

IBM WebSphere Development Studio client (WDSclient) for iSeries V4 SP2

Introduction to CODE

Session Id 230129
Agenda Key 31LF

Inge Weiss and the iSeries Team
iweiss@ca.ibm.com

IBM Toronto Laboratory

Version 4 SP2

homepage: <http://www.ibm.com/software/ad/iseriess>

newsgroup: <news://news.software.ibm.com/ibm.software.code400>

Tenth Edition (September, 2002)

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

Comments concerning this notebook and its usefulness for its intended purpose are welcome. You may send written comments to:

IBM Canada Ltd.

8200 Warden Avenue, Markham, Ontario, L6G 1C7

Attention: Inge Weiss, CODE introduction

or e-mail to: uweiss@ca.ibm.com

Technical Information

For more technical information on CODE or WebSphere Development Studio client for iSeries contact either
Dave Slater at slater@ca.ibm.com
Claus Weiss at weiss@ca.ibm.com

Education

CODE courses:

S6186 CODE/400 for iSeries - Basic (2 days)

S6205 CODE/400 for iSeries - Advanced (1 day)

S6286 iSeries Application Development using WDS for iSeries - Basic (2 days)

Trademarks

IBM is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation.

iSeries
AS/400
VisualAge
DB2/400
ILE
Integrated Language Environment
IBM
OS/400
RPG/400
WebSphere

Trademarks of other companies as shown

'Intel'	'Intel Corporation'
'Microsoft'	'Microsoft Corporation'
'Windows'	'Microsoft Corporation'

Copyright International Business Machines Corporation 1998, 2002. All rights reserved.

This material may not be reproduced in whole or in part without the prior written permission of IBM.

Note to U.S. Government Users - Documentation related to restricted rights - Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Overall Lab Guide

The objective of the CODE Introduction is to explore some of the basic features of the tools that make up CODE, Editor, Designer, and Debugger as well as experience how CODE is integrated into the WDS_c workbench. The example of a small payroll application is used to walk you through the edit, compile, debug cycle. At the end of the lab, the student should know how to:

- Connect to the iSeries.
- Open source in the Editor, make changes, and use different ways to maneuver through the source.
- Invoke the Program Verifier and work with the Error list.
- Start a host compile using the Program Generator.
- Use the DDS tree, the Design Page and the Properties Notebook.
- Invoke the debugger, set breakpoints, display and change variables, and control program execution.
- Create different filters in the Remote Systems Explorer, invoke actions and create new actions.

The exercises for each tool, Editor, Designer, Debugger, and Remote Systems Explorer should be worked on in sequence. You can switch to a different tool though, without completing all exercises for the previous one.

Note: *The pictures in these labs show similar tasks. Some of the names and directories may be different from the environment you are working with.*

Prerequisites

Although not required, familiarity with the RPG language is an asset.

Also, it is helpful if the student is familiar with basic MS Windows operations such as working with the desktop and basic mouse operations such as opening folders and performing drag-and-drop operations.

Overall Lab Guide	5
Prerequisites	6
Introduction to CODE	9
Installing CODE	10
Connecting to the iSeries	11
The CODE Editor	14
CODE invocation choices	15
<i>Using the WebSphere Studio workbench to invoke the CODE Editor</i>	15
<i>Open the Remote Systems Explorer perspective</i>	18
<i>Now you are ready to create a Connection</i>	19
<i>Selecting an iSeries object in the RSE perspective</i>	21
<i>Enabling Extras</i>	25
<i>Opening a second source member</i>	26
Basic Editor Features	28
<i>Column Sensitive Editing</i>	29
<i>SEU Commands</i>	30
<i>Undo and redo</i>	30
Using Language-Sensitive Help	31
Using Prompts	32
Indenting Source	33
Find and Replace	35
Filtering Lines	36
Using the Show Feature	37
Multi-File Search Utility	38
Comparing Files	40
The Navigator	42
Syntax Checking	44
Verifying Your Source	46
<i>Invoking the Program Verifier</i>	46
<i>Fixing Errors</i>	47
Saving a Source Member	47
CODE Remote compile (a.k.a. Program Generator)	48
<i>Compiling Your Source</i>	49
<i>The Command Shell</i>	51
Running the PAYROLL Program	52
The CODE Designer	53
<i>Open a DDS display file member using the WebSphere workbench</i>	53
The DDS Tree	54
The Details Page	55
The Design Page	57
<i>Creating Groups from Existing Records</i>	58
<i>Creating New Screens</i>	60
<i>Field Creation and Design Page Toolbar</i>	61
<i>Switching between Multiple Records</i>	63
<i>Copy and Paste</i>	64

<i>Working with Indicators</i>	65
The Properties Notebook	67
Adding New Keywords	69
Verifying Your Source	71
Switching between Design mode and Edit mode	72
Compiling Your Source	73
Closing CODE Designer	73
The Distributed Debugger	74
Starting the Distributed Debugger	75
Monitoring Variables	81
Running a Program	81
Stepping into a Program	83
<i>The Call Stack</i>	84
The Programs pane	84
Setting Breakpoints in PAYROLLG	85
<i>Deleting a Breakpoint</i>	86
Running the Program	87
Monitoring Variables in PAYROLLG	87
Adding a Storage Monitor	88
Watch Breakpoints	89
Stepping Through the Program	90
Closing the Debug Session	90
Debugger Settings	90
Note: If you want to change Update Production Files or any other debugger settings before running a debug session, you can start the debugger from the CODE editor by selecting the Actions menu, then Debug --> Running application. This will start the debugger with the Attach dialog displayed. Just follow the same steps as above.	92
Closing the Debugger	93
WebSphere Workbench	94
<i>Selecting an iSeries object in the RSE perspective</i>	95
<i>Lets now create an object filter</i>	101
Creating a user action	106
<i>Running commands from the RSE</i>	112

Introduction to CODE

The **IBM WebSphere Development Studio for iSeries** product is a suite of e-business enabling technologies including host and workstation components:

Host Components

- ILE RPG
- ILE COBOL
- ILE C
- ILE C++
- Application Development ToolSet (ADTS) includes PDM, SEU, SDA, RLU

Workstation Components (WebSphere Development Studio client, WDSclient)

- WebSphere Studio Site Developer Advanced

✚ iSeries Plugins

- Remote Systems Explorer
- WebFacing
- Classic Tools
 - VisualAge RPG
 - CODE

The **CoOperative Development Environment**, better known as **CODE**, is a set of integrated development tools that allow you to: create, edit, compile, and maintain your source code; debug programs using a PC connected to an iSeries.

The CODE product includes the following tools:

- **CODE Editor**

A powerful language-sensitive editor that you can easily customize. Token highlighting of source makes the various program elements stand out. It has SEU-like specification prompts for RPG and DDS to help enter column-sensitive fields. Local syntax checking and semantic verification for your RPG, COBOL and DDS source makes sure it will compile cleanly the first time on an iSeries. 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.

- **CODE Program Generator** (Remote compile Capability)

An interface that allows you to submit requests to the iSeries to compile, bind, or build objects on the host. The tool gives you a graphical interface to all the compile options available for all the supported create commands (CRTxxx).

- **CODE Designer**

A rich graphical interface that makes designing or maintaining display file screens, printer file reports, and physical file databases easy and fun.

- **IBM Distributed Debugger**

A source-level debugger that allows you to debug an application running on a host iSeries from your workstation. It provides an interactive graphical interface that makes it easy to debug and test your host programs.

- **CODE Project Organizer (CPO)**

An enhanced and more flexible workstation version of the Program Development Manager (PDM). It ties all the parts of CODE together and allows you to quickly access all the power of CODE and to effectively manage and organize your development projects.

The functionality of CPO is being replaced by the WebSphere Studio workbench, in this Lab you will be using the Explorer perspective, instead of CPO to reflect this fact.

In this session, you will learn some of the basic features and functionality of each of these tools. We are confident that CODE 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. Now let's spend a couple of hours playing and see if you agree.

Installing CODE

The CODE tool of the WebSphere Development Studio Client for iSeries (WDS*c*) product consists of two parts:

1. The 'back-end' which resides on the iSeries.
This part is responsible for handling all the workstation requests such as getting or saving source members, etc. The back-end is shipped with the ADTS host utilities (SEU, PDM, DFU, SDA, ...).
2. The 'front-end' which is installed on your workstation.
These workstation files can be installed from:
 - a local CD drive
 - a LAN drive (assumes that an installable image has been set up on the LAN)
 - an iSeries (assumes that the workstation files have been copied to the iSeries IFS).The workstation install uses the Windows Installer.

The minimum hardware requirements for CODE are an Intel® Pentium II processor or faster with 128MB of memory, a SVGA 800 x 600 monitor, CD-ROM drive, and a mouse or pointing device. The recommended workstation hardware when using the WebSphere Studio workbench with the CODE tools is a workstation with 256MB of memory and a SVGA 1024 x 768 monitor. An install of **WDS*c*** uses about 1.4GB of disk space.

Connecting to the iSeries

Communications between the iSeries and your workstation can be configured for:

- Using an Interactive communications link to the iSeries server. This needs a 5250 emulator to start the interactive job on the iSeries. The communications link will be established thru the emulation session by using an OS/400 command to initiate the session to your workstation. For working developing/testing 5250 applications, this communications link has to be established. You will be using this interactive connection in this Lab.
- Using a non interactive communications link to the iSeries server. This can be used to access source members directly without establishing the interactive connection to the server via a 5250 emulation window. It can be used for all tasks that don't involve 5250 panels. You will need to use the communications console to define the iSeries server you want to access from the workstation

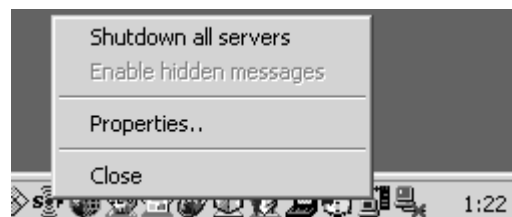
For this lab session, you will use the interactive communications link. On the workstation, the **CODE daemon** needs to be running in order to allow TCP/IP communication with the iSeries. When your PC is restarted after the CODE installation, the CODE daemon is started for you. If you closed the daemon or want to start it manually, you can do so from *Start → Programs → IBM WebSphere Development Studio Client for iSeries → Communications → Communication Daemon*.

- Ensure that the **Code daemon** is running.

The daemon waits and listens for an iSeries to contact it on a specific TCP/IP port and then makes a connection.

An icon will appear in your system tray (bottom right of your desktop).

You can interact with CODE communications by right mouse clicking on the daemon icon and using it's pop-up menu.



- Start a **5250-emulation** session.

- Sign on to the iSeries. Your userid and password should both be **WDSCLABxx** where **xx** is your workstation number (01, 02, etc.).

Note:

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

- Key in the following command

STRCODE RMTLOCNAME(*RESOLVE) CMNTYPE(*TCPIP)

The ***RESOLVE** keyword will get the IP address of your workstation and with this information the **CODE server** will communicate with the **CODE daemon** that runs on your workstation.

```
Start CODE (STRCODE)

Type choices, press Enter.

Host server name . . . . . OS400          Character value
Remote location name . . . . . > *RESOLVE
Communications type . . . . . *PRV          *PRV, *APPC, *TCPIP
```

You should see a screen that has **EVFCLOGO** in the upper left-hand corner. The **CODE** server is now active and waiting for commands from the workstation.



There is a different way to start the CODE server, just for your information we describe it here as well.

If you know your workstation's IP hostname, you could specify it directly in the STRCODE command:

```
STRCODE RMTLOCNAME(PC_hostname) CMNTYPE(*TCPIP)
```

```
Start CODE (STRCODE)

Type choices, press Enter.

Host server name . . . . . OS400          Character value
Remote location name . . . . . PC_hostname
Communications type . . . . . *TCPIP      *PRV, *APPC, *TCPIP
```

The PC_hostname would be either the TCP/IP host name of your PC or its dotted IP address in single quotes. For example '9.21.99.99'. You can determine your PC host name, by typing :

hostname
at a DOS command line.

Note:
If you are still on an OS/400 version lower than V5, the ***Resolve** keyword is not supported. You could instead use the **STRCODETCP** command.

This will call a CL program which automatically figures out which IP address your emulator is using and invokes the STRCODE command that is shipped with the product. This CL program can be found in the CODELABxx library, its source in the QCLSRC source file.

CODE does not require you to be continuously connected to an iSeries -- many of its features are designed to function in 'disconnected' mode as well. This gives you the ability to develop your applications at home or while on the road (oh joy!). In fact, we've added several features, like the caching of iSeries information (copy files, database reference fields, etc.) on your workstation, to help you work in 'disconnected' mode.

The CODE Editor

The CODE Editor has many powerful features that make it the premiere editor for iSeries programmers:

- It supports RPG/400 and ILE RPG, COBOL and ILE COBOL, C, C++, DDS, CL, REXX, HTML and Java language-sensitive editing.
- Seamless access to iSeries source members, ADM parts, and local files.
- Token highlighting of source code elements.
- SEU-like prompting to help you edit column-sensitive languages such as RPG and DDS.
- Built-in syntax checking and program verification (a compile with no object generation) on the workstation for RPG, COBOL, and DDS. With this you can work while disconnected from an iSeries -- saving iSeries cycles and saving you time.
- Lots of on-line help including programming guides and language references.

In this section you will be introduced to some of the features which make the CODE Editor so powerful. For the purpose of the exercise we will be using some ILE RPG source. Don't worry if RPG is unfamiliar to you. Let's get started.

CODE invocation choices

The new WDS product with its workbench gives the programmer some more choices how to invoke the editor. You have the choice to either work with the CODE tools in a stand alone environment, or you could use the new WebSphere Studio workbench capabilities of accessing iSeries objects directly and then invoke the CODE tools from the workbench.

Using the WebSphere Studio workbench to invoke the CODE Editor

In this Lab we will use the WebSphere Studio workbench to access iSeries objects, and then start the CODE tools from the workbench.

Before you start using WebSphere Development Studio Client, you will need to reset it. This will delete any changes to the environment due to previous use of the tool on the PC you are working on.

The instructor will tell you how to reset the WebSphere Development Studio Client environment by removing the **WORKSPACE** folder from the **WDSC** folder.

Warning:

If you work on your own system, be careful as this will remove all project information stored in the workbench. You will lose your work if you reset the workbench!

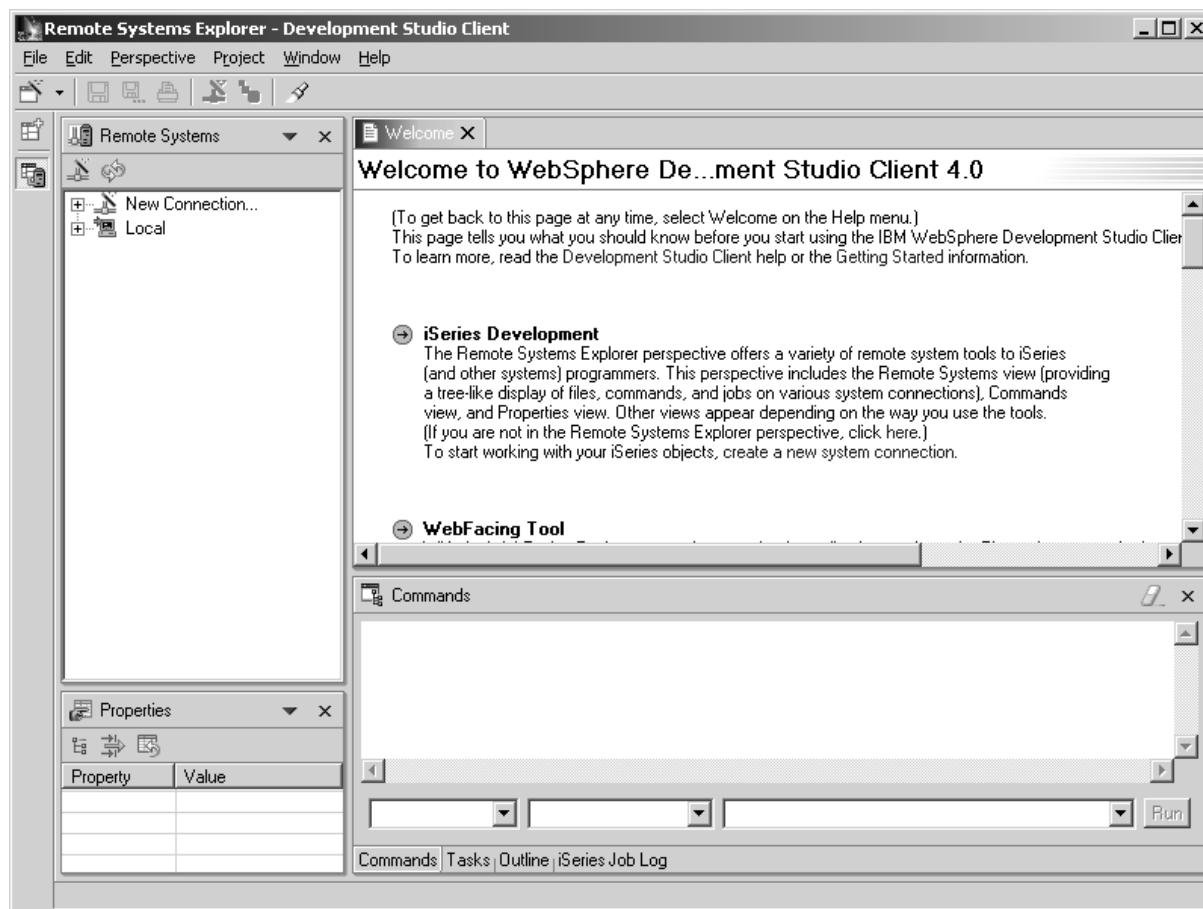
After you have reset the workbench go ahead and start WDS

Invoking WebSphere Development Studio client (WDS)

To start Development Studio client,

- Press the **Start** button on the task bar of your desktop
- Choose **Start→Programs→IBM WebSphere Development Studio Client for iSeries -> IBM WebSphere Studio Site Developer Advanced**

- After a few moments of loading, the workbench appears

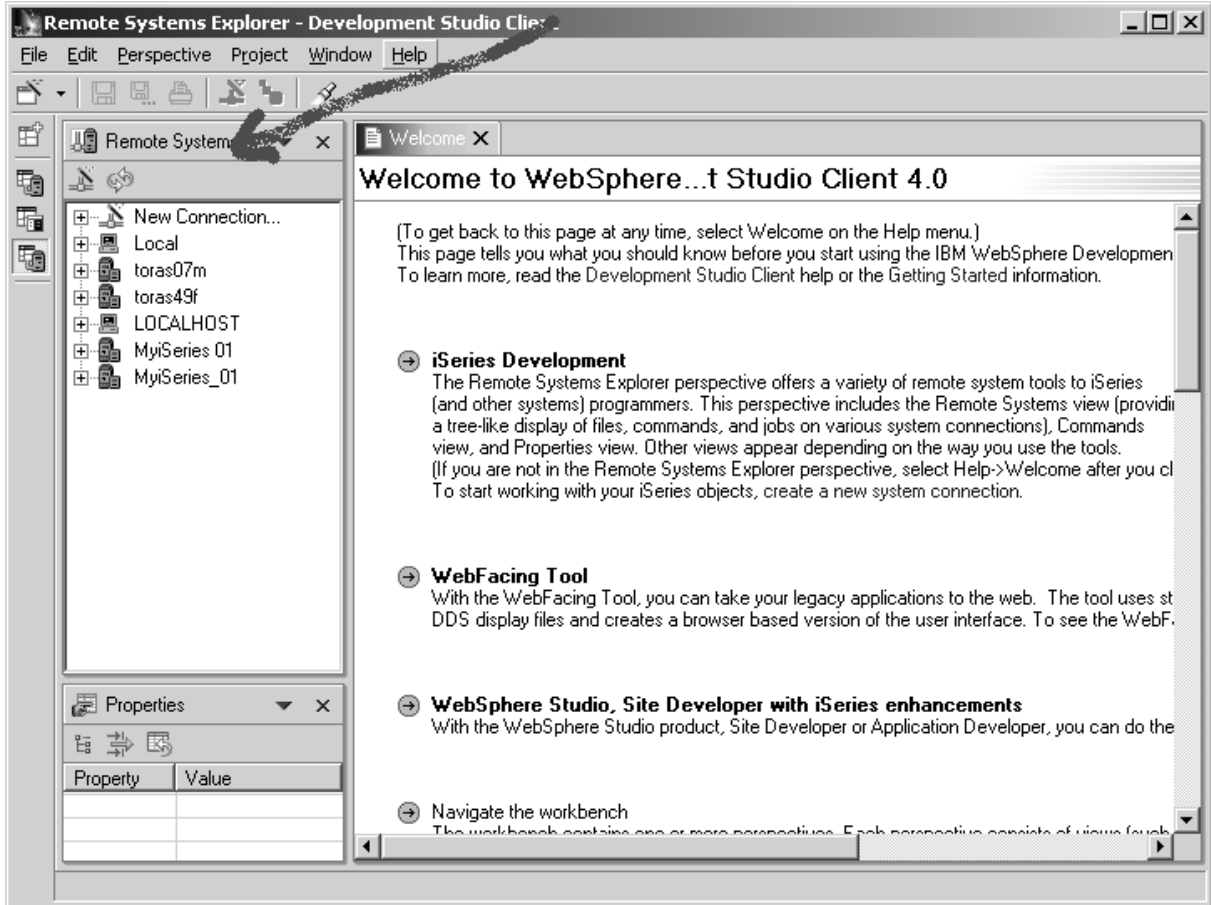


If the workbench has previously been used, it will look different than in the figure above. In case you didn't reset the workbench some pictures will differ from your real lab environment.

You will be working with the **Remote Systems Explorer (RSE)** perspective in the workbench.

A perspective is a specific arrangement of views and tools in the workbench, depending on what role the workbench user has, he/she will use a different perspective. A web developer will use the **web perspective**, a Java developer will use the **Java perspective**, an iSeries developer will use the **Remote Systems Explorer perspective**.

The workbench most likely will already show the Remote Systems Explorer perspective, you might see a different perspective already open in the workbench or no perspective.



- Check for the name of the perspective, the arrow in the figure above indicates where to look for the perspective name.

If the active perspective is already the **Remote Systems Explorer** perspective skip the next steps until you see the heading

Now you are ready to create a connection

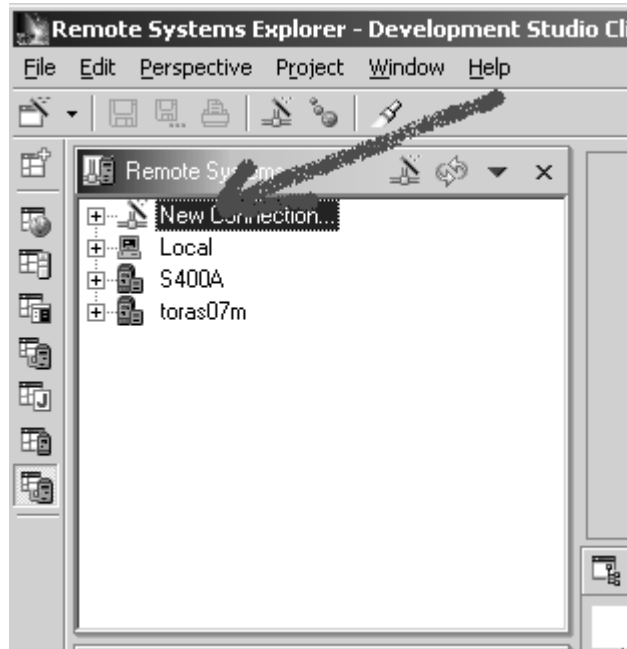
Open the Remote Systems Explorer perspective

If there is **no** perspective open or a **different** perspective open, go through the following steps:

- Start the **Remote Systems Explorer** perspective, by selecting the **Perspective** menu item on the workbench **Menu bar**.
- Then Select the **Open** choice from the pull down menu.
- Select the **Remote Systems Explorer (RSE)** option from the sub menu.

Now you are ready to create a Connection

- Go thru the following steps to create a connection to the iSeries server we will work with in this Lab.



- Locate the **New Connection** node.
- Double click on the **New Connection** node.

The **New Connection** wizard will appear.

New Connection
Remote System Connection
Define connection information

Parent profile: Team

Connection name: S400B

System type: iSeries

Host name: S400B

Default User ID: WDSCLABxx

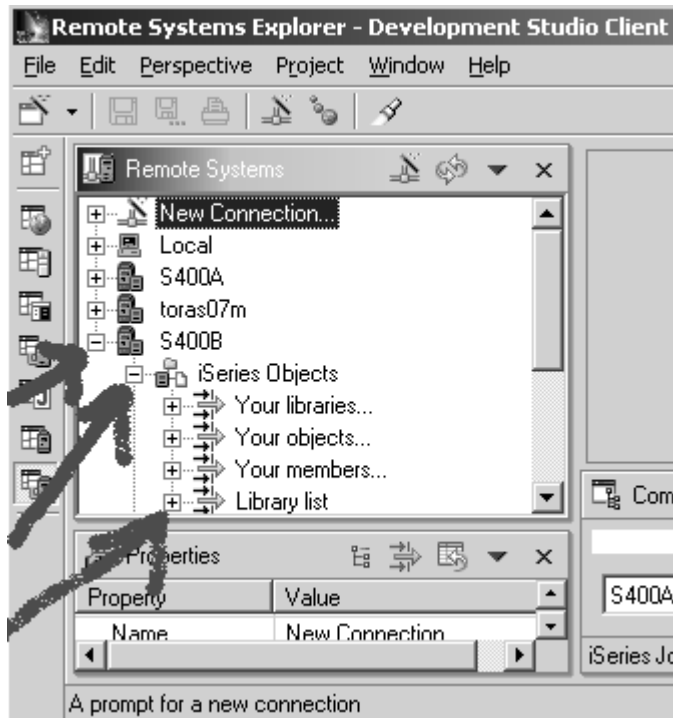
Description:

< Back Next > Finish Cancel

- Leave the parent profile as is, don't change it.
- Specify the **Connection name**, this is the same as your **iSeries host name**.
- Specify the **System type**, which is: **iSeries**.
- Specify the **Host name**, if you don't know it ask your instructor.
- Specify the **Default User ID**, which is **WDSCLABxx**,
xx being your group number, if you are group/team **70**, your userid would be
WDSCLAB70
- Press the **Finish** push button.

Selecting an iSeries object in the RSE perspective

Back in the RSE perspective, you now need to get to the iSeries object you want to work with.



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

- Expand the **connection node** that connects to your iSeries host, by clicking on the **+** sign beside it.
- Expand the **iSeries Objects** node.
- Expand the **Library list** node.

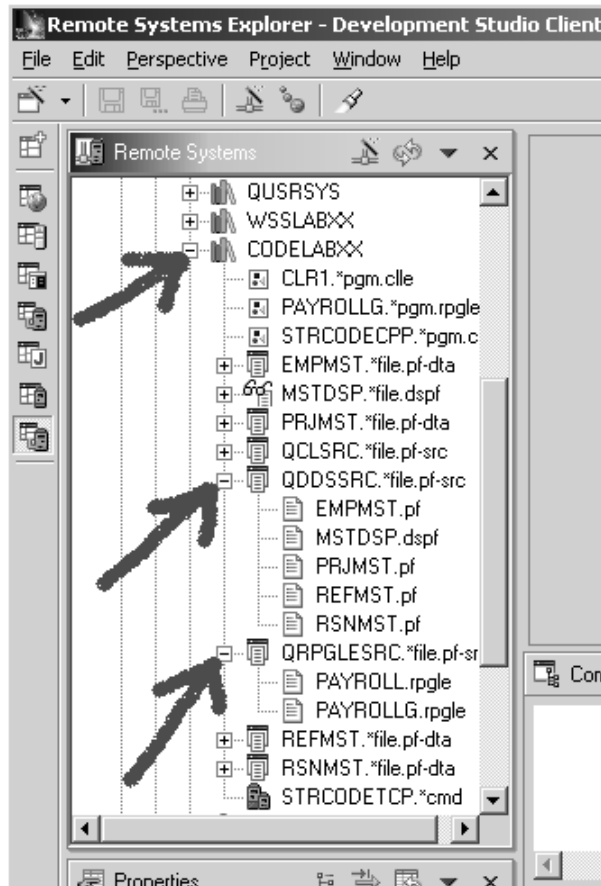
We have setup your **Userid** so that your group/team library has been added to the library list automatically.

You will be prompted to sign on to the iSeries server



- Use Userid **WDSCLABxx** and password **WDSCLABxx**

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



- Locate library **CODELABxx**, **xx** being your group number, and expand it.

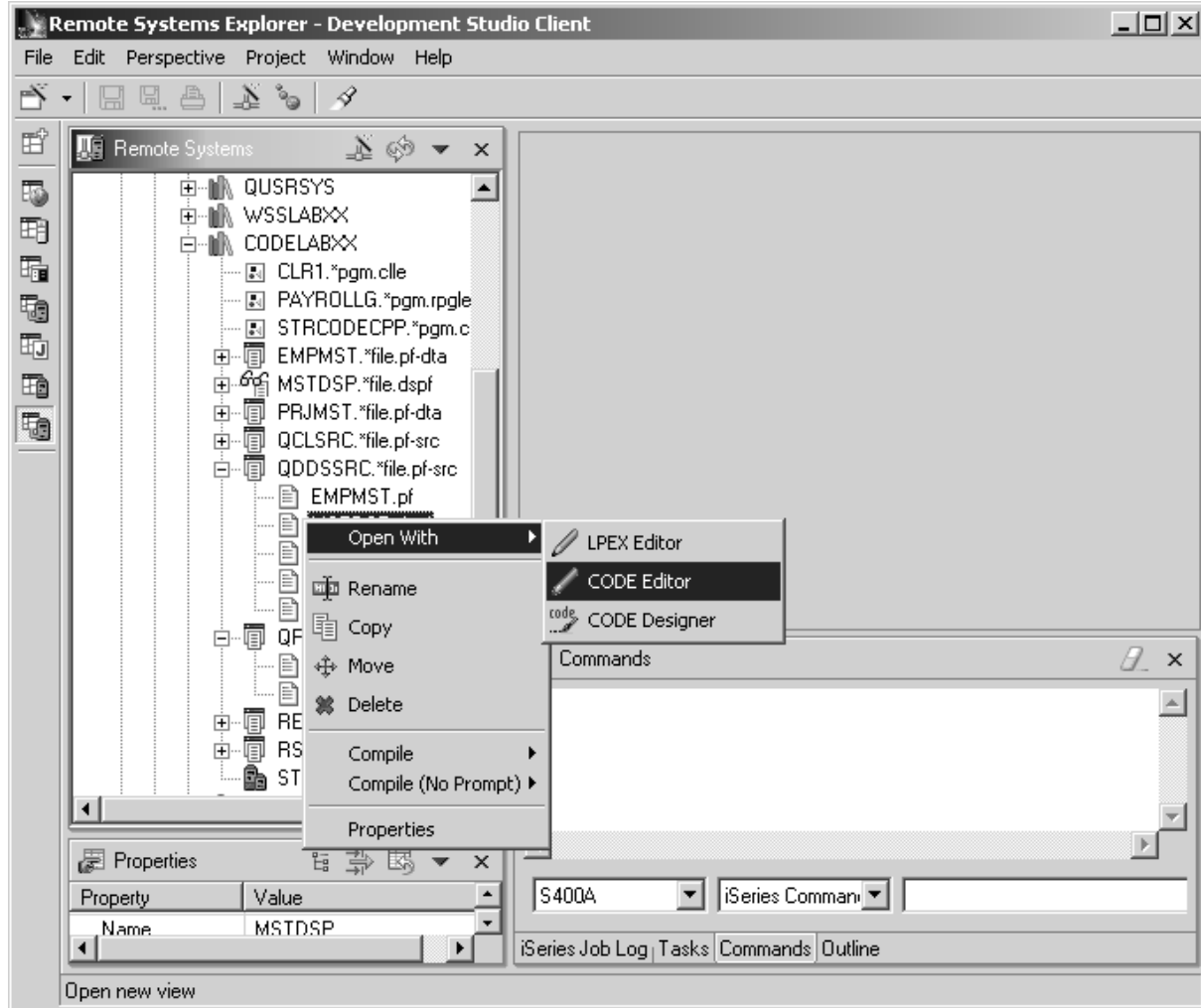
Note:

If you see 2 **CODELABxx** library entries in the list, don't worry, one entry represents the **current library**, the other shows the same library, but in the **library list**.

You will see all objects in this library appear in the expanded list.

- Locate the **QDDSSRC** source file and expand it.
- Locate the **QRPGLSRC** source file and expand it as well.

Now you can access the **members** in these 2 source files.



- Right mouse click on member **MSTDSP** in the QDDSSRC source file.
- Select the **Open with** menu option from the pop up menu.
- Select the **CODE Editor** option from the sub menu.

The CODE editor window will show up with this member loaded.

Enabling Extras

Later on you will use some additional feature in the CODE editor. To enable this, you will now enable these additional features.

- In the CODE editor, check if the Extras menu is listed.

If not,

- select **Actions**→**Enable Extras...** This menu item is only available when **Extras** are disabled.

The **Enable Extras** dialog appears.

- Just press the **OK** push button on this dialog.

The next time the editor gets refreshed, when opening a file for example, the **Extras** menu option will be added to the editor menu bar.

Opening a second source member

We want you to open a second member in the CODE editor, so back in the workbench in the RSE perspective

- Right mouse click on member **PAYROLL** in the QRPGLSRC source file.
- Select the **Open with** menu option from the pop up menu.
- Select the **CODE Editor** option from the sub menu.

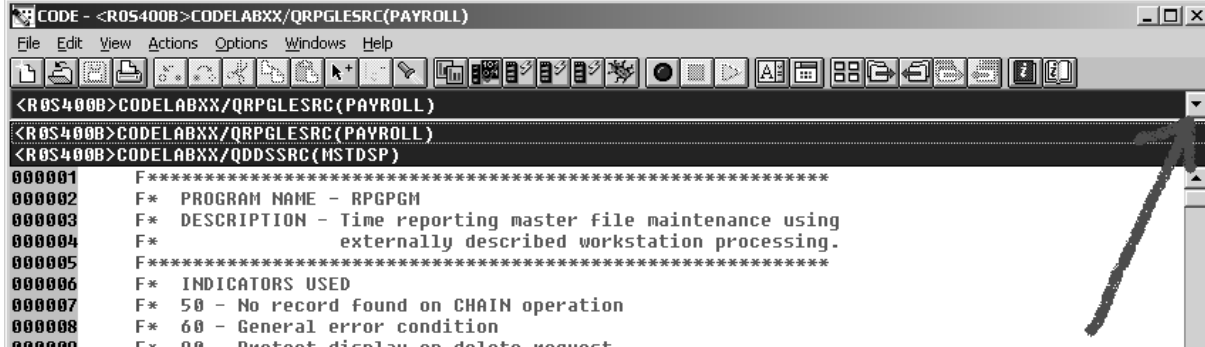
This member will be loaded into the CODE editor as well.

Your editor window will look something like this:

```

CODE - <R0S400B>CODELABXX/QRPGLESRC(PAYROLL)
File Edit View Actions Options Windows Help
<R0S400B>CODELABXX/QRPGLESRC(PAYROLL)
Row 1      Column 1      Replace      Browse
.....1.....+.....2.....+.....3.....+.....4.....+.....5.....+.....6.....+.....7.....+.....8.....+.....9.....+.....
000001      F* *****
000002      F* PROGRAM NAME - RPGPGM
000003      F* DESCRIPTION - Time reporting master file maintenance using
000004      F*                externally described workstation processing.
000005      F* *****
000006      F* INDICATORS USED
000007      F* 50 - No record found on CHAIN operation
000008      F* 60 - General error condition
000009      F* 90 - Protect display on delete request
000010      F* KC - End of job requested
000011      F* KD - Return to application selection
000012      F* KE - Return to employee selection
000013      F* KF - Return to project selection
000014      F* KG - Return to reason code selection
000015      F* LR - Last record
000016      F* *****
000017      F* SUBROUTINES USED
000018      F* EDITSL - Edit application selection display (SELECT)
000019      F* ACDESR - Edit action code for all maintenance requests
000020      F* *****
000021      F* This program uses all externally described files. Files
    
```

To switch back to the DDS source member



- Click at the **drop down list** push button (the arrow in the figure above points to it)

You will see a list of members currently open in the editor. From this list you can quickly select the member you want to work with.

- Select the **MSTDSP** member from the drop down list.

Now you can work with this member.

To get back to the RPG member, use this short cut::

- Press the **ALT key** on your keyboard and hold it
- Press the **---> key** (the key with the arrow pointing to the right) while holding the ALT key. The next member will be active in your Editor window.

This is a fast way to toggle through multiple members that are loaded in the editor.

You can use the **<-- key** to toggle backwards through a loaded member list.

- Make sure you are back at the **RPG member** for the next exercise.

Now you are ready to exploit the editors basic features.

Move on to the next section:

Basic Editor features

Basic Editor Features

The CODE Editor has all the basic functions that you would expect in any serious editor:

- Cut, copy, and paste
- Block marking of lines, characters, or rectangles with copy, move, overlay, and delete operations.
- Powerful find and replace functionality.
- 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 -- 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.

Column Sensitive Editing

CODE provides special support for insertion and deletion in column-sensitive languages.

1. Select the **Options** menu item on the editor menu bar.
2. Select **Language editing** from the sub menu.
3. Select **Column sensitive editing** if it does **not** have a check mark beside it.

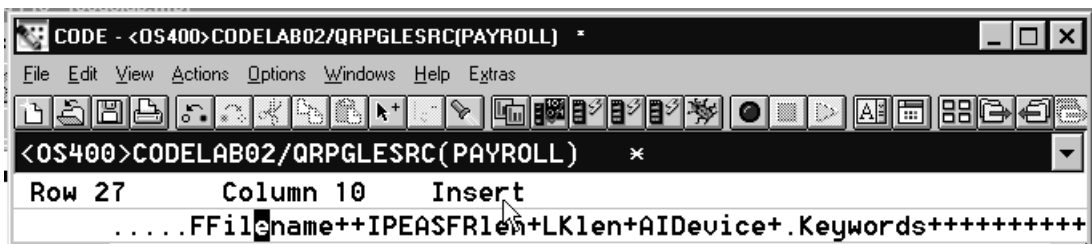
Note:

If you are using any older version of CODE, do the following:

- Press the **Esc** key to move the cursor to the CODE editor's command line.
- Type: **CODE FIELDS ON**
- Press **Enter**.

Now let's see what this does.

1. Move the cursor to **row 27, column 10**.
2. Make sure the editor is in **Insert mode**. If the status area says 'Replace', press the **Insert** key.



3. Hit the **spacebar** 3 times. Notice that only the filename is shifted but none of the other columns to the right are affected.

```
I FMST DSP CF E WORKSTN
  FMPMST UF A E K DISK
  FPRJMST UF A E K DISK
```

4. Press **backspace** 3 times. Once again the filename is affected but no other columns are touched.

SEU Commands

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

- Move the cursor into the **gray sequence number** area to the left of the edit area.
- On any sequence number type **dd**
- Go down a few lines and type **dd** again and press **Enter**.

Notice that the lines have been deleted.

- Now type **i5** in the sequence number area.

Make sure the cursor is within the sequence number area

- Press **Enter**. Five new lines are inserted.

Undo and redo

Now you are going to undo some of the changes you just made to the file.

- ___ 1. From the **Edit** menu, select **Undo**. Notice that the 5 new lines disappear.
- ___ 2. Do another undo action by pressing **Ctrl+Z**. Notice that the deleted lines reappear.
- ___ 3. From the **Edit** menu, select **Redo**. 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:

- ___ 1. From the **File** menu, select **Close view**. A dialog will appear asking you to save the latest changes.
- ___ 2. **Click** the **No** push button.
 - Go back to the workbench to the RSE perspective and reload the **PAYROLL** member in the QRPGLSRC file

NOTE:

You could also use the editor **File** menu to load the **Payroll** member, and select **<OS400>CODELABxx/QRPGLSRC(PAYROLL)** to reload the original source.

Using Language-Sensitive Help

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

- ___ 1. **Position** the cursor over the word **MOVE** in row **56** of the ILE RPG source.
- ___ 2. **Press F1**. Language-sensitive help for the MOVE operation code appears in a browser window.
- ___ 3. Play around in the help window to see what else is available.
- ___ 4. Minimize the Help window.
- ___ 5. Select the **Help** menu, to see what help is available. The **ILE RPG help** menu item will show you all the on-line help that is available for ILE RPG. Select any menu item to find more RPG information. Minimize the Help window when you are done.

Using Prompts

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

- ___ 1. **Move** your cursor to the D-spec on row 33.
- ___ 2. From the **Edit** menu, select **Prompt** (or press F4). A window appears showing the specification line broken down into its individual fields.

Entity name	Externally described	Type of Data Structure	Type of Definition	From	To/Length	Internal Data Type
ERR	<input type="checkbox"/>	<input type="checkbox"/>	S		50	<input type="checkbox"/>

Decimal Positions	Keywords	Comments
	DIM(10) CTDATA PERRCD(1)	

.....DName.....ETDsFrom...To/L...IDc...Keywords.....Comments.....
D ERR S 50 DIM(10) CTDATA PERRCD(1)

- ___ 3. **Move** the cursor to the **Keywords** field using the **Tab** key. **Press F1** or click on the **Help** push button to see help for this field.

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

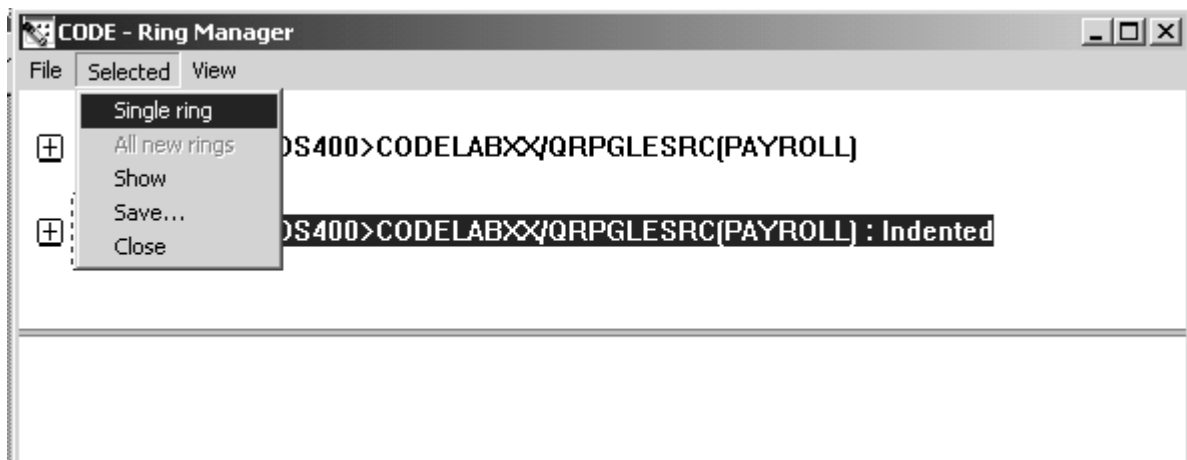
- ___ 1. You will see words in the help that appear in a different color than the regular text. These are help links, and they show that there is additional help available on that word or phrase. Click on any link to see specific help for that item.
- ___ 2. Minimize the Help window.
- ___ 3. **Click** on the **Cancel** push button to close the prompt window without changing your source.

Note: When you use prompts to edit or add to your source and then click on the **OK** push button, the changes are added to the program in the appropriate column and the syntax is checked automatically.

Indenting Source

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

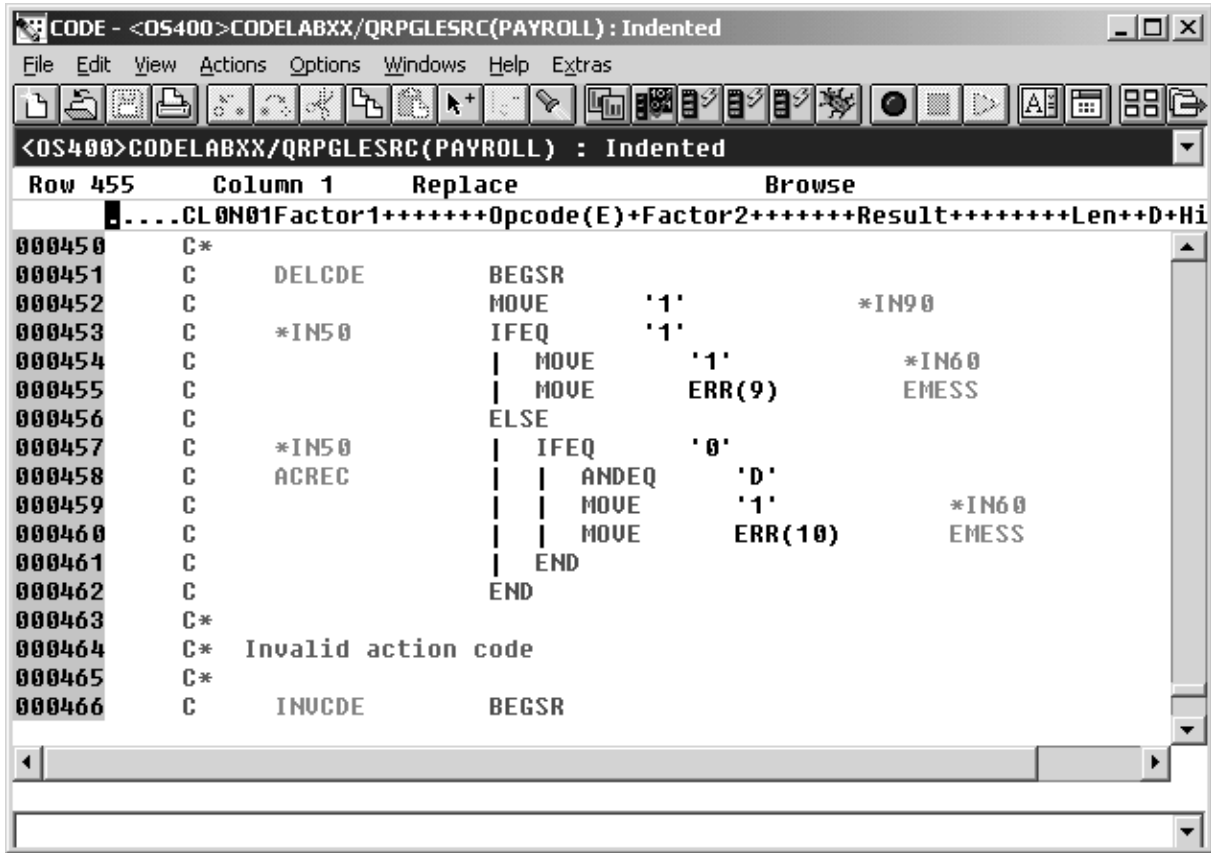
- ___ 1. From the **View** menu, select **Indent**. The indented view is by default displayed in a split view in the bottom half of your edit window.
- ___ 2. To get the indented view displayed as full view, select **Ring manager** from the **Windows** menu. The Ring Manager window comes up.



- ___ 3. In the Ring Manager window select the **Indented** entry, then click on **Single ring** from the **Selected** menu. All views are now on one ring. Close the Ring manager and switch back to the edit window. The indented source is now the current view.

Row 3	Column
*..1....
00001	F*****
00002	F* PROGR
455	F* DESCR
00004	F*
00005	F*****

- ___ 4. **Move** the cursor to row **455** by typing **455** in the sequence number area and press Enter, just as you would using SEU and check what happened to the IFEQ - END blocks.




Use the blue drop-down list below the toolbar to switch between the normal view and indented view to see the difference.

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

- ___ 5. **Switch** to the INDENTED view and **close** it by pressing **F3**.
- ___ 6. **Switch** back to <OS400>CODELABxx/QRPGLESRC(PAYROLL).
(Remember, that you can switch between source members in the editor using the Alt+right arrow or Alt+left arrow keys.)

Find and Replace

The CODE editor also has a powerful find and replace text feature.

- ___ 1. Press **Ctrl+Home** to go to the top of the file.
- ___ 2. From the **Edit** menu, select **Find and replace...** (or press **Ctrl+F**, or click on  in the toolbar). The Find and Replace dialog appears.

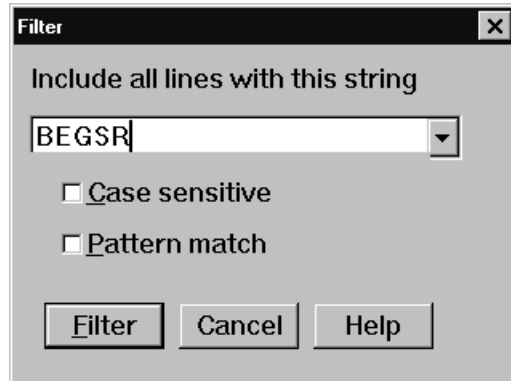


- ___ 3. Click on the **Options** push button to see what we could do if we wanted to -- ie. search all the files loaded in the editor, search only in certain columns, etc.
- ___ 4. Check **Cancel after find** if it is not already checked.
- ___ 5. Click on **OK**.
- ___ 6. In the **Find** entry field, enter **BEGSR** to find the start of a subroutine. Make sure the **Replace with** entry field is blank. You would use this field for text replacement.
- ___ 7. Click on the **Find** push button. The cursor is positioned at the first **BEGSR** in the file.
- ___ 8. Press **Ctrl+N** to go to the next location of **BEGSR** in the file.

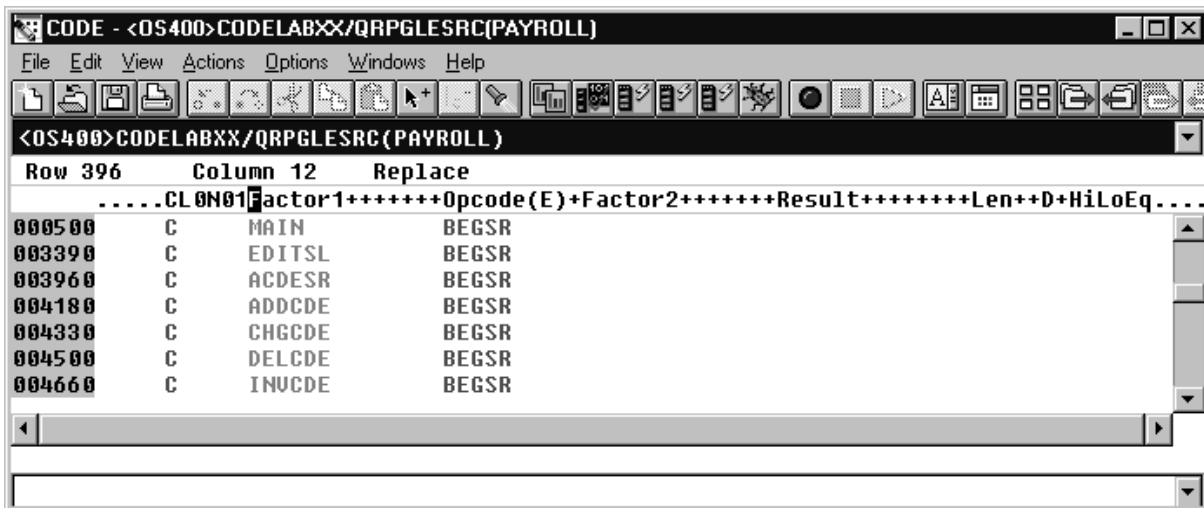
Filtering Lines

The CODE 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.

- ___ 1. From the **View** menu, select **Filter** (or press **Ctrl+I**). The Filter dialog appears.
- ___ 2. Type **BEGSR** in the entry field.



- ___ 3. Click on the **Filter** push button. Only lines that contain the string BEGSR appear in the editor.



- ___ 4. **Move** the cursor down a line or two
- ___ 5. Show all of the source again by selecting **View→Show all** or by pressing **Ctrl+A**.
Your cursor is still positioned on the same line that you moved the cursor to, even though all lines are now showing.

Using the Show Feature

To help you navigate quickly through your ILE RPG source the editor allows you to filter lines based on the line type. In this example, we want to see where all the subroutines are defined in your source:

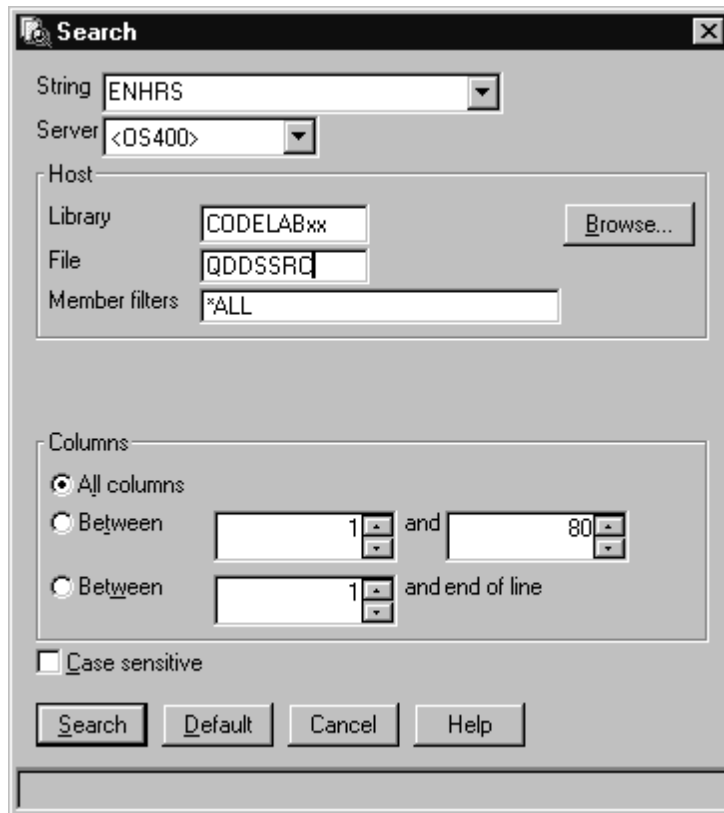
- ___ 1. From the **View** menu, select **Show** and then select **Subroutines**. All subroutines specifications are displayed allowing you to move quickly and easily to the area in your file where the desired subroutine is.
- ___ 2. **Move** your cursor to the line with the subroutine declaration **CHGCDE**. (line 433)
- ___ 3. Show all lines by pressing **Ctrl+A**.

Your cursor is still positioned on the line for the declaration of the subroutine CHGCDE.

Multi-File Search Utility

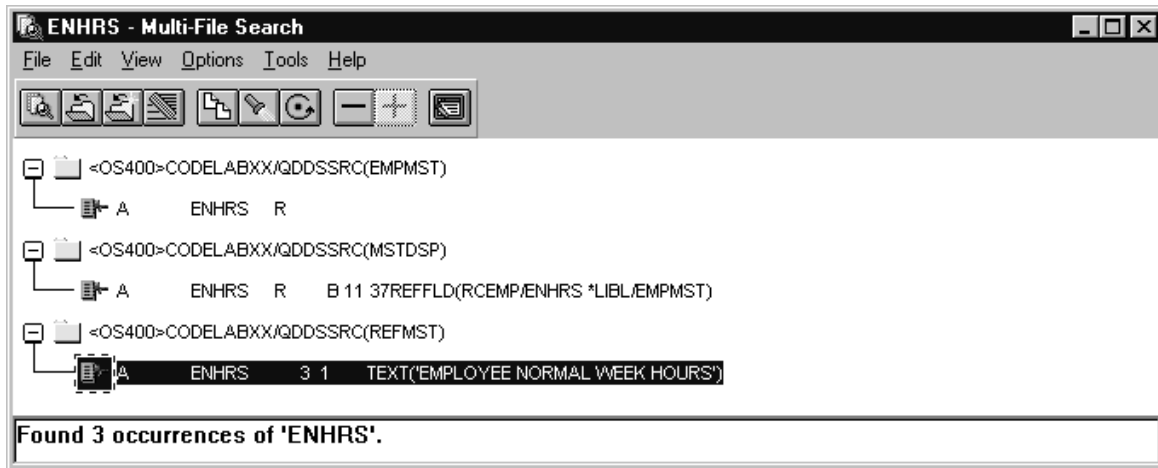
If you would like to search through the members in a source physical file or through the files in a local directory, you can use the CODE Multi-File Search tool.

- ___ 1. From the editor **Windows** menu, select **Multi-file search**. The Search dialog appears.
- ___ 2. In the **String** entry field, type **ENHRS**



- ___ 3. Click on the **Server** combo-box and select **<OS400>**.
- ___ 4. In the **Library** entry field, type **CODELABxx** where **xx** is your workstation number.
- ___ 5. In the **File** entry field, type **QDDSSRC** to search all members in this source physical file.

- ___ 6. Press the **Search** button. The Multi-File Search window lists all the lines in all the files that reference ENHRS.



- ___ 7. **Double-click** on 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 Files

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. You can edit your source directly in the utility -- it contains all of the CODE Editor's features.

___ 1. **Switch** back to <OS400>CODELABxx/QRPGLESRC(PAYROLL) using the blue drop-down list beneath the toolbar.

___ 2. From the **Actions** menu, select **Compare...** The **Compare** dialog appears.

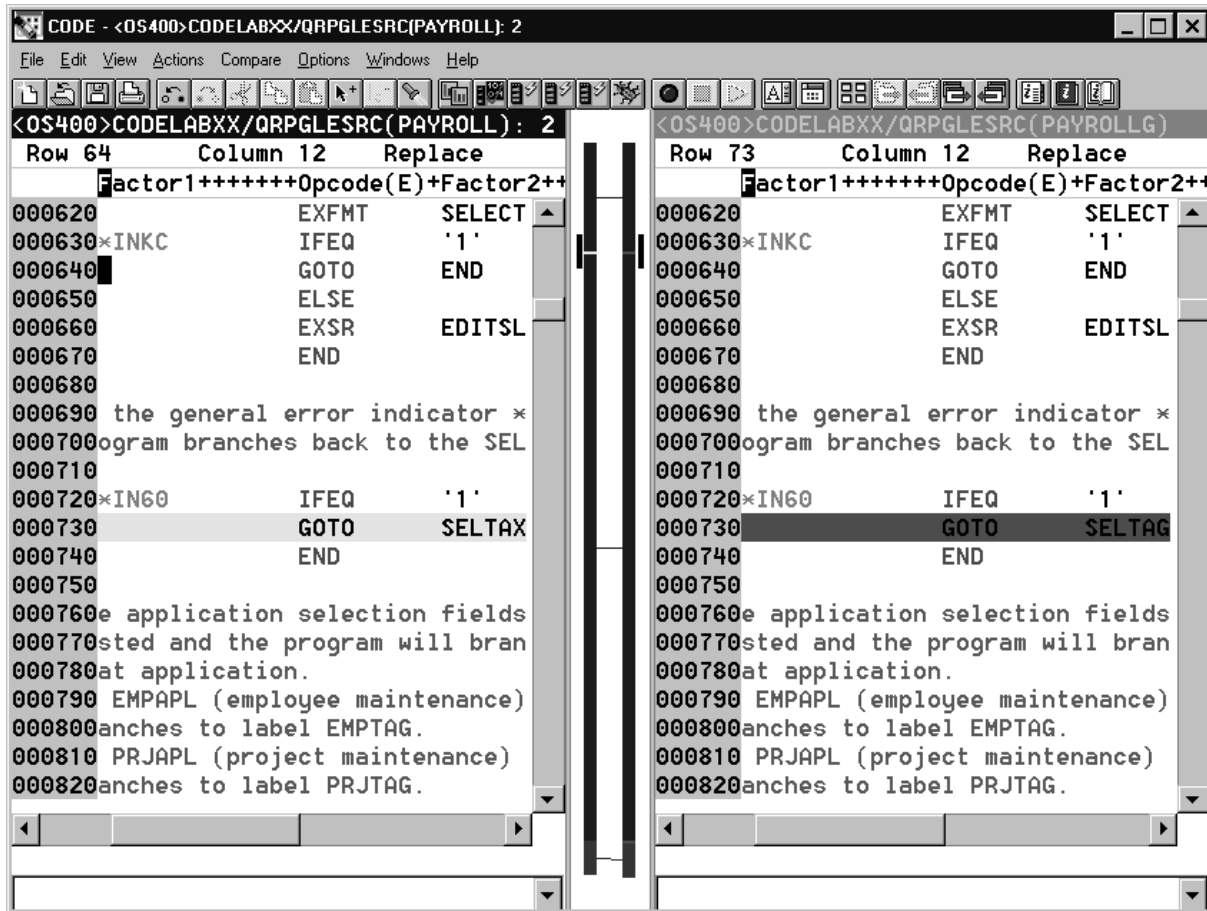
___ 3. In the entry field, type

<>CODELABxx/QRPGLESRC(PAYROLLG)

where <> means use the default host, **xx** is your workstation number. PAYROLLG is a version of the PAYROLL file where some compile errors have been corrected.

___ 4. **Click** on the **Compare** push button. The editor now has the PAYROLL member loaded on the left and member PAYROLLG 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.

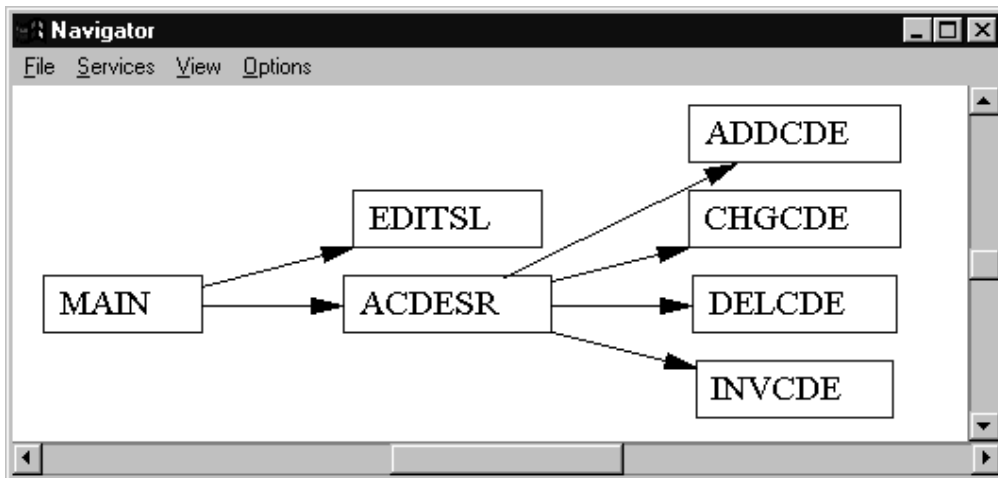


5. Use the vertical scroll bars to move within the files. As you scroll, you will see where the differences are in the members. RPG experts will notice that PAYROLL has some errors in it. We will fix these in a few moments.
6. From the **Compare** menu (which was inserted while performing this action), select **Exit Compare** to go back to the original view.

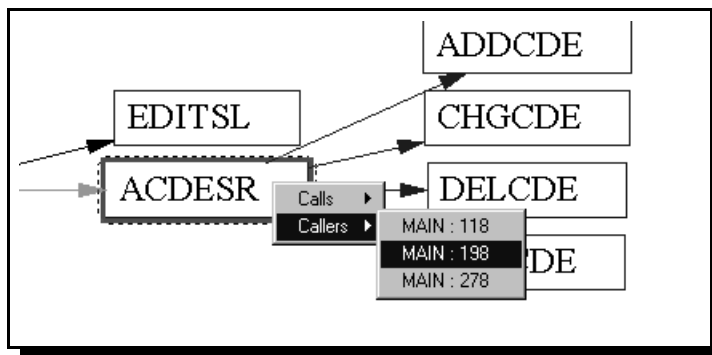
The Navigator

Take full advantage of the Navigator to view how calling and called procedures or functions are related. The arrow direction indicates which procedures or functions are calling and which ones are being called. The Navigator is available for ILE RPG, C, and COBOL languages.

- ___ 1. **Switch** back to <OS400>CODELABxx/QRPGLESRC(PAYROLL) using the blue drop-down list box beneath the toolbar.
- ___ 2. From the **View** menu, select **Navigator** menu and then select **Open**. A Navigator window opens up showing that the MAIN procedure calls EDITSL and ACDESR. The subroutine ACDESR calls ADDCDE, CHGCDE, DELCDE, and INVCDE.



- ___ 3. If you right click on the ACDESR subroutine you can go right to the line where that subroutines is called or where it calls other subroutines.



- ___ 4. From the Navigator's **Services** menu, select **Topology**. This opens a file called NAVIG.OUT that lists the called and calling functions. For large, complex programs these tools are invaluable.

- ___ 5. Press **F3** to close NAVIG.OUT.
- ___ 6. Select the Navigator window and from the **File** menu, select **Close**.

Syntax Checking

One of the powerful features that the CODE 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. In this part of the exercise you will create a syntax error and will be immediately prompted to correct it.

- ___ 1. **Move** the cursor to row 198 which contains 'EXSR ACDESR'. You might already be on that line but if not, you already know different ways of getting there. You could type the line number in the sequence number column, or you could use **Locate Line** (Ctrl+L), **Find and Replace** (Ctrl+F), **Filter** (Ctrl+I) or the **Navigator** function.
- ___ 2. **Append** an **X** to the **EXSR** op-code to make it **EXSRX**.
- ___ 3. **Move** the cursor off of the line. An error message appears to draw attention to the error.

```

001960 C    *INKC      IFEQ      '0'
001970 C    EMPNO     CHAIN     EMPMST      50
001980 C    EXSRX     ACDESR
001980 RNF5014E Operation entry is not valid; specification is ignored.
001990 C    ELSE
002000 C    GOTO      END
002010 C    END
    
```

- ___ 4. **Move** the cursor onto the pink error message.
- ___ 5. **Press F1**. This opens a window with second level help for the error.
- ___ 6. **Minimize** the help window.
- ___ 7. Change **EXSRX** to **EXSR** to correct the error.
- ___ 8. **Move** the cursor off the line you just fixed. The error message is automatically removed from the editor.
- ___ 9. You can toggle automatic syntax checking by using the **Options**→**Language editing**→**Syntax checking** menu item.




Verifying Your Source

Now we get to play with one of the most powerful and unique features of the CODE Editor -- 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 are disconnected from an iSeries. You can do this because CODE 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.

Invoking 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:

1. From the **Actions** menu, select **Verify program** and then select **No prompt** (or click on ). A verify is performed and the **Error List** appears. By default, you will see a number of informational messages that are not shown in the picture below. To filter out all the informational messages toggle the **Options**→**Include**→**Information** menu.

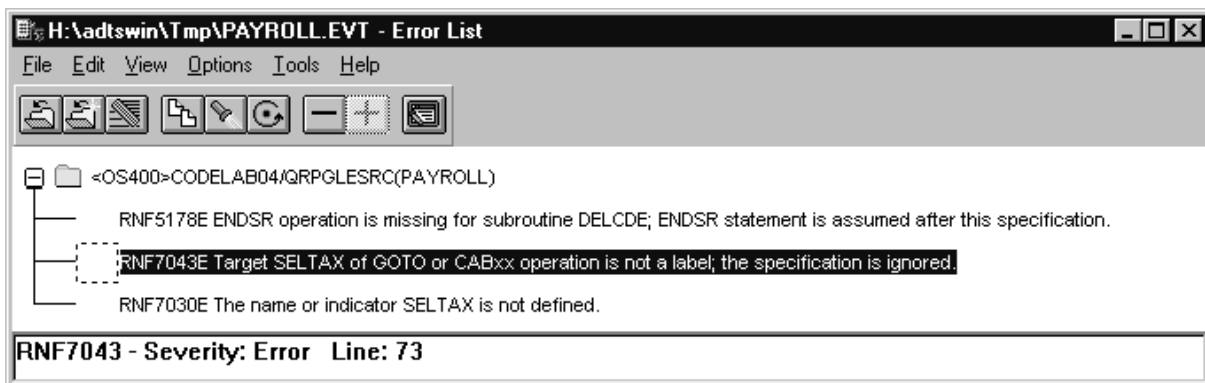


To determine the severity and line number of an error:

1. **Click** on any error in the list. The status line at the bottom displays the severity of the error and the line number in the source.
2. **Press F1**. Second-level help for that error message appears. Read the help.
3. Minimize the Help window.

To insert an error message in the source and place your cursor there.

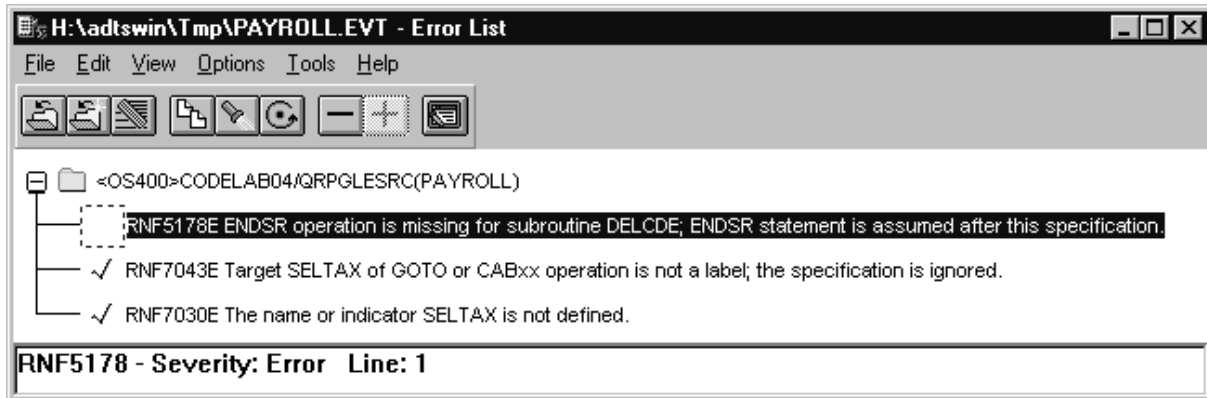
1. **Double-click** on the error **RNF7043E**. The error is inserted into the editor and you are taken to the offending line.



Fixing Errors


Now you will use both the editor and the error list to fix the errors found when verifying the program.

- ___ 1. The error on line **73** is a typo. **SELTAX** should really be **SELTAG**. Make the appropriate change.
- ___ 2. **Switch** back to the Error List window. Several of the errors now have a checkmark beside them. This indicates that you have modified these lines. The only serious remaining error is **RNF5178E**. This error is caused by a missing **ENDSR**.



- ___ 3. **Move** the cursor to row **462** in the editor. From the **View** menu, select **Indent** if you are having troubles seeing the error (or look at the picture in the **Indenting Source** section again).
- ___ 4. Place the cursor in the text area of line 462 and **press Enter** to insert a new line.
- ___ 5. On row **463**, type
C ENSR
(**Hint:** use the Tab key to quickly move to the appropriate column)
All the non-informational errors are now fixed.

Saving a Source Member

Before you lose any of your changes, it's a good idea to save them. You can save the member from either the **File** menu, the toolbar (click on ), or:

- ___ 1. Press **Ctrl+S**. Changes are uploaded to the iSeries
- ___ 2. Verify your source again, everything should be OK, you are ready to compile.

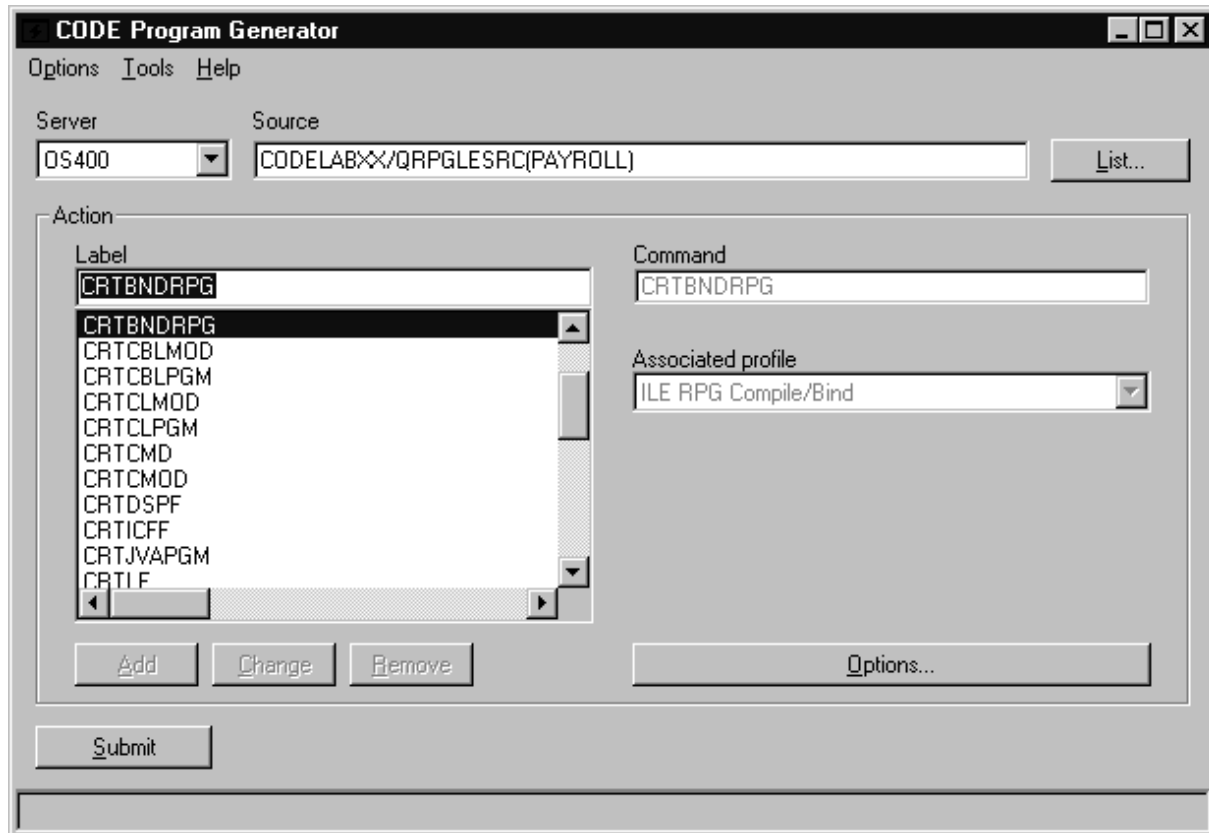
CODE Remote compile (a.k.a. Program Generator)

The remote compile capability is another feature within CODE. It gives you a workstation interface to submit requests to the 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. It remembers your last compile options so that they only need to be entered once. The remote compile tool also supports several features for Java, such as compiling on the iSeries host. 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 Error List window which behaves just as it did when you did a program verify.

Starting the Remote compile

Whenever you request that source be compiled from within the editor, the Remote compile tool is started.

1. From the editor's **Actions** menu, select **Compile** and then select **Prompt...** The Remote Compile dialog appears.

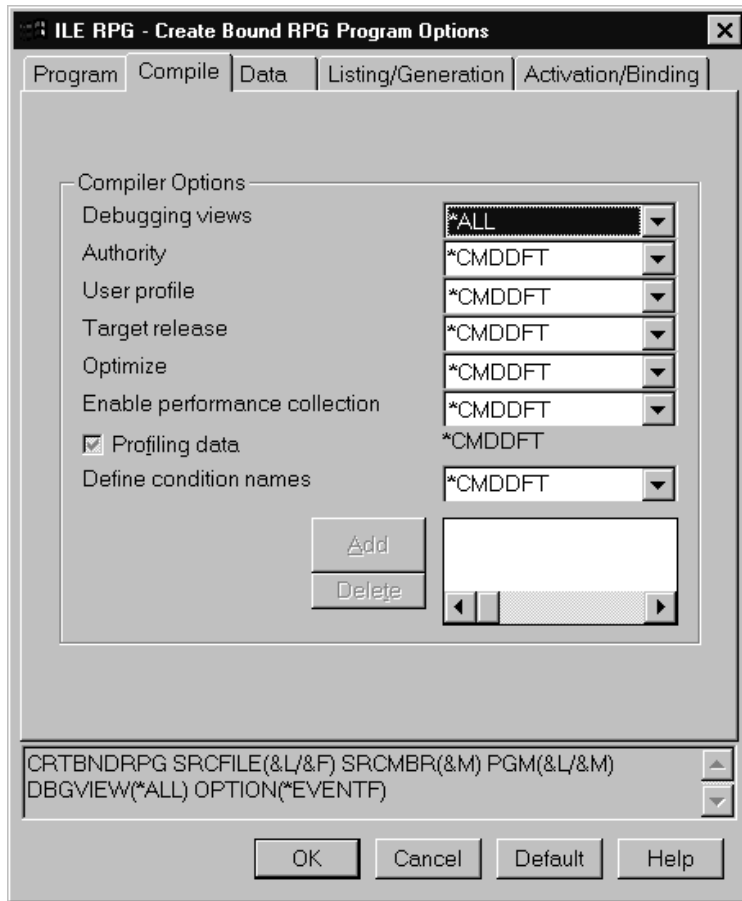


Compiling Your Source

We will now use the Remote Compile tool to select compile options. To start, we have to specify some settings and the program we want to compile.

- ___ 1. From the **Options** menu, select **Settings...** The **Settings** dialog appears.
Note: Don't use the **Options push button**, use the **Options menu item**
- ___ 1. On the **General** page **click** on the **Interactive** action mode and make sure the **Notify on completion** is checked.
- ___ 2. **Click** on the **OK** push button to save your settings and dismiss the dialog.
- ___ 3. Both the **Server** and the **Source** entry fields should already be filled in with **OS400** or if you use the **RSE** with the server name you specified there and **CODELABxx/QRPGLESRC(PAYROLL)**, respectively.
- ___ 4. The **Label** identifies the command and options for the command that will be used to compile the program. **Select** the **CRTBNDRPG** label.
- ___ 5. **Click** on the **Options...** push button near the bottom right-hand side of the notebook to select the compile options for the program.
- ___ 6. **Click** on the **Compile** tab.

- ___ 7. Change the **Debugging views** from ***CMDDFT** to ***ALL**.



- ___ 8. Select any other tab to explore the ILE RPG compile options. Click on the **OK** push button when you are done.
- ___ 9. **Click** on the **Submit** push button to submit your compile request to the iSeries.
- ___ 10. A message will tell you when the compile is complete. **Click** on the **OK** push button in the message dialog.
- ___ 11. **Click** on the **X** in the upper right-hand corner of the Program Generator dialog to close it.

The Command Shell

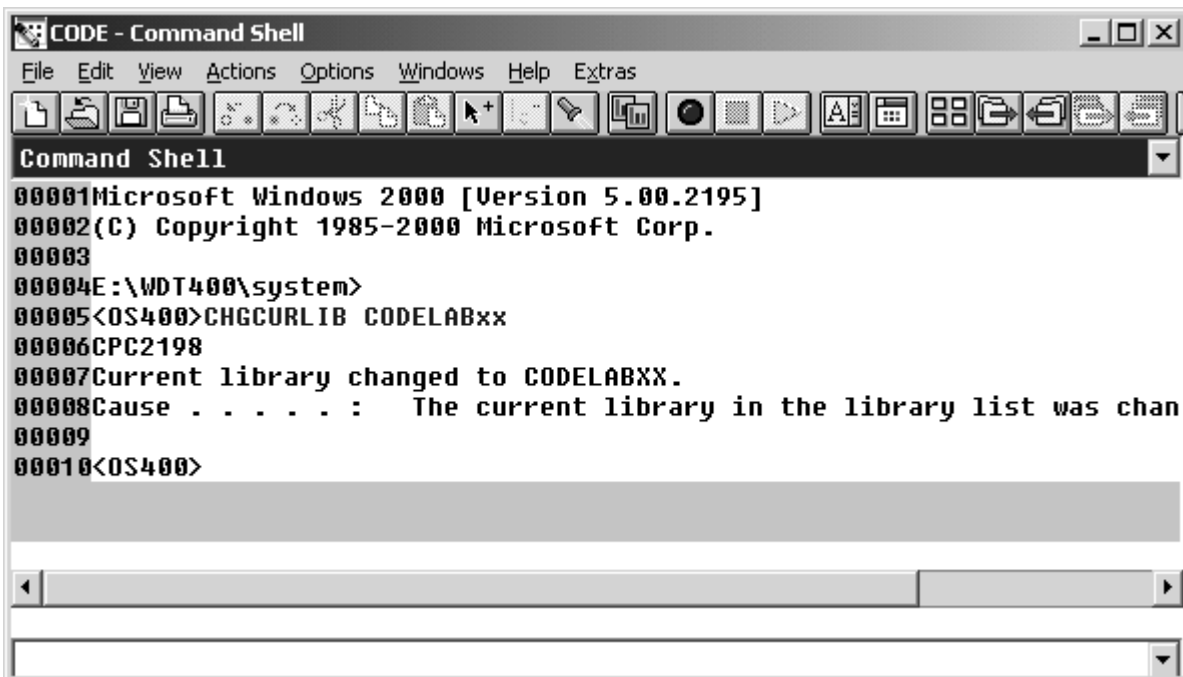
You can use the **Command Shell** inside the CODE Editor to submit commands to the iSeries.

You could change your library list, for example.

___ 1.

From the **Windows** menu, select **Command Shell** (or simply press **F9**). This will bring up the Command Shell where you can enter both workstation and host commands.

___ 2. The library you will use is **CODELABxx** where **xx** is your workstation number. At the **<OS400>** prompt, type **CHGCURLIB CODELABxx** and press Enter.



___ 3. From the **Options** menu, select **Servers** and then select **LOCAL**. This will switch the prompt in the Command Shell to a workstation prompt.

___ 4. Type **c:** and press Enter.

___ 5. Type **DIR** and press Enter. Now you can scroll up and down to view all files and directories in the c drive.

___ 6. From the **Options** menu, select **Servers** and then select **OS400** to switch back to host prompt.

Running the PAYROLL Program

In this part of the exercise we will run the PAYROLL program that we just compiled.

- ___ 1. In the editor, **switch** to the Command Shell by pressing **F9** if you are not already there.

- ___ 2. At the <OS400> prompt, type in the following command:
CALL PAYROLL
then press **Enter**.

```
PRG01                               Time Reporting System          98/04/09
                                   Maintenance Selection          17:26:38

                                   Enter an X beside the application you want to maintain

                                   _ Employee Master Maintenance
                                   _ Project Master Maintenance
                                   _ Reason Code Master Maintenance
```

- ___ 3. **Switch** to your 5250-emulation session.
- ___ 4. Place an 'x' beside **Employee Master Maintenance**. Press **Enter**.
- ___ 5. Type **123** for the Employee Number.
- ___ 6. Type **A** for the Action Code to add employee **123**. Press **Enter**.
- ___ 7. Type any information you like about the employee. Press **Enter**. Play in the application as much as you like.
- ___ 8. Press **F3** to end the PAYROLL job.

Note:

Even if you are working from the RSE, to run this 5250 application, you need to use the CODE server job that was started in an emulation session, not the RSE server job. The program will need a 5250 device to display its Workstation file content.

However, you could use the STRRSESVR command to get an equivalent to the CODE interactive job and then run the application directly from the RSE perspective.

The 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 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. We'll now step through each part of the interface and update some DDS as well.

Open a DDS display file member using the WebSphere workbench

In the workbench, in the RSE perspective use the connection that you used in the exercise before. In the list

- Expand the ***LIBL** filter or if it is still expanded use it as is,
- Expand the **QDDSSRC** file in library **CODELABxx**
- Right mouse click on the **MSTDSP** member and select:

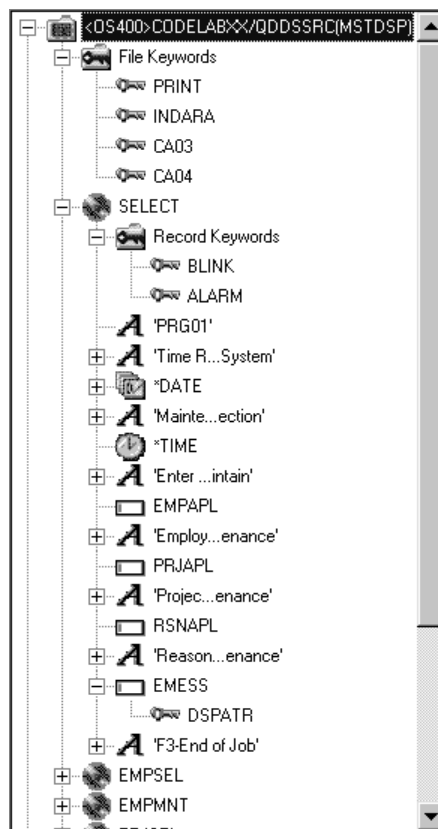
Open with --> Code designer

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

The DDS Tree

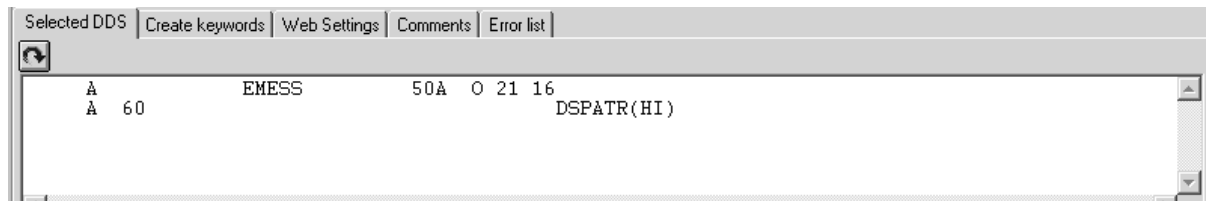
What you are looking at now is basically an explorer view of the DDS. The DDS Tree view on the left-hand side of the Designer displays the DDS source in its file, record, field, and keyword hierarchy. It is a familiar and intuitive way to see the overall structure of the DDS source and to navigate through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

- ___ 1. Click on the **+** beside the folder **<Servername>CODELABxx/QDDSSRC(MSTDSP)**.
- ___ 2. Click on the **+** beside the folder **File Keywords**.
- ___ 3. Click on the **+** beside the record **SELECT**.
- ___ 4. Click on the **+** beside the folder **Record Keywords**.
- ___ 5. Click on the **+** beside the field **EMESS**.



The DDS Tree is now showing you a nice summary of the file-level keywords and of the record SELECT

- ___ 3. In the DDS Tree **click** on the field **EMESS**. The Details page shows its field-level keywords. The Selected DDS page now shows the DDS for the EMESS field.



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

The Design Page

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.

1. Click on the tab labeled **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 make up a single screen or report. It is the set of records that is written by the application to the display or printer device at one time. Grouping records together allows you to work on one record while still seeing the related records in the background. The Workbook has a Design page tab for each group defined for quick access to each group of records.

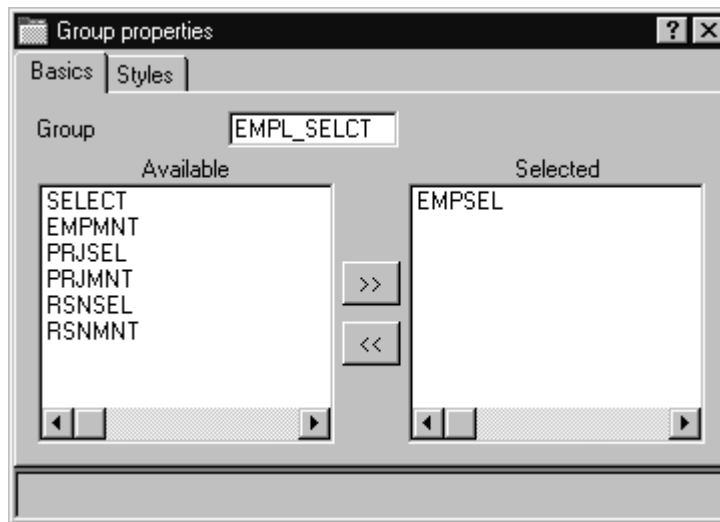
Creating Groups from Existing Records

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 we will create a group for the next screen, where the user selects which employee in the payroll database to maintain.

- ___ 1. **Scroll** to the bottom of the DDS Tree and **click** on the + 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.
- ___ 4. On the Group properties notebook, **click** on the record **EMPSEL** in the Available list box and **click** on the push button. For simplicity this is the only record we will add for now. The Design page now shows us what the record EMPSEL looks like.
- ___ 5. Name the group by **over typing** SCREEN1 with **EMPL_SELECT**



- ___ 6. **Close** the Group properties notebook by clicking on the X in the top right corner.

Well, 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

- ___ 1. **Right-click** on the new EMPL_SELECT 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** on the **Create subfile record** button and then **click** in the dark gray area. A subfile and a subfile control record pair is created.

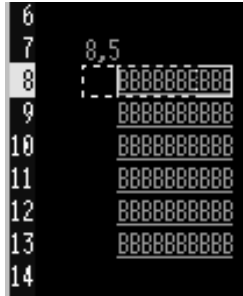
Field Creation and Design Page Toolbar

- ___ 1. Now **click** on 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** on 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 field. Rename the record from RECORD1 to EMPLSTSFL and the field from FIELD1 to OPCODE by simply over typing the text in

the combo box. Change the color of the field by clicking on the  **Color palette** button and selecting pink. Change the usage of the field to input only by clicking on the .

- ___ 5. **Position** the cursor at **row 8, column 9**. Note that the bottom right of the Designer frame shows the current cursor position  If you can't see the field with the

cursor position on your screen, press the Maximize button in the top right corner.


You can use the cursor keys or the mouse to move the cursor.

- ___ 6. If you are creating a long field with an exact length, the SDA syntax can be easier.

Type:

+O(30)

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

- ___ 7. **Hit the Convert string to field**  button on the toolbar or **press F11** to convert the SDA syntax into a character output field of length 30.

- ___ 8. **Rename** the new field to **ENAME** using the toolbar. This will show the name of the employee.

- ___ 9. **Position** the cursor to **8, 41**.

- ___ 10. Now we will add a field for the employee's salary. Now wouldn't it be nice 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 hit the **back arrow**.

- ___ 11. **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!

- ___ 12. **Rename** the field to **SALARY** and change its color to yellow, using the toolbar.

- ___ 13. Our subfile seems a little scrunched to the left. It would be nice to space it out evenly.

Just select the field and hit  on the far right side of the toolbar. The other buttons in the vicinity will do your typical align, left, right, center and top.


- ___ 14. 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

- ___ 1. Now let's fix up the Subfile control record. The group we created contains 2 records.

You can verify this by dropping down the record combo box in the toolbar.



Change the current record by selecting RECORD1CTL from the combo box or by hitting the  or simply by pressing **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.

Copy and Paste

Let's provide a 'position to' entry in the subfile control header.

___ 1. **Position** the cursor at **4,9** and type:

Position to:

___ 2. Now we need an employee name field. We could create a named field with the right characteristics like we did in the subfile, or we could create a source reference using the  button in the palette or we 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 +) the **EMPMNT** record.


___ 3. **Click** on the **ENAME** field and **press Ctrl+C**. (The pop-up menu or Edit pulldown would give the Copy menu item as well).

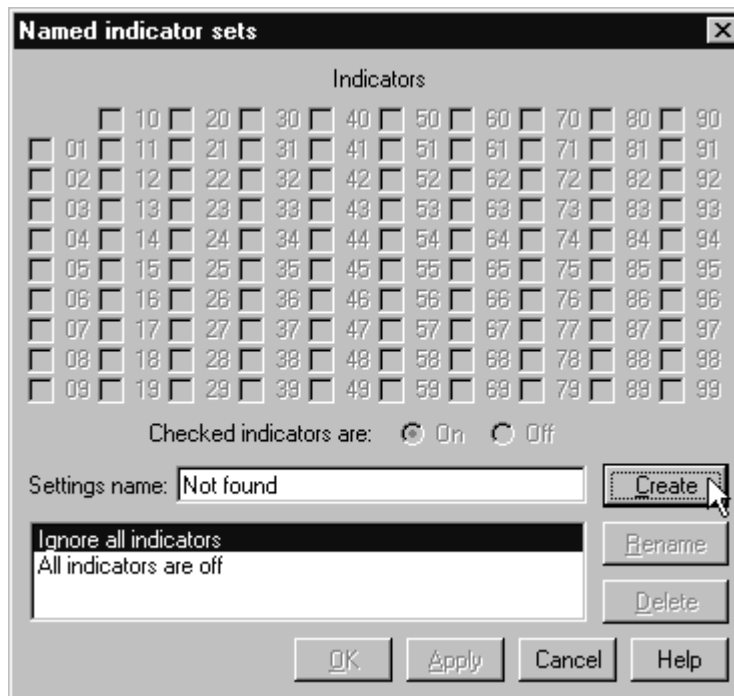
___ 4. **Position** the cursor to **4, 23** and **press Ctrl+V**. Voila! Now that was easy!

___ 5. **Rename** the field to **POS_TO**.

Working with Indicators

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

- ___ 1. Click on the  button on the design page toolbar (or **press F7**). The **Named indicator sets** dialog appears.
- ___ 2. In the **Settings name** field, type:
Not Found
 and hit the **Create** button.



- ___ 1. Click on the checkbox next to **60** and press **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 combo box on the bottom left.

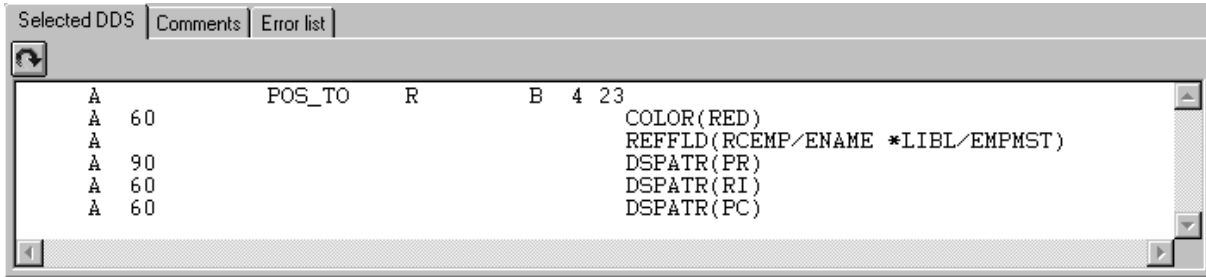


- ___ 2. Now **select** the **POS_TO** field.

- ___ 3. On the toolbar, select the color **red** and the display attributes **reverse image** and **position cursor**. (The set of latchable toolbar buttons representing the current display attributes is found just below the color button). The toolbar should look as follows:

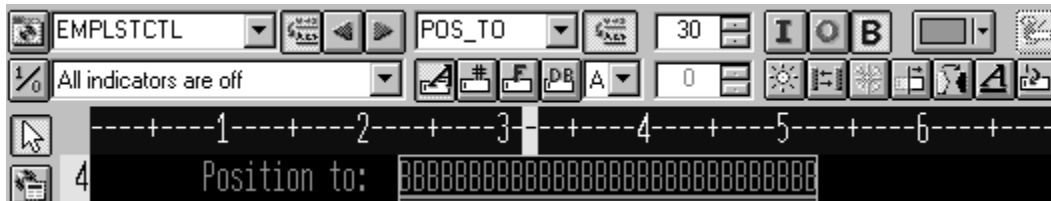


- ___ 4. **Examine** the DDS generated in the **Selected DDS** page.



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

- ___ 5. Now let's try it out! From the **Select named indicator sets** combo box, select **All indicators are off**.



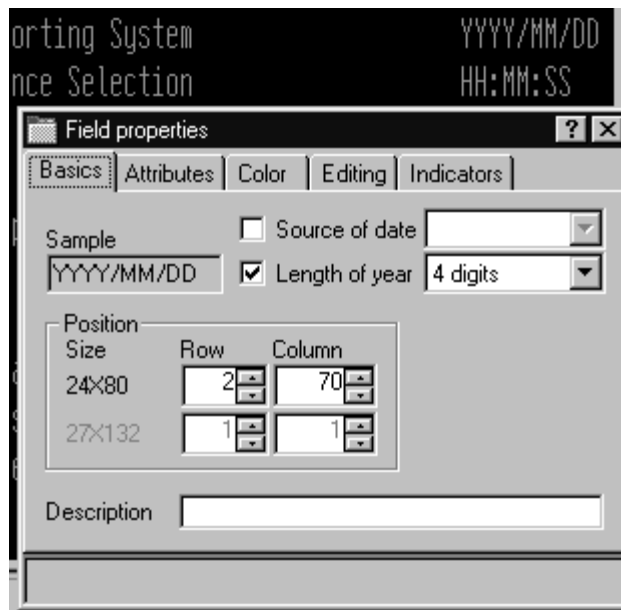
Hey! This stuff really works!

- ___ 6. From the **Select named indicator sets** combo box, select **Not found**. The field appears red and reverse imaged.

The Properties Notebook

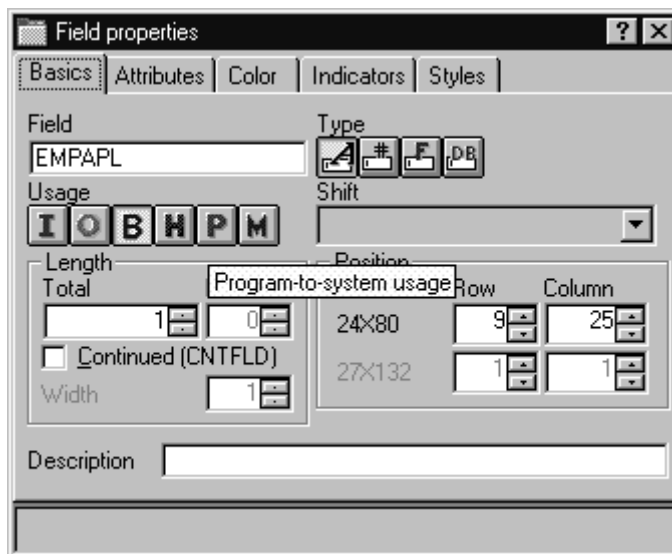
Second to the direct manipulation and the toolbar on the Design page, the easiest and quickest ways of getting access to the properties of a field, record, or entire file is a Properties notebook. You can get to a Properties notebook from the **Selected** menu, by pressing **F4**, or **double-clicking** on anything in the DDS Tree or the Details page or Design page.

- ___ 1. In the DDS Tree, **click** on 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** on 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** on the **Length of year** checkbox.
- ___ 4. Select **4 digits** from the combo 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,

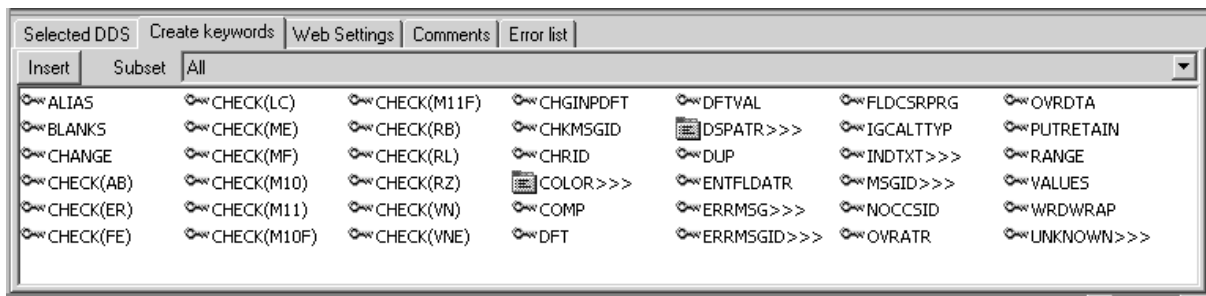
length, type, screen position. The other pages give you quick access to other properties for this field.





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.”

- ___ 1. Click back on the **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.




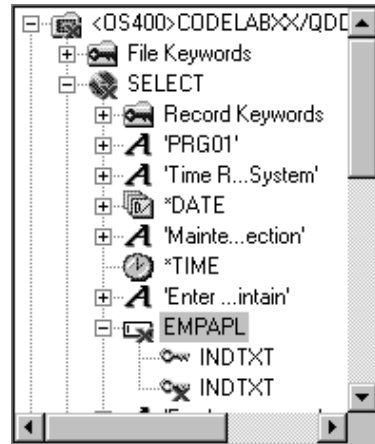
- ___ 3. With the **EMPAPL** properties notebook at the Basics page, click on the  button to change the field to numeric type. Notice that the list of keywords in the **Create keywords** page has changed.
- ___ 4. Click on the  button to change the field back to alphanumeric. Notice that the list of keywords in the **Create keywords** page has changed.
- ___ 5. Click on 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. Feel free to 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** on 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 at your leisure.
- ___ **8. Double click** on the INDTXT keyword again. The keyword is created with the same default values creating a conflict.

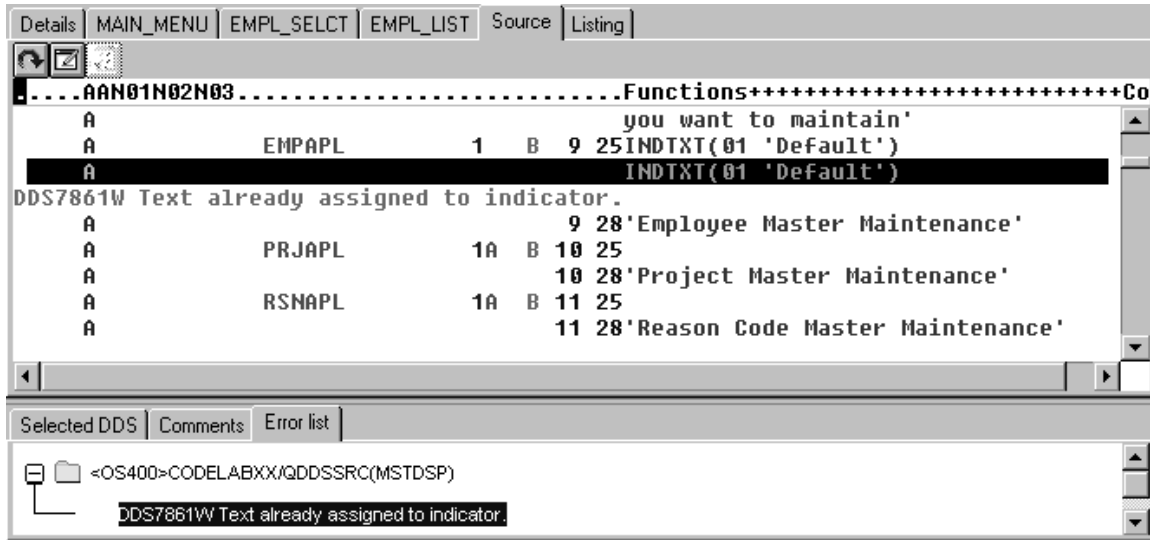
Verifying Your Source

We 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 we need to make sure that there are no semantic errors. We just introduced one in the last section by creating two INDTXT keywords describing the same indicator.


- ___ 1. From the **Tools** menu, select **Verify file** (or **click** on the  button on the main toolbar). The DDS source is checked using the same verifier that the CODE Editor uses. A message appears on the status line at the bottom of the designer stating that the verify completed with errors.
- ___ 2. In the DDS Tree, there is a trail of red x's leading to the culprit. The file icon has a red x, as does the SELECT record, the EMPAPL field and finally the second INDTXT keyword.
- ___ 3. **Click** on the **MAIN_MENU** tab in the workbook. The EMPAPL field is highlighted in red.
- ___ 4. **Click** on 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. **Click** on the **Error list** tab.
- ___ 6. **Double-click** on the warning DDS7861 in the Error list. (Hitting F1 would get 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 nice if you could just zap 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

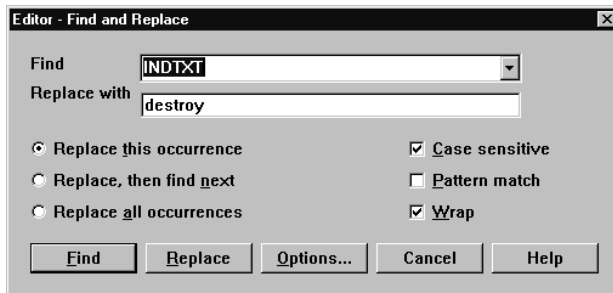


modes at the push of a button.



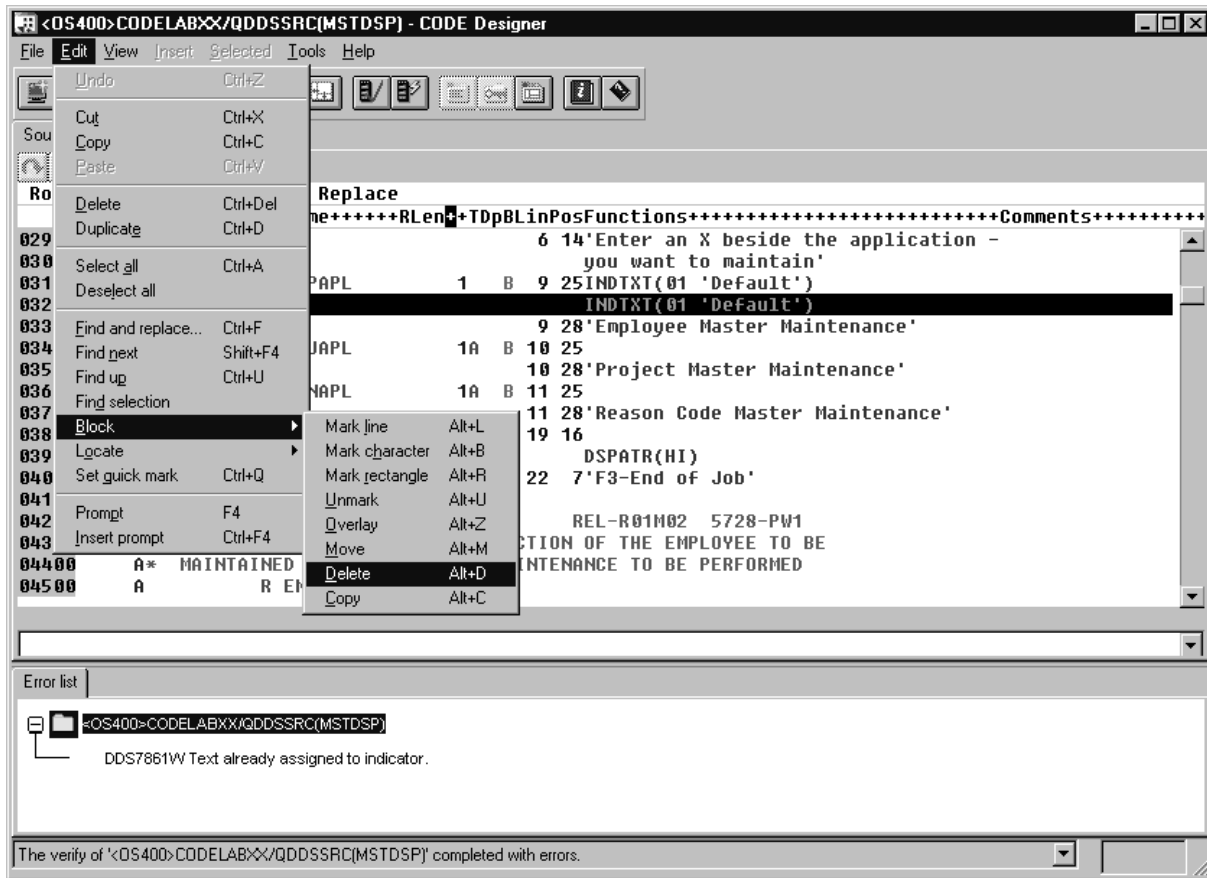
Switching between Design mode and Edit mode

- ___ 1. Click on the  button or select the **File**→ **Edit DDS Source** menu item. You now have access to the full power of the editor.
- ___ 2. Explore the **Edit** and **View** pulldowns.
- ___ 3. Press **Ctrl-F** to bring up the Find/Replace dialog
- ___ 4. Type in **INDTXT** and hit the **Find** button.




- ___ 5. Press **Ctrl-N** to find the next occurrence.

6. Delete the second INDTXT line.



Compiling Your Source

Now we will compile the source on the iSeries just as we did in the CODE 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  button on the main toolbar).
3. A message will tell you when the compile is complete. **Click** on the **OK** push button in the message dialog.

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

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

Now let's play with the debugger.

The Distributed Debugger

The CODE debugger is a source-level debugger that allows you to debug and test an application running on a host iSeries 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 host.
- 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 debuggable 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 we will be debugging an ILE RPG/400 program. Don't worry if you don't know RPG.

Starting the Distributed Debugger

You will be working with the ILE RPG program PAYROLLG.

Note: PAYROLLG is the same RPG program as PAYROLL but without compile errors. We are using it instead of PAYROLL in the debugger section to accommodate everyone who decided to skip right to this section without completing the editor exercises.


You can start the Debugger in several ways: from a DOS prompt, from the Windows Start menu; from the Remote Systems Explorer; or from the CODE Editor. We will use the CODE editor:

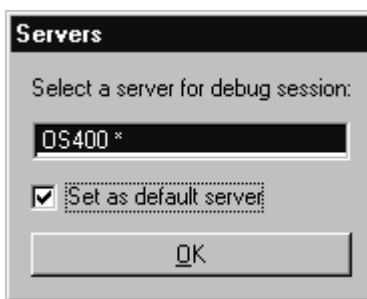
Use the Remote Systems Explorer to load the **PAYROLLG** in source file **QRPGLESRC** in library **CODELABxx** into the CODE editor. Just expand the connection until you see the member, right mouse click on it, and use the **Open with** option and CODE editor option:

<connection name>Library

list--CODELABxx--QRPGLESRC--PAYROLLG--Open with--CODE editor

where **xx** is your team number.

1. From the editor's **Actions** menu, select **Debug** and then select **Interactive application** (or click on  in the toolbar or even faster **press Ctrl+Shift+D**). Since this is the first time you invoke the debugger and you did not specify a default server to be used for debugging, the Servers dialog comes up.

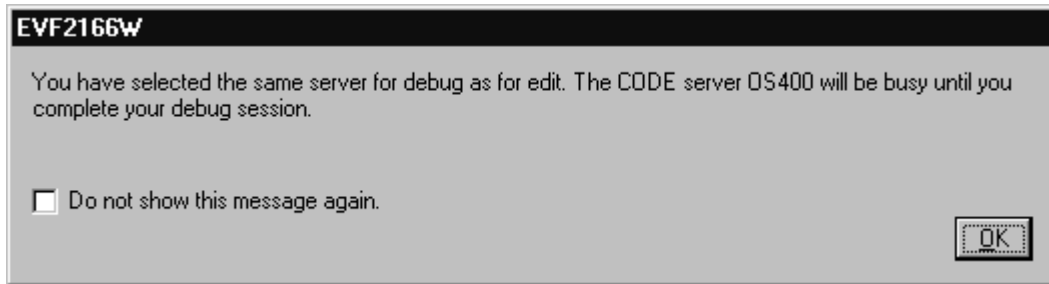


2. Select **Set as default server** and

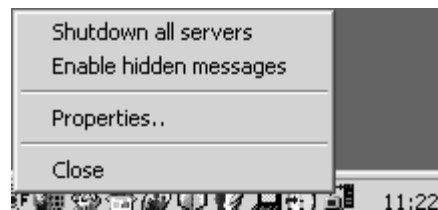
3. Press **the OK** push button.

This server will now be used for your subsequent debug sessions.

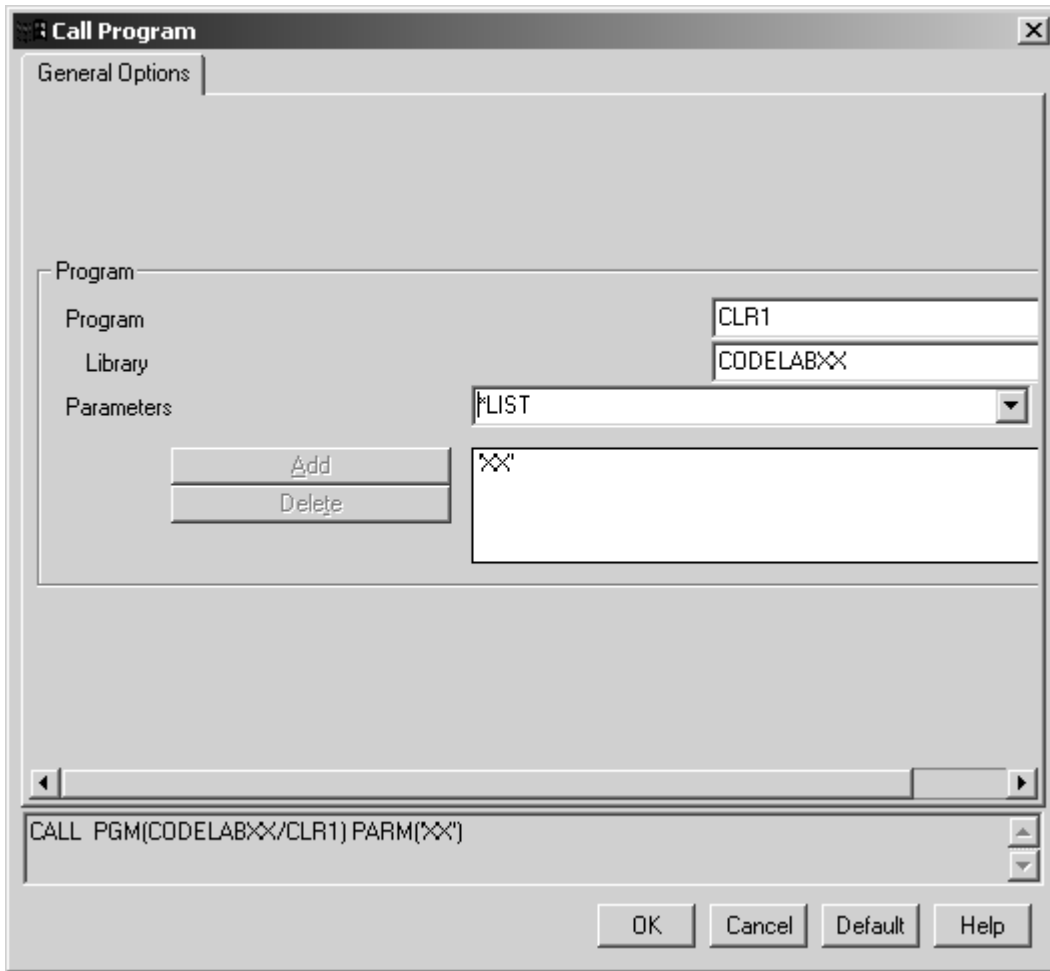
Since we only started one server and are using it for editing and debugging, the following message is displayed:



1. Select **Do not show this message again** and click **OK**. If you want to re-enable the message, select **Enable hidden messages** from the **CODE Daemon** context menu.



The **Call Program** dialog appears next.

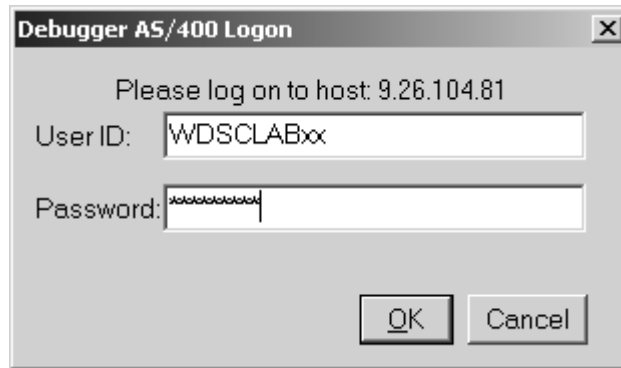


Program and Library are pre filled with the respective entries from the Editor. If your program requires parameters you have to enter them here. We want to start the debugger for a CL program **CLR1** which calls the **PAYROLLG** program.

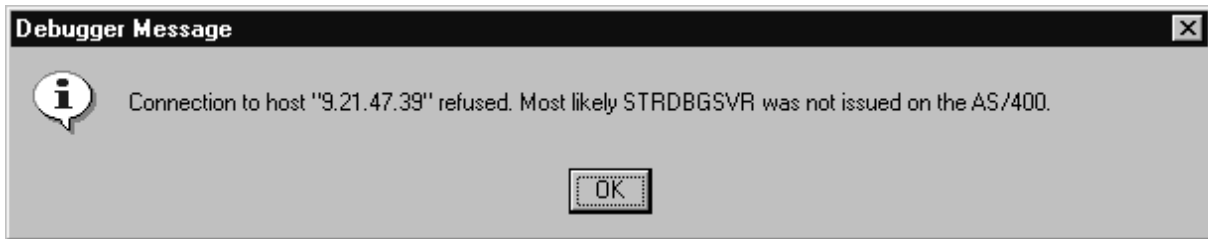
- ___ 2. Change the program entry in the Call Program dialog to **CLR1**. This program requires your team number as parameter.
- ___ 3. Enter your **team number in single quotes** in the Parameters field and click on **Add**. The parameter is now displayed in the parameter list.
- ___ 4. Click **OK**.

The debugger window comes up. This window is open as long as the debugger is active. If this is the first time the debugger is invoked, you will be presented with a Logon dialog.

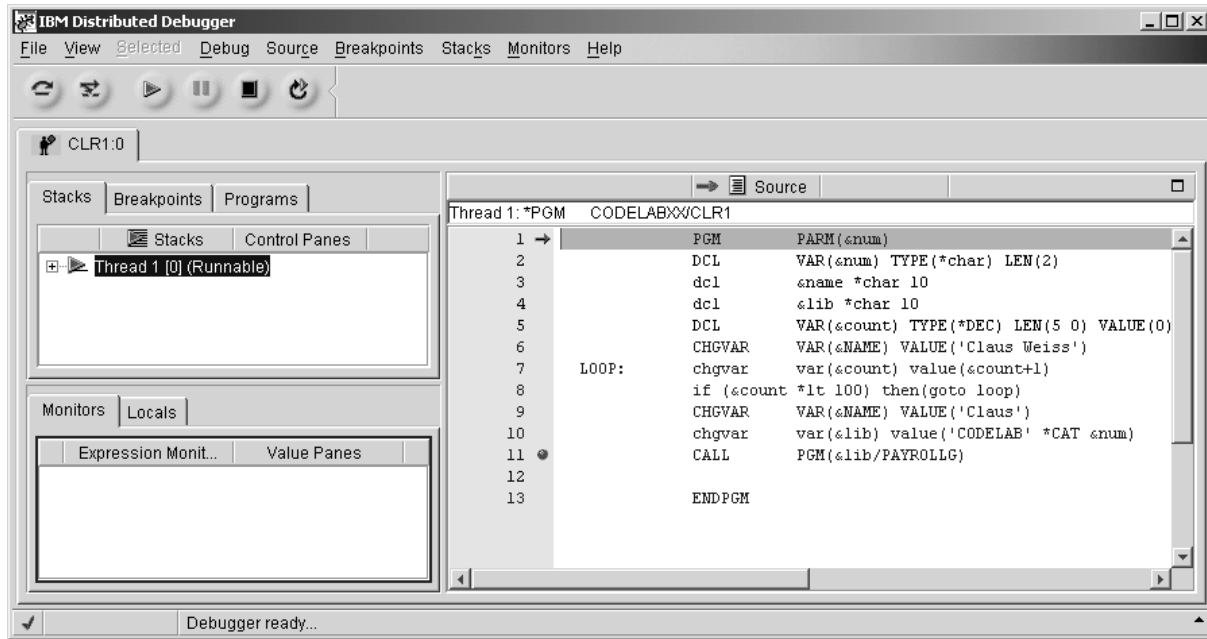
Enter your userid and password (**WDSCLABxx**) and click **OK**.



Note: There is a server that runs on the iSeries which services requests from the CODE Debugger. This server is started with the command **STRDBGSVR** and will run until the next IPL or until it is explicitly terminated (**ENDDBGSVR**). If no one has run this command since the last IPL you may get the following message:



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

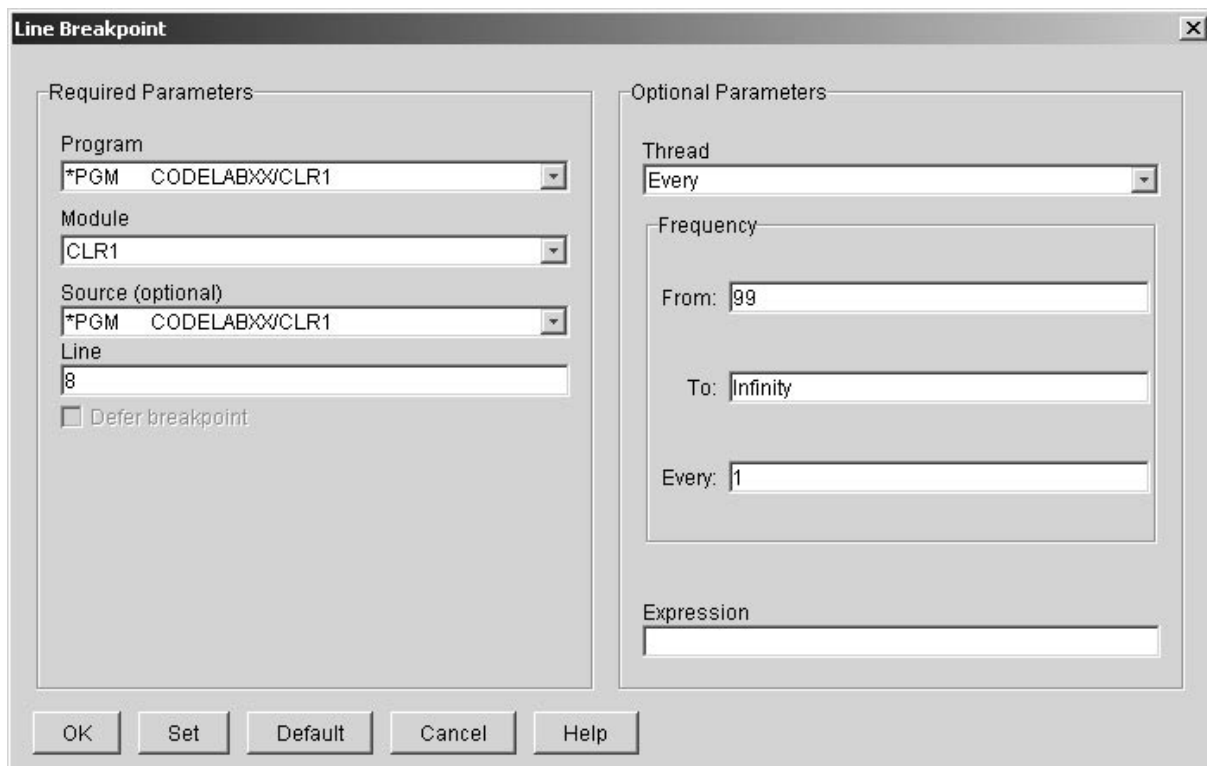


Setting Breakpoints

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

The easiest way to set a breakpoint is to double-click in the prefix area of the Source pane:

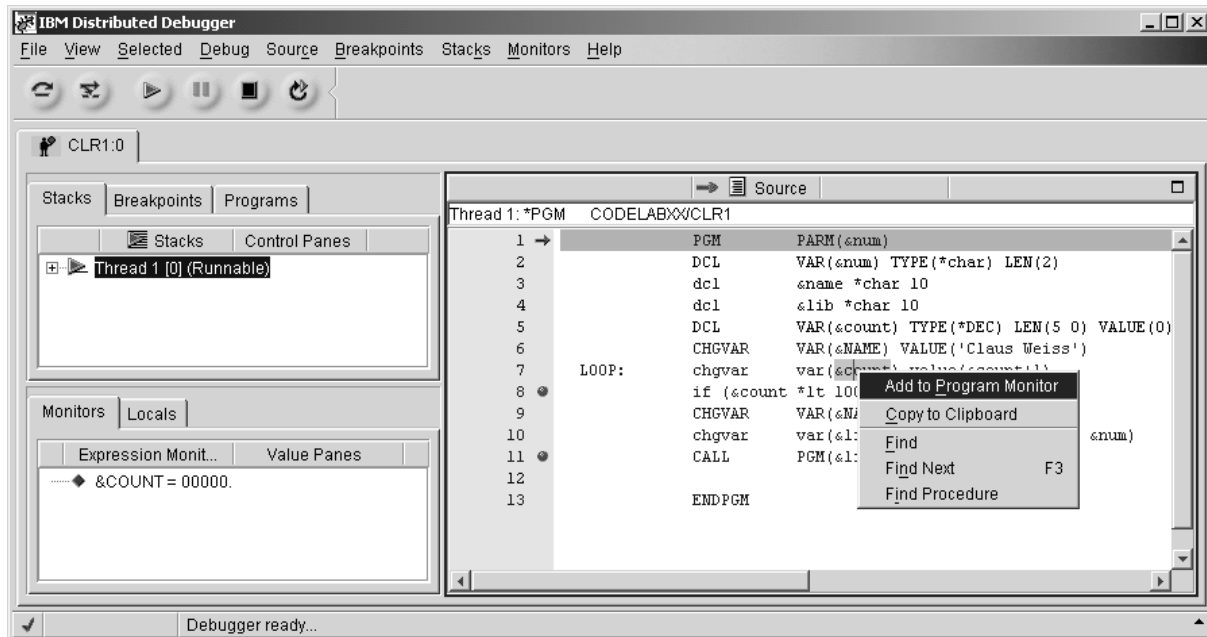
- ___ 1. **Move** the mouse over the prefix area (left margin) of line **11** and **double-click**. A red dot is added to the prefix area of this line to indicate that a breakpoint is set.
- ___ 2. Single click on the **8** in the prefix area to select the line and from the Breakpoints menu, select **Set line...** The breakpoints dialog is displayed.
- ___ 3. We only want to stop in the loop when it executes for the 99th time or more. We can do that by setting the From field of the Frequency group to 99. Enter **99** and click on OK.



Monitoring Variables

You can monitor variables in the Monitors pane. In this exercise, you will monitor the variable `&COUNT`.

1. In the Source pane, **double-click** on the variable `&COUNT`, right click and select **Add to Program Monitor**.




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, you can select **Allow Tool Tip Evaluation** on the **Source** menu. Leaving the mouse pointer on a variable for a second or so will now display its value in a little popup.

Running a Program


Now that some breakpoints are set, you can start running the application:

1. Click the **Run** button  from the toolbar. The program starts running and hits 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 that `&COUNT` now has the value 99.

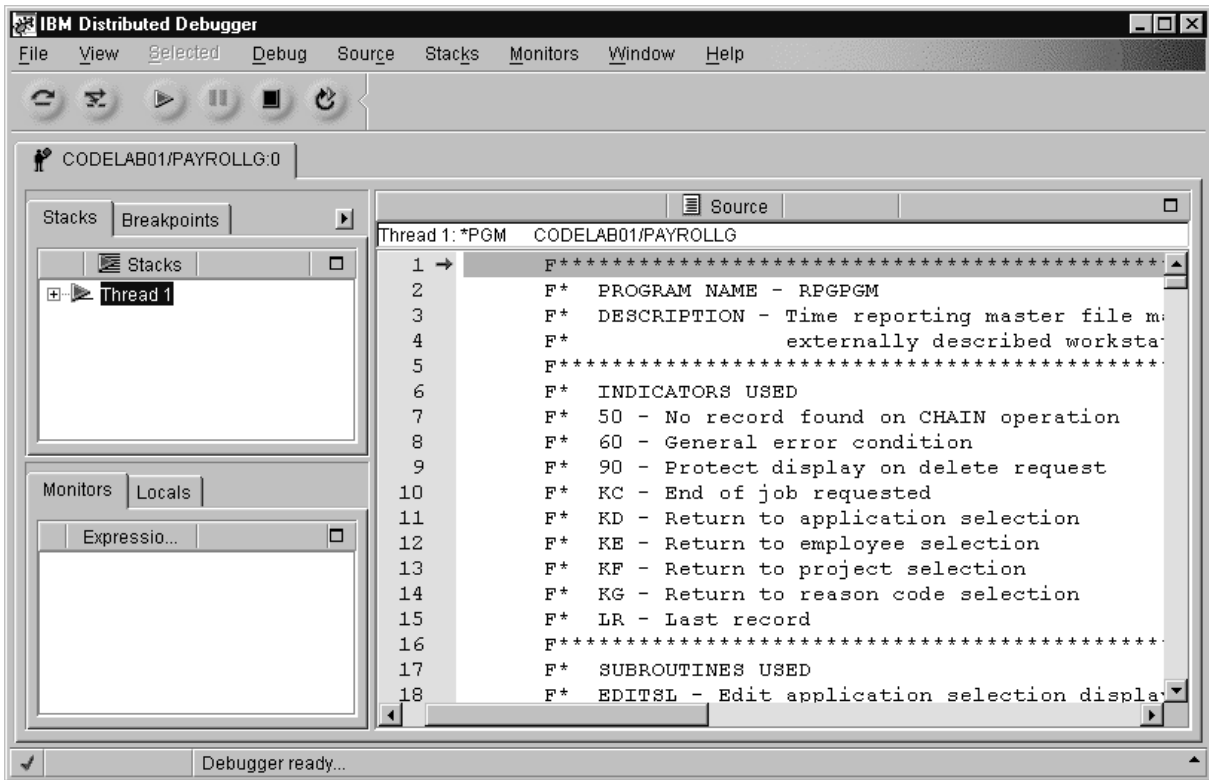
- ___ **2. Click** on the **Run** button again. The program stops at the breakpoint at line 8 again and &COUNT now has the value 100.
- ___ **3. Run** once more so that the program hits the breakpoint at line 11.

Stepping into a Program

The CODE 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. We want to step into the Payroll program.

- Press the Step into button  on the toolbar. The source of PAYROLLG is displayed.

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

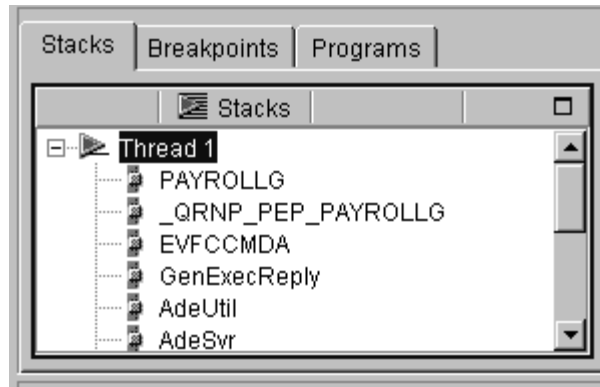


- From the **Source** menu, select **Listing View**. Page down in the source and take a look at the expanded file descriptions. We don't have any /Copy member in our PAYROLL program but these would also be shown in a listing view
- Switch back to the source view. From the **Source** menu, select **Source view**

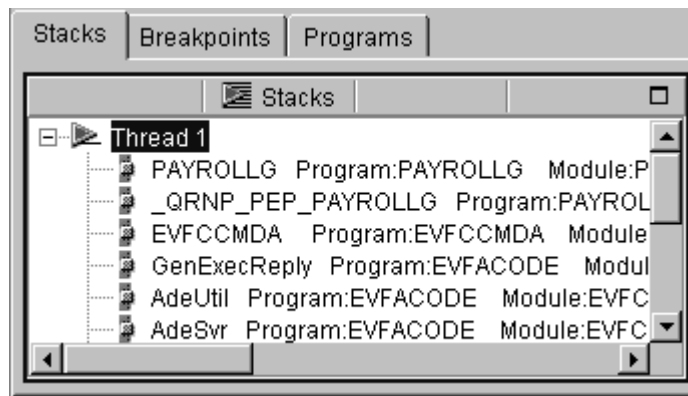
The Call Stack

The **Stacks** 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. Clicking on a stack entry will display the corresponding source if it is available. Otherwise the message **Context not available** appears in the source pane.

1. In the **Stacks** pane, expand the stack entry of Thread1 by clicking on the + sign in front of it or by double clicking on Thread1.



2. To view detailed information about each stack entry, select **Stacks** from the menu bar and click on **Show all Stack Information**.



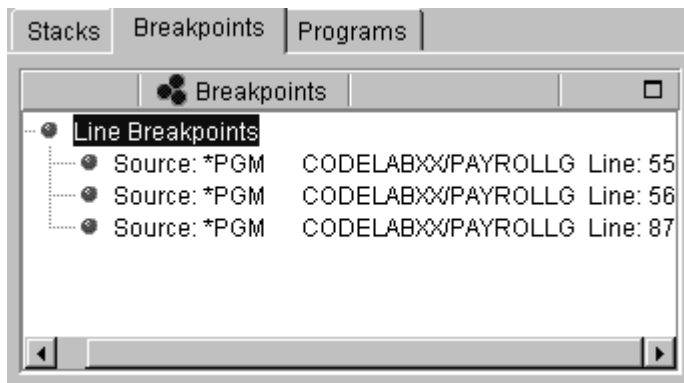
It allows you to work with and switch between different programs and/or ILE modules.

The Programs pane

The Programs pane lists all programs that are currently known to the debugger. Programs can be expanded to show modules and procedures. Clicking on a module or procedure entry displays its source. Adding or removing programs is available from the Programs menu.

Setting Breakpoints in PAYROLLG

- ___ 1. **Move** the mouse over the prefix area (left margin) of line **55** and **double-click**. A red dot is added to the prefix area of this line to indicate that a breakpoint is set.
- ___ 2. **Repeat** the above step for line 56.
- ___ 3. **Right click** in the prefix area of line 87 and select **Set Breakpoint** from the pop-up menu.
- ___ 4. Select the **Breakpoints** tab from the top left pane. Expand the **Line Breakpoints** entry by clicking on the + sign in front of it or by double clicking on **Line Breakpoints**. This gives you a tree view of all line breakpoints currently set in your program.



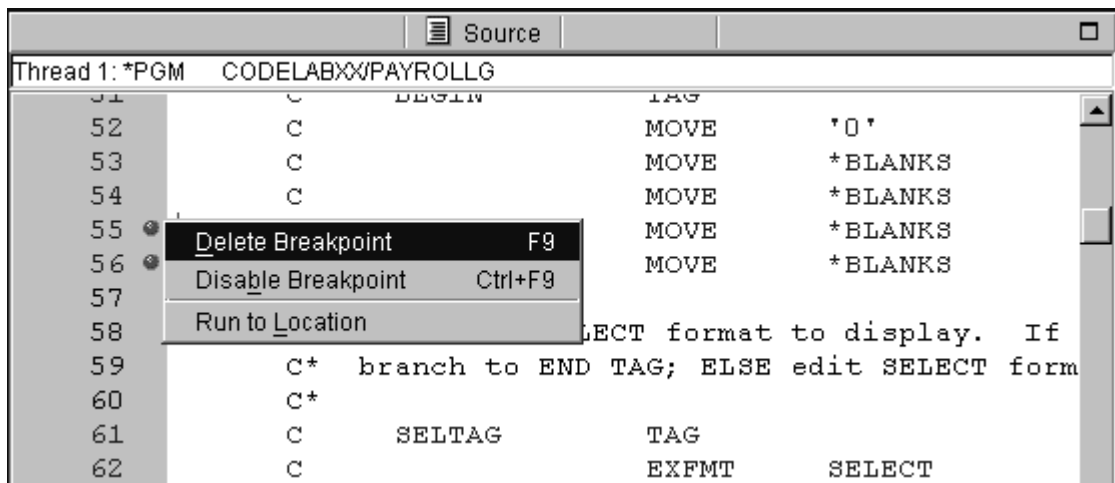
This is a convenient place to work with breakpoints. You can delete, disable/enable, or modify a breakpoint. These tasks are available from the Selected menu or from the breakpoints pop-up menu. Clicking on any entry selects it and also brings up the source where the breakpoint is set. We will use a different method to work with breakpoints.

Deleting a Breakpoint


It is also easy to manipulate breakpoints from the Source pane.

1. **Move** the mouse over the prefix area of line **56** and double-click on it. The red dot is removed from the prefix area indicating that no breakpoint is set on that line.

Note: You could also right-click in the prefix area and select **Delete breakpoint** from the pop-up menu.



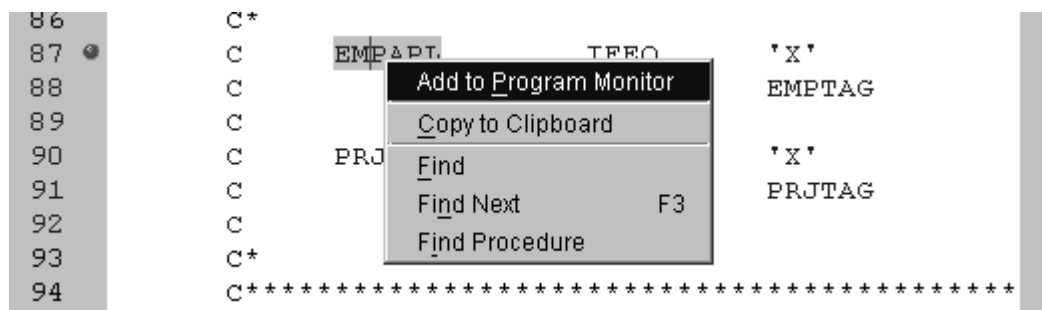
Running the Program

- ___ 1. Click the **Run** button  from the toolbar. The program starts running and hits the breakpoint at line **55**.
- ___ 2. Click on the **Run** button again. The program waits for input from the 5250-emulation session.
- ___ 3. Type an **X** beside the **Project Master Maintenance** option, then press **Enter**. The program hits the breakpoint at line **87**.

Monitoring Variables in PAYROLLG

You can monitor (or change) variables and indicators in the Monitors pane. In this exercise, you will monitor variables and change their values.

- ___ 1. In the Source pane, **double-click** on the variable **EMPAPL** on line **87**, right click and select **Add to Program monitor**.



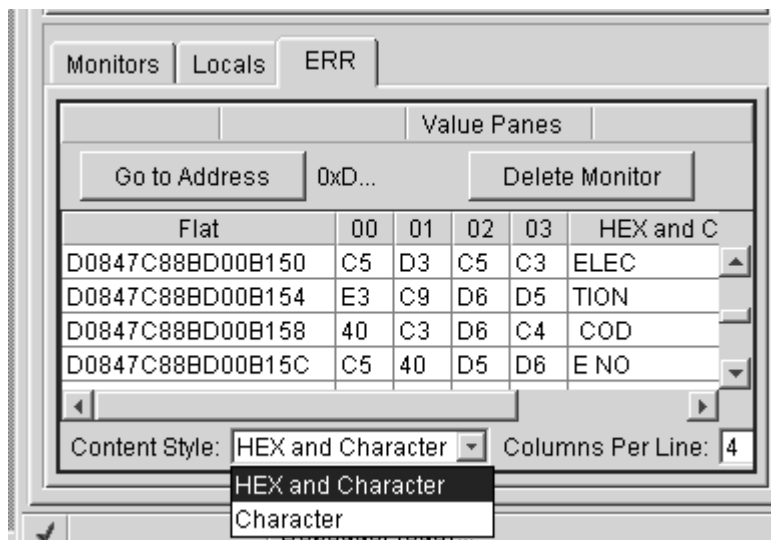
- ___ 2. The variable appears in the Monitors pane. Its value is blank because you did not select the Employee Master Maintenance option.
- ___ 3. In the same way add the variables **PRJAPL** on line **90** and **RSCDE** on line 102 to the monitor. Variable **PRJAPL** equals **'X'** because you did select the Project Master Maintenance option.
- ___ 4. In the Monitors page, **double-click** on the variable **RSCDE**. The value changes into an entry field.
- ___ 5. Type in the new value **X** for the variable.
- ___ 6. Press **Enter**. The variable is successfully changed.

Adding a Storage Monitor

Note: You have to be on a **V5R1 iSeries** to be able to use this function.

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.

- ___ 1. In the Source window, **double click** on **ERR** in line 33, right click and select **Add to Storage Monitor**. A new page is added to the Monitors pane. The tab shows the name of the variable.

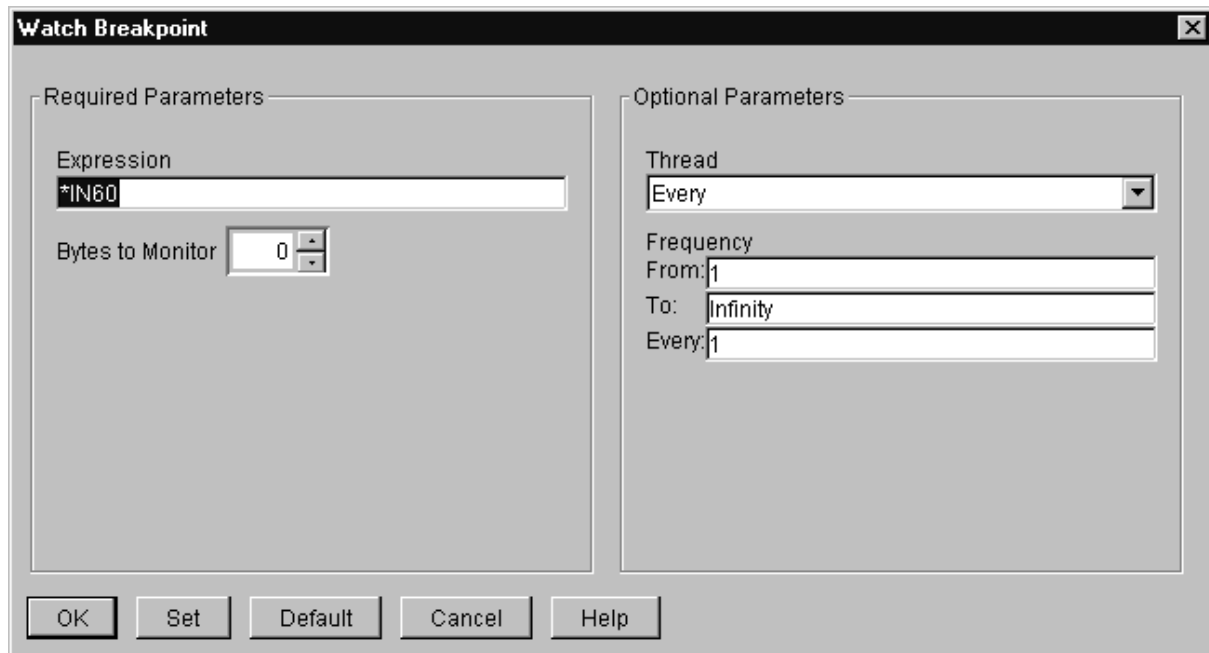


- ___ 2. Change columns per line to **8** by clicking on the number 4 and selecting 8 from the popup.
- ___ 3. Use the arrows on the right of the storage monitor to scroll down.
- ___ 4. Click the **Go To Address** button to get back to the starting address.
- ___ 5. Press **Delete Monitor** to remove the storage monitor.

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.



- ___ 1. In the Source window, go to line **122**.
- ___ 2. **Double click** on the variable ***IN60** to highlight it.
- ___ 3. From the **Breakpoints** menu, select **Set watch...** The Watch Breakpoint window appears. The **Expression** entry field is pre filled with the highlighted variable ***IN60**.



- ___ 4. **Click** on the **OK** push button. The watch breakpoint is now set.
- ___ 5. **Click** on the **Run** button from the toolbar. The application waits for input from the 5250-emulation session.
- ___ 6. In the 5250-emulation session, **type 123** for **Project Code** and **D** (for delete) in the **Action Code** field.
- ___ 7. **Press Enter**. A message is displayed indicating that the variable ***IN60** has changed.
- ___ 8. **Click OK**. The program stops at line **454**. This line is located immediately after the statement which caused the variable ***IN60** to change.

Stepping Through the Program

The CODE Debugger allows you to step through your source code one line at a time.

- ___ 1. Press the step over  and step debug  buttons on the toolbar.
- ___ 2. From the **Debug** menu, select **Step Over** and **Step Debug**.

Closing the Debug Session

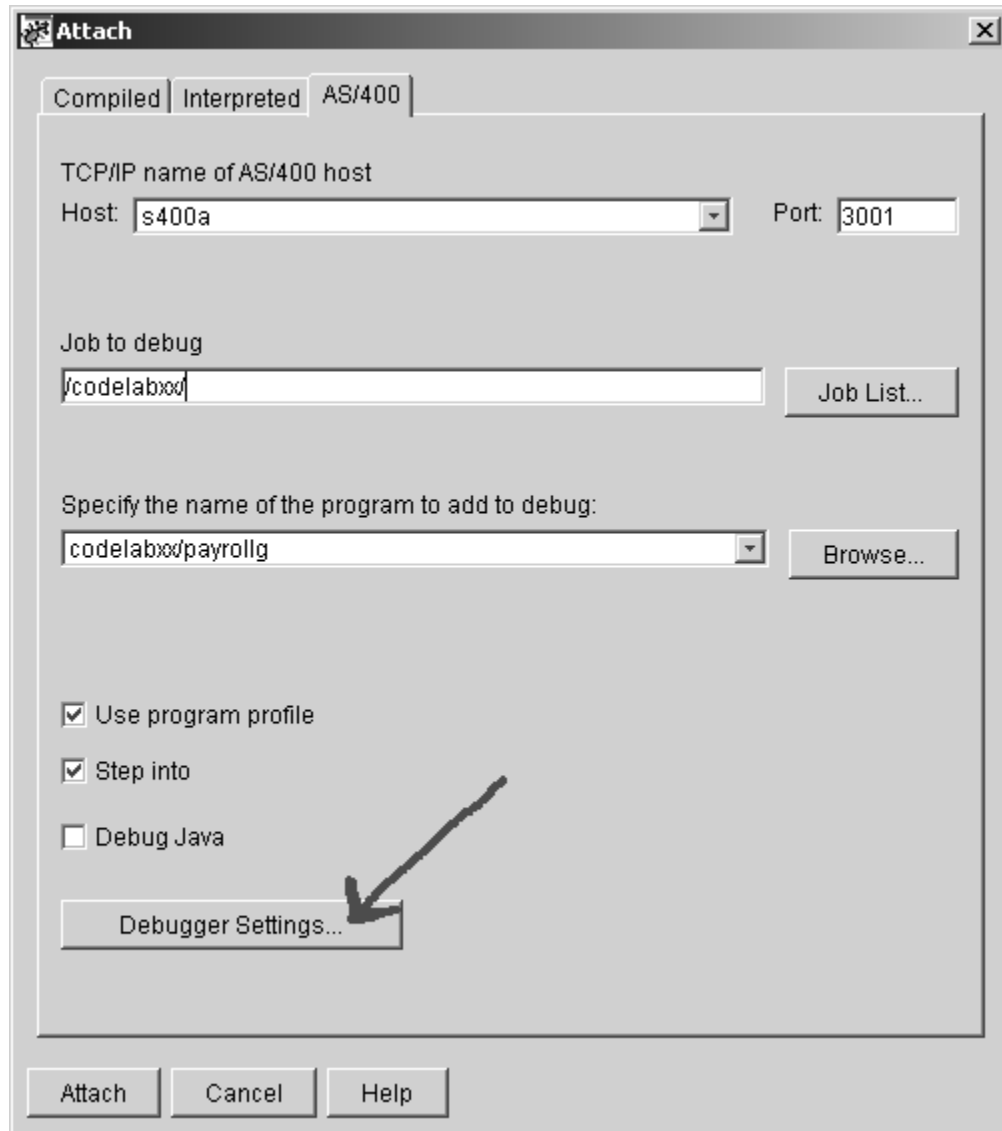
This quick tour has demonstrated the main features of the CODE Debugger.

- ___ 1. **Click** on the **Run** button from the toolbar. The application waits for input from the 5250-emulation session.
- ___ 2. Switch to the 5250-emulation session. Press **F3** to end the program.
- ___ 3. A message **Debug session terminated** comes up. Click on **OK**.

Debugger Settings

In our lab setup, we are using test libraries. If you want the debugger to be able to access production libraries, you have to change the default setting for Update Production Files.

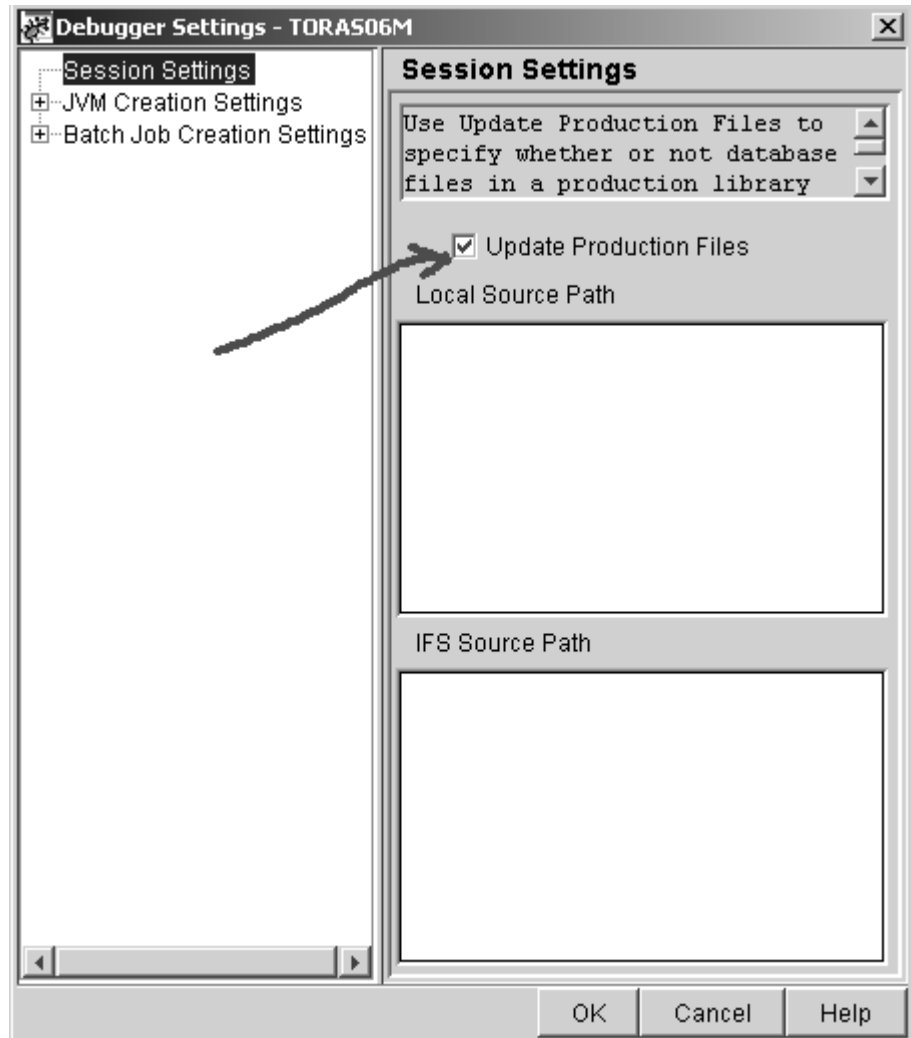
1. From the **File** menu, select **Attach...**



Note: The Debugger Settings Dialog is also available from File --> Load program... .

2. Click on the **Debugger Settings...** push button.

- ___ 3. Click on **Update Production Files** so that it gets a check mark.



- ___ 4. Click **OK** to close the Debugger Settings dialog. Update Production Files is now enabled for all subsequent debug sessions for the selected host.

- ___ 5. Click **Cancel** in the Attach dialog to dismiss it.

Note: If you want to change Update Production Files or any other debugger settings before running a debug session, you can start the debugger from the CODE editor by selecting the Actions menu, then Debug --> Running application. This will start the debugger with the Attach dialog displayed. Just follow the same steps as above.

Closing the Debugger

If you know that you will use the debugger again, don't close it. The next session will start up faster. Our debugger exercises are done though, so we will close it now.

___ 1. From the File menu, select **Exit** to close the debugger.

Now let's move on and see how the Remote Systems Explorer can help you organize your work.

WebSphere Workbench

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

The **Remote Systems Explorer** is the replacement for PDM (Program Development Manager) on the workstation. It currently doesn't have all the functions of PDM but will over time be a full replacement for PDM.

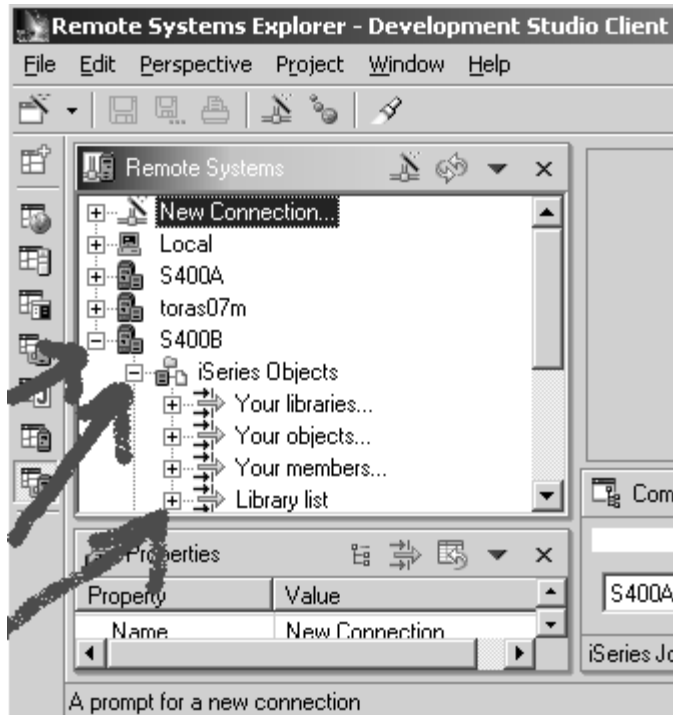
The RSE allows you to:

- 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 CODE 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.

In the following exercise you will use the **Remote Systems Explorer (RSE)** perspective to work with the iSeries objects that you used in the previous exercises. You will also see how easy it is to perform actions and define your own actions. In short, you'll see how the **Remote Systems Explorer** can organize and integrate your work and make that work easier.

Selecting an iSeries object in the RSE perspective

In the RSE perspective, you now need to get to the iSeries objects you want to work with



In the previous exercises you have worked with the library list filter, now you will create your own library filter.

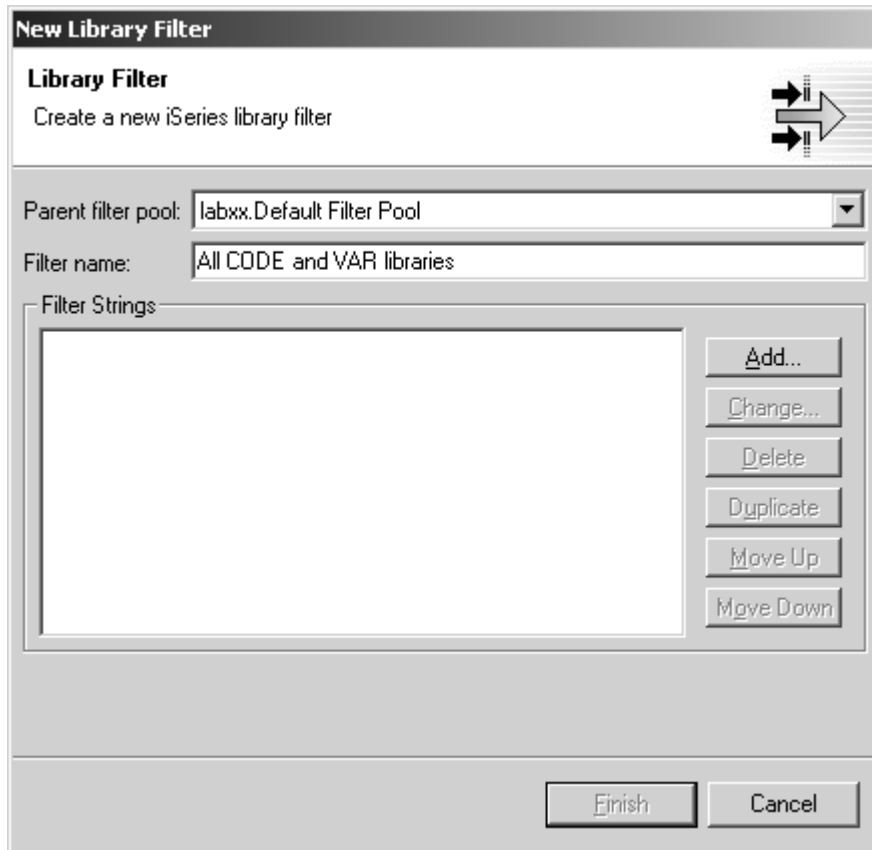
First you will need to specify the library you want to work with:

- Expand the **connection node** that connects to your iSeries host, by clicking on the **+** sign beside it.
- Expand the **iSeries Objects** node.

To create a new library filter

- Expand the **Your libraries...** Node.

You see this dialog show up



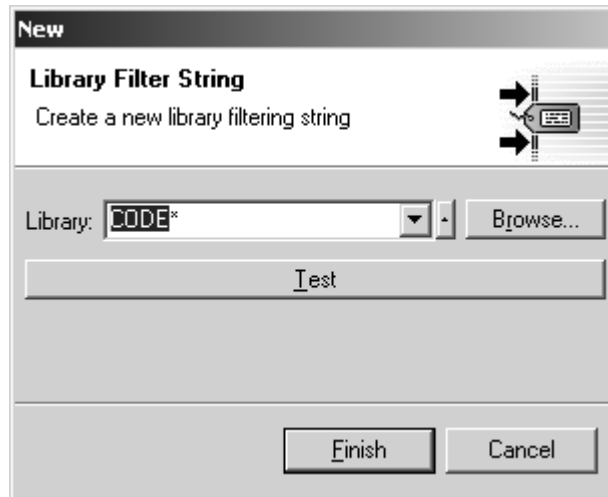
You are going to create a filter to specify the libraries you want to work with, so they will show in the RSE object list. We want you to create a filter that shows all libraries on the iSeries with the name **CODExxxxxx** and **VARxxxxxxx**, xxx being any character.

- Specify a name for this filter by keying into the **Filter name** entry field:

All CODE and VAR libraries

- Press the **Add...** push button, beside the **Filter Strings** list

The following dialog will come up

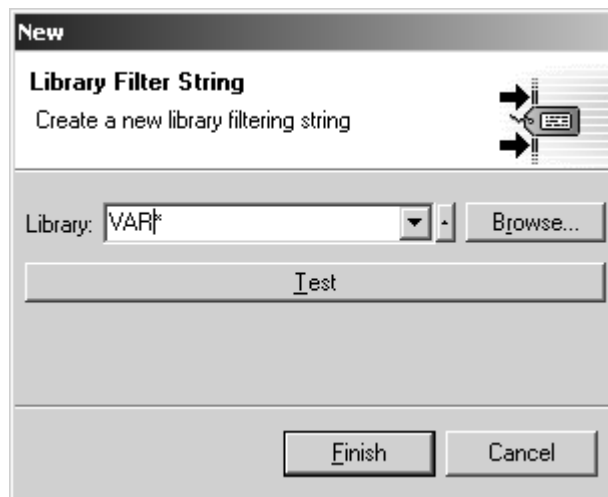


- Specify the first filter string that selects the libraries starting with CODE, by keying into the **Library:** entry field **CODE*** , using the * wild card character.
- Press the **Finish** push button

Back in the **New Library Filter** dialog:

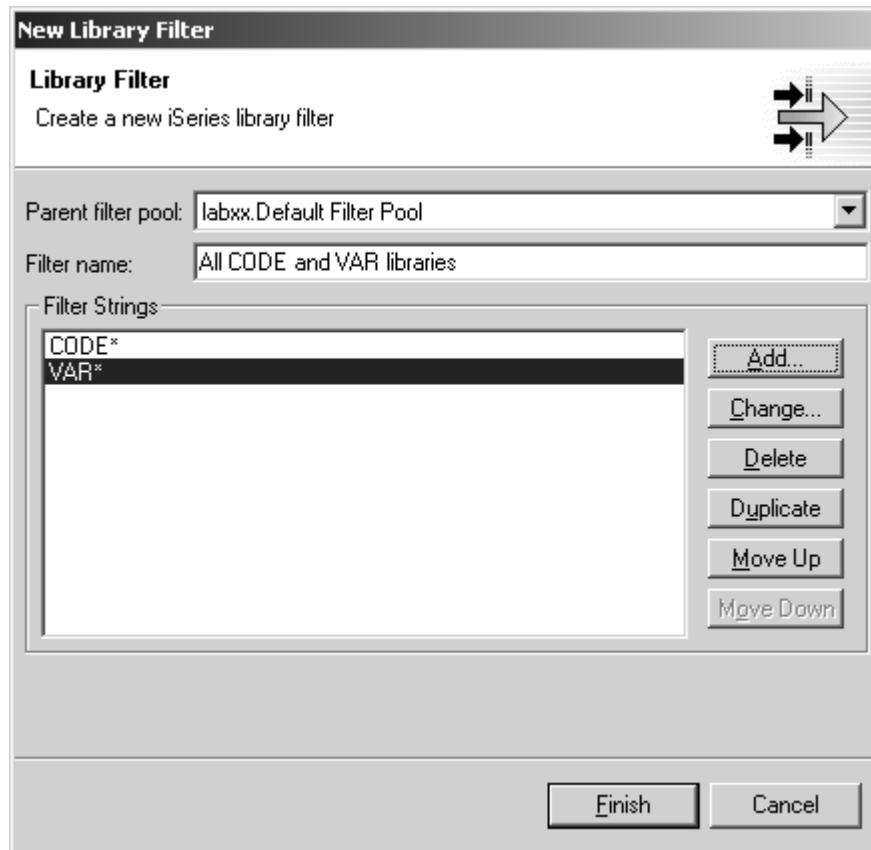
- Press the **Add** push button.

The **New Library Filter** string dialog shows up again.



- This time, specify the Filter string to select all libraries starting with **VAR***
- Press the **Finish** push button.

The Library Filter dialog will now show the 2 Filter strings you specified:



- Press the **Finish** push button on this dialog.

You are now back in the main workbench dialog.

You will see the list in the RSE perspective being expanded to include your filter.

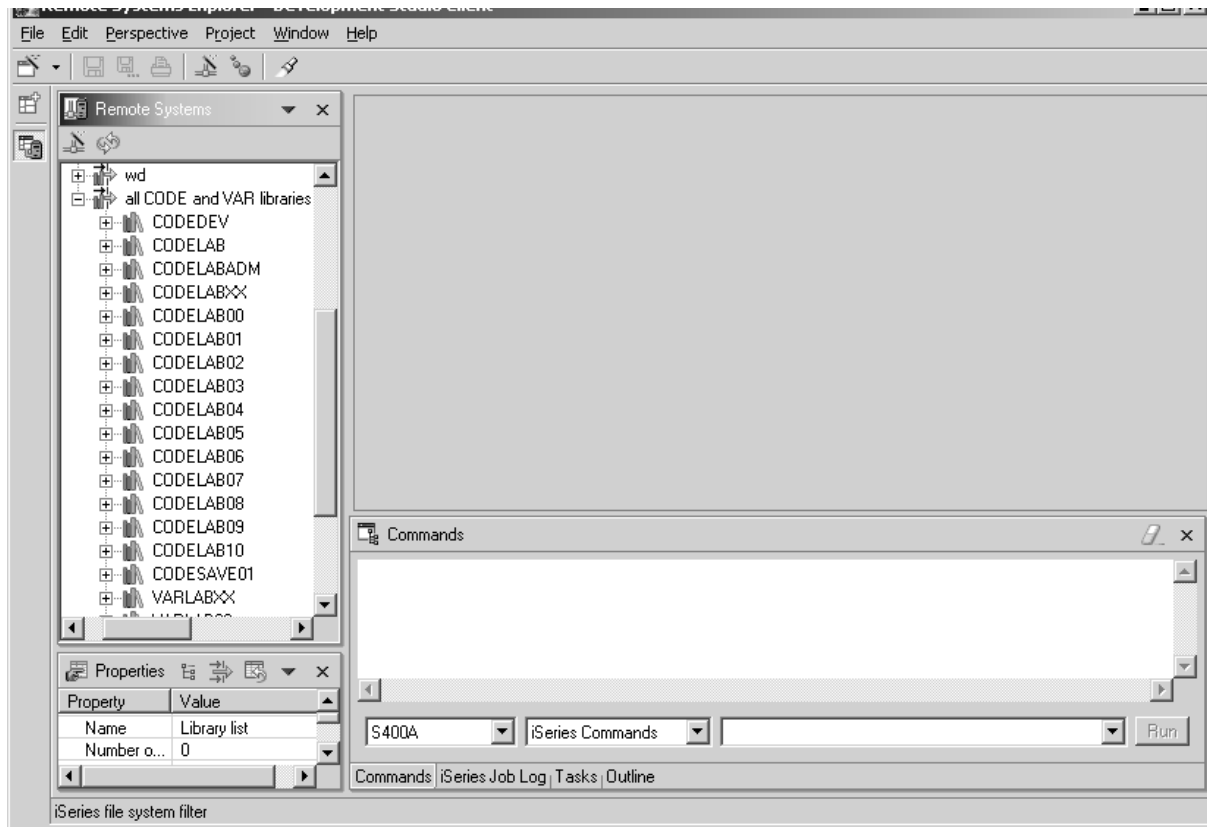
- Expand **your new filter**.

If this is your first attempt to use this connection, you will be prompted to **Sign on** to the iSeries.



Use Userid **WDSCLABxx** and password **WDSCLABxx**

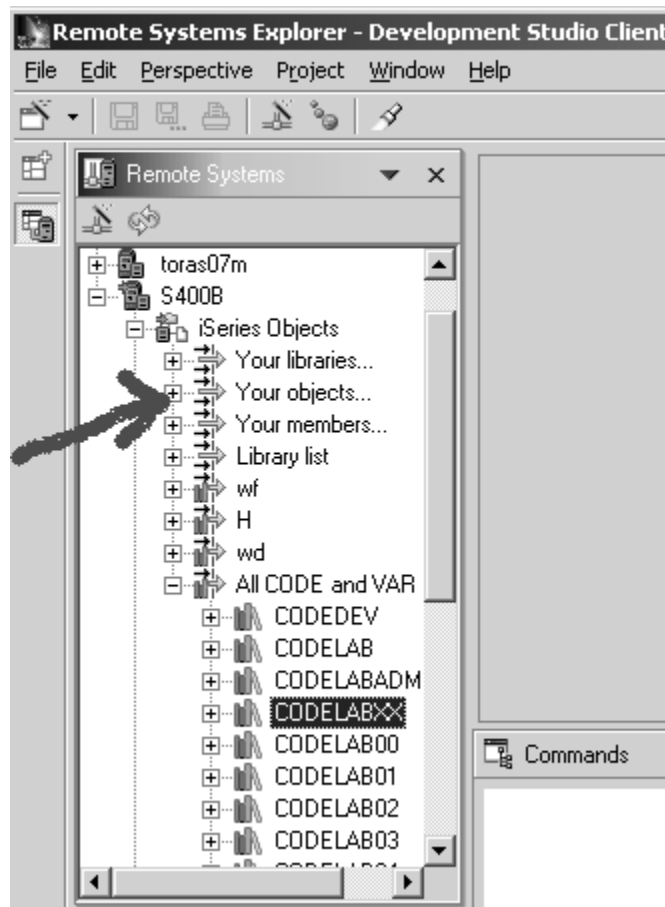
Back in the workbench in the RSE perspective you will see the libraries in your filter.



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

Lets now create an object filter

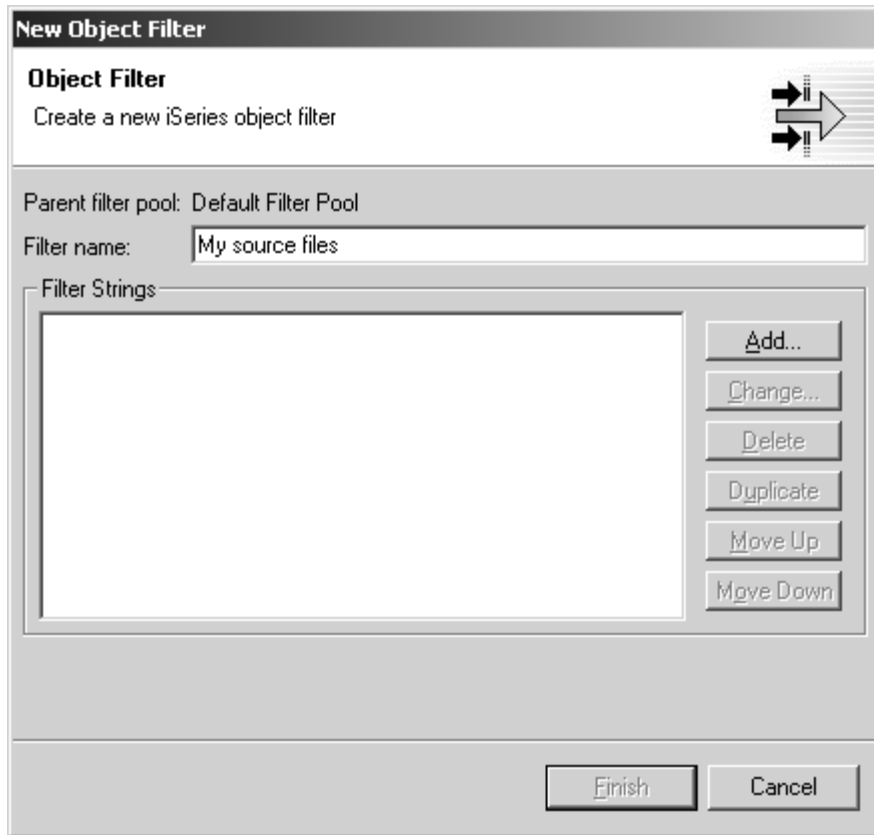
In the RSE list, under the connection node you are using,



- Find the **Your objects...** Node,
- Expand it.

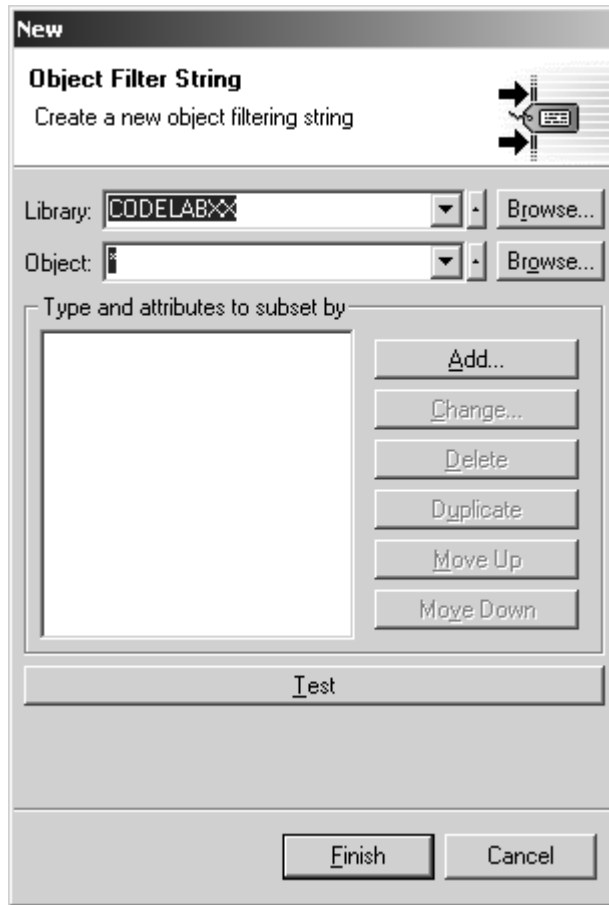
This will show the New Object Filter dialog.

Create a filter to show all your source files in your **CODELABxx** library



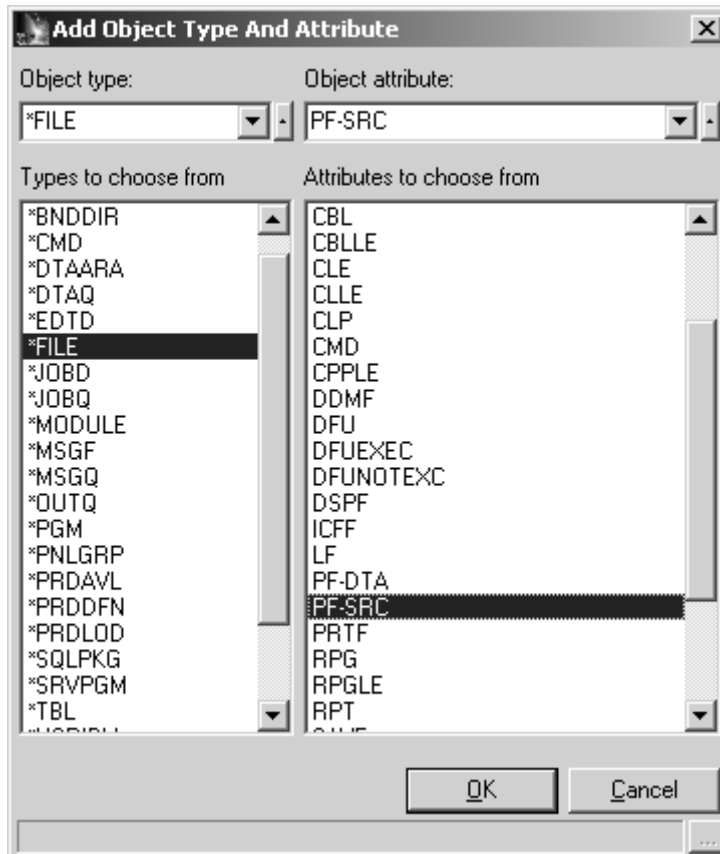
- Specify the **Filter name: *My source files***
- Press the **Add...** Push button, beside the **Filter strings** list

In the Object Filter string dialog



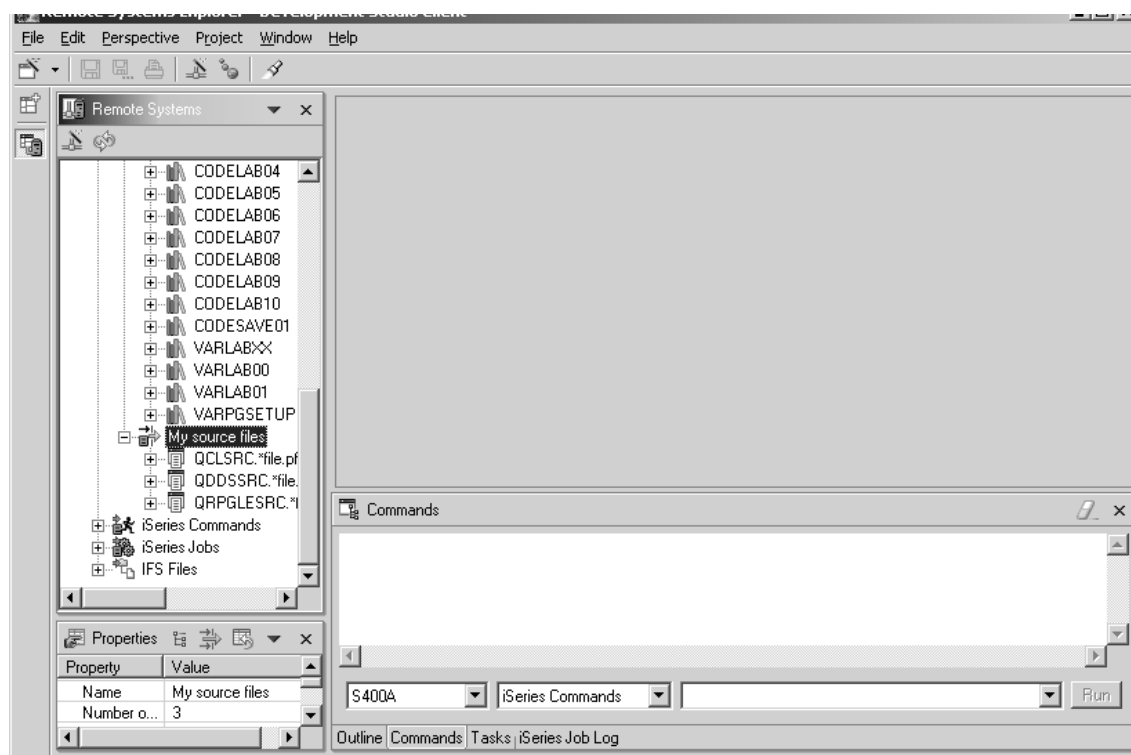
- Specify your **library**: **CODELABxx**
- For object leave the * wild card character since we want you to list all source files.
- Press the **Add...** Push button, beside the **Type and attributes to subset by** list .

In the **Add Object Type and Attribute** dialog



- Select ***FILE** as the Object type to be chosen.
- Select **PF-SRC** as the Object attribute to be chosen.
- Press the **OK** push button on this dialog.
- Press the **Finish** push button on the **Object Filter String** dialog.
- Press the **Finish** push button on the **New Object Filter** dialog.

The new filter will show up in the RSE list



You got the idea how to create filters and tailor your development environment. Filters can also be specified for non iSeries servers and your local system.

Now you will work with the objects you have in your RSE list.

Assuming you want to edit the member **Payroll**, you just:

- Right mouse click on the member **Payroll**.
- From the pop up menu select **Open with**.
- From the sub menu select **LPEX editor**.

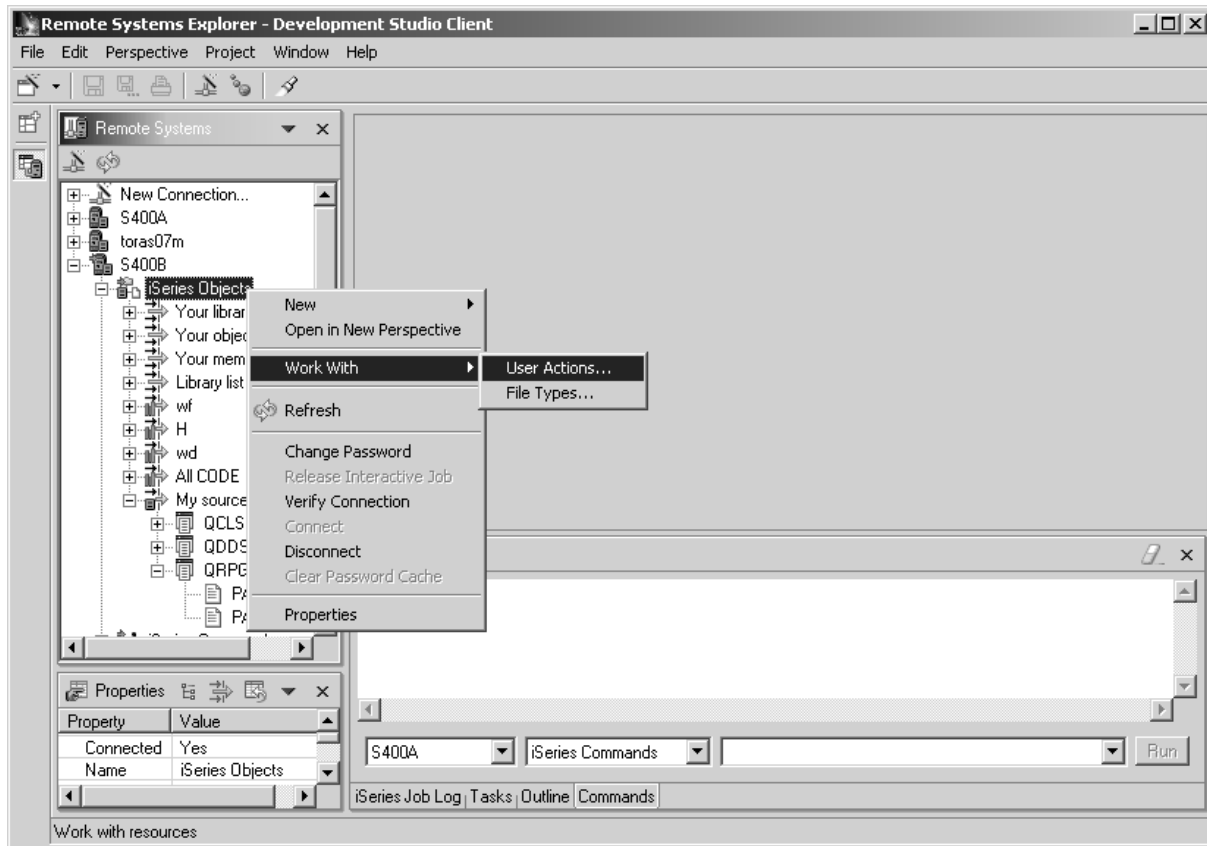
This will download the source member and open the editor with this member.

After you have edited the member you could save it and then compile it from the RSE list by using the pop up menu options on this member.

You can also create your own actions in addition to the default actions.

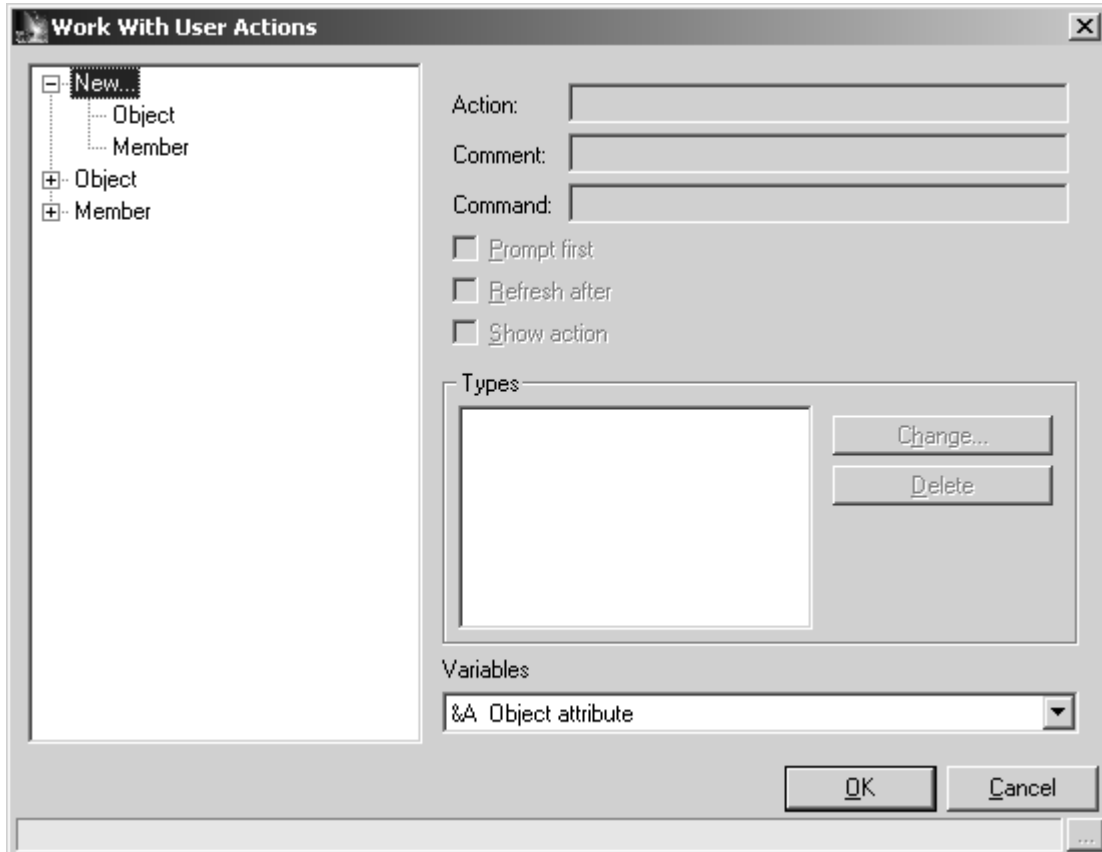
Creating a user action

In the RSE perspective, locate your **connection node** and the **iSeries Objects** node underneath it.



- Right mouse click on the **iSeries Objects** node.
- Select the **Work with** option from the pop up menu.
- Select the **User Actions** option from the sub menu.

You will see the **Work with User Actions** dialog



- Expand the **New** node in the list.
- Click on the **Object** node.

The New User Action dialog appears

New User Action

User Action
Define a new user-defined action

Action: Copy source file

Comment: Copy source file with data

Command: CRTDUPOBJ OBJ(&N) FROMLIB(&L) OBJTYF

Prompt first

Refresh after

Show action

Finish Cancel

We want you to create a user action that copies a source file with data to a new source file.

- Enter a name for the user action in the **Action** entry field: **Copy source file**
- Enter a comment in the **Comment** entry field.
- Key in the COMMAND to execute
**CRTDUPOBJ OBJ(&N) FROMLIB(&L) OBJTYPE(&T) NEWOBJ(QJUNKSRC)
DATA(*YES)**

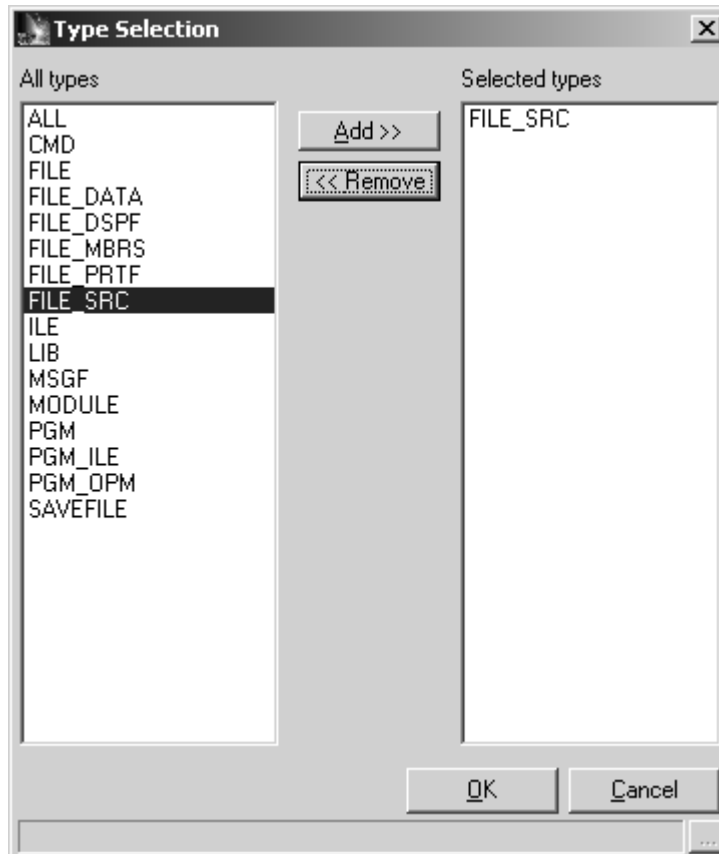
The name of the new source file is QJUNKSRC

- Select the **Refresh after** check box.
- Press the **Finish** push button.

You will be back at the **Work with User Action dialog**.

- Press the **Change...** push button, in the middle of the dialog.

The Type selection dialog will show up



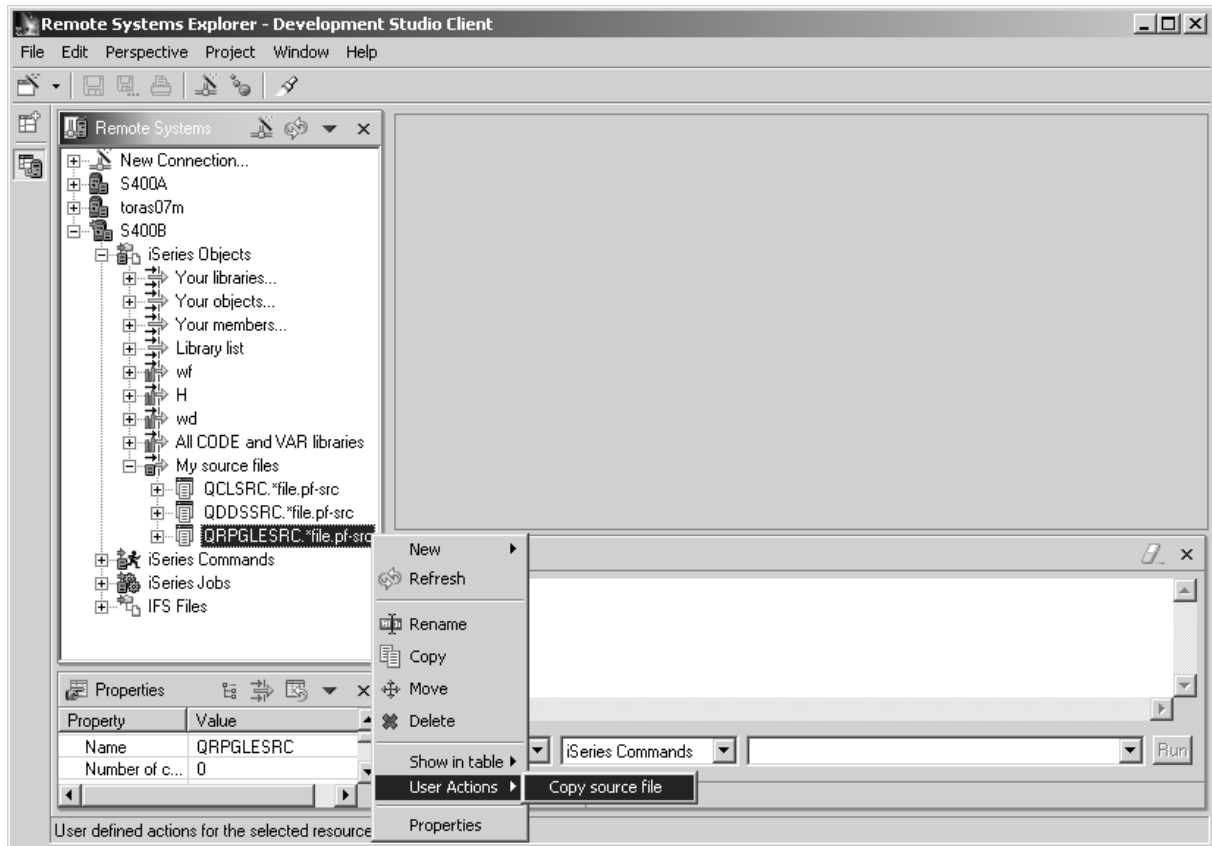
Since we want this **User Action** only to show up for **source files**, you will have to remove ***ALL** from the **Selected types** list and add **File_SRC** to it.

- Select **FILE-SRC** in the left list box.
- Press the **Add>>** push button in the middle between the list boxes to add it to the Selected types.
- Select ***All** in the right list box.
- Press the **<<Remove** push button in the middle between the list boxes.
- Press the **OK** push button.

Now, only when you right mouse 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.

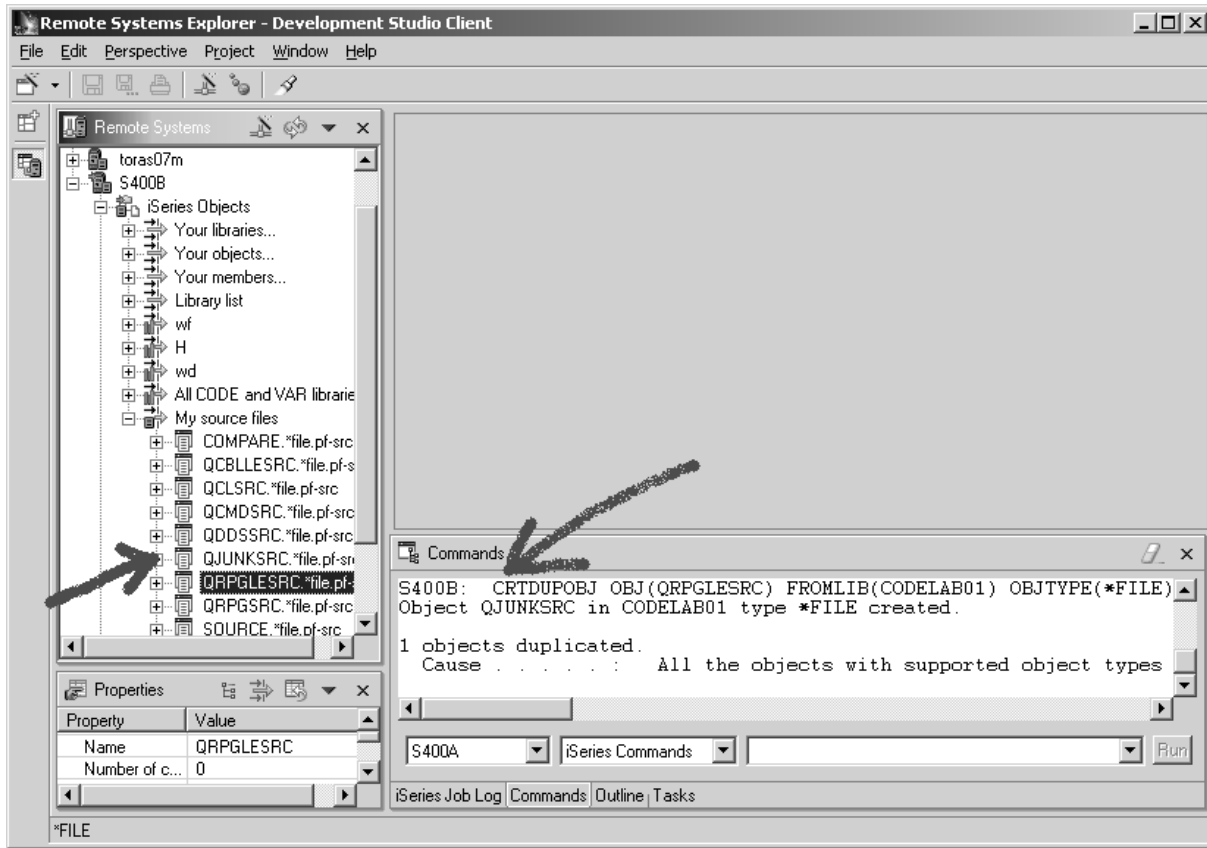
Back in the workbench and the RSE perspective, give it a try.

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



- Expand the **filter**, if it is not already expanded.
- Right mouse click on the **QRPGLESRC** file.
- Select **User Actions** from the pop up menu.
- Select **Copy source file** from the sub menu.

The file gets duplicated and the list gets refreshed, your new source file will show in the list.



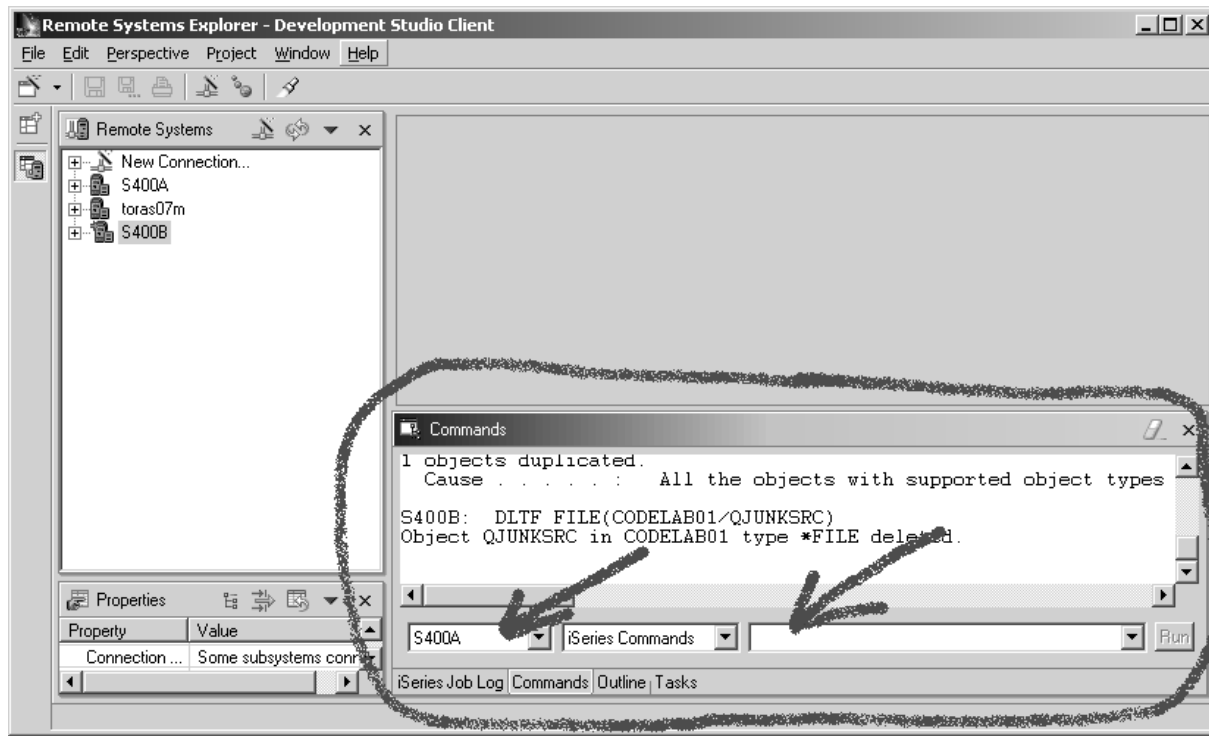
You can check the messages of the CL commands you are running in the RSE job by looking at the Commands view that by default shows up in the right hand side pane of the work bench. The arrow in the figure above points to it.

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

- Right mouse click on its **node** in the list.
- Select **Delete** from the pop up menu.

Running commands from the RSE

The Commands view should be up in the workbench



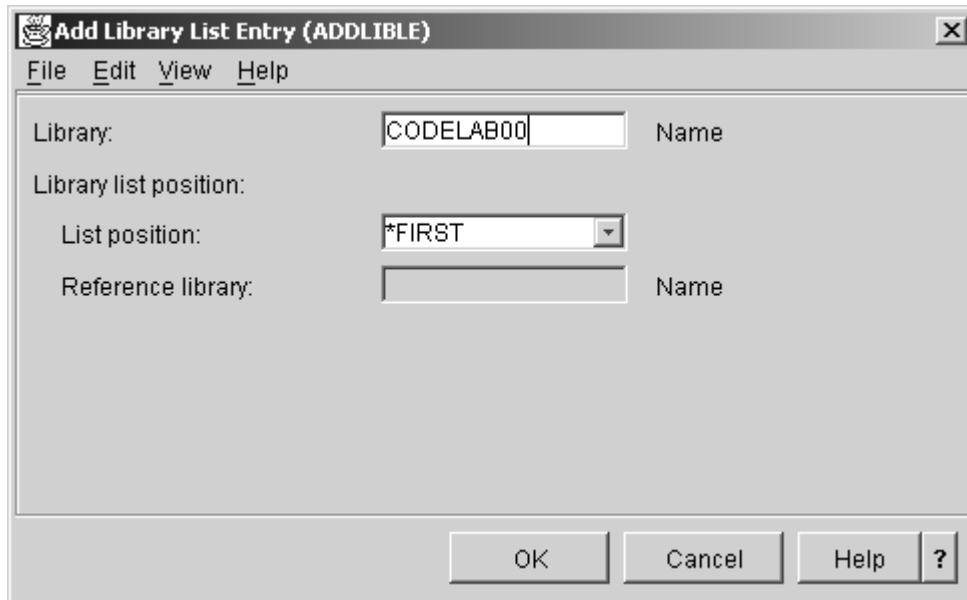
If not

- Click on the **Perspective** menu option in the workbench.
- Select **Show view** from the sub menu .
- Select **Commands** from the sub menu if it is available
 - Otherwise select **Other...** from the sub menu,
 - Expand the **Remote Systems** node in the **Show view** dialog,
 - Select **Command** in the expanded branch of Remote Systems,
 - Press the **OK** push button in the **Show view** dialog.

Now in the Command view you can select the iSeries server you want to run the command on, the left arrow in figure above points to the combo box.

- Select **your server** in the combo box.
- Key in an iSeries command for example **?ADDLIBLE**

The question mark is there to display a prompt screen



In the prompt dialog

- Key in **CODELAB00**, that will add this library to the library list of your RSE job on the iSeries server.

You could also use the **iSeries** commands node in the RSE view underneath the iSeries objects node and run predefined commands or define your own commands.

We hope this exercise gave you a first taste of the capabilities the RSE perspective provides to iSeries Application Developers.

Congratulations!

You have successfully completed the Introduction to CODE lab. More material can be found at our web site at <http://www.ibm.com/software/ad/iseriess>

Happy CODEing!