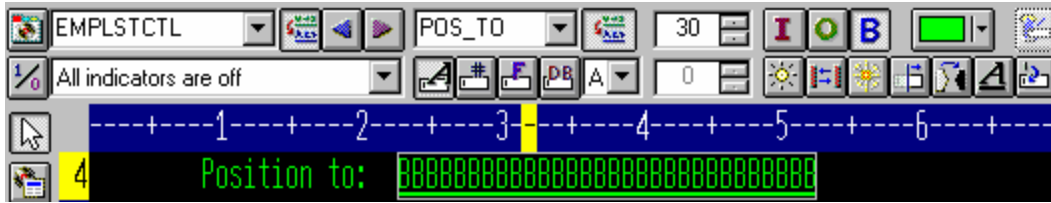
Wow! This really works!

9. From the **Select named indicator sets** drop-down list box, select **Not found**. The field appears

   red and reverse imaged.

Accessing field properties

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

The Properties notebook lets you view and update the properties of the currently selected DDS object. You can open this notebook from any view, pop-up menu, or menu bar 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-clicking 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. Now 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. Let's say we had to change the year from 2 to 4 digits. Click the **Length of year** check box.

4. Select **4 digits** from the drop-down list box. Notice how the sample is updated on the Properties

   notebook.
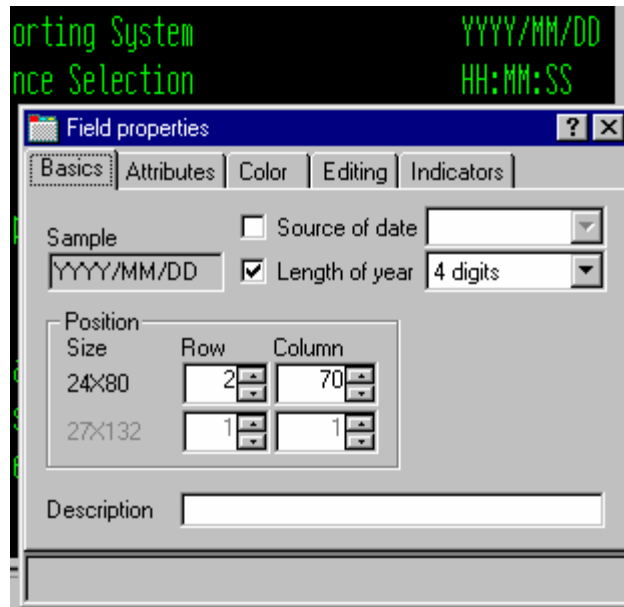
5. To test the Design page, click on the **MAIN_MENU** tab in the workbook and look at the upper right corner of the screen. The date now has 4 digits.

6. Now click on the **EMPAPL** field in the SELECT record. On the Field Properties notebook click on the **Basics** tab. On this page you can change the field's name, usage, length, type, and screen position. The other pages give you quick access to other properties for this field.
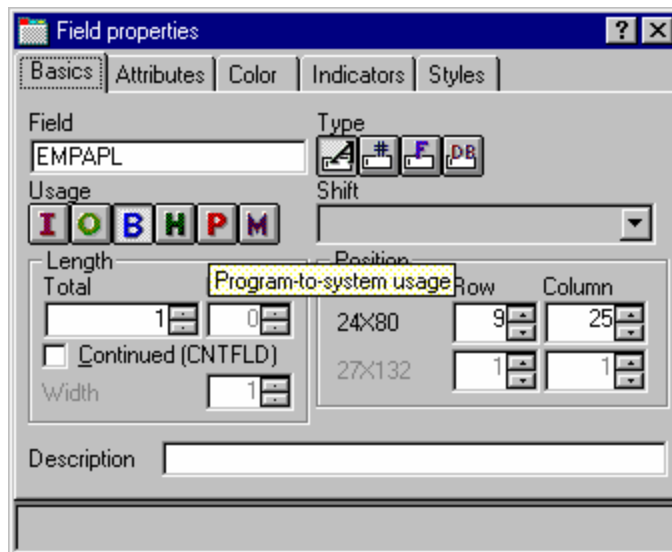
Adding New Keywords

"OK", you might think, "I can see where CODE Designer helps me manage the visual aspects of my displays and reports. But I have real work to do. I need access to the full power of DDS. I need to access keywords."

To add keywords:

1. Click **EMPAPL** field in the DDS Tree.

2. Press **F5** or select **Insert keywords** from the pop-up menu.

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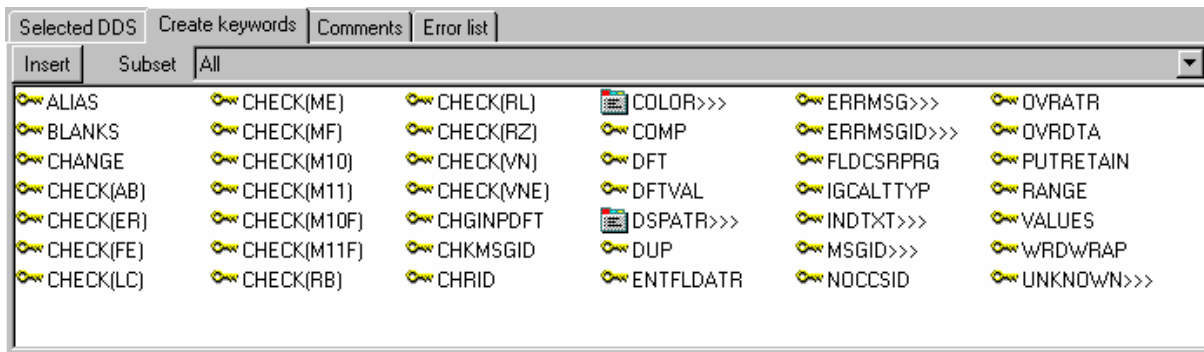
3. With the **EMPAPL** Properties notebook at the **Basics** page, click the button to change the field to numeric type. Notice that the list of keywords in the **Create keywords** page has changed.

4. Click the button to change the field back to alphanumeric. Notice that the list of keywords in the **Create keywords** page has changed.

5. Click the **ALIAS** keyword and press **F1**.

    The DDS Reference help for the ALIAS keyword appears. It is important to mention that the CODE Designer has lots of on-line help. Press F1 anywhere you want to see help for an item, icon or notebook. You will get help relevant to what you are currently trying to do. From the

Help menu you can get quick access to the DDS Language Reference as well as several other useful sources of information.

6. Minimize the Help window

7. Double-click the **INDTXT** keyword. (You may have to scroll to the right to find it). The keyword is created with default values which can be changed when you want.

8. Double-click on the **INDTXT** keyword again. The keyword is created with the same default values creating a conflict.

Verifying Your Source

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

To verify your source:

1. From the **Tools** menu, select **Verify file** (or **click** on the ![button] button on the main toolbar). 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.
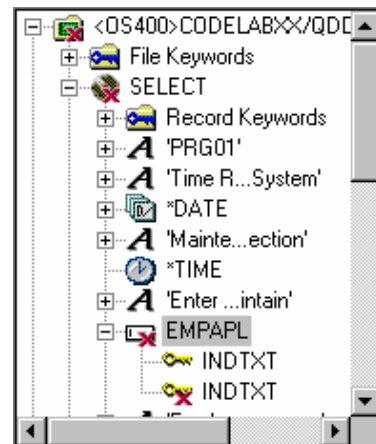
6. Double-click the warning **DDS7861** in the Error list.

(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.
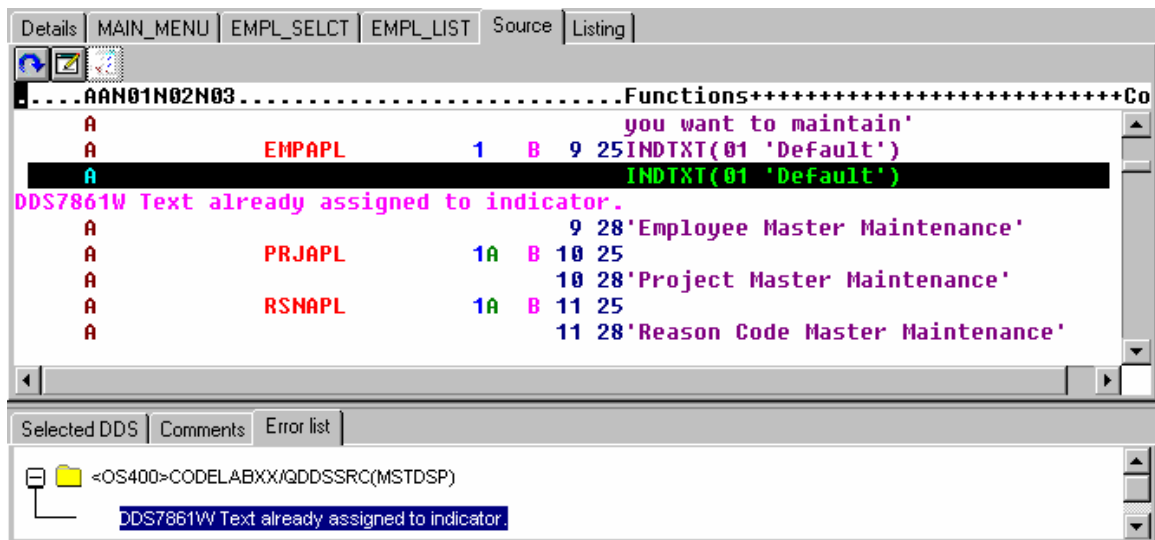
Switching between Designing and Editing your screen

To switch between the Design mode and the Edit mode:

1. Click the [button icon] button or select **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** pull downs.

3. Press **Ctrl-F** to bring up the Find/Replace window

4. Type in `INDTXT` and click **Find**.
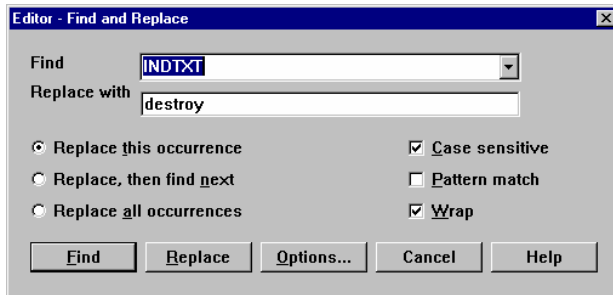
5. Press **Ctrl-N** to find the next occurrence.
6. Delete the second INDTXT line.
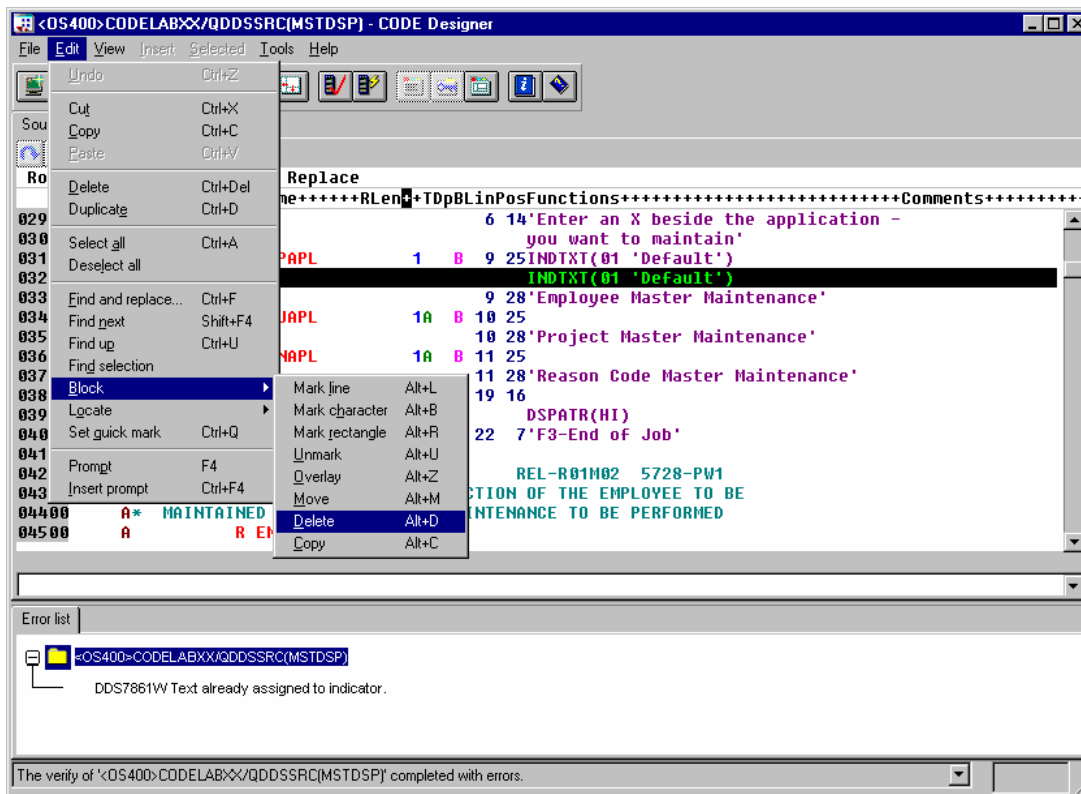
Compiling Your Source

Now you will compile the source on the iSeries just as you did in the Remote Systems LPEX Editor:

1. From the **File** menu, select **Save** to save your source to the iSeries.

2. From the **Tools** menu select **Compile** and then select **No prompt** (or **click** the ![icon] button on the main toolbar).

3. A message will tell you when the compile is complete. Click **OK** in the Message window.

If you run the PAYROLL program, you will see the 4 digit year change you made to the opening screen of the program.

Closing CODE Designer

From the **File** menu, select **Exit**.

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

Checkpoint

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. The DDS tree view presents:
   A. The loaded DDS source as a tree structure
   B. A hierarchy of files, records, field, help specifications, keys and keywords in each selected DDS object
   C. Groups of records, which represent the screens or reports you are designing, as peers of the file in the tree hierarchy
   D. All of the above
3. Match the item with its correct description.
   A. Workbook
   B. Details page
   C. Utility notebook
   D. Design page
   E. Properties notebook
   F. Group
   G. Indicator

   A. View specific elements of the DDS source such as errors and selected objects
   B. Lists the contents of the currently selected item in the DDS tree
   C. Contains several different tabbed pages
   D. For visually designing screens and reports
   E. A collection of one or more DDS records that make up a single screen or report
   F. Access properties of a field, record, or entire file
   G. Two digit variable to signal whether or not certain events have occurred during processing
4. The Utility notebook contains:
   A. Selected DDS page

    B. Error List page
    C. Create keywords page
    D. Comments page
    E. All of the above
5. There are two modes in the Details page: Details and List. The Details mode shows more about each DDS object while the List mode shows more DDS objects in a page. (T, F)
6. 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
7. Grouping records together allows you to work on one record while still seeing the related records in the background. (T, F)
8. The Properties notebook is modeless. When you change an object's properties, the selected object changes immediately. (T, F)
9. The DDS source is checked using the same verifier that the LPEX Editor or CODE Editor uses. (T, F)
10. You can switch between design mode and edit mode in CODE Designer. (T, F)
11. You can compile sources on the iSeries just as you in the LPEX Editor. (T, F)

Practice

Given your experience in this exercise 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 this information as you create your report. When you are familiar with the file REFMST, you can create your printer file. Use the Development Studio Client for iSeries online help to assist you in this task.

What you just did

In this exercise you: You used CODE Designer to modify a display file to add a screen. You learned how to add groups, records and fields. You saved your DDS and then verified it for compile errors. When you had no compile errors you then compiled your DDS.

In the next exercise you will learn how to debug and test an application running on an iSeries system.

Exercise 7: Debug an ILE COBOL Program

The goal of this exercise is to review the Debugger features. You then 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.

At the end of the exercise, you should be able to:
* Describe some of the features of the Debugger
* Start a Debug session
* Add and delete line breakpoints
* Display a variable
* Change a variable
* View the call stack
* Run, Step and Step into a program
* Switch between Source and Listing view
* Add a storage monitor
* Add a watch breakpoint

Debugging iSeries programs from the Debug perspective

The debugger is a source-level debugger that allows you to debug and test an application running on an iSeries system from your workstation. 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 list.

* Set watch breakpoints to make the program stop whenever a specified variable changes.

* View the call stack of your program in the Stacks page. 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 page. 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 Programs page.

- Debug multithreaded applications, maintaining separate Stacks for each thread with the ability to enable and disable any individual thread.
- Load source from the workstation instead of the iSeries – useful if you don't want the source code on a production machine.
- Debug client/server and distributed applications.

The Debugger supports Java as well as RPG/400 and ILE RPG, COBOL and ILE COBOL, C, C++ and CL.

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

Starting the Distributed Debugger

You will be working with the ILE COBOL program **PAYROLLD**.

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

You can start the Debugger in several ways: direct from the Remote Systems view, or from the Launch Configurations window. Starting direct 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 includes specifying parameters. To make the exercise interesting you will use CL program CLC1 to call PAYROLLD and you will pass one parameter to CLC1. This means you will use the Launch Configurations window.

Use the Remote System Explorer to start the Launch Configurations:
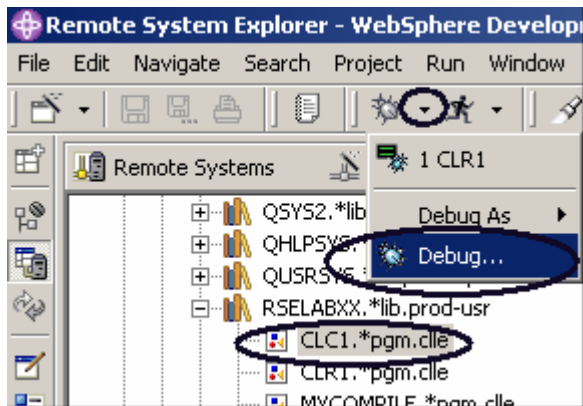
Figure 44: Start the Launch Configurations window

In the Remote Systems view:

1. Expand the **Library list** filter, if it isn't expanded already

2. Expand library **RSELABxx**, if it isn't expanded already

3. Select program **CLC1** in library RSELABxx

On the workbench menu:

4. Click the menu drop-down arrow beside the **DEBUG** icon as shown in Figure 44

5. Select **Debug** from the drop-down menu

The Launch Configurations window appears:
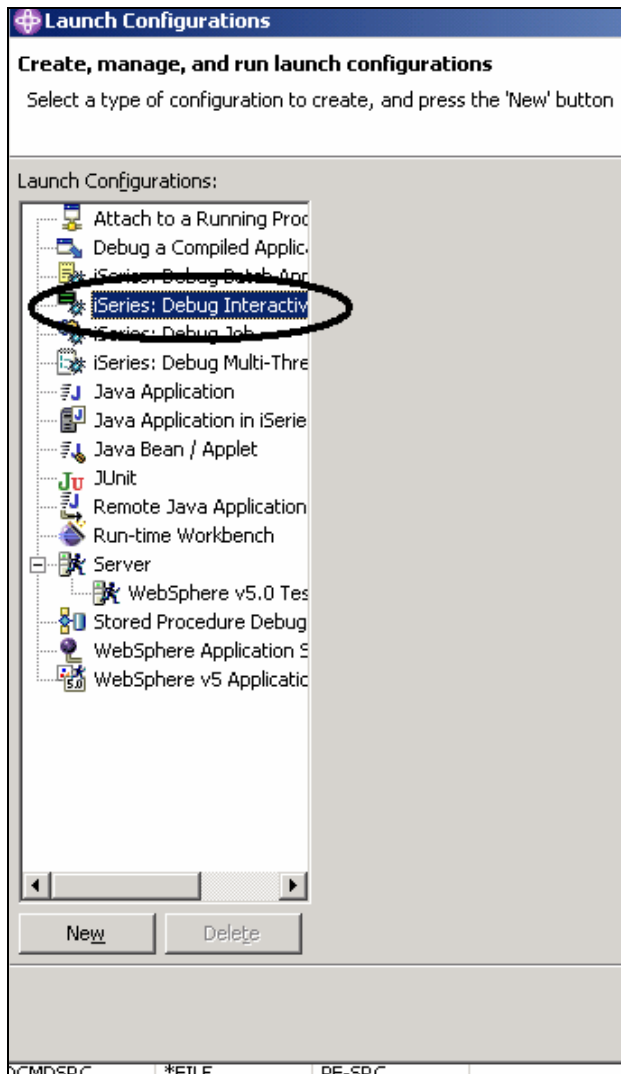
Figure 45: Launch Configurations window

In the tree view:

1. Select **iSeries: Debug Interactive Application**

2. Click **New**.

The Debug Interactive Application pane appears on the right side of the window:
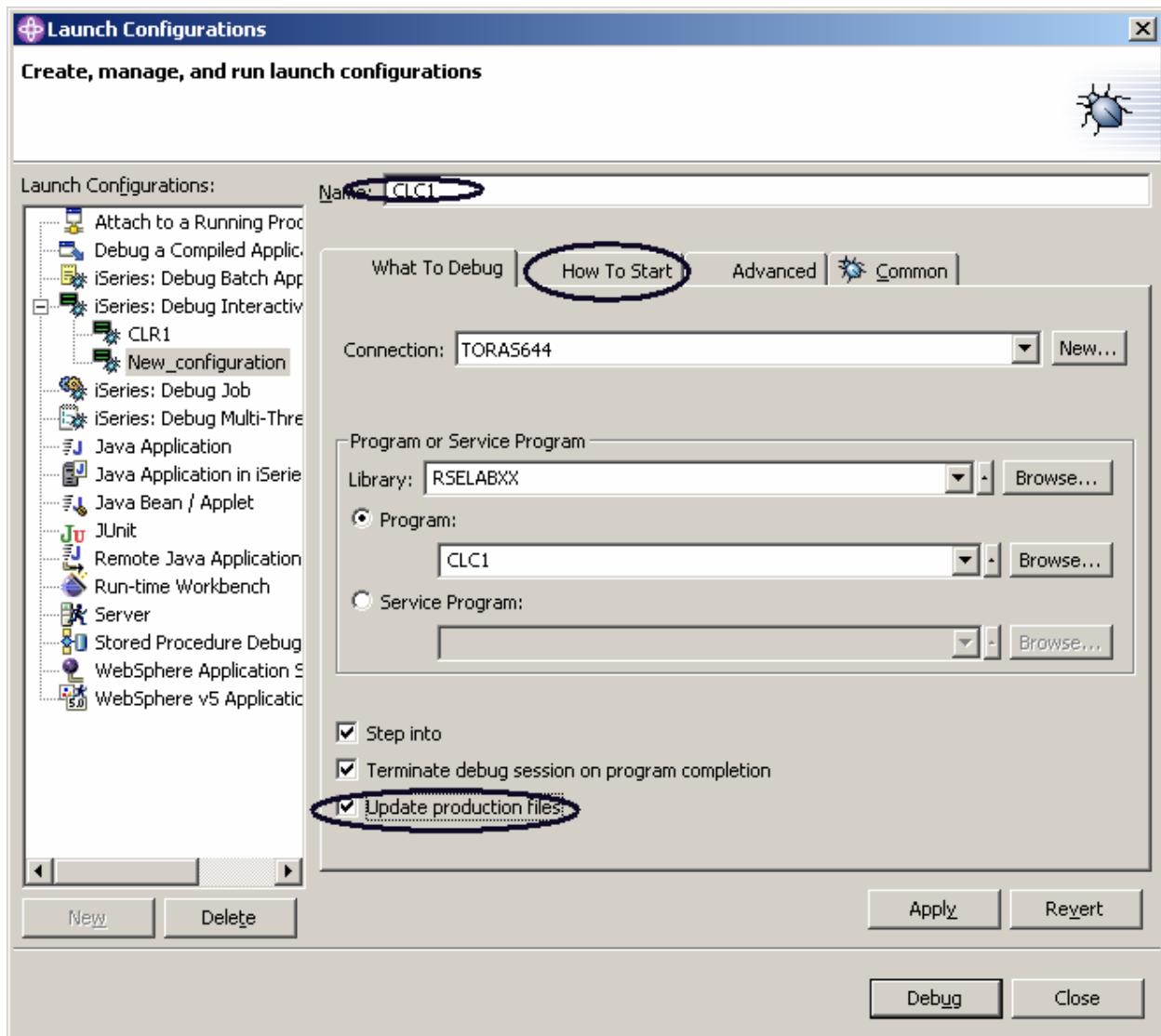
Figure 46: Debug Interactive Application

3. Enter the program name `CLC1` in the **Name** field

4. Select the **Update production files** check box

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

On the How To Start page:
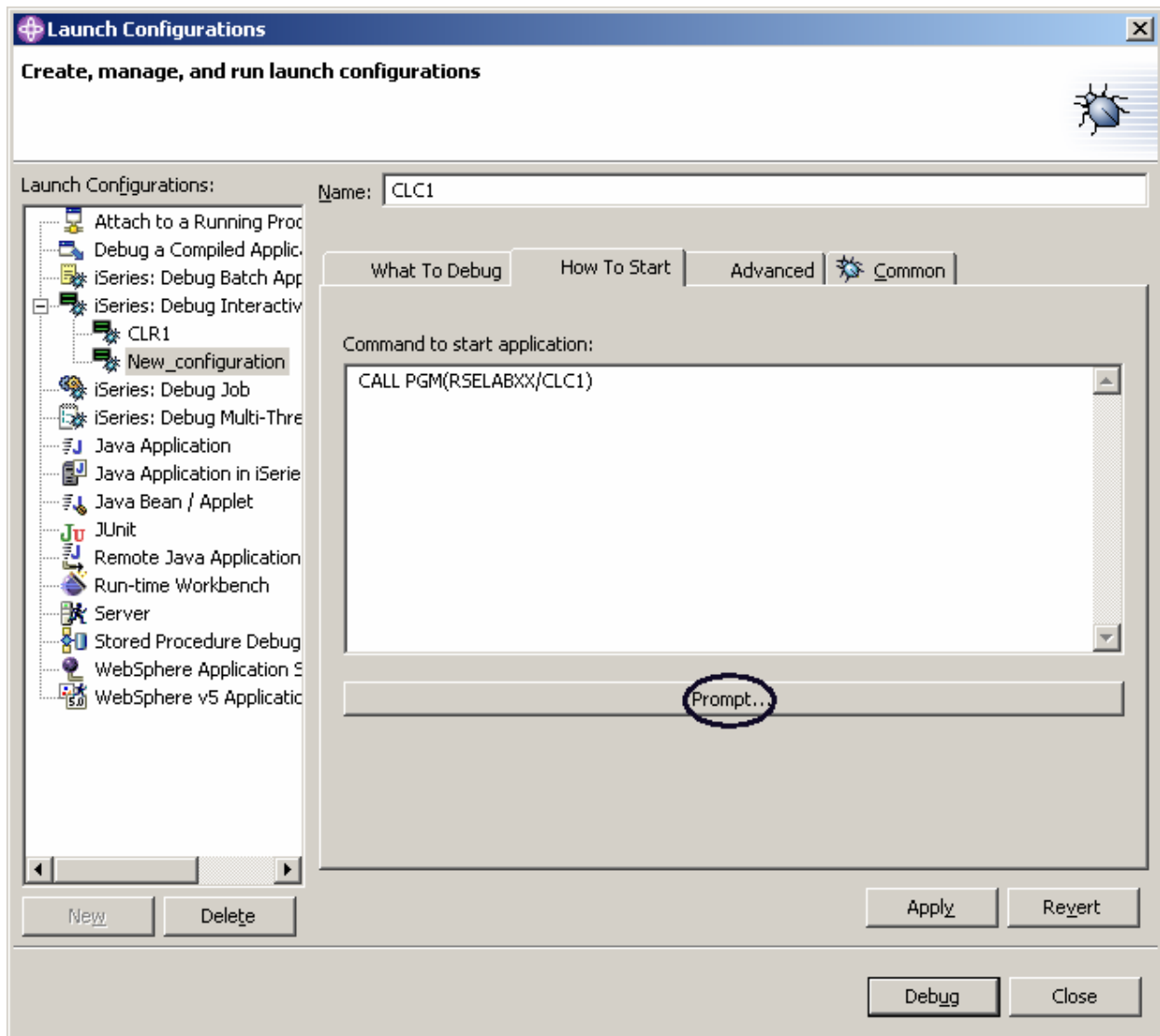
Figure 47: How to Start page

6. Click **Prompt**.

    The Prompt window opens.
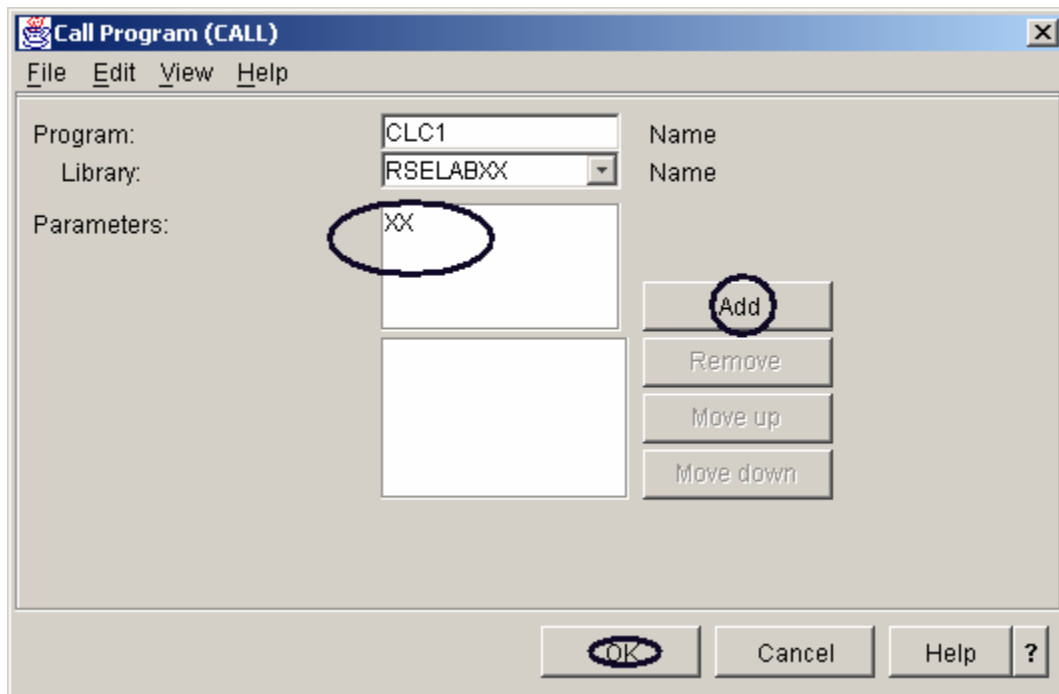
Figure 48: Prompt window

7. Enter **'**xx**'** in the **Parameters** field

8. Click **Add**.

    The parameter value will appear in the lower list box.

9. Click **OK**.

    The complete start command for the program appears in the window:
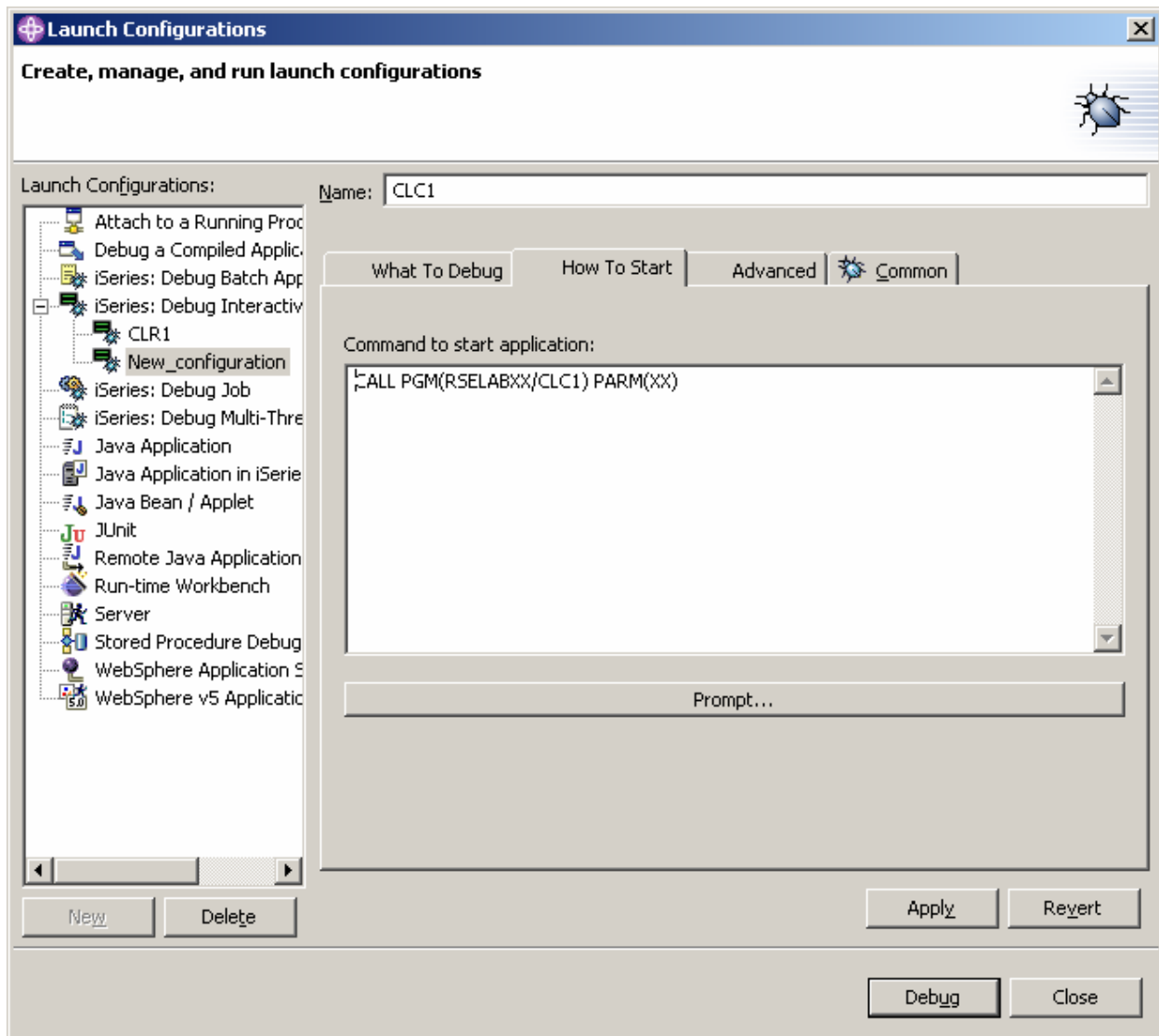
Figure 49: Ready to debug

10. Click **Debug**

    If an error message like this appears:
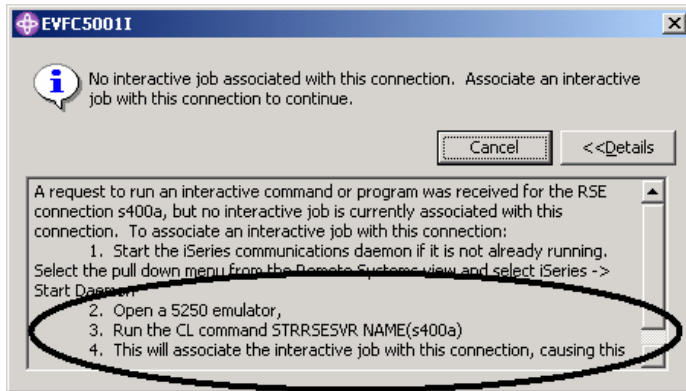
Figure 50: Error message if interactive session doesn't exist

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.

Cancel the message in the Debug perspective.

Now the Debug perspective is loaded in the workbench:
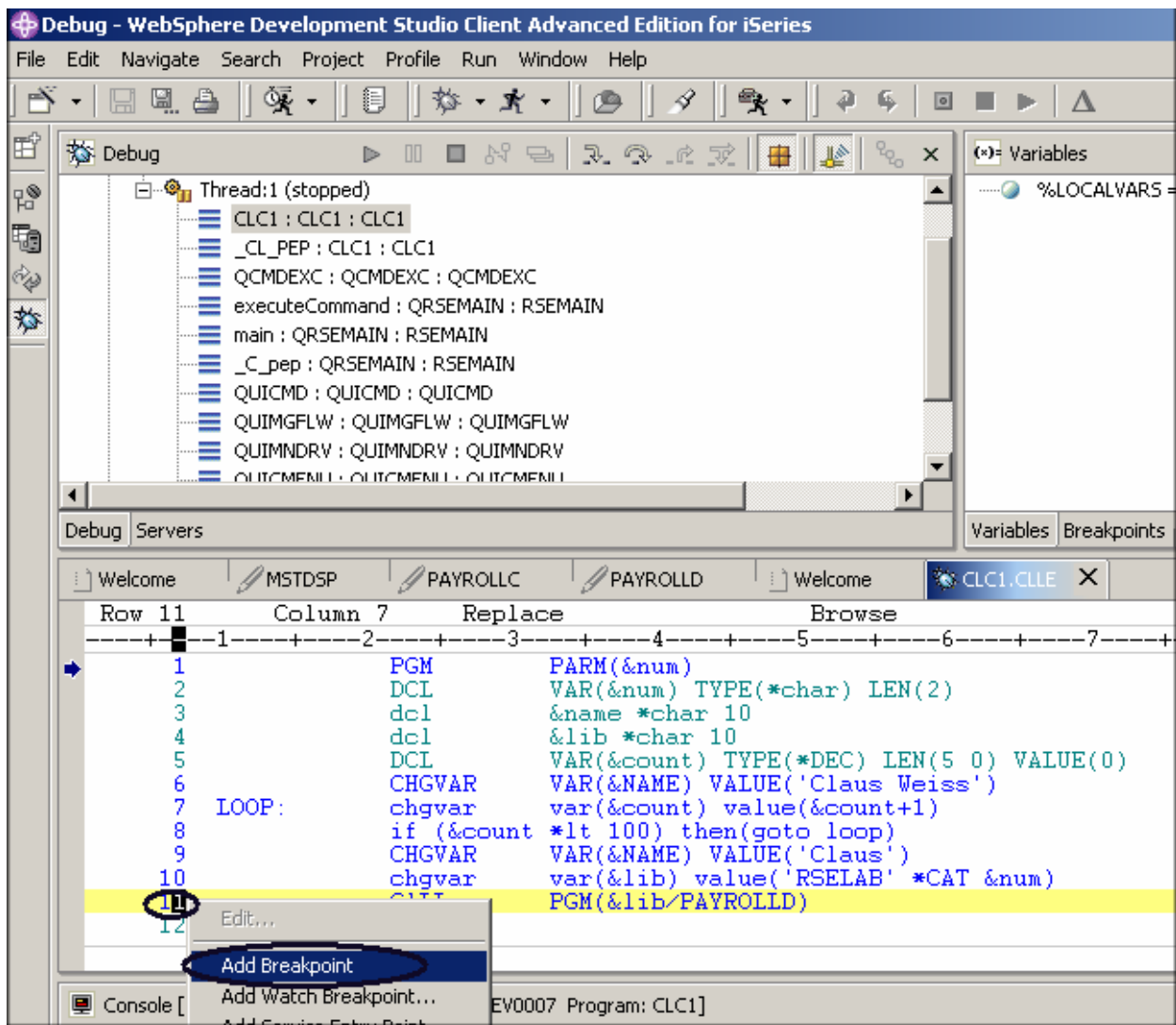
Figure 51: Debug perspective, adding breakpoint

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

Setting Breakpoints

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

The easiest way to set a breakpoint is to right-click on the line in the **Source** pane:
1. Right-click anywhere on **line 11**

2. Select **Add breakpoint** from the pop-up menu.

To add a conditional breakpoint in the loop when it loops the 99th time:
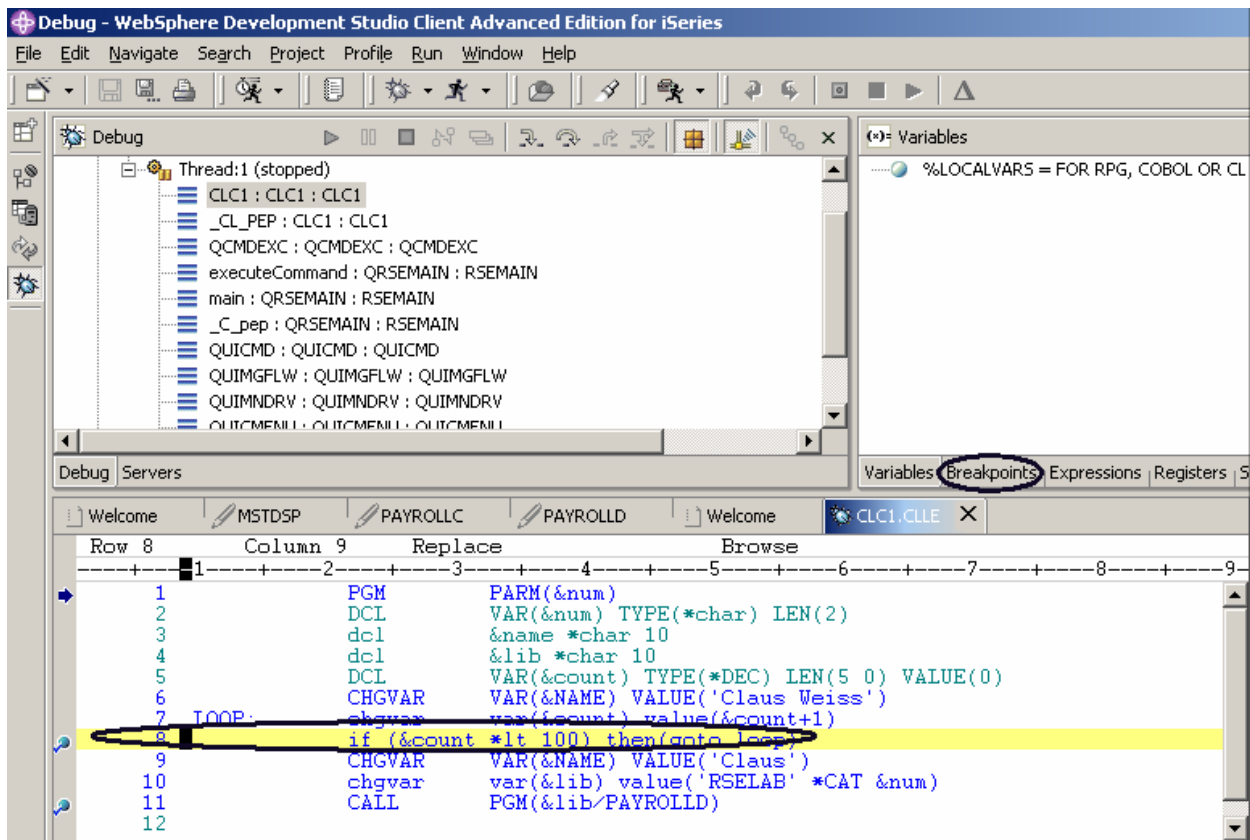
Figure 52:Adding a break point and selecting the breakpoint view

3. Right-click on **line 8**

4. Select **Add breakpoint** from the pop-up menu.

5. Select the **Breakpoints** tab in the upper right pane of the workbench as shown in Figure 52

    In the **Breakpoints** view:
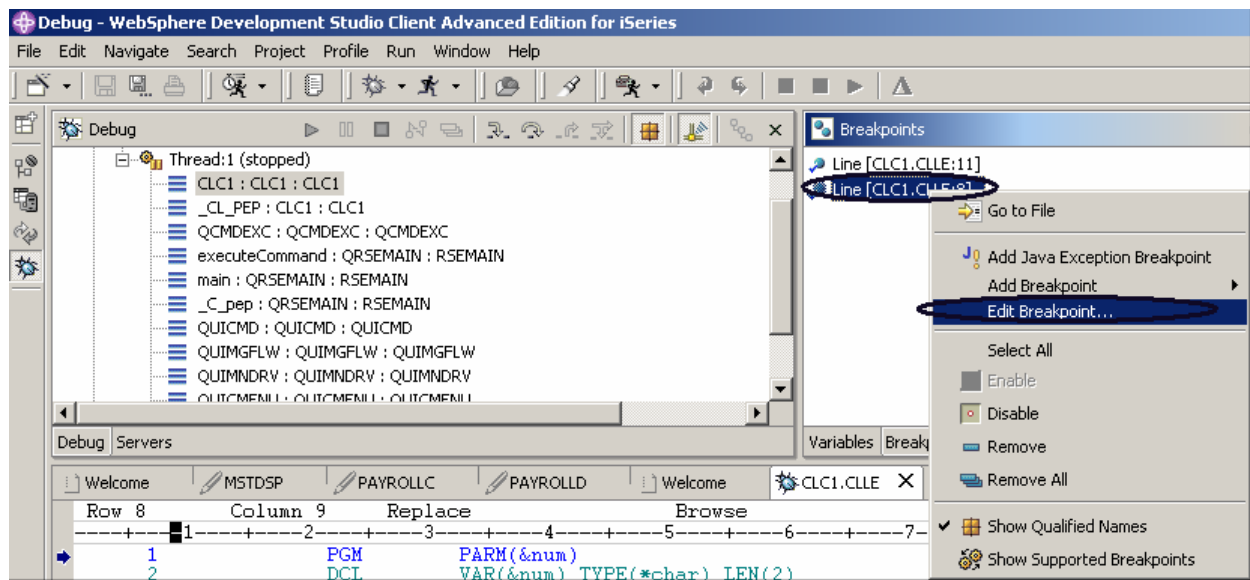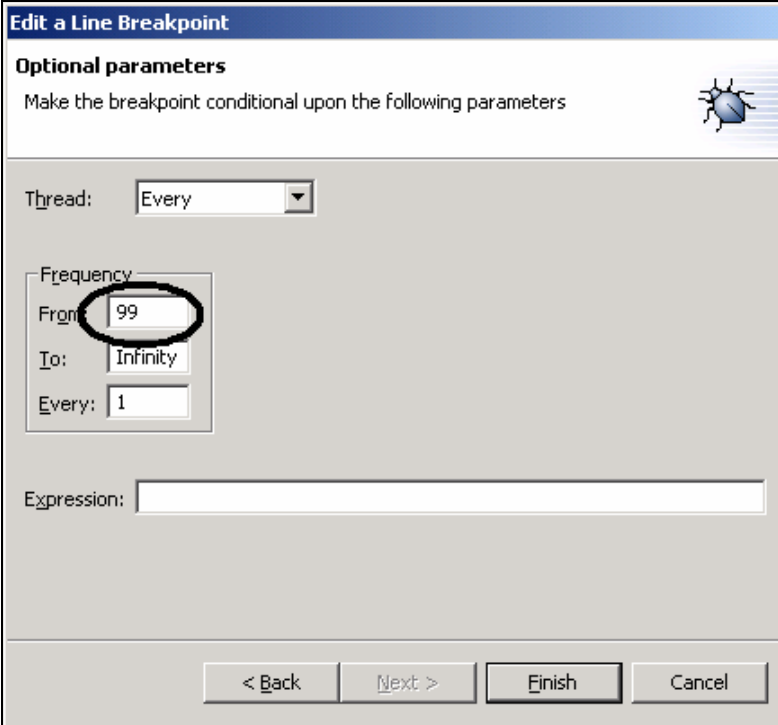
Figure 53: Edit breakpoint in Breakpoint view

6. Right-click **line 8** breakpoint

7. Click **Next** in the **Edit a line breakpoint** window

   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.

Figure 54: Specify condition for breaking

8. Enter 99 in the **From** entry field

9. Click **Finish**.

Monitoring Variables

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

In the Source pane:
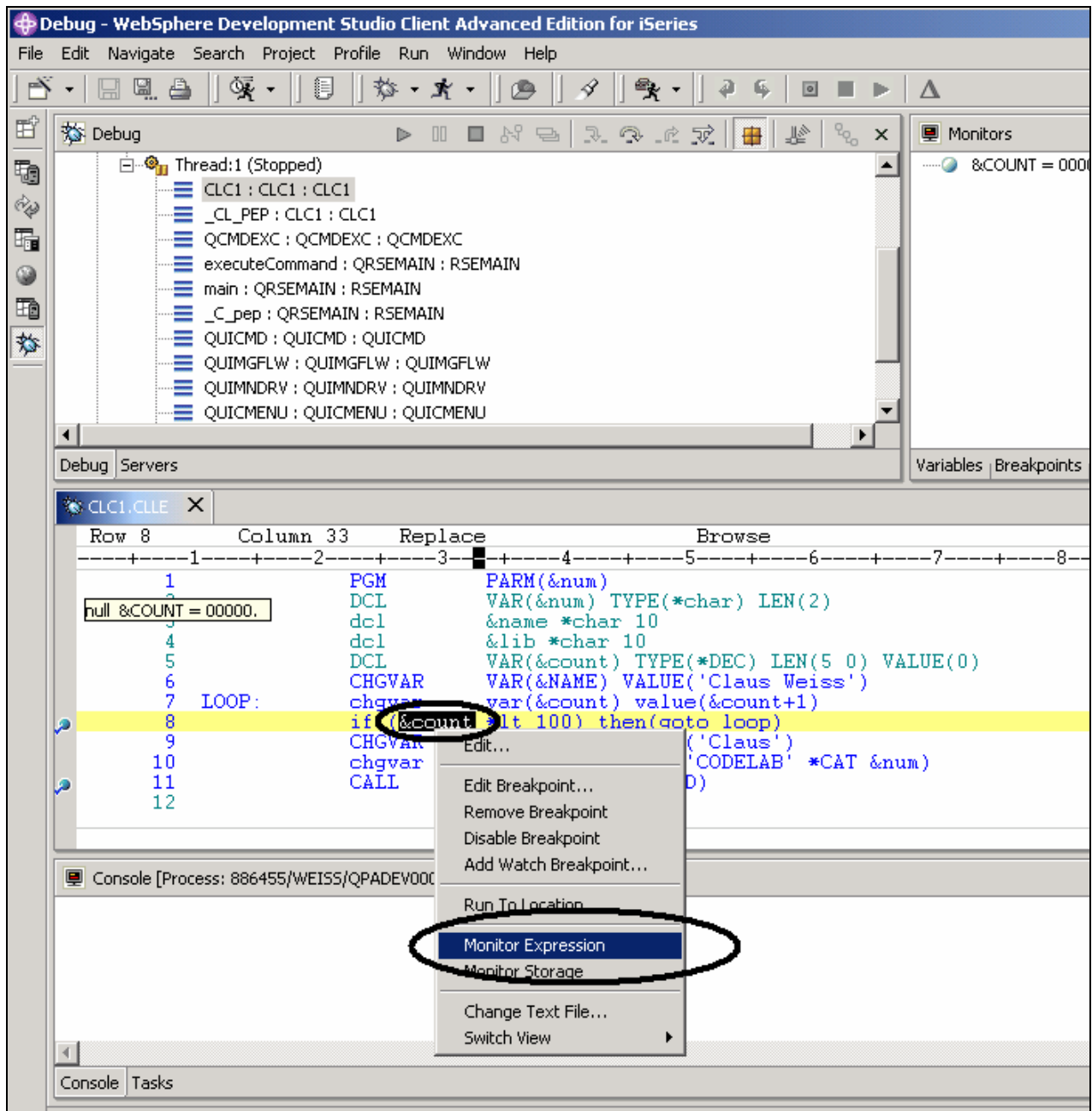
Figure 55: Select variable to be monitored

1. Double-click on the variable &COUNT

2. Right click &COUNT

3. Select **Monitor Expression** from the pop-up menu.
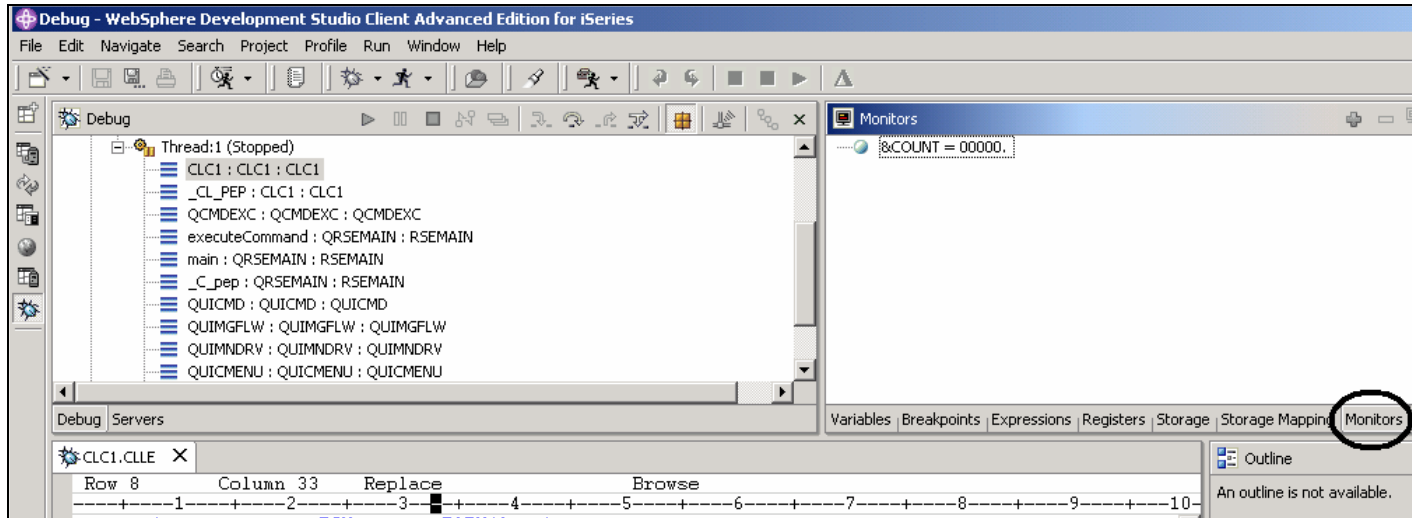
Figure 56: Monitor view with variable &COUNT

The variable appears in the Monitors pane.
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:

Figure 57: Run icon on Debug toolbar

4. Click **Run** ▶ 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 **Run** again.

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

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

Stepping into a Program

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

Figure 58: Debug toolbar with Step Debug icon

1. Click **Step into** on the Debug toolbar. The listing view of PAYROLLD is displayed.

    Depending on the option you used to compile the program (*SRCDBG or *LSTDBG for COBOL, or *SOURCE, *LIST, or *ALL for ILE COBOL), this window displays either the Source or Listing View.
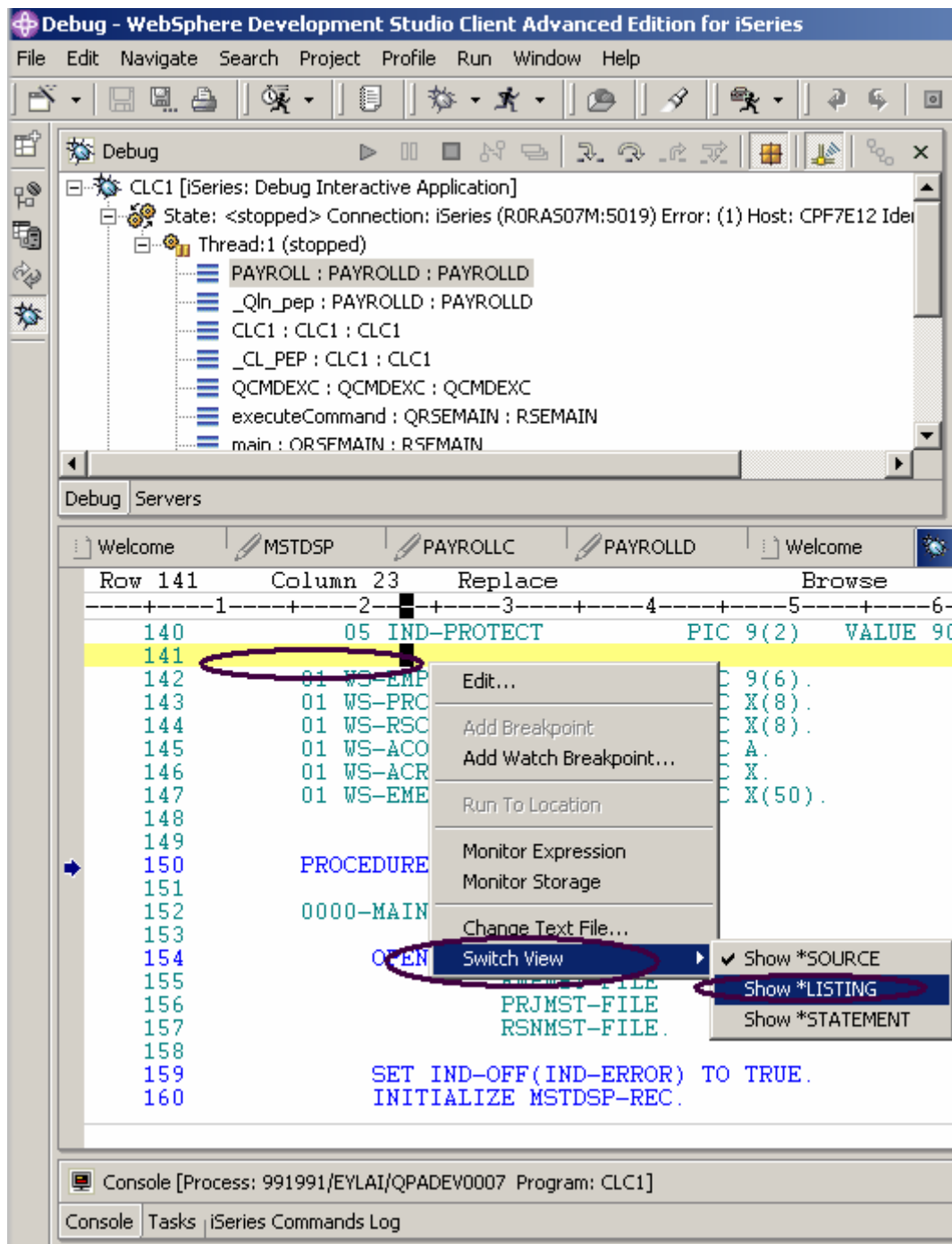
Figure 59: COBOL source view

2. Right-click anywhere in the **Source view**

3. Select **Switch view > Show *LISTING** from the pop-up menu.

```
Welcome    MSTDSP    PAYROLLC    PAYROLLD    Welcome    PAYROLLD.LISTING  X
  Row 390      Column 1      Replace                    Browse
---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9-
    381              000141
    382    195       000142 01 WS-EMPNO                 PIC 9(6).
    383    196       000143 01 WS-PRCDE                 PIC X(8).
    384    197       000144 01 WS-RSCDE                 PIC X(8).
    385    198       000145 01 WS-ACODE                 PIC A.
    386    199       000146 01 WS-ACREC                 PIC X.
    387    200       000147 01 WS-EMESS                 PIC X(50).
    388              000148
    389              000149
 >  390    201       000150 PROCEDURE DIVISION.
    391              000151
    392              000152 0000-MAIN.
    393              000153
    394    202       000154     OPEN I-O MSTDSP-FILE
    395              000155              EMPMST-FILE
    396              000156              PRJMST-FILE
    397              000157              RSNMST-FILE.
    398              000158
    399    203       000159     SET IND-OFF(IND-ERROR) TO TRUE.
    400    204       000160     INITIALIZE MSTDSP-REC.
```

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

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

5. Select **Switch view > Show *Source**

Listing Call Stack Entries

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

In the Debug view, expand the stack entry of Thread1 if it is not expanded already, then click the plus sign in front of it.
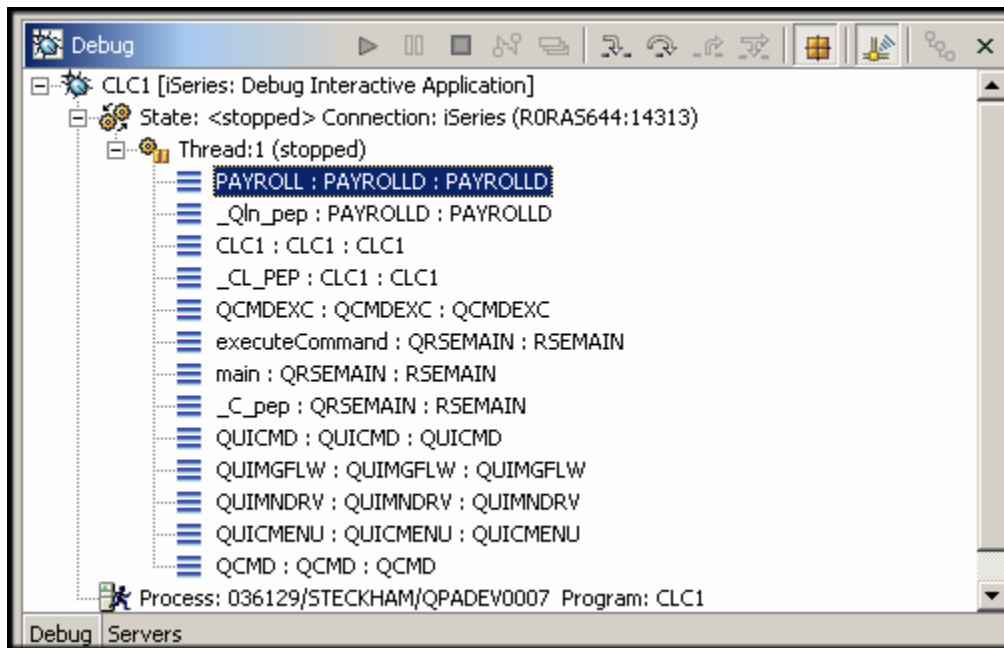
Figure 60: Call Stack in Debug view

It allows you to work with and switch between different programs and/or ILE modules.

Setting Breakpoints in PAYROLLD

Now you add some breakpoints in PAYROLLD.

To add breakpoints:

1. Right-click line **201**

2. Select **Add Breakpoint** from the pop-up menu.

A blue breakpoint icon is added to the prefix area of this line to indicate that a breakpoint is set.

3. Repeat the above step for line **206**.

4. Repeat the above step for line **218**

To view all breakpoints, select the **Breakpoints** tab from the top left pane. This view shows all breakpoints currently set in your programs.
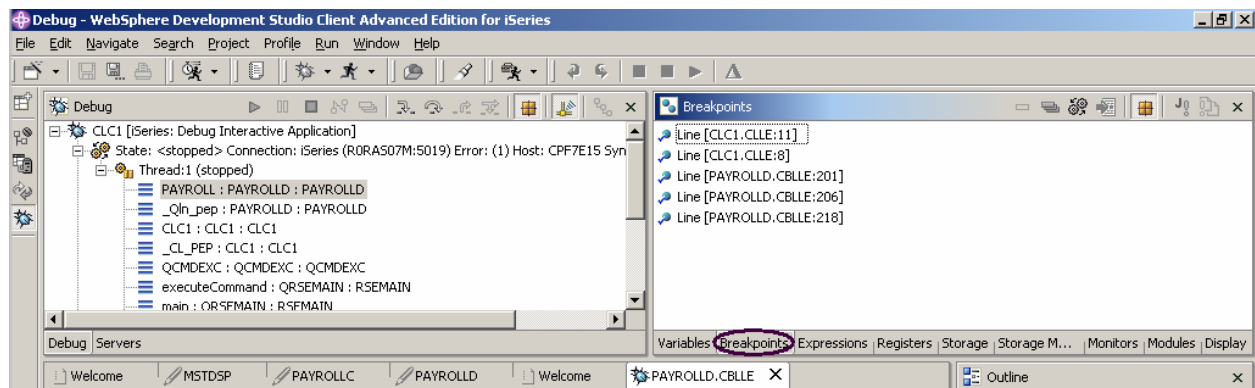
Figure 61: Breakpoints view

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

Removing a Breakpoint

It is also easy to remove breakpoints from the **Source** pane:

1.   Right click line **206**

2.   Select **Remove Breakpoint**

The blue icon is removed from the prefix area indicating that no breakpoint is set on that line.
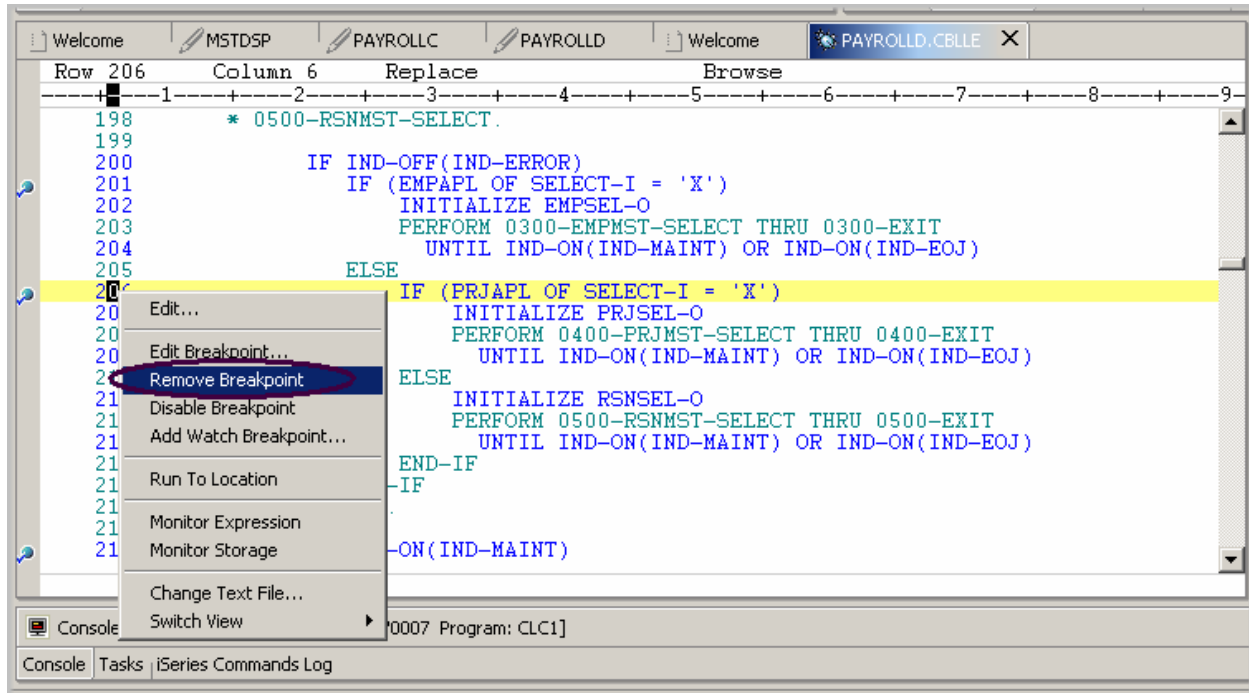
Figure 62: Remove breakpoint

Now you are ready to run the PAYROLLG program:

3. Click **Run** ▶ from the Debug toolbar.

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

4. Type an `X` beside the **Project Master Maintenance** option

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

The program runs to the breakpoint at line **201**.

Monitoring Variables in PAYROLLD

Now let's monitor variables and change them in PAYROLLD.

In the Source pane:

1. Double-click the variable `EMPAPL` on line **201**

2. Right click the variable

3. Select **Monitor Expression**
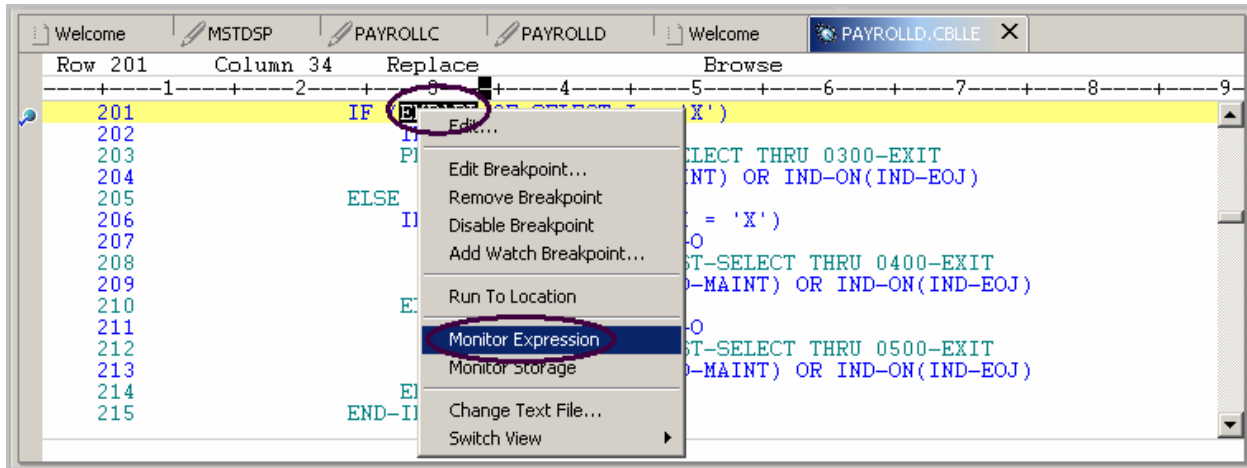
Figure 63: Add variable to monitor

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 **206** and RSNAPL on line **244** to the monitor.

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

   In the **Monitors** view:

Figure 64: Change variable content

6. Double-click the variable RSNAPL.

   The value changes into an entry field.

7. Type in the new value X for the variable.

8. Click another variable to leave the field**.**

The variable is successfully changed.

Adding a Storage Monitor

Adding a storage monitor for a variable allows you to view the storage starting with the address where the variable is located. You can choose between two display formats: hexadecimal and character or character only.
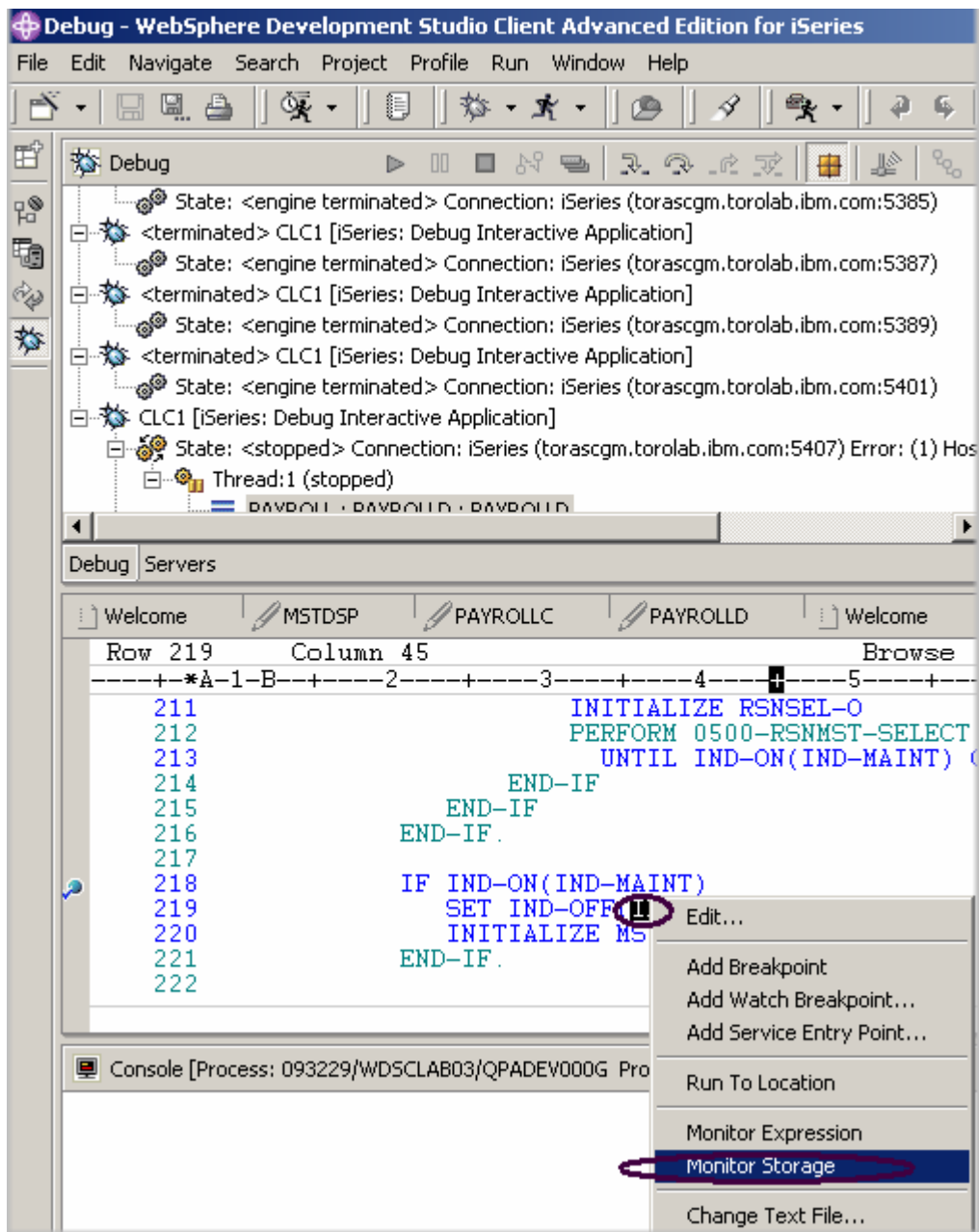
Figure 65: Adding a Storage Monitor

In the Source window:

1. Double click the variable IND-ERROR in line **219**

2. Right click and select **Storage Monitor** from the pop-up menu

   A new page is added to the Storage view. The tab shows the name of the variable.
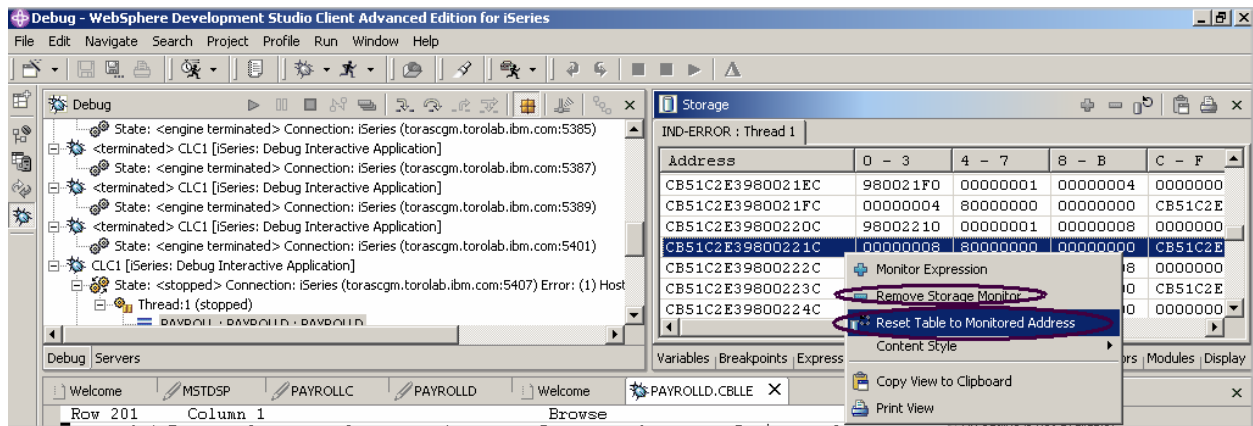
Figure 66: Change storage monitor

3. Use the scroll bar on the right of the Storage view to scroll down. You can see the content of the memory at this moment

4. Right-click in the **view area**

5. Select **Reset Table to Monitored Address** to get back to the starting address.

6. Right click the **view area**

7. Select **Remove Storage Monitor** to remove the storage monitor.

Setting Watch Breakpoints

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

In the Source view:

1. Go to line **118**.

2. Double-click variable IND-TABLE to highlight it.

3. Select **Add Watch Breakpoint.**

The **Watch Breakpoint** window appears. The **Expression** field is pre-filled with the highlighted variable IND-TABLE.

Add a Watch Breakpoint

Required information

Sets a breakpoint to stop execution when data changes at a specific address

Address or expression: `IND-TABLE`

Number of bytes to watch: `0`

< Back   Next >   Finish   Cancel
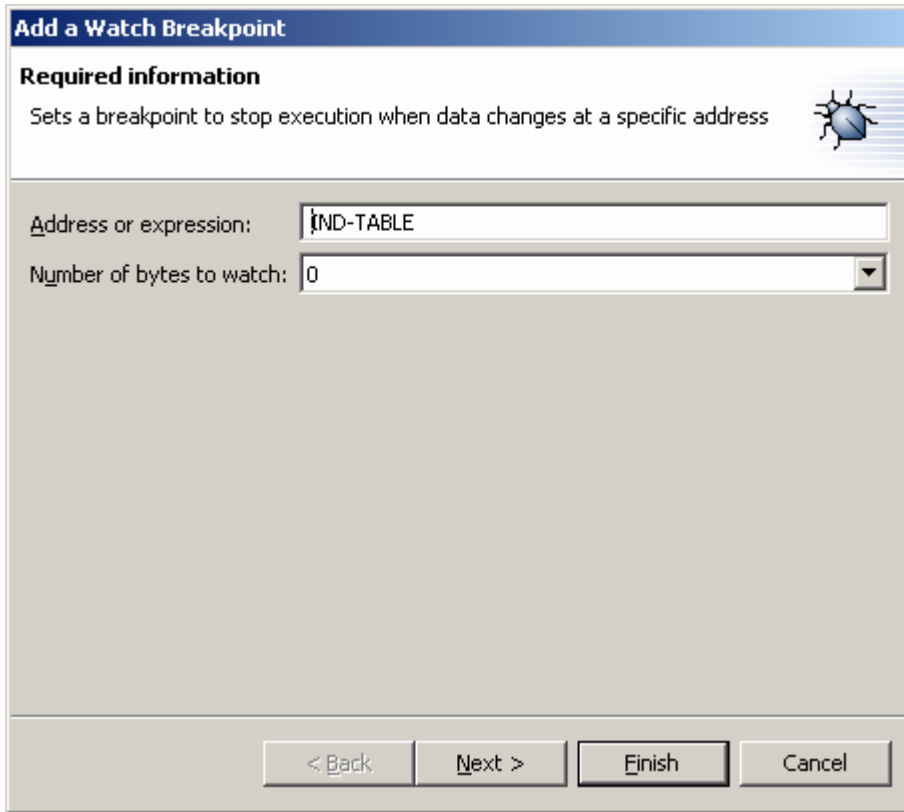
Figure 67: Add Watch Breakpoint

4. Click **Finish**.

   The watch breakpoint is now set.

5. Click **Run** on the Debug toolbar.

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

   In the 5250-emulation session:

6. 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 `IND-TABLE` has changed.
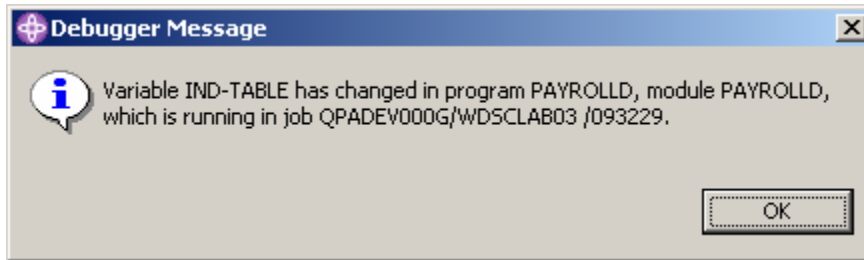
Figure 68: Watch breakpoint message

8. Click **OK**

The program stops at line **307**. This line is located immediately after the statement which caused the variable `IND-TABLE` to change.

Closing the Debug Session

1. Click **Run** on the Debug toolbar.

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

2. Switch to the 5250 emulation session.

3.   Press **F3** to end the job.

    A message **Debug session terminated** comes up.

4. Click **OK**

    Close the Debug perspective by performing the following steps:
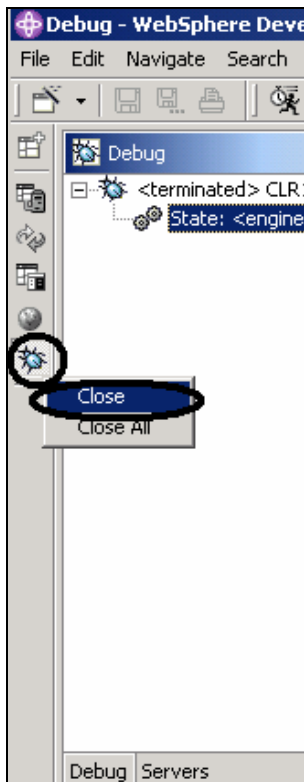
Figure 69: Close debug perspective

5. Right click the Debug icon ⊞ on the left hand frame in the workbench

6. Select **Close** from the pop-up menu

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

Checkpoint

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

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

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

What you just did

In this exercise you learned about the Debugger features. You then started the debugger, set breakpoints, monitored variables, ran and stepped into a program, viewed the call stack in the Debug view, removed a breakpoint, added a storage monitor, and set watch breakpoints and all from the Debug perspective.

In the next exercise you will learn how to create filters and actions to manage your iSeries objects all from the Remote Systems Explorer. In short, you'll see how Remote System Explorer can organize and integrate your work and make that work easier.

Exercise 8: Exploring Remote System Explorer

In this exercise you will use the Remote System Explorer perspective to work with the iSeries objects that you used in the previous exercise. You will also see 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.

At the end of the exercise, you should be able to:
- Describe Remote System Explorer
- Describe Remote System Explorer, filters, user actions and running commands
- Open Remote System Explorer
- Create filters (library, object)
- Change the library filter
- Create a user-defined action
- View properties
- Run commands from iSeries Table view

Introducing the Remote System Explorer

Most of the function of the CODE Project Organizer has been replaced by WebSphere Studio function with the exception of accessing ADM parts.

The Remote System Explorer is replacing PDM (Program Development Manager) on the workstation. It currently doesn't have all the function of PDM but will overtime be a full replacement for PDM.

 Remote System Explorer allows you to:
- Simplify your work by giving you quick access to lists of iSeries libraries, objects, members, IFS files, UNIX files, and local files.

- Use the context-sensitive pop-up menus on these lists to perform actions such as start the LPEX Editor, CODE Designer, or Distributed Debugger or other common iSeries actions.

- 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.

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

Creating a library filter

In the Remote System Explorer perspective, you now want to work with specific iSeries objects.
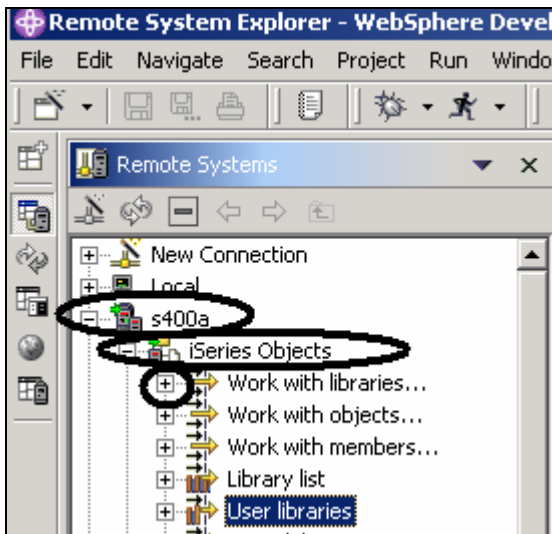
Figure 70: Expand Work with Libraries

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

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

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

The Remote Systems Explorer supports the Native File System (QSYS) and allows you to query the objects that are on your iSeries host. Using filters, you can easily organize how you wish to view the objects within your system. When you access the list by expanding the filter, you will see the Native File System objects (such as libraries or files) and can perform actions against these remote objects.

If you have been using the Remote Systems Explorer for some time, your workspace may contain too many filters to easily navigate. In this case, filters can be partitioned into pools, where one pool contains filters for many different elements. For example, one filter pool may contain filters for your accounts receivable program while another contains filters for your payroll program.

The Remote Systems Explorer supports the Native File System (QSYS) and allows you to query the objects that are on your iSeries system. Using filters, you can easily organize how you wish to view the objects within your system. When you access the list by expanding the filter, you will see the Native File System objects (such as libraries or files) and can perform actions against these remote objects.

First you will need to specify the library you want to work with directly:

1. In the Remote Systems view expand the connection that connects to your iSeries system, by clicking on the plus sign beside it.

2. Expand **iSeries Objects**

   To create a new library filter:

3. Expand **Work with libraries**. (You can also right-click iSeries Objects and select **New > Library Filter**).

   Expanding Work with libraries corresponds to the WRKLIBPDM command, plus creates and expands the filter in the Remote Systems view. You see this window open:
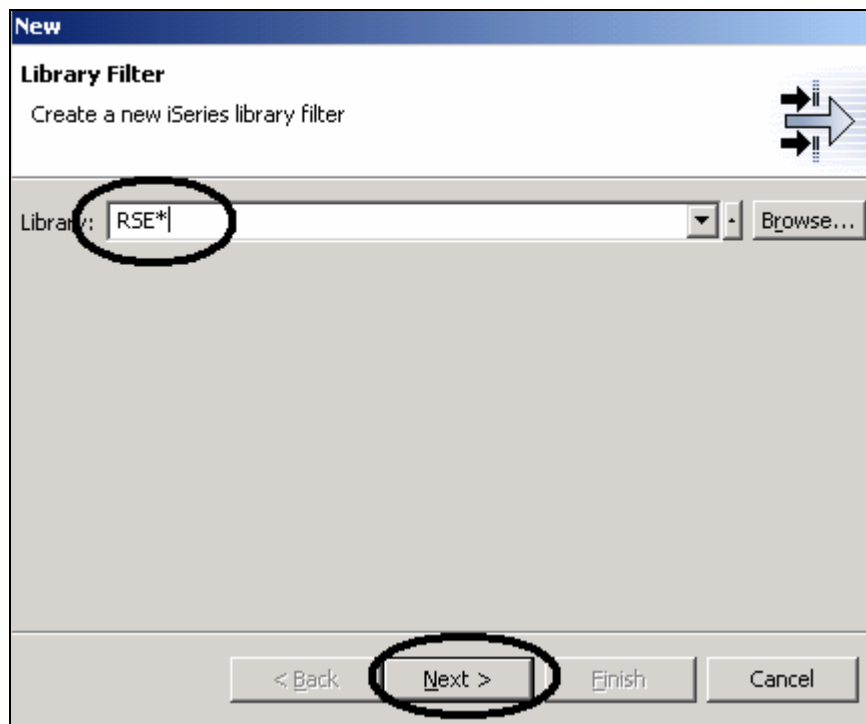
Figure 71: Specify a filter string

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

Specify the first filter string that selects the libraries starting with Remote System Explorer:

1. Type RSE* into the Library field, using the * wild card character.

2. Click **Next**

    The following window will show:
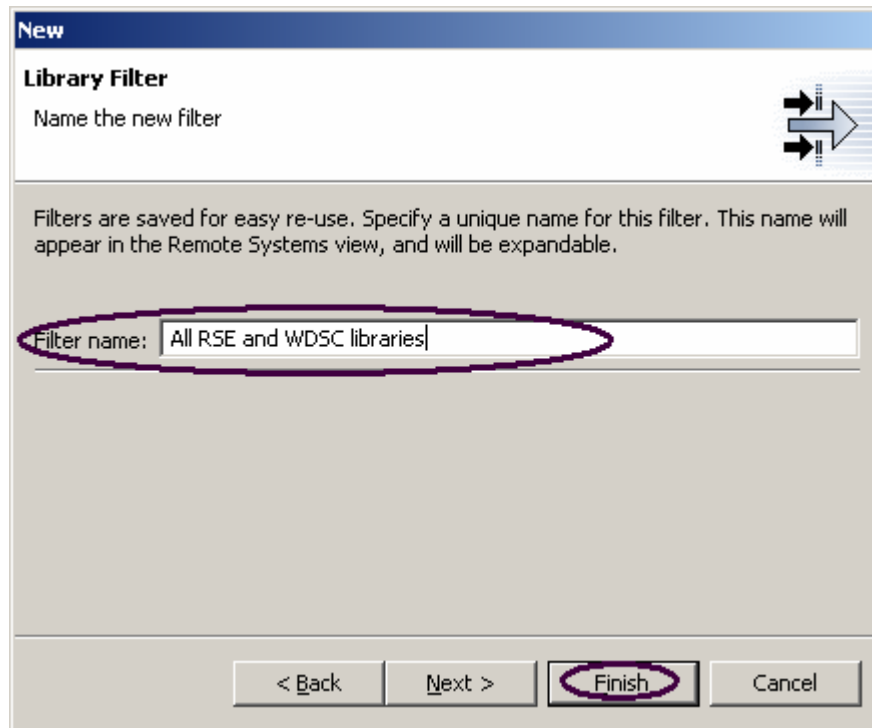
Specify a name for this filter:

3. Type All RSE and WDSC libraries into the **Filter name** field.

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

4. 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 WDSC libraries:
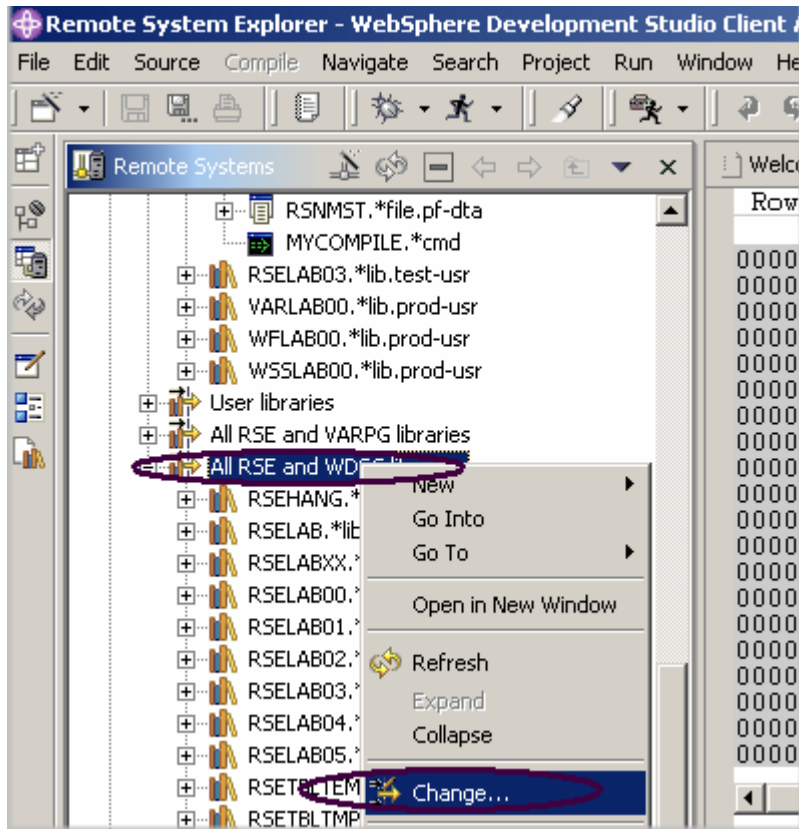
Figure 72: Select the Change option

5. Right click the filter **All RSE and WDSC libraries** and select **Change.**

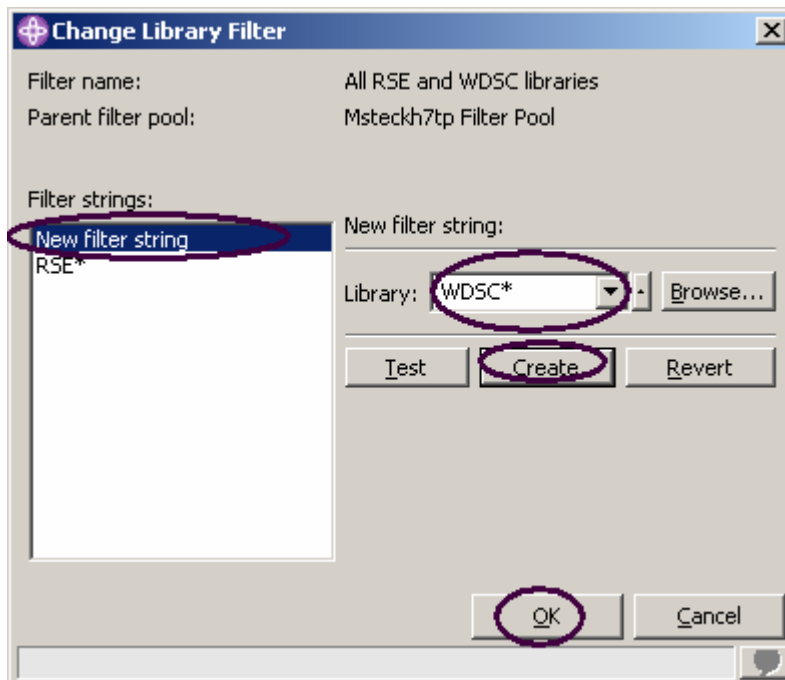The **Change Library Filter** window appears:

Figure 73: Add second filter string

6. Click **New filter string** in the list box

7. Enter WDSC* in the **Library** field

8. Click **Create**

    The WDSC* filter string is added to the list box.

9. Click **OK**.

You are now back in the **Remote Systems** view. You will see the list expanded to include your filter.
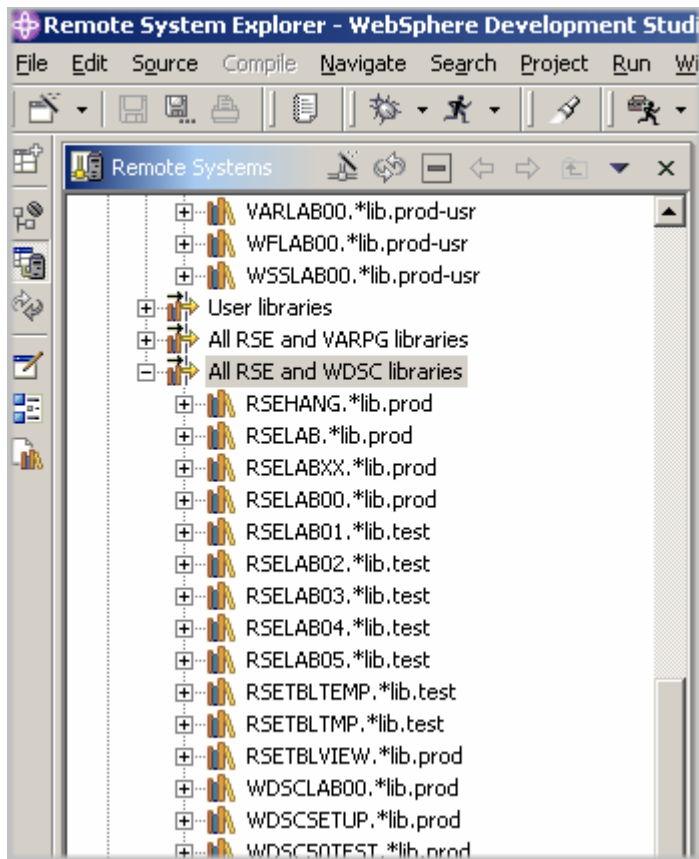
Figure 74: Expanded filter

Now you can work with the libraries directly and can drill down to the object you want to work with directly.

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

Creating an object filter

1. In the **Remote Systems** view, expand your connection and then expand **iSeries Objects**.
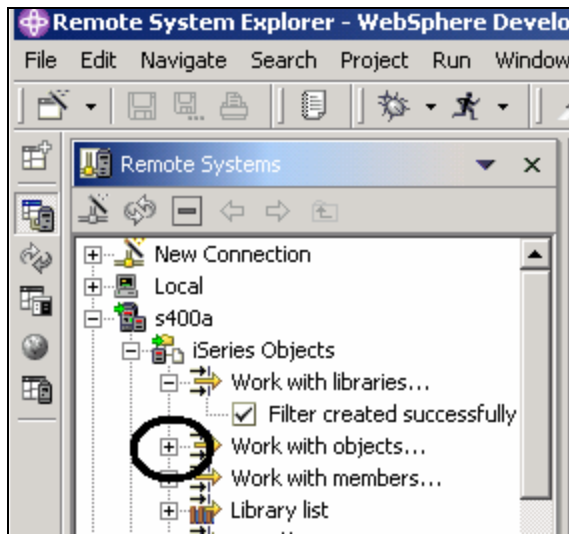
Figure 75: Create object filter

2. Expand **Work with objects**. You can also right-click **iSeries Objects** and select **New > Object** filter.

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

   This will show the **New Object Filter** window

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

Figure 76: Specify filter string

3. Enter RSELABXX in the **Library** field

4. Click **Browse** beside the **Object type** field

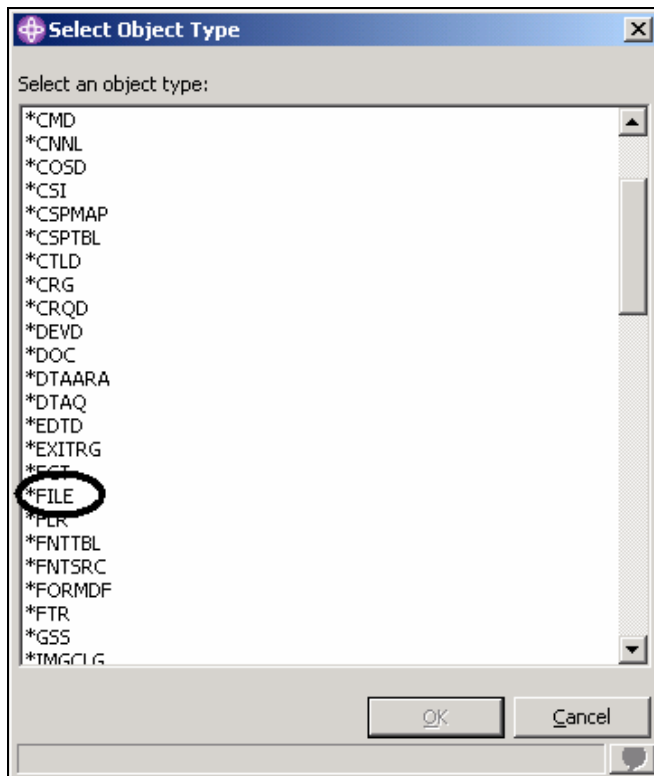*Figure 77: Select Object Type*

5. Select the ***File** object type from the list

6. Click **OK**.

   Back in the Object filter window:

Figure 78: Browse for Object Attribute

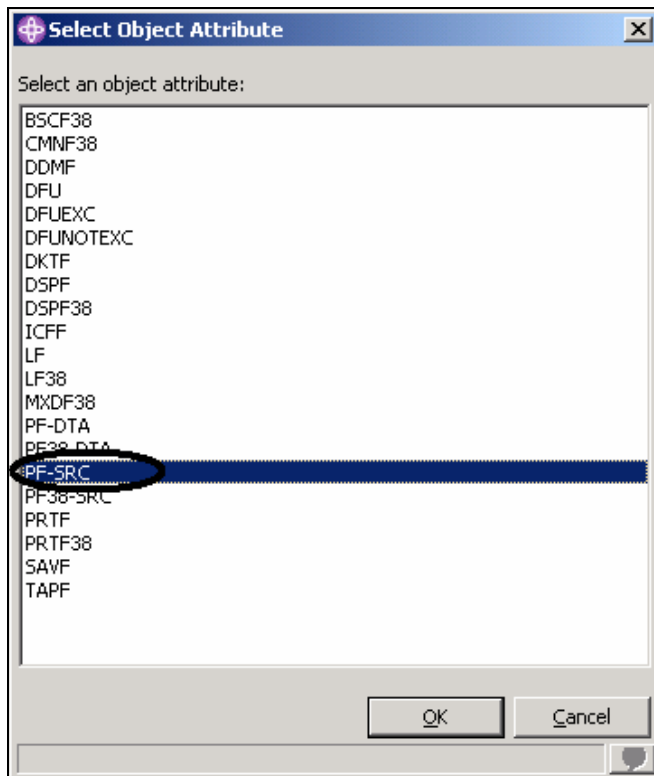7.   Click **Browse** beside the **Object Attribute** field

Figure 79: Select filter Object Attribute

8. Select **PF-SRC** from the Object Attribute list

9. Click **OK**

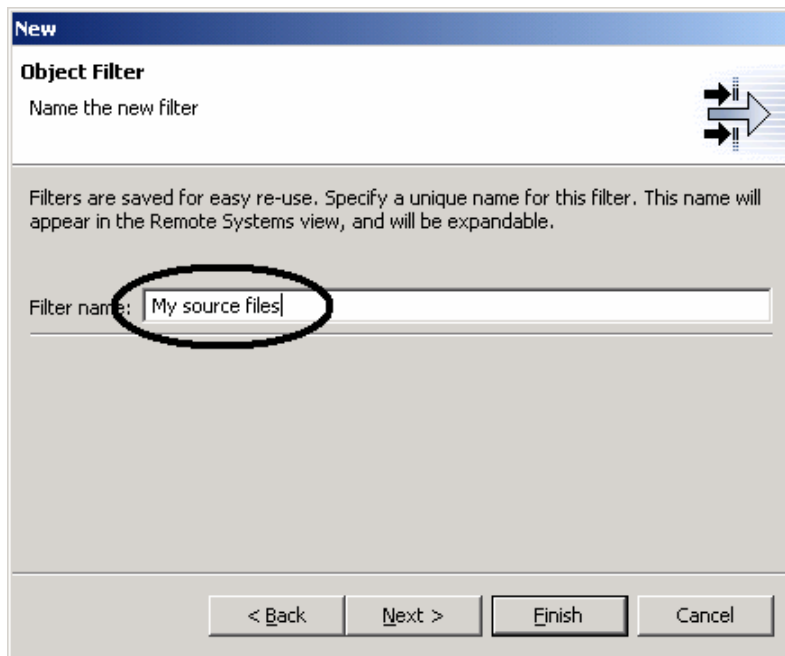10. On the Object Filter window click **Next**.

Figure 80: Specify filter name

11. Specify the Filter name: `My source files`

12. Click **Finish**.

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

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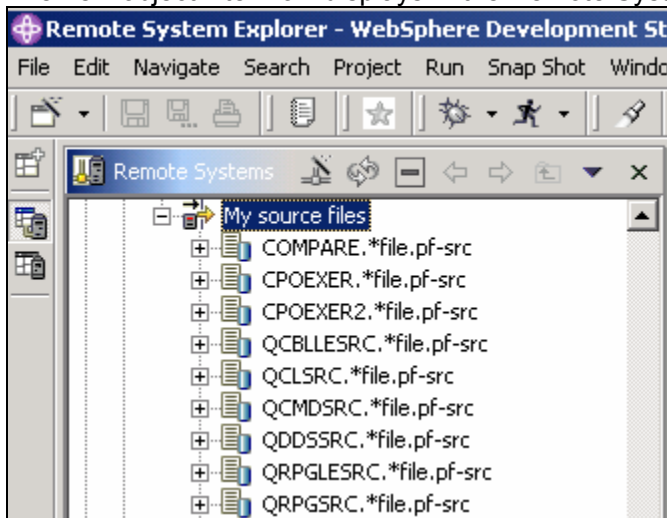

Figure 81:Object filter shows all source files in library

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.

Tip:. If you have been using the Remote Systems Explorer for some time, your workspace may contain too many filters to easily navigate. In this case, filters can be partitioned into pools, where one pool contains filters for many different elements. For example, one filter pool may contain filters for your accounts receivable program while another contains filters for your payroll program.

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

Assuming you want to edit the member PAYROLLC in QCBLLESRC you:

1. Expand **QCBLLESRC**

2. Right-click member **PAYROLLC**

3. Select **Open with > Remote Systems LPEX Editor**

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.

Creating a user action

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

To create a user action:

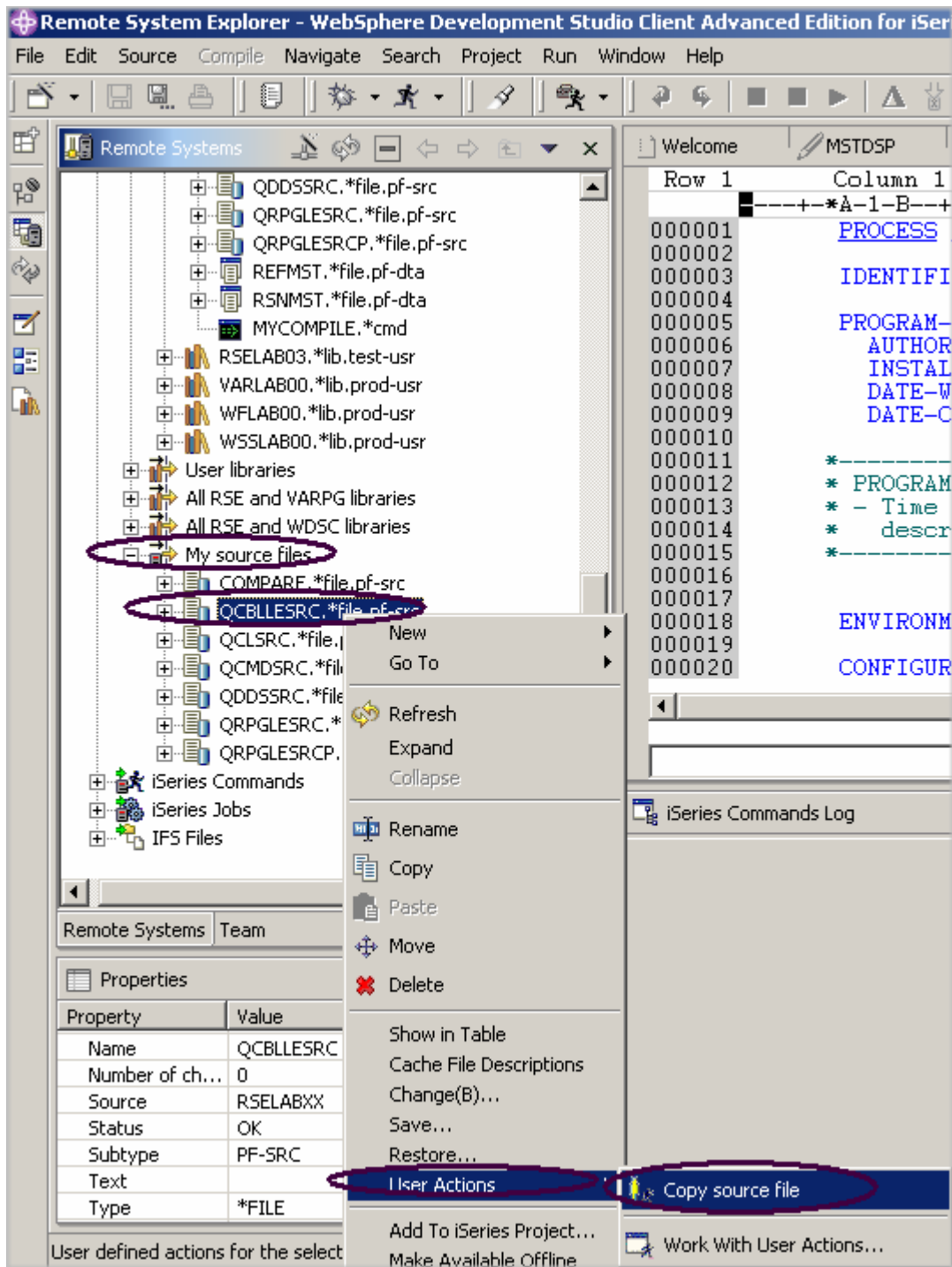1. Expand your iSeries connection and expand **iSeries Object**.

Figure 82: Work with user actions

2. Expand **Library list** filter

3. Right-click **RSELABxx**

4. Select **User Actions > Work with User Actions**

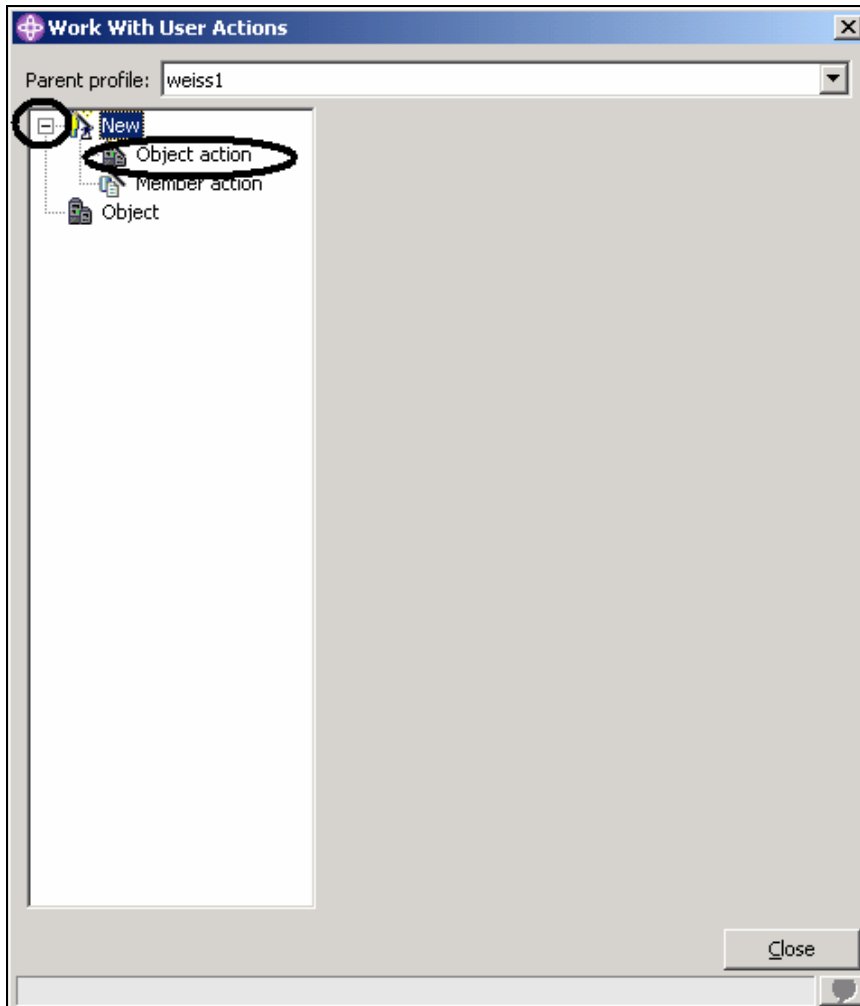   You will see the **Work with User Actions** window:

Figure 83: Work with User Actions window

5. Expand **New** in the list, if it is not expanded already
6. Click **Object action**

   The **Work with User Actions** window appears

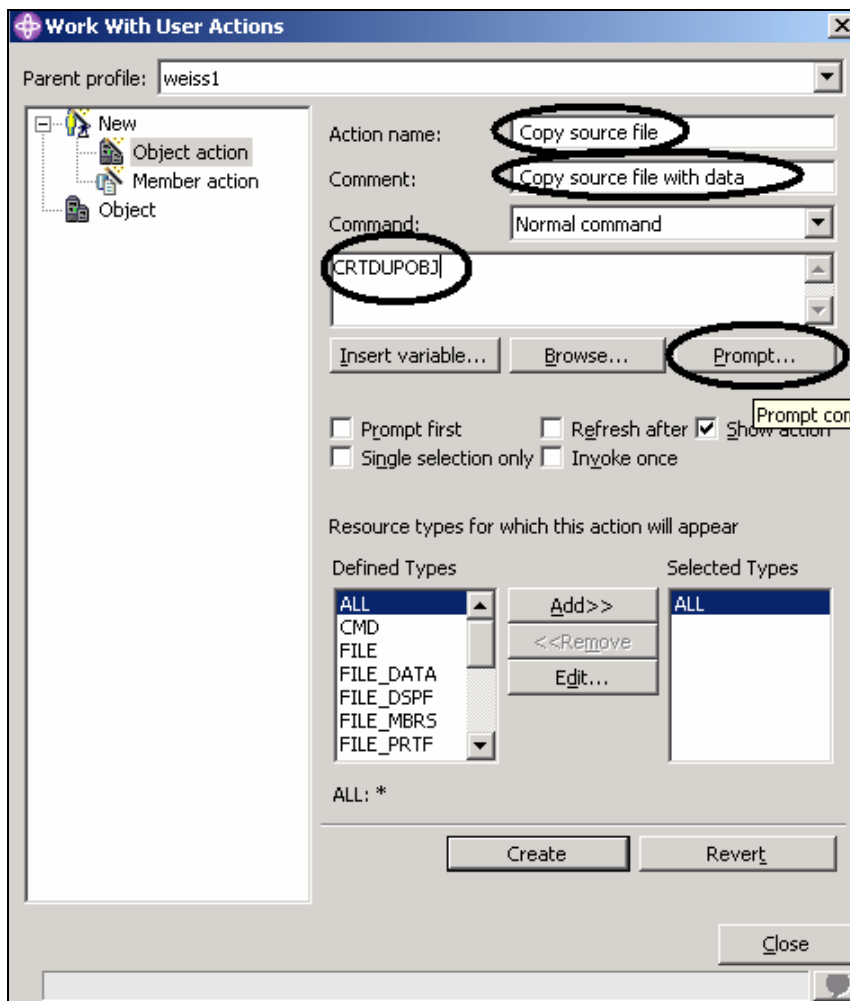*Figure 84: Create an Object action*

Now 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. Enter a name for the user action in the **Action** field: `Copy source file`

8. Enter a comment in the **Comment** field

9. Key in the command to execute `CRTDUPOBJ`

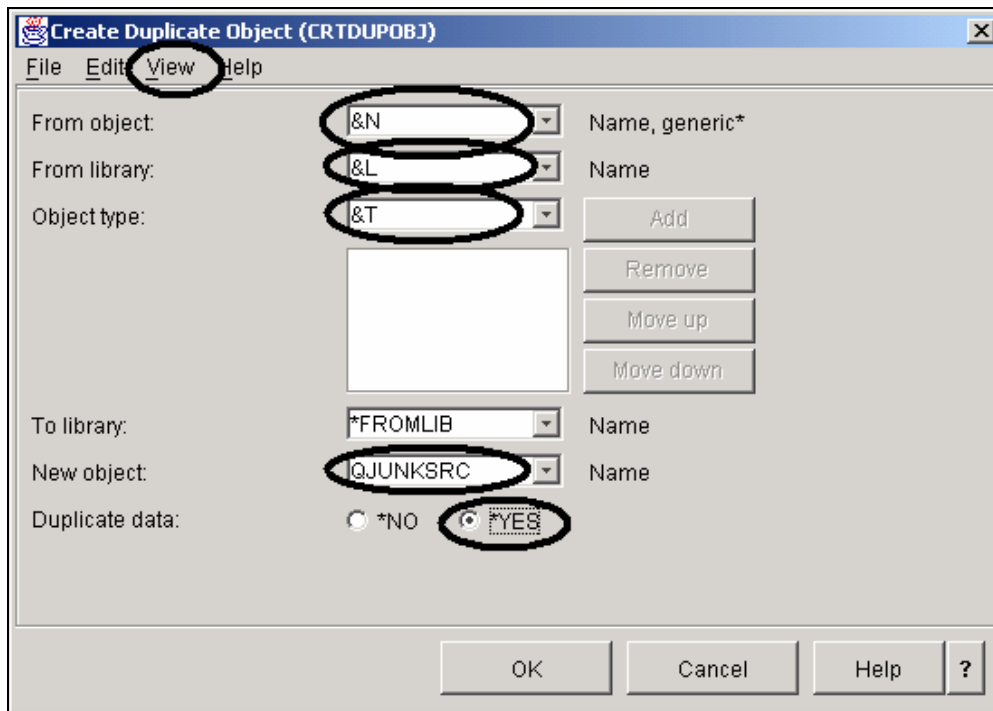10. Click **Prompt** to get the command prompter for this command

Figure 85: Prompt for CRTDUPOBJ command

This is the command you will be running:

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

11. Enter &N in the **From Object** field, to indicate to use the name of the selected object in the Remote Systems view

12. Enter &L in the **From Library** field, to pick up the library name form the selected object

13. Enter &T in the **Object Type** field, to pick up the object type from the selected object

14. Enter QJUNKSRC in the **New object** field

    Now you get the additional Duplicate Data parameter:

15. Select the **View** option on the menu bar of this window

16. Select the **All parameters** option from the pull down menu

    Now the **Duplicate Data** parameter is also shown on the prompt window.

17. Select the **Yes** radio button for the **Duplicate Data** parameter

18. Click **OK**

    You will be back at the Work with User Actions window

    Refresh the Remote Systems view after the action has been run:

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

    Tip: Press **Insert variable** to bring up a list of valid replacement variables with the explanation of what they do.
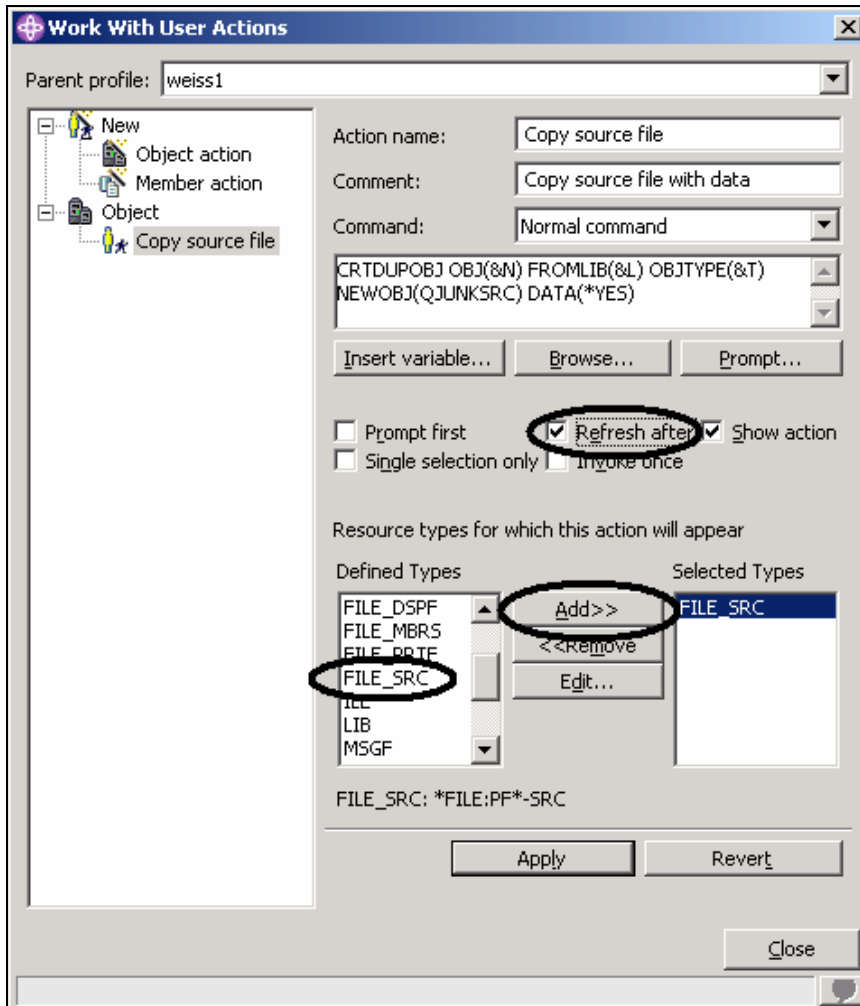

Figure 86: Limit user action to source files

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

20. Locate the **Defined Types** box

21. Select **FILE_SRC** in the list

22. Click **Add** beside the list

    FILE_SRC is now one of the selected types. Actually since you only selected this one it is the only one.

23. Select the **Refresh** check box to refresh the Remote System view after the user action has been performed

24. Click **Close**.
    Now, only when you right-click on a source file, will this user action appear on the pop-up menu selected. For any other object type it will not appear.

    Back in the workbench and the Remote System Explorer perspective, give it a try.
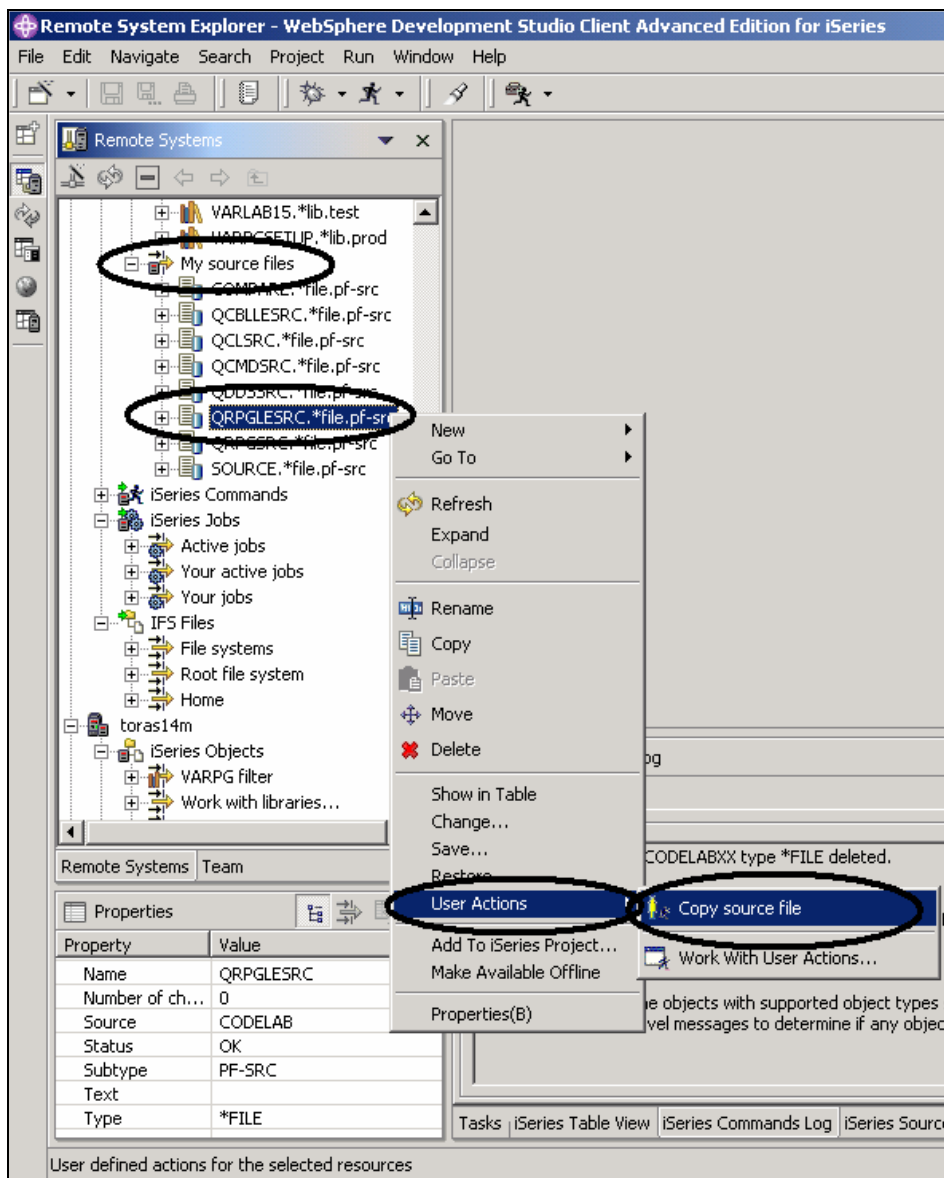
Figure 87: User action shows in pop-up menu

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

26. Expand the **filter**, if it is not already expanded

27. Right click on the **QCBLLESRC** file

28. Select **User Actions > Copy source file**

The file gets duplicated and the list gets refreshed. Your new source file will show in the list.
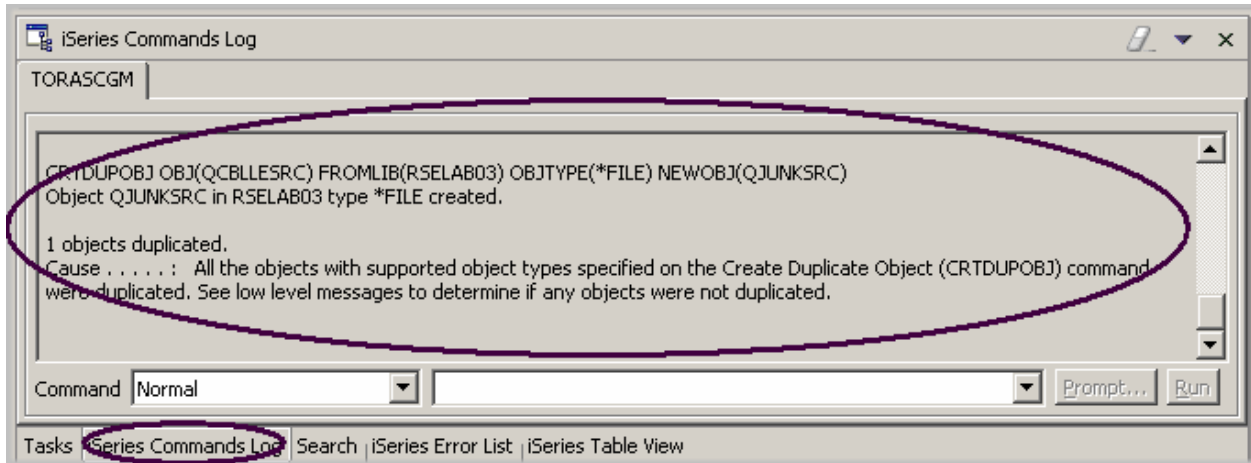
Figure 88: CRTDUPOBJ command in user action ran successful

You can check the messages of the CL commands you are running in the Remote System Explorer

job by looking at the Commands log in the right hand side bottom pane in the workbench.

To delete the source file **QJUNKSRC** that you just created:

29. Right-click **QJUNKSRC**
30. Select **Delete** from the pop-up menu.

Running commands from the Remote System Explorer

The Command entry is part of the Remote Systems view.
1. Check if you have an **iSeries Table View** tab in the bottom right view pane where your command log appeared in the previous task.
2. If you have it click on it
3. If you don't have it
   a) In the Remote Systems view, right-click on your source file filter
   b) Select **Show in table** from the pop-up menu

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

Figure 89: Table view with command entry

In the **command** field:

4. Key in an iSeries command for example **?ADDLIBLE**

    The question mark is there to display a prompt screen

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

5. Click **Run**

    The Command Prompt window for the ADDLIBLE command appears:

Figure 90: Command prompt window

In the Prompt window:

6. Key in `RSELAB00`. That will add this library to the library list of your Remote System Explorer job on the iSeries server.

    The messages field will confirm the successful completion of this command.

    To get to the command history, similar to F9 on the green-screen, press the down arrow on the **Command** field.
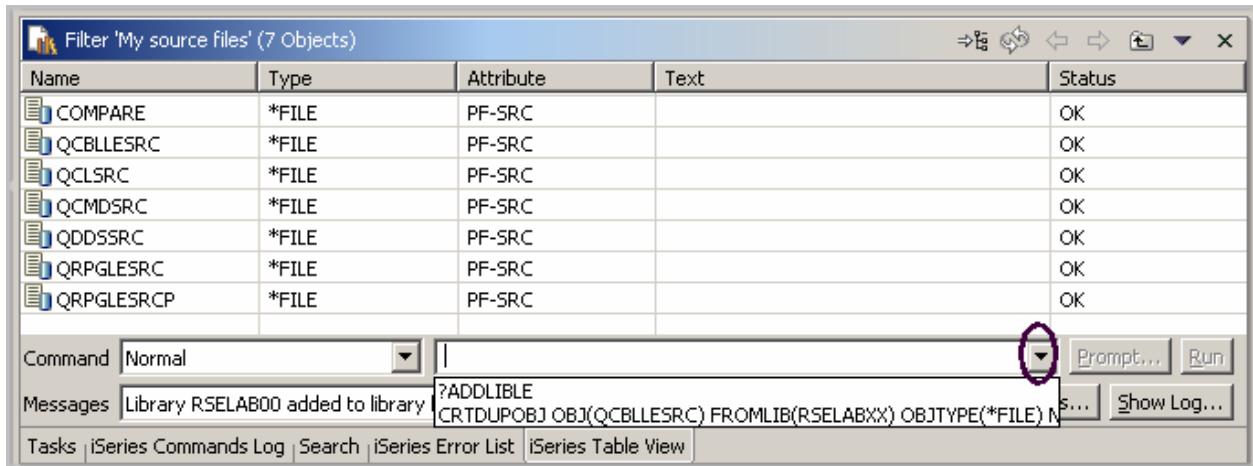
Figure 91: Command history

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.

Complete the checkpoint below to determine if you have mastered the content in this exercise.

Checkpoint

1. The Remote System Explorer is a replacement for:
    A. PDM
    B. SDA
    C. CGU
    D. DFU
2. A filter:
    A. Queries objects on your iSeries system
    B. Easily organizes how you wish to view objects within your system
    C. Shows the native file system objects (QSYS)
    D. Allows you to perform actions against these remote objects
    E. All of the above
3. Expanding Work with libraries corresponds to the:
    A. WKRLIBPDM
    B. WKROBJPDM

    C. WRKMBRPDM

    D. All of the above

4. You give your filters a name because the Remote System Explorer saves them for future use. This is different from PDM that does not save filters. (T, F)

5. You can create:

    A. Library filters

    B. Library list filters

    C. Object filters

    D. Member filters

    E. Job filters

    F. All of the above

6. In PDM you can create user actions in addition to the pre-supplied system actions. In Remote System Explorer you can do the same. (T, F)

7. You can run commands on the iSeries server that the iSeries Table view is connected to. (T, F)

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.

What you just did

In this exercise you used the Remote System Explorer perspective to work with the iSeries objects that you used in the previous exercises. You also learned how easy it was to perform actions and define your own actions. In short, you saw how Remote System Explorer can organize and integrate your work and make that work easier.

Maintaining an iSeries application: Introduction to Remote Systems Explorer Tutorial Summary

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

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

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

Exercise Checkpoint Answers

1	2	3	4	5	6	7	8
1. B	1. D	1. C	1. T	1. D	1. B	1. C	1. A
2. G	2. E	2. A	2. A	2. D	2. D	2. T	2. E
3. G	3. B	3. F	3. T	3. G	3. AC,	3. A	3. A
4. E	4. AA,	4. C	4. D	4. T	BB,	4. C	4. T
5. E	BD,	5. T	5. C	5. T	CA,	5. C	5. F
6. A	CB,	6. E	6. B	6. T	DD,	6. T	6. T
7. T	DC	7. E	7. C	7. F	EF,	7. F	7. T
8. A	5. E	8. C	8. C	8. A	FE,	8. T	
9. T		9. T	9. A	9. D	GG	9. E	
10. E		10. C	10. E	10. D	4. E	10. A	
11. B		11. B	11. D	11. D	5. T		
12. F		12. T	12. D	12. T	6. H		
13. E		13. D	13. B		7. T		
14. T		14. D	14. C		8. T		
15. T			15. T		9. T		
			16. T		10. T		
					11. T		

Notices

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 the 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 in this document. The furnishing of the document does not give you any license to these patents. You can send license inquiries, in writing, to:

Director of Licensing
Intellectual Property & Licensing
International Business Machines Corporation
North Castle Drive, MD – NC119
Armonk, New York 10504-1785
U.S.A

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,K EITHER EXPRESS OR IMPIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR 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 contain 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 service 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.

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:
IBM Canada Ltd. Laboratory
Information Development
B3/KB7/8200/MKM
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.

Programming Interface Information

This publication is intended to help you to create and manage applications and user interfaces on the workstation, in a client/server environment. It contains examples of data and reports that are 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 that are used by an actual business enterprise is entirely coincidental. This publication documents General-Use Programming Interface and Associated Information provided by IBM WebSphere Development Studio Client for iSeries.

Trademarks and Service Marks

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

> Application System/400
> AS/400
> AS/400e
> DB2/400
> COBOL/400
> e-business
> IBM
> iSeries
> Integrated Language Environment
> OS/400
> RPG/400
> VisualAge
> WebSphere

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

Active X, Microsoft, Windows, and Windows NT and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, or other countries, or both. Other company, product, or service names may be trademarks or service marks of others.