UNISYS

ClearPath OS 2200 IDE for Eclipse
Getting Started Guide

Version Control

Version	Date	Summary of Changes
1.0	4 Mar 2011	Initial draft
1.1	20 Apr 2011	Added template notes plus C and FORTRAN info.
1.2	5 May 2011	Added info on obtaining updates
1.3	25 May 2011	Minor updates
2.0	Feb 2012	Upgrade for Eclipse 3.7. Added UCOB debug notes.
2.1	Mar 2012	Added more Unisys branding. Added RDMS-JDBC client information.
2.2	June 2012	Updated for Eclipse 3.7.0.4 IC. Includes MHFS topics.
2.3	Oct 2012	Added info on Java and Unisys RAs
3.0	July 2013	Based on 3.7.2 IC2. Moved Java and Unisys RAs to another guide.

Contents

Introduction

Overview

Traditionally many OS 2200 clients have used editors like ED and IPF to maintain their 3GL programs such as COBOL, FORTRAN and C. Unisys has enabled the OS 2200 environment to run many Java solutions including a Java Virtual Machine (JVM) that can run on a J-Processor specialty engine or on OS 2200, Java access to DMS/RDMS/BIS databases and the JBoss Java EE environment. Eclipse is the leading open source IDE and is based on Java. Unisys has added a plug-in to Eclipse to support the development of legacy 3GL programs developed in COBOL, FORTRAN, C and other languages. The Unisys plug-in also assists in the development of composite applications.

The Unisys ClearPath OS 2200 IDE for Eclipse provides many features including:

- Windows containing the entire program source
- Easy navigation being open windows
- Windows GUI features like search, copy/paste & drag/drop
- Content Assistant to provide COBOL structures and statements for the programmer
- Use of different colours for reserved words, comments and variables
- Error windows for compilations with links to source code lines
- Ability to update OS 2200 source files including ECL and COBOL
- Ability to build a project (e.g. compile programs) with compilation on the OS 2200 host
- Compare differences with older code versions
- Support COBOL, Java, Java EE, C and other development languages from a single tool
- Supports interactive PADS debugging for UCS programs.

As the ClearPath OS 2200 IDE for Eclipse runs on the programmer's PC, OS 2200 CPU cycles associated with editing code are off-loaded from the OS 2200 host. The productivity of OS 2200 programmers is expected to improve from using the IDE. The ClearPath OS 2200 IDE for Eclipse should also assist OS 2200 clients in finding new programmers as many OS 2200 proprietary commands and tools are not used.

Purpose

The purpose of this document is to assist OS 2200 programmers in understanding and using the Unisys ClearPath OS 2200 IDE for Eclipse for 3GL development through screen snapshots and written descriptions. This document is based on the Unisys ClearPath OS 2200 IDE for Eclipse 3.7.2 IC2 release.

This document was compiled with the help of numerous Unisys colleagues both in the Eclipse engineering team and field delivery staff. Any changes or suggestions to this document should be emailed to Robert Jamieson (mailto://robert.jamieson@unisys.com).

Projects are the Core of an IDE

Eclipse is an open source IDE that Unisys has developed plug-ins for so it works with OS 2200 environments. IDEs are designed to work with projects and Eclipse is no different. However Unisys has provided some additional features like OFCS to assist to the OS 2200 developer. OFCS does not use Eclipse projects and therefore some functionality may not be available when using OS 2200 files and elements that do not belong to an OS 2200 project in your workspace.

Terminology

Throughout this document, the Unisys ClearPath OS 2200 IDE for Eclipse is often referred to as simply Eclipse.

Related Documents and Information

Eclipse IDE for OS 2200 Installation Guide (47292107-005)

CIFS for ClearPath OS 2200 User, Programmer and Administrator Reference Manual (78596137-nnn)

Definitions and Acronyms

Acronym	Definition
Eclipse	Open Source IDE
IDE	Integrated Development Environment
Dorado	Unisys ClearPath Plus server running OS 2200 and Windows OS
ACOB	ANSI COBOL-74 compiler with Unisys extensions
UCOB	ANSI COBOL-85 compiler with Unisys extensions
CIFS	Common Internet File System
ADMLP	DMS2200 preprocessor
MHFS	Multi-Host File Sharing
OFCS	Open File from Configured Serrver

Table 1 - Definitions and Acronyms

Installation

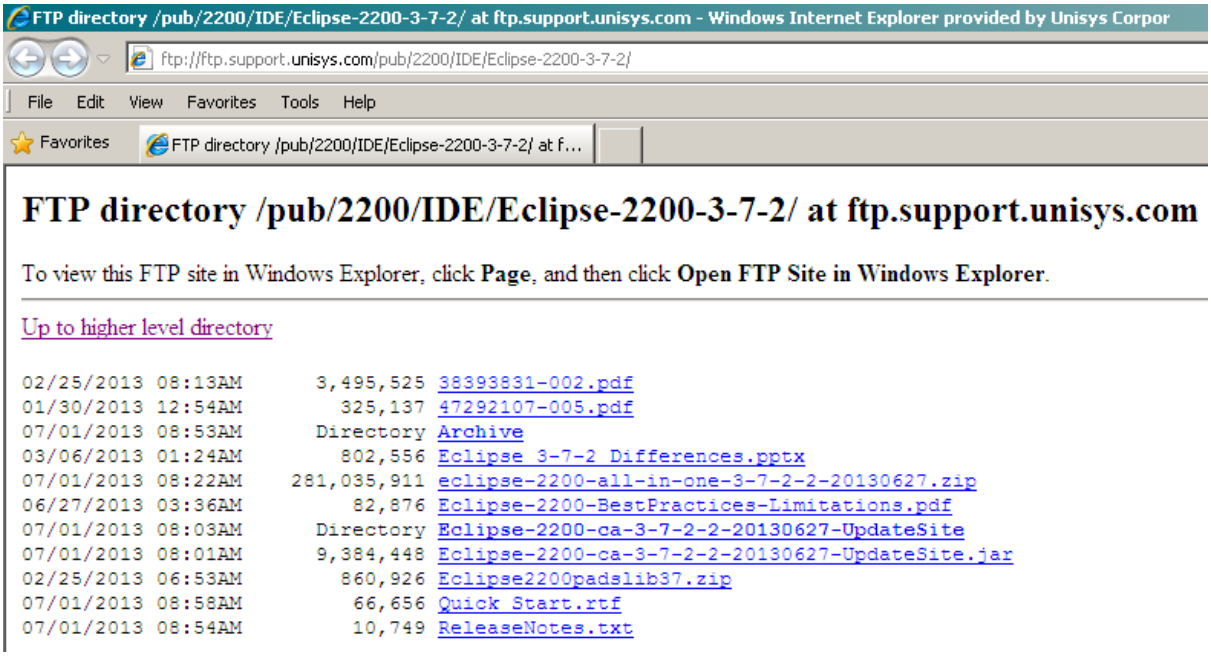
This section describes how to install the Unisys ClearPath OS 2200 IDE for Eclipse and related software products on a developer's workstation. For a more detailed description of the installation process please refer to the related production information mentioned earlier.

Hardware Requirements

The workstation should have at least 2GB memory to perform at a satisfactory level. However 4GB memory is recommended.

Download Files to Install

Download for the ClearPath OS 2200 IDE for Eclipse All-In-One is available from ftp://ftp.support.unisys.com/pub/2200/IDE/Eclipse-2200-3-7-2/. The following shows the contents of this download site.



There are two options for installing Eclipse.

Choose one of the following options:

1. Install ClearPath OS 2200 IDE for Eclipse All-In-One
   eclipse-2200-all-in-one-3-7-2-2-20130627.zip
   A package containing a complete installation of Eclipse, Web Tools, Data Tools, and other associated features including the Unisys Composite Application. For details and configuration instructions, see Section Three of the Installation Guide.
   Use this option if you are not currently using Eclipse as an IDE.
2. Install ClearPath OS 2200 IDE for Eclipse Composite Application Feature
   eclipse-2200-ca-3-7-2-2-2013-0627-updatesite.jar
   The ClearPath OS 2200 IDE for Eclipse Composite Application Feature is ready to install in your existing installation of Eclipse. For details and configuration instructions, see Section Three of the Installation Guide.
   Use this option if you are using Eclipse already as an IDE and just want to add the Unisys OS 2200 feature.

Install the Java Environment

Double click on the ***jdk-nnnn-windows-i586.exe*** file. This will launch the Windows Installer to install the Java environment on your workstation. Take the defaults (recommended).

Installing ClearPath OS 2200 IDE for Eclipse

- Identify a folder on your workstation to install Eclipse..
- Unzip the eclipse-2200-all-in-one-3-7-2-2-20130627.zip file to this folder.

Host OS 2200 Software Dependencies

The host OS 2200 Software Dependencies can be found in the Installation Guide. Basically you need CIFS (6R2 or later) and CPCOMM/CPCOMMOS.

Configuring an OS 2200 Telnet Session

Eclipse uses a Telnet session to the OS 2200 host for some functions so a Telnet process needs to be configured in CPCOMM/CPCOMMOS. An example is:

```
PROCESS,TELRSI PASSWORD,RSI                          . Telnet
```

SILAS also needs to be configured for Telnet. For example:

```
PROCESS,RSDCSU   INTERNET-ADR,INTADD ;
                 TELNET-ATTACH-NAME,TELRSI TELNET-ATTACH-PASSWORD,RSI ;
                 TP0-ATTACH-NAME,TP0RSI    TP0-ATTACH-PASSWORD,RSI
```

CIFS Parameter Setting

The CIFS parameter CIFS$WAITROLBAK determines how long before CIFS responds to Eclipse when the OS 2200 file is rolled-out. By default, this parameter has a value of 600 seconds. It is recommended to set this value to 1 in the CIFS-BACK runstream so the user gets an immediate response.

Starting ClearPath OS 2200 IDE for Eclipse

Eclipse 3.6 Migration Considerations

This section only applies if you are installing Eclipse 3.7 on a workstation that was running Eclipse 3.6.

Eclipse maintains host and login details in two XML files that are local settings for each user. The files are HostAccount.xml and LoginAccount.xml. The location of the XML files is

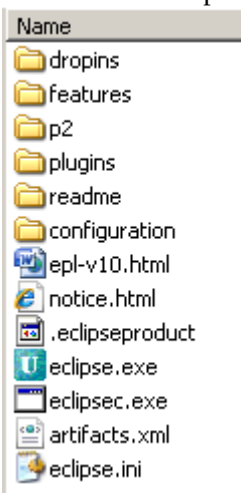
Windows XP
C:\Documents and Settings\<User>\Local Settings\Application Data\Unisys\os2200\ide
Windows Vista \ 7
C:\Users\<User-id>\AppData\Local\Unisys\os2200\ide

The format of these files has changed with Eclipse 3.7 and the 3.6 format is incompatible. You should delete these files when installing Eclipse 3.7. Host connection and login details will need to be re-entered. However if you do intend to run both Eclipse 3.6 and 3.7 on the same workstation, you may want to save the 3.6 files to a different name when using 3.7. Restore them when using Eclipse 3.6 after saving the 3.7 files.

Launching Eclipse

Eclipse can be launched by the following steps:

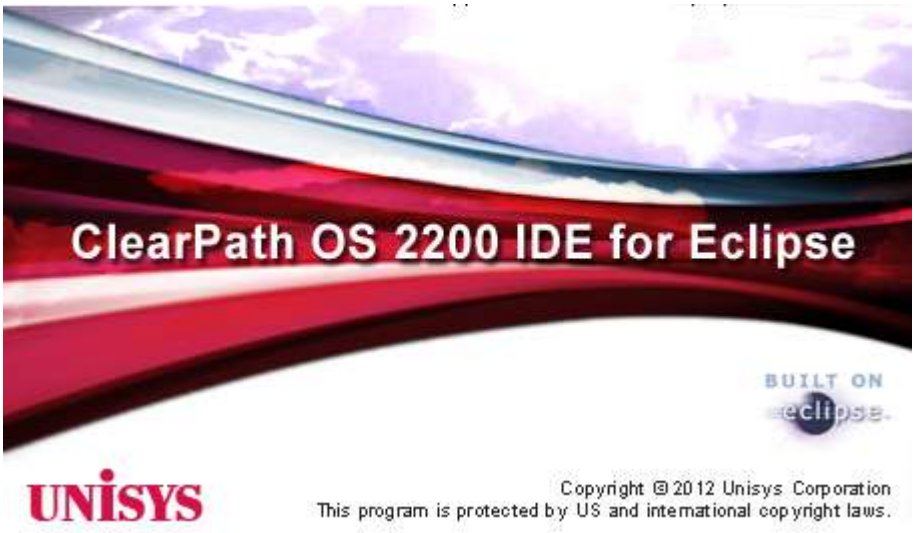
- Use Windows Explorer to expand the folder where Eclipse was installed



- Double-click on the **eclipse.exe** file

Note: It is recommended to create a short-cut from the Eclipse.exe file and place on your desktop.

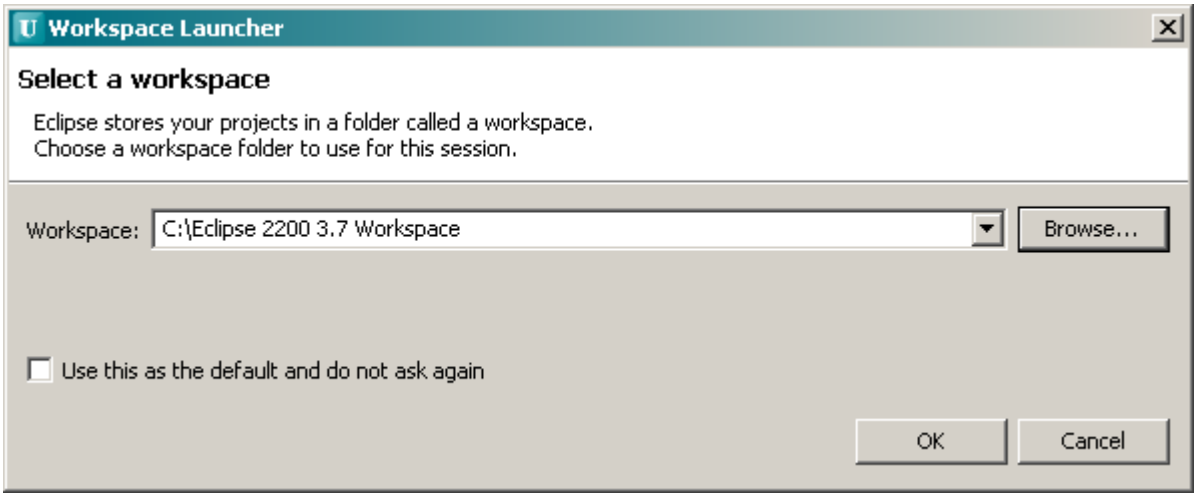
Eclipse will display the following screen. Note that Eclipse 3.7 is based on the Indigo release from the Eclipse Foundation.



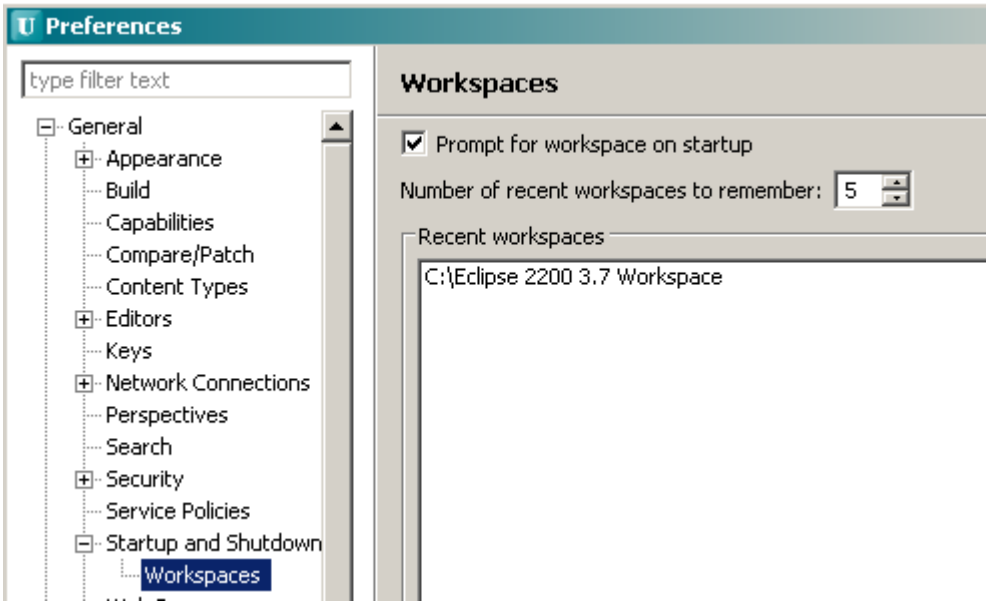
Selecting the Eclipse workspace

Eclipse will display the following window asking for the workspace location. Normally this is found in the Documents and Settings folder for the user but you can use another location. In the example below, the workspace is **C:\Eclipse3-7 Workspace**.

Check the box if you don't want this window to appear in future.



The start-up option to prompt for the workspace can be set in the Eclipse preferences. Go to **Window → Preferences → General.**



No Workspace Prompt

On occasions, Eclipse will launch with a Workspace even when the user has requested to prompt for the workspace. There have been issues with the prompt for workspace at the startup. Some of the forums they have been discussing about this and one of them is:
http://stackoverflow.com/questions/7058782/eclipse-default-workspace-problem

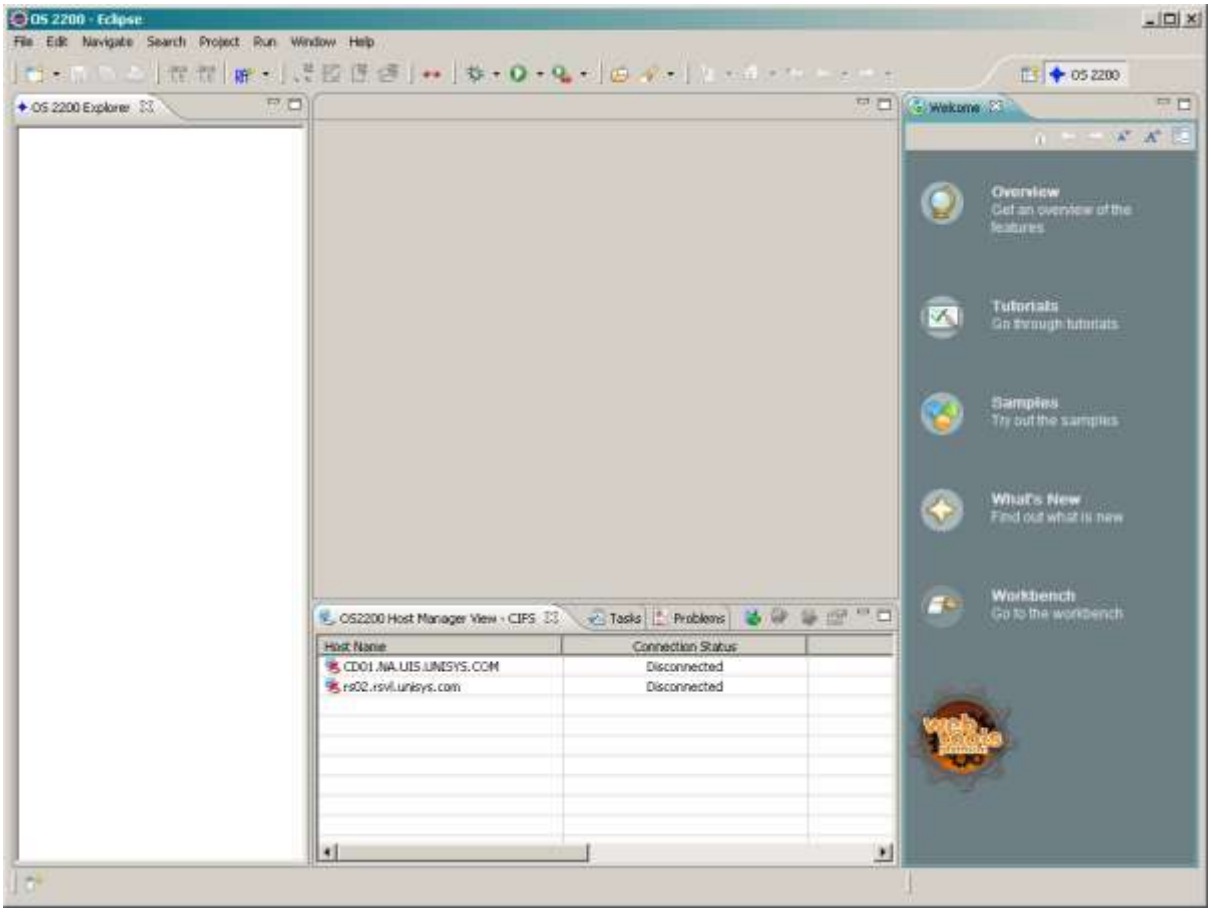
One option is to start Eclipse with the following option.

<eclipse path>/eclipse.exe –clean

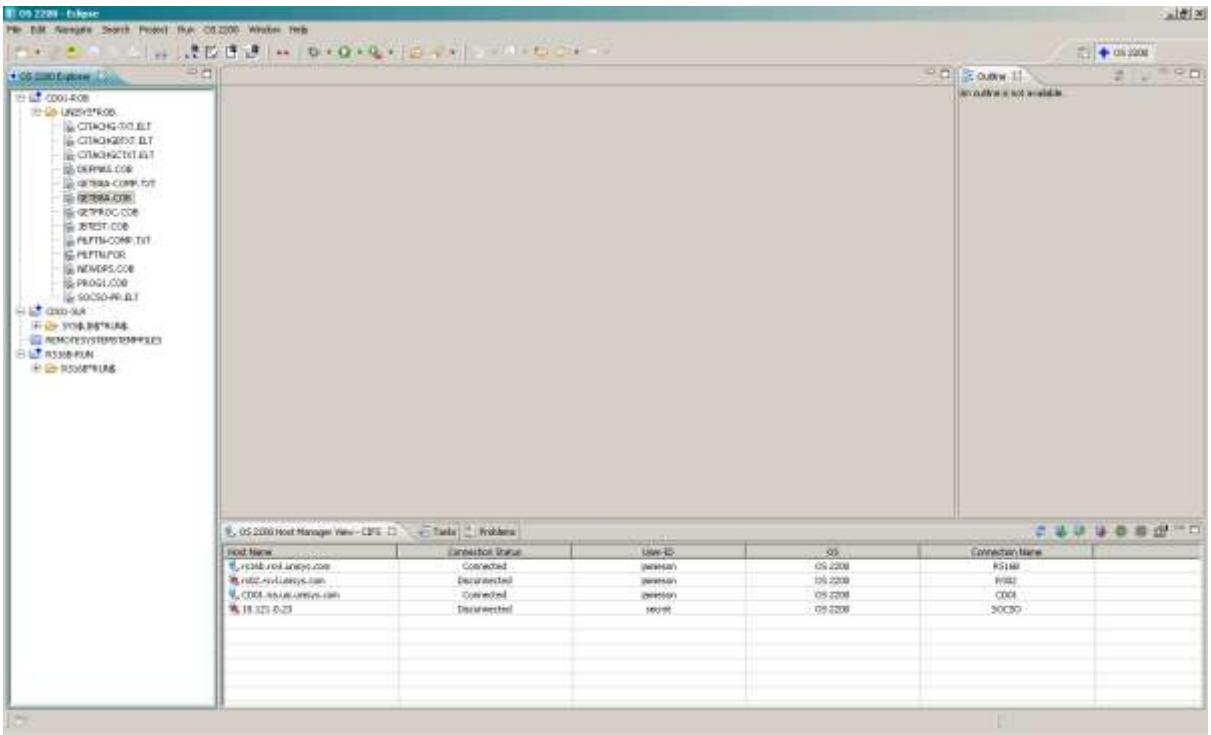
If you have created a shortcut on the desktop, right click on the shortcut and click on properties and append the key "-clean" to the target path.

Selecting the Eclipse workbench

For first time users, Eclipse will display a welcome screen similar to the following.



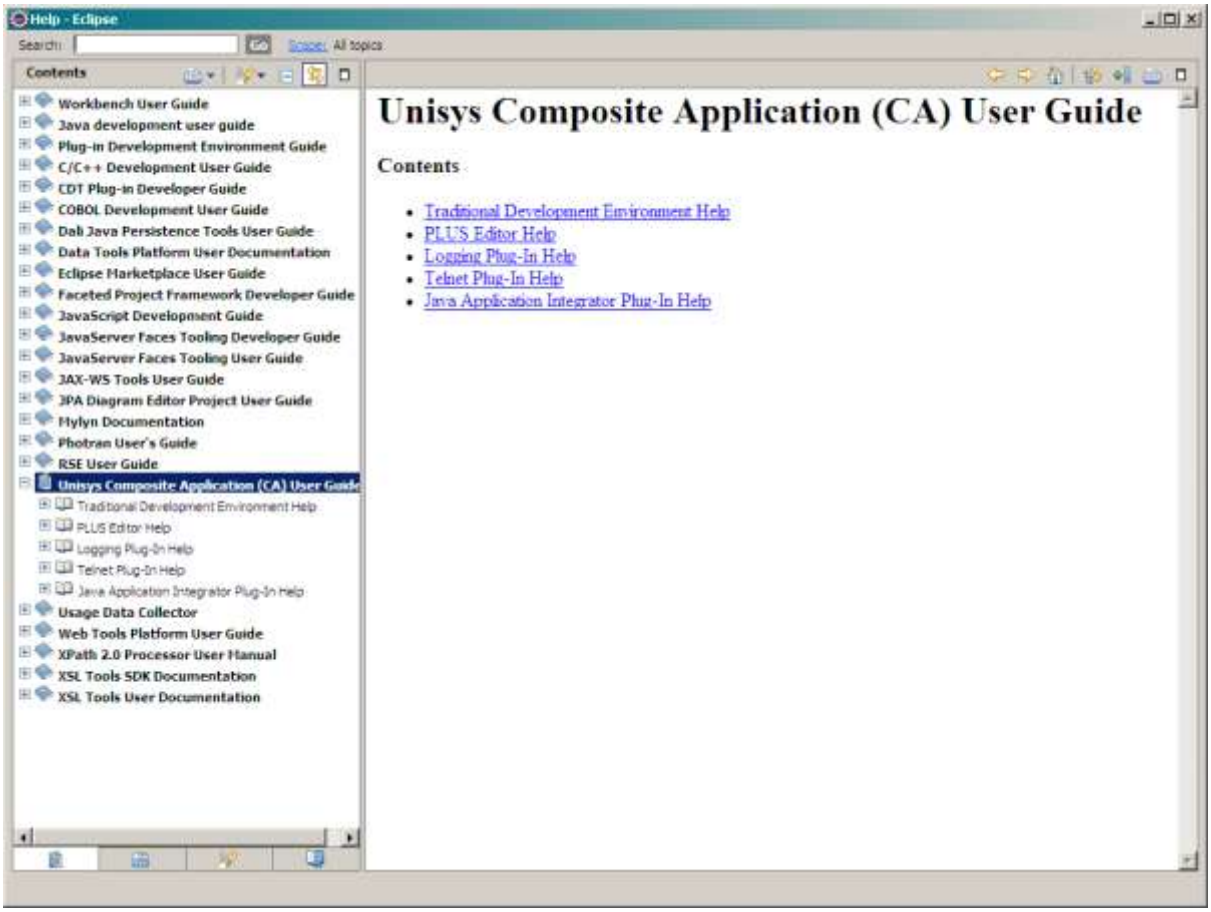
Close the Welcome view. The result will be the Eclipse workbench similar to below.



At the top right, you will see a box containing “OS 2200”. This is known as the Eclipse perspective. Different perspectives are available e.g. Java. By default the OS 2200 perspective is set as the default when installing the all-in-one release.

Eclipse Help for Unisys Plug-ins

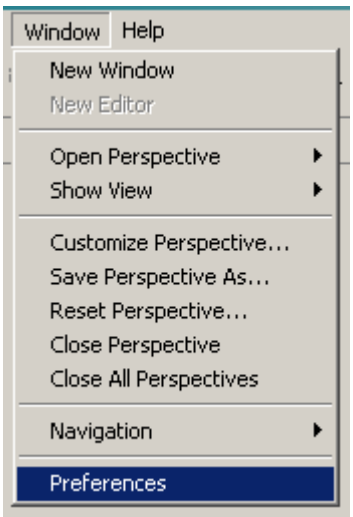
Go to **Help → Help Contents**.



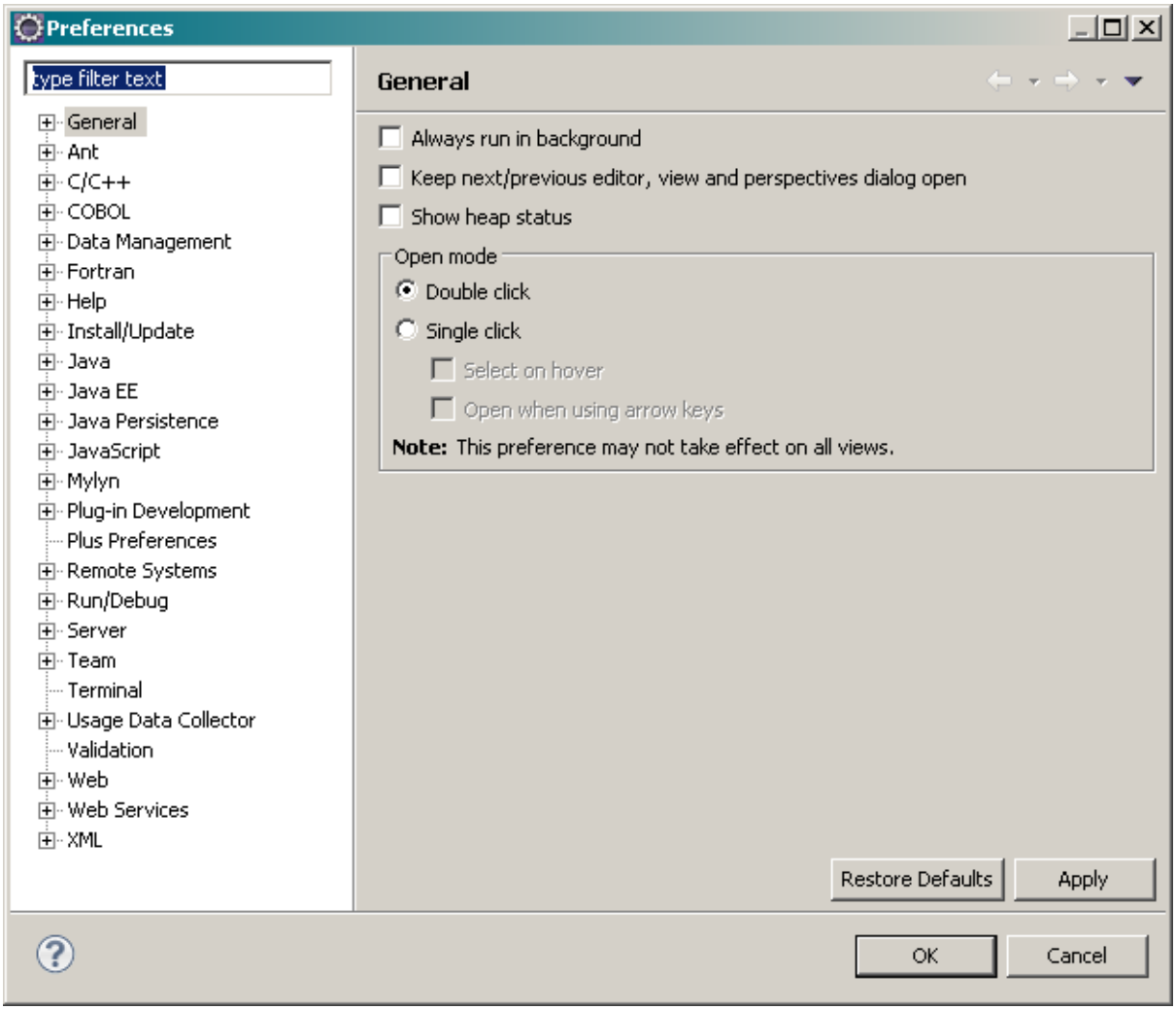
Understanding OS 2200 Projects

Configuring Eclipse Preferences

Before we start defining an OS 2200 project, we have to configure Eclipse to support the OS 2200 environment. In the menu bar, go to **Window → Preferences** and click.



The following screen will appear:



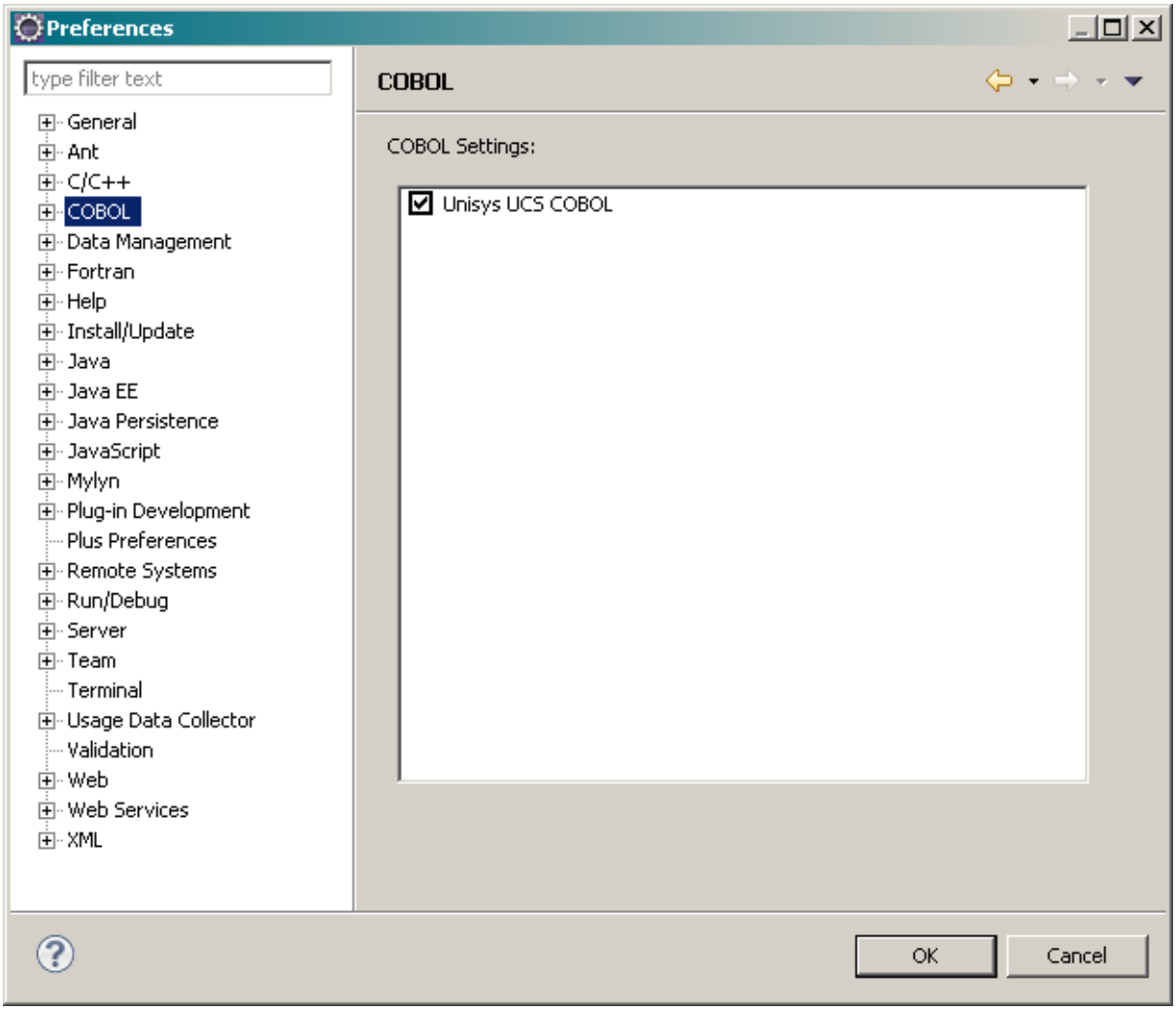
Displaying the Java Heap Status

If you check the "Show heap status" option above, Eclipse will show the Java heap status in the status line:

This information can be useful to provide in some cases as requested by Engineering. Clicking the garbage bin icon will initiate a Garbage Collection task.

Configuring the Unisys COBOL preferences

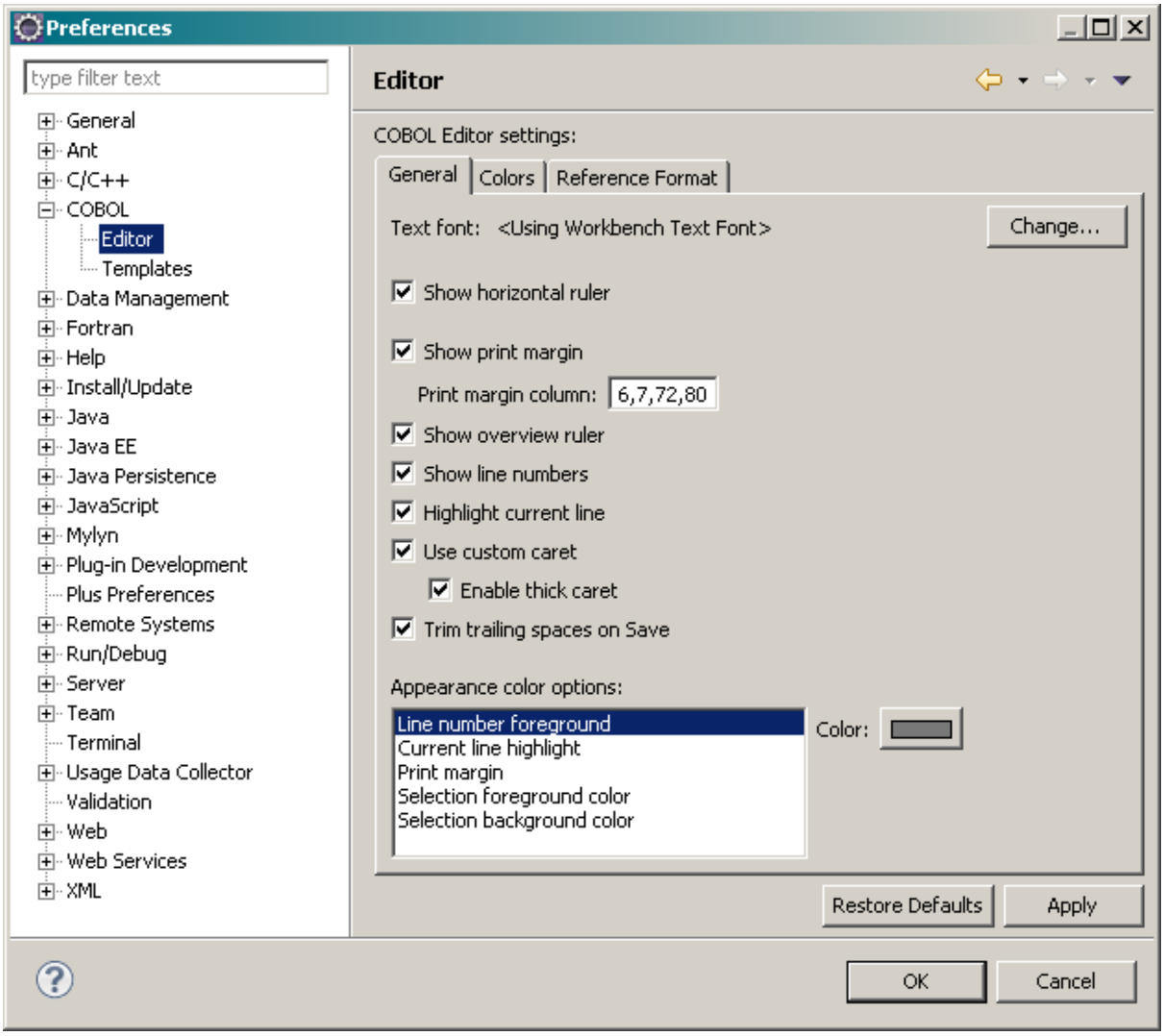
Click on the entry COBOL (and not the '+' sign) to display this screen:



Check the Unisys UCS COBOL entry. This selects the editor template that matches UCOB – the ANSI COBOL-85 compiler on the OS 2200 system. Note that ACOB (ANSI COBOL-74) source can be used with the COBOL editor as there are only minor differences between the compilers regarding the language structure e.g. format of statements. For example, TALLY is an ACOB verb but not a UCOB verb so it will not be handled as a verb when the COBOL editor displays the source.

Now expand the COBOL entry and click the Editor entry.

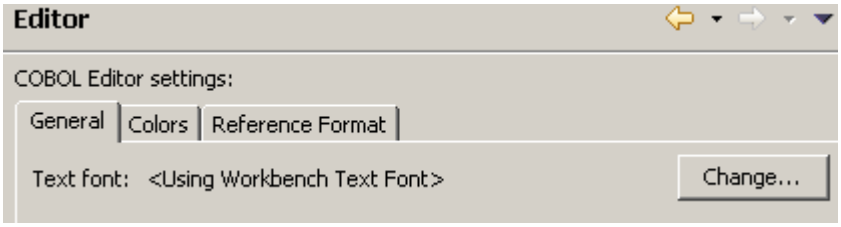
COBOL General Preferences



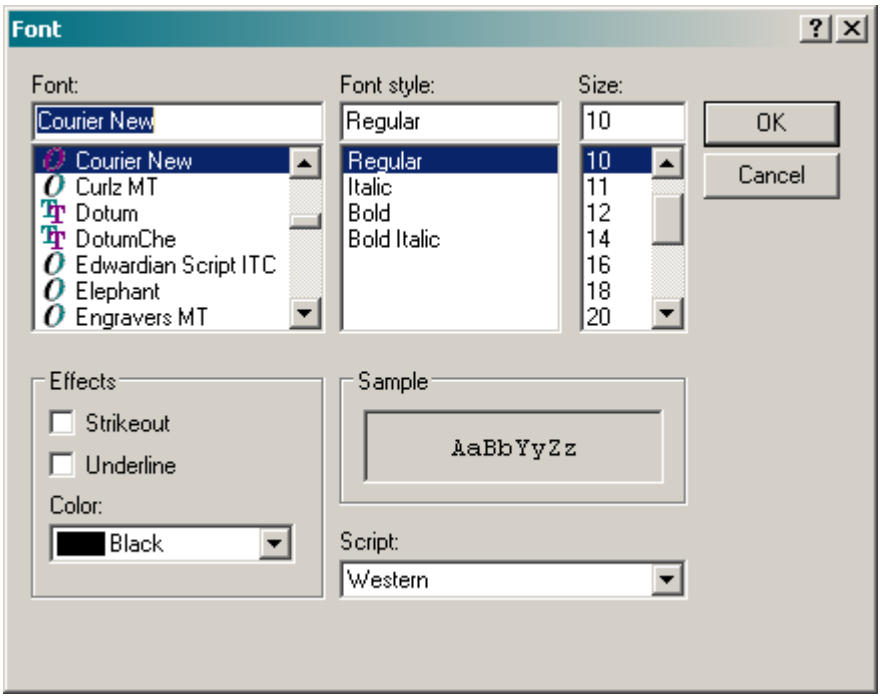
The General tab is used to set various options on how the COBOL edit pane is prepared and handled. Modifying these settings affects only the current workspace, so each user can have their own preferences.

Preference	Description
Show horizontal ruler	Displays a horizontal rules in the edit pane
Show print margin	Puts vertical divider lines as the margins mentioned
Show overview ruler	
Show line numbers	Line numbers are displayed on the left edge of the edit window
Highlight current line	Highlights the current line for easier identification
Use custom caret	Cursor being used
Trim trailing spaces on save	When a COBOL file is saved, Eclipse will trim trailing spaces from each line

On the top of the General tab, Eclipse displays the current font. You can change the font by clicking on the Change button:

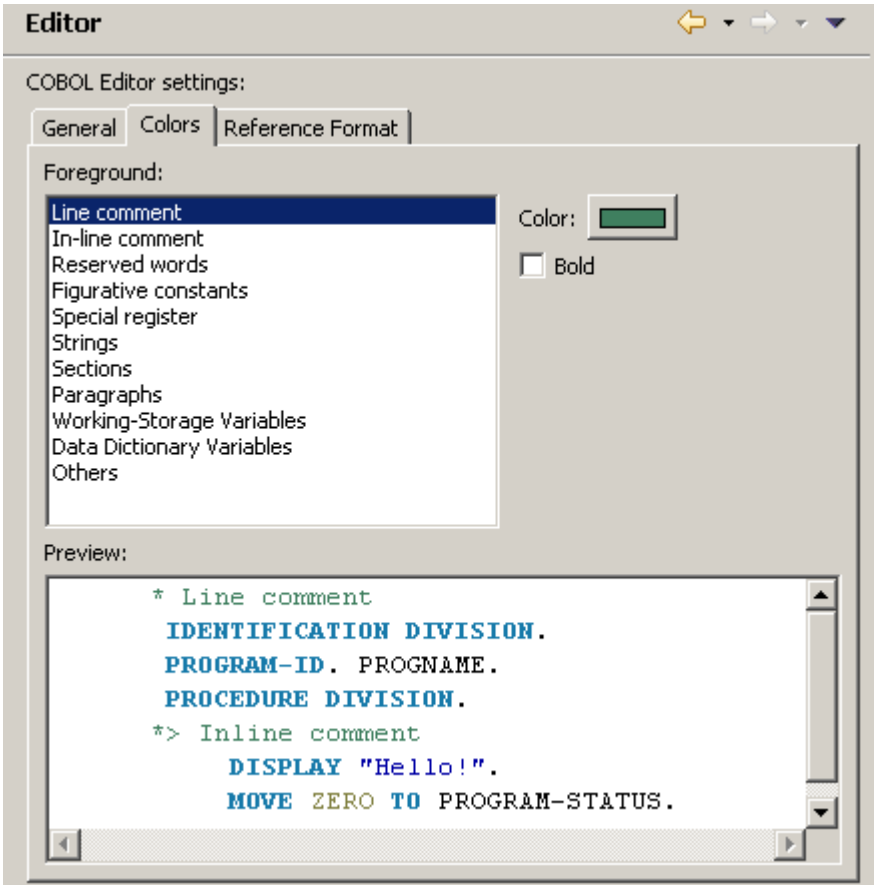


Eclipse allows you to set the font. This might be important when Character Conversion is used. For example, with Japanese it might be preferred to use the MS MINCHO or MS Gothic font. If you want to preserve columns etc then use a fixed font rather than proportional.



COBOL Colors Preferences

Click the Colors tab.

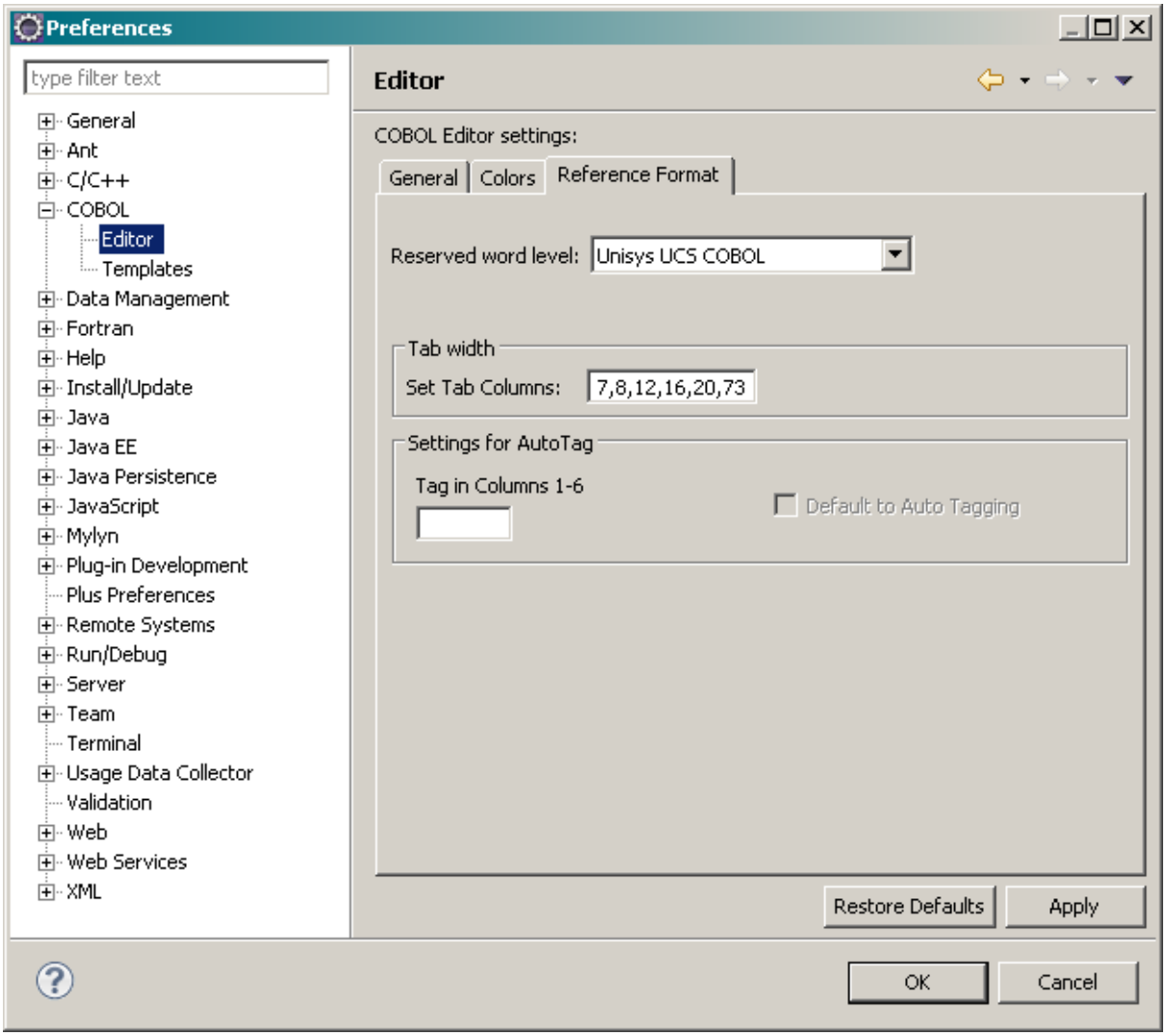


Note that Sections, Paragraphs and Working-Storage Variables do not have their colors set in real-time. From the COBOL editor, you must do a F4 to refresh the colors.

The Data Dictionary Variables is not available to all clients.

COBOL Reference Format Preferences

Click the Reference Format tab.



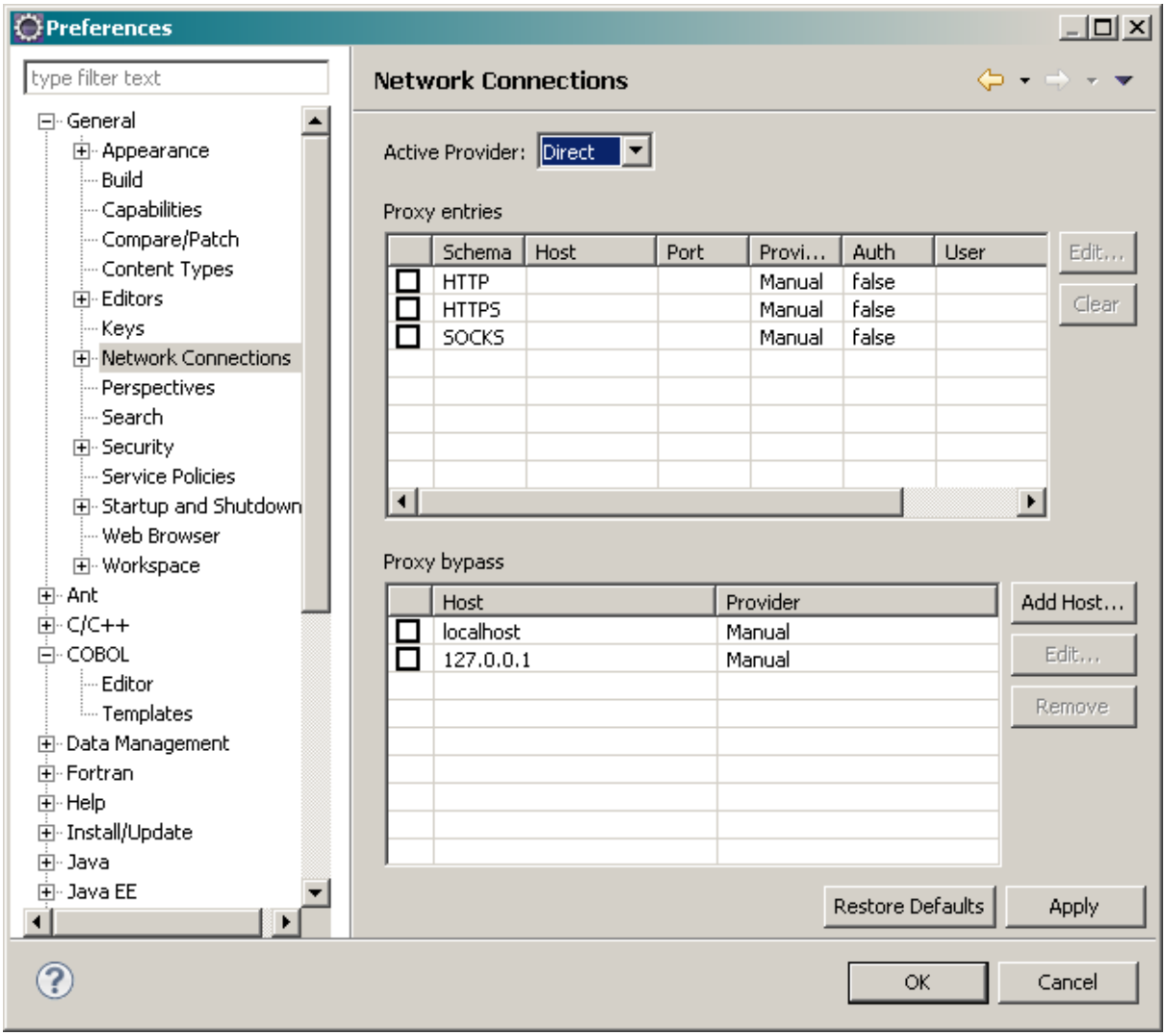
Configuring the Proxy Preferences

Some sites use a proxy server to provide internet access. When this is configured on the workstation, Eclipse will get an error message when trying to connect a telnet session to the OS 2200 host.

The problem can be resolved by setting the Eclipse network settings to the correct value.

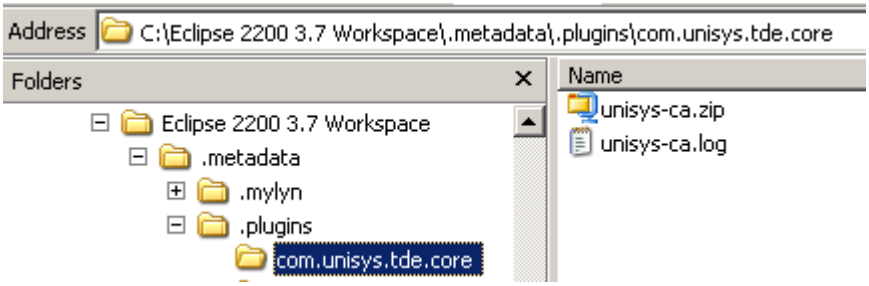
Go to **Window → Preferences** and then expand General in the tree structure. Then highlight Network Connections. Set the Active Provider to Direct.

Refer to the figure below.

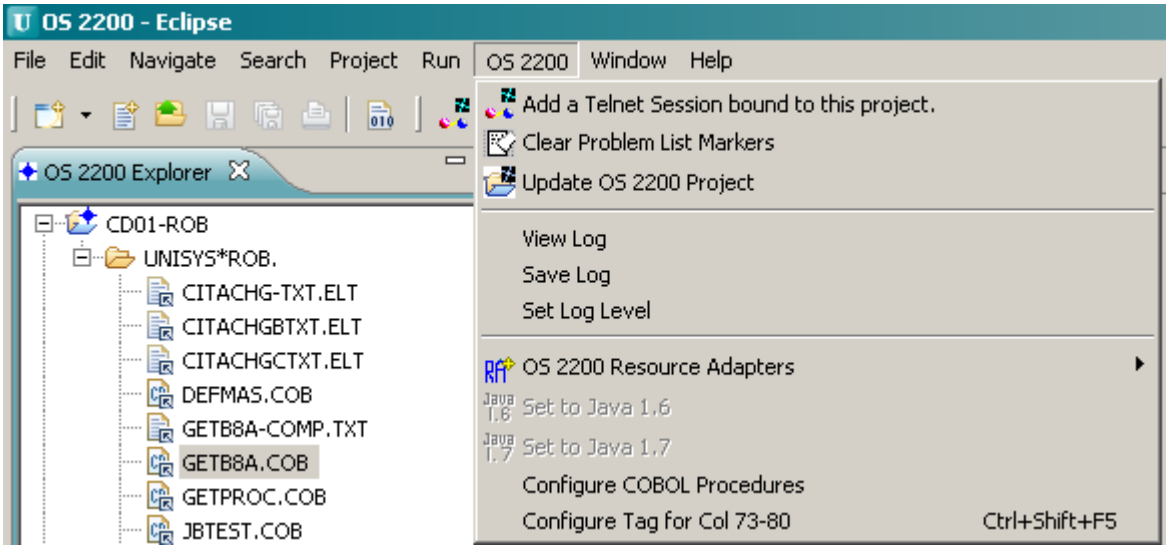


Setting the OS 2200 Log Level

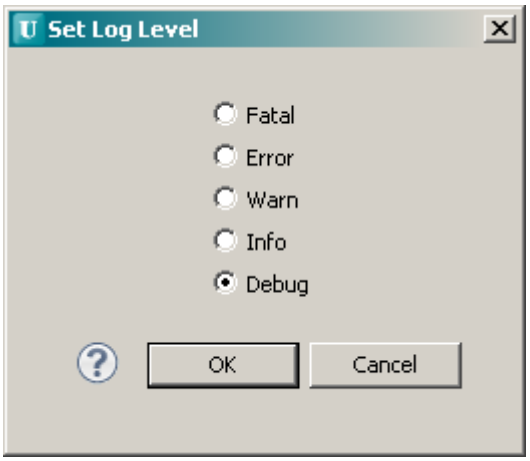
Eclipse writes a log file for the Unisys plug-ins to a file called **unisys-ca.log** into the selected workspace folder:



This log file could grow in size so the ability to set the log level was introduced. To set the log level, go to **File → OS2200 → Set Log Level:**



The current log level setting is displayed.



The user can choose the logging level to be logged. There are 5 different options arranged according to the priority.

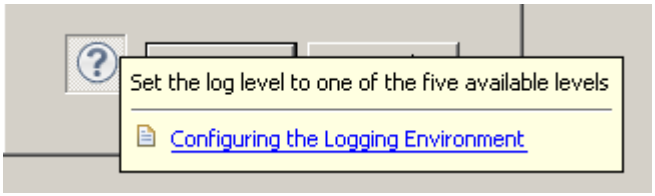
- **Fatal**
  This level will designate very severe error events that will presumably lead the application to abort. Logs only Fatal data.
- **Error**
  This level will designate error events that might still allow the application to continue running. Logs Error and Fatal data.
- **Warn**
  This level will designate potentially harmful situations. Logs Warn, Error and Fatal data.
- **Info**
  This level will designate informational messages that highlight the progress of the application at coarse-grained level. Logs Info, Warn, Error and Fatal data.
- **Debug**
  This level will designate fine-grained informational events that are most useful to debug. Logs everything (Info, Warn, Debug, Error and Fatal).

Debug is the default setting.

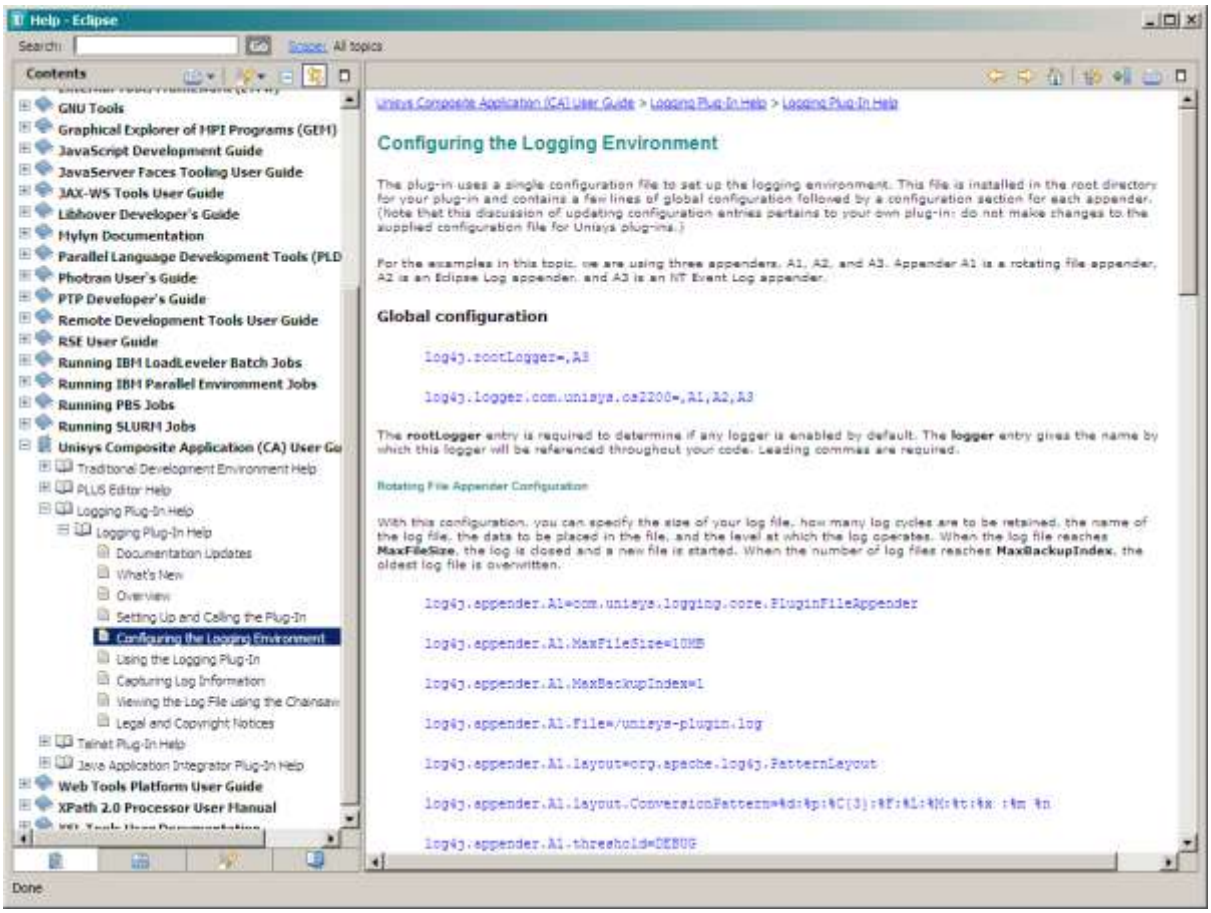
If the Eclipse OS 2200 IDE encounters a problem or issue, the user can change the log level accordingly to gather more information to assist in analyzing the issue. Unisys may recommend clients to set the log level in response to reported problems. At present the log file has to be deleted or the contents of the log file has to be emptied to create a fresh log file. Note: the mechanism to clear the log automatically or to create another log after exceeding certain size does not currently exist but is being considered as an enhancement for a future release.

Log Level Help

Click on the question mark to launch the help. Then click on the 'Configuring the Logging Environment'.



Eclipse will display the online help information.



Configuring OS 2200 connections

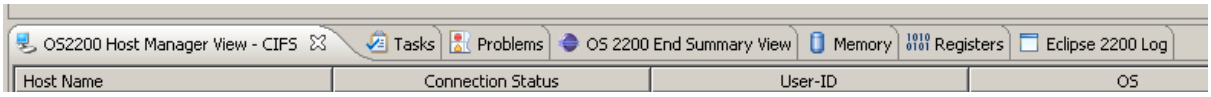
Eclipse uses two connection methods to the OS 2200 host:

1. Telnet is used to send commands to the host. You can open a demand session via telnet at any time.
2. CIFS is used to access the OS 2200 program file that contains the program source and other elements e.g. ECL. CIFS allows OS 2200 files to be exposed as network shares and then accessed from Windows using mapped drives.

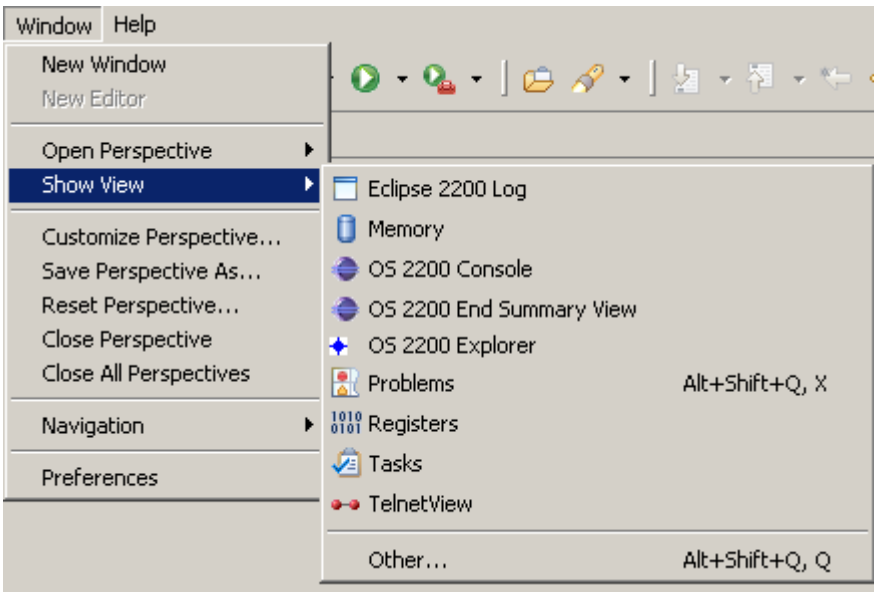
There are two methods to define OS 2200 connections as explained below.

OS2200 Host Manager View

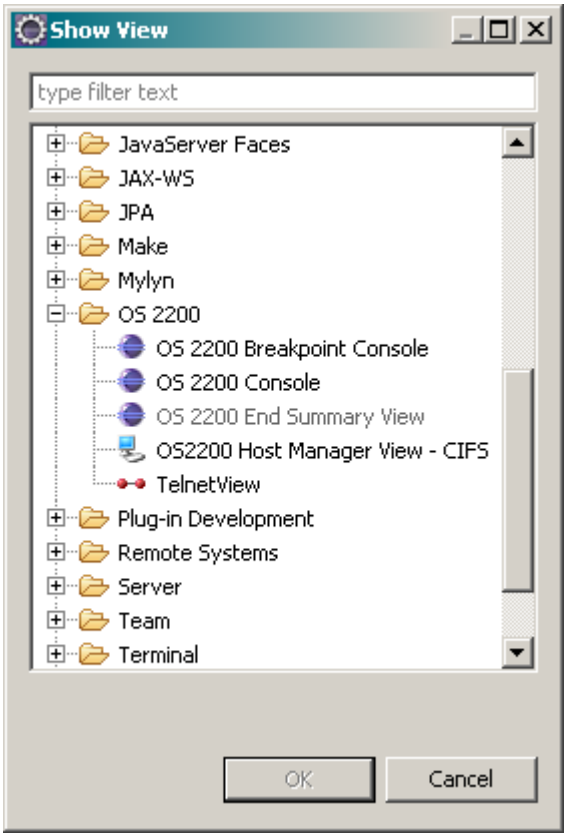
At the bottom of your Eclipse window, you should a number of tabs including the OS2200 Host Manager View:



If you do not see this tab, go the menu bar and select **Window → Show View.**

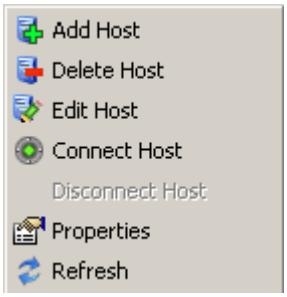


Click **Other** then scroll down to OS 2200 and expand the node.



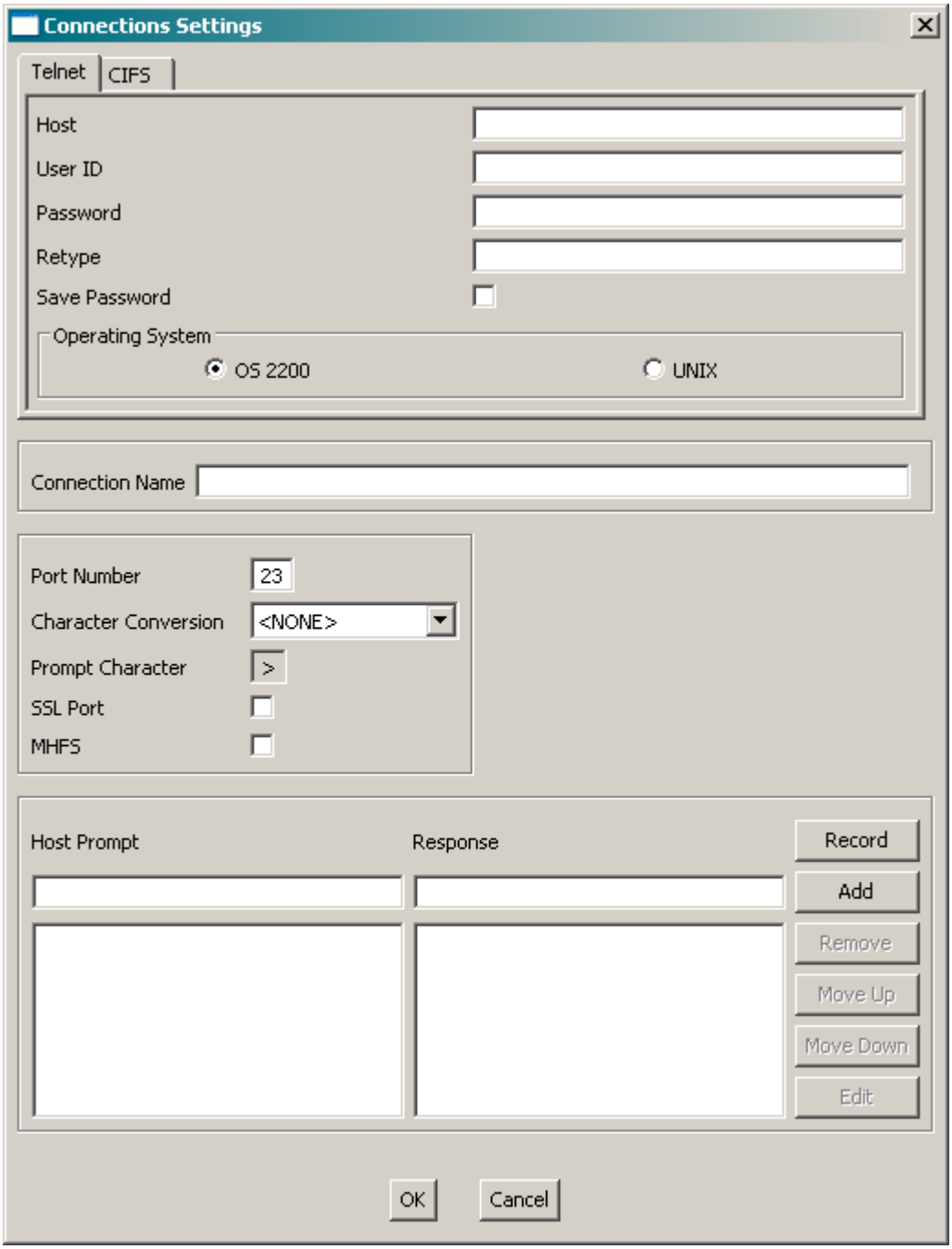
Now click the **OS2200 Host Manager View- CIFS** entry.

In the OS2200 Host Manager View tab, right click. Eclipse will present a dialog:



Click **Add Host**

The Connection setting dialog appears with 2 tabs – one for Telnet and one for CIFS. As you fill in the Telnet tab, the CIFS tab is also populated with the same details.



The Host field contains the DNS name or IP address for the OS 2200 host.

In the User ID field, enter your OS 2200 demand userid. Enter your demand password in the Password and Retype fields. Check the Save Password box. *Note: For Windows 7, your userid and password must be entered in upper case.*

Enter a Connection Name.

The port number must be 23 for Telnet so don't change this value.

If your OS 2200 supports a language requiring character conversion e.g. Japanese, use the list box to select the entry otherwise leave as <NONE>.

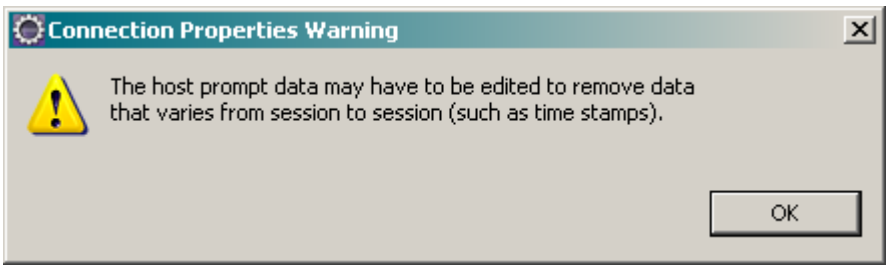
Check the SSL Port box if using secure Telnet.

Check the MHFS box if your system uses Multi-Host File Sharing. This implies that your system runs XTC or PAEXEC.

The Record button can be used to record your normal responses to OS 2200 generated prompts when you sign-on to a demand session. For example, "Enter your project-identifier" might be a prompt. After clicking the record button, Eclipse will open a demand session via Telnet. Perform your usual steps and responses when using demand session.

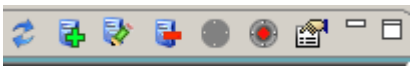
Check the CIFS tab information and then click OK.

The following warning window appears when you finish the recording.



To edit the OS 2200 Connection, you can highlight the connection in the OS200 Host Manager View and right click. Then select Edit Host.

At the right side of these tabs are some icons. When OS2200 Host Manager View is in focus, the following icons are displayed:

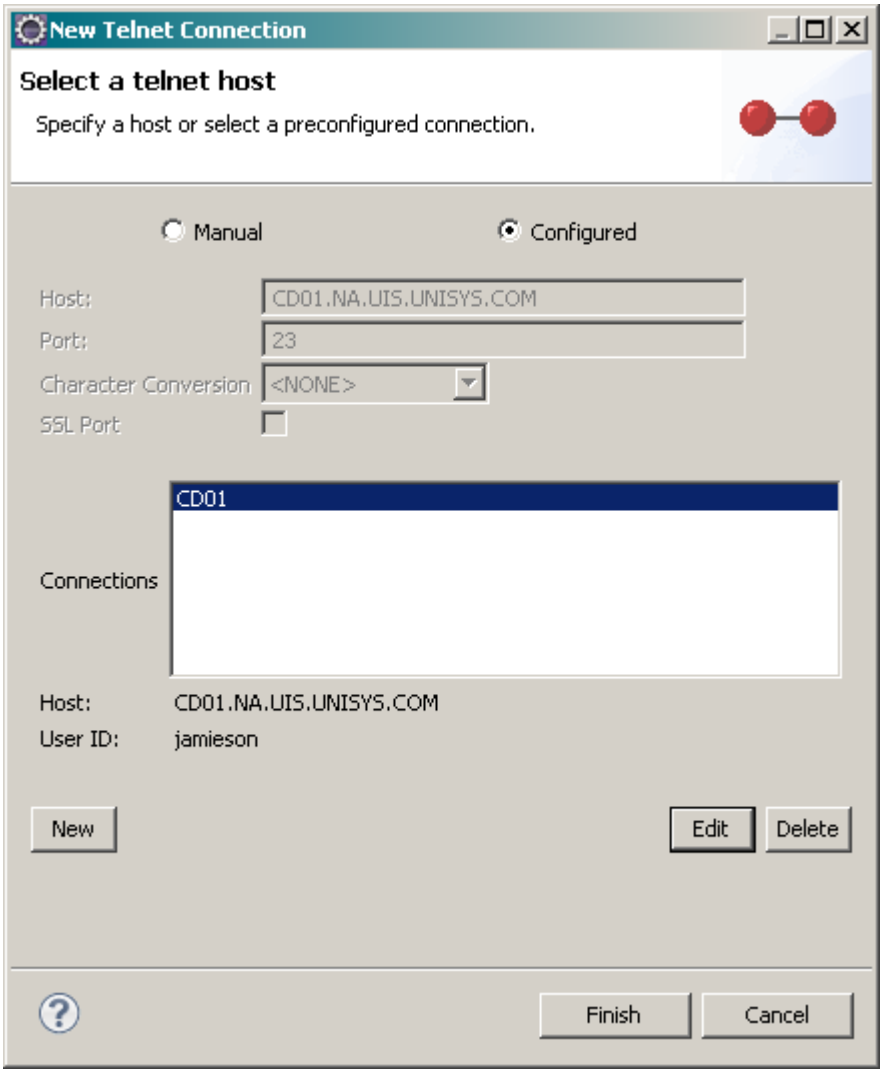


The second icon from the left can be used to add a new host. Hover text will display the function of the icon.

Or when using the Telnet Connection method below, highlight the Connection and then click Edit.

Using the Telnet Connection method

In the Icon menu, click the Telnet connection icon:



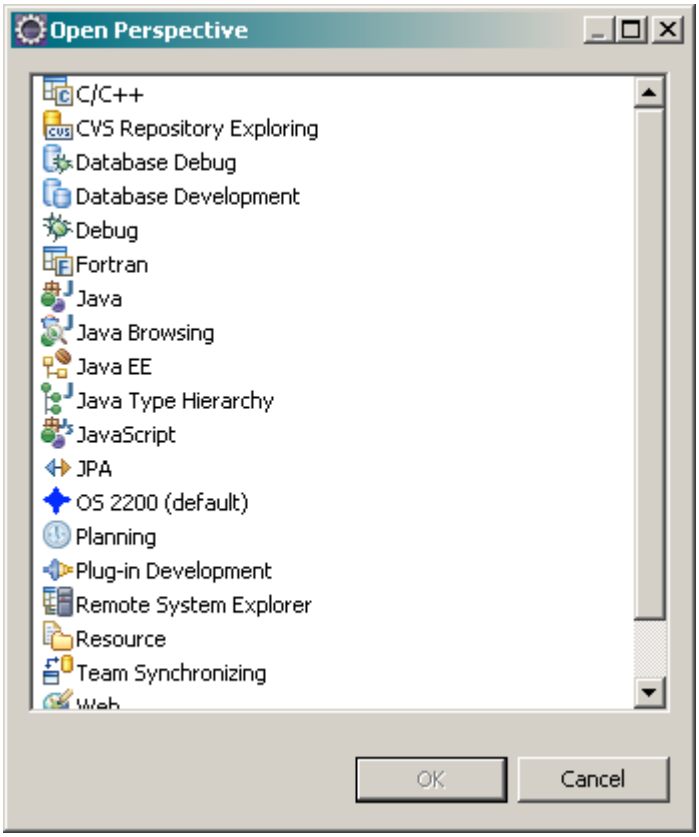
Click **New** to display the Connection Settings window as shown earlier. Follow the above steps.

Using the OS 2200 Perspective

Eclipse has different layouts for the IDE that are designed for the type of work being performed e.g. Java, Debug, and Java EE etc. After the Unisys Eclipse OS 2200 features have been installed (and at this stage they have), there is a perspective called OS 2200.

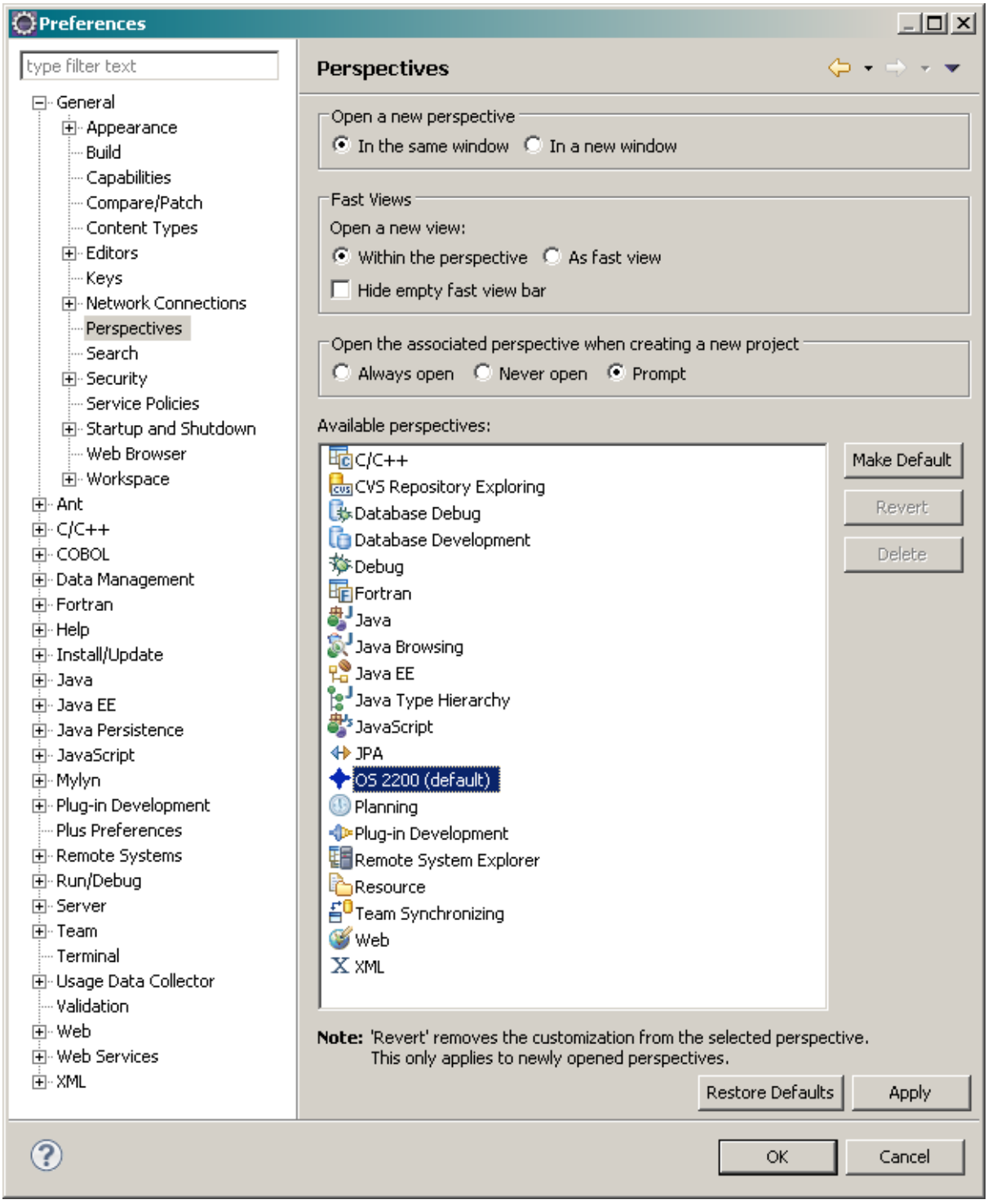
The perspective can be selected from the top right of the screen. In the following case, OS 2200 is greyed out as it is the default perspective that is open.

You can also go to the menu and click **Window → Open Perspective → Other**



Then select OS 2200

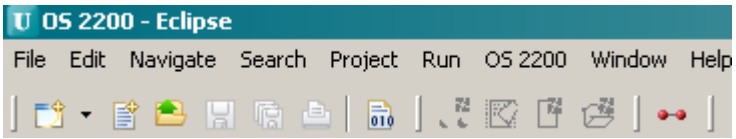
When the OS 2200 perspective is open, the IDE should look like:



Using the OS 2200 Telnet Connection

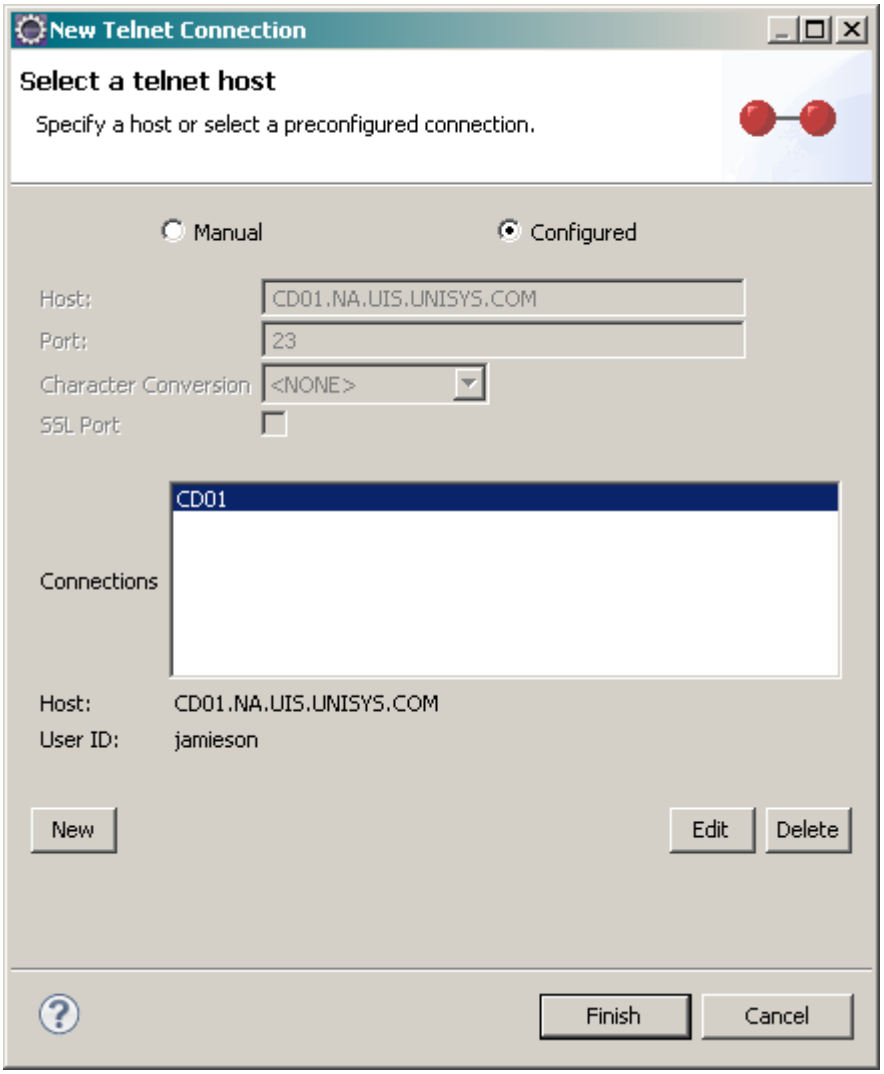
Now that Eclipse has been configured with OS 2200 connections, we can use the Telnet connection to run a demand session. Note that Telnet only supports a line-at-a-time session so full screen modes like IPF full screen mode, UDSMON etc will not work. However the Telnet connection can be used to submit ECL statements etc.

Look at the icon on the tool bar that looks like a set of dumbbells:

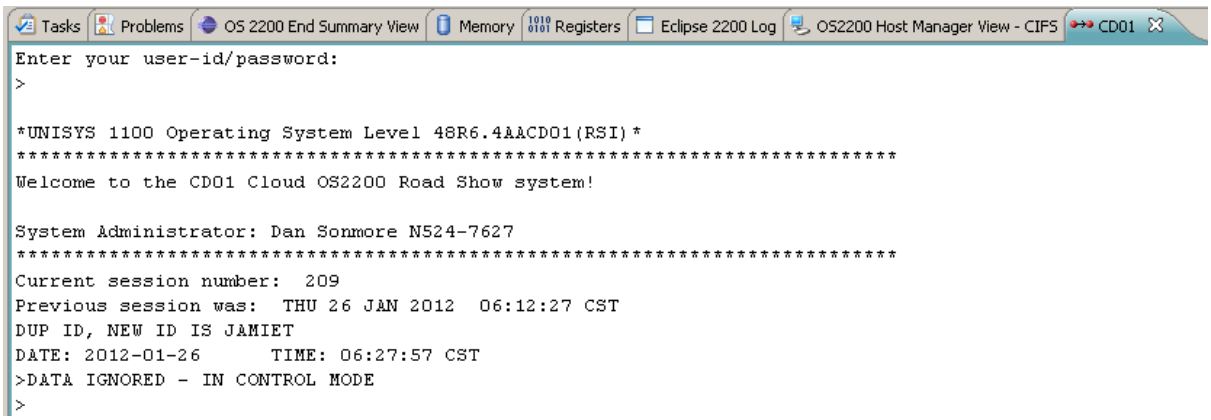


Click on this icon to launch the Telnet session.

Eclipse will present the following screen to show the configured connections plus allow you to enter a manual connection.



Select the connection defined earlier and click Finish. (Don't click New as that will launch the dialog to define a new connection.) Eclipse will open a new window pane for the connection and automatically start a demand session using the defined userid/password.



Commands are submitted as if you are using a demand session such as via a terminal emulator except that full screen mode is not supported. The session can be closed by clicking on the "X" in the tab.

If you want to save the contents of the Telnet pane, there is the following icon on the right side of the pane title bar:

Preparing Your OS 2200 Program File

Your OS 2200 program file that contains the program source code needs to be shared with CIFS so Eclipse can access the file. This is accomplished by creating a share using the CIFSUT processor in an OS 2200 demand session.

```
@CIFSUT
share /os2200/<qual>/<filename>          <sharename>
```

In the following example, the OS 2200 program file is dorado*rob. The commands are:

```
@CIFSUT
share /os2200/dorado/rob    doradorob
```

In this example, a Windows network drive could be mapped to \\<server>\doradorob.

To unshare the program file, enter the commands:

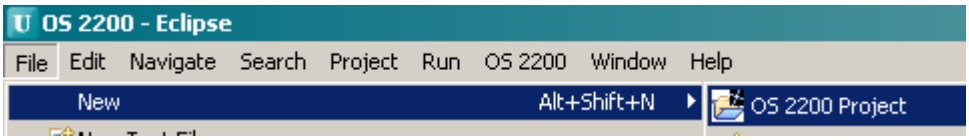
```
@CIFSUT
unshare doradorob
```

However it can be easier than this. A default share called “os2200” is recommended to be created over the OS 2200 MFD and we will use this share for the examples. This means that you don’t have to define your own shares.

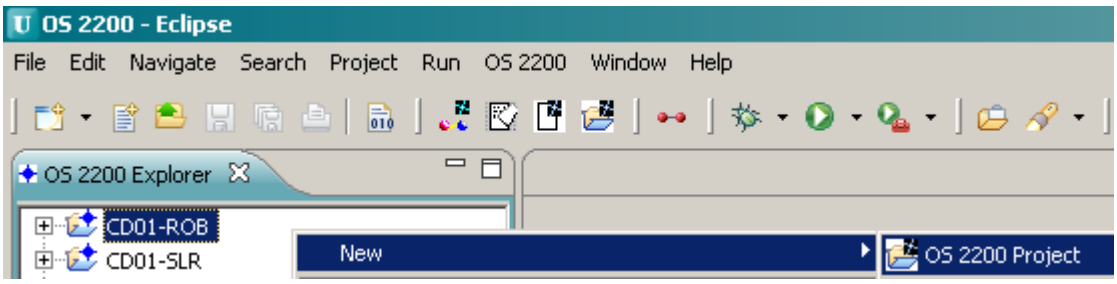
More details on CIFS can be found in the CIFS manual.

Creating an OS 2200 Project

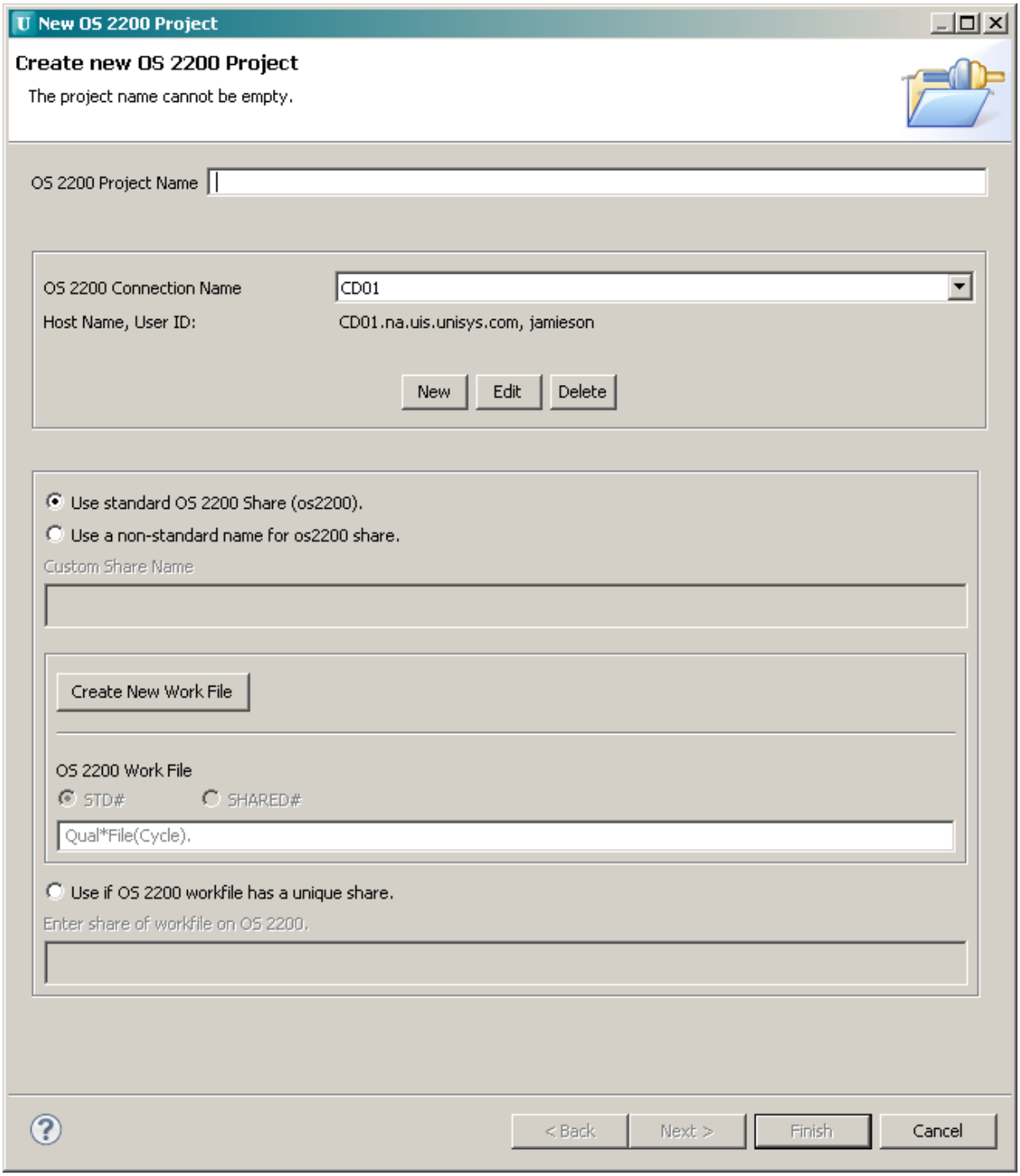
Now that a CIFS share has been created for the OS 2200 program file to be used for OS 2200 3GL development, Eclipse can be used to define the OS 2200 project. From the Eclipse workbench and in the OS 2200 perspective, go to the menu bar and do **File → New → OS 2200 Project** and click.



Alternate Option: Right click in the OS 2200 Explorer View and choose **New → OS 2200 Project**.



The following Window appears:



Enter a name for your project in the field “OS 2200 Project Name”.

Select the OS 2200 Connection to be used.

Check the radio button called “Use standard OS2200 Share (os2200)” if your site is using the os2200 share in CIFS. However if your site has used a different name for the share, then check the “Use a non-standard name for os2200 share” and enter the share name in the Custom Share Name field. Both of these options provide access to the entire OS 2200 MFD directory. However you may have a share defined over a single OS 2200 file. For example:

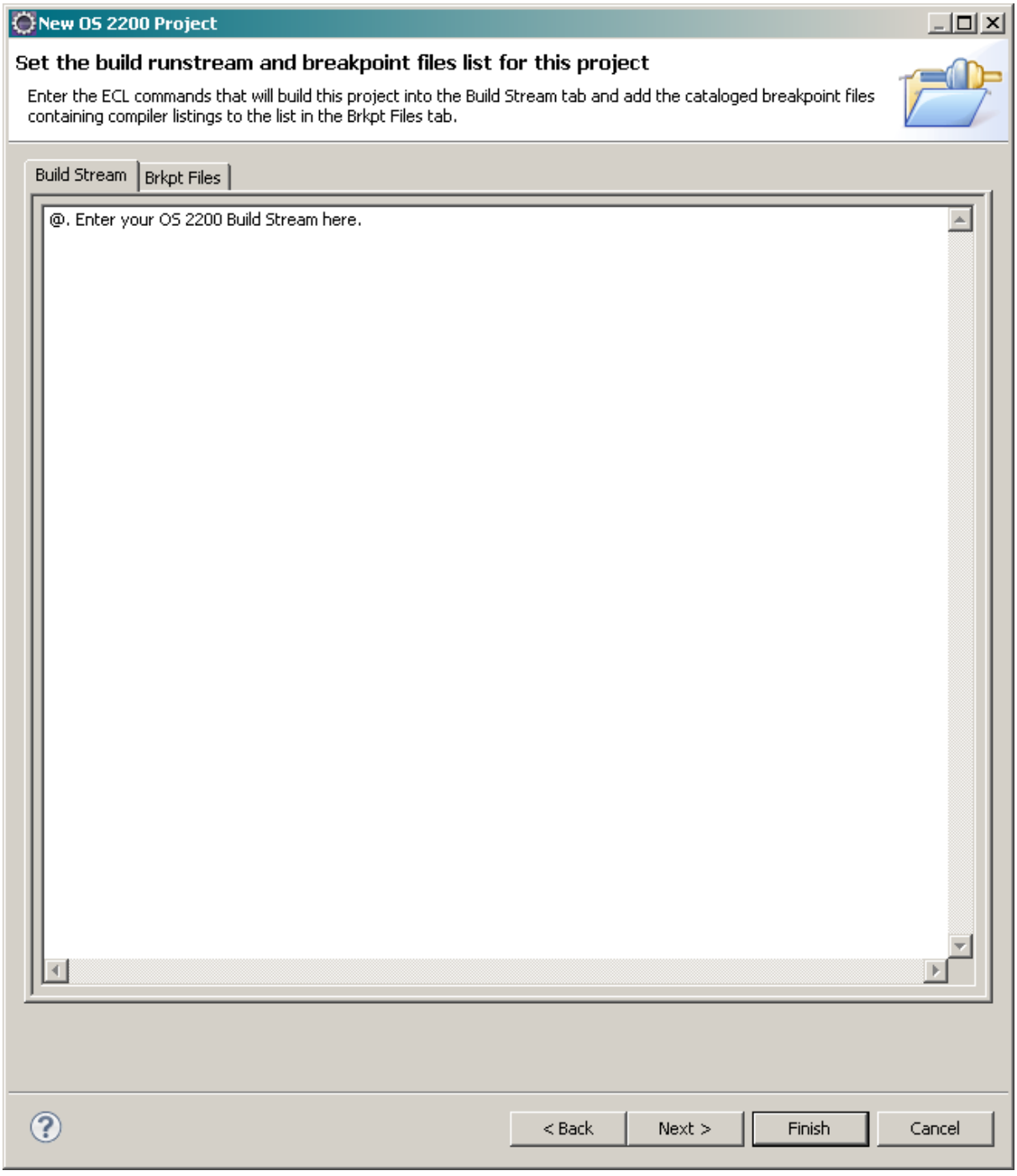
```
@CIFSUT
share /os2200/unisys/rob    robj
```

would create a share called ‘robj’ over file unisys*rob. In this case, check the ‘os2200 share cannot be used” radio button.

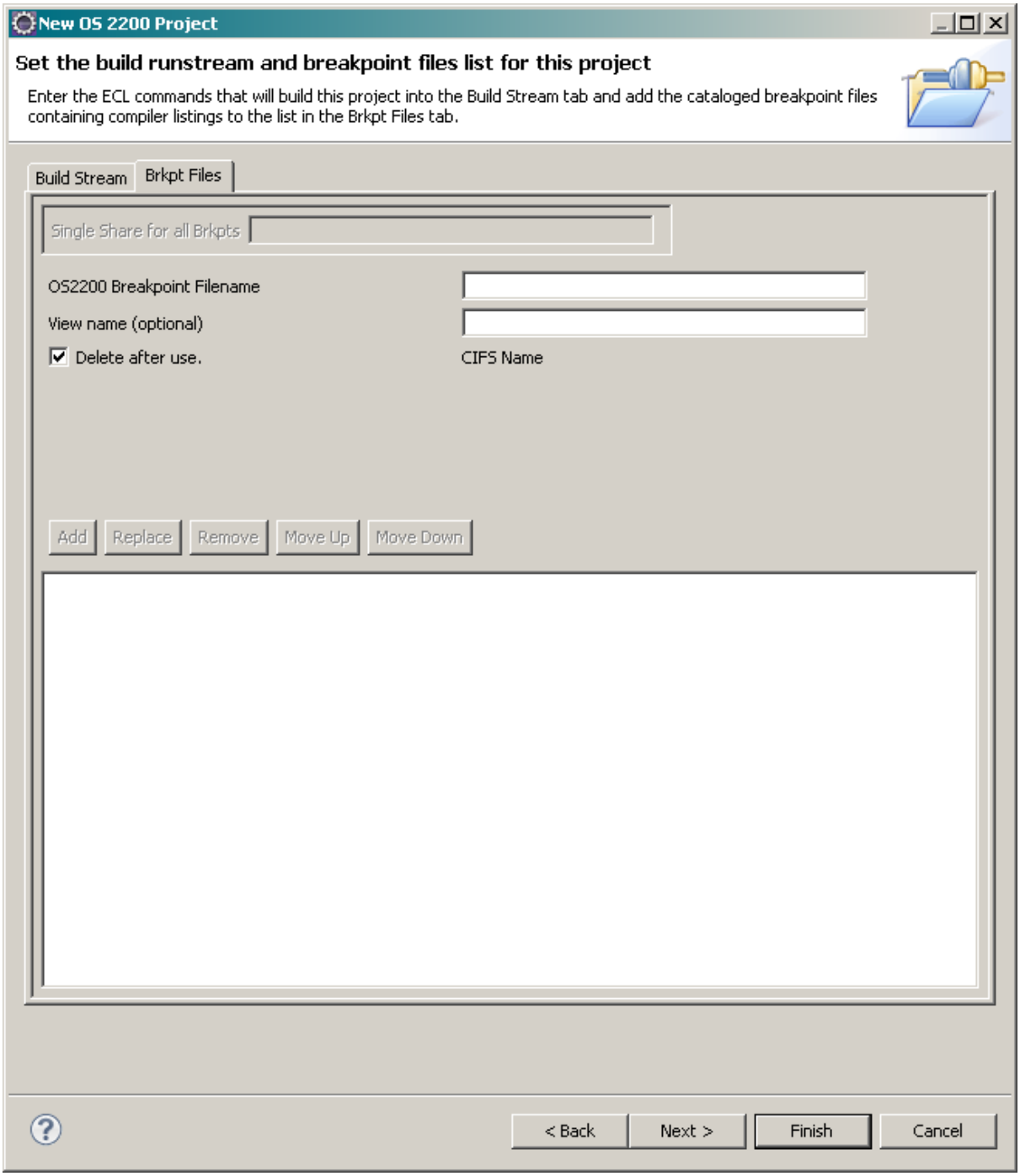
If using MHFS, refer to the MHFS Considerations section later in this document.

Now click **Next**.

The Eclipse wizard displays a screen for the build runstreams and breakpoint files to be used as shown below. If Eclipse can't access the OS 2200 Work File, an error will be presented. Check the Unisys-CA log (access via the menu option **OS2200→View Log** if additional information on the cause of the error is needed.

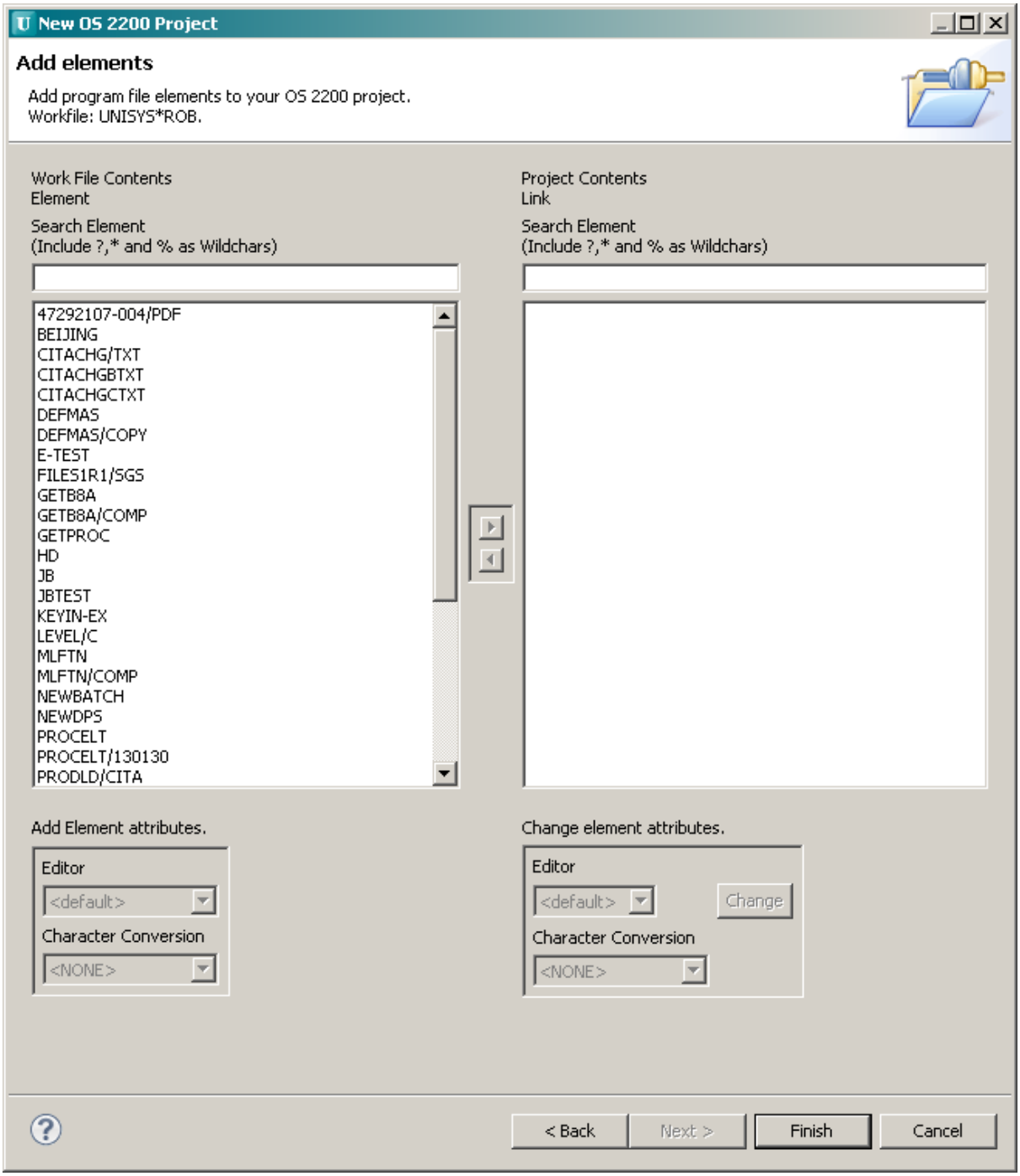


Click the "Brkpt Files" tab.



We will return to define these later.

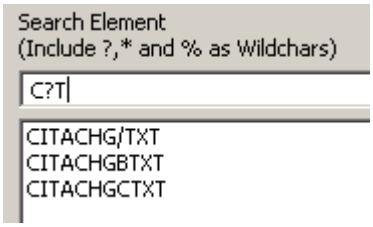
Click ‘Next’. It may take Eclipse a little time to access the OS 2200 program file and retrieve the names of all the source elements in the file. (ABS, REL, OM, ZM elements are not returned.) For this example using UNISYS*ROB, the following screen was returned.



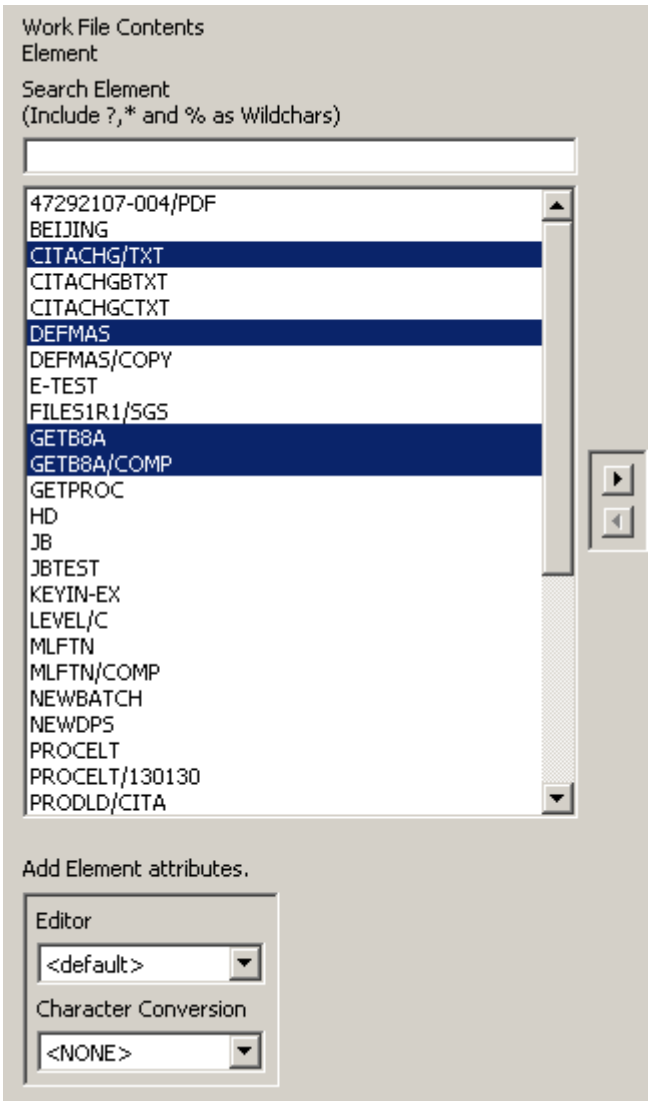
The left pane shows the OS 2200 source elements in OS 2200 format. (The only difference is for elements which were created by dragging a non-2200 file via CIFS to the work file. CIFS maintains the Posix format for these elements but the 2200 TOC will be in standard 2200 format.)

Now highlight the elements that you want to be part of this project. You can use standard Windows selection methods to choose multiple elements. Note that after you have selected at least 1 element, the ">" arrow is displayed and the drop-down boxes for 'Editor' & 'Character Conversion' are enabled. The user may change the type (extension) for the selected elements from 'Editor' drop-down.

Use the Search Element text box to filter the elements you want to display by entering the starting characters of the element name. Wildcard searches using '*' for any number of characters and '?' for a single character. For example, using '?':

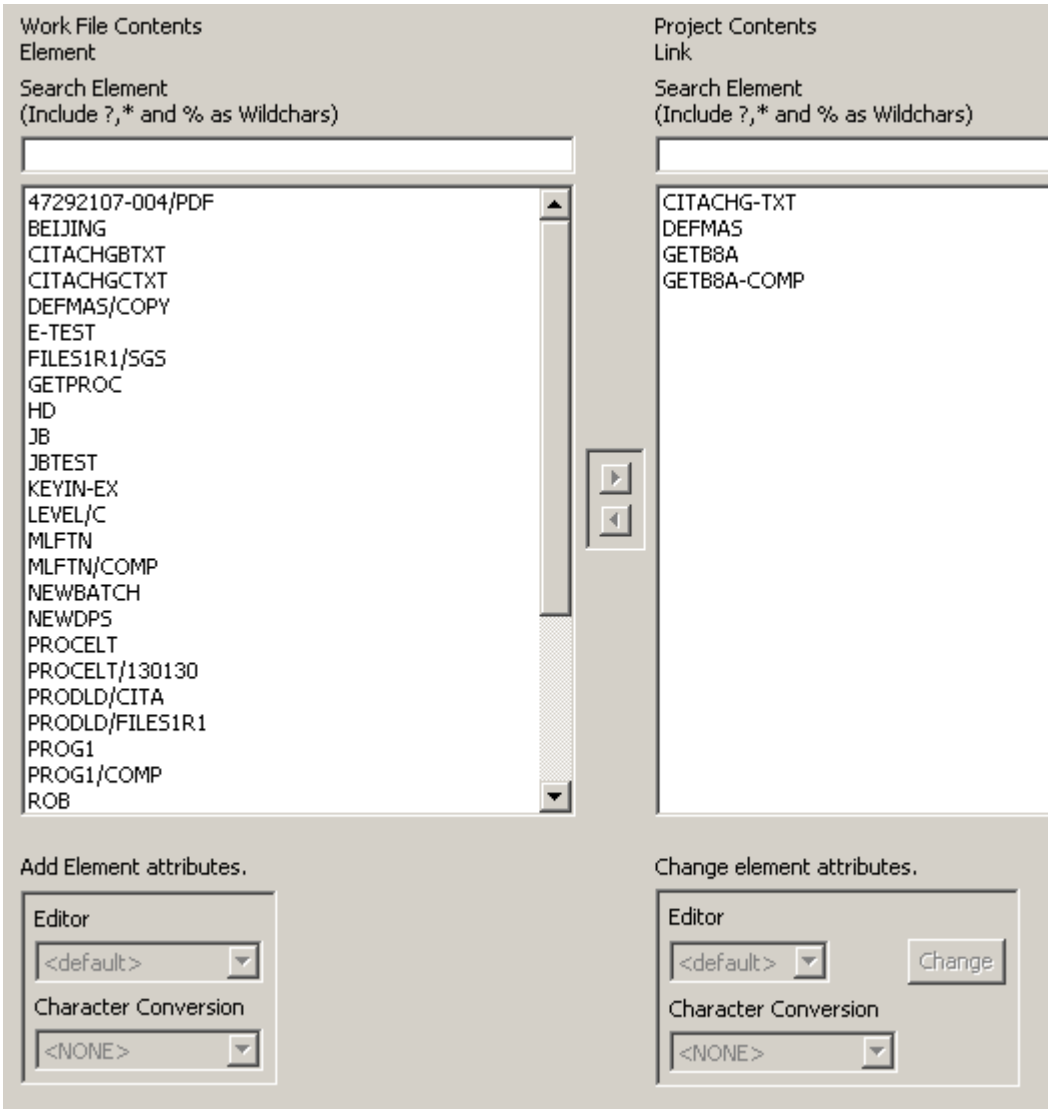


Note all elements are set to same type and/or character conversion.

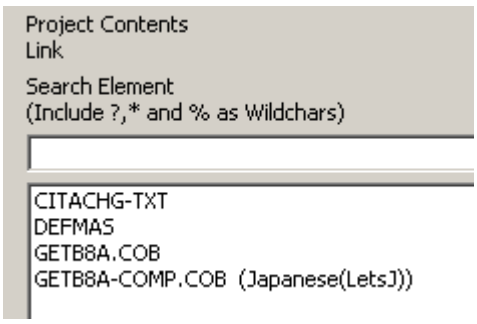


Click this arrow icon to add the selected files to your Eclipse project.

Note: Without selecting any elements, the user can create a project. Later, the user can either update the workfile or create a new element. This can be done by clicking ‘Finish’ from any of the screens in ‘New OS 2200 Project’ wizard, provided the project-details (Name, workfile & connection) have been furnished correctly on the first-screen.



The selected elements are moved from the left pane to the right pane. At this stage also, we can define the type of element. For example, for any COBOL source programs, highlight the element and select 'COB' from the Editor list box and then click 'Change'. Change any elements containing ECL etc to ELT editor type. Changing the type ensures that the element is opened in the desired editor, i.e. a COB element would open in the COBOL-editor and an ELT element would open in the general Text-editor.

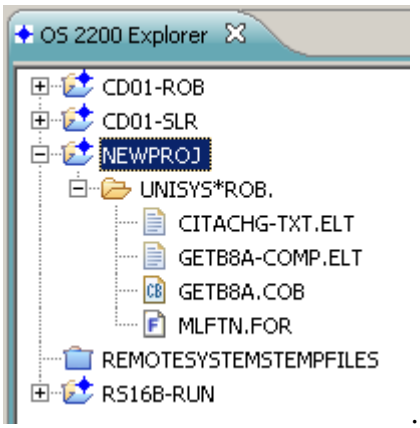


Note the Character Conversion should be set correctly if your OS 2200 supports a different language like Japanese.

Note: Even before moving elements from right pane to left pane, a user can change the element type. The user has to select file(s) and choose what type of element he wants from the drop down under "Add Element attributes" and then move to left pane

Click 'Finish'.

Eclipse opens your project in the OS 2200 Explorer View

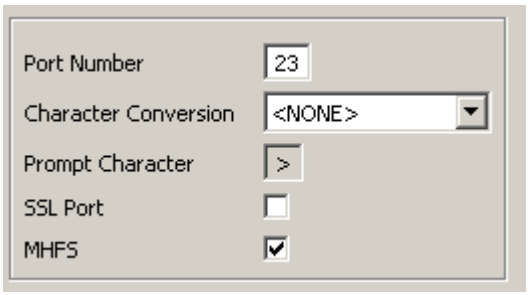


Automatic Open of a Single File

If a single source element is added to a project, Eclipse will automatically open the element with the appropriate editor.

MHFS Considerations

When a new OS 2200 Host is defined, there is a check box to indicate if the host uses MHFS:

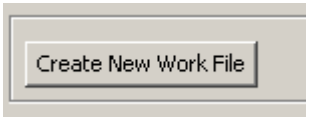


Eclipse does verify if the system is configured for MHFS. The incorrect setting for this option could lead to unexpected errors during other operations.

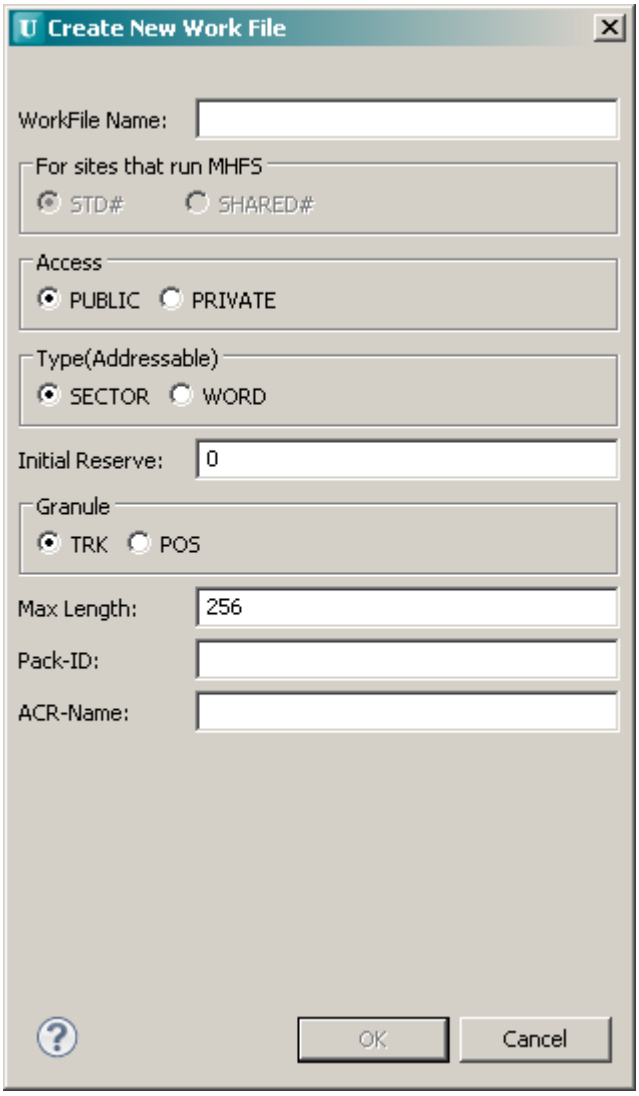
Select STD# for files stored on local discs or SHARED# for files on shared storage. Note that the same file name could exist in both STD# and SHARED#. Eclipse uses this setting to access to desired file.

Creating a new OS 2200 Work File.

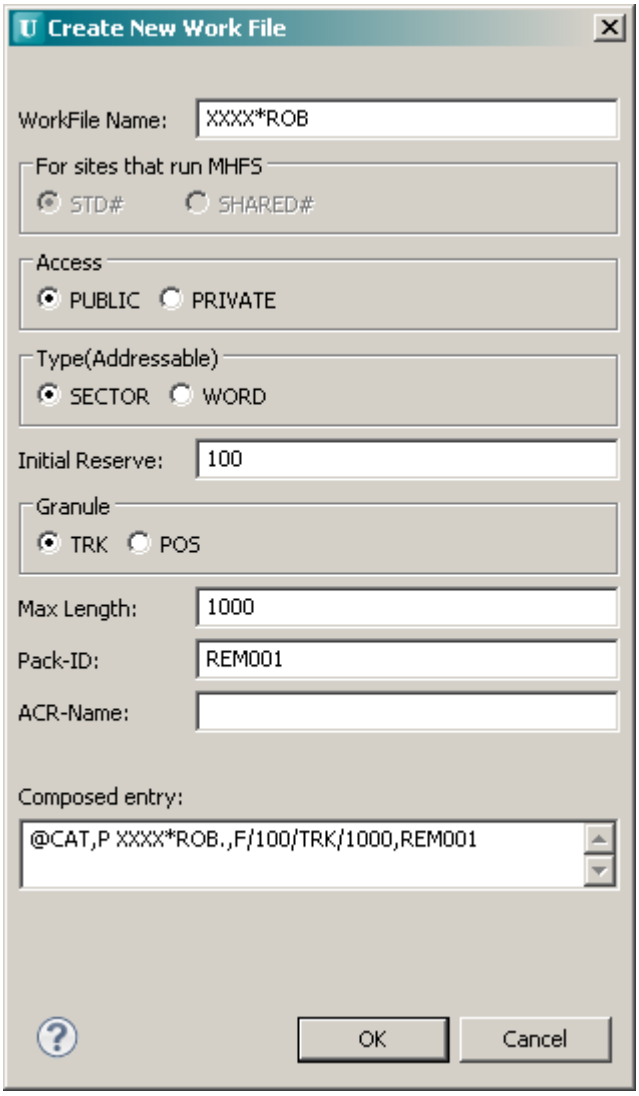
When running the Create New OS 2200 Project wizard, an existing OS 2200 work file can be entered. However Eclipse has the option to create the OS 2200 work file. After entering the project name and connection, click on the Create New Work File button.



The next dialog allows you to enter the OS 2200 Work File name:

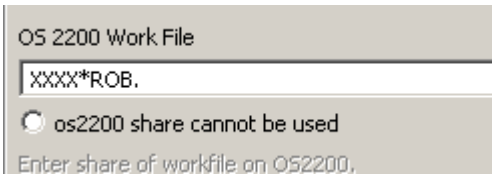


Enter the parameters as desired:



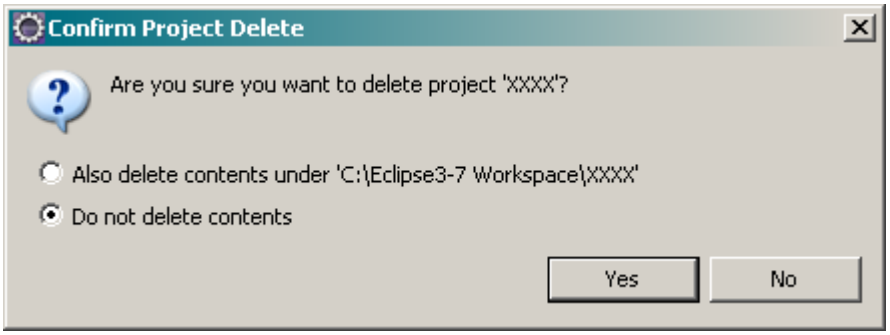
The 'Composed entry:' field can be further edited. Click OK.

After cataloguing the file on the 2200, Eclipse fills in the Work File name and then you proceed with the wizard as explained above.



Deleting an OS 2200 Project

Right click on the project and select Delete. Eclipse displays the following dialog.

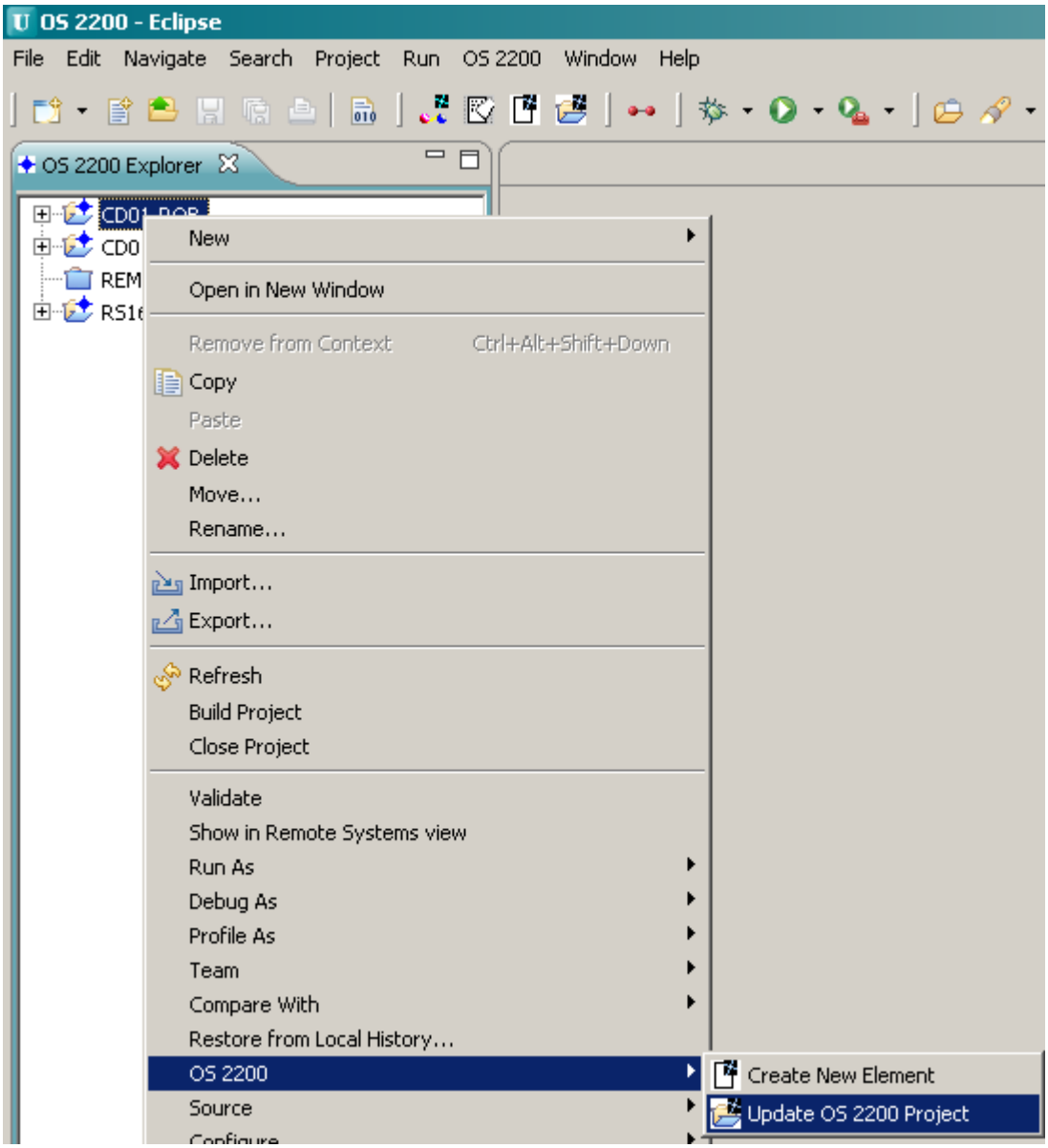


This operation does not delete the OS 2200 work file.

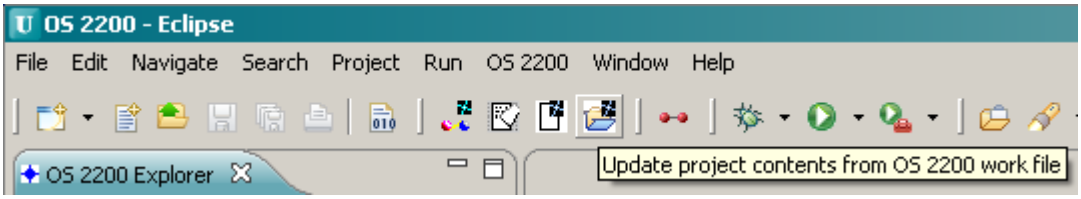
Maintaining your OS 2200 Project

The wizard is used to define the initial project properties but these can be maintained later. In a later section there is an explanation of how to maintain the Build and Brkpt properties.

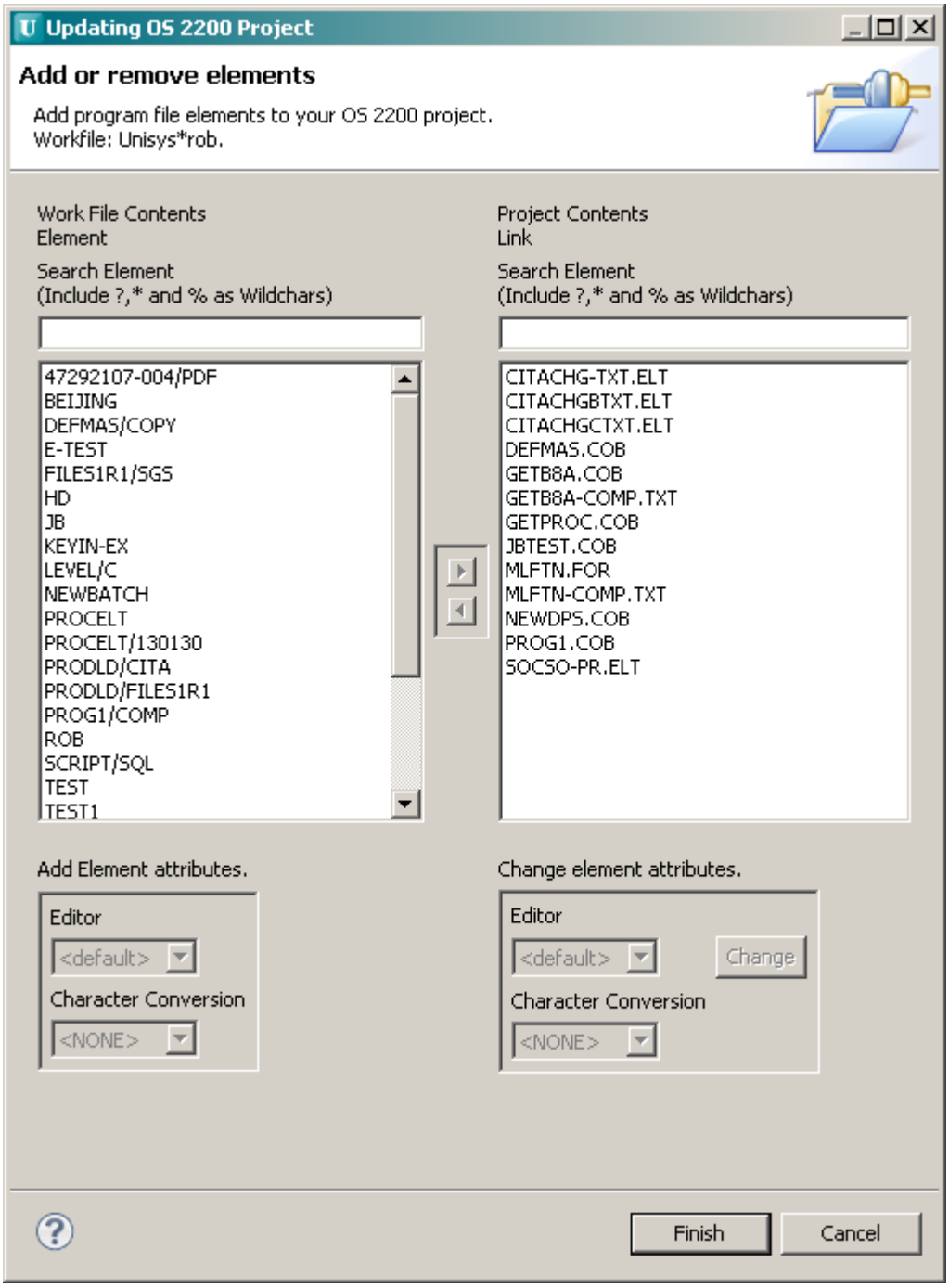
To maintain the existing project properties, right-click on the project name and then select OS 2200 and then Update OS 2200 Project as shown below.



Or click the  icon:

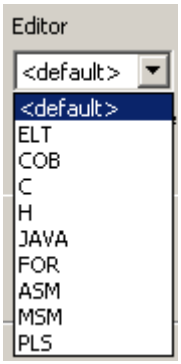


This launches a wizard that is similar to the new project wizard.



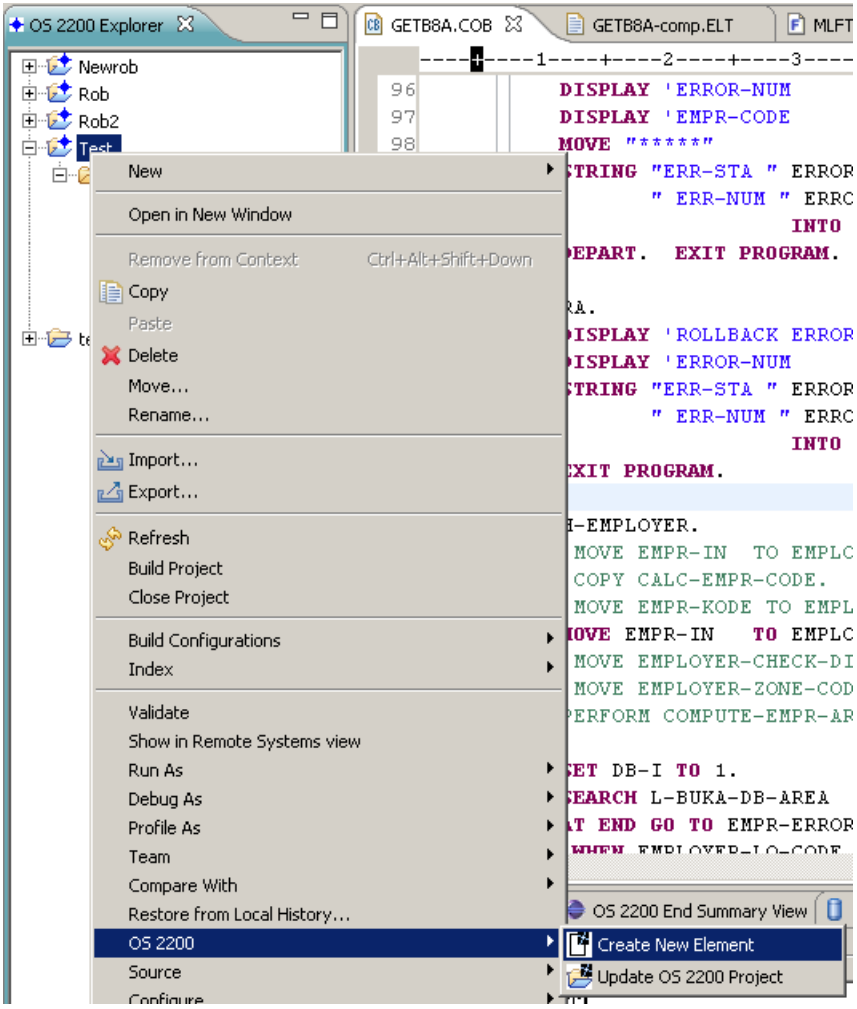
Note the Search text boxes can be used to filter the contents of either pane. Use '*' and '?' as wildcard characters.

Note to change the editor for a file, you must highlight the file, select the required editor and click Change. The available editors are:

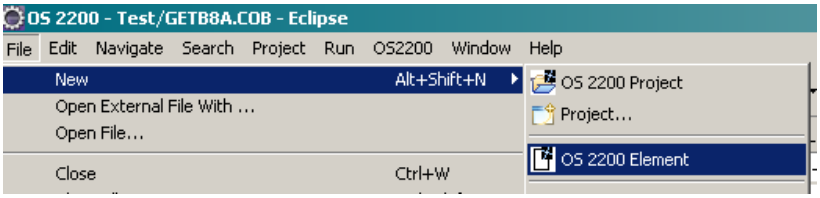


Adding a new OS 2200 Element to a Project

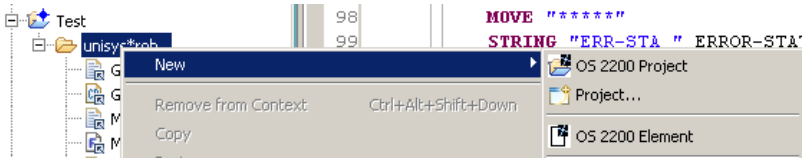
You can add a new element to your OS 2200 program file by creating an element in your Eclipse OS 2200 project. For example, you may want to write a new COBOL program. In the Explorer tree, right-click on the project name and then select OS 2200 → New OS 2200 Element.



An alternate option is to use the menu **File → New → OS 2200 Element.**

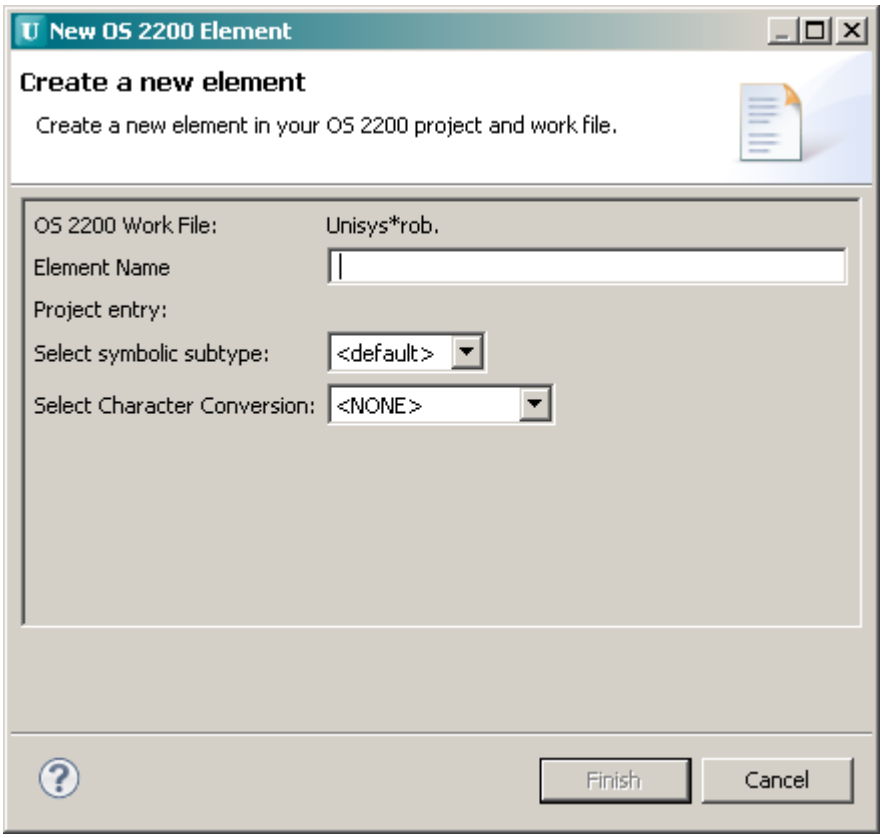


Yet another option is to right click the OS 2200 Work File in the Explorer View and select **New → OS 2200 Element.**

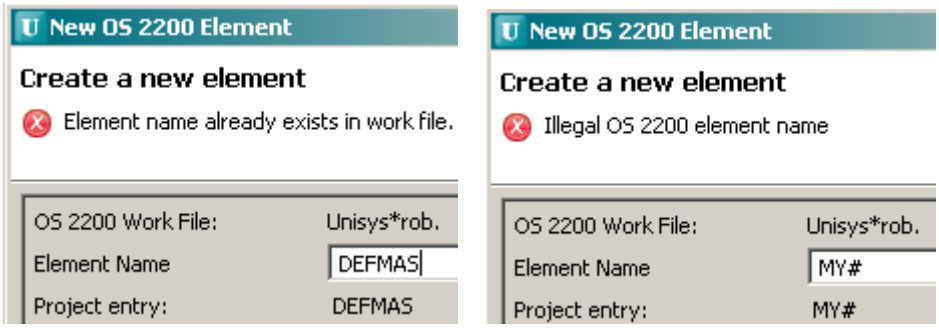


Yet another method is to click the  icon in the Icon menu bar.

Eclipse will respond with the following wizard:

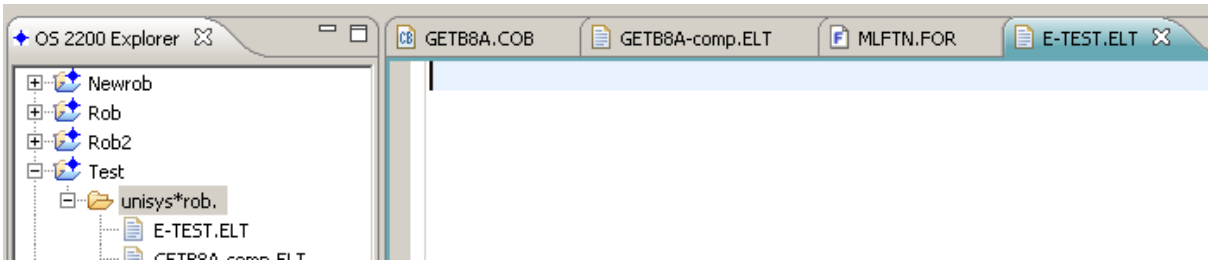


Enter your OS 2200 element name and select the symbolic subtype (choose COB for COBOL). If Character Conversion is required, please select the appropriate language otherwise leave as <NONE>. Note that the element name must comply with standard OS 2200 file naming conventions for syntax and characters etc. Using element/version is valid. Note the status message displayed in the window.



After entering a valid element name, click **Finish**.

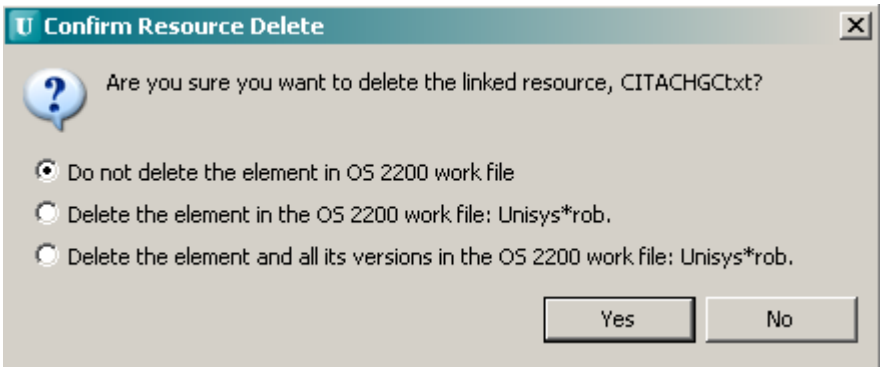
A new editor pane will be opened to allow the data entry of the source. Note the editor is based on the symbolic subtype that was used when creating the element.



Note that the navigation tree has been updated as well.

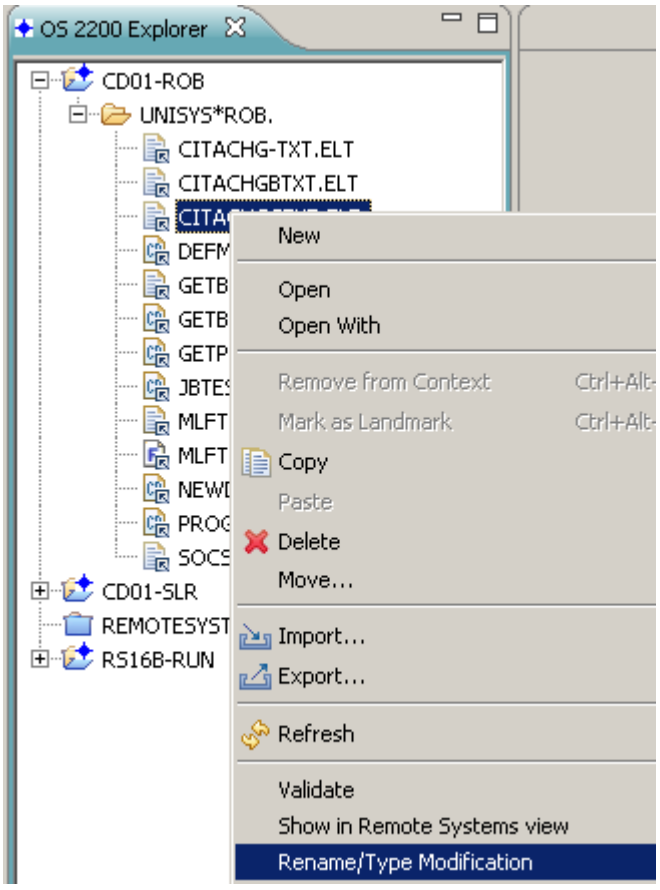
Deleting OS 2200 Elements from your Project

Right click on the file in the project and select **Delete**. Eclipse displays a dialog to control the actions to be performed. Select the appropriate option and click Yes.

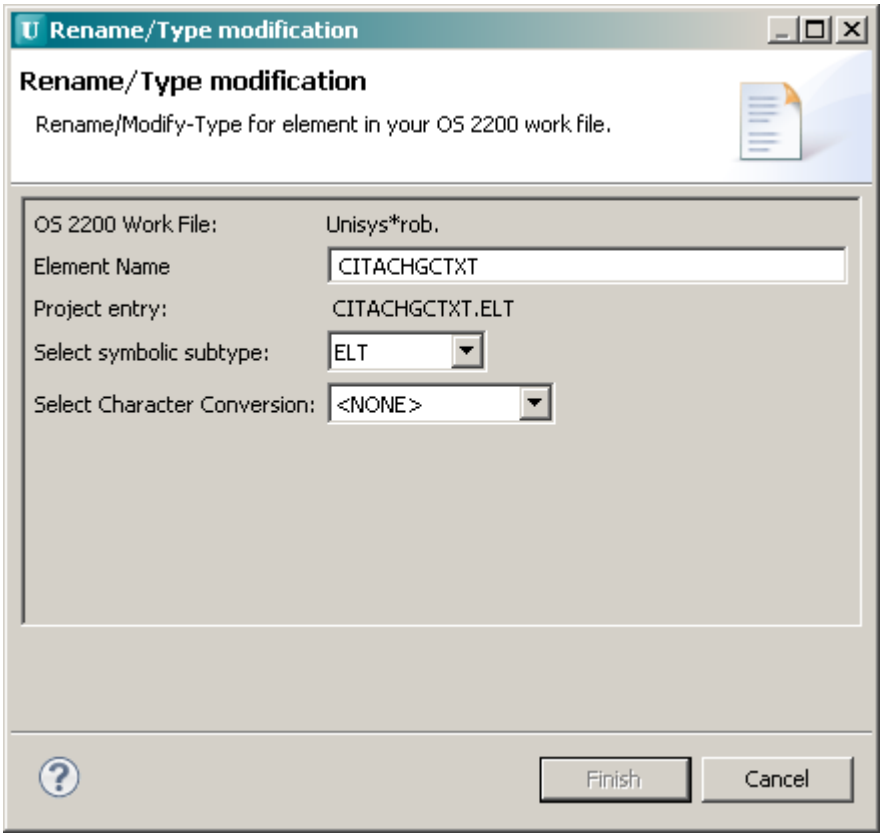


Renaming or Changing the Type of OS 2200 Elements

Right click on the file in the project and select **Rename/Type Modification**.

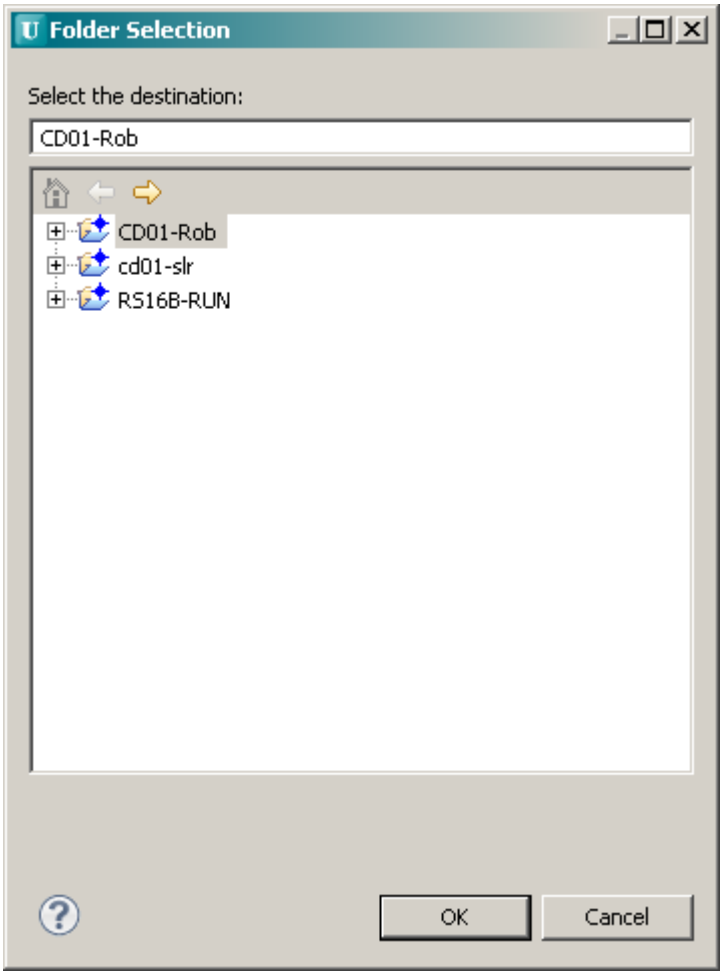


Eclipse displays a dialog where you enter the new element name, symbolic subtype and character conversion language. Click Finish to action.

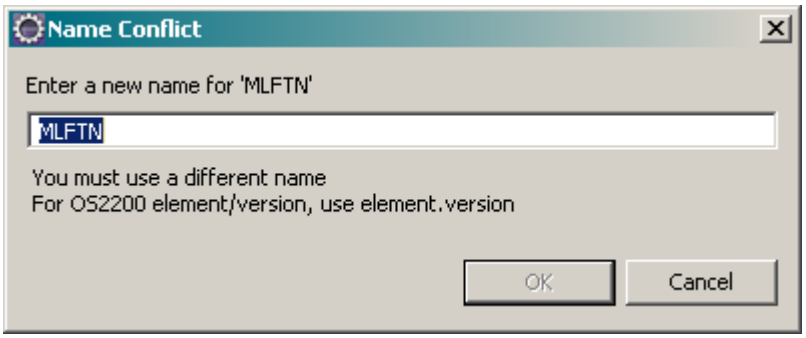


Moving OS 2200 Elements

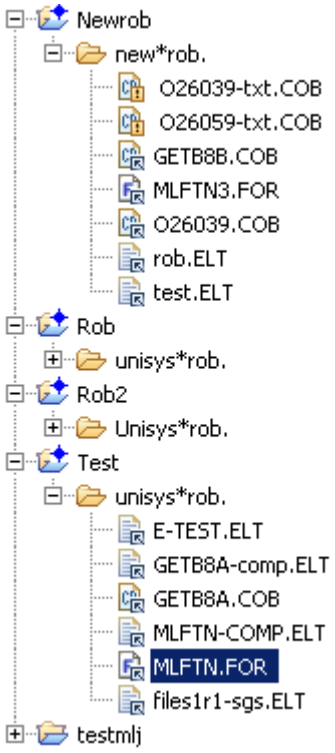
A file in one project can be moved to another project. Right click on the file in the project and select **Move**.



Eclipse will try to move the element to the new project. If the element already exists in the destination OS 2200 work file, the following dialog is displayed.



After the move, the destination project is updated with the new element. (Note I used MLFTN3 to overcome the name conflict.)

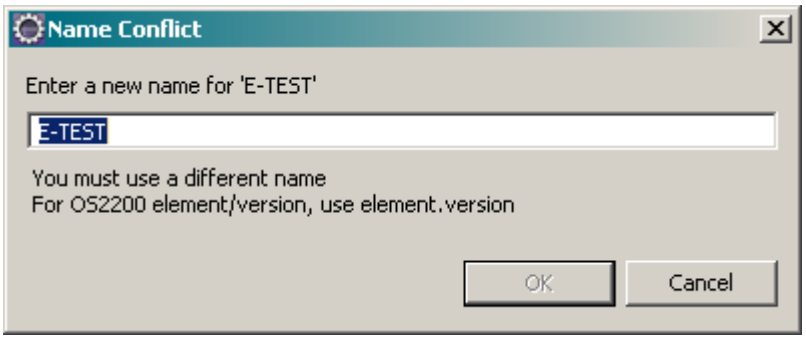


Copying OS 2200 Elements

Right click on a project file and select Copy.

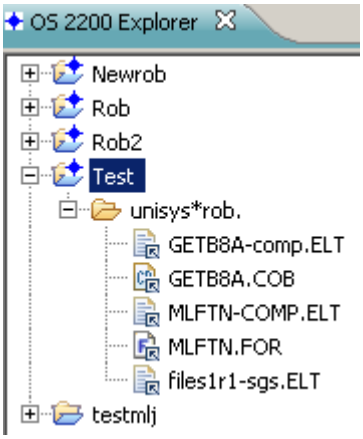
Highlight the destination project, right click and select Paste.

Eclipse will copy the file to the new project. If the element already exists in the OS 2200 work file, the following dialog appears.



Using the Explorer View

The Explorer View lists the OS 2200 Projects defined in this workspace. If you expand the node, the next level shows the OS 2200 work file name. If you expand this node, you will see the source elements that comprise the files in your project.

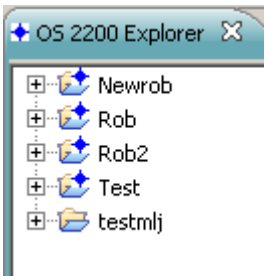


The project type is indicated by the icon next to the project name. Note the difference between testmlj and the other projects above – all are OS 2200 projects except for testmlj.

Note if the source element has a version, the version is appended to the element name after a hyphen. For example, element MLFTN/COMP becomes file MLFTN-COMP.ELT. The extension on the file is important as it determine what Eclipse editor is used. COB will launch the COBOL editor, FOR will launch the Photran editor (Eclipse Fortran editor). ELT will launch the generic editor. The editor to be used is displayed in the icon next to each file. Note that Eclipse will try to determine the editor to use if the extension is TXT.

Supporting Multiple Projects

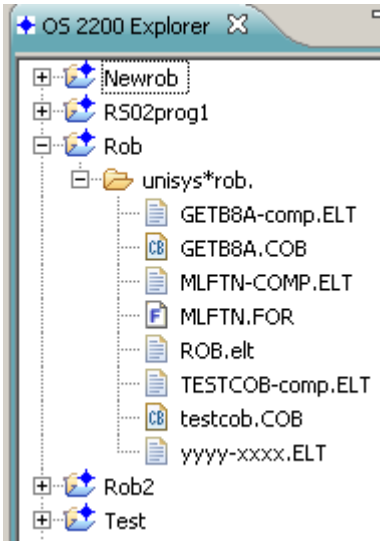
Follow the procedures above to add more projects. The projects are added to the navigation tree and sorted in project name order.



Note that the same OS 2200 Work file can be used in many different projects.

Using the OS 2200 Project

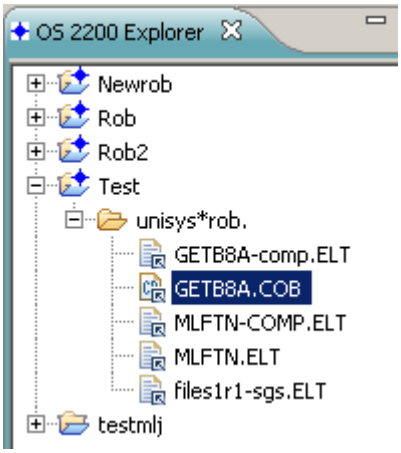
This section will cover how to edit and build (compile) your project using Eclipse. The following screen shows the OS 2200 Project perspective. This section focuses on using the COBOL editor. Later sections discuss other editors like the Fortran editor and Generic editor.



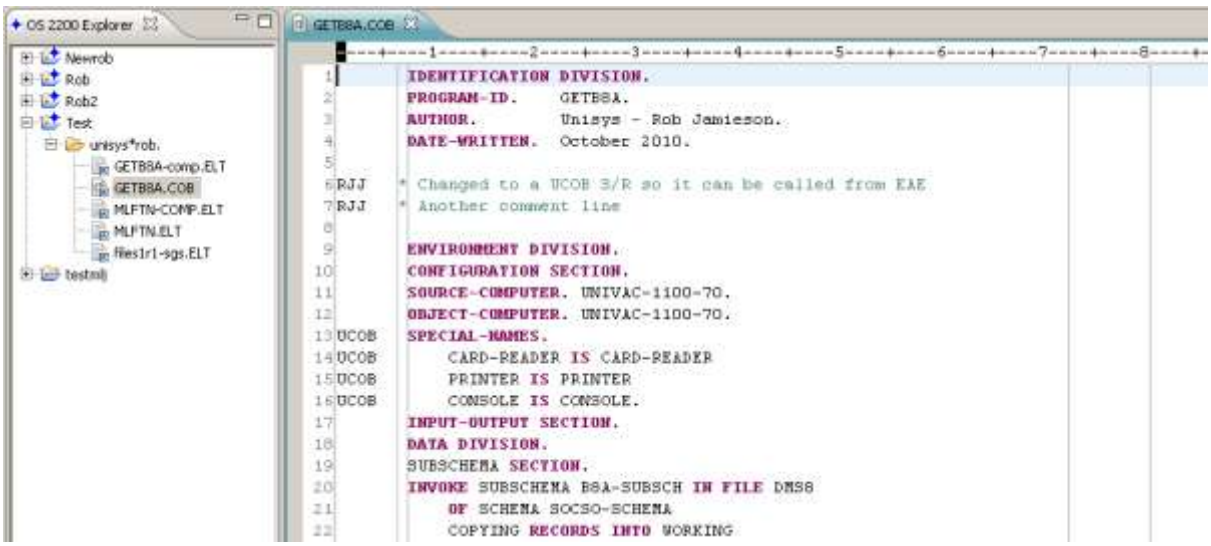
Note that the project name is the root of the explorer tree and the OS 2200 program file is shown but only those selected elements as project files are listed.

Editing and Saving

Now double click on an element that represents a COBOL program. In the project navigation tree, the file will have an extension of COB. In this example, GETB8A.COB is selected.

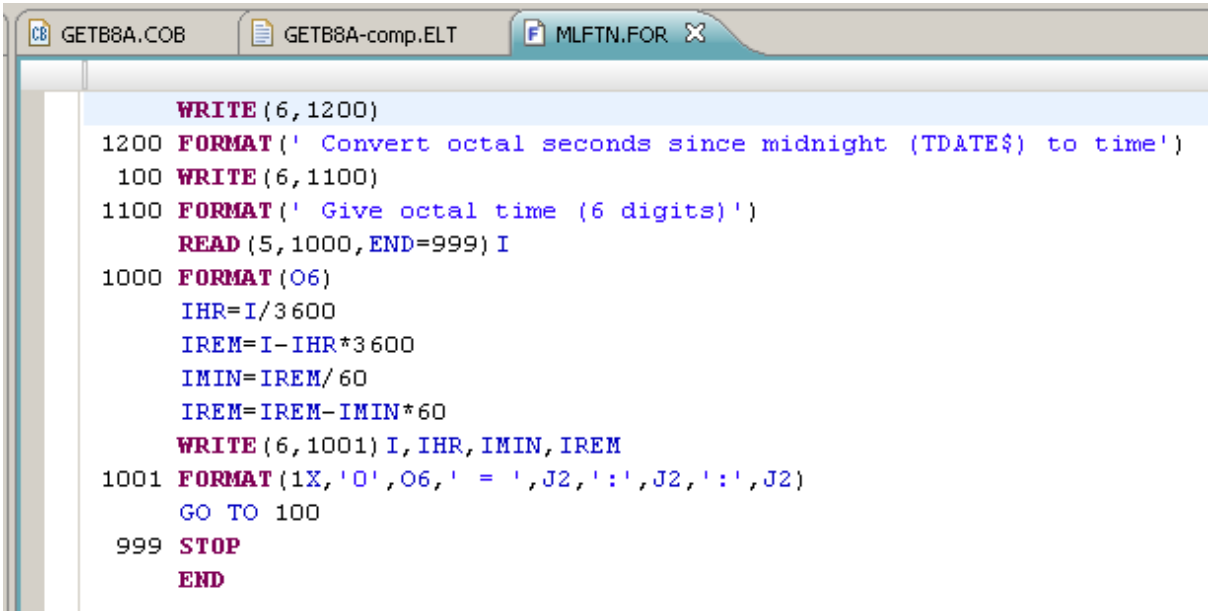


Eclipse will retrieve the source from the OS 2200 work file and open using the COBOL editor.



Eclipse has now displayed the contents of the element in a separate tab in the editor window. Since we had identified this element as a COBOL program, the editor uses the UCS COBOL template to highlight the code in different colours for reserved words, statements, comments and user variables. These were the COBOL preferences that we configured earlier.

One of the advantages of using Eclipse is that multiple files (OS 2200 elements) can be open at any time. These could be from one project or multiple projects. You can double click a file in the navigation tree to open or right click the file name and then select **Open**. Eclipse will open a new pane for each file as it is opened as shown below.



Therefore you could have multiple files representing different COBOL programs open at the same time. It is a simple matter to click on the appropriate tab to change the focus to the required source for that file. Code can be copy-pasted between panes – even if the files are in different projects.

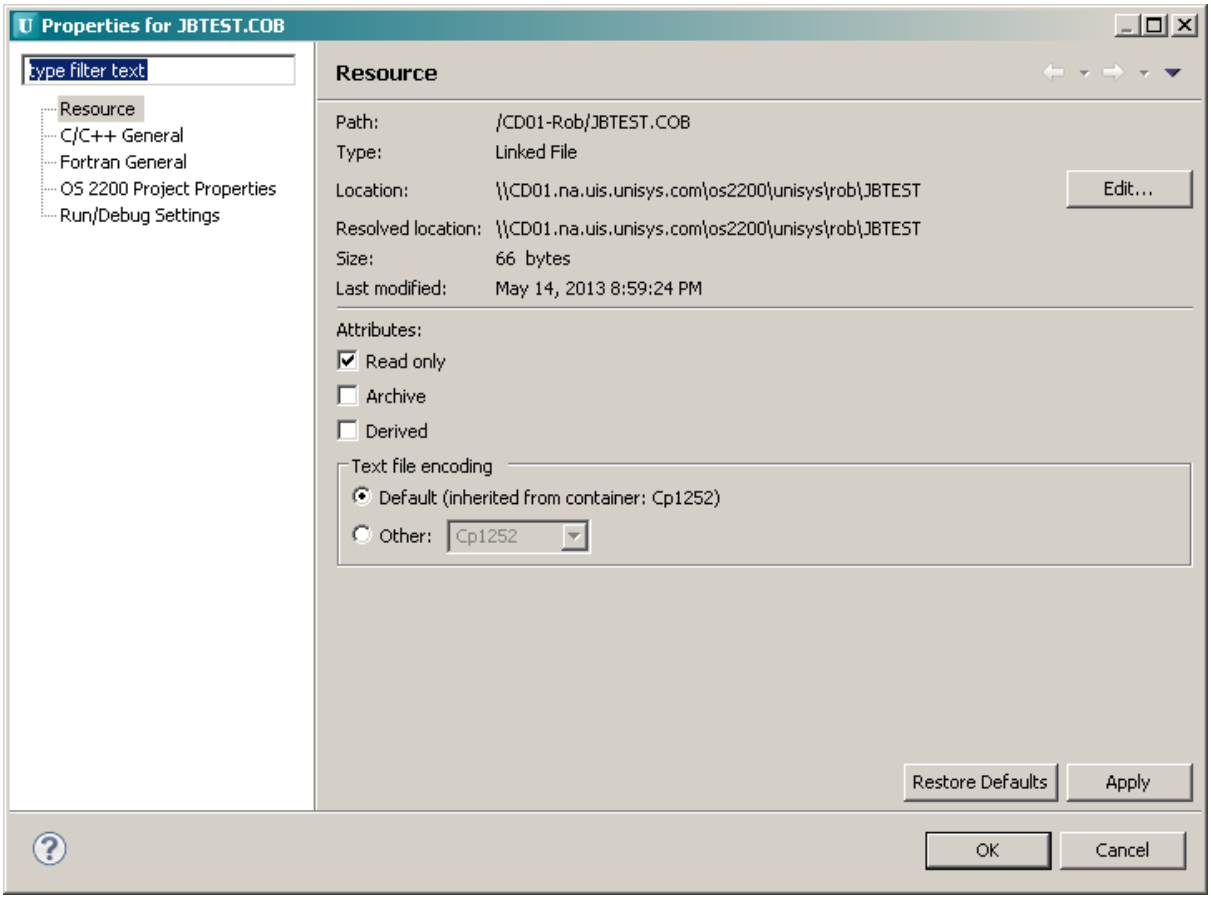
Note that after you edit the source code, an '*' appears before the element name in the tab. See below for an example.

These edits have only been made to the version of the file (element) stored on the workstation and not on the OS 2200 host. When you save the element, the updates are applied to the element in the OS 2200 program file. Eclipse will also maintain a local history of saved versions that will be discussed later.

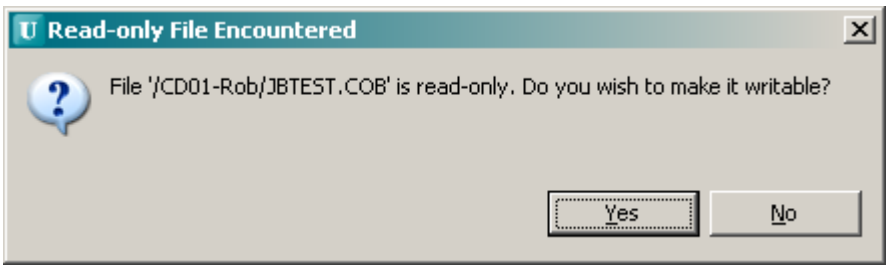
Note that Eclipse does no syntax checking or compiling. The editor template provides assistance to improve your productivity. We will discuss some of the editor features later. Incorrect syntax will be reported after the program is complied on the OS 2200 host when you build the project.

Opening an Element in Read-Only Mode

By default, Eclipse will open elements in update mode. If you want to open an element in read-only mode, right click on the element in the Explorer View and select Properties. (You can also use File -> Properties.) Check the 'Read only' attribute.



If the user tries to update the element, the following prompt is received.



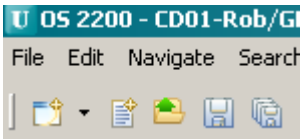
The user can decide whether they want to proceed and change the element so it can be updated.

Using the Save option

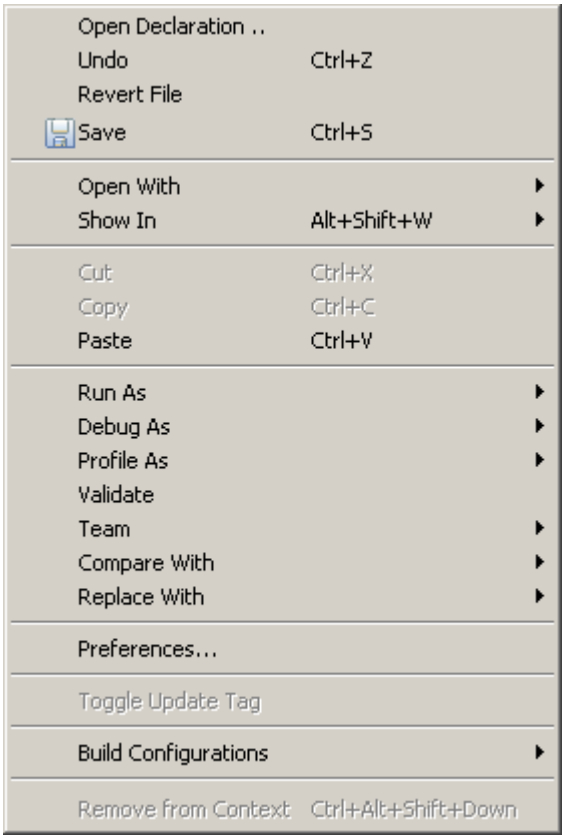
To save an element, you can

a) click the diskette icon in the tool bar,
b) right-click in the editor pane and select ‘Save’ or
c) use File → Save.

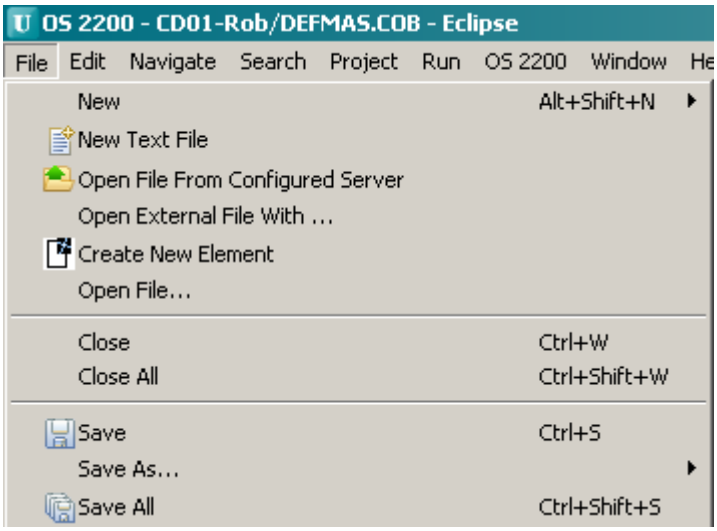
Eclipse uses CIFS to update the OS 2200 program file element. The following example shows using the diskette icon: Clicking the single diskette icon will save the current program. Clicking the multiple diskette icon with save all updated elements.



The next example using right-click in the editor pane appears in the next screen snapshot:



Finally you can use **File → Save**.

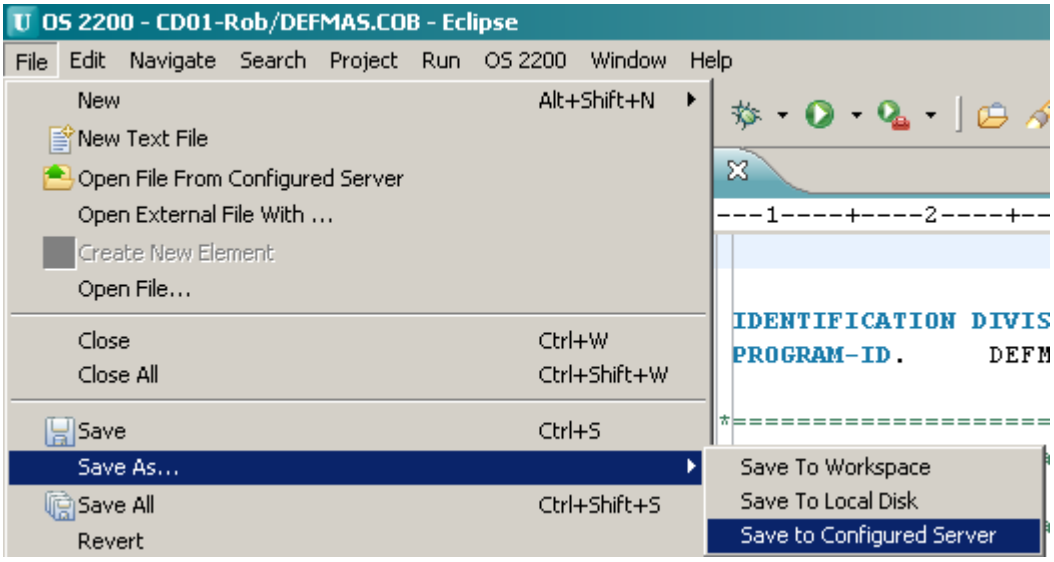


Using the Save-As option

Eclipse provides the ability to use a Save-As function with 3 options:

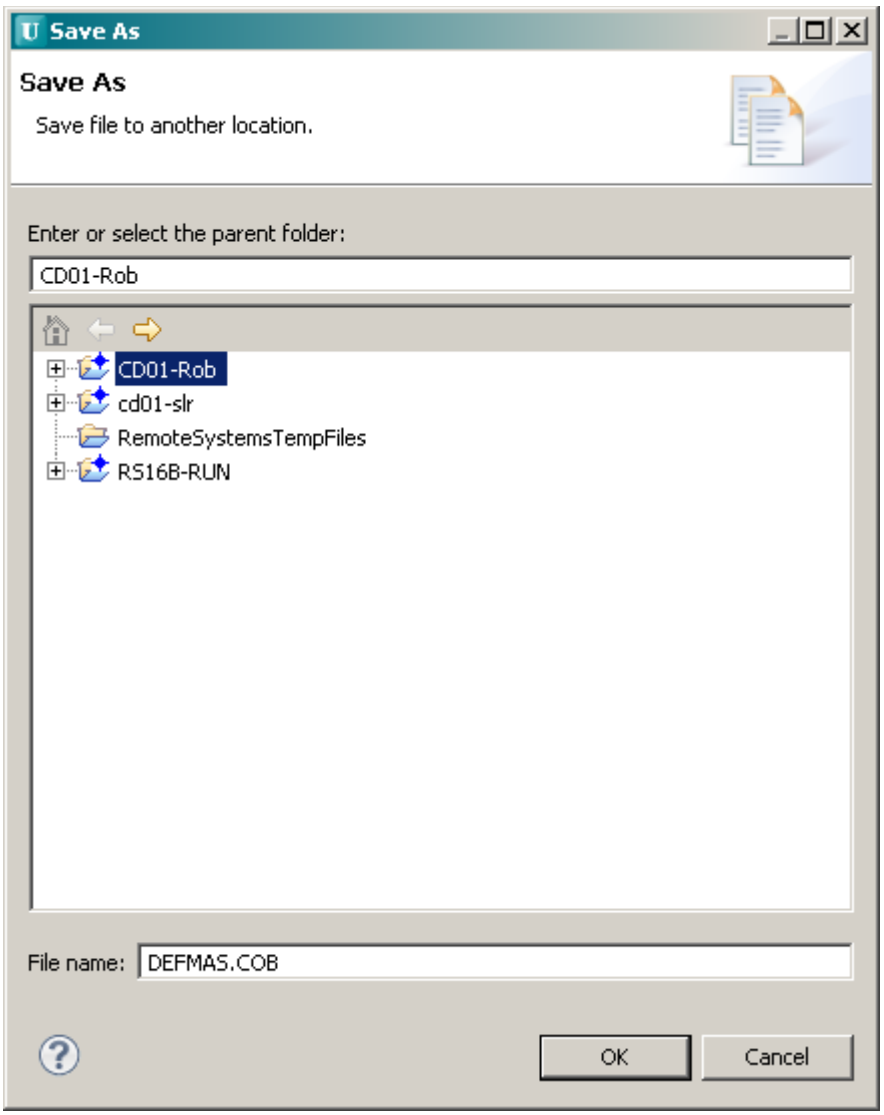
1. Save to the Workspace
2. Save to Local Disk
3. Save to a Configured Server

Go to **File -> Save As…**



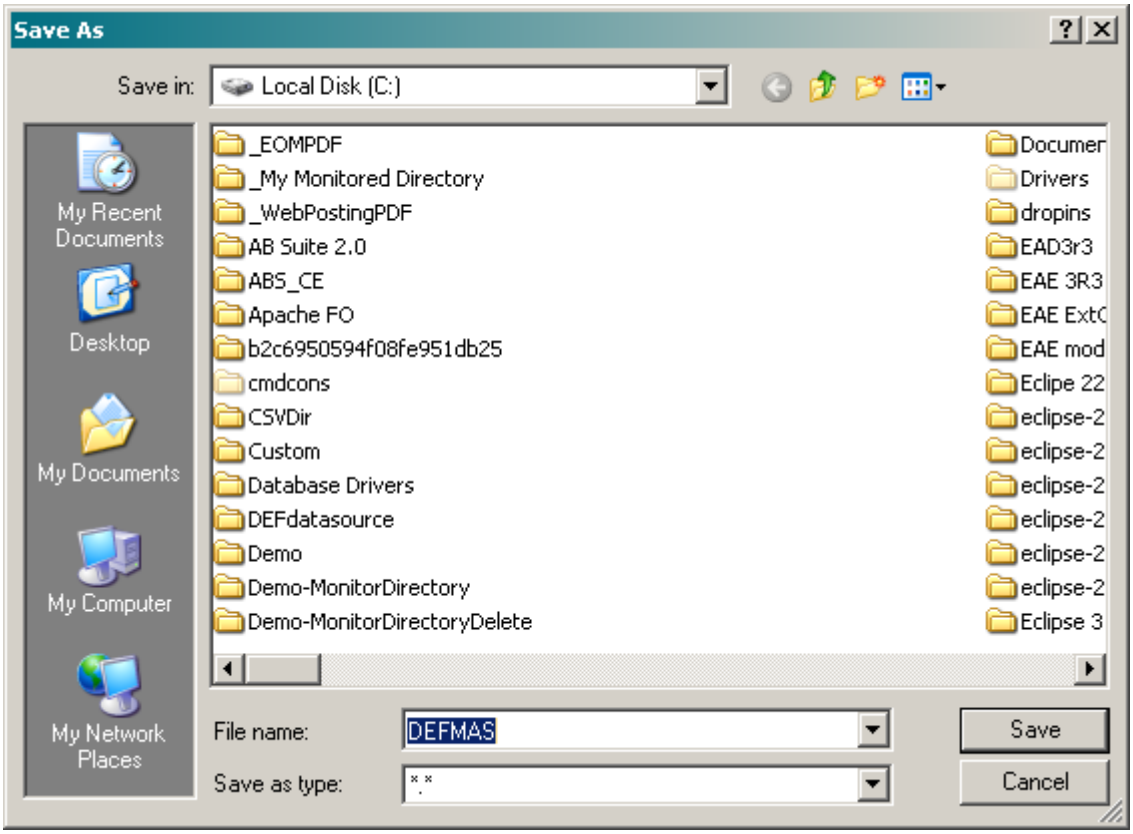
Save To Workspace

This option allows you to save the source file to a project defined in your workspace. For OS 2200 Projects, it does not save the source to the associated 2200 work file. Highlight the destination project, change the file name if required and then click OK.



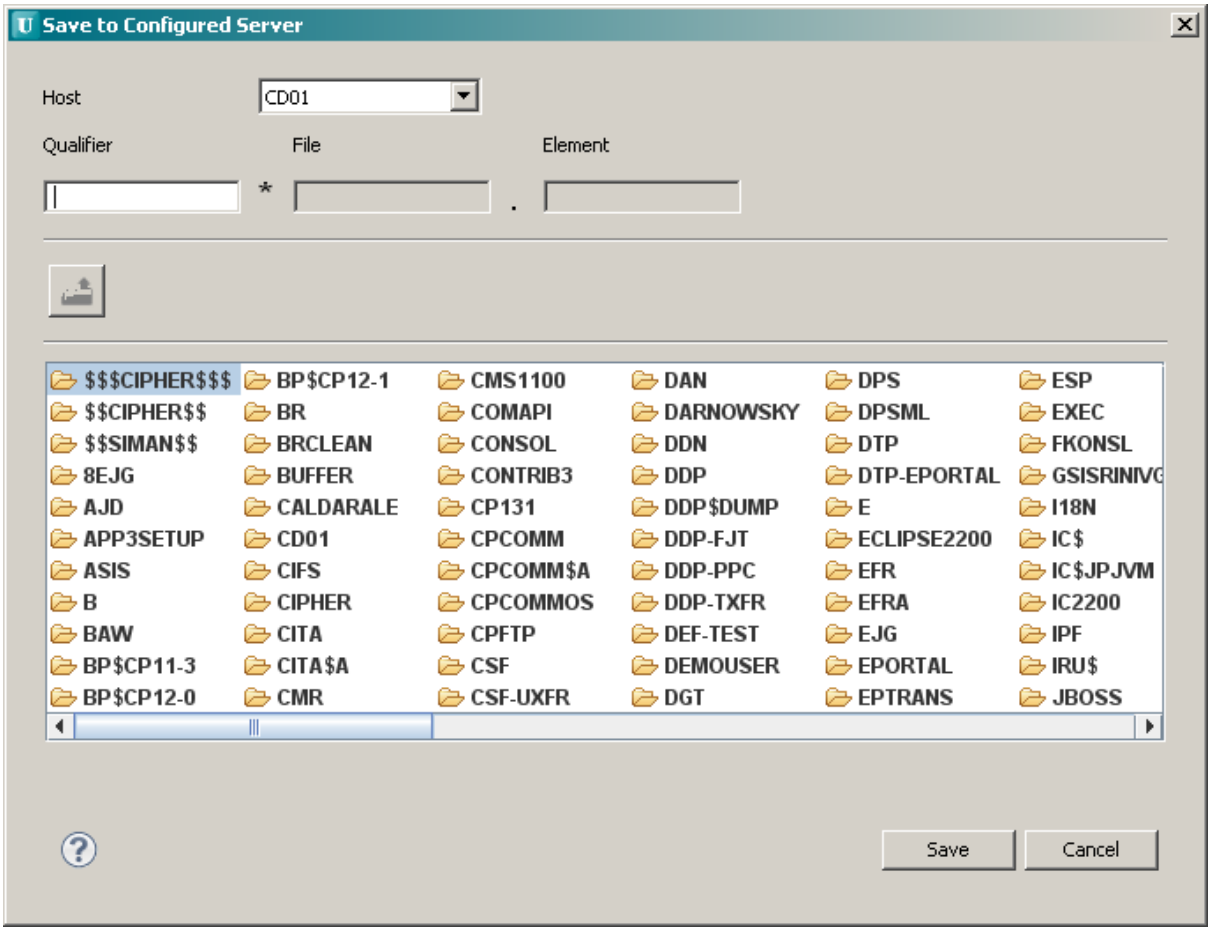
Save To Local Disk

Use this option to save the source file to any folder accessible by your workstation. You can browse to select the drive from the "Save in:" text box. The File name and file type can also be changed if required. Then click OK.

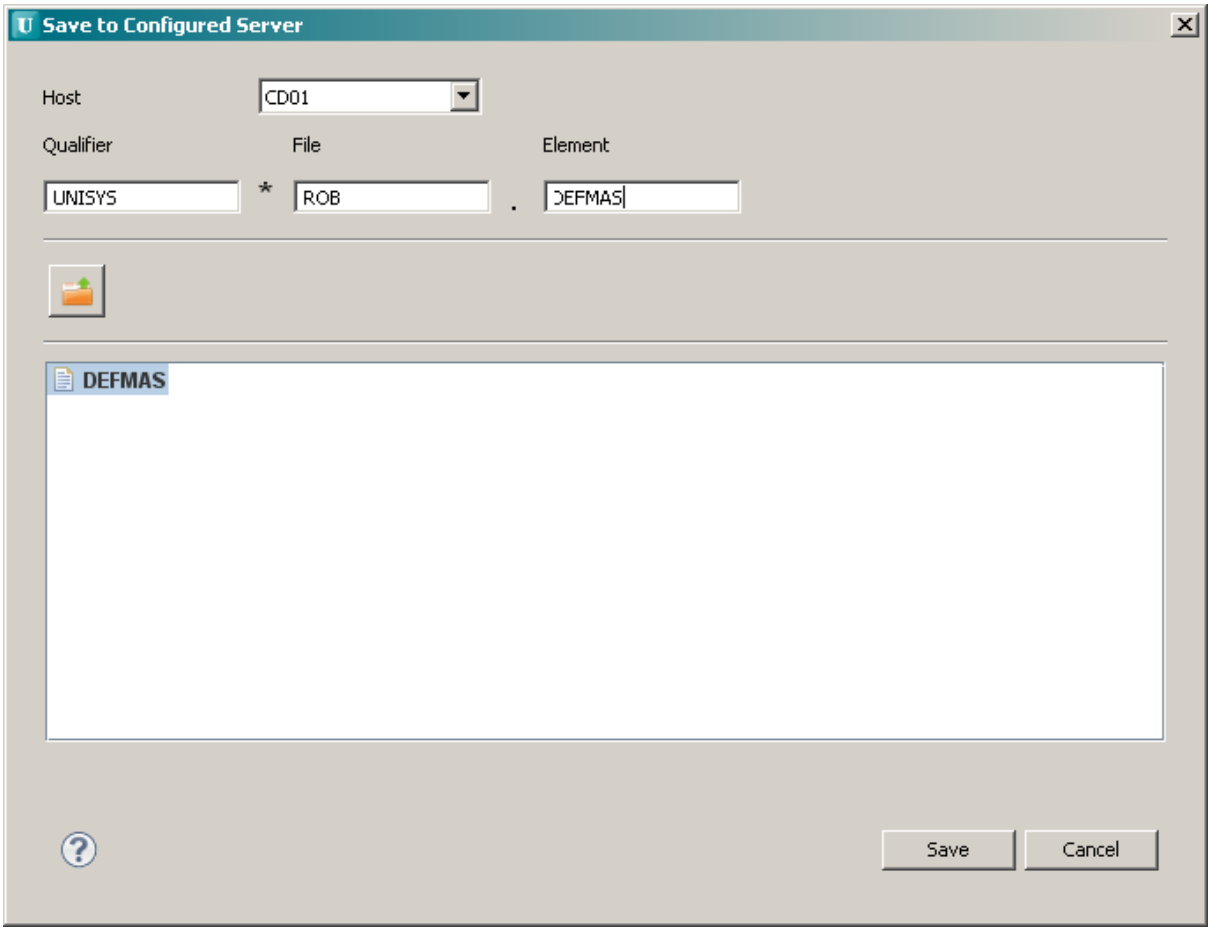


Save To Configured Server

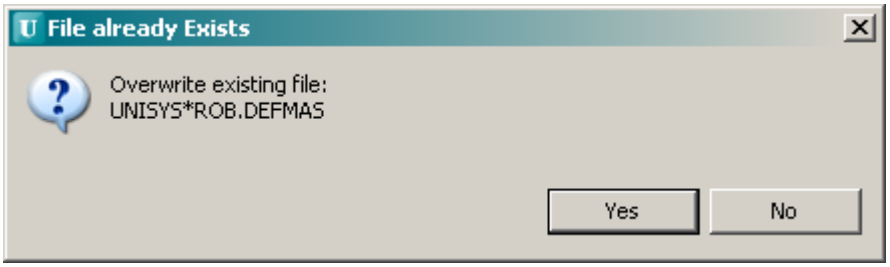
This option will save the file to the selected OS 2200 host. The available OS 2200 hosts are based on the host connections defined for this workspace.



The format of the qualifier, filename and element are validated. The element can also have a version e.g. elt/ver.



If the destination file or element exists, Eclipse will report this:



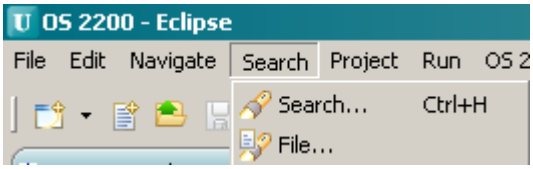
If you try to "Save As" to an element, the OS 220 work file must exist. You can save to a new file in which case Eclipse creates the file as private on fixed with default options.

```
@prt,f unisys*rob2.
FURPUR 32R5B (110805 0929:33) 2013 Jun 23 Sun 1031:46
 * *  PROJ: RJJ              ACCNT: 0            * *
UNISYS*ROB2(1),F/0/TRK/262143
MODES: PRIVATE
NO. OF GRANULES ASG-D:      14      GP2=14
HIGHEST GRANULE ASG-D:      13      TOTAL ASSIGNMENTS:     0
HIGHEST TRACK WRITTEN:      13
CAT: 06/23/13 AT 10:30:42, LAST REF: 06/23/13 AT 10:31:10
```

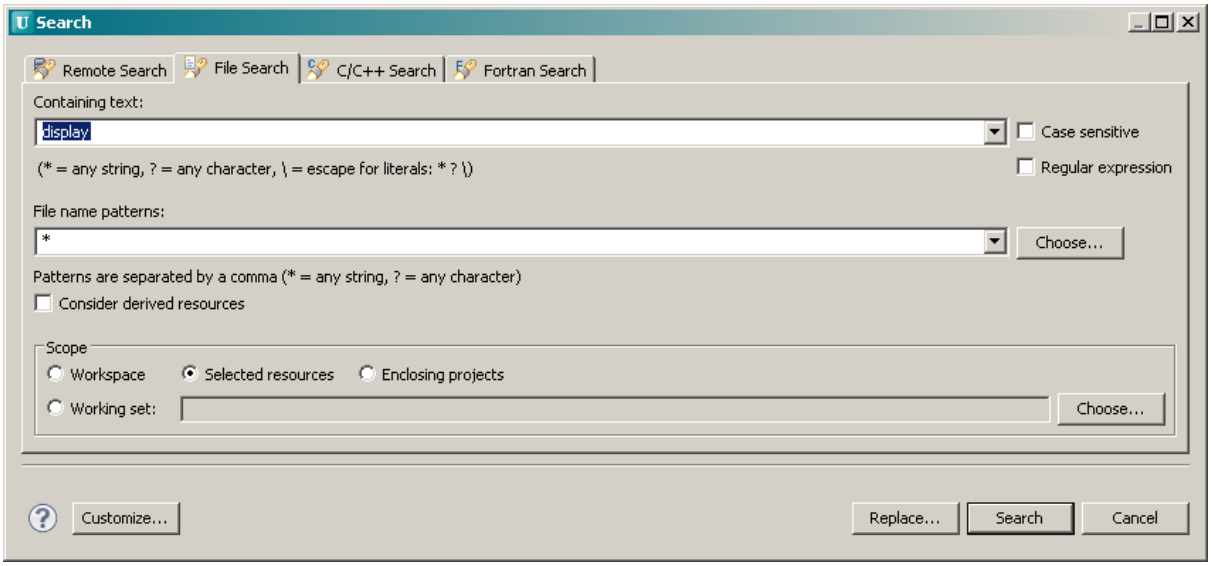
Searching Files, Projects and Workspaces

Eclipse provides an inbuilt searching capability that works on files, projects and workspaces. It does not work on data files or OFCS opened elements.

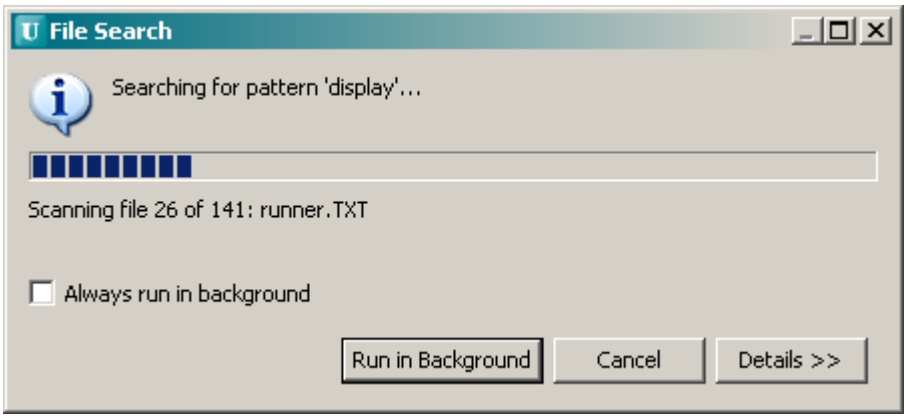
Go to the **Menu -> Search**



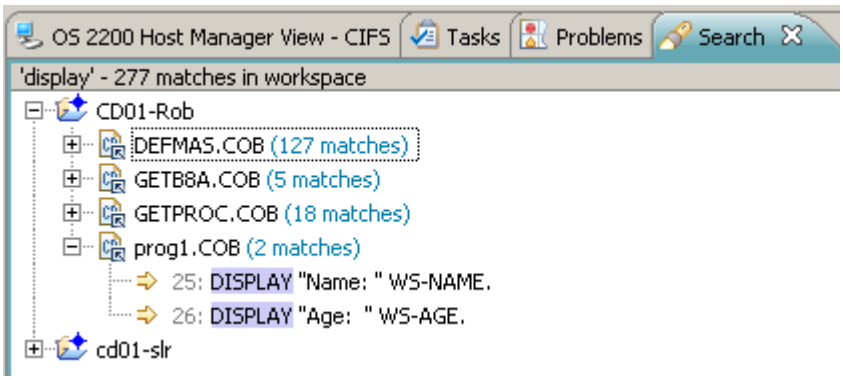
Select either Search or File. File is probably the most likely used option for OS 2200 Projects. In the Search wizard, enter the search string and then set the Scope.



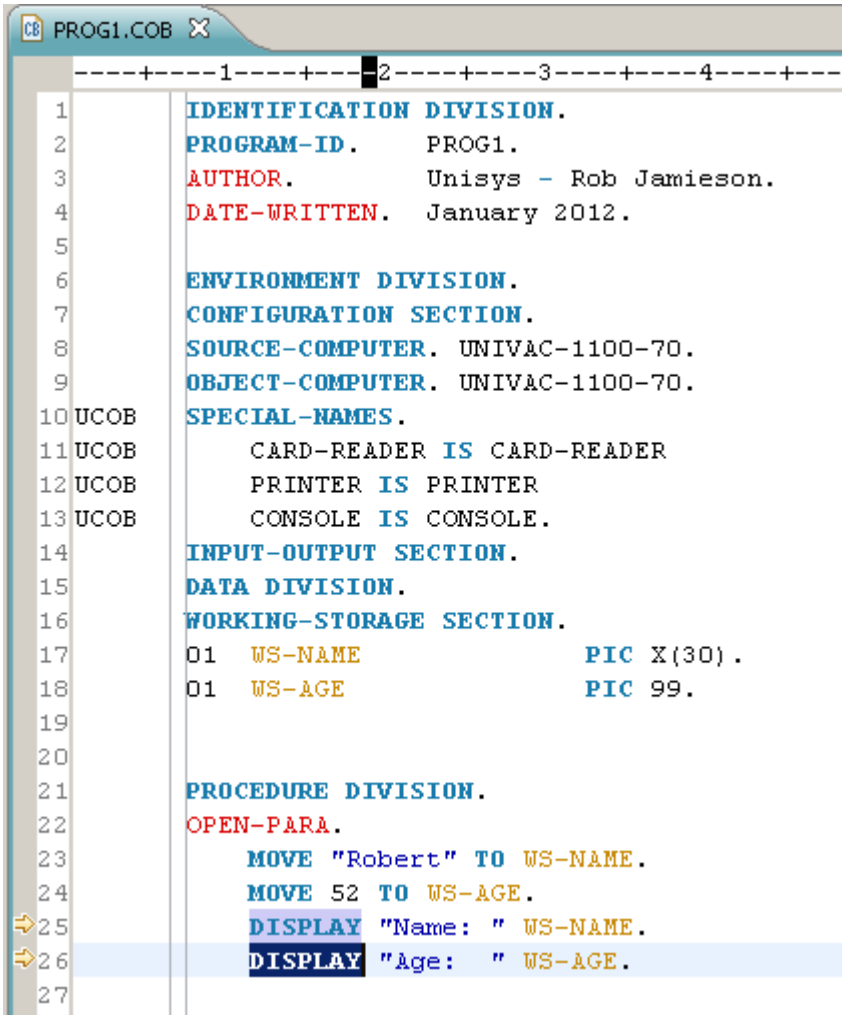
The Scope option limits the search. “Workspace” searches all projects in the workspace. “Selected resources” just searches the projects or elements highlighted in the Explorer View. Only elements in the Project are searched. Eclipse shows a progress dialog.



The results are returned in a Search View:



The results can be expanded to show the element and line number. Double click the line number to open the element in the editor and position the cursor on the line:

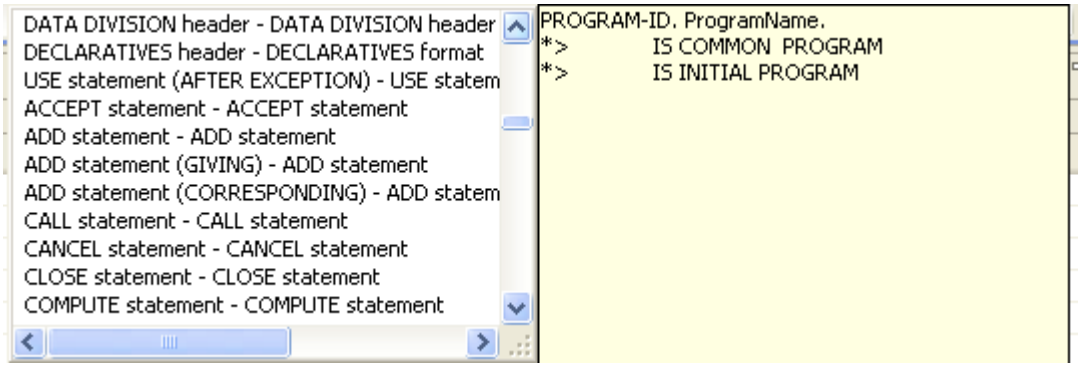


Notice the Icon  next to all lines that matched the search.

Note that data files or elements opened with OFCS can't use the Search option as the file/element is not included in an OS 2200 project. You would need to add the file/element to a project to use the search feature.

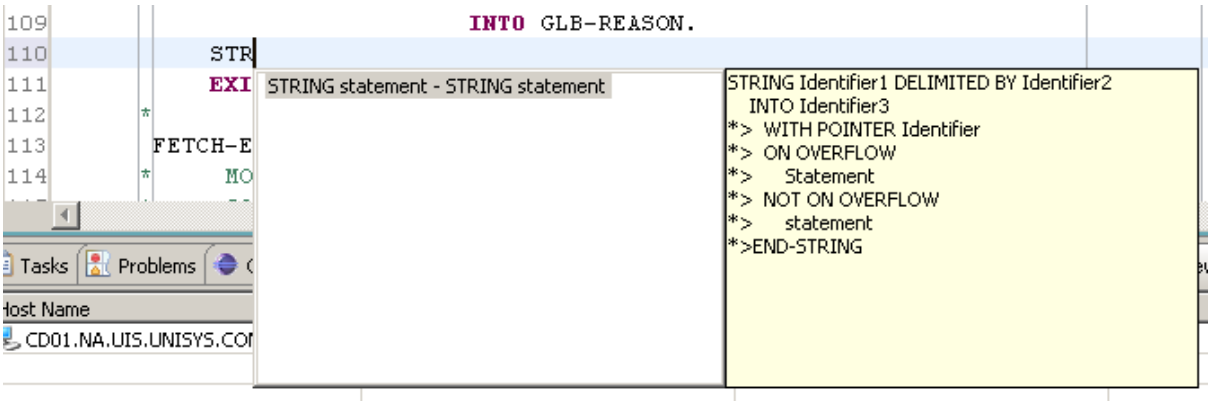
COBOL Content Assistant and Auto Completion

The Editor template also provides for content assistance and auto-completion to assist the programmer. For example, a programmer is developing a new program and cannot remember the format of certain COBOL structures and statements. The programmer can hit 'Ctrl-Space' or select Edit → Content Assistant from the menu bar. Eclipse will display a pop-up with the COBOL statements as shown below. The statements are limited to the COBOL division that you are working in.

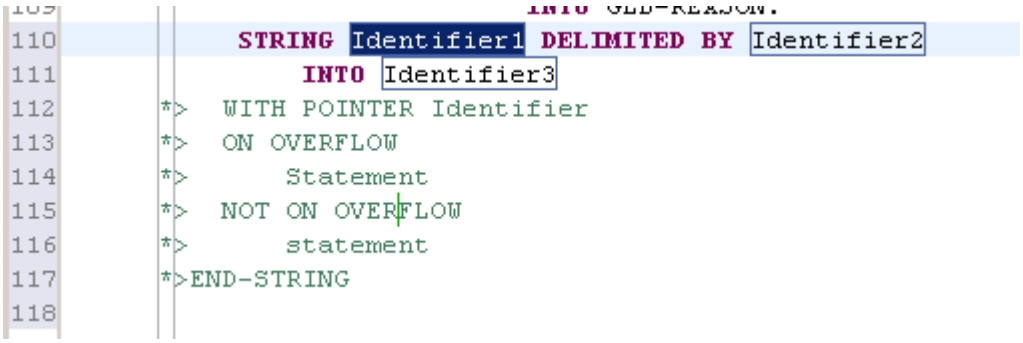


The programmer can then scroll thru the list to find the required statement. Hitting 'Enter' will result in Eclipse putting a template of the statement in the program source where the cursor was positioned.

If the programmer knows the first character/s of the verb but not the rest of the statement, then they can use auto-completion via the content assistant to help. For example, if you enter 'STR' as the first characters on a new line and then invoke the content assistant (Ctrl-Space), Eclipse will show a pop-up with only those COBOL statements that begin with 'STR'.



By transmitting 'Enter', the STRING statement template as shown in the right pop-up is inserted in the code.



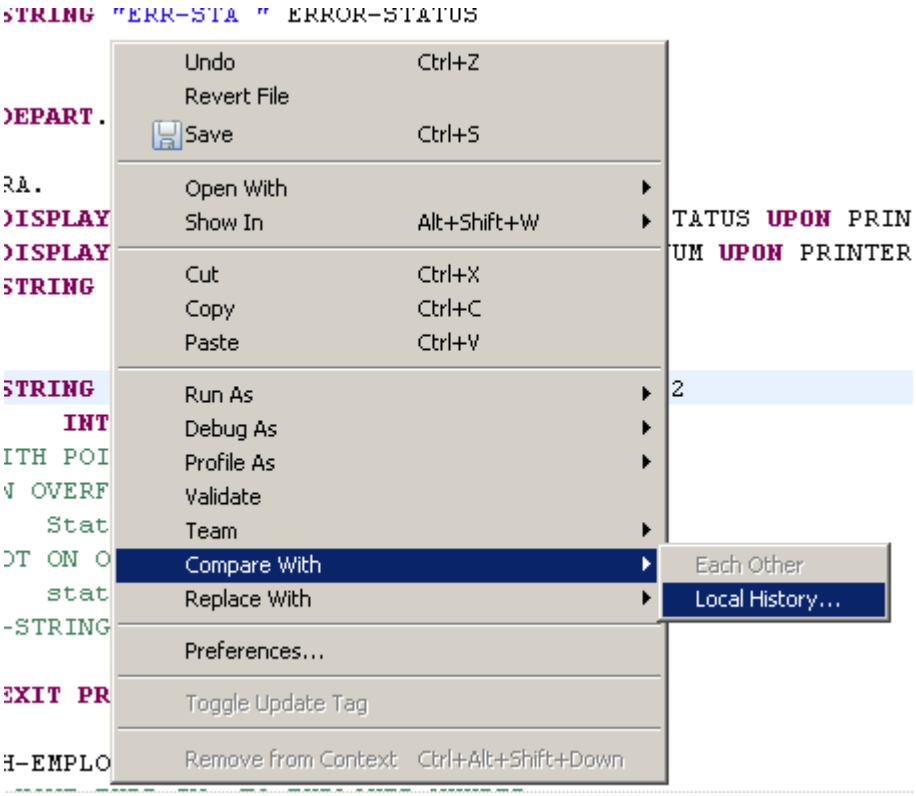
The cursor is moved to the Identifier field where the programmer can type the appropriate variable name. If the statement has more than one variable, Eclipse will display multiple Identifier fields that can be tabbed to.

Comparing different source versions

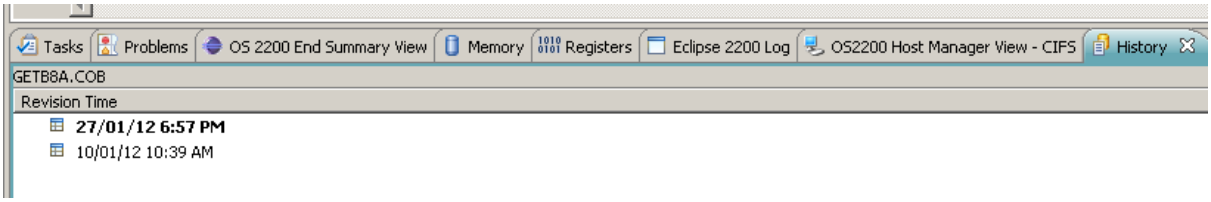
As mentioned earlier, Eclipse maintains different source versions in the workspace on the workstation.

Local History Compare and Replace

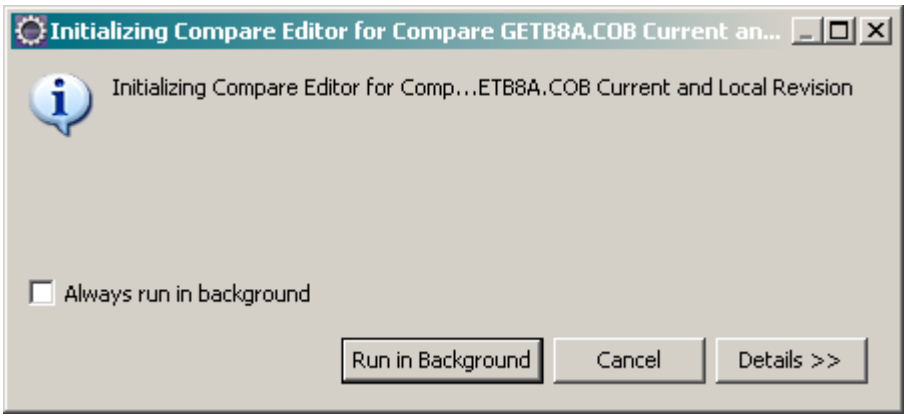
You can compare the current source to older sources in the same project by right-clicking the program name in the Explorer tree (or right click on the source code tab) and then selecting the Compare With option followed by the Local History option.



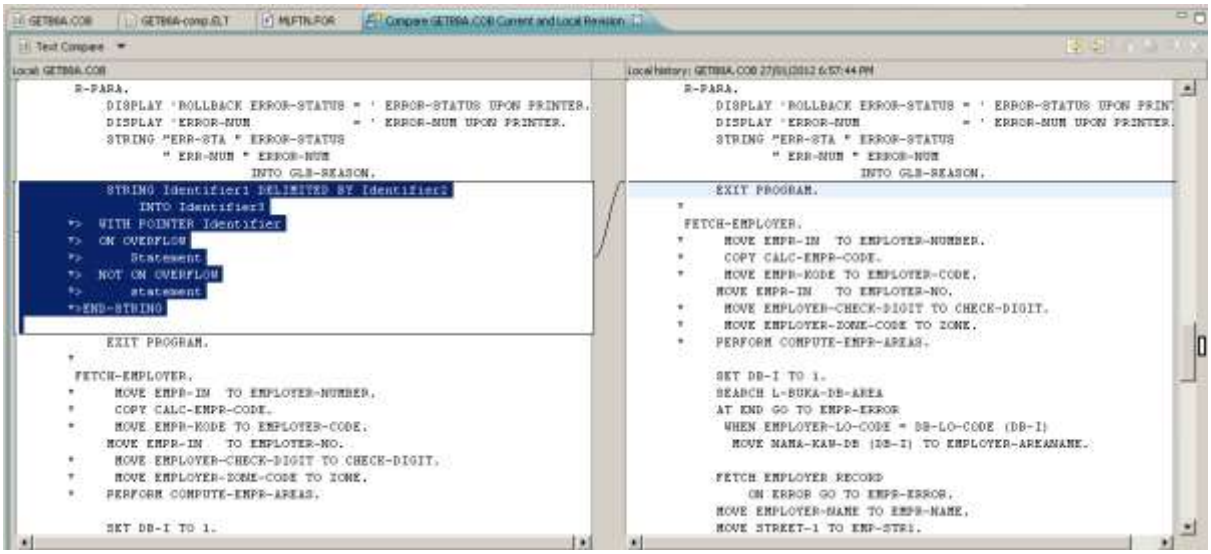
Eclipse will present the list of versions of the program in its' local history. A new tab, History, is opened in the diagnostic window at the bottom of the screen.



Double-click on the version to be compared with.

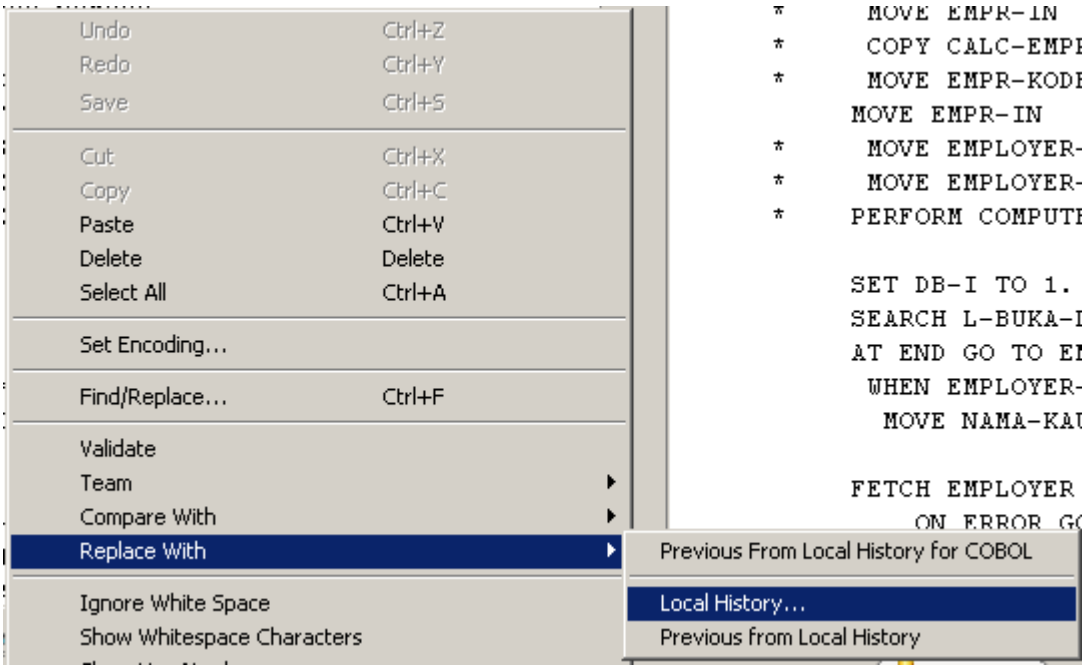


After the Eclipse Compare Editor is complete, you will see a window with two panes for each version listed side-by-side. Notice the title of the tab and that each window has a title indicating where the source is from.

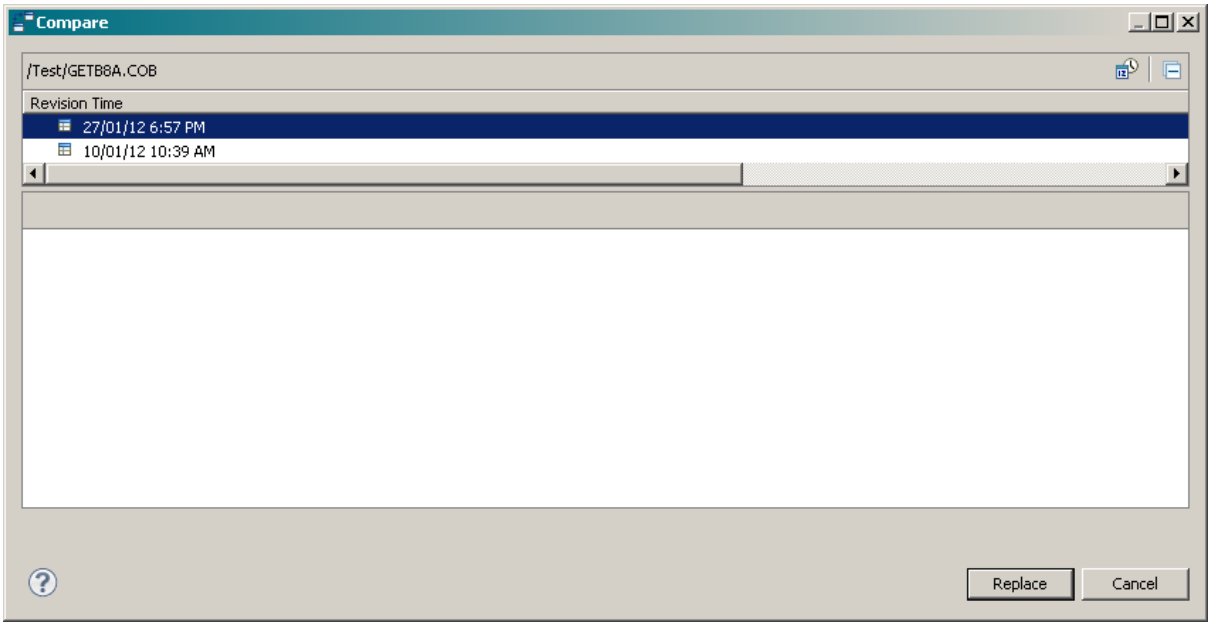


You can easily determine the differences in the source code. In the above example, the latest saved version has the STRING statement that we inserted earlier. This code is in a box and a line shows where it would be inserted into the previous saved version (in local history). Eclipse does support a restore command if you want to restore from a local history version.

Right click in the editor pane and select **Replace With** and then **Local History…**



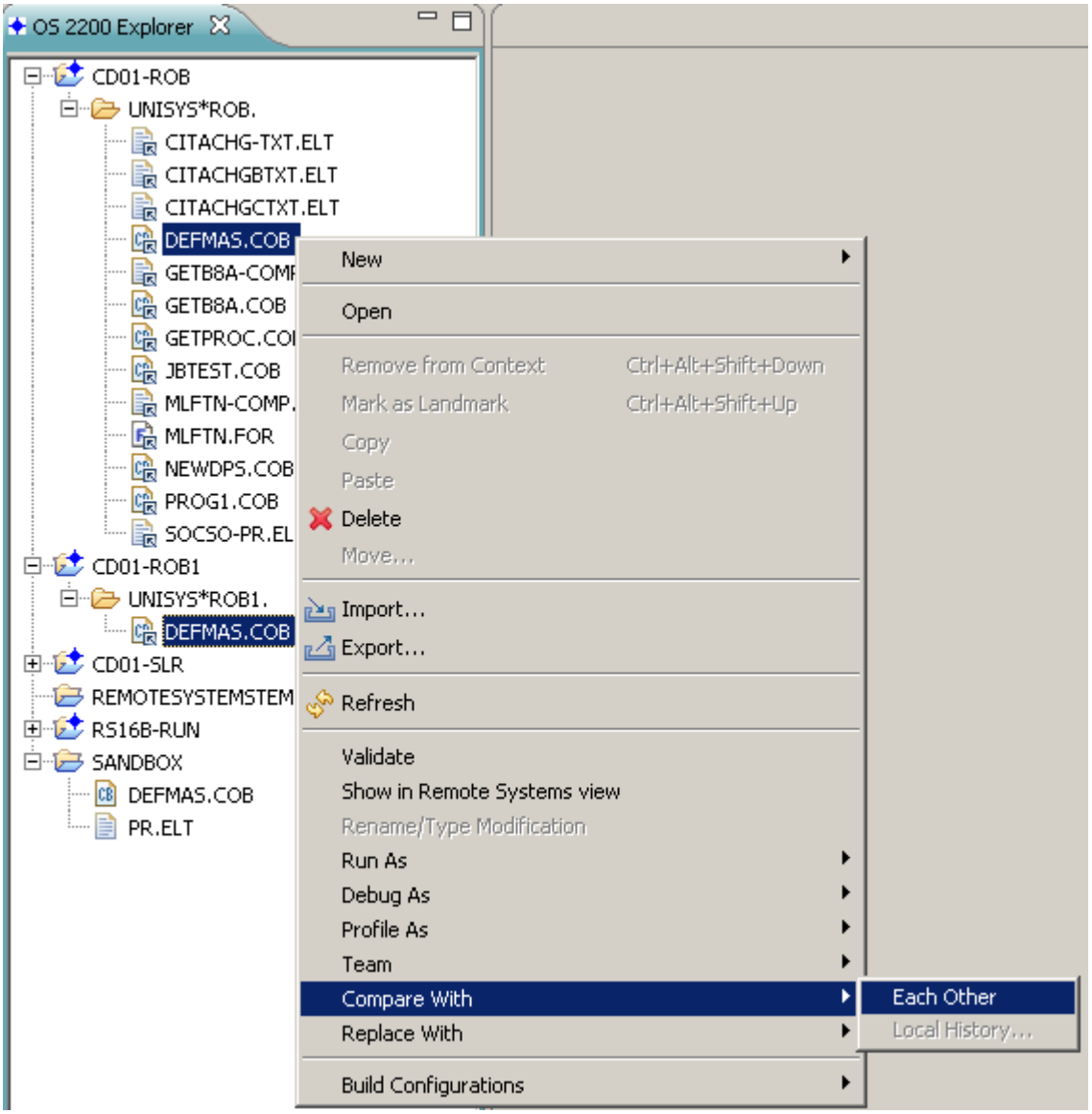
The local history versions are displayed. Select the version to restore and click **Replace.**



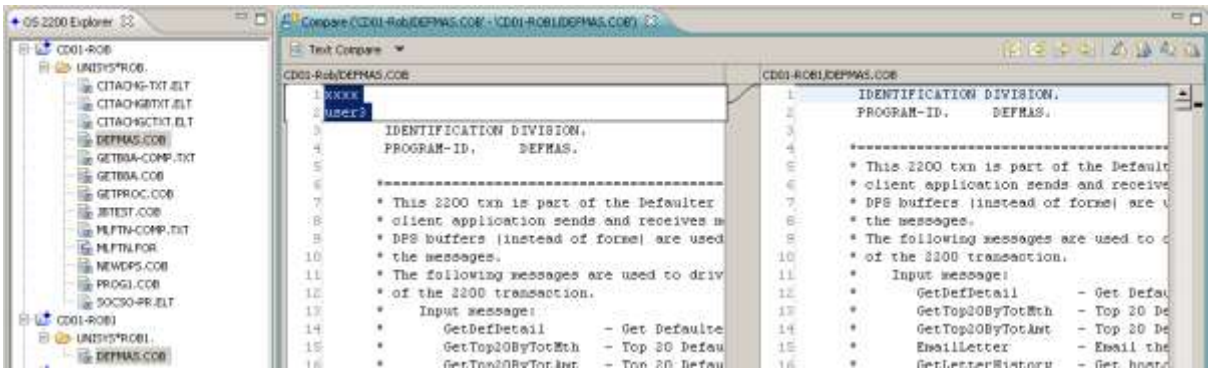
Note this will replace and save the file so the OS 2200 work file element is also updated. The new source appears in the editor pane.

Comparing Different Source Element

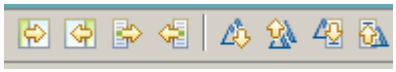
Eclipse allows you to highlight different source elements and then compare the two sources. Highlight one source and then move the mouse to the other source and Ctrl+Click. Right click on a highlighted element and select **Compare -> Each Other**



The result appears in an editor window:



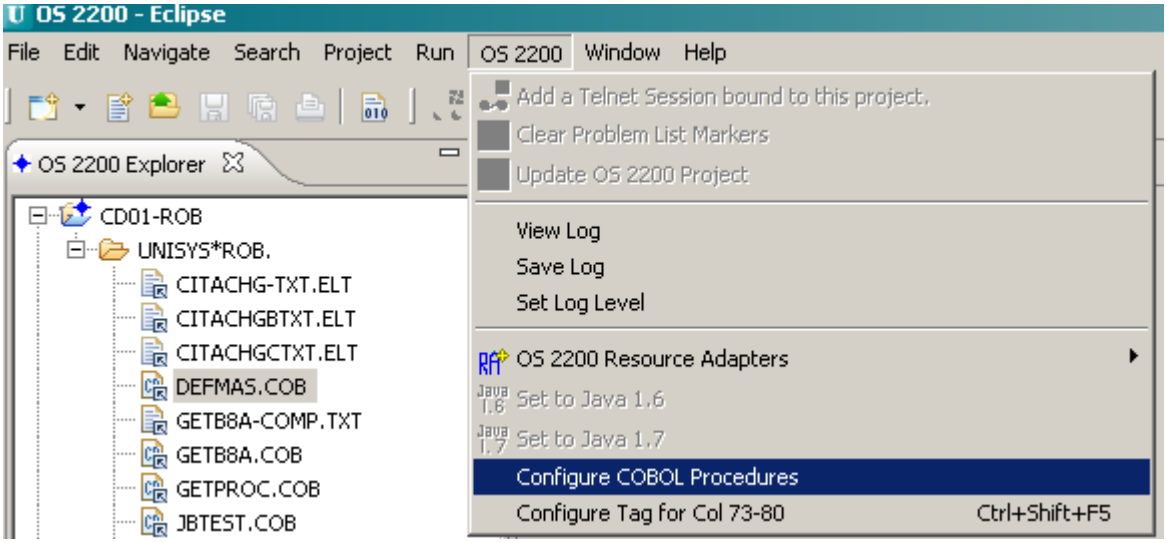
Note the highlighted entry used for the right-click to initiate the compare is the source and appears in the left pane.

The icons on the top right  provide a means to copy all source in either direction or only the changes. Other icons provide navigation through the compare window. Hover over each icon for more details.

Referencing COBOL COPY Procedure Source

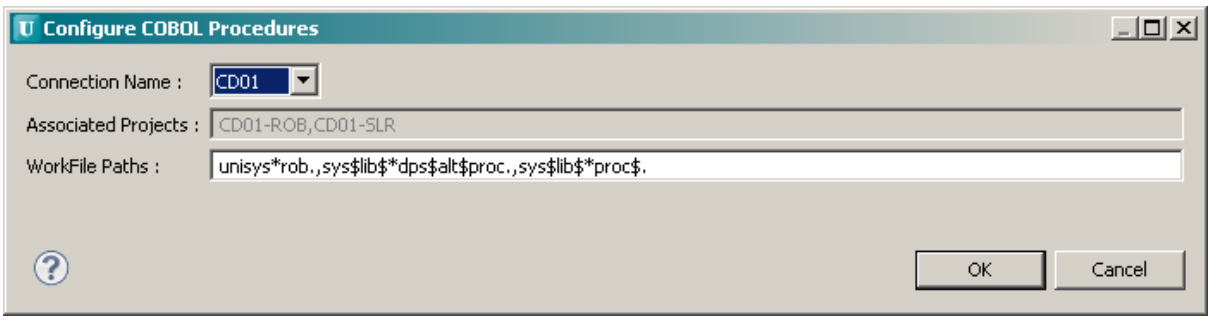
Eclipse provides a method to define the locations of the COBOL copy procedures used by the COPY statements in a program. You can refer to the COBOL compilation stream to determine what files are used. For example, there maybe an @USE to COB$PF or an @USE to a use name that appears on the COPY <name> IN/OF <PDP file>.

Go to **Menu -> OS 2200 -> Configure COBOL Procedures**

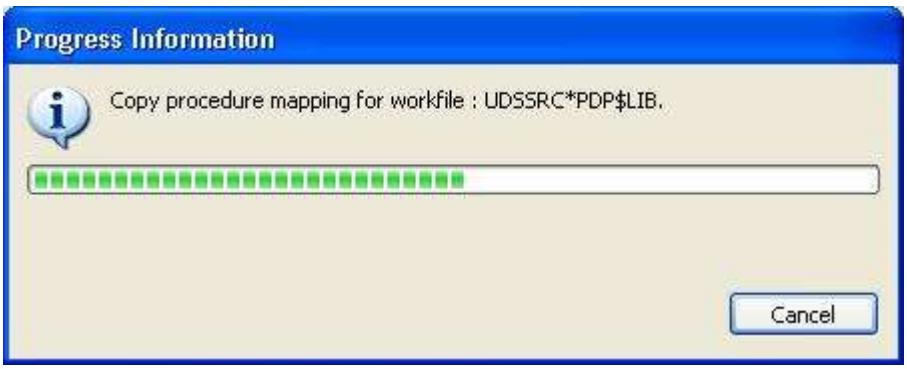


Select the host connection name.

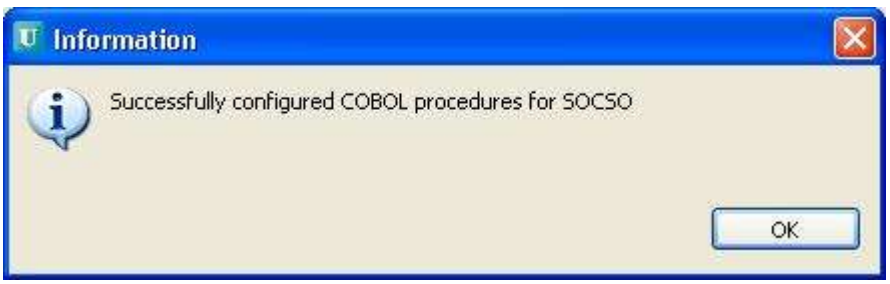
In the Workfile Paths, enter Q*F.,Q*F., etc. Unlimited number of files can be entered.



When you click OK, Eclipse starts the process and shows a progress dialog:



There is a complete message for the host connection:



Eclipse will build information into either of the following folders:

Windows XP
C:\Documents and Settings\<User>\Local Settings\Application Data\Unisys\os2200\dd

Windows Vista \ 7
C:\Users\<User-id>\AppData\Local\Unisys\os2200\dd

If you check the DD folder, you will see the following files.

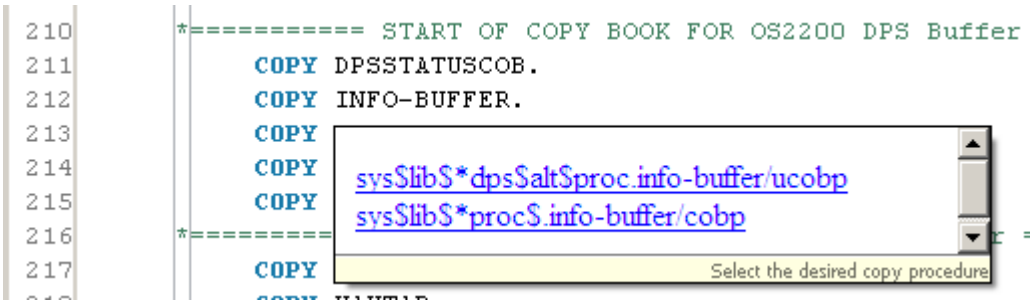
The 'copyProc.properties' file contains the files from the wizard:

```
#
#Thu Feb 14 16:24:47 GMT+08:00 2013
CD01=unisys*rob.,sys$lib$*dps$alt$proc.,sys$lib$*proc$.
SOCSO=SYS$LIB$*DPS$ALT$PROC.,Y2K*SOURCE.,UDSSRC*PDP$LIB.,UDS$$SRC*T-SCHEMA.
```

If you open one of the host connection files, you will each COBOL procedure entry name and the element it is found in including the host IP address, share, qualifier, filename and element name.

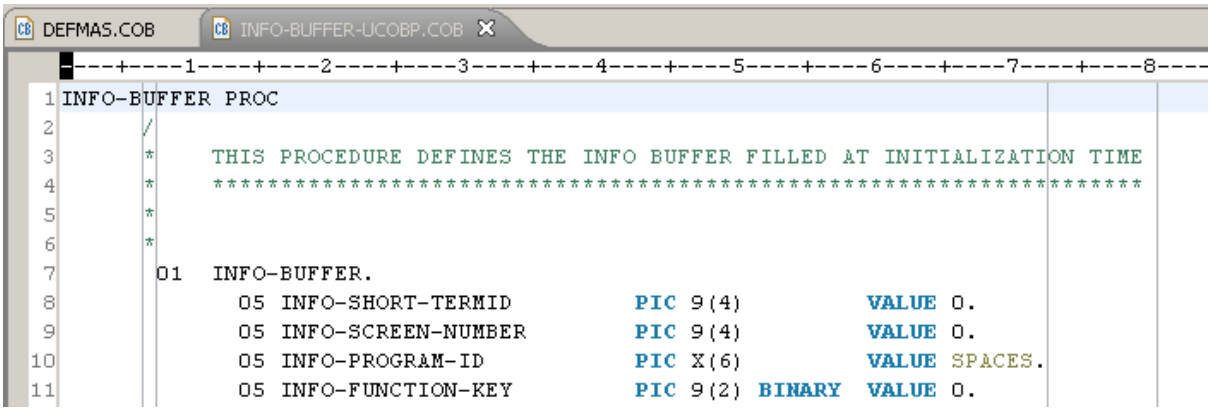
```
ABC \\cd01.na.uis.unisys.com\os2200\unisys\rob\procelt.130130
ALONGPROCEDURENAME \\cd01.na.uis.unisys.com\os2200\unisys\rob\procelt.130130
SCREEN-BUFFER \\cd01.na.uis.unisys.com\os2200\sys$lib$\dps$alt$proc\screen-defs.ucobp
SCREEN-DEFINITIONS \\cd01.na.uis.unisys.com\os2200\sys$lib$\dps$alt$proc\screen-defs.ucobp
SCREEN-PRINT-BUFFER \\cd01.na.uis.unisys.com\os2200\sys$lib$\dps$alt$proc\screen-defs.ucobp
GETSCREEN \\cd01.na.uis.unisys.com\os2200\sys$lib$\dps$alt$proc\get-screen.ucobp
PUTSCREEN \\cd01.na.uis.unisys.com\os2200\sys$lib$\dps$alt$proc\put-screen.ucobp
DPS-GET-WS-BUFF \\cd01.na.uis.unisys.com\os2200\sys$lib$\dps$alt$proc\get-ws-buff.ucobp
SCREEN-TEMP-0 \\cd01.na.uis.unisys.com\os2200\sys$lib$\dps$alt$proc\screen-0.ucobp
DPS-NAME-TABLE \\cd01.na.uis.unisys.com\os2200\sys$lib$\dps$alt$proc\name-table.ucobp
DPS-ATTR-TABLE \\cd01.na.uis.unisys.com\os2200\sys$lib$\dps$alt$proc\attr-table.ucobp
DPS-CONFIG-BUFFER \\cd01.na.uis.unisys.com\os2200\sys$lib$\dps$alt$proc\config-buff.ucobp
```

With the COBOL source open in the editor, hove the cursor over the COPY Procedure name . In the following example, this is INFO-BUFFER. A dialog with options to select the COPY procedure source element appears. Note al element could contain many COBOL procedures.



If no COPY Procedure has been associated with the name, then an error is displayed in the status line:

By selecting an element name in the list, Eclipse will open the element in the COBOL editor and position the cursor at the PROC statement. Note this element is not part of a project and so Search plus other functions cannot be used.

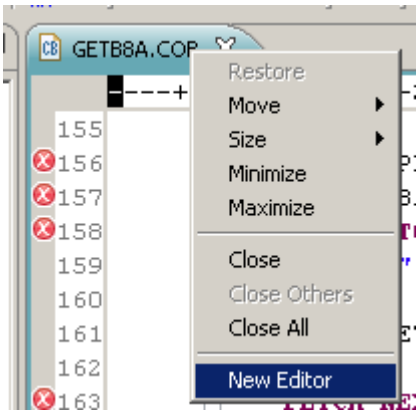


Note that if a COBOL source has been opened with OFCS, this feature is unable and the following error is shown on the status bar:

Splitting the Editor Pane

Sometimes a developer is coding the business rules in the Procedure Division but needs to reference some other code in the program e.g. the Working Storage section. Eclipse provides the functionality to split the editor pane into two panes. Modifications made to one pane are reflected in the other pane. If you save one pane, it saves the common course displayed in both panes.

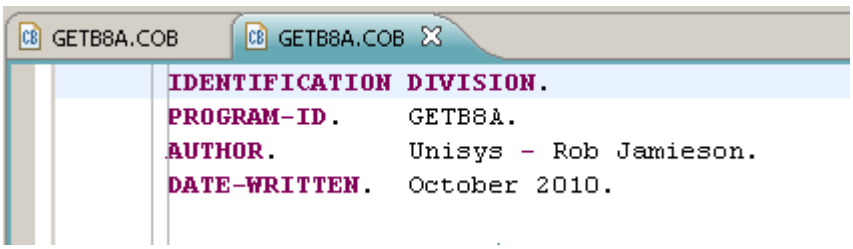
To perform this, right click on the editor pane tab e.g. GETB8A.COB in the picture below.



Then click **New Editor**.

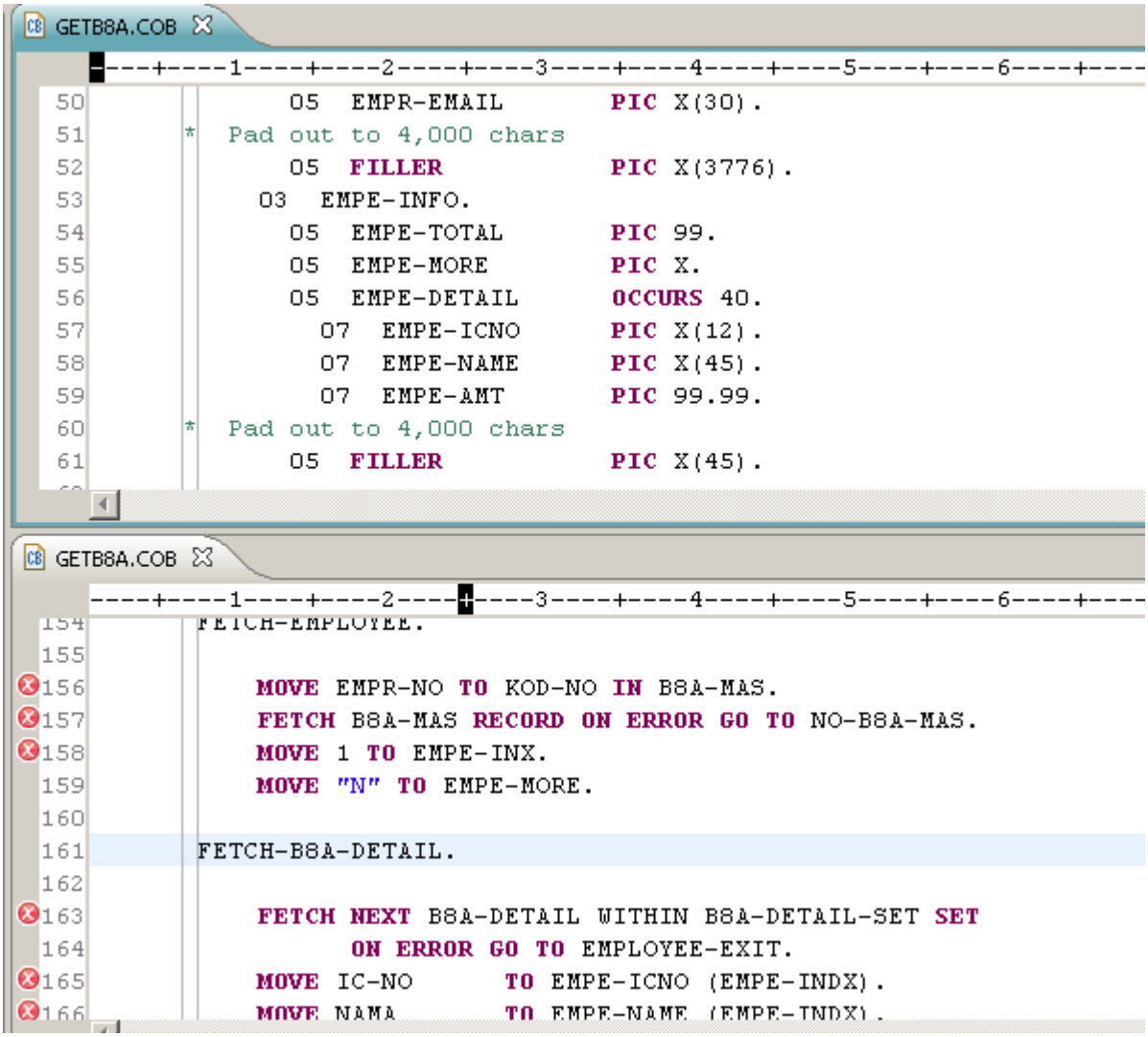
Or go to the menu **Window → New Editor.**

Eclipse will open a new COBOL editor pane as shown below. Both panes are displaying the same source. Updating the source in one pane will also update the source in the other pane. In fact there is only one source being edited but two panes used for editing.



Now with the cursor on the tab, hold down the left mouse key and drag the editor pane down the screen. When the cursor is close to the bottom of the edit window, you will see a dotted line across the

middle of the edit window and a bold black down arrow on the screen. Release the left mouse key and you will have two editor panes for the same source.



Each pane can be manipulated independently but both panes are working on the same source element. In this way, one pane could show part of Working-Storage and the other pane might show part of the Procedure Division. This is useful if you can't remember data names when coding the program.

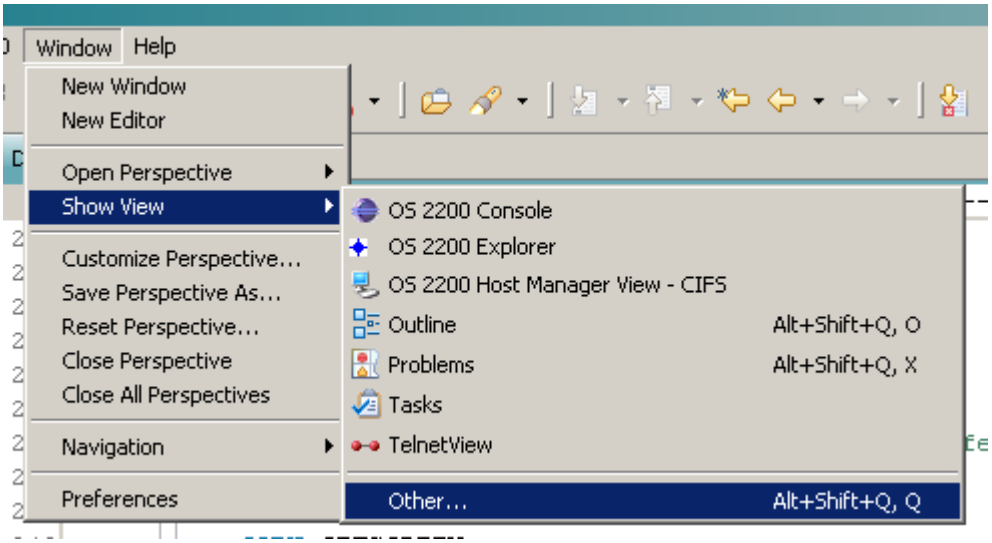
Navigating your COBOL Program Source

Eclipse provides a couple of methods to simplify the navigation of your COBOL program source:

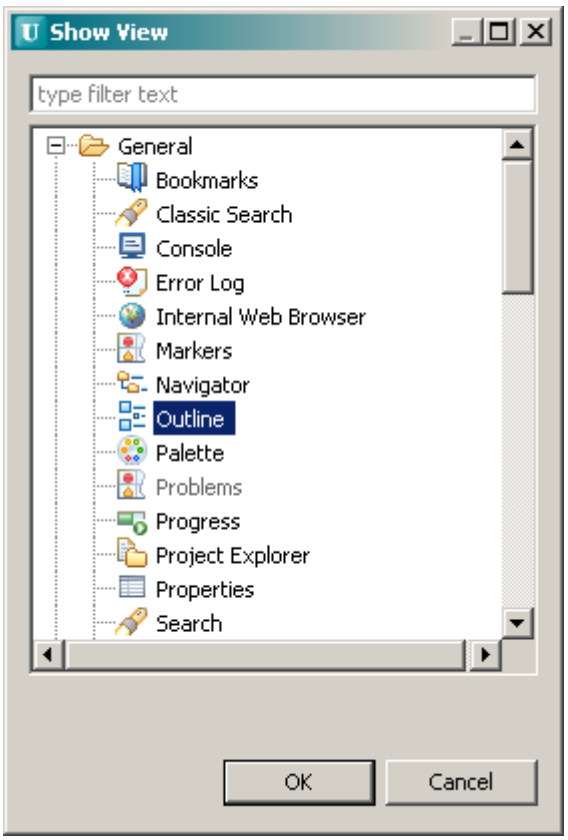
1. Using the Outline View
2. Using Sections and Paragraphs

Outline View

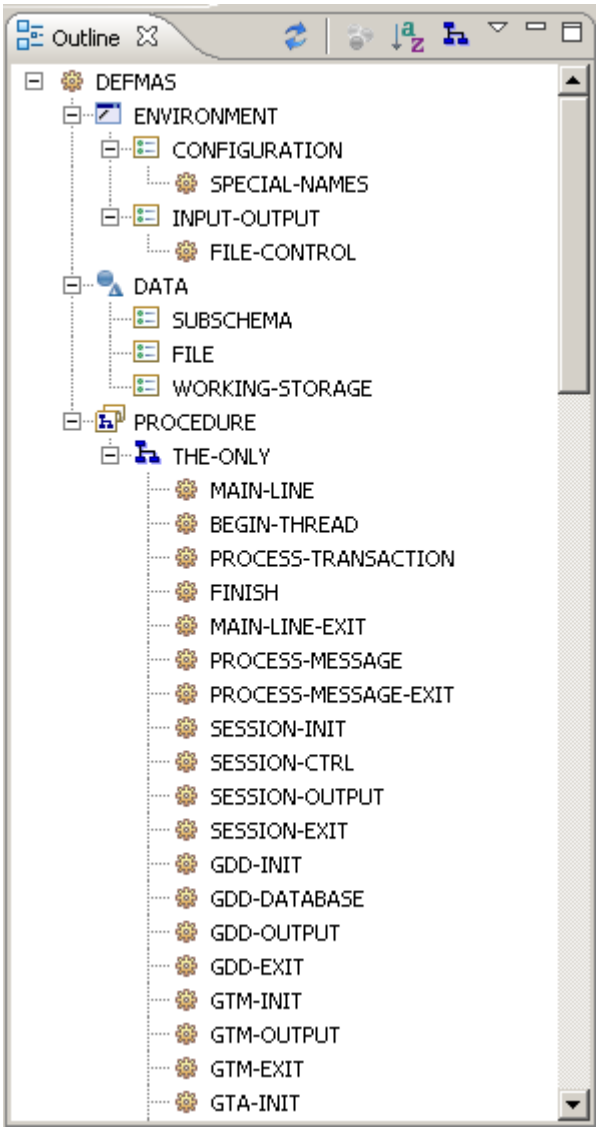
In the COBOL perspective, you select the Outline view to see the program structure. Go to **Window → Show View → Other.**



Then select **General** and expand the node. Then click on **Outline**.



The perspective will show the program outline with the different COBOL divisions and paragraph names.



Drag this view to a suitable location on your workbench. My preference is to have this open on the right edge.

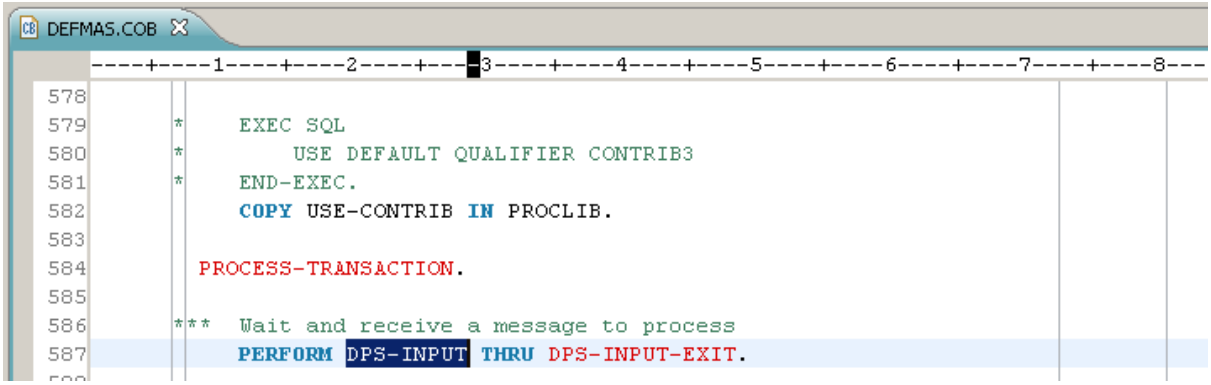
Note the refresh icon,  , can be used to refresh the outline as program editing is performed.

Clicking on an entry in the Outline will position the Editor pane at that location in the program. This provides a quick navigation method to areas of the code.

Using Sections and Paragraphs

Highlight the section or paragraph name in a PERFORM or GO TO statement.

Press F3 or

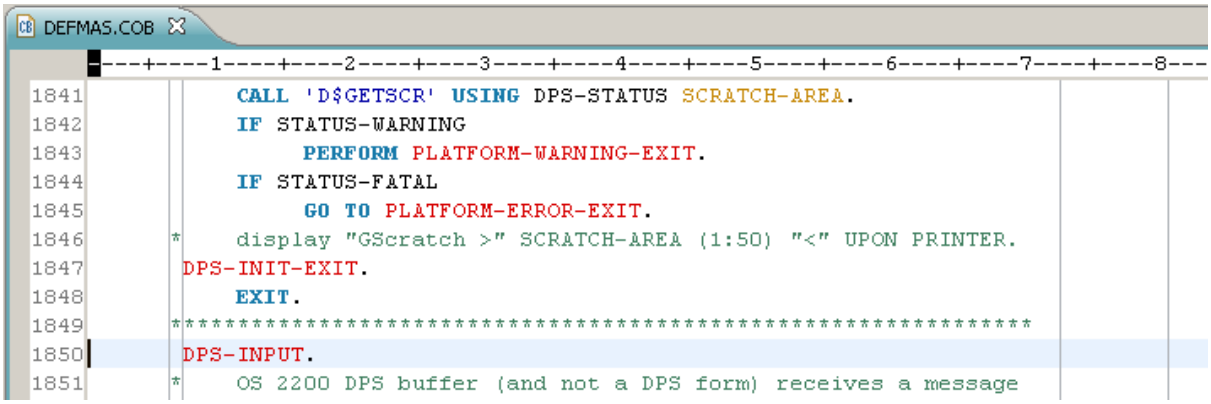


To navigate to the section or paragraph, press F3.

Or right-click and select Open Declaration:

Another method is to <Ctrl>+Click and then hover over the section/paragraph name and select the hyperlink.

The cursor is positioned to the section or paragraph name.

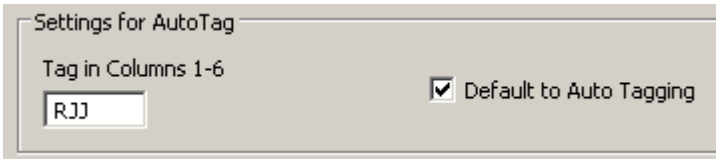


Use the Go Backwards icon  (or press <Ctrl>-Q) to return to the PERFORM or GO TO statement.

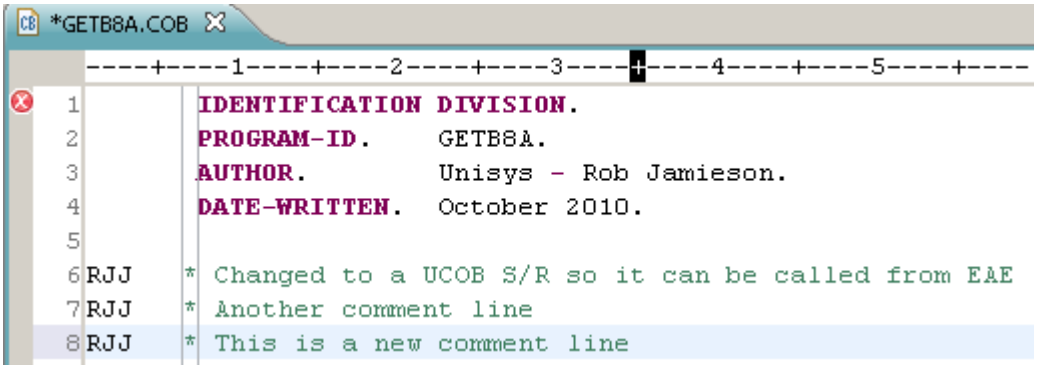
Auto Tag in Columns 1-6

The Auto Tag feature allows for up to 6 characters of data to be placed in columns 1-6 of each updated line of code in the COBOL source. Often these columns are used for version information.

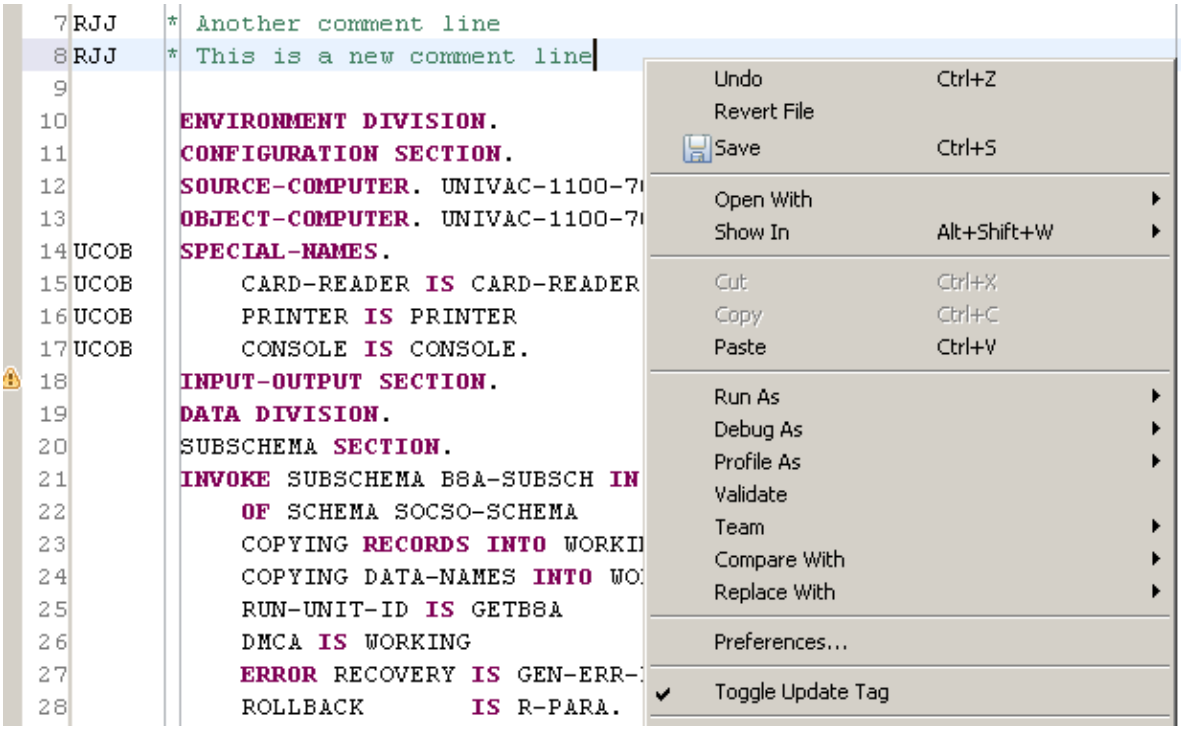
The value to be used is defined in the **Properties → Preferences → COBOL** options. Go to the **Reference Format** tab. Key you value in the field and check the "Default to Auto Tabbing" box. In the example below, RJJ is the value entered.



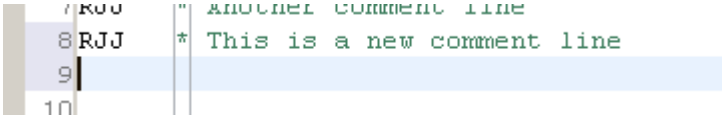
Then as you update the source (modify or insert), the tag value is automatically inserted in columns 1-6.



If you need to toggle whether the tag value appears, right click in the edit pane and select Toggle Update Tag. Lines can be highlighted as well.

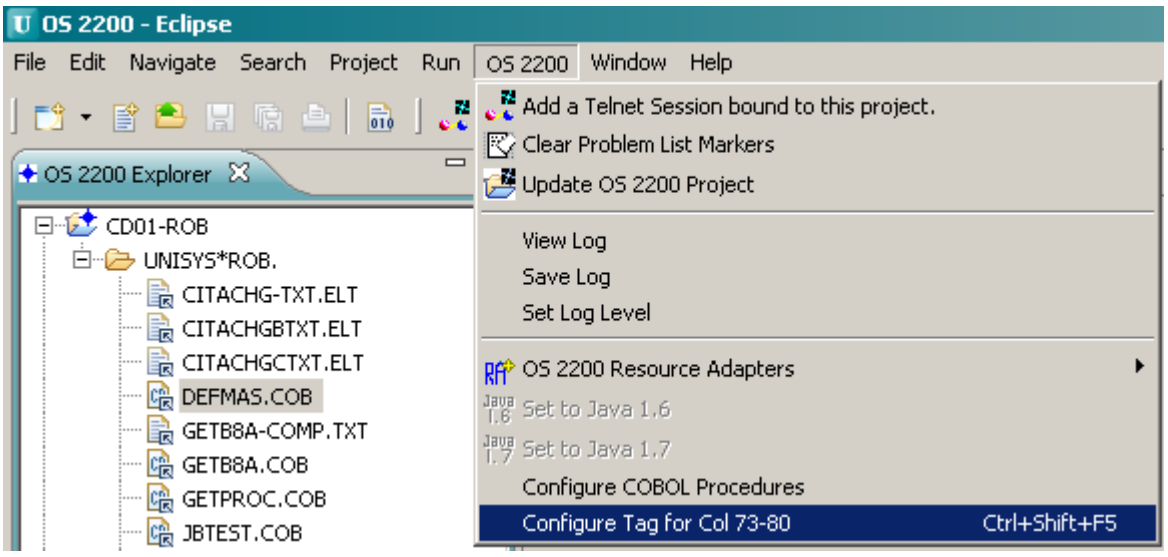


Notice that when a new line was created, the tag was not entered as shown below.

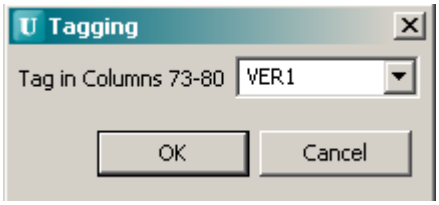


Auto Tag in Columns 73-80

Some sites want to use columns 73-80 instead of columns 1-8 to store some data like a version number.



The following prompt appears:



Data can be entered which is converted to Upper Case. The last 10 values are stored and can be selected.

When you highlight text in the Cobol editor, using <Ctrl>-T will place the selected value in columns 73-80.

```
42      INPUT-OUTPUT SECTION.
43      FILE-CONTROL.                                                    VER1
44          SELECT PRT-FILE ASSIGN TO DISC TPRINT.                       VER1
45          SELECT SRT-FILE ASSIGN TO DISC SRT-DEFAULT.                  VER1
46      DATA DIVISION.
```

Block Comment and Uncomment of Code

One or more lines of code can be marked as a comment (* in column 7) by highlight the lines of code and then doing Ctrl + /. Commented lines can be uncommented by the same process.

Literal Length

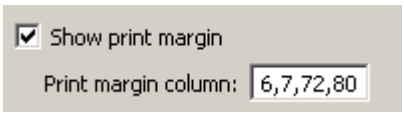
Often in COBOL, the length of a literal is needed e.g. to make sure the destination field is sized correctly. This is achieved by highlighting the code:

```
668             MOVE "InitSession" TO IN-TRANS-CODE.
```

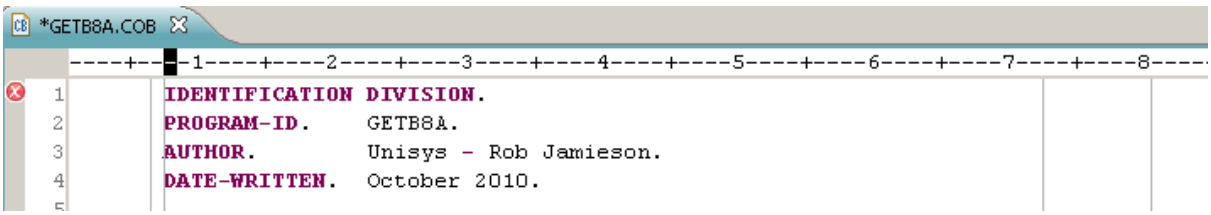
In the status line, the length of the highlighted field is displayed:

Viewing the COBOL Areas

The different COBOL areas can be clearly indicated in the COBOL Editor by setting the Show Margins options. Go to **Windows → Preferences → COBOL → General** and you will see the settings for the Print margin column.

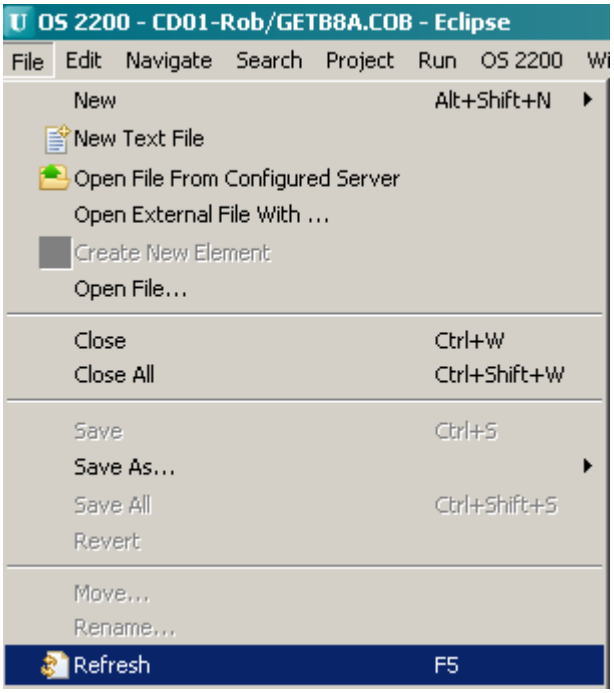


Note how the "Print margin columns" field contains 6, 7, 72 and 80. The editor will show a vertical line after these columns to indicate where the different COBOL areas are. You could modify these values for your own use.



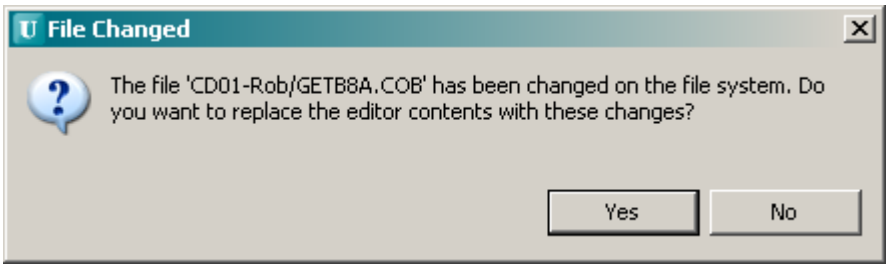
Refreshing the Editor Content

At times the developer might want to refresh their editor contents with the version of the data file or element from the OS 2200 host. This can be achieved by Menu -> File -> Refresh or by clicking F5.



Automatic Refresh when Host Contents Changed

Eclipse will be notified when the data file or element contents on the OS 2200 host are changed. In this case, a prompt is shown to allow the developer to take the necessary action.



Select Yes to refresh your editor with the latest host contents. This will result in any unsaved changes being lost.

Select No to keep working on your editor contents.

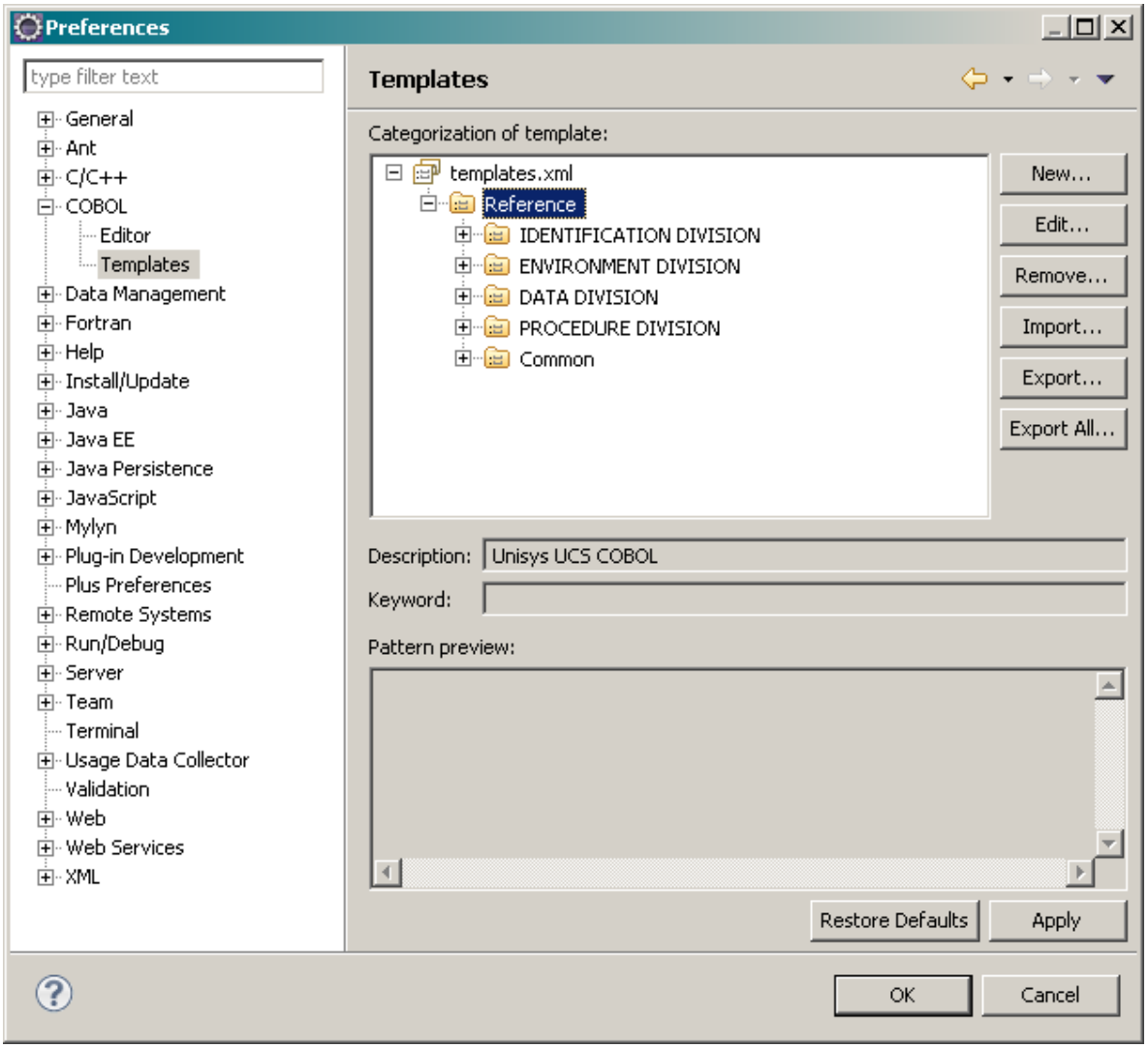
Using Templates

Eclipse can support the use of Templates to save coding. We have already seen how auto-completion can allow us to select the desired COBOL statement or command and then insert the code into our program source. Unisys has developed templates for UCOB but a client can define their own templates. For example, you might require a complete program template or you might want to define site standard code for database error handling etc.

If the following section, the creation, maintenance and sharing of templates is discussed. Note that this section only applies to using templates with the COBOL editor.

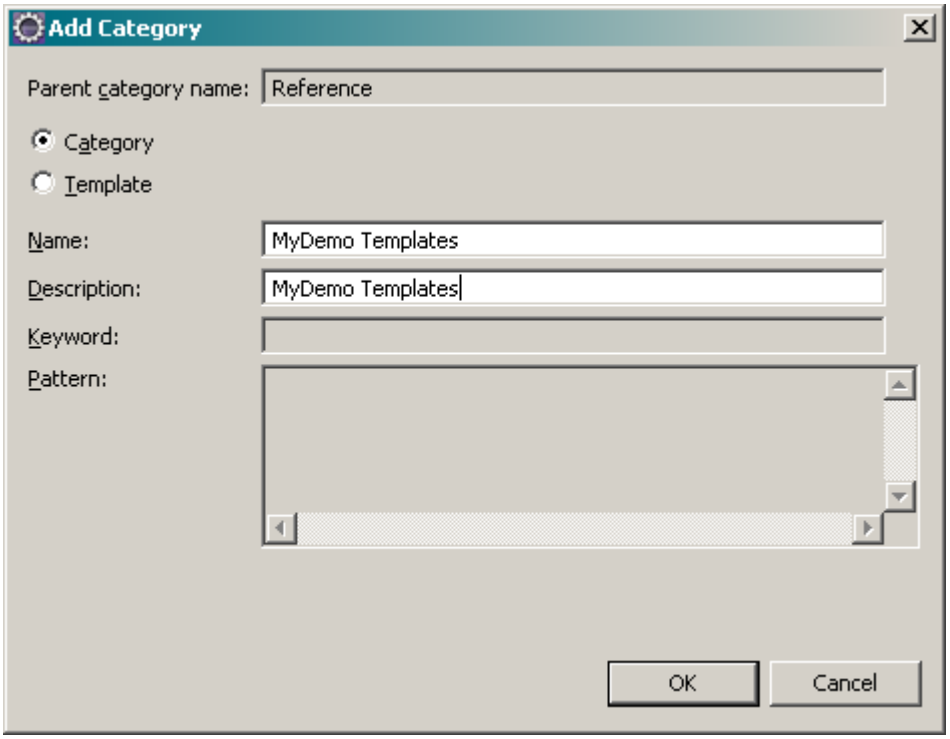
Creating Templates

To create your own templates, you need to go to **Window → Preferences** in the menu bar. Then example the COBOL entry and click on the Template entry.

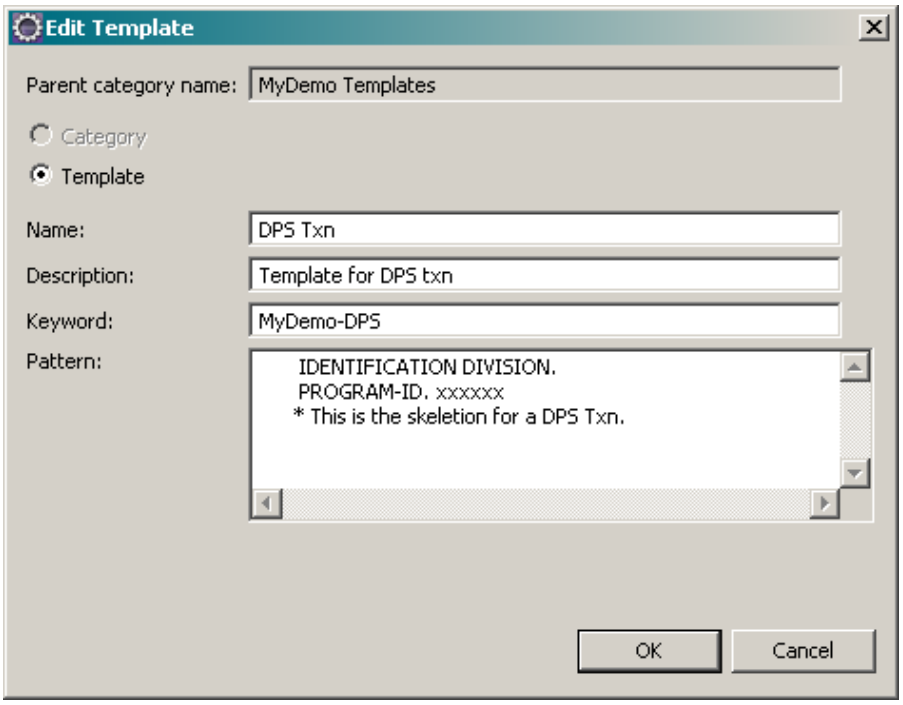


The template structure consists of categories and then templates. Categories can be nested. So for our case, let's create a new category under the Reference category. Highlight the Reference folder and then click "New".

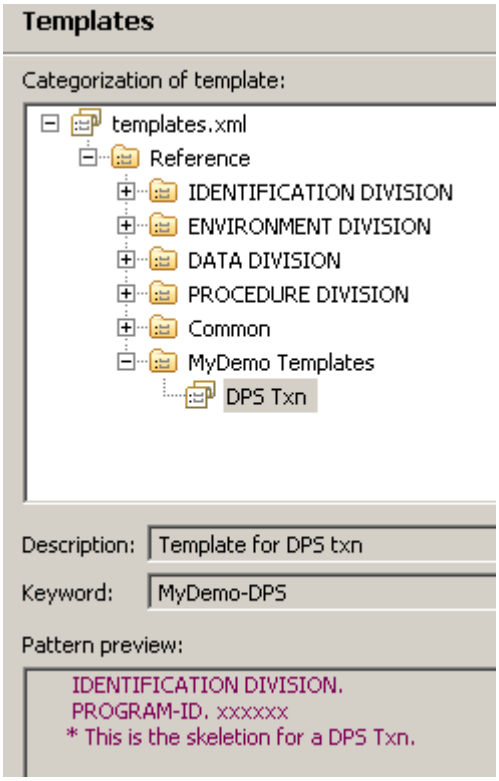
Select the Category radio button and then enter meaningful name and description. Click "OK".



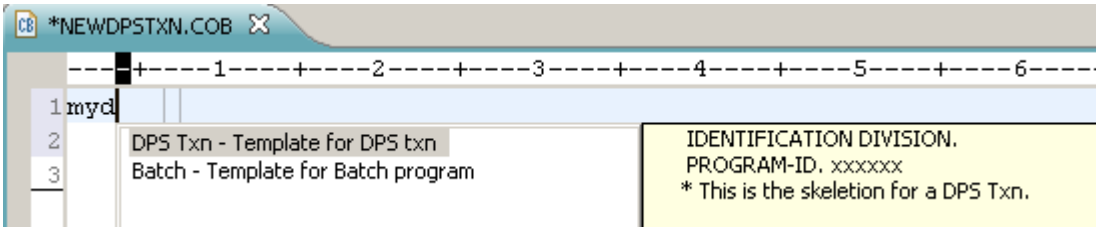
Now we need to create our template. With our category highlighted, click the New button. Enter a meaningful name and description. The most important field for the template is the Keyword. The Keyword is used by Auto-Completion (Ctrl + Space) to present entries matching the keyword (or the leading characters that have been entered.) So it is recommended to use a prefix on the keyword that easily identifies your local templates. In the following example, MyDemo- was used as the prefix. So if the user types "MyDemo" and then clicks Ctrl+Space for auto-completion, they will see all templates with a keyword beginning with MyDemo.



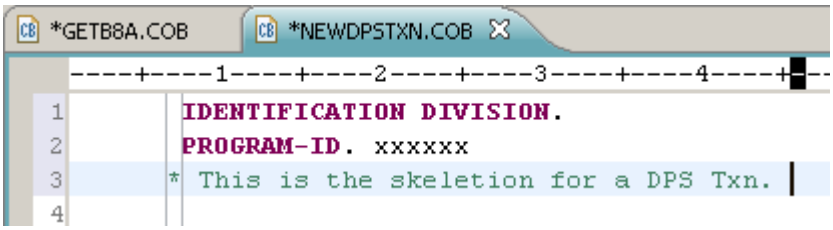
Click OK to save.



It is recommended to use a prefix on the keyword that easily identifies your local templates. In the following example, MyDemo- was used as the prefix. So if the user types "MyDemo" and then presses Ctrl+Space for auto-completion, they will see all templates with a keyword beginning with MyDemo as shown below. (We only need enough characters to match the category hence the demo shows "myd".)



So if we double-click on the DPS TXN entry, the template will be copied to our editor working pane as shown below.

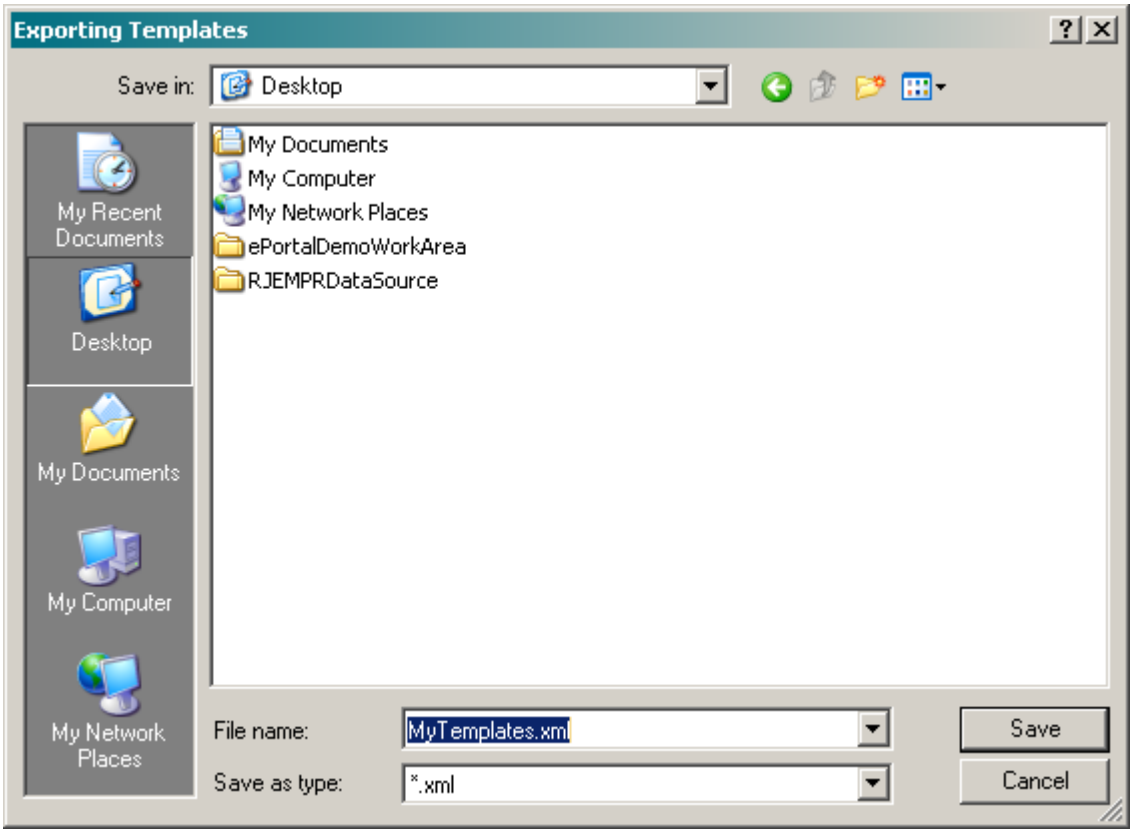


TIP: Turn off Auto Tag when using templates.

For this example, we also created a Batch template called Batch. The result shows how we have our 2 templates under our category of MyDemo Templates.

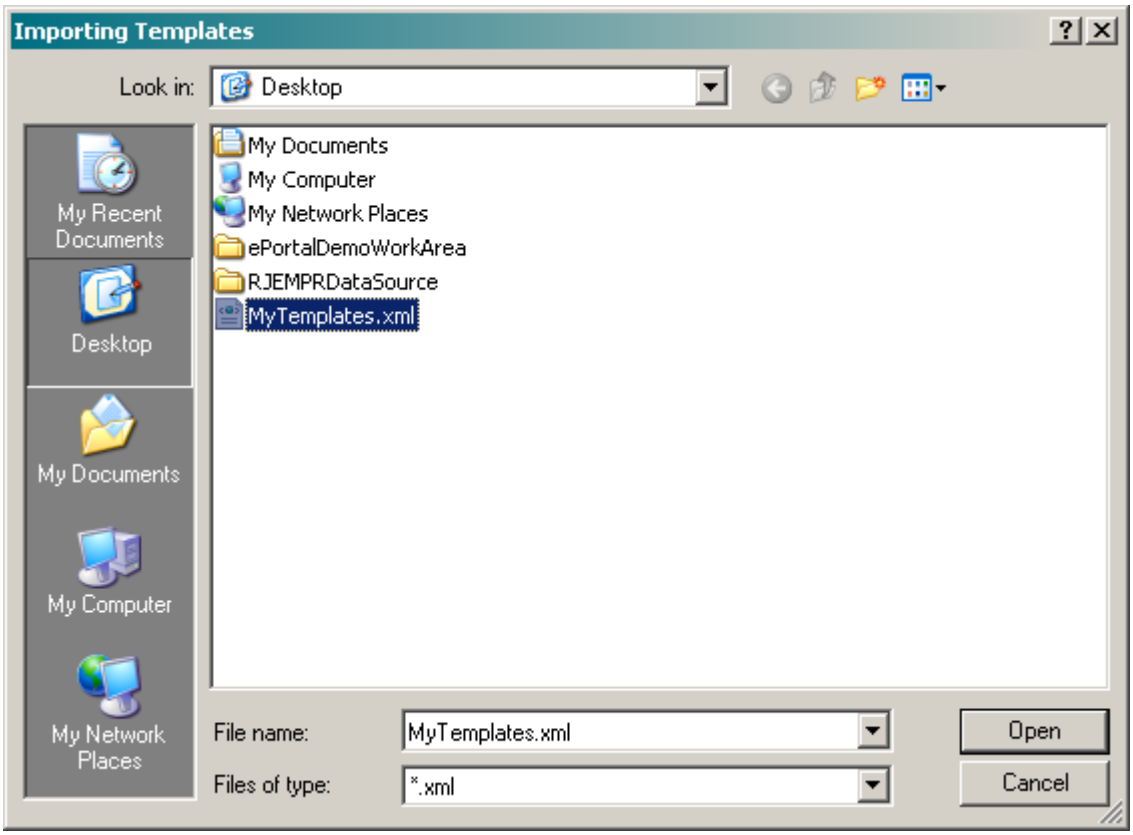
Maintaining and Sharing Templates

Now that we have defined our templates, we may need to share them with other users. Currently the templates only exist on the local workstation. Eclipse provides Export and Import functions to assist in maintaining your templates. (Refer to the previous screen and you will see the Export/Import buttons on the right side of the window.) The Export function will create a XML file in a directory that you select by specifying the location in the Save In field. You can export categories or templates.



To share with other users, this export file needs to be provided to the other users so they can import it. If you maintain an OS 2200 file that contains your Eclipse information (All-In-One file, manuals, JDK installation file etc) then this might be a good vehicle to make the file easily accessible to other users.

So the other users need to do an Import as shown below.



Templates and New Eclipse Releases

If you do use local templates, remember to export them from the current Eclipse environment when upgrading to a new Eclipse release. These local templates will not be in the Unisys Eclipse release files.

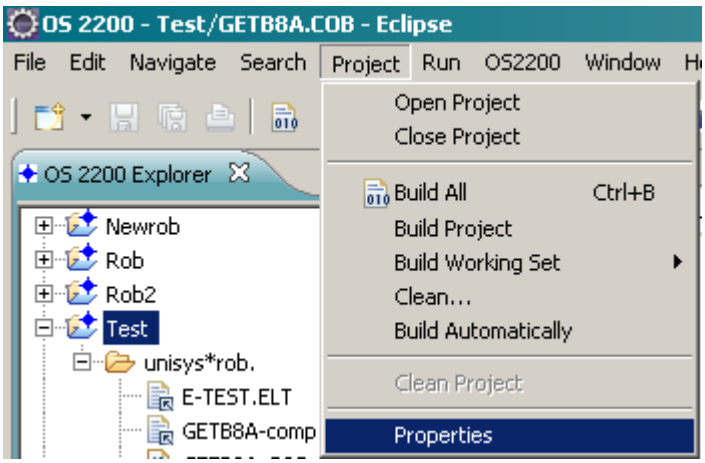
Building an OS 2200 Project

At this stage we have coded our program and now we need to build the project. When you build a project in Eclipse, a predefined set of ECL commands is sent to the OS 2200 host via a Telnet session. These commands could compile and link (MAP) one or more programs plus do other OS 2200 tasks like using SUPUR to update a TIP transaction library file.

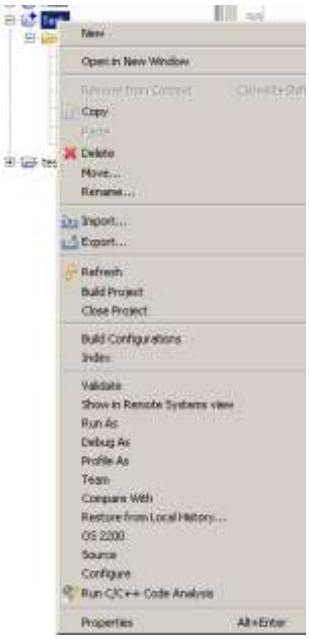
Configuring the Build and Brkpt properties

Earlier when we used the wizard to define a new project, we skipped the screen that requested Build and Breakpoint information. We will now update these properties of the project as it is the Build commands defined in these properties that are submitted to the OS 2200 host.

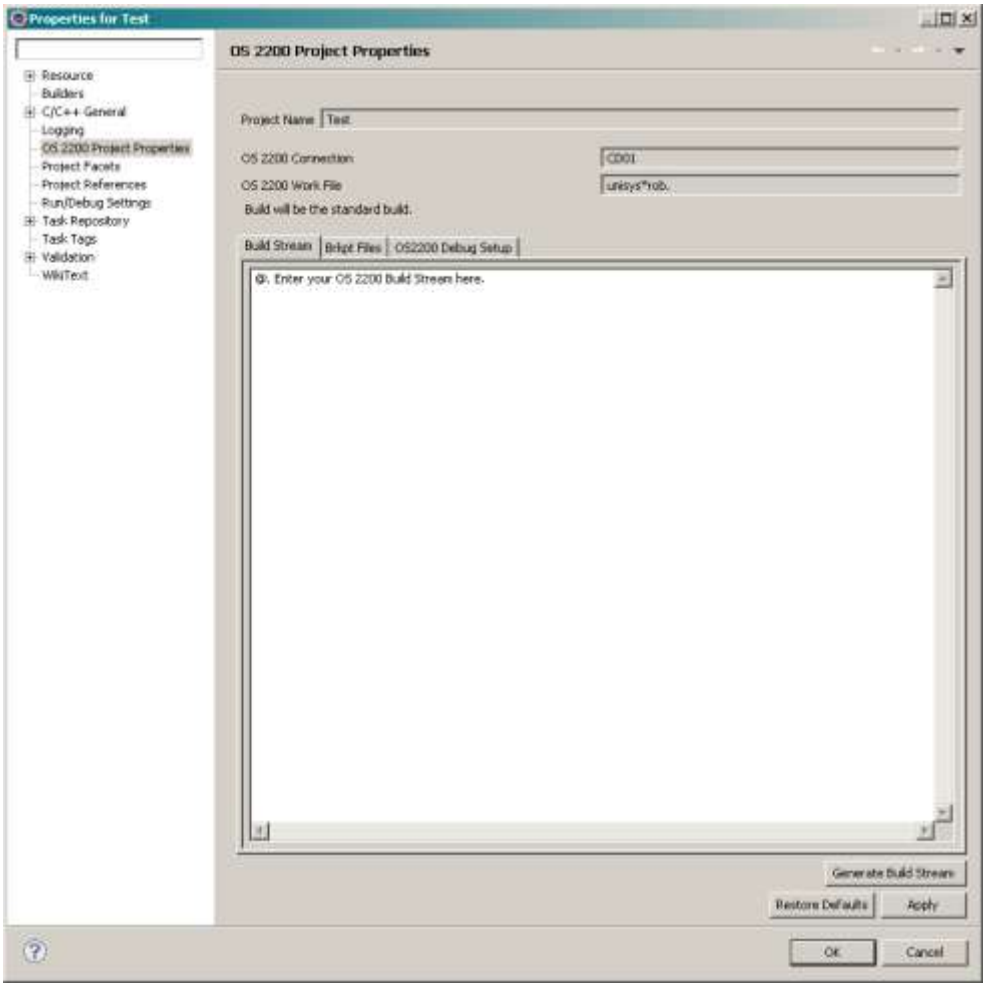
From the menu bar, go to **Project → Properties** and click.



Or right click on the project and select **Properties** from the displayed windows:



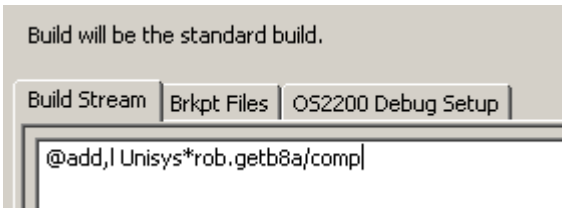
This results in the following screen being displayed for an OS 2200 project.



Configuring the Build Stream

Note that the Project Name, OS 2200 Connection and OS 2200 Work File are fields that are not updateable.

At the Build tab, we can enter various ECL statements that will be used to build our project based on what the programmer wants to do. In the above screen, there is the default entry. Below I have updated the Build Stream to @add an element. However you could have an @SSG call here, or compiler commands etc. Any valid ECL is allowed – just like you would do in a demand session.

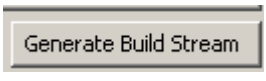


It is important to note the build type. The above screen shows this is a standard build as opposed to a debug build.

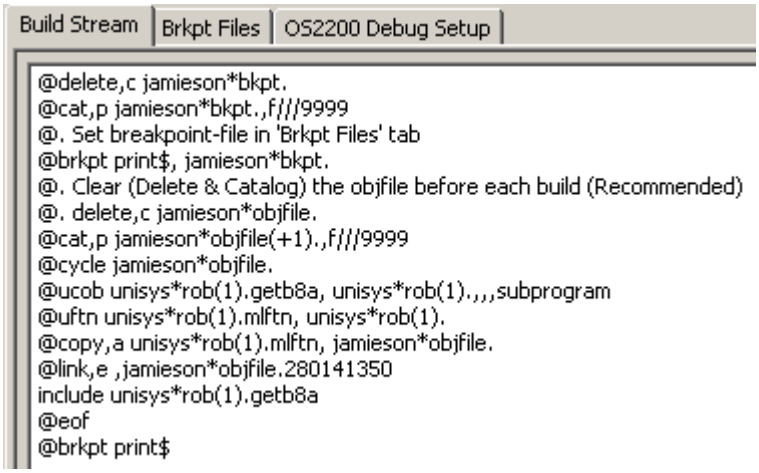
Note: If you want Eclipse to detect Error, Warning and Information messages as described later, you have to set the L option on an @ADD or other processor options that generate listings.

Creating a default Build Stream

A new option in Eclipse 3.7 can generate a Build Stream based on your project details. Click on the Generate Build Stream button at the bottom of the wizard:



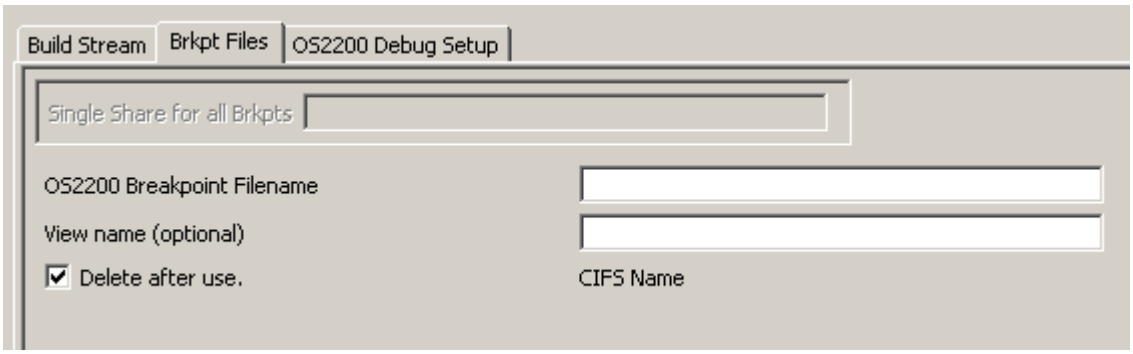
The result in my example is:



This might not meet your site standards but maybe a starting point.

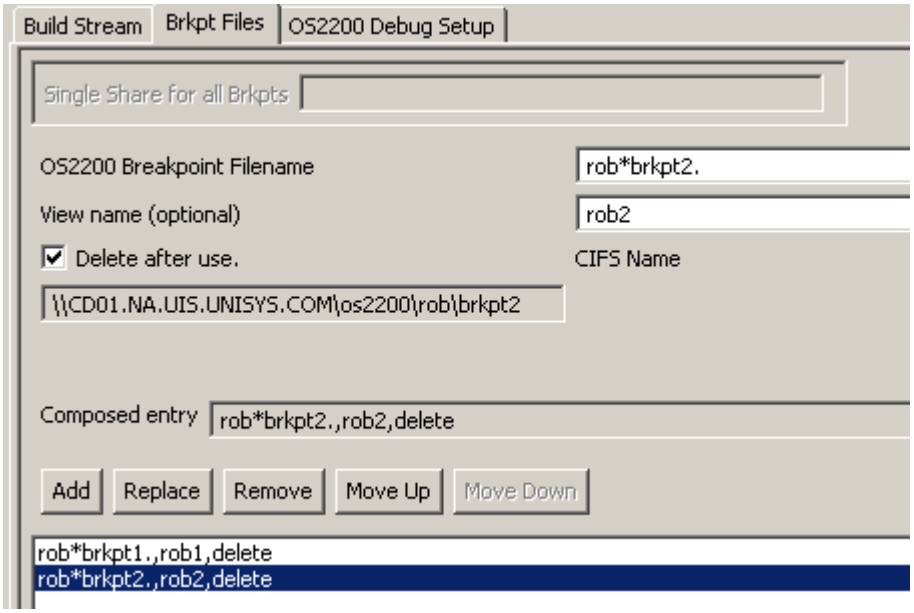
Configuring the Brkpt Files

Now go to the Brkpt Files tab that looks like the following screen:



If your Build Stream commands (or the elements they use) create a brkpt file, then you can enter the OS 2200 Breakpoint Filename here. (Actually it could be any valid OS 2200 data file.) Note that if you enter a View name, then Eclipse will open a pane in the diagnostic window with the contents of the Brkpt file at the end of the project build process. Be careful with the 'Delete after use.' check as the Brkpt file will be deleted if this is checked. Note that multiple Brkpt files can be defined.

Note that Eclipse does a timestamp check on the brkpt files and will only process and display the file if the last reference timestamp is after when the build was started.



By highlighting a brkpt file, you can modify the settings.

Configuring the OS2200 Debug Setup

Click on the OS2200 Debug Setup tab. Eclipse shows the setup information for a debug build. Debug builds are used with the Eclipse debugger module to provide an interactive debug session. Programs must be complied with a UCS compiler like UCOB and using the appropriate compiler options. DEBUG/FULL and NO-OPTIM are mandatory. The Eclipse PADS library must be installed on the 2200 host. Check with your system administration for the name of the installed Eclipse PADS library file – it is needed for debug builds.

Note that when the **Use Debug Build** checkbox is selected, the build type is changed to indicate this will be a debug build.

The **Debug Callback Id** is used when creating your debug session. Use a unique value but a good practice is to choose a value that matches your program to be debugged.

The **Debug Callback Port Number** is the port number used by PADS to send data to the Eclipse debug session running on a workstation. Generally use the default value.

The **Callback IP address** is the IP address of the workstation where the Eclipse debug session will be run. (PADS will send information to this port.) Generally it is your local workstation IP address.

The **MASM Element Name** is the name of the element in your Project file (OS 2200 work file) where Eclipse writes the MASM element required for PADS to interface with the Eclipse debugger. With Eclipse 3.7, different elements can be created. If you are debugging multiple programs in the same OS 2200 work file, then use a value to identify each program e.g. ABC/DEBUG for program ABC.

The **OS2200 Debug Library Name** is the name of the Eclipse PADS library file on the 2200.

Eclipse creates a set of LINK statements that must be used in the @LINK of the program. You can copy/paste these statements as required.

In the edit window, Eclipse creates a default compilation build stream that is required for a debug build. You can edit this as required. (Note that this build stream is used for a debug build and not the build stream under the Build Stream tab but only if the Use Debug Build box is checked.) It is necessary to call the MASM processor to generate an object module from the appropriate MASM element. The LINK statements are also essential.

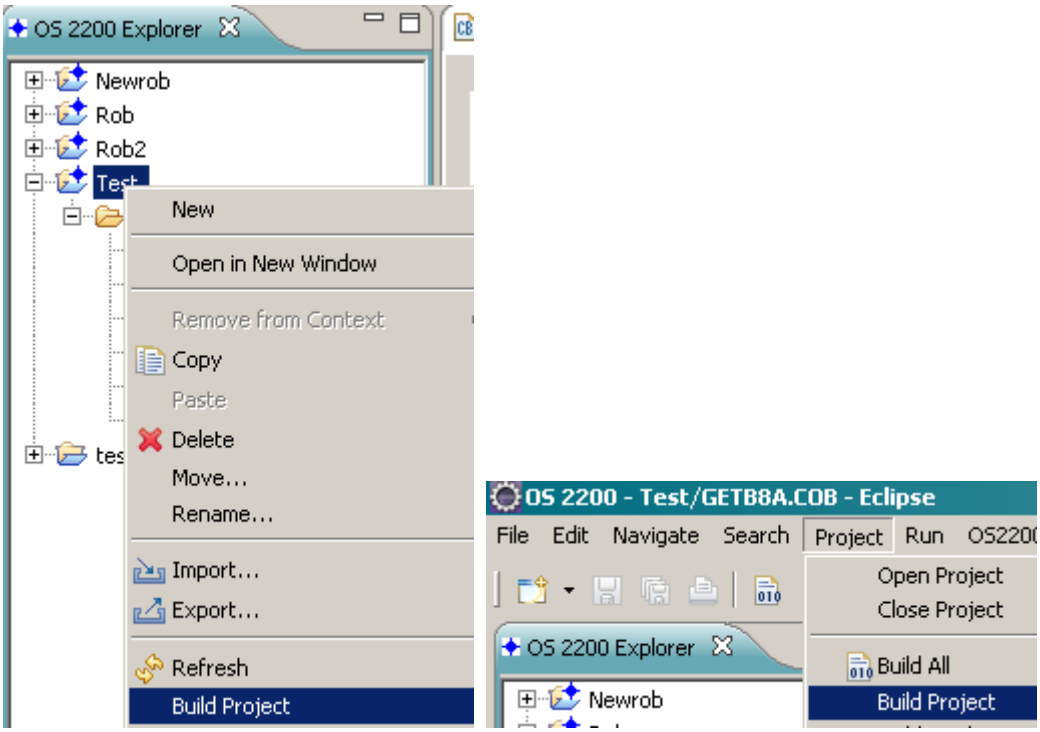
Before a UCS executable can be debugged:

- A special debug element must be created. This element contains information the PADS subsystem will use to call back to the PC at startup.
- The executable needs to be static linked to:
  - Include the special debug element;
  - Include a certain debug library OM;
  - Cause a special PADS entry point to be called at start time.

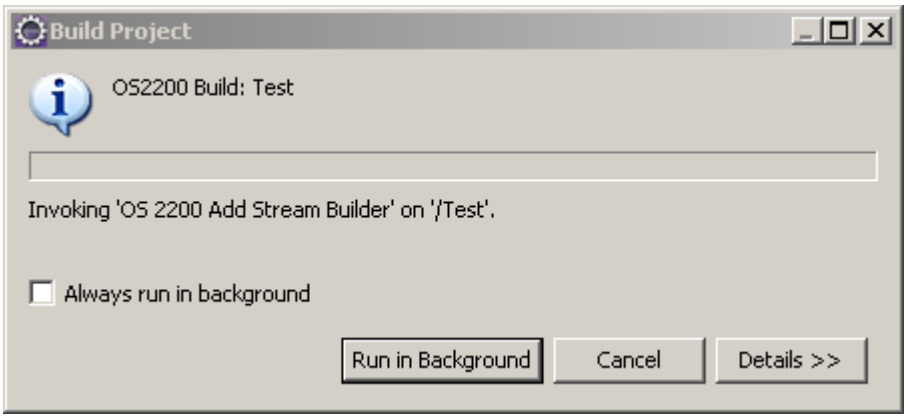
Refer to the section on UCOB Debugging for more details. For now, leave the Use Debug Build unchecked.

Doing the Project Build

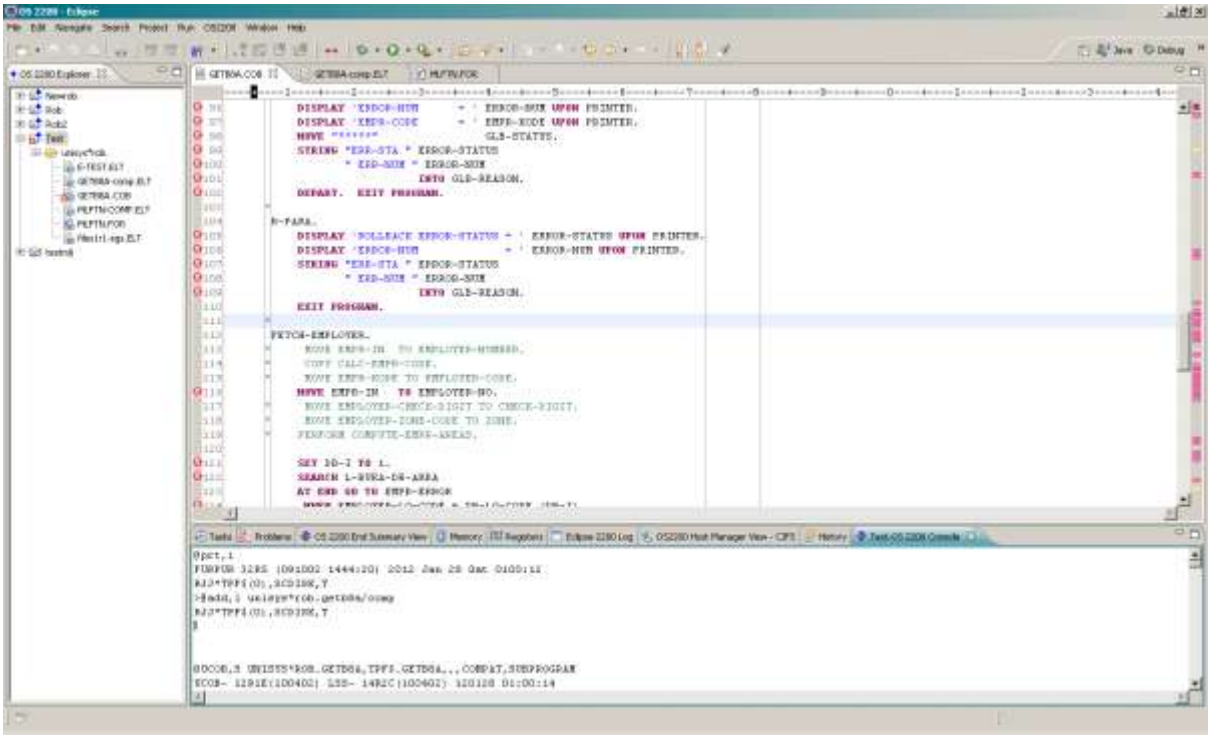
Now that we have defined the build commands, we can perform the project build. Go to the Explorer pane and right-click on the project name. Or go to the menu **Project → Build Project.**



Click the **Build Project** entry. Eclipse will pop-up a dialog with the build status.

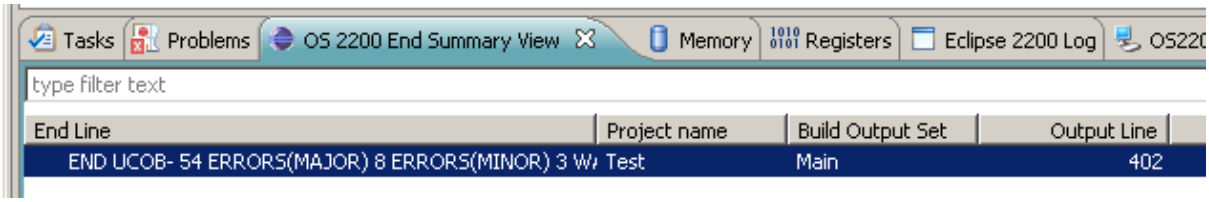


You will notice that Eclipse has opened a new pane called the OS 2200 Console. After the Build Project process has finished, this window appears at the bottom of the workbench as a tab in the diagnostic window.



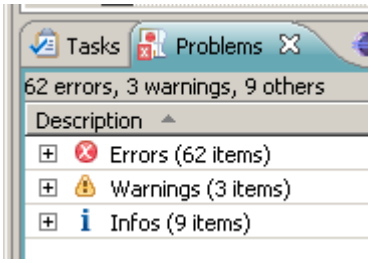
As you can see, this OS 2200 Console pane shows the results of the build process. Firstly Eclipse has performed a @PRT,I and then submits the Build Stream ECL that we defined earlier. Note that if a Brkpt file was defined and it had a View name, it would appear as a pane next to the OS 2200 Console. You can enlarge this pane or use the scroll bars to look at the output. However Eclipse does check the OS 2200 Console and any Brkpt files for you looking for Errors, Warnings and Informational messages.

There is a pane titled “OS 2200 End Summary View” that contains the output for all lines beginning with END. (In a demand session using @ED, you may have issued FC END to get this info.) You can quickly check if any errors were reported.

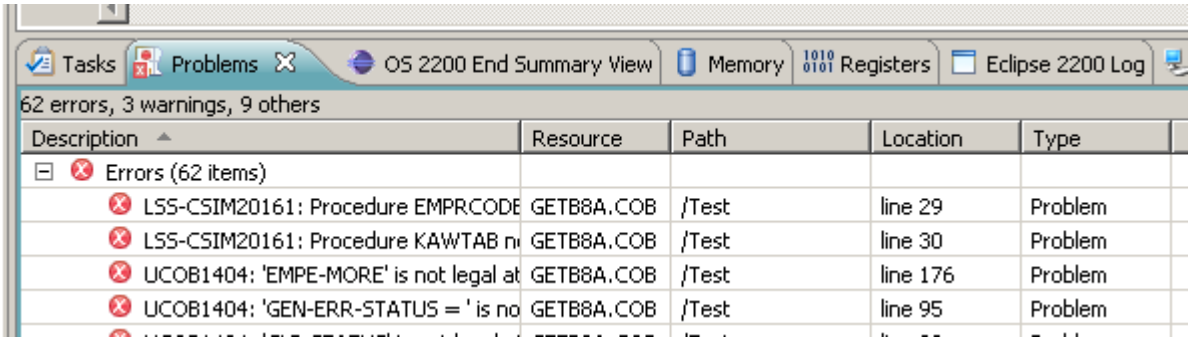


Note that OS 2200 End Summary View contains the output from many project builds. This can lead to confusion so you can clear this view by clicking the following icon on the right side of the pane title:

By clicking on the Problems tab, you will see a summary of the results of these checks.

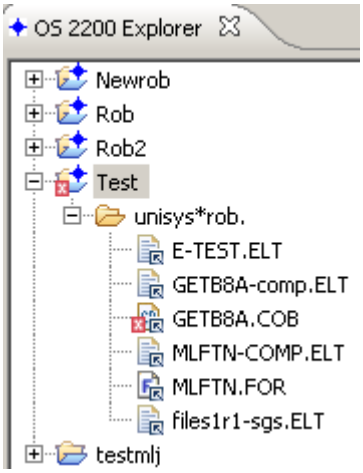


You can expand each type of error by clicking on the ‘+’ sign.

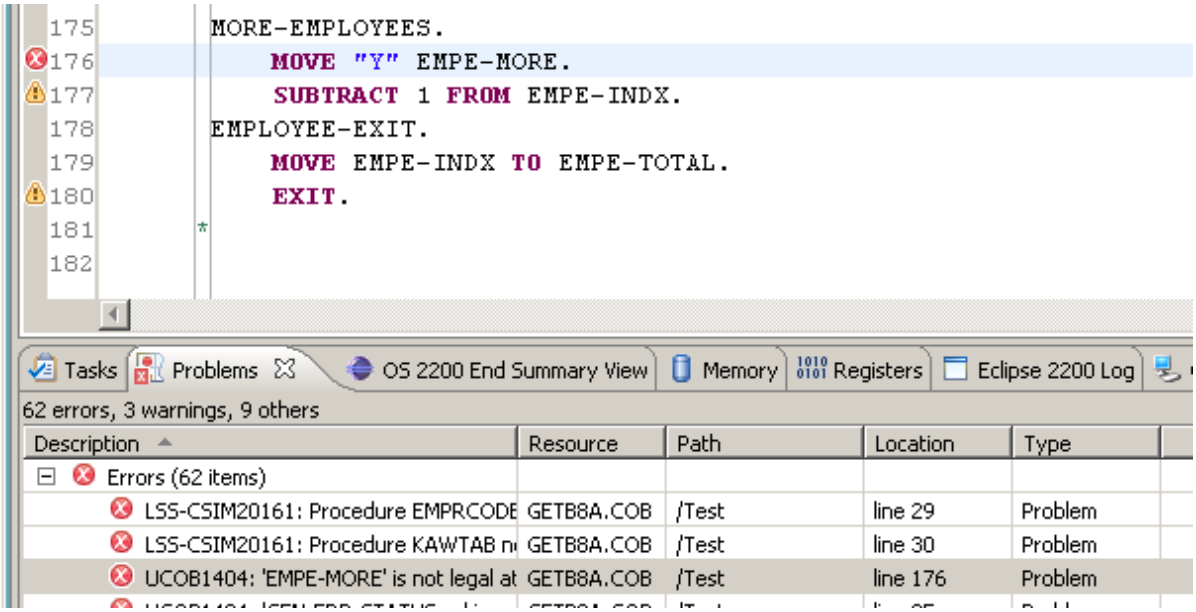


Eclipse prints the error/warning message, the project file with the problem and the source line number.

The navigation pane will also indicate the project files with errors by placing the same error icon on the name. A programmer can easily see which files have errors.



As a programmer, you would want to go to that line to correct the problem. By doubling-clicking on the problem line, Eclipse will open the element in question and position the cursor on the line in error.



In the above case, the problem entry for line 176 was double clicked and the COBOL editor opened the GETB8A file and positioned the editor at line 176. Notice the editor also indicates lines with problems – different icons are used for errors, warning and informational.

Interactive Debug for UCOB

Follow the procedures described earlier for performing a debug build for the project.

Perform the Debug Build

The UCS interactive debugger operates by causing the OM or ZOOM to call back to the PC doing the debugging during the normal run of the OM or ZOOM. To accomplish this action the executable must be static linked to include certain OMs and cause PADS to be invoked through a special entry point. This is accomplished by the debug build. The OM or ZOOM is acting as a TCP/IP client and the PC as the listener. It is necessary then that the 2200 be configured to allow calling out and that firewalls do not prevent a PC from receiving connection requests from the 2200.

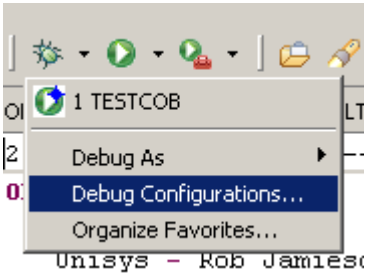
Debug Build Best Practice

The OM or ZOOM output with the debug settings should be placed in a different file to where the UCOB source program is located. If the source and OM/ZOOM are in the same file when the debug session is run, problems can occur.

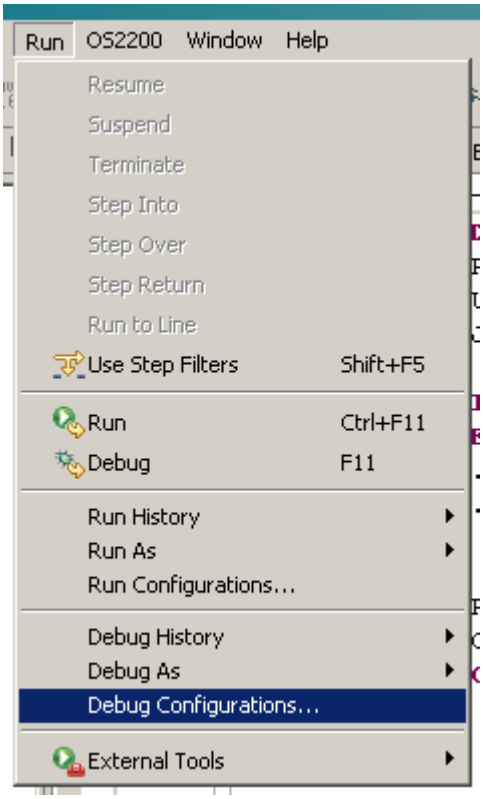
Defining a Debug Configuration

There are a number of ways to initiate the debug session but first we need to define a debug configuration. One the main icon, click on the debug 'bug'.

Then select Debug Configurations.

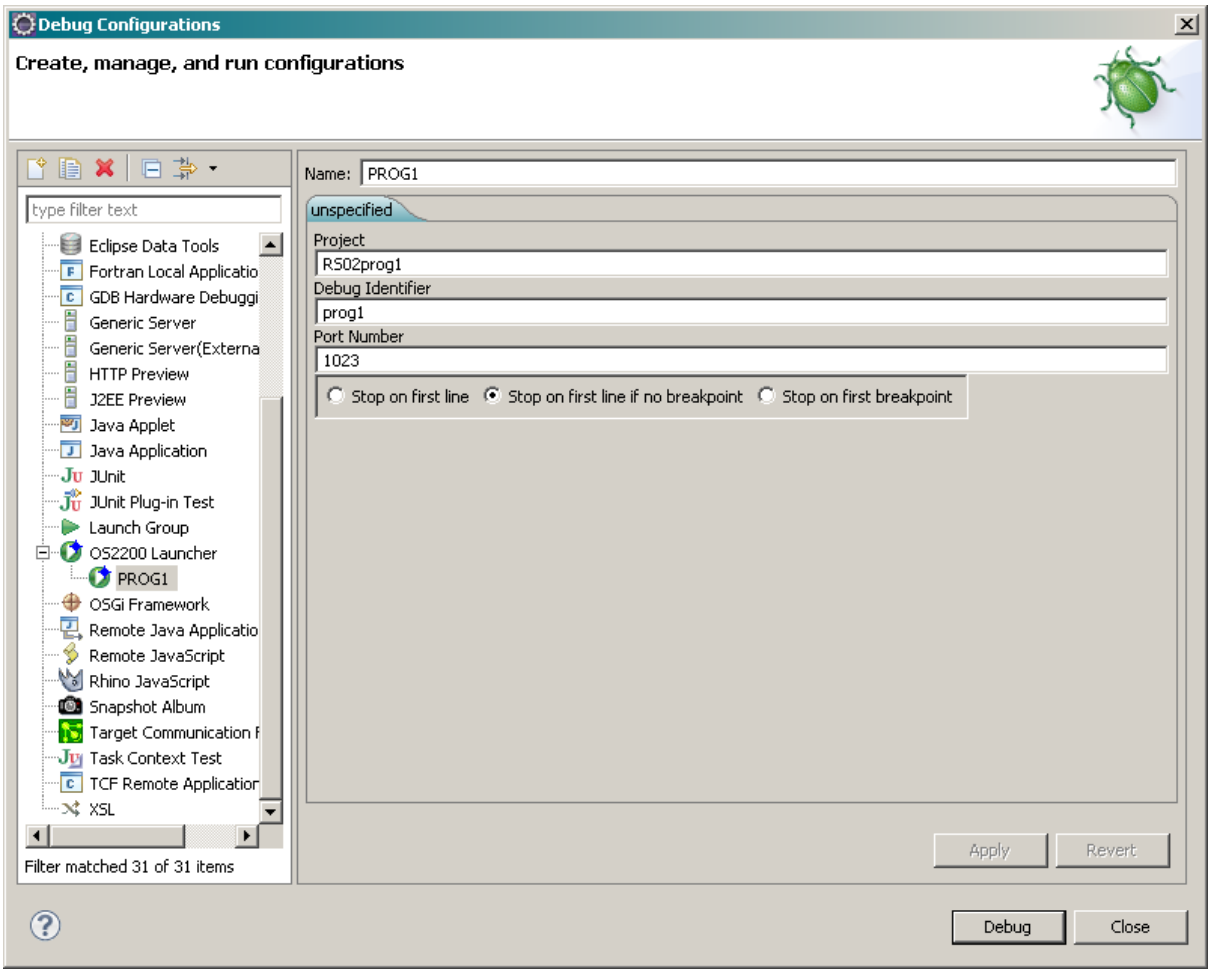


Or go to the menu **Run → Debug Configurations.**



The wizard displays a dialog to be filled in with information for this debug configuration. Multiple configurations can be defined.

Click the OS2200 Launcher option.

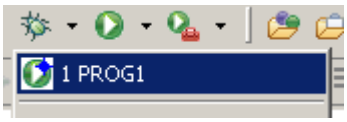


Define a meaningful name for this configuration.

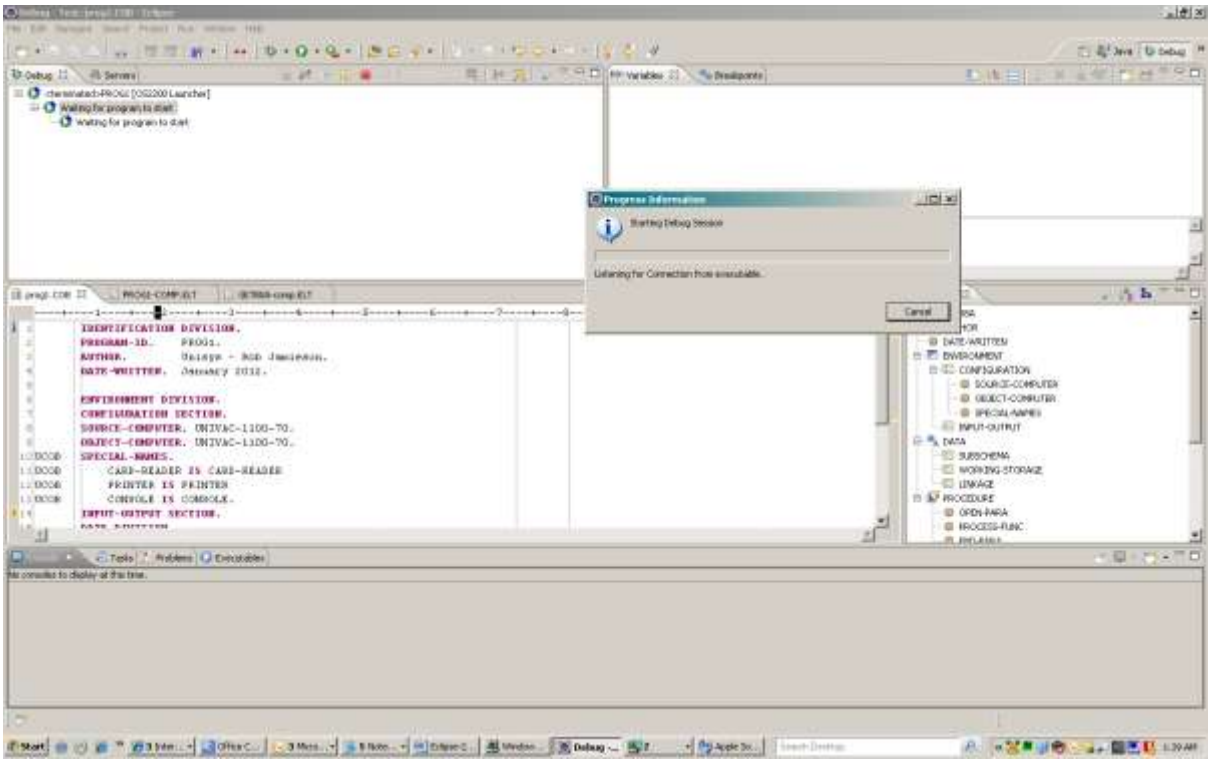
The Project, Debug Identifier and Port Number should match the information used in the project debug build stream.

Running a Debug Session

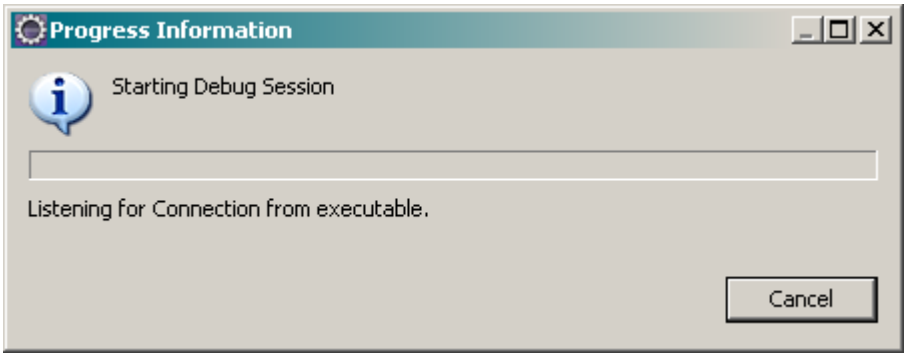
Click on the ‘bug’ and select the debug configuration to be used.



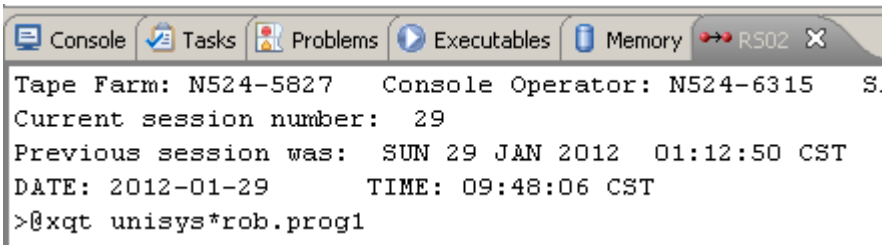
Eclipse will open the Debug Perspective.



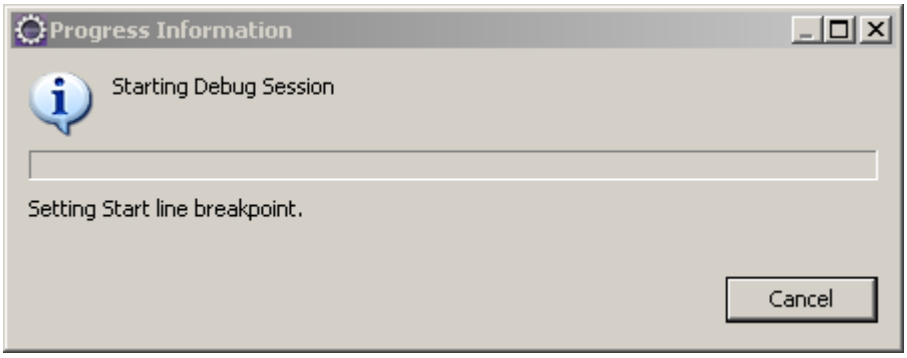
Then following dialog is displayed.



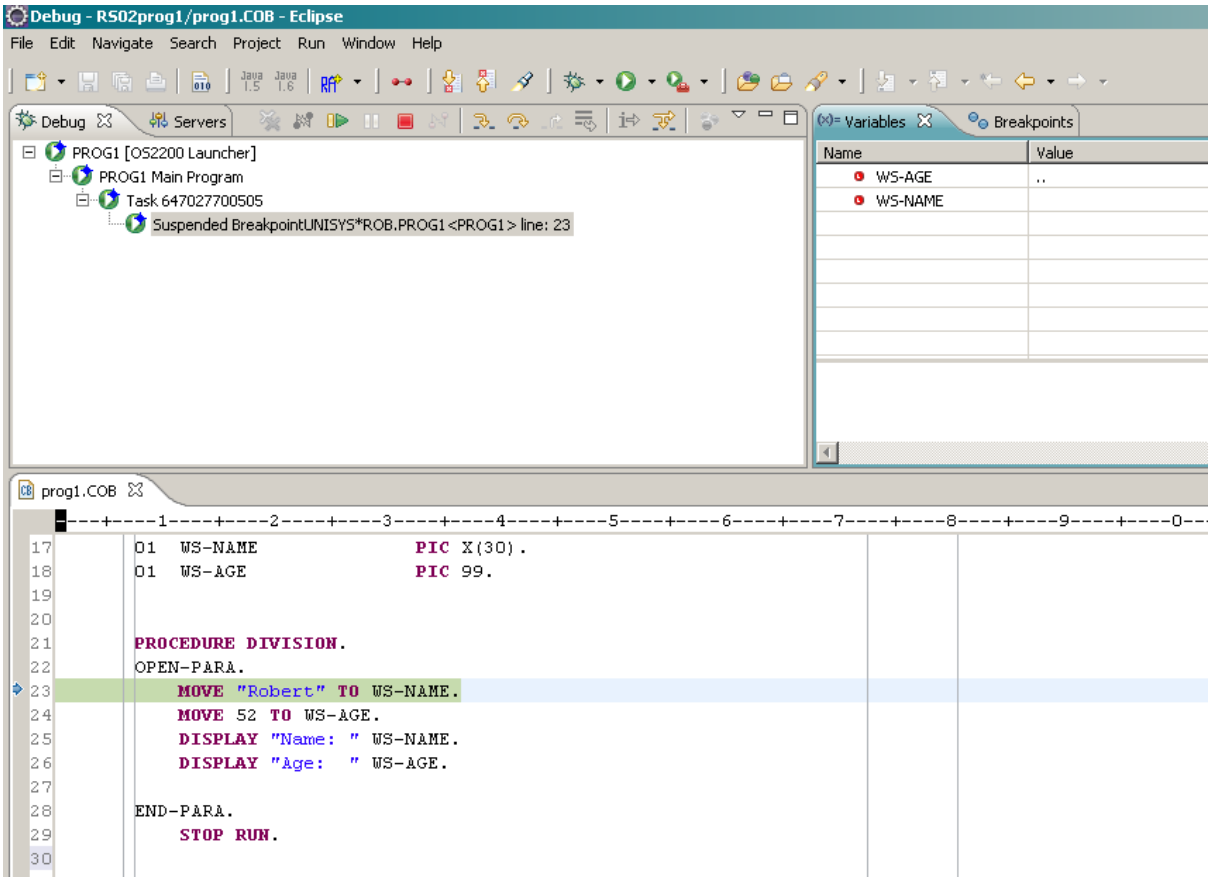
At this stage you need to execute the debug OM/ZOOM on the 2200. For demand or batch jobs this can be done from the Eclipse Telnet client. For full-screen programs or transactions it must be done from an emulator.



The 2200 program will communicate with Eclipse. Various messages are displayed in the dialog indicating that information is being downloaded from the 2200 to the Debug session.



Eventually the Debug session will pause with the source code in a pane and some variables listed in the Variables view.

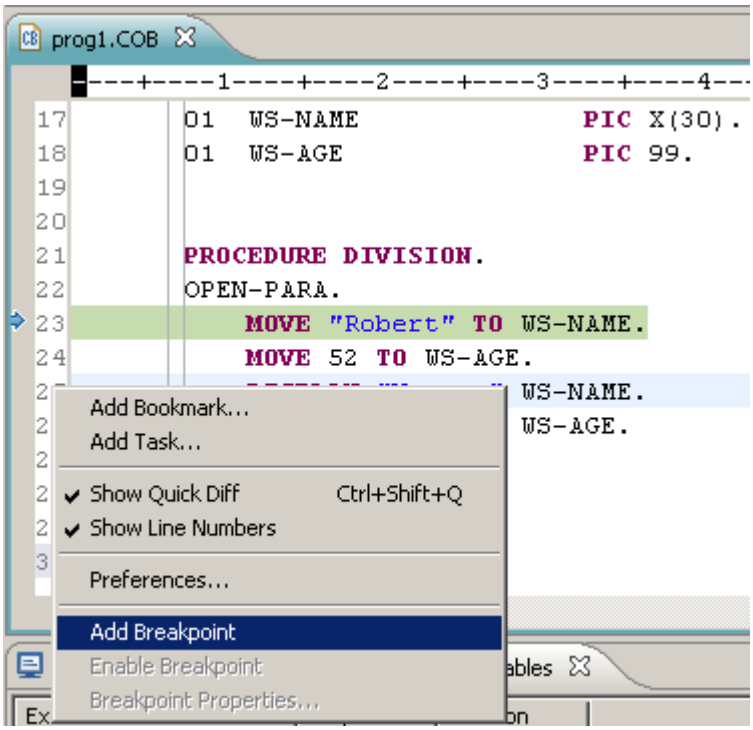


NOTE: The time it takes to call back can vary. A demand program over a fast connection generally calls back almost immediately. A transaction can take over two minutes. The speed of the connection to the 2200 effects the callback time.

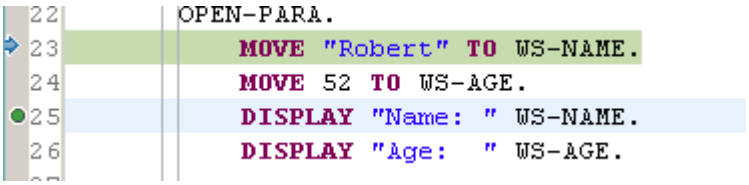
The upper left of the debug perspective shows the stack. Below that is the code which is the current execution point. In the upper right is a pane which holds the variables, breakpoints, registers, and

watch views. The variables view displays all local and global variables in the selected stack frame. The registers are global, that is, registers to not change from frame to frame.

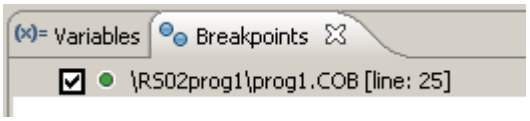
Breakpoints can be added to source by going to the required line and doing a right click.



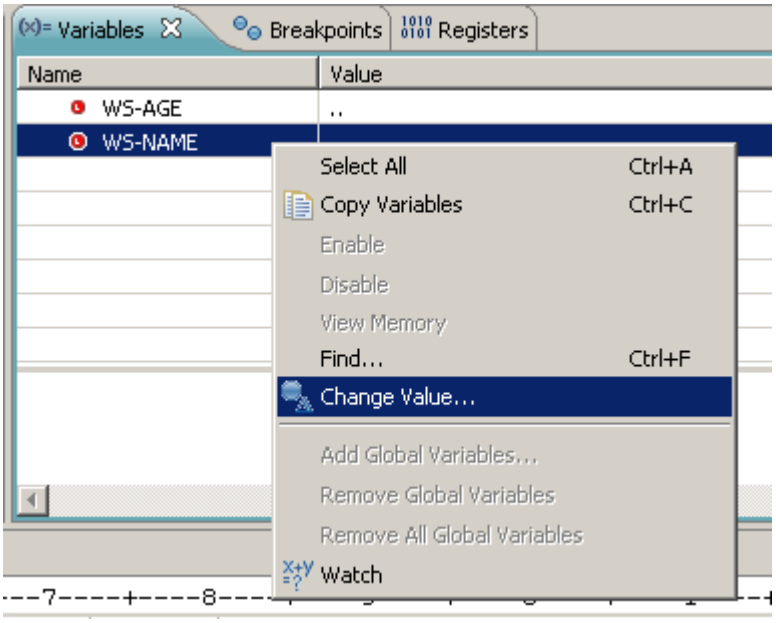
Selecting Add Breakpoint marks the line with an indicator.



An entry is also written to the Breakpoints view.



The Variables view lists the variables defined in your source. You can change the value of a variable by highlighting the variable and doing a right click.



There are a number of ways of stepping thru the logic. From the debug view one can "resume", 'step into', 'step over', or 'step out'.

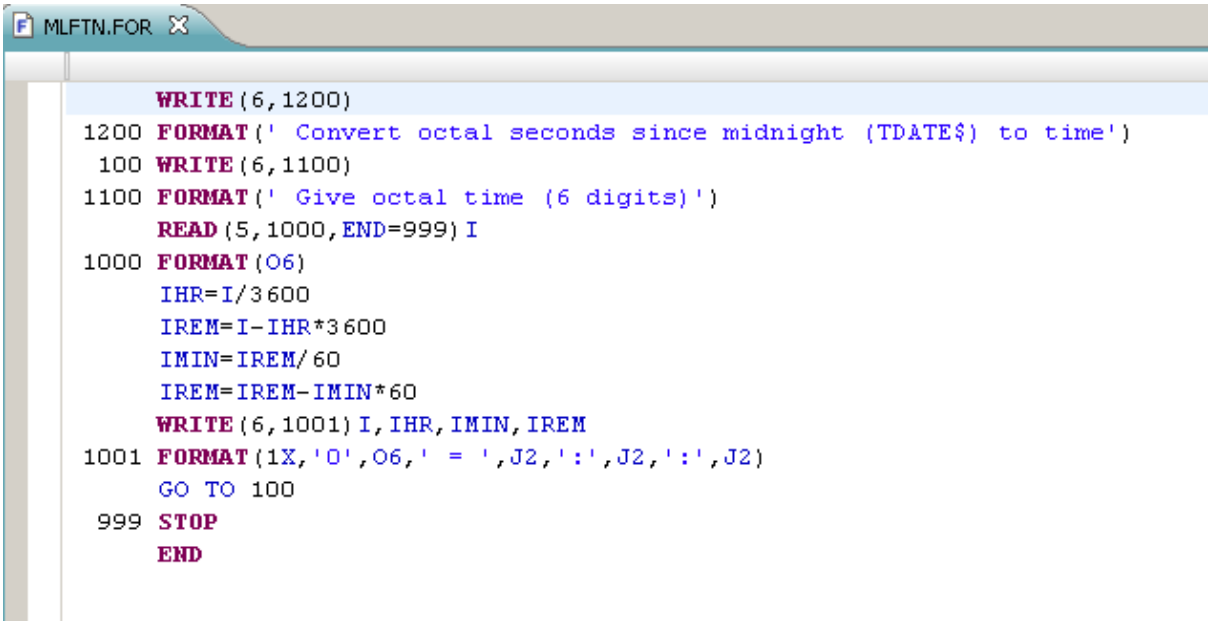
- 'Resume' proceeds until a breakpoint is hit or the program exits.
- 'Step into' goes to the next executable statement whether it is in the same subroutine or a different subroutine.
- 'Step over' goes to the new executable statement in the current subroutine or the calling routine. If the next statement is in a subroutine be called by this routine, it will perform the subroutine and return, unless there is a breakpoint in the called routine.
- 'Step out' will finish executing the current subroutine and stop in the next line in the caller unless a breakpoint is hit along the way.

Other 3GL Editors

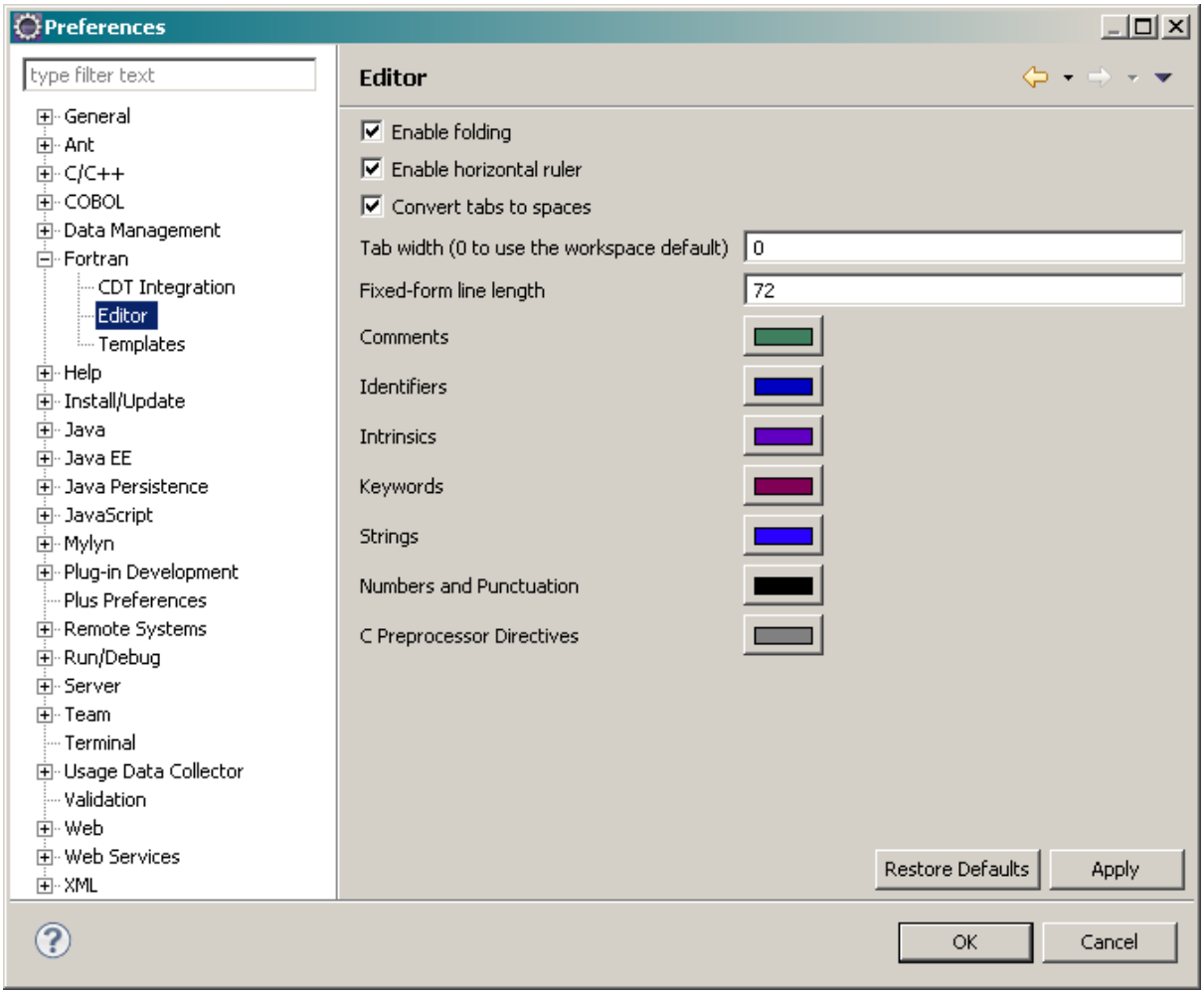
Eclipse can support many types of editors that maybe used for OS 2200 and other development. This section describes some of the other editors.

Fortran Editor

The Fortran editor is invoked if an OS 2200 element is given an editor type of FOR. Unisys does not provide a Fortran Editor plug-in like we do for COBOL but instead uses the Photran editor available from the Eclipse community.

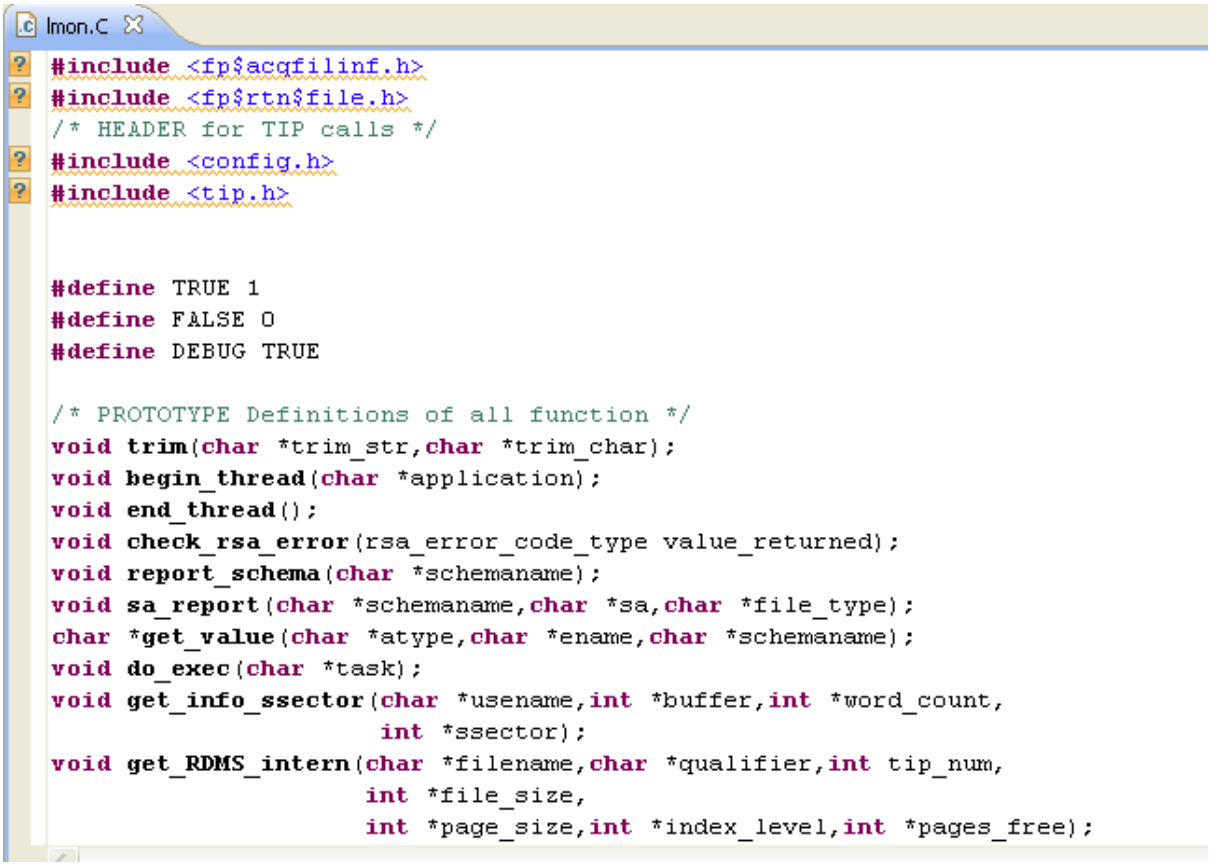


When you open the element, similar features as the COBOL editor in terms of colour coding are available. The editor preferences can be found from the menu **Window → Preferences → Fortran**.

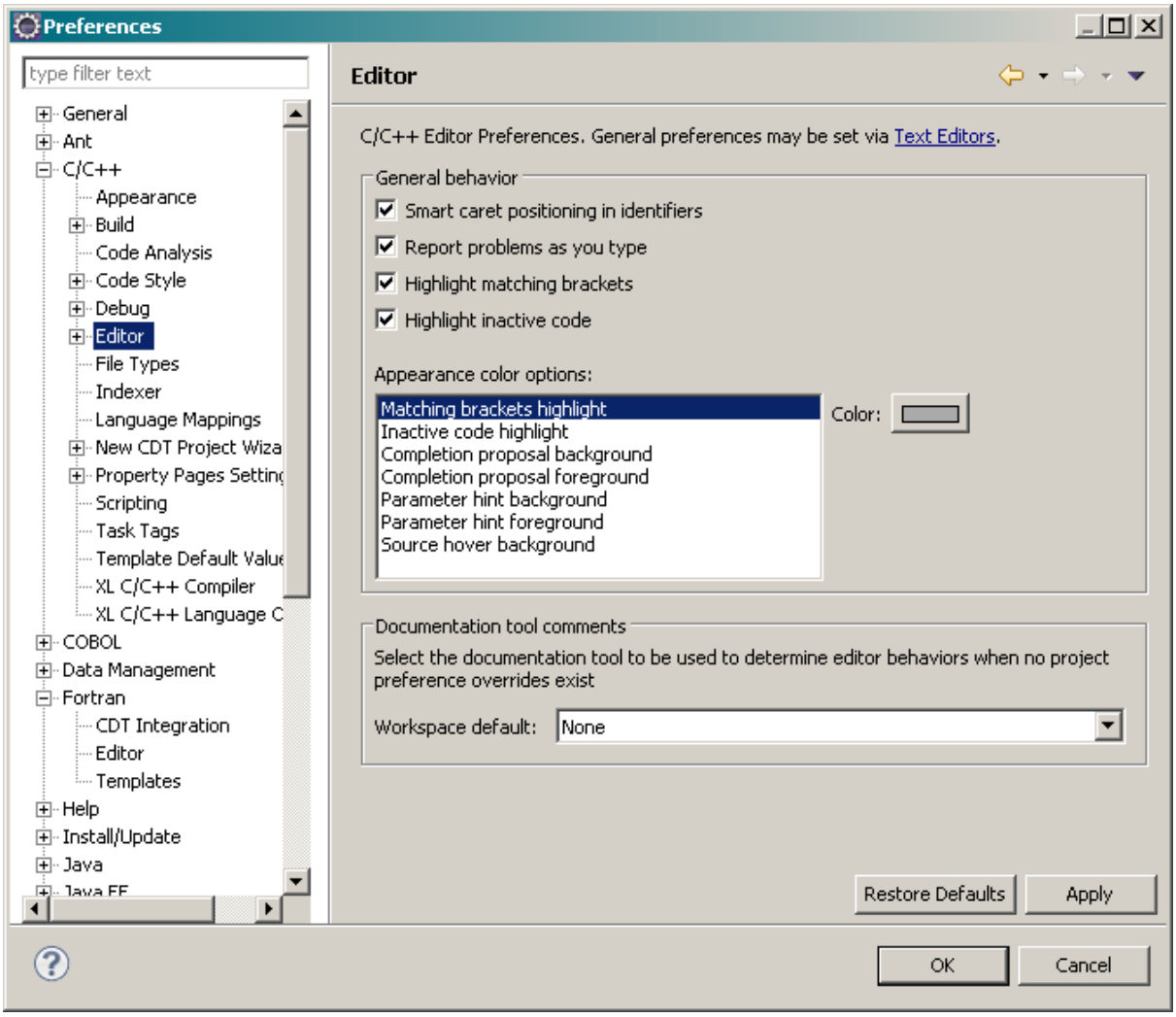


C Editor

The C editor is invoked if an OS 2200 element is given an editor type of C. Unisys does not provide a C Editor plug-in like we do for COBOL but instead uses the C editor available from the Eclipse community.

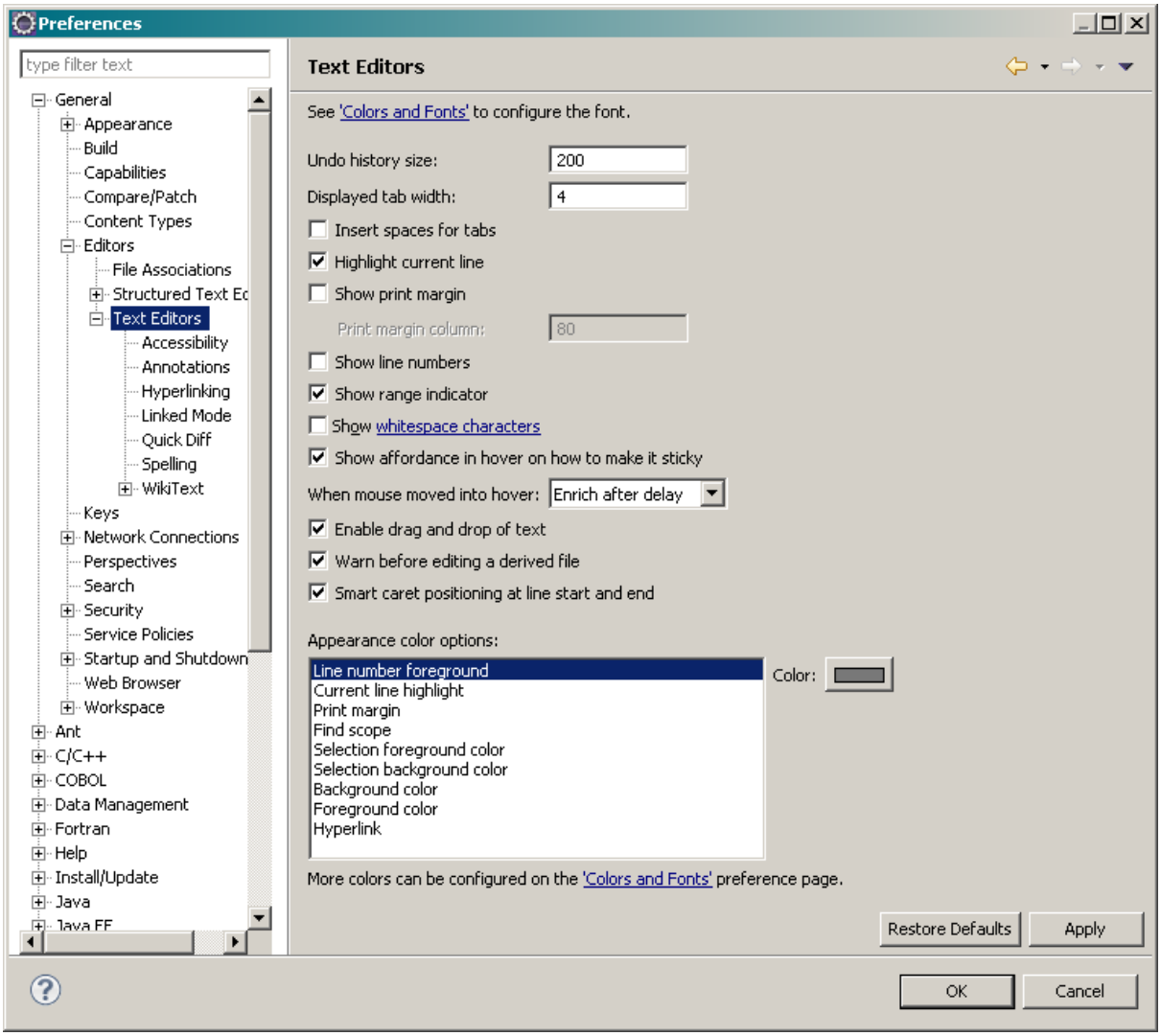


When you open the element, similar features as the COBOL editor in terms of colour coding are available. The editor preferences can be found from the menu **Window → Preferences → C/C++**.



General Text Editor

This editor is used for files with the ELT subtype etc. This is just a basic text editor with no special features. The preferences can be updated by **Window → Preferences → General → Editors → Text Editors.**

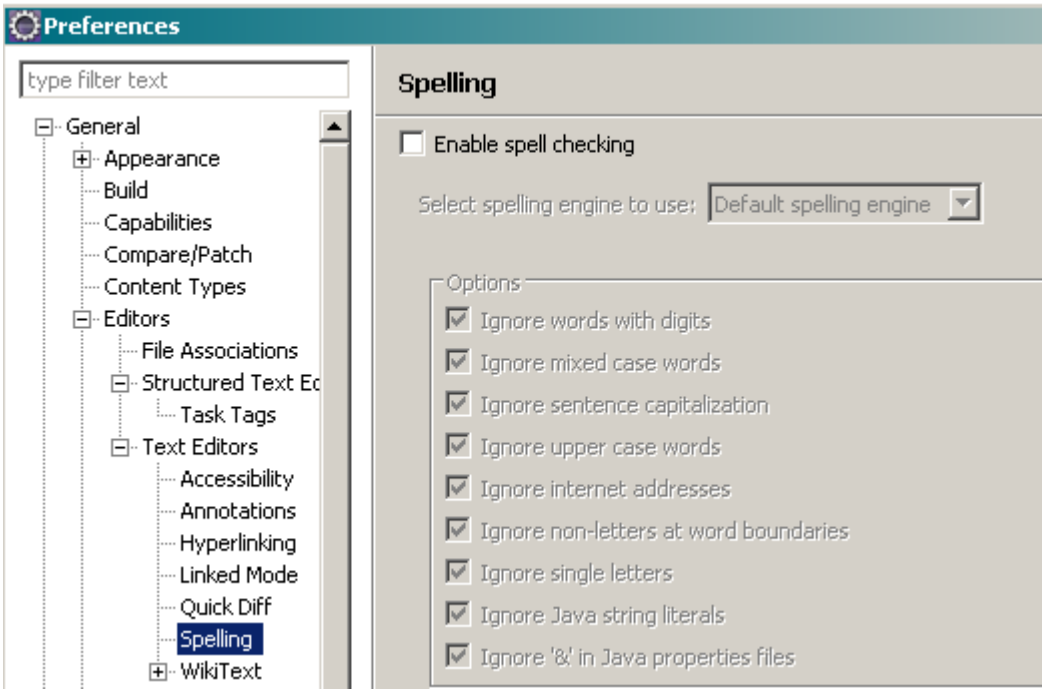


Turning Off the Spell Checker

By default, Eclipse turns on spell checking. When editing an ECL, SSG or similar element, many words are underlined in red as they are not matched by the spell checker. For OS 2200 elements, this often applies so readability is affected. Also when you copy the contents and paste to Microsoft Word, the highlighted words are inserted with underlines!

It is easy to turn off the spell checking function.

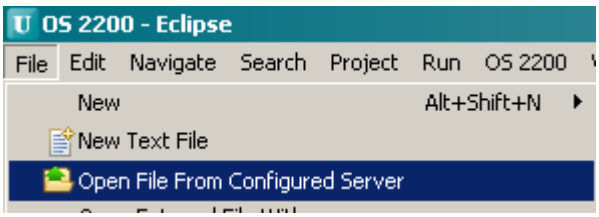
Go to **Window → Preferences → Editors → Text-Editors → Spelling** and uncheck ‘’Enable Spelling Check’.


Preferences
type filter text
General
Appearance
Build
Capabilities
Compare/Patch
Content Types
Editors
File Associations
Structured Text Ed
Task Tags
Text Editors
Accessibility
Annotations
Hyperlinking
Linked Mode
Quick Diff
Spelling
WikiText
Spelling
Enable spell checking
Select spelling engine to use:
Default spelling engine
Options
Ignore words with digits
Ignore mixed case words
Ignore sentence capitalization
Ignore upper case words
Ignore internet addresses
Ignore non-letters at word boundaries
Ignore single letters
Ignore Java string literals
Ignore '&' in Java properties files

Open File for Configured Server (OFCS)

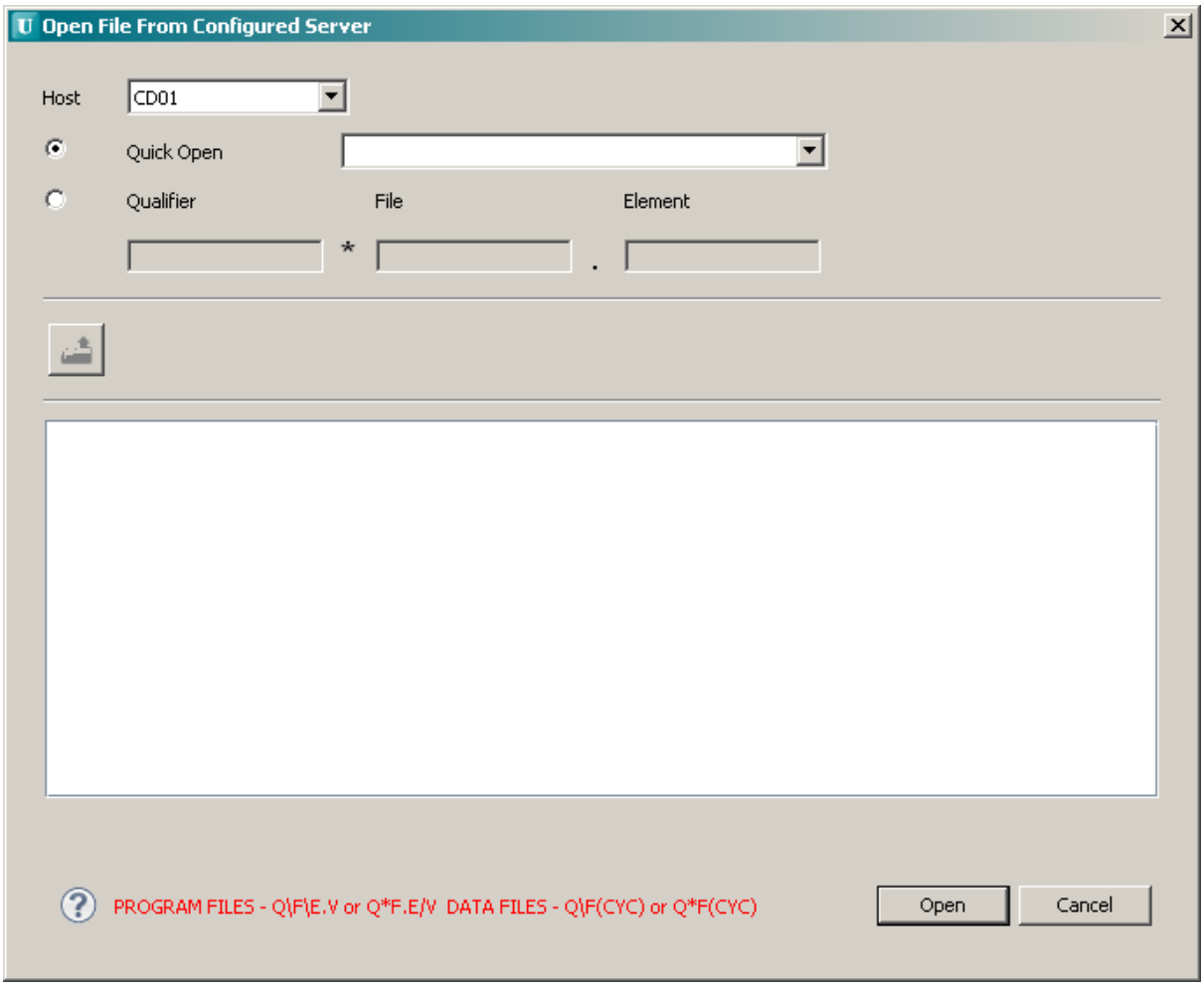
Often an OS 2200 developer will need to review the contents of a data file or perhaps need to review source elements quickly without the use of a project, Eclipse has the Open File for Configured Server (OFCS) feature. As this feature is outside of the planned use of Eclipse as an IDE using projects, some other features may not be available.

OFCS can be launched by going to **Menu -> File -> Open File From Configured Server.**



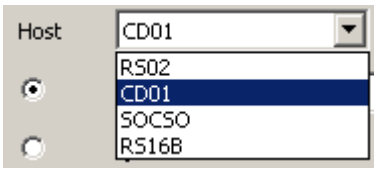
An alternative method is to click the OFCS icon  in the icon menu bar.

OFCS will display the following wizard.

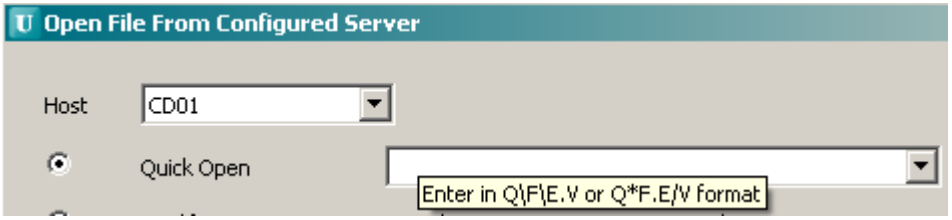


The wizard allows for the selection of the host as well as selecting either a Quick Open or browse mode for selecting the file.

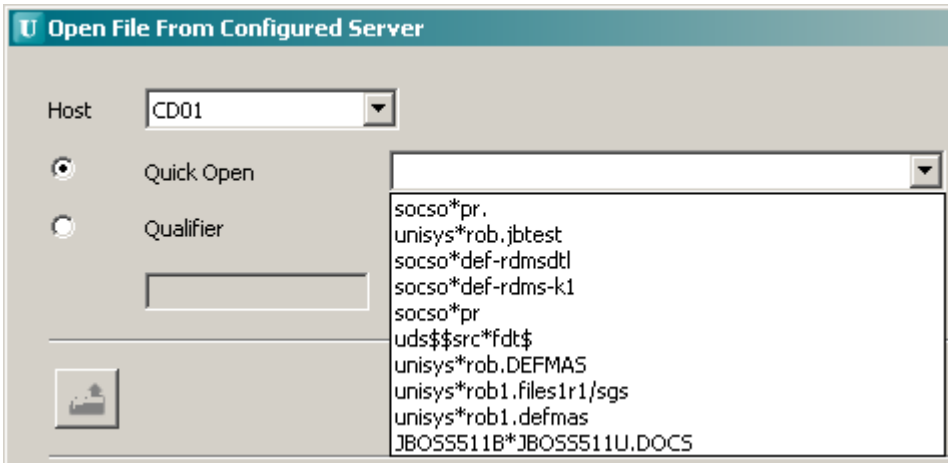
The Host field has a list box of all the configured hosts for the workspace.



The Quick Open method is selected by checking the radio button. The text box allows for the entry of a valid OS 2200 filename in either standard OS 2200 format (i.e. Q*F.E/V) or in Posix format (i.e. Q\F\E.V). Note that either a source element or a data file can be entered. Cycle information can also be used.



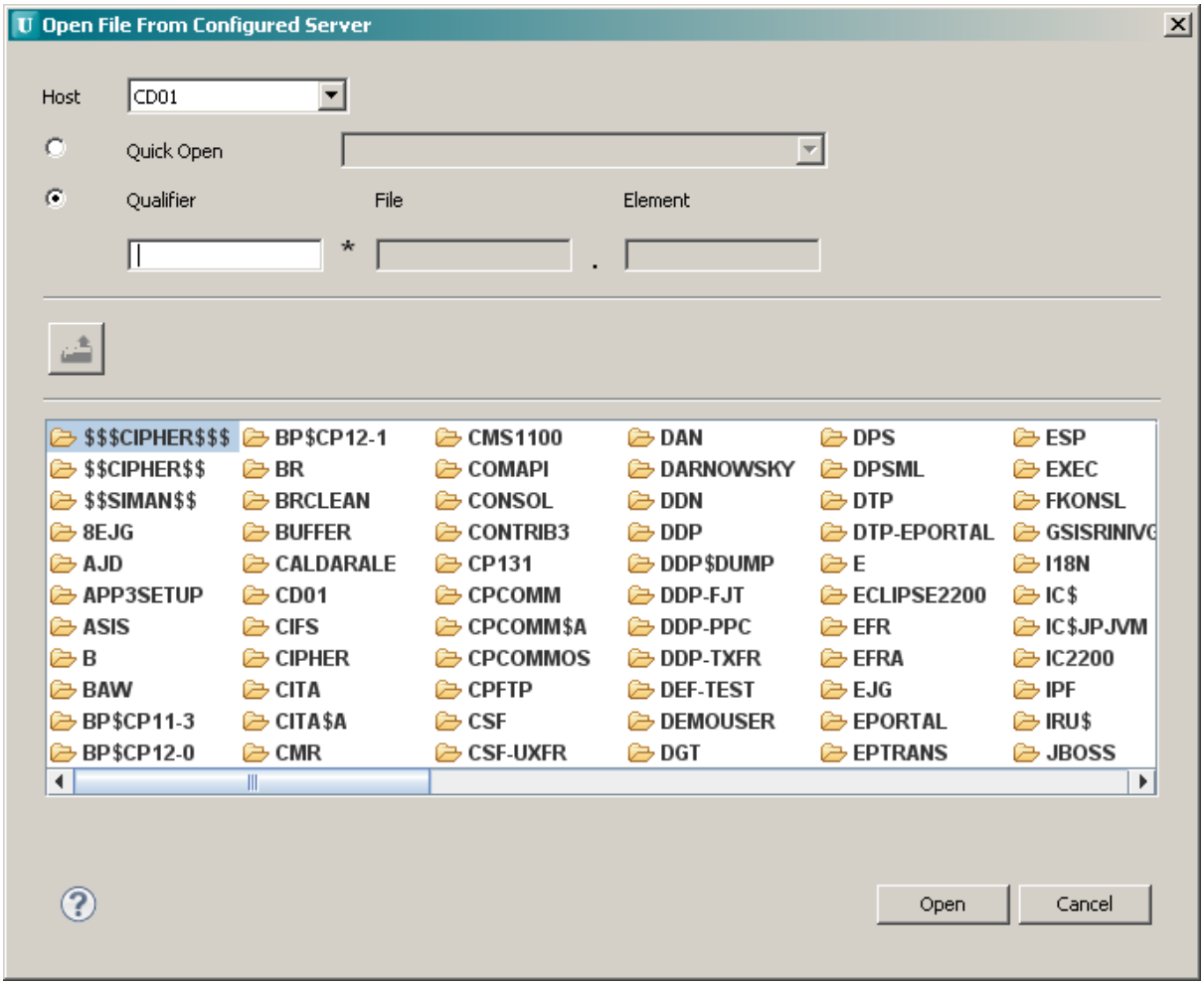
The list box contains the last 10 files opened using OFCS to simplify selection of a previous search.



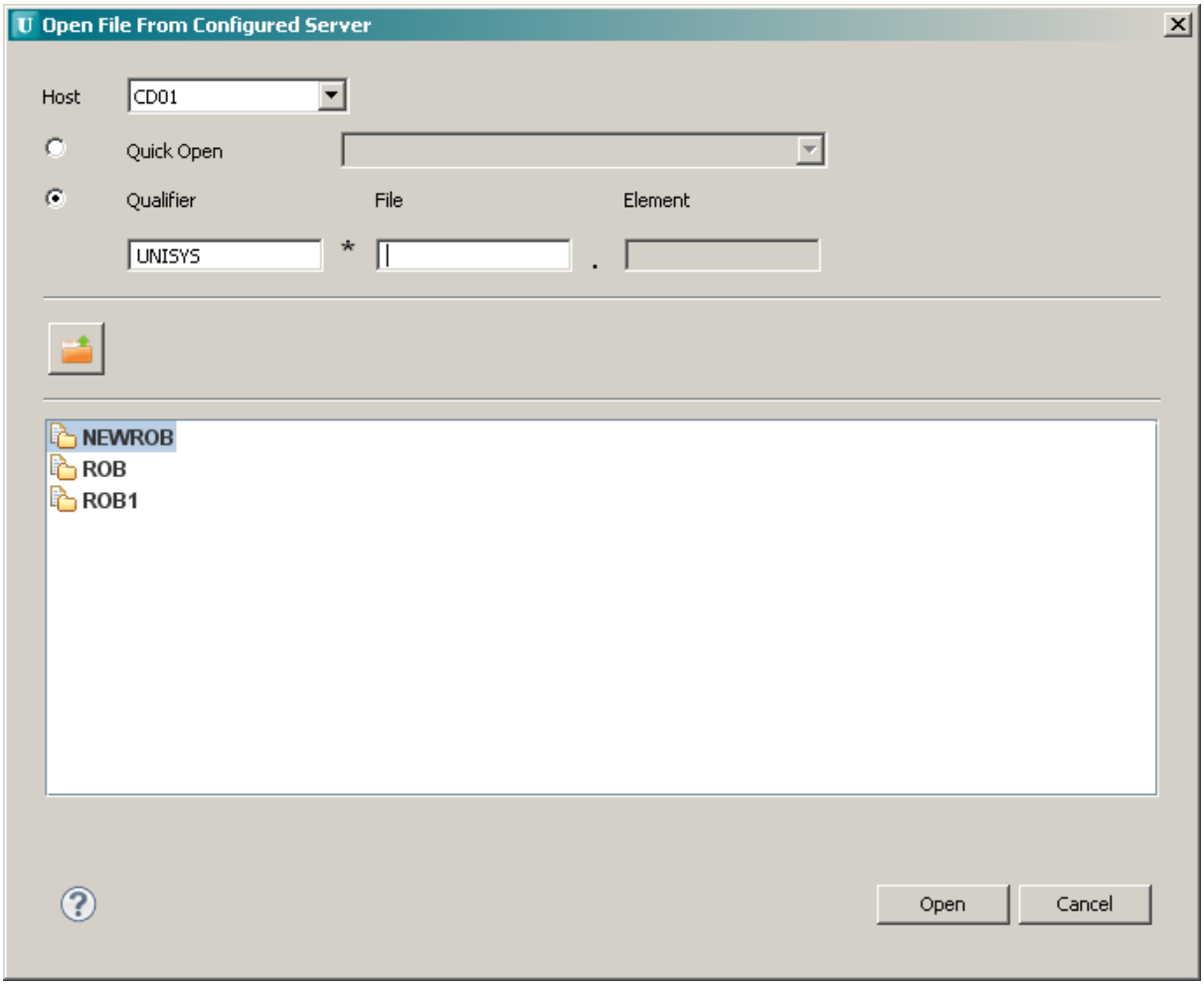
The browse mode is selecting by checking the second radio button.

This mode allows the user to browse the OS 2200 MFD to select their file. The user can also enter the qualifier, filename or element in text boxes to help refine their selection criteria. Eclipse will use techniques to filter the matching files.

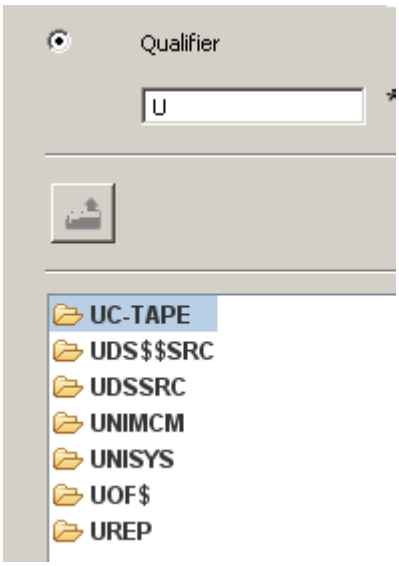
The example below shows the initial wizard screen where all qualifiers are shown in the browse window.



You can either type the required qualifier in the Qualifier text box or you can double click the required qualifier in the browse pane. If you enter a qualifier, you can use the Tab key to put the cursor in the filename text box. As shown below, when a qualifier is selected, only files with that qualifier are shown in the browse window.

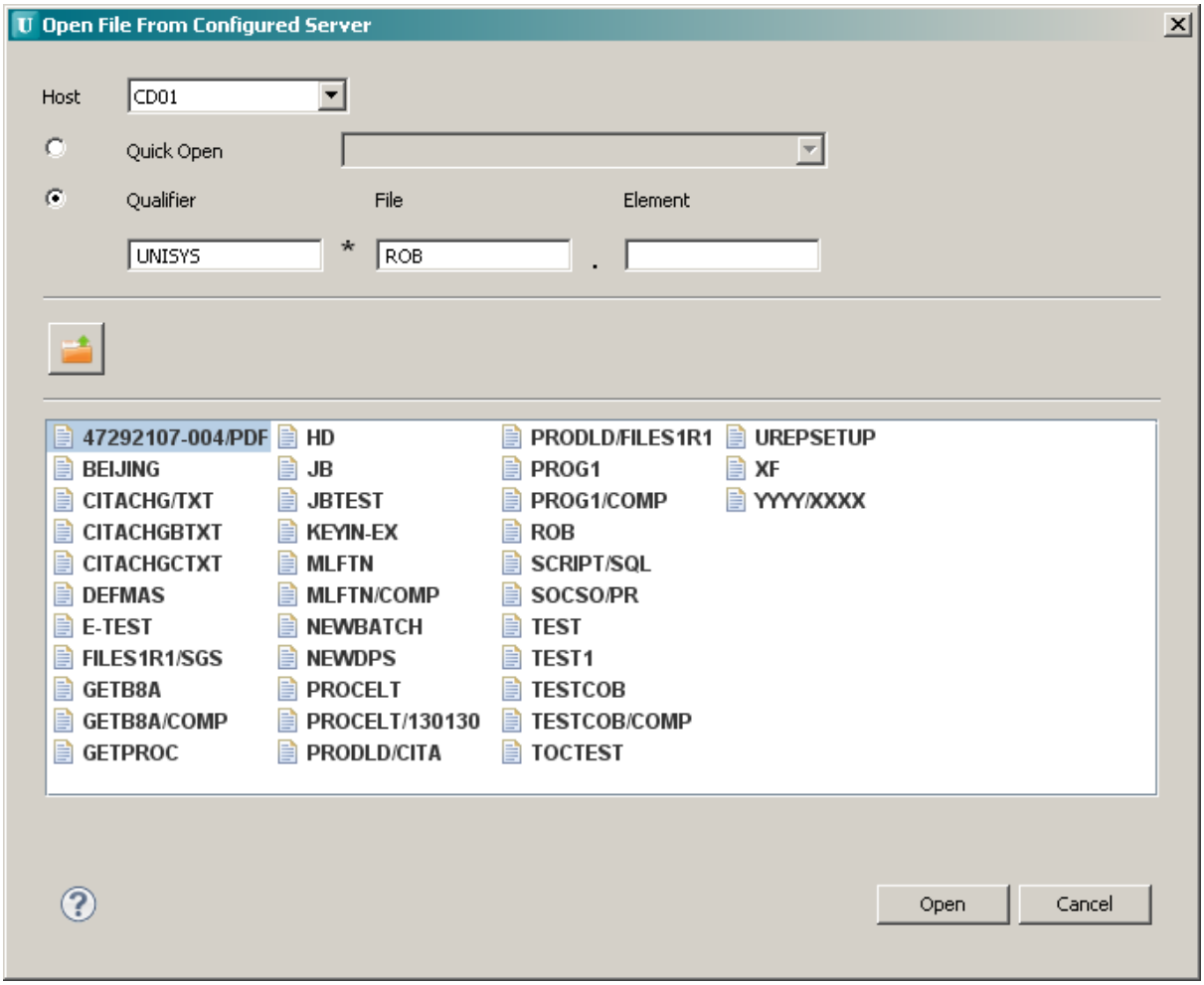


As you type a value into a text box in the browse mode, Eclipse will showing entries that match the characters typed. For example, typing 'U' in the qualifier text box will filter the qualifiers in the browse window to only those starting with 'U':

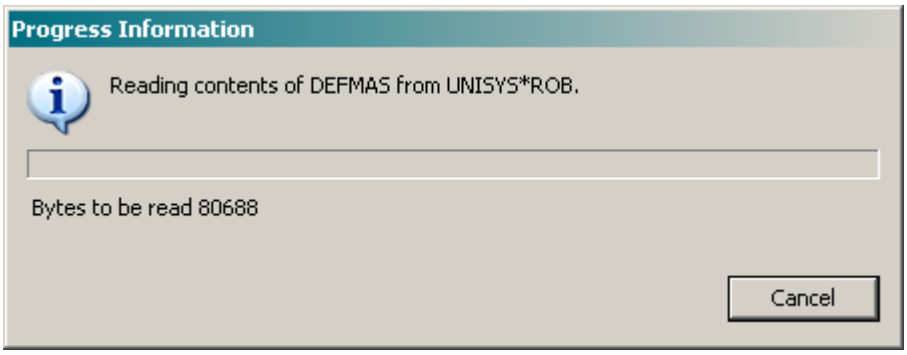


You can use the same techniques to select a file name.

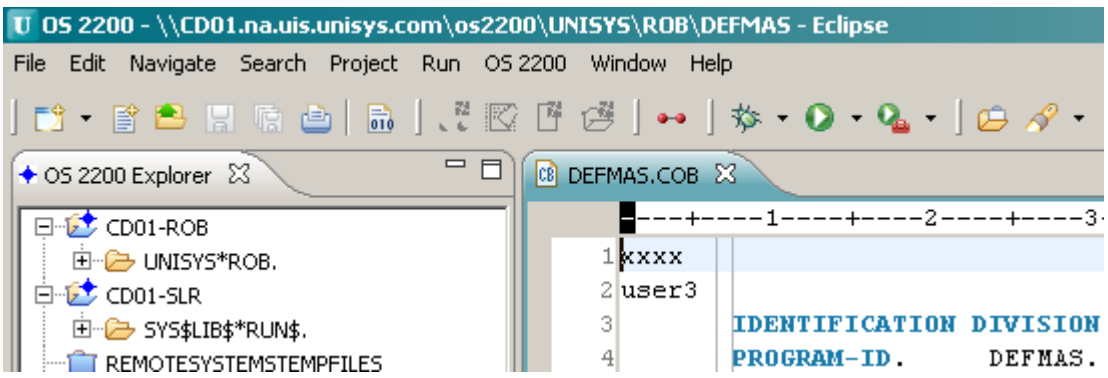
After selecting a file name, the browse window will contain all the source elements for a program file. For a data file, only the file is shown.



The same technique can be used to select the element name. Either enter the element name in the text box or double click on the element. OFCS will try to open the selected element from the OS 2200 program file. The following progress dialog will be displayed.

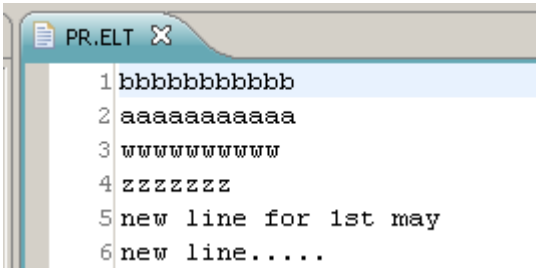


After the file contents have been read, Eclipse will open then in an editor. The editor type will be dynamically determined by Eclipse.

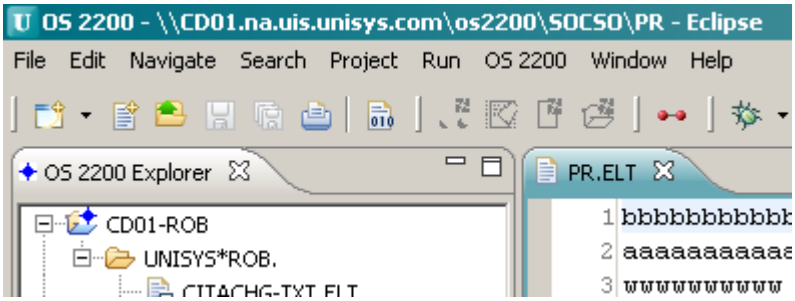


As mentioned above, OFCS can be used to open a data file. The following example shows a data file being opened using the Quick Open method.

Note for a data files, the editor has a title of the filename.elt.

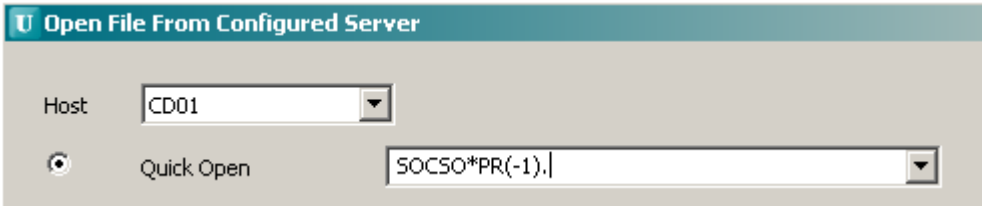


You can determine the actual OS 2200 file name by looking at the title of the Eclipse session:

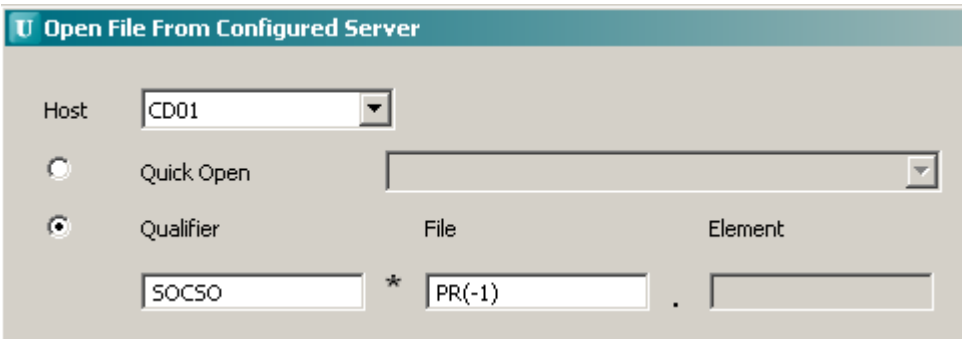


In the above example, CD01.na.uis.unisys.com is the host IP address, os2200 is the CIFS share name, SOCSO is the qualifier and PR is the filename.

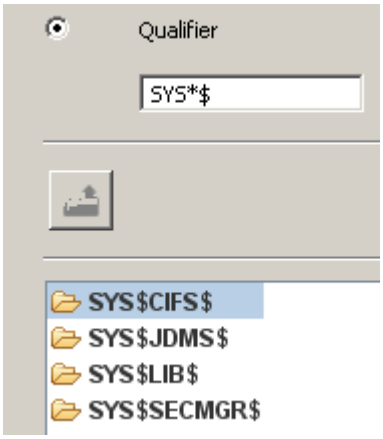
You can even use the file cycle information to select a different cycle. By default, Eclipse will select the latest cycle unless a specific cycle is requested.



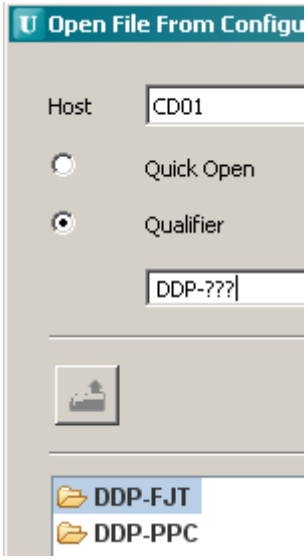
The browse mode text boxes can also be edited.



Note that the browse mode also allows for the use of wildcard searches. '*' can be used for any number of characters and '?' used for a specific character. The example below shows the use of '*' to match any number of characters.



The next example shows the use of the '?' to mask a specific character.



Restrictions with OFCS

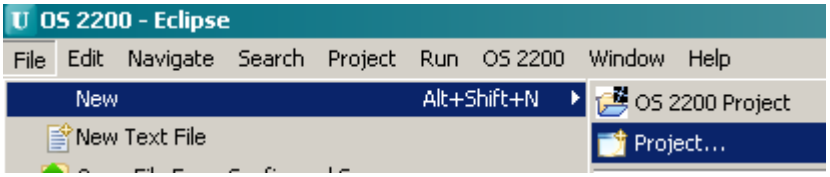
Eclipse is an IDE based around using projects. As OFCS doesn't use projects, some features are not available. These include:

- Search capability
- COBOL COPY Procedure reference
- Compare feature can't be used

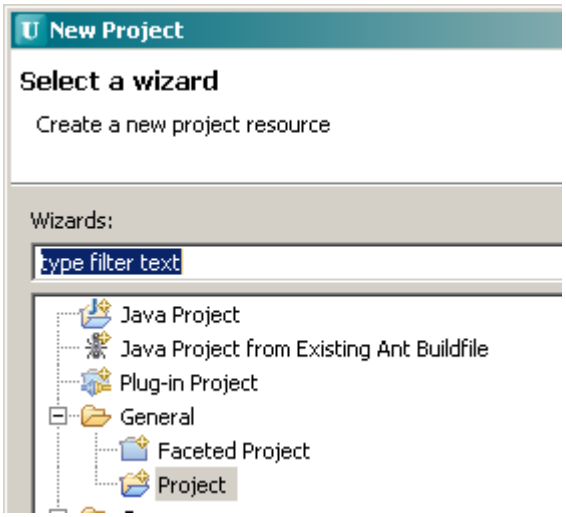
Possible Workaround for Search/Compare with OFCS

A possible workaround to provide Search/Compare capabilities for not only OFCS files but local files as well is to use a 'dummy' general project in your workspace.

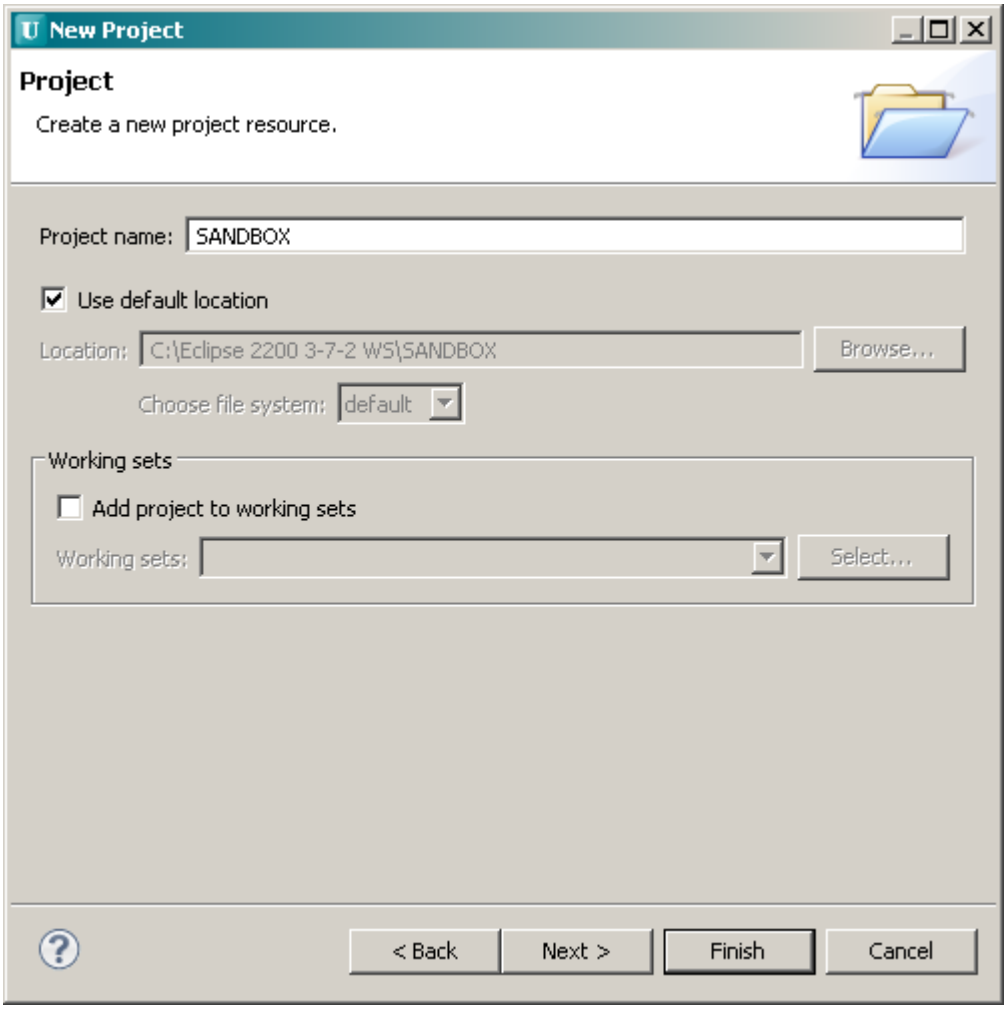
In your OS 2200 perspective, go to File -> New -> Project. An alternative option is to right click in the OS 2200 Explorer View and select New -> Project.



Then select the Project node under General.

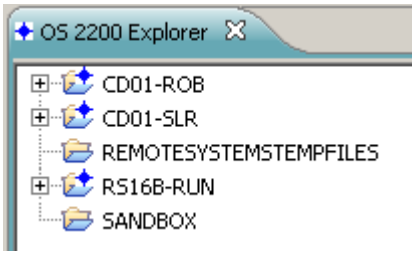


Give your project a name that helps describe it as ‘dummy’ project. In the following example, the project is called ‘SANDBOX’.

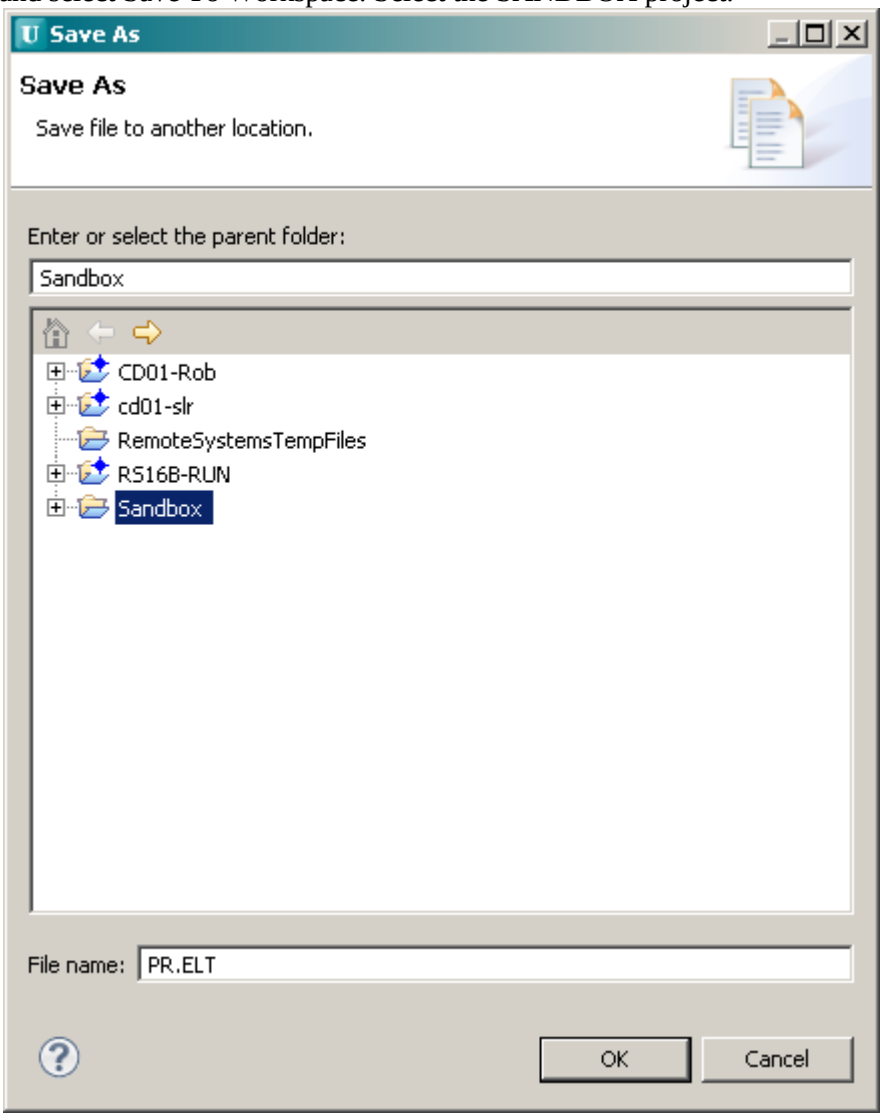


Click Finish.

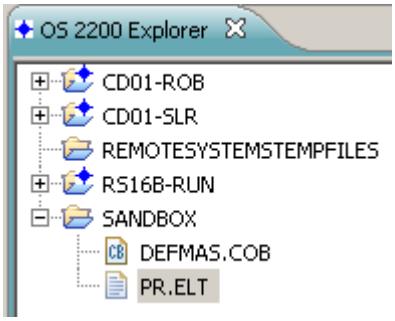
A general project is created in your workspace.



In this example, open a data file with OFCS. To move the file to the SANDBOX project, use Save-As and select Save To Workspace. Select the SANDBOX project.

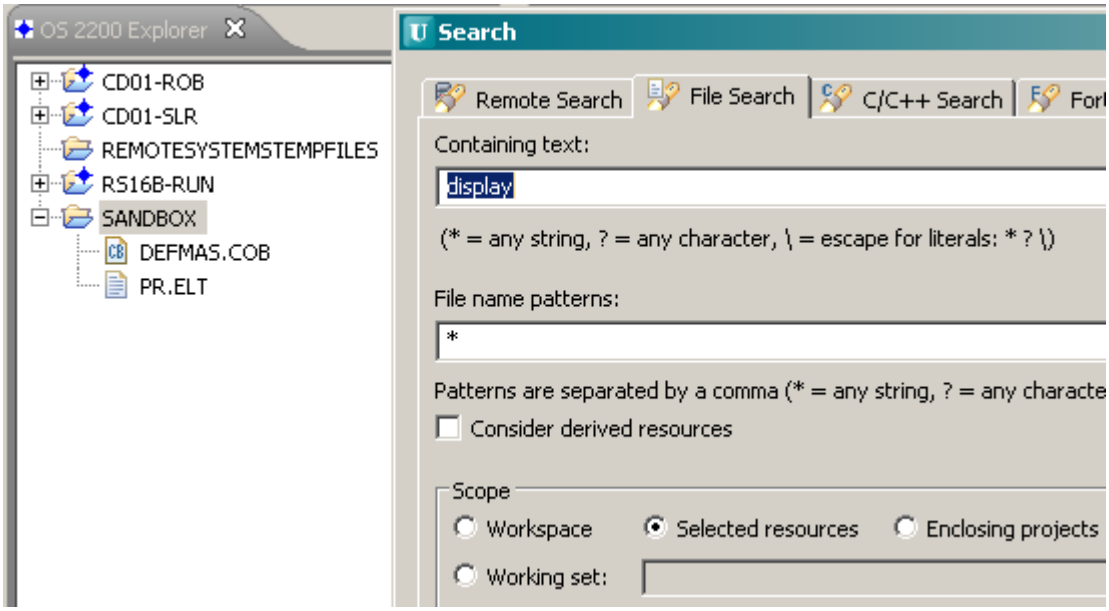


Now open a COBOL source element and use Save-As – Save To Workspace as well. Note the two files are now associated with the SANDBOX project.

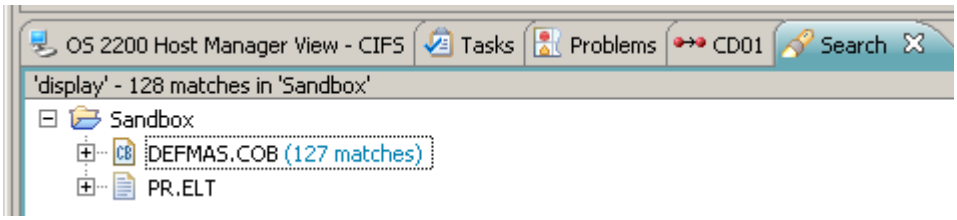


Note that using Save-As in this case also closed the editor - when using the Save-As with 2200 projects and OFCS files, the editor remains open and merely changes the name/property as needed.

Now we will do a search over the project. Highlight the project. Go to Menu -> Search.



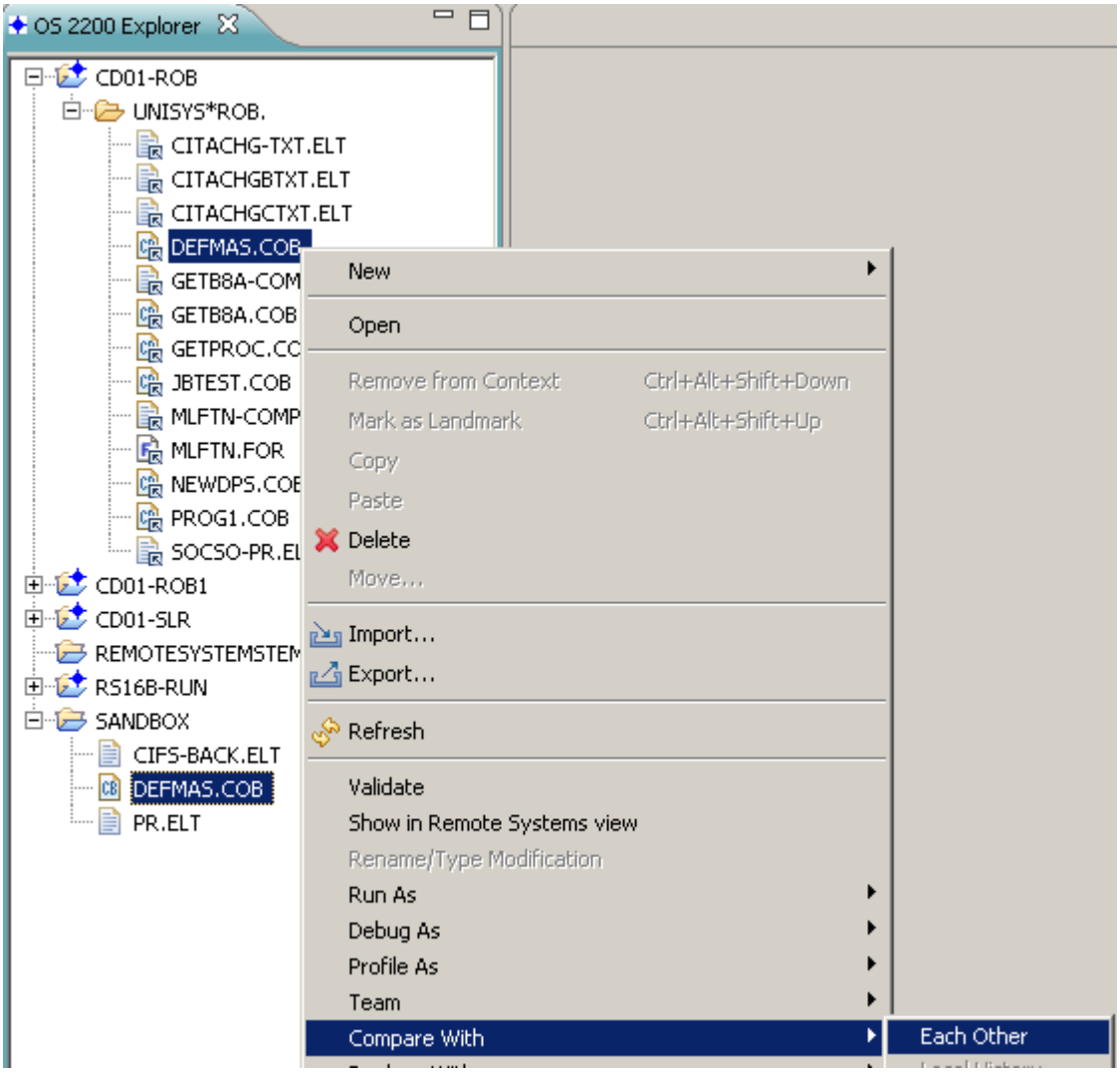
And the results are displayed in the Search View:



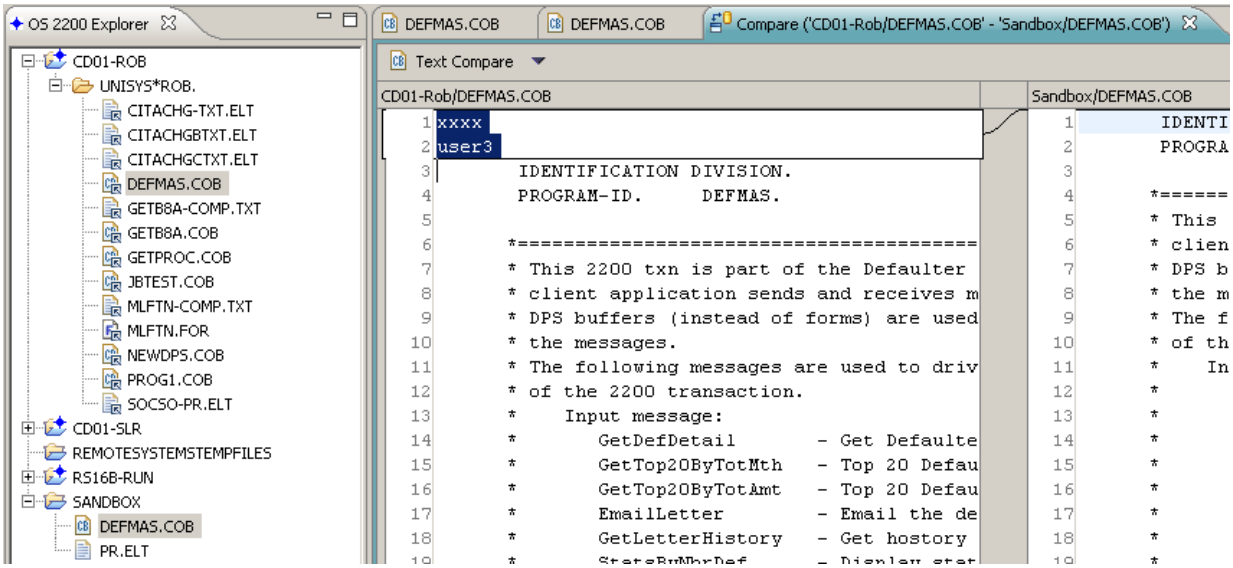
The Search scope could be changed to the entire workspace or just selected files.

When the COBOL source, DEFMAS.COB, was opened, Eclipse opened the COBOL editor – outline worked but COBOL Copy proc reference did not work. Code navigation using sections/paragraphs and opening declarations worked fine.

Now compare the file in the SANDBOX project with a COBOL source element in an OS 2200 project. Highlight the two COBOL elements, right click on the source file and select ‘Compare with’ followed by ‘Each other’.



The results are shown in the Editor pane.



You can edit and update the files in the SANDBOX project. If you want to save the updates to a 2200 host, use the Save-As with Save To Configured Server.

Eclipse and Rolled-Out Files

If an OS 2200 file (program or data) is stored on Fixed disk, it maybe subject to be rolled-out when mass storage reaches a threshold. When using Eclipse project or OFCS methods to access a file, the handling of a rolled-out file can be impacted by the system security level and a CIFS parameter. The information below is a summary of how Eclipse handles rolled-out files in these cases. If you are using the Eclipse Telnet session, then the same handling as per any Demand run is performed.

Fundamental Security

- CIFS does not initiate a ROLBAK since the CIFS subsystem runs in privileged mode
- Error status returned to Eclipse and user gets error dialog saying file is not accessible and maybe rolled-out
- User must manually initiate the ROLBAK using a demand session e.g. @ASG,A
- Recommend CIFS$WAITROLBAK be set to 1 so Eclipse responds immediately

SECOPTn Security

- CIFS will initiate a ROLBAK. (If not, CIFS not installed correctly. In this case, it is likely the privileges for the userid that owns and runs the CIFS subsystem are incorrect.)
- CIFS waits until the earlier of 2 events. Note Eclipse will show no activity during this time.
    1. File is restored to text and CIFS then continues with Eclipse request e.g. to open the file.
    2. CIFS$WAITROLBAK timer expires. Error status is returned to Eclipse and user gets error dialog.
- Recommend CIFS$WAITROLBAK in CIFS-BACK runstream be set to 1. This would avoid the Eclipse user thinking Eclipse is 'hung'. The default value is 600 seconds.
  System profile should set CIFS$WAITROLBAK to default 600. This is for OS 2200 batch and demand connections to CIFS.
- If other CIFS network connections require a longer CIFS$WAITROLBAK period, a userid profile should be created with the appropriate value.

Contact your system administrator to modify the CIFS-BACK runstream to set the CIFS$WAITROLBAK to 1 second. The following example shows the suggested change to the SYSLIB*RUN$.CIFS-BACK runstream:

```
@cifsut
set
set         CIFS$WAITROLBAK=1
@free       SYS$LIB$*CIFS$LIB
@xqt        modps-z
```

Using the RDMS JDBC Client

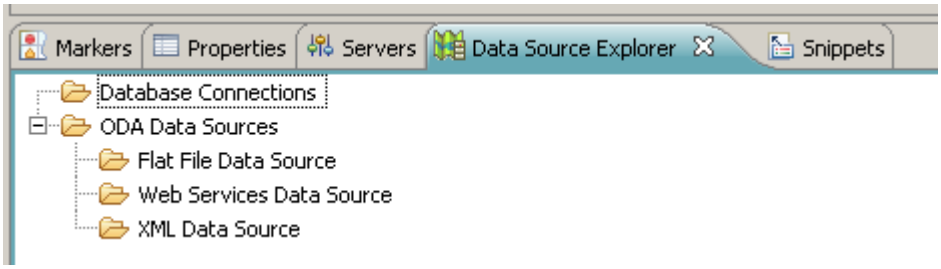
This section provides some guidance in using the JDBC interface located in the Data Source Explorer view of the J2EE perspective in Eclipse. This can be useful for RDMS programs where the developer wants to test a SQL command before including it in a program. However this section will also prove useful to Java developers.

It is assumed that the JDBC-RDMS software has been installed and configured on the OS 2200 host plus the relevant background service jobs are running. Review to the Relational JDBC Driver for ClearPath OS 2200 User Guide for information on the installation and configuration of the OS 2200 software. Consult your system administrator if unsure.

Configuring the JDBC Client

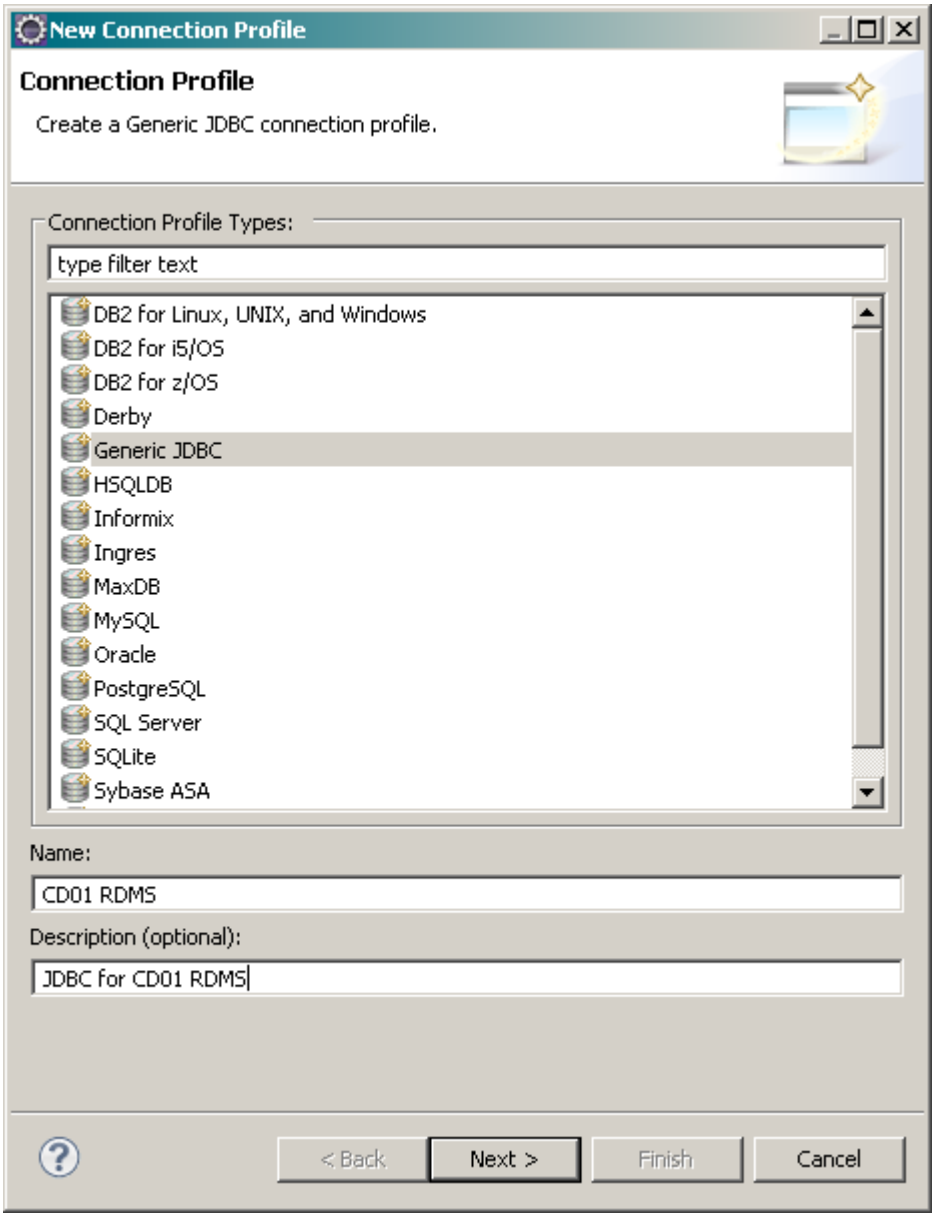
Refer to the section 4 of the ClearPath OS 2200 IDE for Eclipse Application Development Guide for Java EE. It is assumed a folder "C:\Database Drivers" has been created and the appropriate jar files copied into this folder from the <appl group>*JDBC$CLIENT installed file.

Open the Java EE perspective and click on the Data Source Explorer tab.



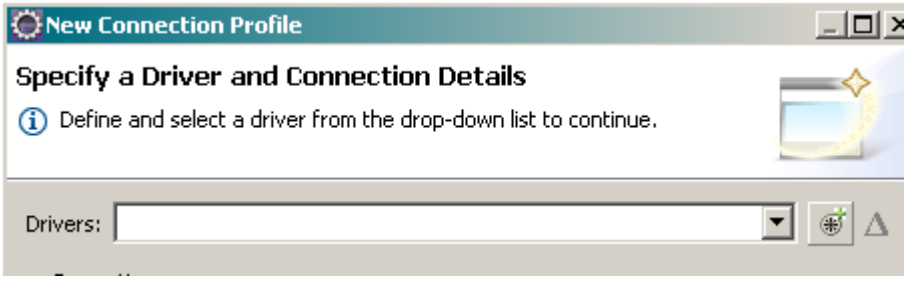
Right click on **Database Connections** and select **New.**

The following dialog appears.

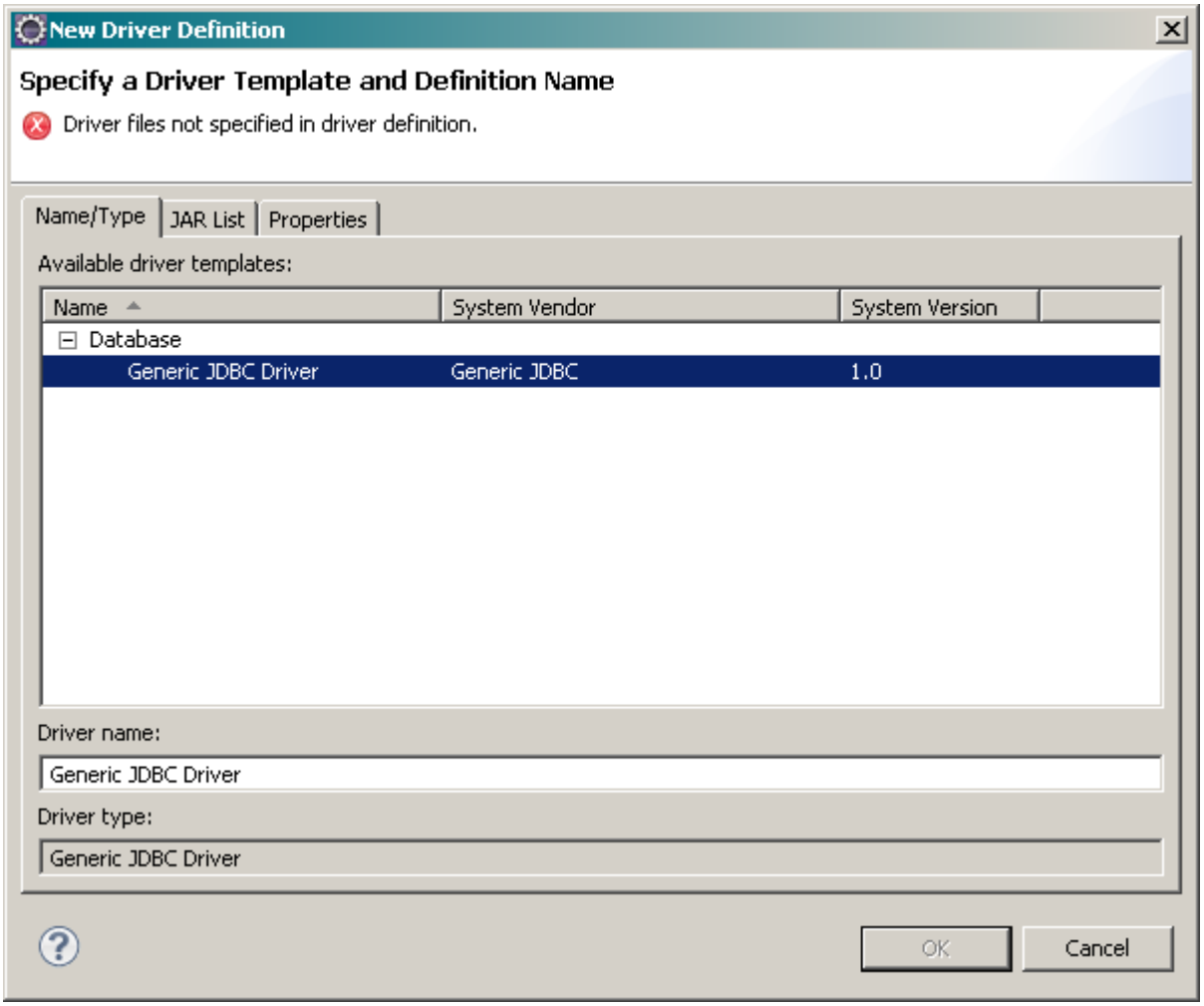


Select the **Generic JDBC** connection profile type.

Enter a value in the Name field to identify this connection. Optionally enter a description. Then click **Next.**



Click the  icon to the right of the Drivers field. A dialog showing available drivers is displayed.



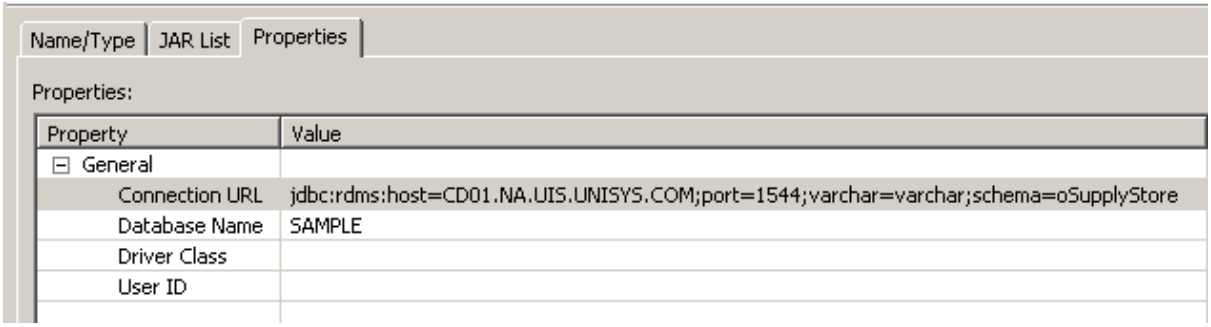
Select the Generic JDBC Driver.

Click on the **JAR List** tab so the JAR files copied into the C:\Database Drivers folder can be added.

Click on the **Add JAR/Zip** button and browse to the C:\Database Drivers folder. Select the rdmsdriver.jar file.

Repeat for the unsiys-jca.jar file. The result should look like below.

Click the **Properties** tab.



In the Connection URL, type **jdbc:rdms:host=name; port=1544; varchar=varchar; schema=OSupplyStore**

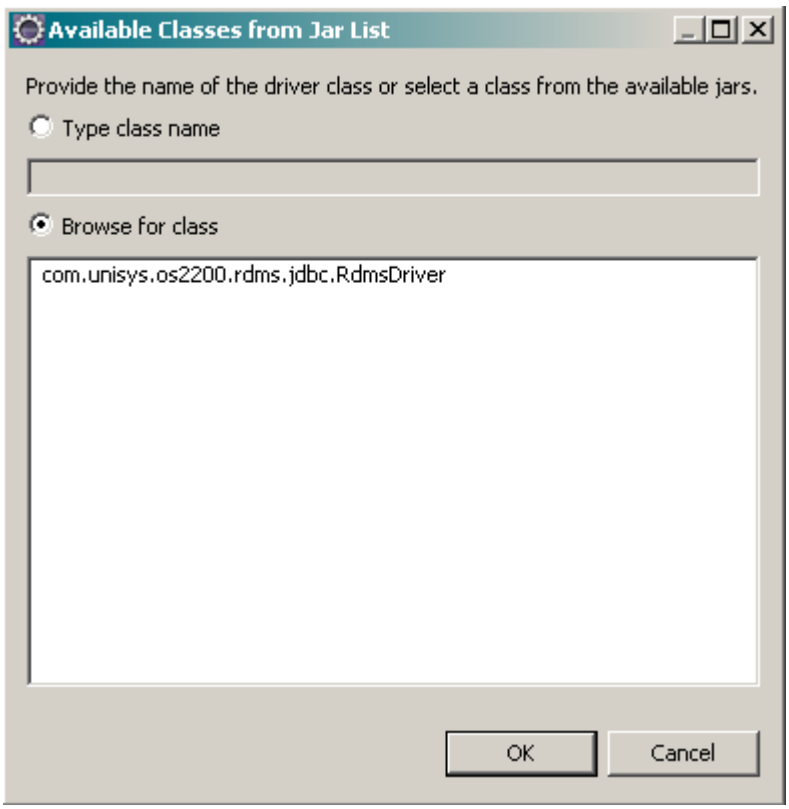
where

name is the name of your OS 2200 host.

1544 is the default port number for application group 3; your port number can differ if you are using an application group other than 3 or are not using the default port.

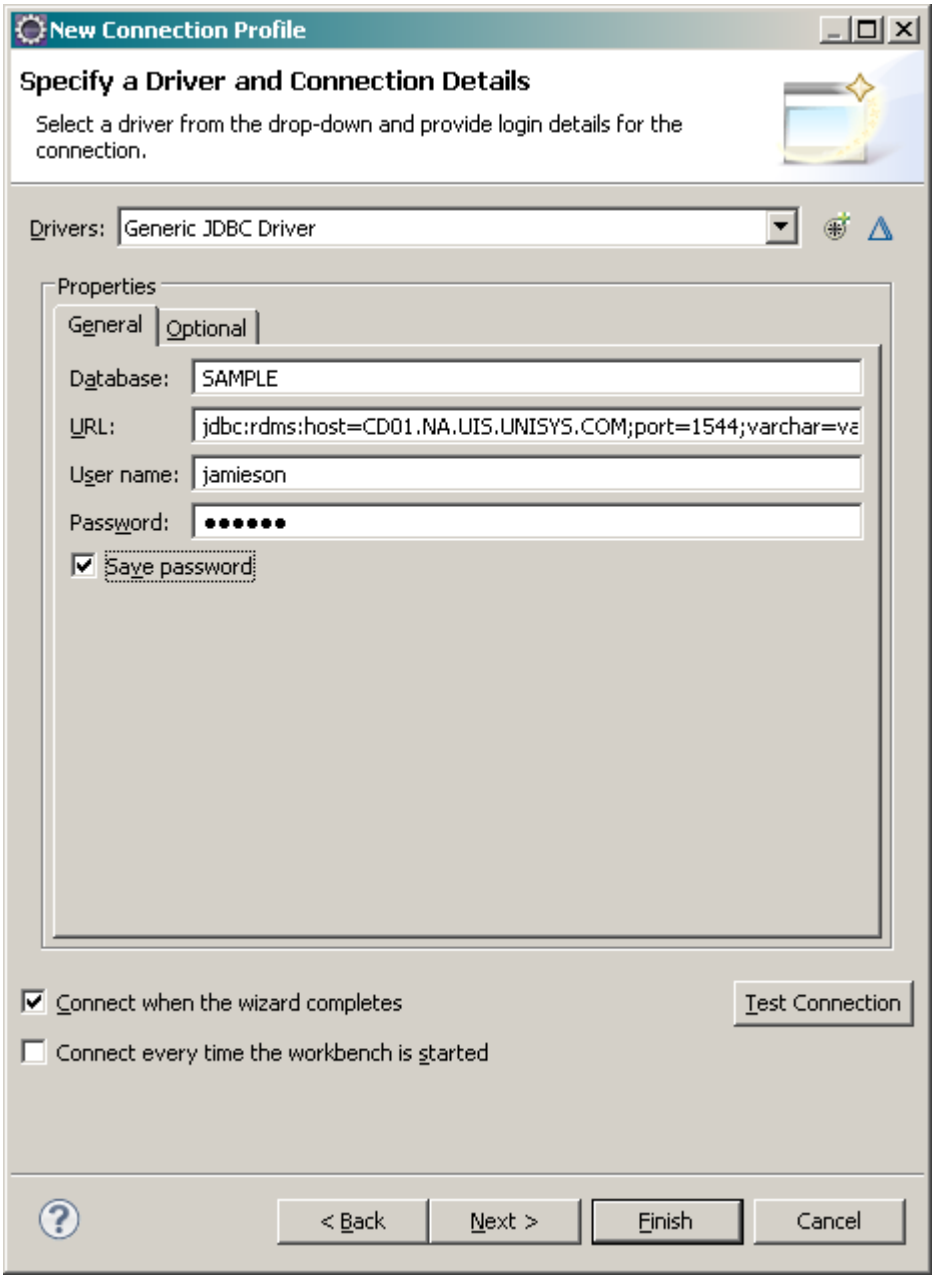
The schema name should match the RDMS schema you want to access.

Highlight the Driver Class field and click the icon in the right edge of the field.



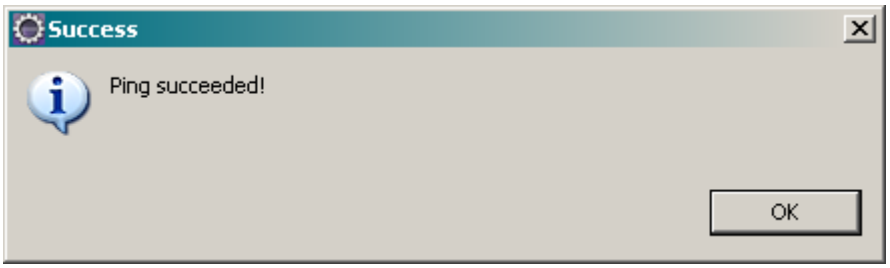
Click on Browse for class and highlight com.unisys.os2200.rdms.jdbs.RdmsDriver.

Click **OK.**



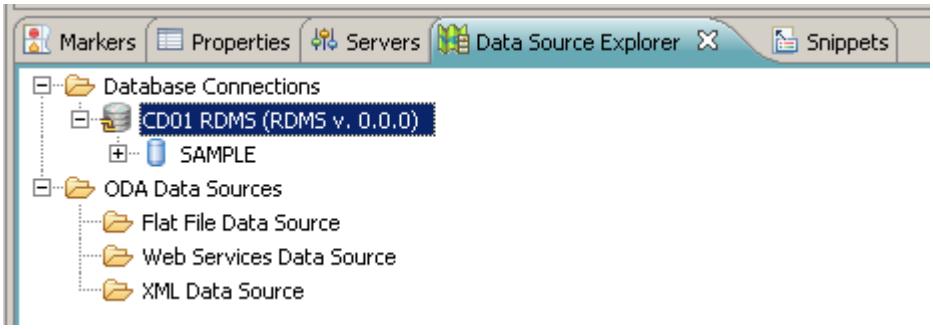
Enter your OS 2200 credentials.

At this stage, we can test the connection. Click **Test Connection**. If OK, the following pop-up appears.



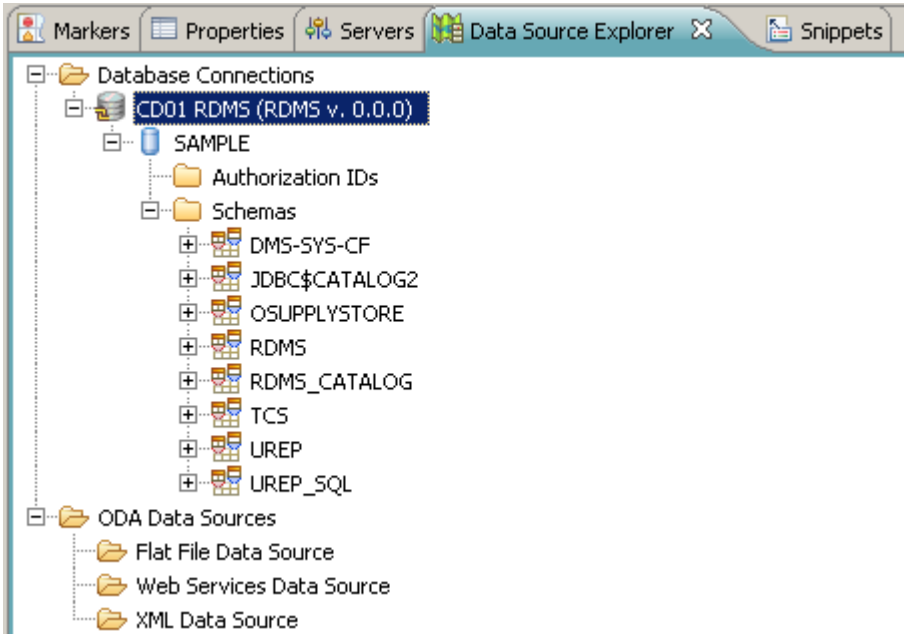
Click **OK** and then click **Finish**.

The Data Source Explorer pane will now show the connection details that we have just defined as shown below.



Retrieving RDMS Schema Information

Expand the database name entry. In this example, it is SAMPLE. This make take a little time as the JDBC driver is used to read key information from the RDMS catalog on the 2200.

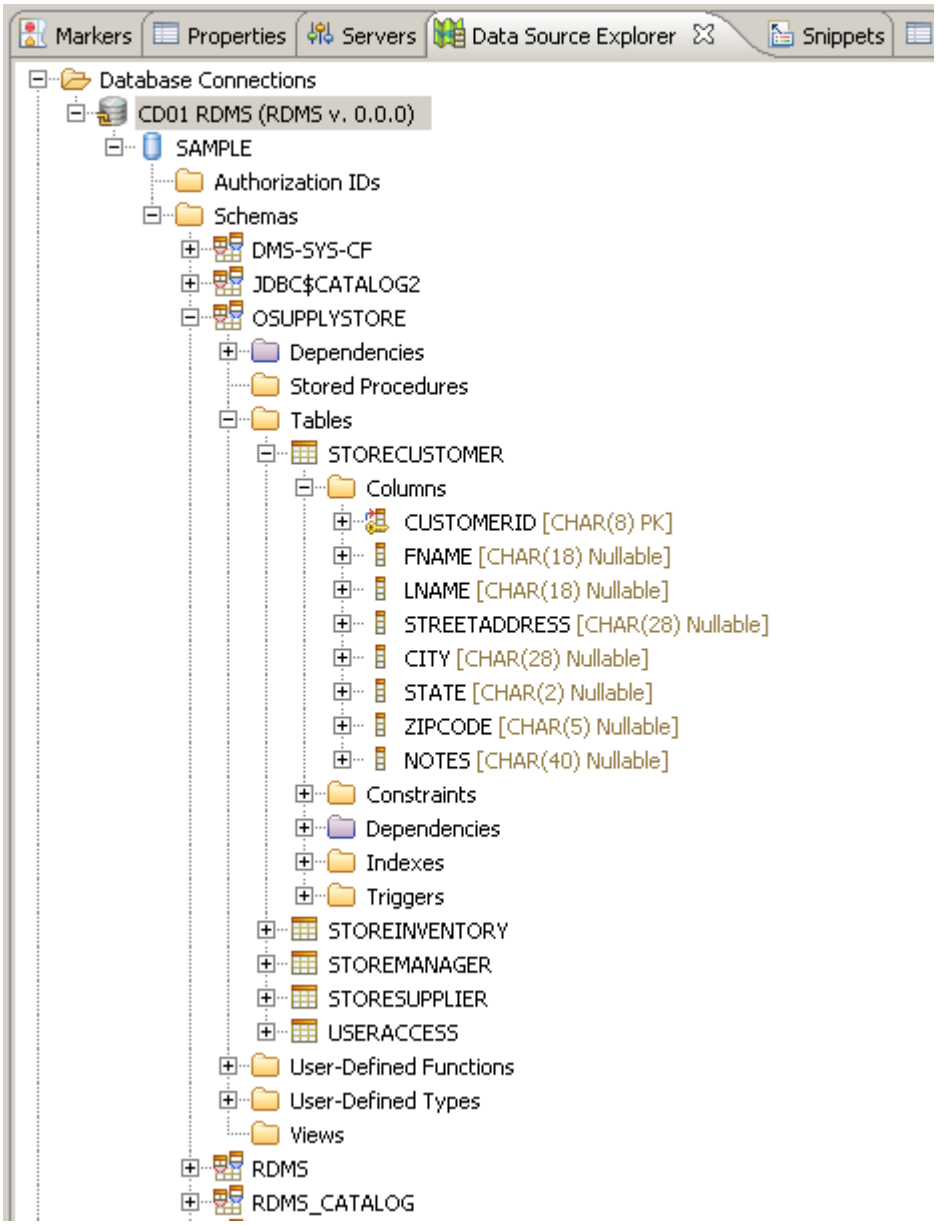


Expanding the Schemas entry shows the available schemas.

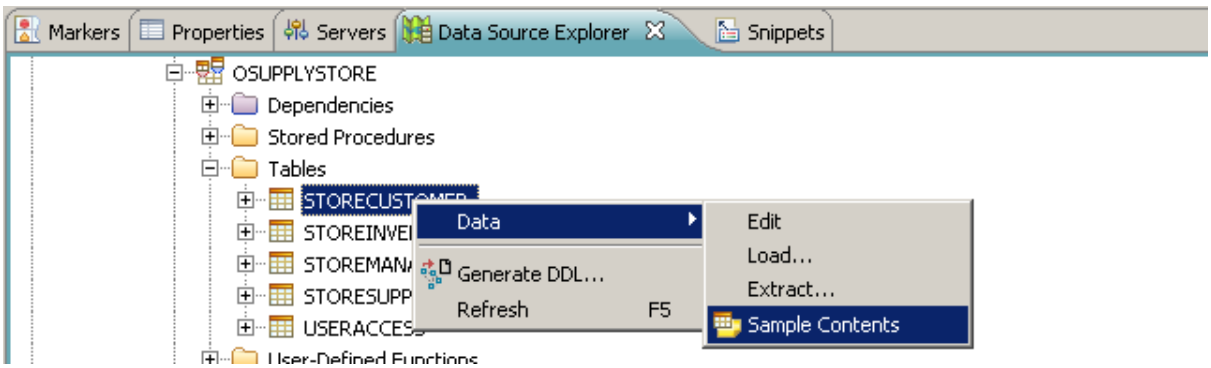
Only RDMS schemas are displayed including internal RDMS schemas that describe DMS schema information.

Expand a RDMS schema entry and then a table entry for this schema.

You will see the columns defined for the table and their data definition.



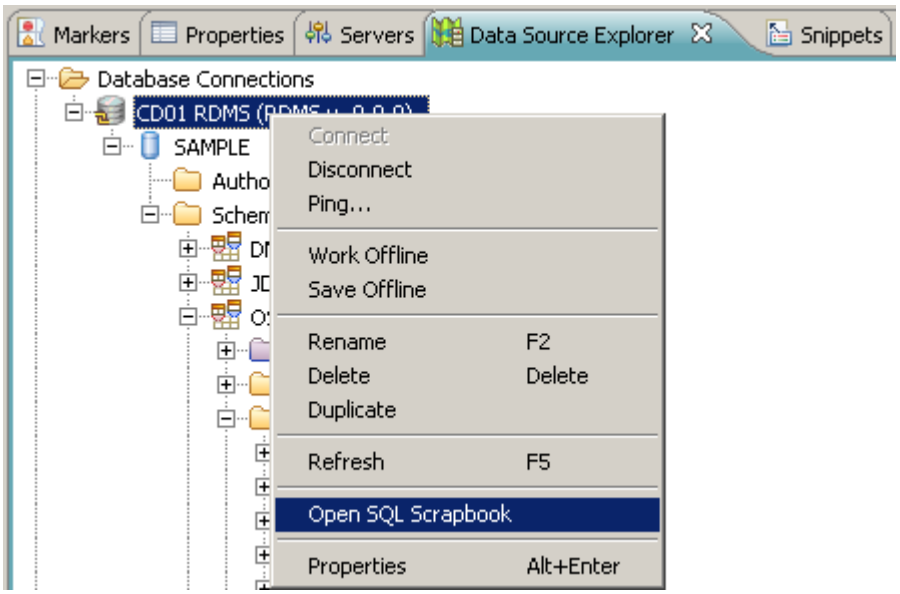
Highlight a table entry, right click, select **Data** and then **Sample Contents**.



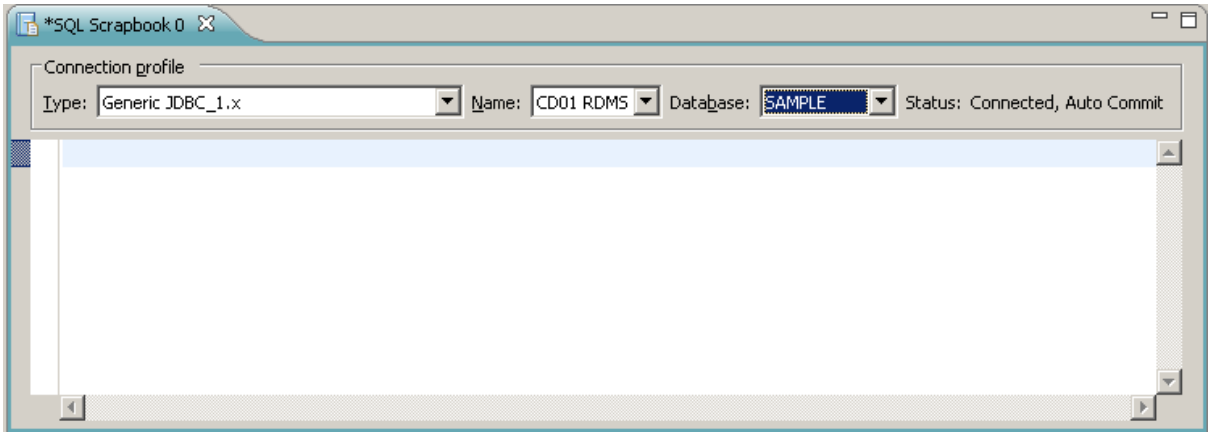
A new view called SQL results is opened and the data in the table is displayed.

Developing and Testing SQL Commands

Right click on the Database Connection entry.

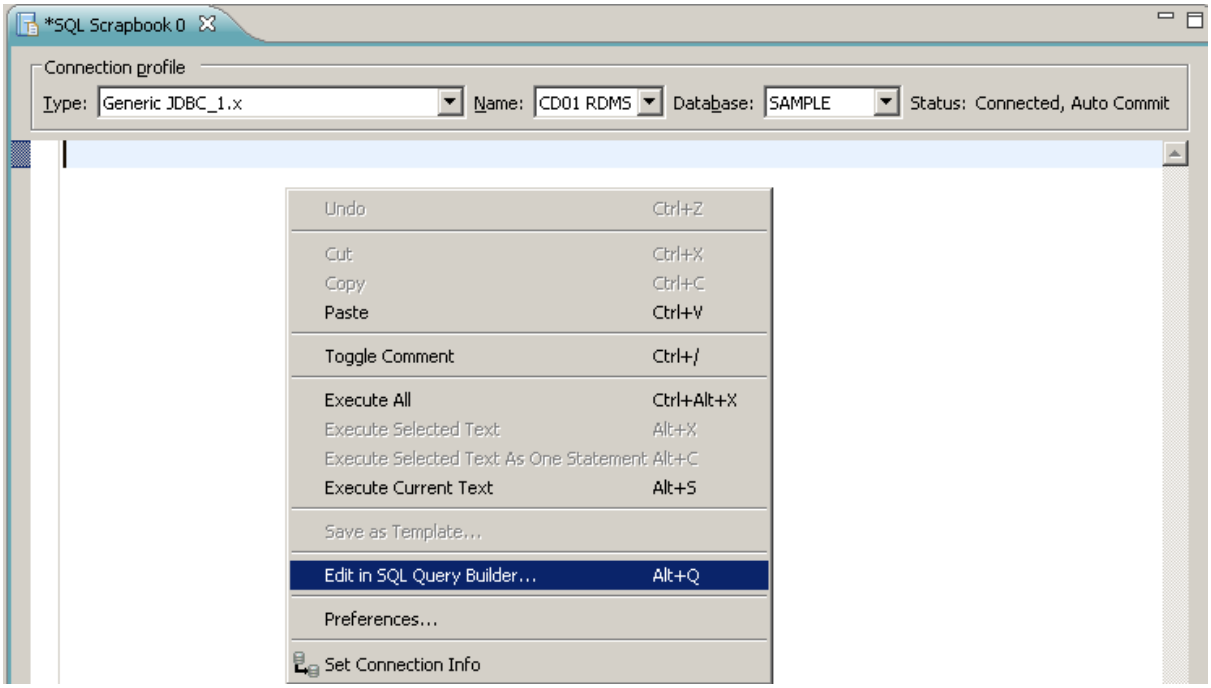


Select **Open SQL Scrapbook**. A new Window appears as shown below.



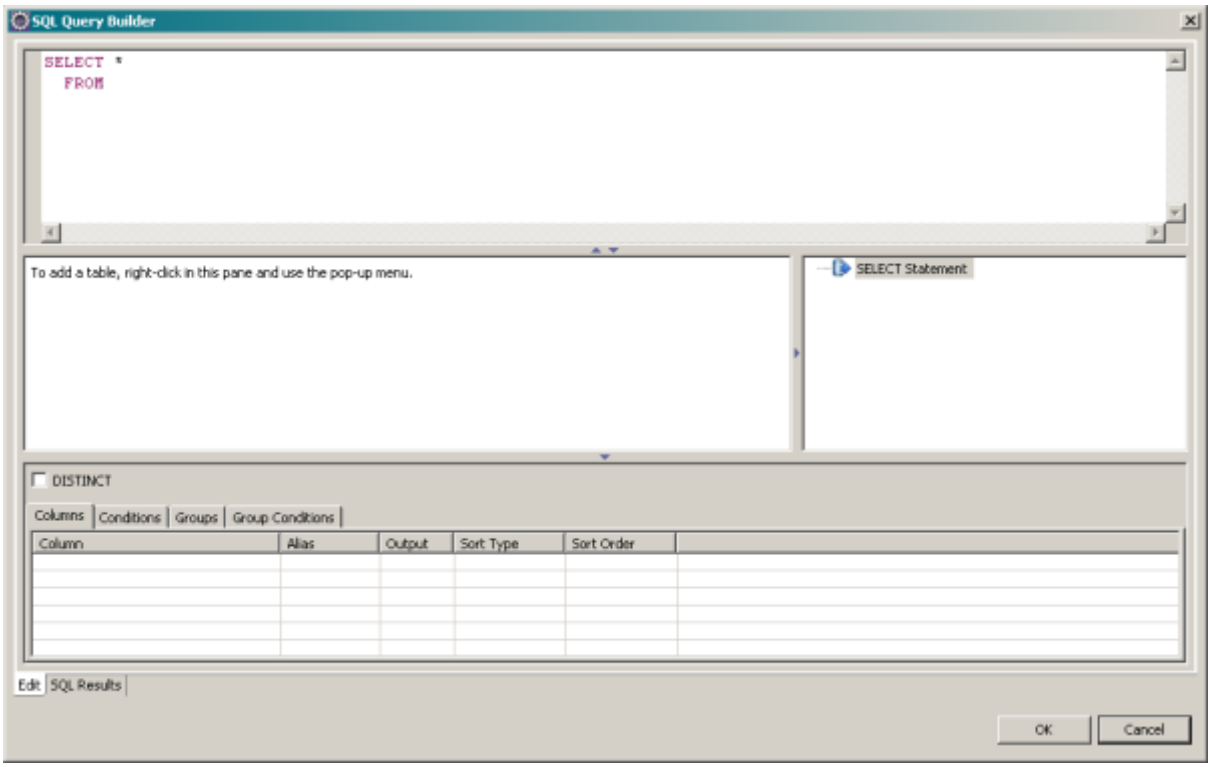
Use the list box to select the Type, the Name and the Database. These are the values we defined earlier.

Right click in this window.

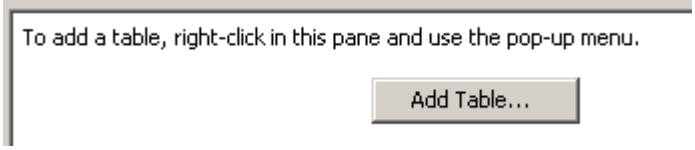


Click the **Edit in SQL Query Builder** entry.

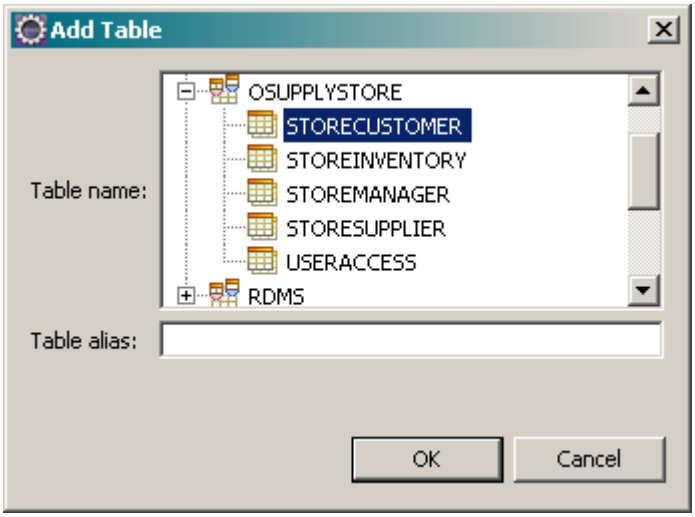
The window for the SQL Query Builder opens. This allows you to manually type in the SQL command in the upper pane or use the advanced features described below to help automate this process.



In the middle left pane (with the "To add a table…" text), right click and then click on Add Table.



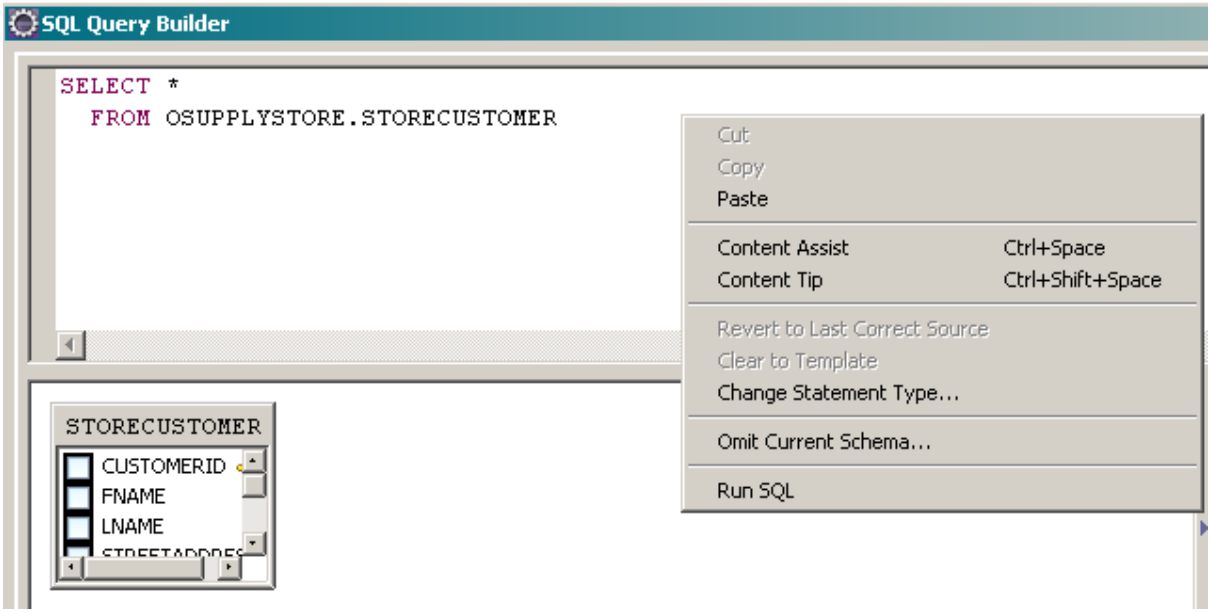
Scroll to the correct database and expand the entry to show the tables.



Highlight the required table and click **OK.**

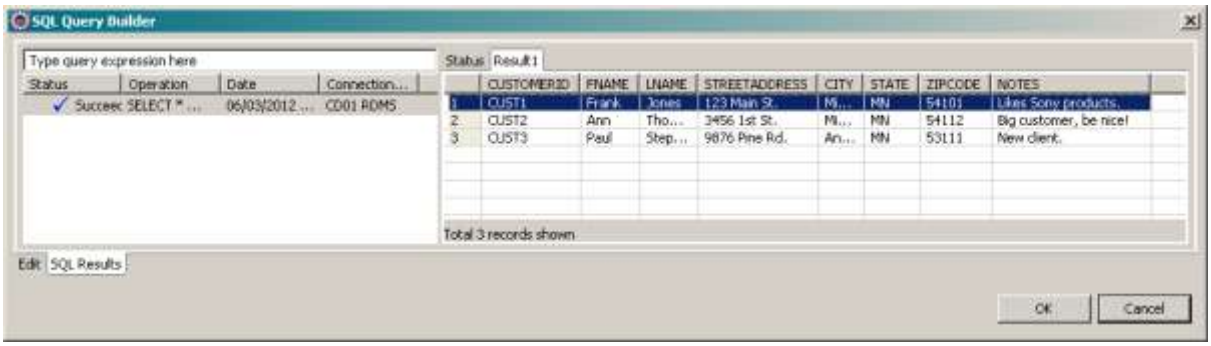
A small box appears with the table name as the title and all columns listed. A check box appears next to each column.

Notice how the SELECT command in the upper pane now has this table added to the FROM clause.

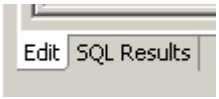


Right click in the upper pane and select **Run SQL.**

The JDBC driver will pass the command to the 2200 for processing and return the results. If the command is in error, this is reported.

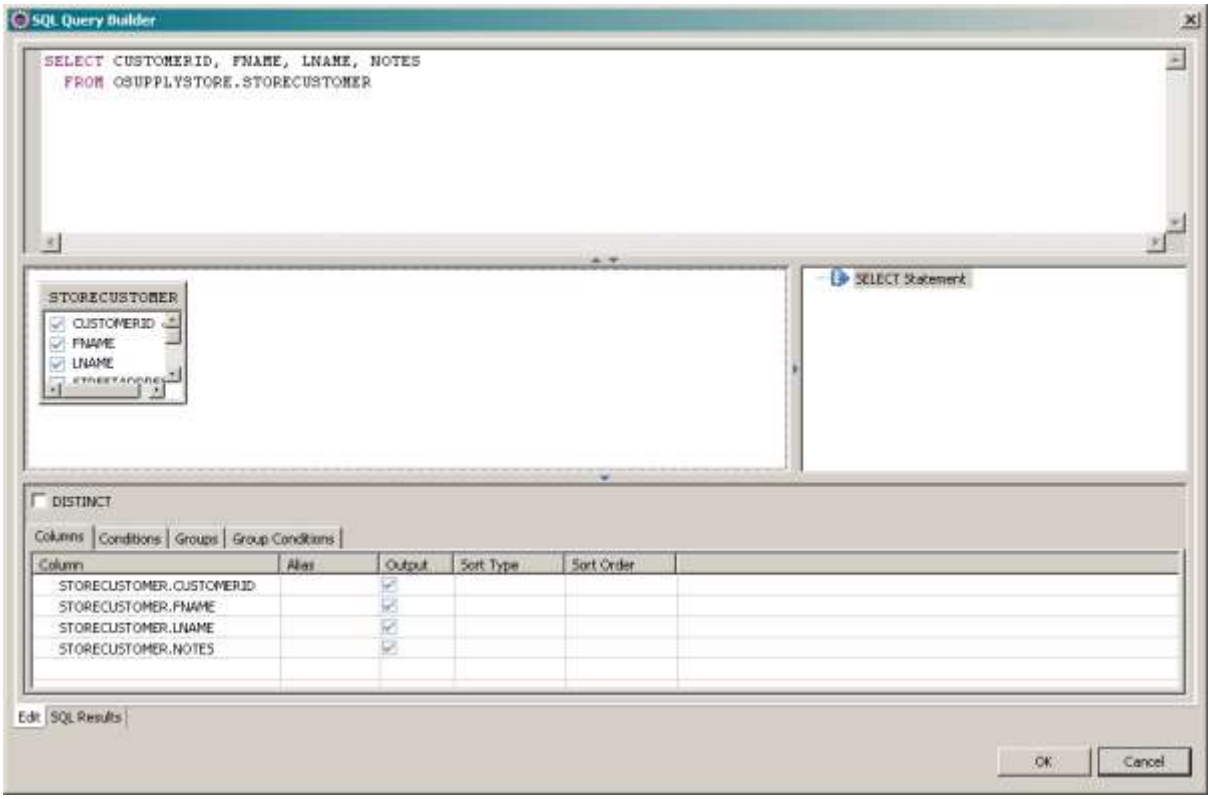


Note that the bottom left of the SQL Query Builder window now has the option to select **Edit** or **SQL Results.** If you click on SQL Results, the data returned from the 2200 based on the submitted SQL command is shown.

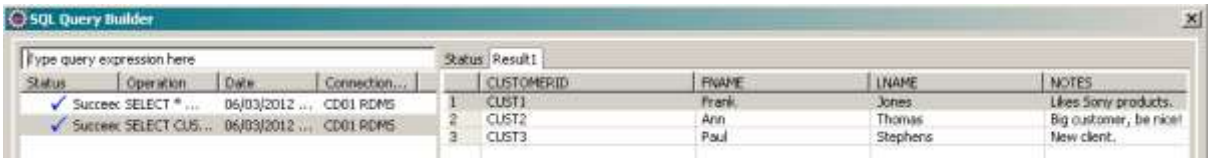


Click **Edit** to return the query builder editor.

In the bottom pane, we can set different options for the SQL command. In the middle left pane, check the columns that you want to display. Now look at column tab in the bottom pane. We see the columns we selected are listed as Output is checked.



Run the SQL command and confirm that the output only contains the selected columns.

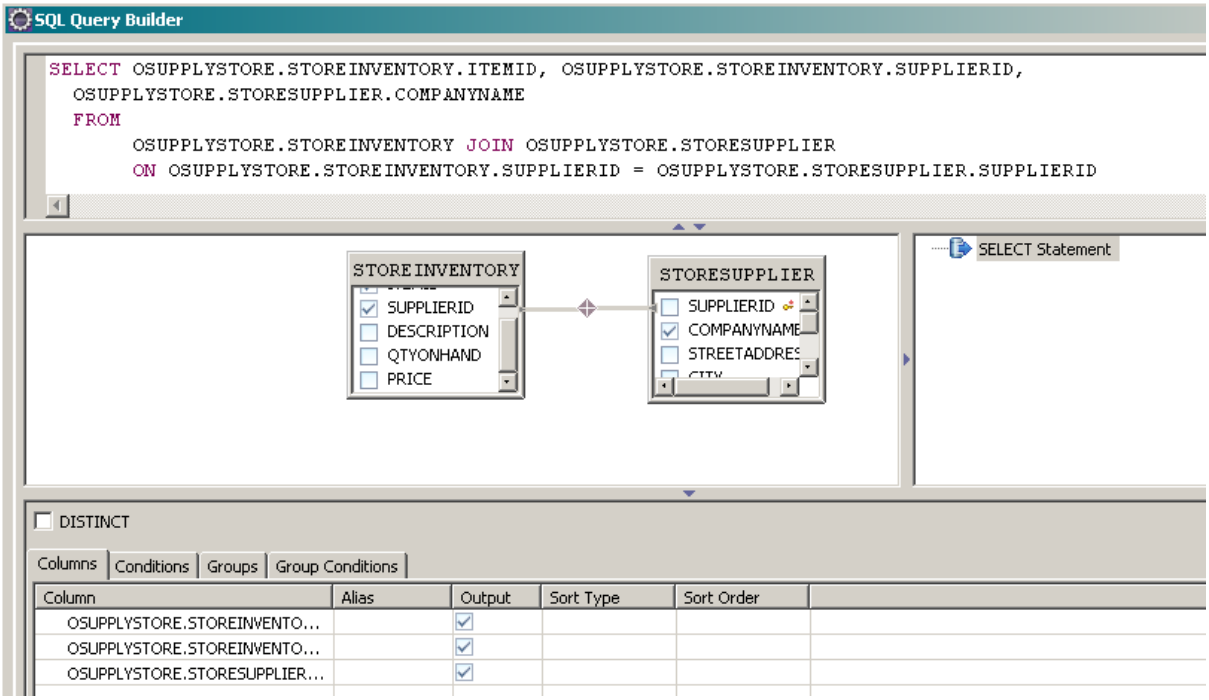


In the middle left pane, add another table.

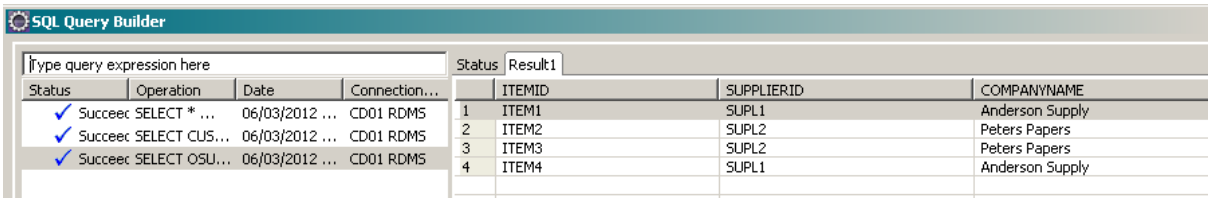
Note the SQL command is updated with the new table in the FROM clause.

Select some fields in the new table.

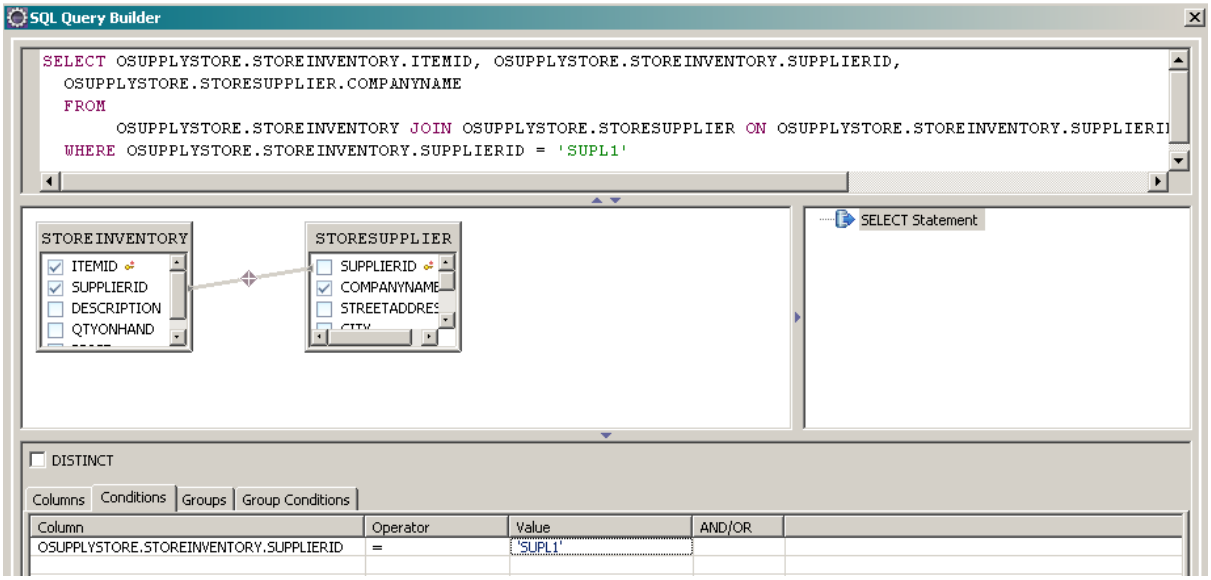
We can also select the fields to be used to join the tables. In the example below, SUPPLIERID is highlighted in the STOREINVENTORY table. The mouse is held down and the cursor dragged to the SUPPLIERID in the STORESUPPLIER table. Note that the system shows a line linking these fields. Also note that the SQL command now has a JOIN and ON clauses added.



Run the SQL command and check the results. As shown below, we get the required information. ITEMID and SUPPLIERID come from the STOREINVENTORY table while COMPANYNAME comes from the STORESUPPLIER table.

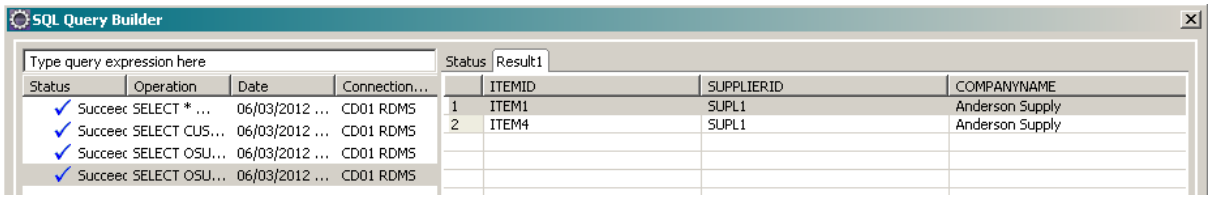


In the bottom pane, click on the **Conditions** tab. This is how we can easily define the WHERE clause to apply to our SQL command. Click on the column field and only valid columns are shown. Select a value. Then select the Operator and finally select the Value. Note the value could be a literal or a column entry from a table.

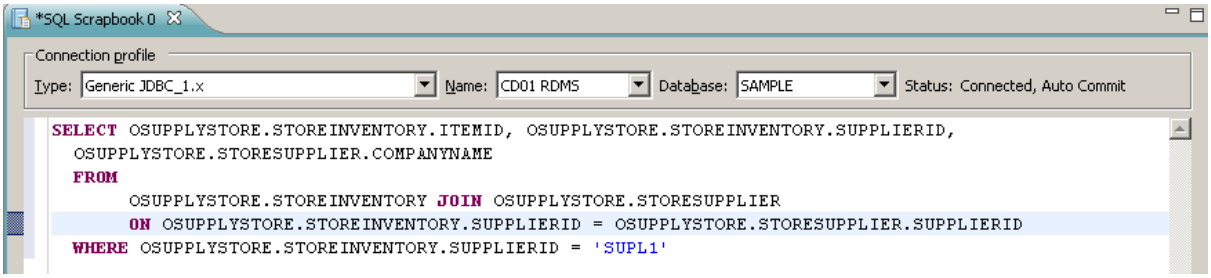


Note the WHERE clause is now added to our SQL command. We only wanted to see SUPPLIERIDs equal to ‘SUPL1’.

Run the command and check the results.



We can still edit the command in the upper pane. This might be necessary if you want to format in a certain way. For example, you might want to copy the command into a COBOL program and therefore the columns restrictions of 12 through 80 apply. After editing the command, you can run it again to make sure no editing mistakes were made.

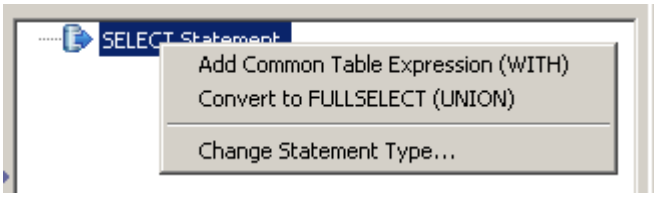


When satisfied, you can save this command or copy the command to paste into a program source. If pasting into a 2200 program source, remember to open the OS 2200 perspective first. Then you can open the appropriate source program if not already open.

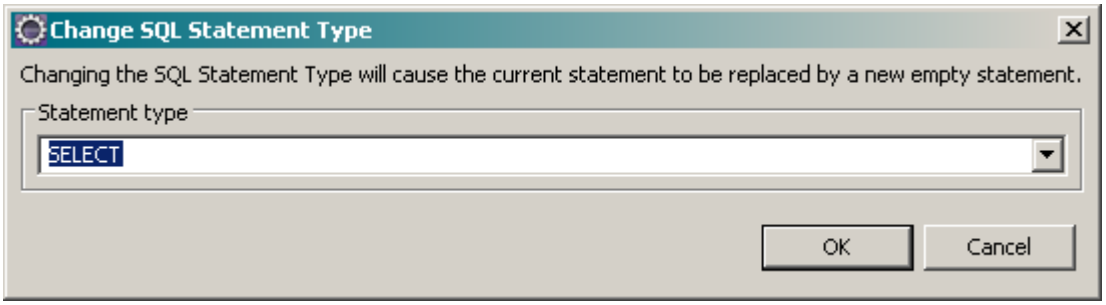
Changing SQL Commands

The above example referred to the SELECT command which is used by default.

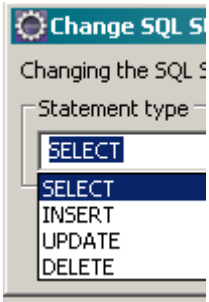
In the SQL Query Builder window, go to the middle right pane and right click on **SELECT Statement**:



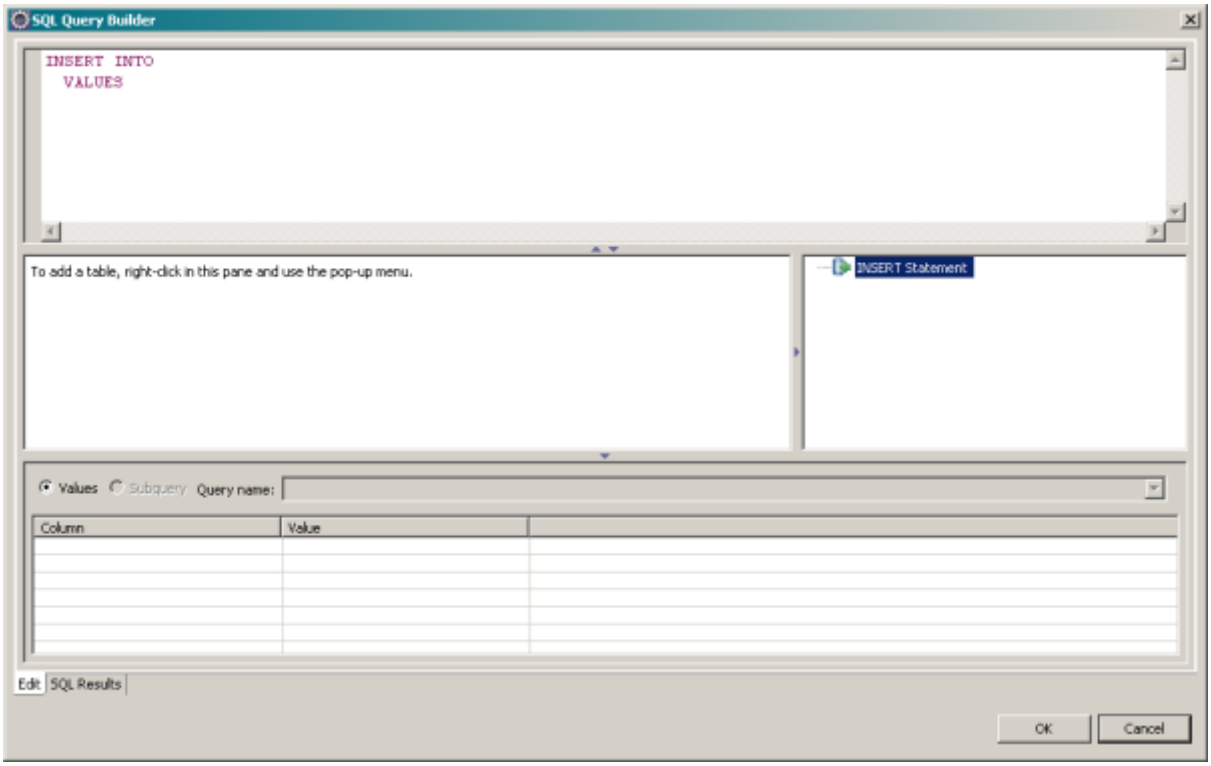
Click on **Change Statement Type.**



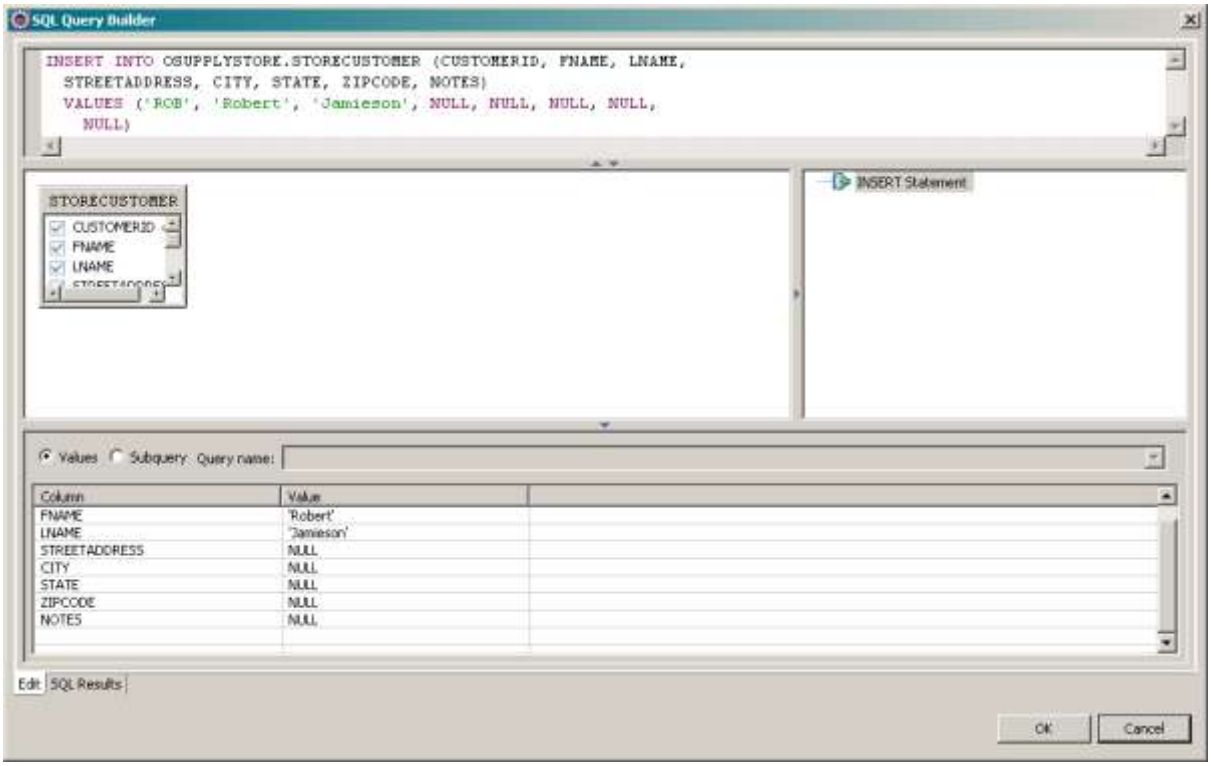
Expand the list box to view the available SQL commands.



Select the required command and then click **OK.** The SQL Query Builder changes for the selected command. In the following example, INSERT was selected.



Note the difference in the lower pane. You can also only have 1 table shown in the middle left pane since we are doing an INSERT.



If the column is defined to allow NULLs, then NULL is used as the default value but can be overwritten.

Avoiding the Schema Qualification

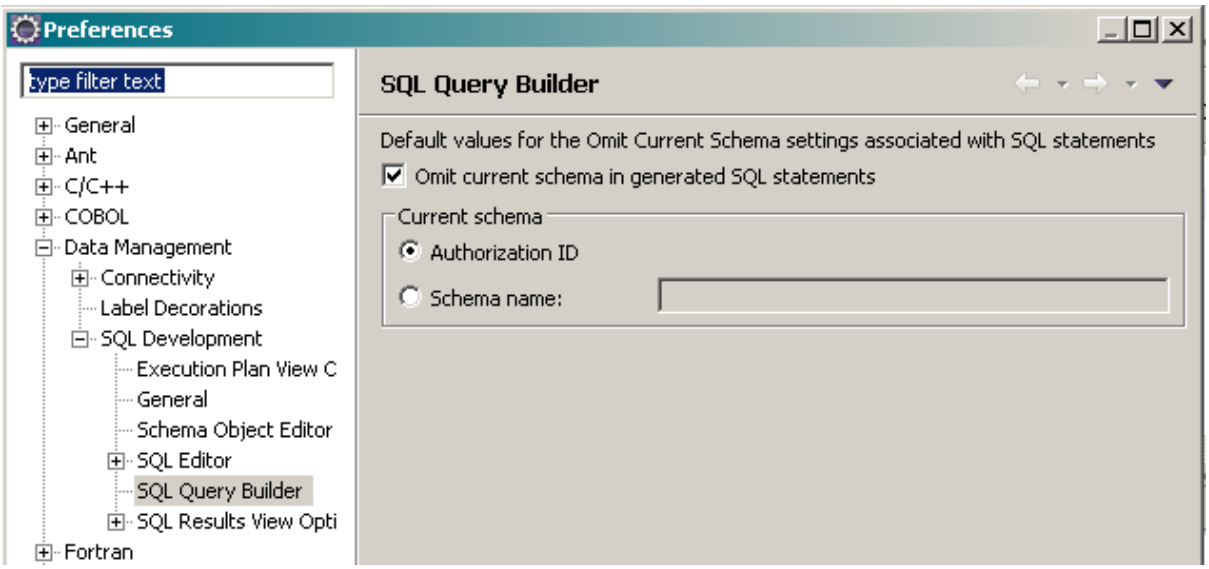
In the above examples, each field is qualified by schema, table and column e.g.

```
OSUPPLYSTORE.STORESUPPLIER.SUPPLIERID
```

Generally in 3GL development on the OS 2200, a developer would issue a USE DEFAULT QUALIFIER or USE DEFAULT SCHEMA statement to define the schema name. Both options are valid but experienced OS 2200 developers maybe more familiar with the former while new OS 2200 developers may know the latter. If the field is not qualified with a schema name, it uses this default qualifier value.

In many database systems, the default schema name is the user ID (or Authorization ID). However, in RDMS the default schema name is always "rdms". If you want to avoid using the schema name to qualify all table names, you must set the default schema (also called the QUALIFIER) for the connection. This is done by adding a connection property when you define the connection (e.g., schema=OSUPPLYSTORE). You can also tell the Eclipse SQL Builder to not add schema names. Check the "Omit current schema" box, then select the "Schema name" option and enter your schema name that you added to the connection properties (e.g., OSUPPLYSTORE).

Eclipse by default will include the schema in the field definition but it can be configured to omit the schema name. Go to **Preferences → Data Management → SQL Development → SQL Query Builder**.



Check the "Omit current schema" box.

The result is:

```
SELECT SUPPLIERID, COMPANYNAME
  FROM OSUPPLYSTORE.STORESUPPLIER
```

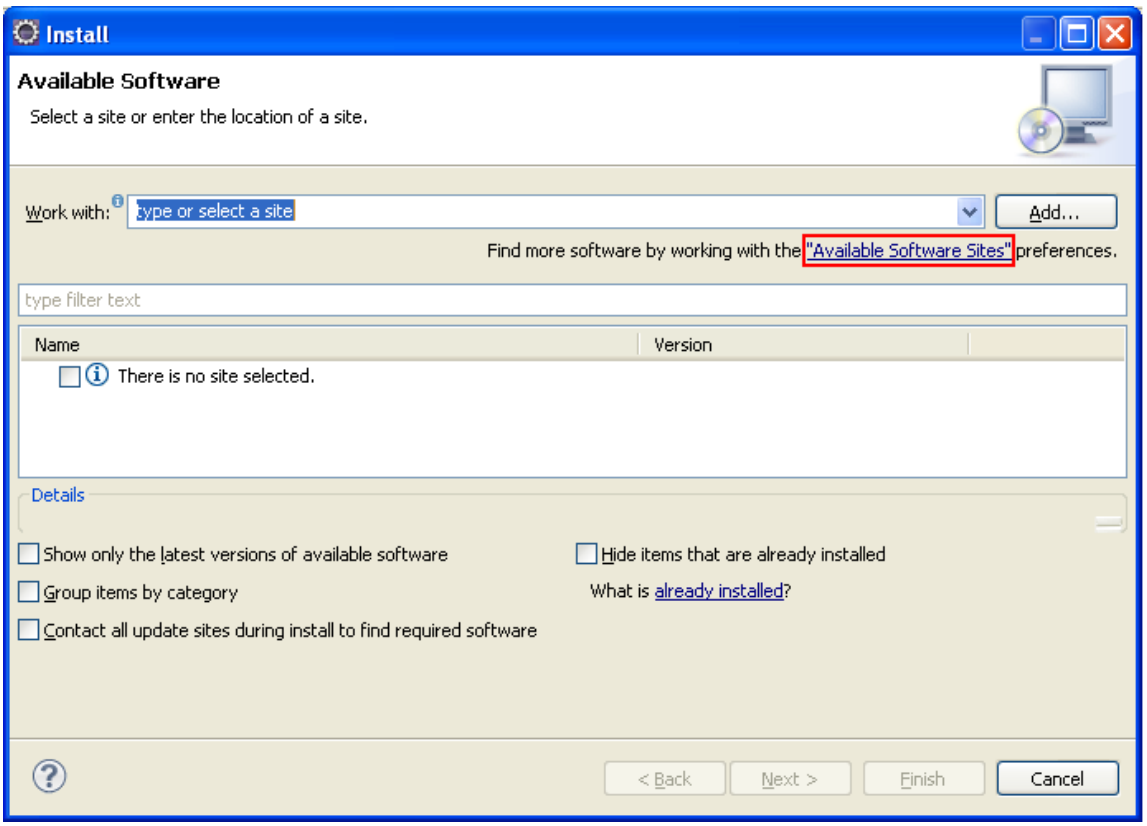
Obtaining Eclipse Updates

Unisys provides an Eclipse update site to provide updates to a released version. The update site is available at:

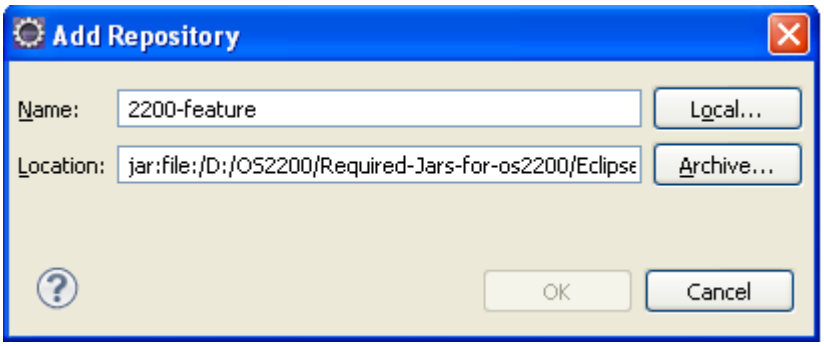
ftp://ftp.support.unisys.com/pub/2200/IDE/Eclipse-2200-3-7/com.unisys.ca.updatesite.

The following instructions provide the steps to process the update. Please follow the below process to update.

1. In the Eclipse window, Go to "*Help*" menu and choose "Install new Software". In the Install window, click on the "Available Software Sites" as shown below.



2. Now click the "Add" button to specify the location for the Eclipse 2200 All-in-One update site. This should bring up the "Add Site" window. Enter the URL specified above in the Location field.



3. We shall choose the OS2200-feature label from the drop-down box with the label "*Work with*" at the top of the dialog. (Note : We shall make sure that the "*Contact all update sites*

during install to find required software" option is UNCHECKED)

4. Now the two feature should be listed in the "Name" box, namely;
   - com.unisys.ca.feature
   - org.eclipse.cobol.feature

   We shall choose both these features and click on "Next" and in the following window accept the licenses and finish the update process.

If you are using C or C++ editors, you have to update Eclipse with mandatory plug-ins (CDT and JST) from the eclipse community (www.eclipse.org). Unisys also maintains these plug-ins on our product site. Please be advised that this is the old version of CDT and JST. In case any customer needs the latest plug-ins, they have to download it from www.elipse.org .

Clients can check for updates if there are any PLE's released. The client will know from product site if there are any PLE raised and what were the issue reported and the fix provided in the update. Unisys recommends for receiving alerts when new releases or updates are available.

Appendix A – Eclipse Quick Keys

Eclipse Shortcut Keys

CTRL+SHIFT+L		Show all shortcuts	
Editor Shortcuts		**Search**	
CTRL+D	Delete line	CTRL+H	Search
ALT+Up	Move line up (or down)	CTRL+J	Incremental
ALT+Left	Previous/next editor/file	CTRL+K	Find next
CTRL+SHIFT+O	Organize Imports	CTRL+SHIFT+K	Find previous
CTRL+1	Quick Fix		
CTRL+M	Maximize tab	**Debugging**	
CTRL+I	Correct Indentation	F5	Step Into
CTRL+SHIFT+F	Format Code	F6	Step Over
CTRL+L	Goto Line Number	F7	Step Return
CTRL+Q	Last Edit Location	F8	Resume
CTRL+T	Display type hierarchy	F11	Debug Last Launched
F4	Show type hierarchy	CTRL+SHIFT+B	Toggle Breakpoint
F3	Type declaration		
F2	Show info	**Opening**	
CTRL+O	Quick outline	CTRL+SHIFT+T	Open type

		CTRL+SHIFT+R	Open resource
CTRL+.	Next Error	CTRL+E	Show open editors
CTRL+,	Previous Error	CTRL+F6	Open editors
		CTRL+W	Close editor
CTRL+Space	Content Assist	CTRL+SHIFT+S	Save all
CTRL+SHIFT+Sp	Parameter Assist	CTRL+SHIFT+W	Close all
CTRL+/	Comment	CTRL+N	New project
ALT+/	Word Completion		
		Moving	
Refactoring		CTRL+F7	Move between views
ALT+SHIFT+R	Rename	CTRL+F8	Move between perspectives
ALT+SHIFT+L	Extract to local variable		
ALT+SHIFT+M	Extract to method	**Running**	
ALT+SHIFT+Y	Change method signature	CTRL+F11	Run application
ALT+SHIFT+Z	Undo refactoring	CTRL+ALT+P	Publish apps
		CTRL+ALT+R	Run In Appserver
		CTRL+ALT+D	Debug In Appserver
		ALT+SHIFT+X+T	

Quick Key	Function
Ctrl + /	Toggle Comment (*) indicator in col 7
Ctrl Space	Auto completion mode is invoked