Spectrum Software, Inc.
11445 Johns Creek Pkwy.
Suite 300
Duluth, GA – 30097
www.spectrumscm.com

Subject: **SpectrumSCM Plugin for the Eclipse Platform**

Original Issue Date: **February 2nd, 2005**
Latest Update Date: **October 2nd, 2007**

From: **Sudarshan N Raghavan**
sudarshan@spectrumsoftware.net
(770)448-8662

1.0 Introduction

Eclipse is an open source software development project dedicated to providing a robust, full-featured platform for the development of highly integrated tools The Eclipse Platform is an Integrated Development Environment (IDE) for anything and for nothing in particular. Most of the functionality provided by the platform is generic and additional tools are required to focus this generic functionality on something specific. The platform is built on the mechanism of *extension points* and *plugins*. Each tool provider writes his tool as a separate module called a *plugin*, which plugs into *extension points* defined by the platform or by other tool providers. The platform provides tool providers with well defined APIs, classes and methods for developing new plugins. The plugin operates on resources in the Eclipse workspace and provides tool specific UI to the Eclipse workbench. When the platform is launched it presents the user with an environment composed of all available plugins, thus facilitating seamless interoperability between various tools developed by different tool providers. In addition, the Eclipse Platform is Java based and thus offers cross-platform operation without limiting platform diversity.

Through its Team Support, the Eclipse Platform allows a project in the workspace to be placed under Source Configuration Management (SCM) with an associated team repository. The platform defines extension points and a repository provider API, which facilitates different SCM tool providers to develop plugins specific to their SCM systems. Since each SCM tool provider defines a separate plugin, multiple team repository providers can co-exist peacefully within the platform. The SpectrumSCM plugin for Eclipse provides the necessary functionality for associating projects in the Eclipse workspace with a SpectrumSCM repository and defines various SpectrumSCM specific operations for managing projects shared with a SpectrumSCM repository, thus providing a tight integration with the Eclipse platform. The plugin not only provides access to version control features but also to the integrated issue management and advanced branching features supported by SpectrumSCM.

This document discusses the SpectrumSCM plugin for Eclipse and describes how to use its features. It is assumed that the reader is familiar with the concepts and operations of the Eclipse Platform and SpectrumSCM. The reader is referred to http://www.eclipse.org for documents and articles related to the Eclipse platform and to http://www.spectrumscm.com for guides and tutorials related to SpectrumSCM.

2.0 Getting Started

This section discusses the steps for installing the SpectrumSCM plugin for Eclipse. Note, that while the original SpectrumSCM plugin was compatible with Eclipse versions back to v2.1, newer features require an Eclipse platform of at least version 3.1. At the time of writing Eclipse Europa (version 3.3) is fully functional with SpectrumSCM plugin version 31.26.1.

It is assumed that the user has installed Eclipse and has the necessary access to a running SpectrumSCM server.

Download and Install

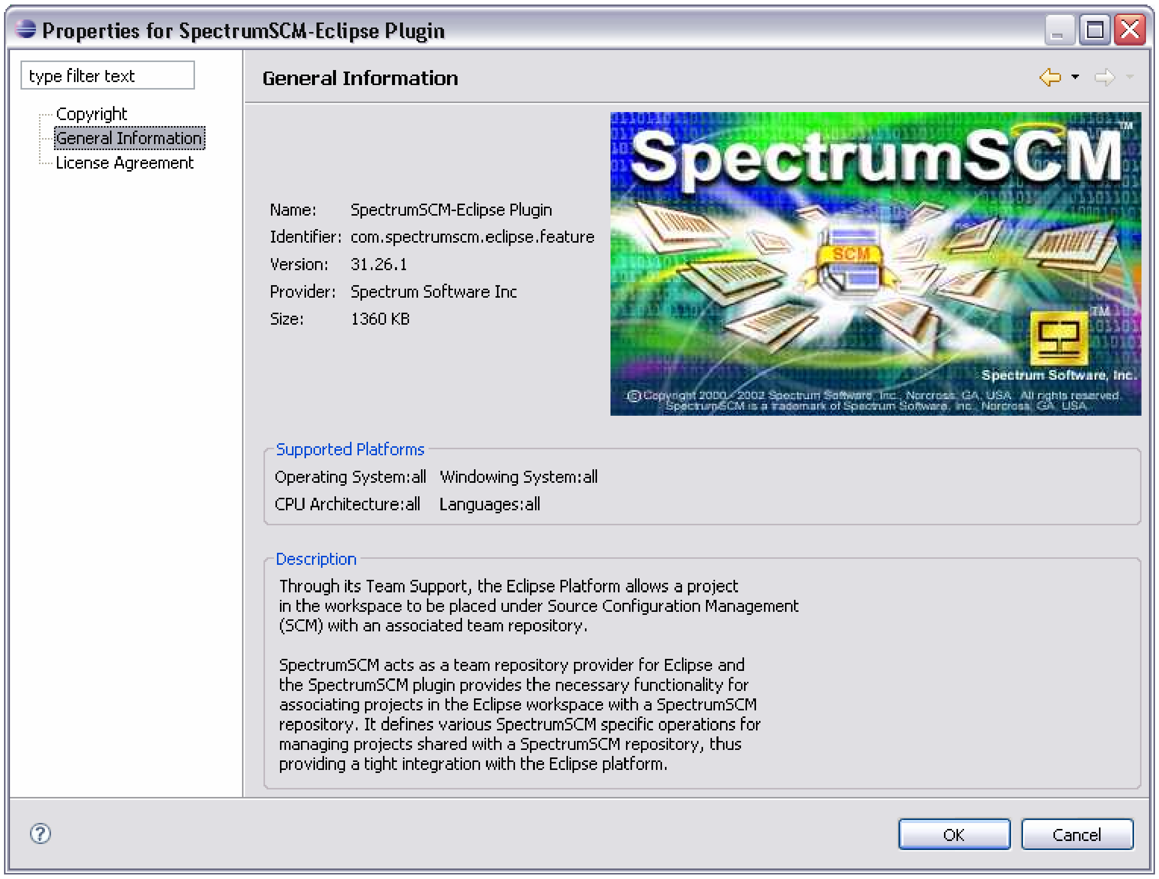
- Start Eclipse and select **Help→Software Updates→Find and Install …** from the menu bar. This opens the Install/Update dialog.

- Choose the **Search for new features to install** option and click **Next**. Add a new remote site entry for SpectrumSCM using the **New Remote Site** option and enter **http://www.spectrumscm.com/eclipse/plugins** in the URL field.

- Expand the tree for the SpectrumSCM entry in the tree viewer and select the SpectrumSCM Plugin option. Click **Next**.

- Select the version of the plugin to install and follow the on-screen instructions to complete the installation.

- If you are running an older version of the SpectrumSCM application, navigate to the URL **http://www.spectrumscm.com/eclipse/plugins** with your web-browser for the location of the version specific plugins. Note: At the end of the license agreement (visible during the installation dialog, and through the plugin properties panel), there is a statement as to which SpectrumSCM application version a particular Eclipse plugin version is meant for.



Name: SpectrumSCM-Eclipse Plugin
Identifier: com.spectrumscm.eclipse.feature
Version: 31.26.1
Provider: Spectrum Software Inc
Size: 1360 KB

© Copyright 2000, 2002 Spectrum Software, Inc., Norcross, GA, USA. All rights reserved. SpectrumSCM is a trademark of Spectrum Software, Inc., Norcross, GA, USA.

Supported Platforms
Operating System:all Windowing System:all
CPU Architecture:all Languages:all

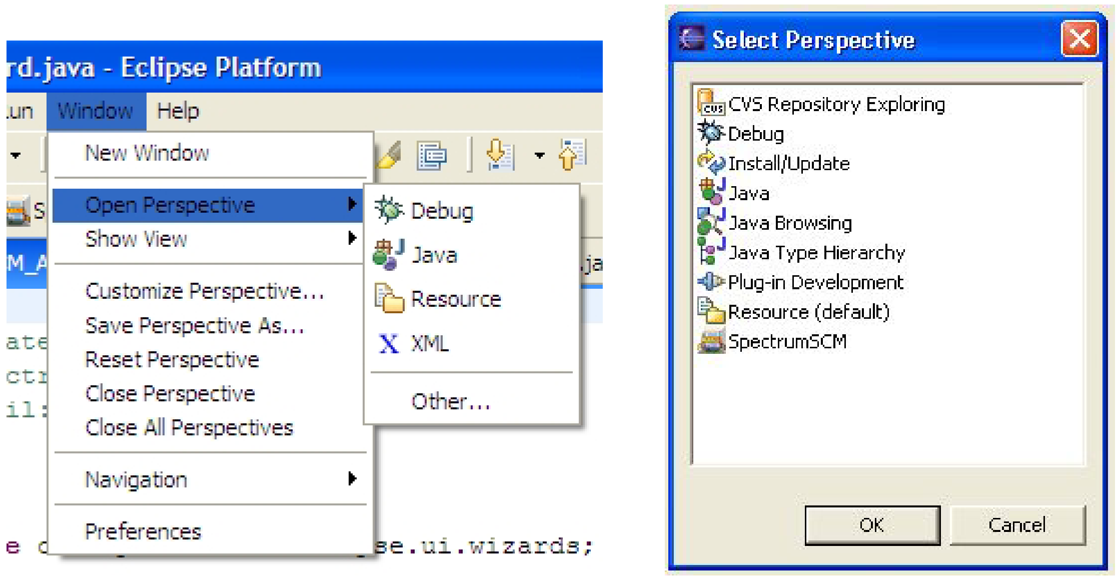
Description
Through its Team Support, the Eclipse Platform allows a project in the workspace to be placed under Source Configuration Management (SCM) with an associated team repository.

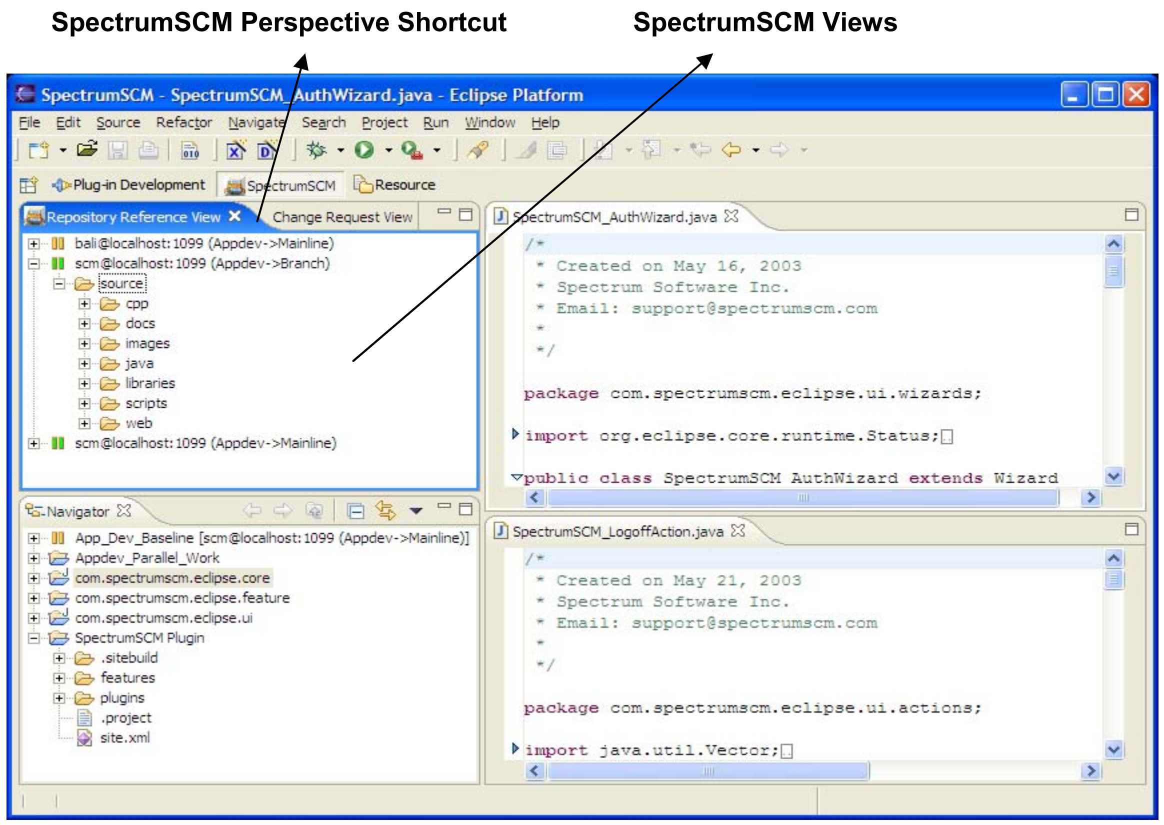
SpectrumSCM acts as a team repository provider for Eclipse and the SpectrumSCM plugin provides the necessary functionality for associating projects in the Eclipse workspace with a SpectrumSCM repository. It defines various SpectrumSCM specific operations for managing projects shared with a SpectrumSCM repository, thus providing a tight integration with the Eclipse platform.

Figure 1: Install the SpectrumSCM Plugin

3.0 SpectrumSCM Perspective and Views

The SpectrumSCM plugin for Eclipse adds a *SpectrumSCM Perspective* and two new views, the *Repository Reference View* and the *Change Request View*, to the Eclipse workbench. The SpectrumSCM Perspective can be opened by selecting the **Window→Open Perspective→Other…** menu item from the workbench menu bar. This presents a list of all perspectives currently defined for the workbench. Select SpectrumSCM and click OK to open the perspective. Figure 1 illustrates this operation and Figure 2 shows a snapshot of the SpectrumSCM perspective. Note that this operation adds a SpectrumSCM button to the shortcut bar.




SpectrumSCM Perspective Shortcut
SpectrumSCM Views
SpectrumSCM - SpectrumSCM_AuthWizard.java - Eclipse Platform
File
Edit
Source
Refactor
Navigate
Search
Project
Run
Window
Help
Plug-in Development
SpectrumSCM
Resource
Repository Reference View
Change Request View
bali@localhost:1099 (Appdev->Mainline)
scm@localhost:1099 (Appdev->Branch)
source
cpp
docs
images
java
libraries
scripts
web
scm@localhost:1099 (Appdev->Mainline)
Navigator
App_Dev_Baseline [scm@localhost:1099 (Appdev->Mainline)]
Appdev_Parallel_Work
com.spectrumscm.eclipse.core
com.spectrumscm.eclipse.feature
com.spectrumscm.eclipse.ui
SpectrumSCM Plugin
.sitebuild
features
plugins
.project
site.xml
SpectrumSCM_AuthWizard.java
SpectrumSCM_LogoffAction.java

```
/*
 * Created on May 16, 2003
 * Spectrum Software Inc.
 * Email: support@spectrumscm.com
 *
 */

package com.spectrumscm.eclipse.ui.wizards;

import org.eclipse.core.runtime.Status;

public class SpectrumSCM_AuthWizard extends Wizard
```

```
/*
 * Created on May 21, 2003
 * Spectrum Software Inc.
 * Email: support@spectrumscm.com
 *
 */

package com.spectrumscm.eclipse.ui.actions;

import java.util.Vector;
```

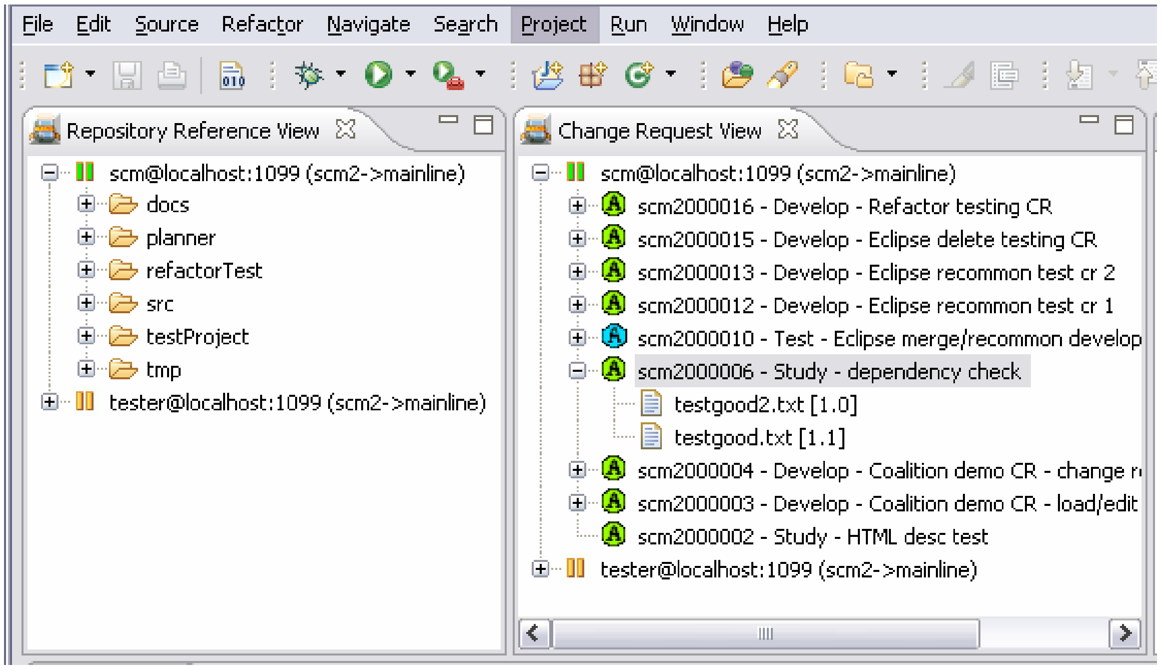
Figures 2 & 3: SpectrumSCM Perspectives

Each repository association in the SpectrumSCM plugin is defined by the 5-tuple *{server address, port number, user, project, generic}*. This 5-tuple is called a *Repository Reference*. Each Repository Reference is identified by the unique string, *user@host:port (project→generic)*, within the plugin. Users familiar with SpectrumSCM can readily understand the concept of a project and generic. In simple terms, a project refers to a separate space within a repository and a generic refers to a branch within a project. Each generic contains the files and resources that are under source control for that particular (project, generic) pair. The plugin allows different branches of the same SpectrumSCM project to be opened as independent Eclipse projects.

3.1 Repository Reference and Change Request Views

The Repository Reference View presents a logical view of all the resources under a particular Repository Reference. The view provides operations for browsing the resource tree, adding/removing Repository References, extracting files from the repository into a project, synchronizing a local workspace project with the Repository Reference and user authentication.

Figure 4: Repository Reference and Change Request Views


File Edit Source Refactor Navigate Search Project Run Window Help
Repository Reference View
Change Request View

Repository Reference View:

```
scm@localhost:1099 (scm2->mainline)
    docs
    planner
    refactorTest
    src
    testProject
    tmp
tester@localhost:1099 (scm2->mainline)
```

Change Request View:

```
scm@localhost:1099 (scm2->mainline)
    scm2000016 - Develop - Refactor testing CR
    scm2000015 - Develop - Eclipse delete testing CR
    scm2000013 - Develop - Eclipse recommon test cr 2
    scm2000012 - Develop - Eclipse recommon test cr 1
    scm2000010 - Test - Eclipse merge/recommon develop
    scm2000006 - Study - dependency check
        testgood2.txt [1.0]
        testgood.txt [1.1]
    scm2000004 - Develop - Coalition demo CR - change r
    scm2000003 - Develop - Coalition demo CR - load/edit
    scm2000002 - Study - HTML desc test
tester@localhost:1099 (scm2->mainline)
```

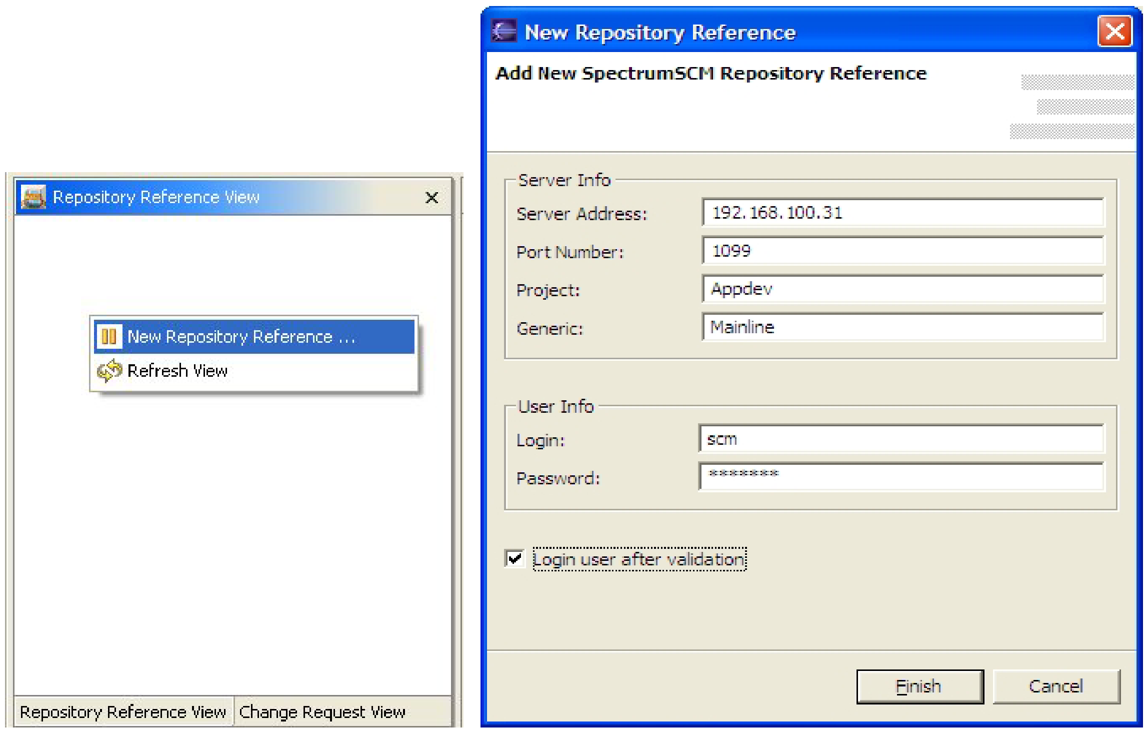
The Change Request View presents the list of all Change Requests (CR) that are currently assigned to the user associated with the particular Repository Reference. Under each CR, child-links are provided to all the files edited under that CR. The view provides operations for creating new CRs (self-assigned or general), progressing and re-assigning CRs, adding notes to CRs and viewing a CR's details. We shall deal with these operations in detail later in this document. Edit operations (Check-out and check-in) can be directly performed against the CR View Files.

3.2 Add a New Repository Reference

Once the plugin has been installed, start Eclipse and open the SpectrumSCM Perspective as described in the previous section.

- Choose the Repository Reference View and select **New Repository Reference …** from the context menu. This opens the New Repository Reference Wizard.

- Enter the required information and click **Finish**. The details will be validated against the server and a Repository Reference will be added to the view. Make sure that the server is up and running and that you enter the correct values for the various fields.


Repository Reference View
New Repository Reference ...
Refresh View
Repository Reference View
Change Request View
New Repository Reference
Add New SpectrumSCM Repository Reference
Finish
Cancel

Server Info

Server Address:	192.168.100.31
Port Number:	1099
Project:	Appdev
Generic:	Mainline

User Info

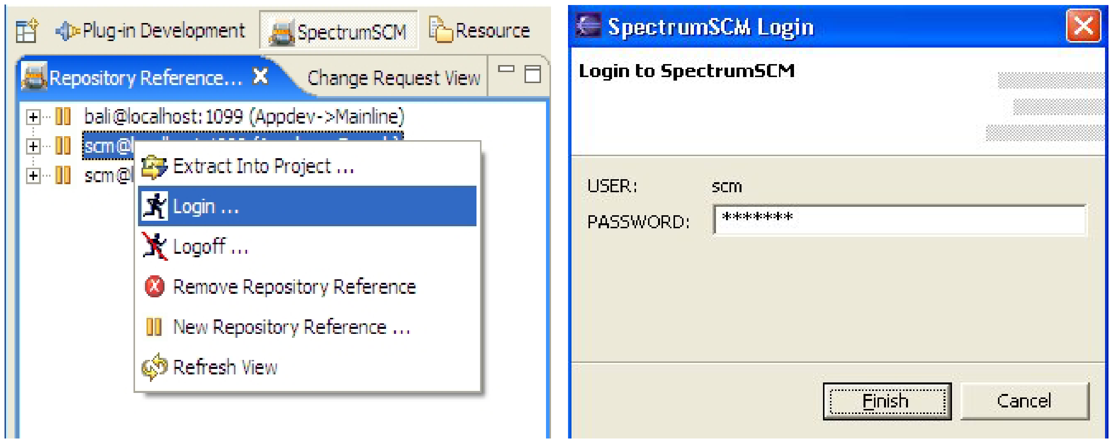
Login:	scm
Password:	********

☑ Login user after validation

Figure 5: Add a New Repository Reference

3.3 Login to the Server

Choose the Repository Reference from the view and select Login from its context menu. This opens the authentication wizard. Enter the password for the user associated with the Repository Reference and click **Finish**. If the authentication is successful, the decoration for the Repository Reference changes to reflect the fact that the user is logged in (green icon). At this stage, the user can browse the resource tree for the Repository Reference.

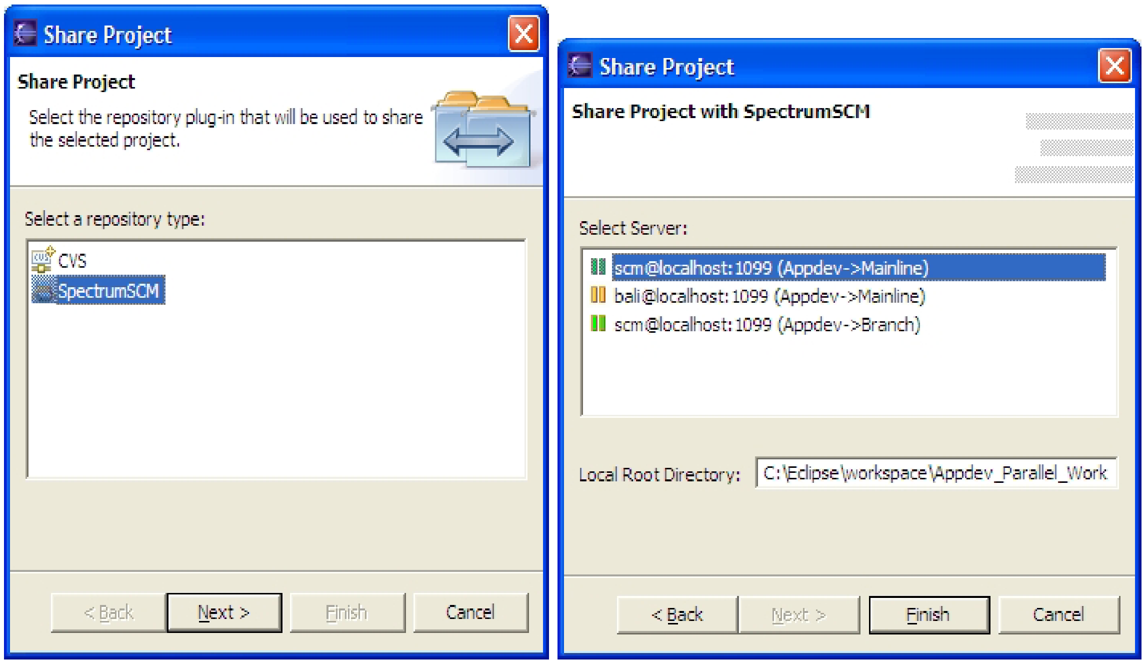


USER: scm
PASSWORD: *******

Figure 6: Login to the Server

3.4 Create a Project and Share it with SpectrumSCM

- Using the Navigator View, create a new project within the Eclipse workspace. The project could be a simple project, a Java project or any other project depending on the type of files present in the repository.
- Once the project has been created, select **Team→Share Project …** from the project's context menu. This opens the team configuration wizard for the project. Select SpectrumSCM and the proper Repository Reference from the wizard pages. Click **Finish** to associate the project with the Repository Reference. You will notice that the label decoration for the project changes to indicate that the project is shared with the Repository Reference.


Share Project

Share Project

Select the repository plug-in that will be used to share the selected project.

Select a repository type:

CVS
SpectrumSCM

Share Project

Share Project with SpectrumSCM

Select Server:

scm@localhost:1099 (Appdev->Mainline)
bali@localhost:1099 (Appdev->Mainline)
scm@localhost:1099 (Appdev->Branch)

Local Root Directory: C:\Eclipse\workspace\Appdev_Parallel_Work

Figure 7: Share Project with SpectrumSCM

<u>Local Root Directory</u>

The local root directory determines the relative path for files within the SpectrumSCM repository. By default, the plugin chooses the project path as the local root directory. However, users can set the local root directory to any valid path in the project path hierarchy. In the above figure, the local root directory has been specified as “**C:\Eclipse\Workspace\Appdev_Parallel_Work**”. If the project has a subdirectory called “**source**” under it, the relative path for files under the subdirectory will be “**/source**”, when the files are added into or checked out from the SpectrumSCM repository. However, if the user specifies the local root directory as “**C:\Eclipse\Workspace**”, the relative path will be “**/Appdev_Parallel_Work/source**”. Thus, the local root directory setting can be used to add files directly under the project, for example, the project.xml file. It also allows users to extract a project as a unit from the SpectrumSCM repository into the Eclipse workspace.

3.5 Extract Files into the Project

- Choose the Repository Reference or a folder within the resource tree in the Repository Reference View and select **Extract Into Project …** from the context menu. You will be presented with a wizard to choose the project to extract the files into.

- By default, the wizard lists all valid projects in the workspace. Choose the *Show projects shared with the repository reference* filter to list the projects shared with the Repository Reference. Select the project that you created in the previous step and click **Finish**

- The project's location on the file system will be used as the Local Root Directory by default. If a local root directory has been specified for the project (projects shared with SpectrumSCM), the user can extract files relative to the local root directory by selecting the **Use local root directory** option. The selected project will be examined to check whether the files to be extracted are already present. If present, the user will be presented with an option to overwrite or skip the files. Once all the files have been extracted, a result dialog will be presented giving the details of the extract operation.

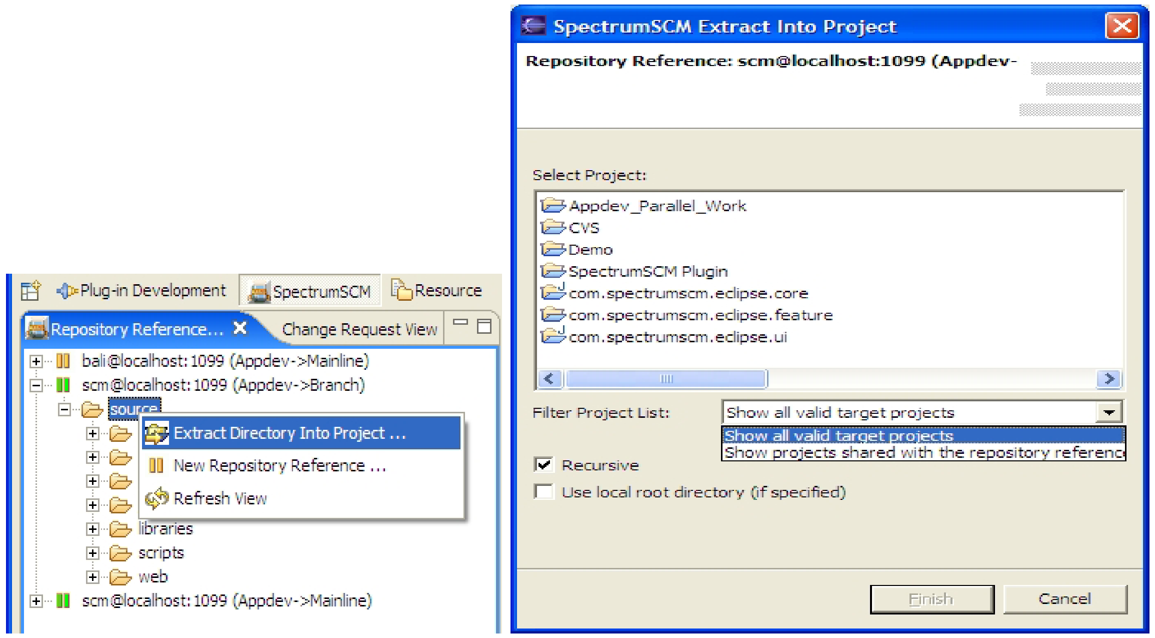

Plug-in Development
SpectrumSCM
Resource
Repository Reference...
Change Request View
bali@localhost:1099 (Appdev->Mainline)
scm@localhost:1099 (Appdev->Branch)
source
Extract Directory Into Project ...
New Repository Reference ...
Refresh View
libraries
scripts
web
scm@localhost:1099 (Appdev->Mainline)
SpectrumSCM Extract Into Project
Repository Reference: scm@localhost:1099 (Appdev-
Select Project:
Appdev_Parallel_Work
CVS
Demo
SpectrumSCM Plugin
com.spectrumscm.eclipse.core
com.spectrumscm.eclipse.feature
com.spectrumscm.eclipse.ui
Filter Project List:
Show all valid target projects
Show all valid target projects
Show projects shared with the repository referenc
Recursive
Use local root directory (if specified)
Finish
Cancel

Figure 8: Extract Files into Project

3.6 Logoff

The plugin authenticates users based on the 3-tuple *{user, server address, port number}*. When a user logs into to a server, he will consume a user license on the server and will be authenticated against all Repository References that use the same 3-tuple. To release the license, the user can log off from the server and login again when required. To log off from a server, select **Logoff** from the context menu for a Repository Reference in the Repository Reference View. You will be logged off from all Repository References using the same *{user, server address, port number}* 3-tuple.

4.0 Basic Operations

Once a project is shared with a SpectrumSCM Repository Reference, new SpectrumSCM specific operations are added to the Team, Compare With and Replace With menus for the project and the resources under it. This section discusses a few basic operations provided by the SpectrumSCM plugin. All operations involving the remote repository will prompt the user to log in (if they are not already logged in) before the operation can be executed.

4.1 View the Head Revision of a File

To view the head revision of a file, navigate to the file using the resource tree for the repository reference in the repository reference view. Double click the file to open the file. The editor settings defined within Eclipse for the file type will be used for opening the file. To define a custom editor for a file type, open the Eclipse Preferences wizard using the **Window → Preferences** option. Navigate to the **File Associations** node under the **Workbench** preferences and add a mapping for the file type.

4.2 Check Out a File

By default, SpectrumSCM uses the pessimistic model for version control. Every file has to be checked out against a Change Request before it can be modified. All files under a shared project are read-only by default. When a file is checked out for edit, it is locked by the user and cannot be checked out by other users until the file is checked in. Multiple users can checkout the same file using the common-concurrent option.

- To check out a file, select **Team→Check Out File …** from the file's context menu. Assuming that you have CRs assigned to you and that you have the requisite permissions, you will be presented with a list of CRs assigned to you.

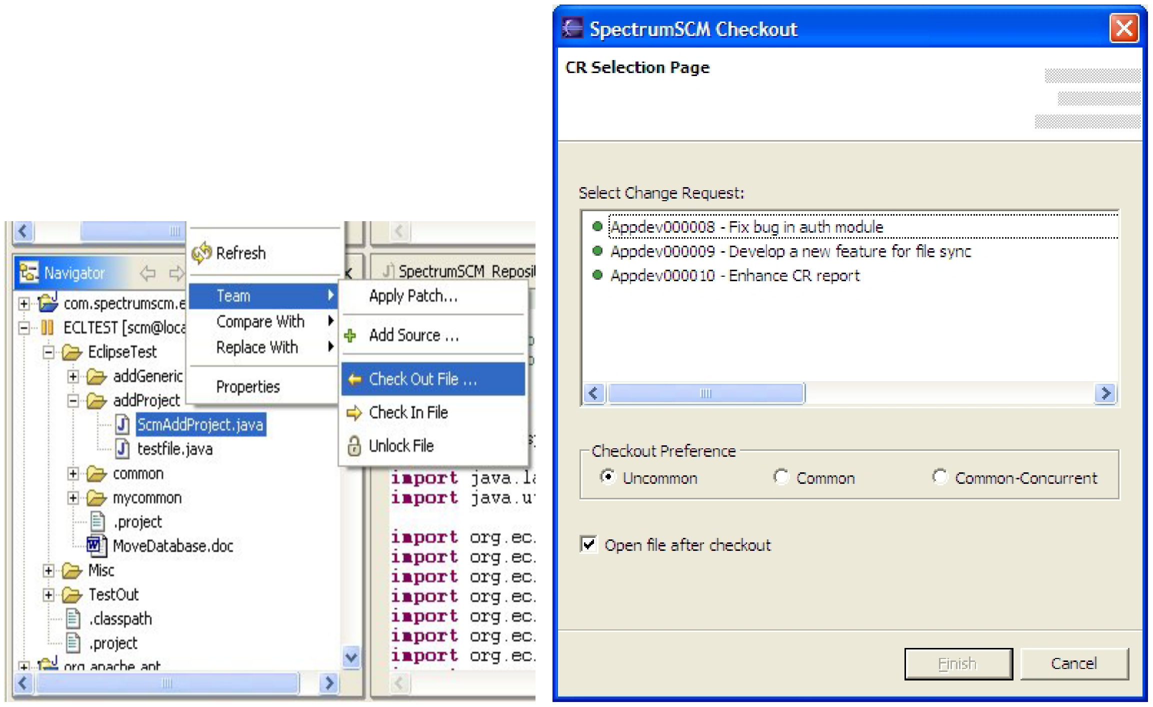
- If you have a CR selected in the CR View, it will be selected in the check-out dialog by default.

- Choose a CR and the Checkout Preference. SpectrumSCM supports the concept of file sharing across multiple generics. When a file is checked out Common, the sharing is kept intact. When a change made to the file and the file is checked in, the change will be reflected across all branches the file is common with. An Uncommon checkout isolates the file changes to the current generic. A Common-Concurrent checkout establishes a "soft-lock" on the file, thus allowing other users to check out the same file against a different CR.

- Click **Finish**. A result dialog will be presented giving the details of the check out operation. If the file is successfully checked out, a ✓ icon will be added to the file's decoration and the label decoration for the file will change to indicate that it has been checked out for edit.

- At this stage, the file becomes writable and the user can make modifications to the file.

- The **Open file after checkout** option can be used to open the file for edit immediately after the checkout operation. The editor preference for the file extension, defined within Eclipse, is used for this operation.



Select Change Request:
- Appdev000008 - Fix bug in auth module
- Appdev000009 - Develop a new feature for file sync
- Appdev000010 - Enhance CR report

Checkout Preference: [✓ Uncommon] Common Common-Concurrent

[✓] Open file after checkout

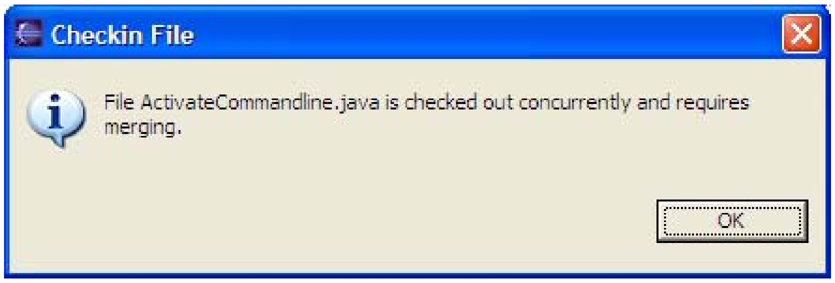
Figure 9: Check Out a File

4.3 Check In or Unlock a File

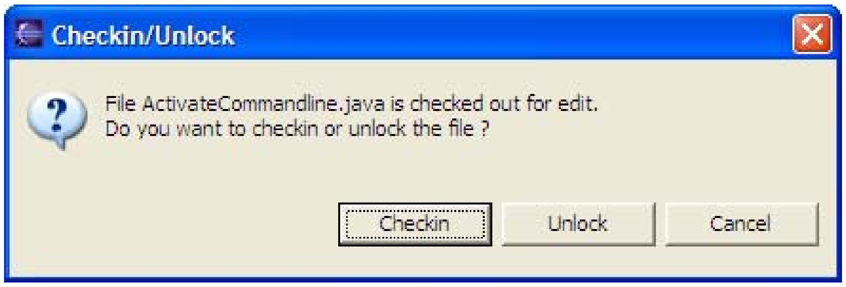
Once the necessary changes have been made to the file, the file should be checked back in to the repository. To check in a file select **Team→Check In File** from the file's context menu. If the check in is successful, the decorations for the file will change to indicate that it has been checked in. The file will be unlocked if there are no differences between the file and the latest version of the file in the repository. Before checking in your work, it is always a good practice to verify that your work is compatible with the changes that other users have made in parallel to your work.

Unlocking a file is similar to the Check In operation. Select **Team→Unlock File** from the file's context menu to unlock a checked out file.

If a file has been checked out common-concurrent, and another user has made changes to the same file and checked in his changes, the checkin operation will automatically recognize that the file has changed in the repository. At this point, if the file is an ASCII file, the user will be prompted to merge his changes using the Eclipse diff/merge editor. Once the necessary changes have been reviewed and made, and the user closes the diff/merge editor, he'll be prompted to checkin or unlock the file.

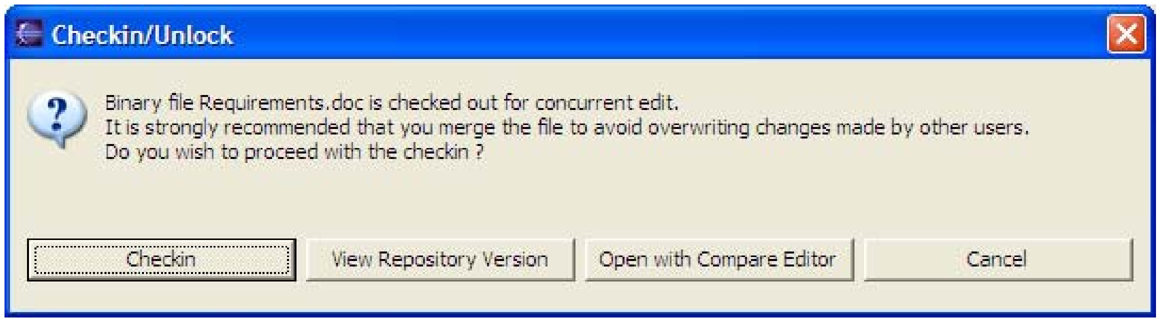


File ActivateCommandline.java is checked out concurrently and requires merging.



File ActivateCommandline.java is checked out for edit.
Do you want to checkin or unlock the file ?

If the file is a binary file, the user will be presented with the following options:

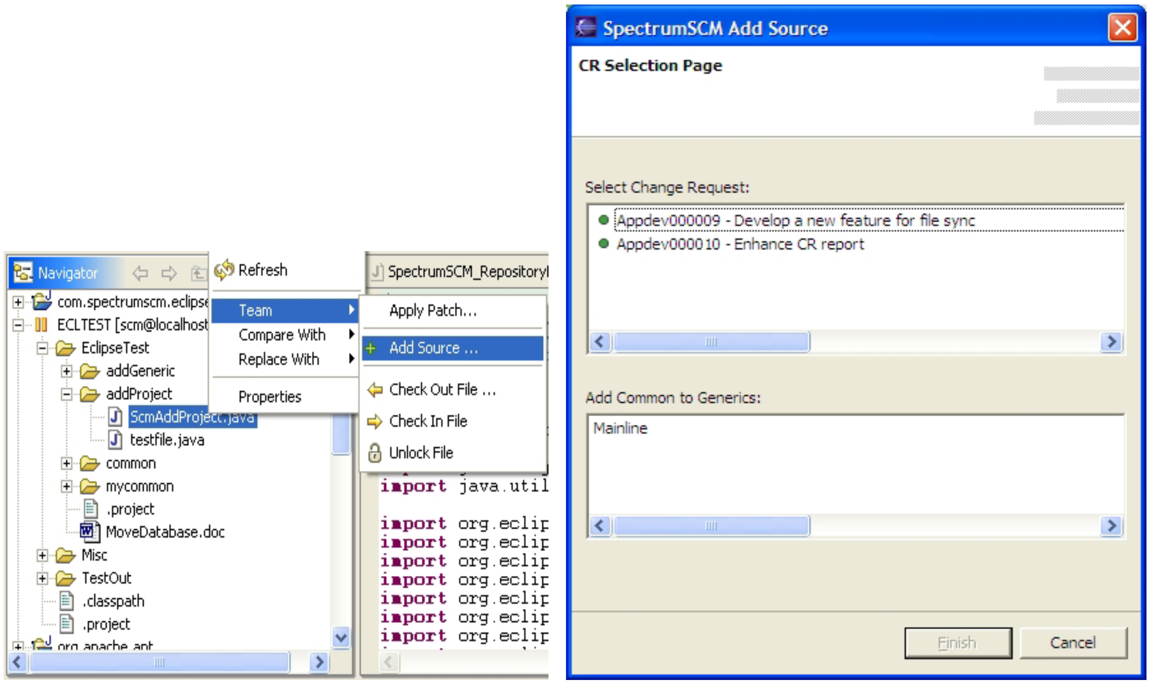


Binary file Requirements.doc is checked out for concurrent edit.
It is strongly recommended that you merge the file to avoid overwriting changes made by other users.
Do you wish to proceed with the checkin ?

The **Checkin** option will update the top version of the file in the repository with the file in the user's workspace. Use this option wisely to prevent overwriting changes made by other users. Note that in case of an erroneous checkin, the previous version is still available and thus no information is lost overall. The **View Repository Version** option will allow the user to view the top version of the file in the repository so that they can review changes made by other users. **Open with Compare Editor** will try to open the file with the default diff/merge editor. Most diff/merge editors do not support differencing of binary files.

4.4 Add a New File to the Repository

During the course of your development work, you may add new files to you local workspace and work in an isolated fashion from the files in the repository. Once you are done with your work, you can determine which files need to go into the repository. To add a file to the repository, select **Team→Add Source …** from the file's context menu. You will be presented with a dialog to choose a CR for the operation and the list of generics to add the file common with. When a file is added common, it is shared across all the generics that have been selected for the add operation. If the generics selection is empty, the file will be added only to the current generic. Select the appropriate CR and click **Finish** to add the file. The relative path for the file in the SpectrumSCM repository is determined based on the local root directory setting for the project.



```
import java.util

import org.eclip
import org.eclip
import org.eclip
import org.eclip
import org.eclip
import org.eclip
import org.eclip
```

Appdev000009 - Develop a new feature for file sync
Appdev000010 - Enhance CR report

Mainline

Figure 10: Add a New File to the Repository

4.5 Replace a File

The Replace With menu provides the menu items to replace a local file with the head revision or a specific version of the file from the repository. To replace a file with the head revision, select **Replace With→Head Revision** from the file's context menu. If the **Replace With→Version …** menu option is selected, you will be presented with a list of all versions for the file in the repository. Select a version and click **Finish** to replace the file in the workspace with the selected version. By default, Eclipse provides the **Replace With Local History** feature for files in the workspace.

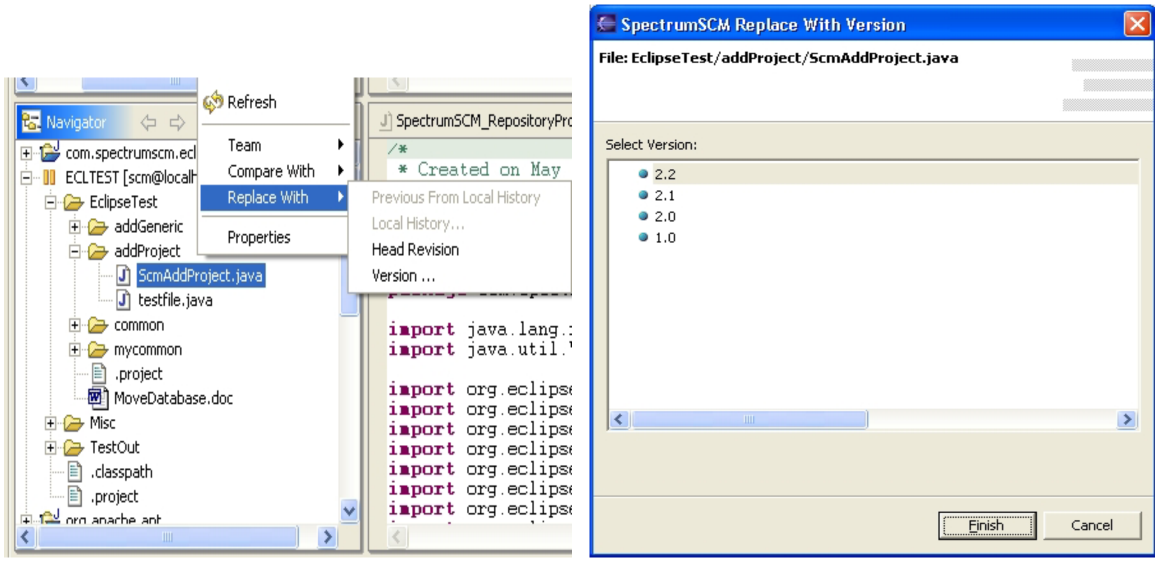


Figure 11: Replace a File

4.6 Ignoring Resources from Version Control

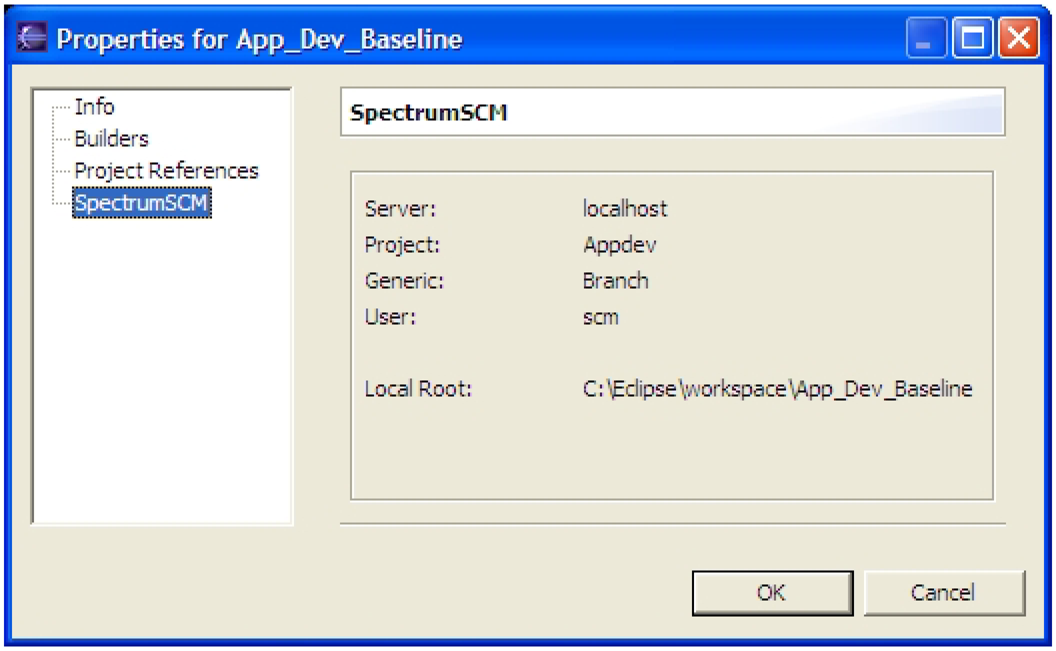
By default, the plugin excludes derived resources like class files from version control. The plugin also honors the ignored resources settings under the team preferences. To ignore a particular pattern from version control, open the Eclipse Preferences wizard using the **Window → Preferences** option. Navigate to the **Ignored Resources** node under the **Team** preferences and add an entry for the pattern.

4.7 Refreshing Views

The SpectrumSCM plugin is based on the *pull* model i.e. the user has to pull information from the repository as and when required. The server never tries to *push* any information to the client. To keep the views in sync with the repository and with the local file system, the user should periodically refresh the views using the **Refresh** menu option. The Refresh option in the Repository Reference and Change Request Views query the repository to determine whether any changes have been made to the resource tree or the CR list since the last refresh operation. The plugin automatically refreshes these views when the user invokes file-related or CR-related operations that affect the contents of these views.

4.8 View the SpectrumSCM Properties for a Project

Once a project has been shared with a SpectrumSCM repository, the SpectrumSCM specific properties for the project can be accessed using the Properties option under the context menu for the project. Select the SpectrumSCM option from the properties wizard to view the details for the shared repository reference. This feature is useful if the user wants to quickly find out the local root directory settings for a project. To change the local root directory, the user can disconnect and share the project again with the repository reference.



Server:	localhost
Project:	Appdev
Generic:	Branch
User:	scm
Local Root:	C:\Eclipse\workspace\App_Dev_Baseline

Figure 12: Server Properties

4.9 Disconnect a Project

When a project no longer needs to be shared with the Repository Reference, it can be disconnected by selected the **Team→Disconnect** option from the project's context menu. All checked out files under the project have to be checked in before a project can be disconnected. The disconnect action does not automatically logoff the user from the server. The user has to manually logoff if he/she wants to release the license on the server.

4.10 Remove a Repository Reference

When a Repository Reference becomes invalid or is no longer needed, it can be removed using the **Remove Repository Reference** menu option in the Repository Reference View. A Repository Reference cannot be removed if there are projects in the workspace that are shared with the Repository Reference.

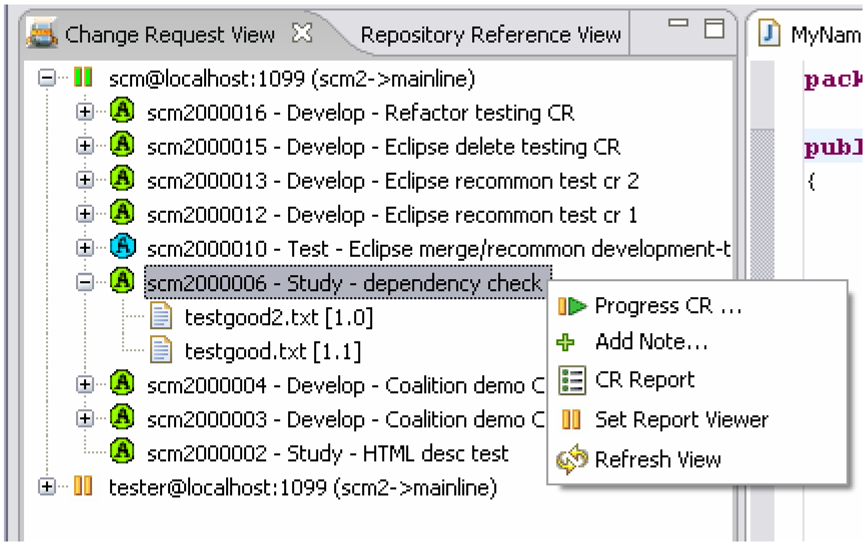
This operation automatically logs off the user from the server associated with the Repository Reference.

5.0 Change Request View Related Operations

In addition to listing all the CRs assigned to the user associated with a Repository Reference, the Change Request View provides operations for viewing the details of a CR, progressing CRs and creating new CRs.

Selecting a CR in the CR view also establishes this CR as the default selection for any subsequent edit operations. This statement applies to operations performed from the CR View, Navigator, Package Explorer or any other team provider enabled view.

A CRs relationship to files currently being edited, or previously edited, is also shown. This allows quick and easy "continuation" type edits, where the user wishes to continue work that has been previously checked-in.



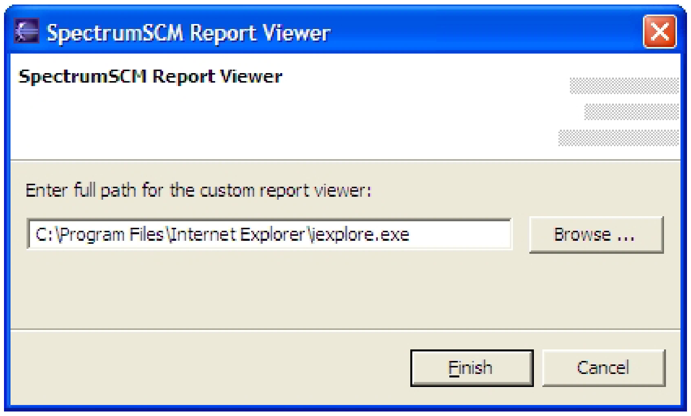
scm@localhost:1099 (scm2->mainline)
scm2000016 - Develop - Refactor testing CR
scm2000015 - Develop - Eclipse delete testing CR
scm2000013 - Develop - Eclipse recommon test cr 2
scm2000012 - Develop - Eclipse recommon test cr 1
scm2000010 - Test - Eclipse merge/recommon development-t
scm2000006 - Study - dependency check
testgood2.txt [1.0]
testgood.txt [1.1]
scm2000004 - Develop - Coalition demo C
scm2000003 - Develop - Coalition demo C
scm2000002 - Study - HTML desc test
tester@localhost:1099 (scm2->mainline)

5.1 View the Details of a CR

By default, running a CR report to view the details of a Change Request will open in an internal Eclipse HTML viewer (this is new under release 2.6). The Change Request Report includes the full history of the CR, when it was created, what the full description of the issue is, and what state transitions (approvals, rejections, code reviews etc (as determined by your SpectrumSCM life-cycle)). Any attribute information can also be seen through the report.

The user can also double click on a CR in the Change Requests View to launch the report for the CR.

If you wish to view the report in another HTML viewer, this can be specified through the "Set Report Viewer" option. The Report Viewer dialog will allow the user to choose a HTML viewer for displaying reports and save the setting. On Windows based systems, the program chooser automatically defaults to Internet Explorer's installation directory.


SpectrumSCM Report Viewer

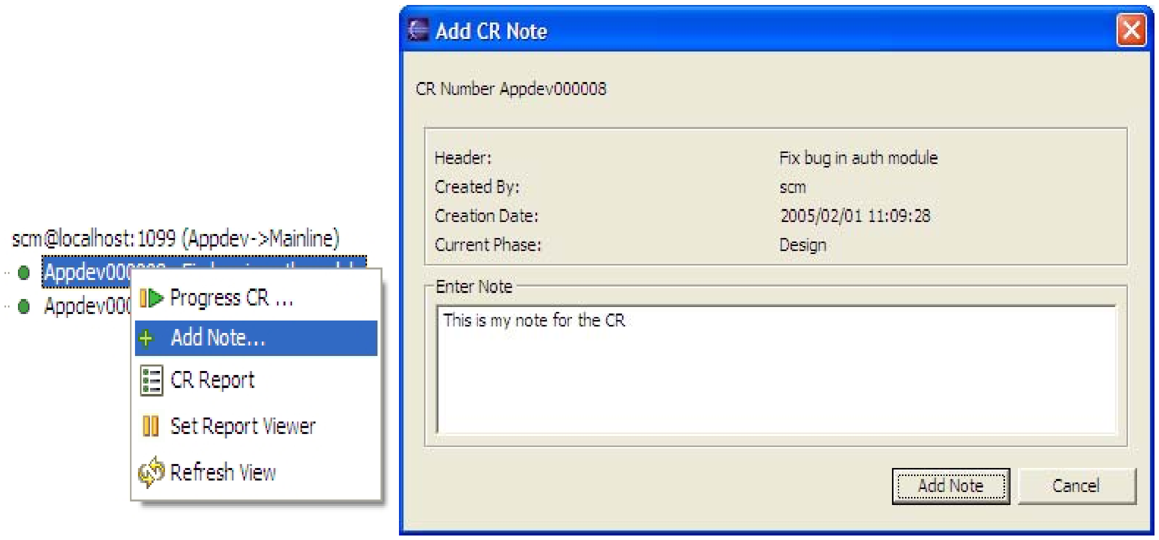
Enter full path for the custom report viewer:

C:\Program Files\Internet Explorer\iexplore.exe

Figure 13: Change Request Details

5.2 Add a Note to a CR

To add a note to a change request, select the Add Note … option from the context menu for the CR. You will be presented with a dialog that gives basic details for the selected CR and allows you to enter the notes. Click Add Note to finish the operation.


scm@localhost:1099 (Appdev->Mainline)

CR Number Appdev000008

Header:	Fix bug in auth module
Created By:	scm
Creation Date:	2005/02/01 11:09:28
Current Phase:	Design

Enter Note

This is my note for the CR

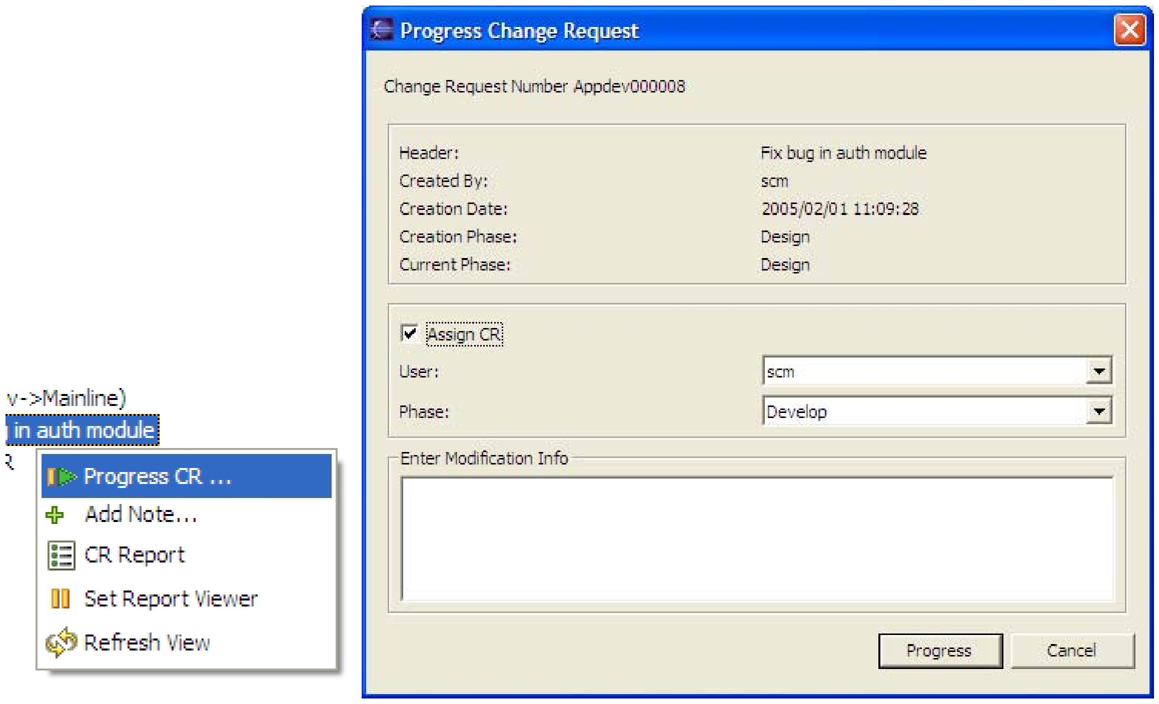
Figure 14: Add a Note to a Change Request

5.3 Progress or Re-assign a CR

To progress or re-assign a CR, select **Progress CR …** from the CR's context menu. You will be presented with a dialog that gives basic details regarding the CR and allows you to enter the assignment and modification info.

Progressing a change request in SpectrumSCM refers to promoting the CR to the To Be Assigned (TBA) super-state. This allows a responsible entity (e.g. a project manager) to review the CR before assigning the CR to the next person in the next phase, as dictated by the project's process life cycle. If a user has the necessary permissions, he will be able to directly assign the CR to the next user and phase without promoting it to the TBA state. Self-assign permissions will allow the user to assign the CR to himself in another phase, while Assign to User permissions will allow him to assign the CR to a different user. The permissions can be set using the Roles and Permissions feature in SpectrumSCM (**Chapter 5 – SpectrumSCM user guide**).

Based on the permissions for the user, the **Assign CR** checkbox will be enabled or disabled. If the user only has self-assign permissions, the User combo box will contain the user ID as the only entry. Select the Assign CR option, select the next user and phase, and click Progress to assign the CR. Uncheck the option (if enabled) and click Progress to promote the CR to the TBA state. The CR view is automatically refreshed after the operation. Unless the CR is self-assigned, it will disappear from the CR View, since the view only lists the CRs that are currently assigned to the user.



Change Request Number Appdev000008

Header:	Fix bug in auth module
Created By:	scm
Creation Date:	2005/02/01 11:09:28
Creation Phase:	Design
Current Phase:	Design

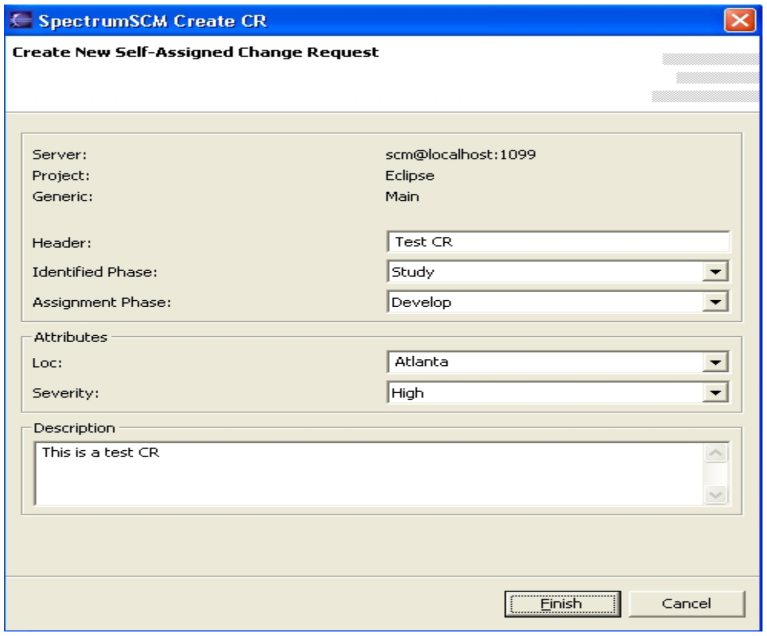
☑ Assign CR
User: scm
Phase: Develop

Figure 15: Progress/Assign a Change Request

5.4 Create a New Self-Assigned CR

A user with the necessary permissions can create a self-assigned CR using the **Create Self-Assigned CR …** menu option. This menu option is available under the context menu for the Repository Reference inside the Change Request View.

The operation queries the repository to determine the fields required to create a CR and presents a dialog to enter values for these fields. Enter values for the header, lifecycle phases, CR attributes and description fields and click **Finish** to create the CR. If a CR attribute is editable, the corresponding combo box will also be editable. If an attribute is forced, it will prompt the user to choose a value for the attribute before the CR can be created. This operation automatically refreshes the Change Request View.



Server: scm@localhost:1099
Project: Eclipse
Generic: Main
Header: Test CR
Identified Phase: Study
Assignment Phase: Develop
Loc: Atlanta
Severity: High
This is a test CR

Figure 16: Create a Change Request

5.5 Create a New General CR

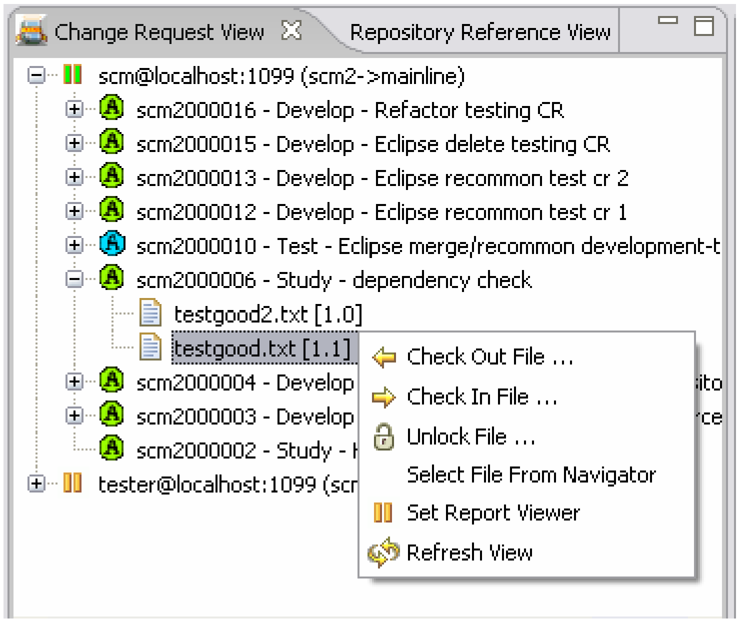
A user with the necessary permissions can create general CRs i.e. CRs either not assigned (To Be Assigned), or CRs assigned to someone else. Use the **Create General CR …** menu option available under the context menu for the Repository Reference in the Change Request View.

The operation queries the repository to determine the fields required to create a CR and presents a dialog to enter values for these fields. Enter values for the header, lifecycle phases, CR attributes and description fields as with a self-assigned CR. However, in addition, you can either leave the "**Assign To**" user as "**None**", or select the desired recipient of this issue/task.

Click **Finish** to create the CR. If a CR attribute is editable, the corresponding combo box will also be editable. If an attribute is forced, it will prompt the user to choose a value for the attribute before the CR can be created. This operation automatically refreshes the Change Request View.

5.6 CR View – File operations

When a file is checked-out for edit, or subsequently checked-in, that action is recorded against the Change Request. This information is displayed in the Change Request View and allows further operations –


Change Request View
Repository Reference View
scm@localhost:1099 (scm2->mainline)
scm2000016 - Develop - Refactor testing CR
scm2000015 - Develop - Eclipse delete testing CR
scm2000013 - Develop - Eclipse recommon test cr 2
scm2000012 - Develop - Eclipse recommon test cr 1
scm2000010 - Test - Eclipse merge/recommon development-t
scm2000006 - Study - dependency check
testgood2.txt [1.0]
testgood.txt [1.1]
scm2000004 - Develop
scm2000003 - Develop
scm2000002 - Study
tester@localhost:1099 (scr
Check Out File ...
Check In File ...
Unlock File ...
Select File From Navigator
Set Report Viewer
Refresh View

From this display, for example, you can see that CR scm2000006 has edited 2 files, testgood.txt and testgood2.txt and that the latest versions touched are versions 1.1 and 1.0 respectively.

By selecting a file, its full path will be shown in the Eclipse message bar in the lower left of the workbench window.

By right-clicking on a file the context menu is presented as shown above.

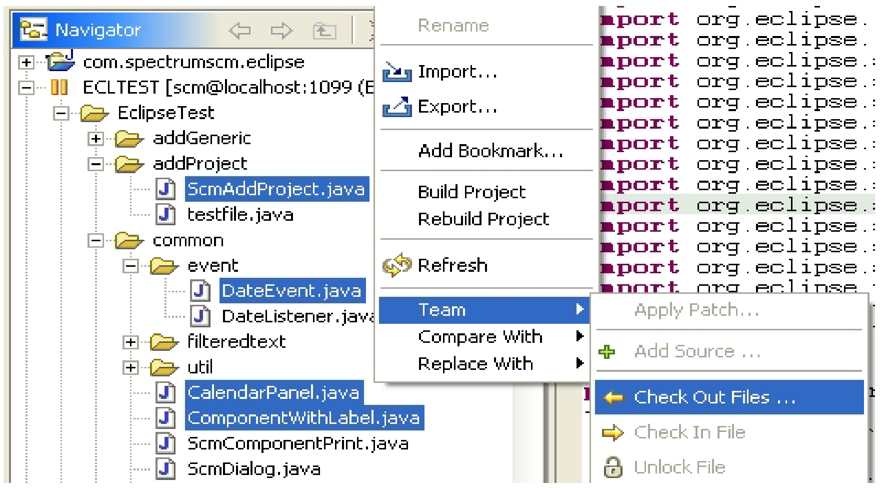
Selecting the “**File From Navigator**” option will open the current project tree to the selected file (if available). If the Package Explorer View is being used instead of the Navigator, the file tree will be opened there instead.

6.0 Advanced Operations

In addition to providing the above-mentioned basic operations, the SpectrumSCM plugin provides advanced features for synchronizing files with the repository and for checking out, checking in and unlocking files in bulk. The following sections explain these features in detail.

6.1 Multiselect Check Out

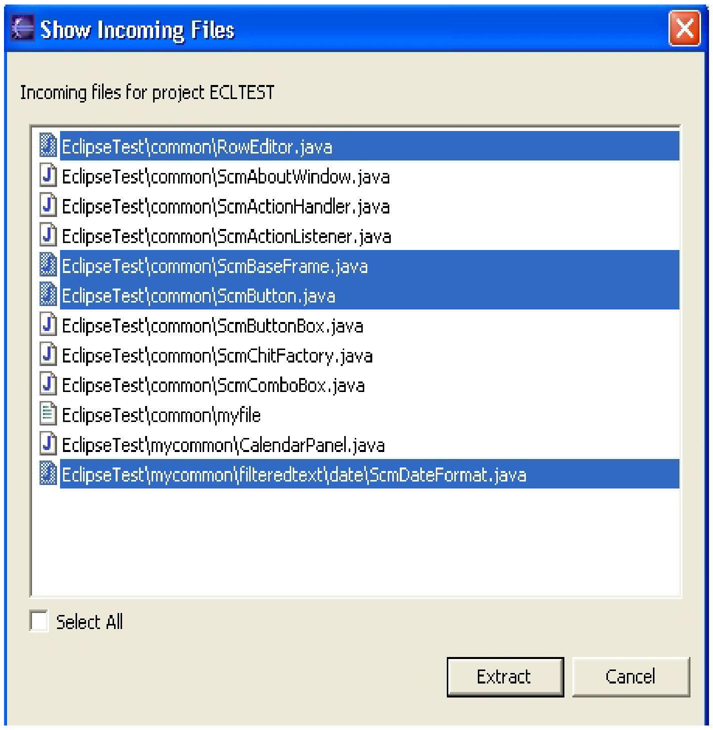
To check out multiple files, select the required files in the appropriate view and choose **Team→Check Out Files …** from the context menu for the files. You will be presented with a table listing all the CRs assigned to you. Select the CR for the operation and click **Finish** to check out the selected files. The details of the check out operation will be presented after all the files have been checked out. Note that the decorations for the selected files will be updated to indicate that they have been checked out.



6.2 Show Incoming Files

For a given Repository Reference, this feature compares the files in the repository with the files under a project in the workspace and gives a list of all files that need to go into the project. A user can select the required (or all) files from this list and extract these files into the project.

To display the list of all incoming files for a particular project or folder, select **Team → Show Incoming Files …** from the context menu for the project or a folder under the project. The project should be shared with a SpectrumSCM repository for this option to be enabled.



Incoming files for project ECLTEST

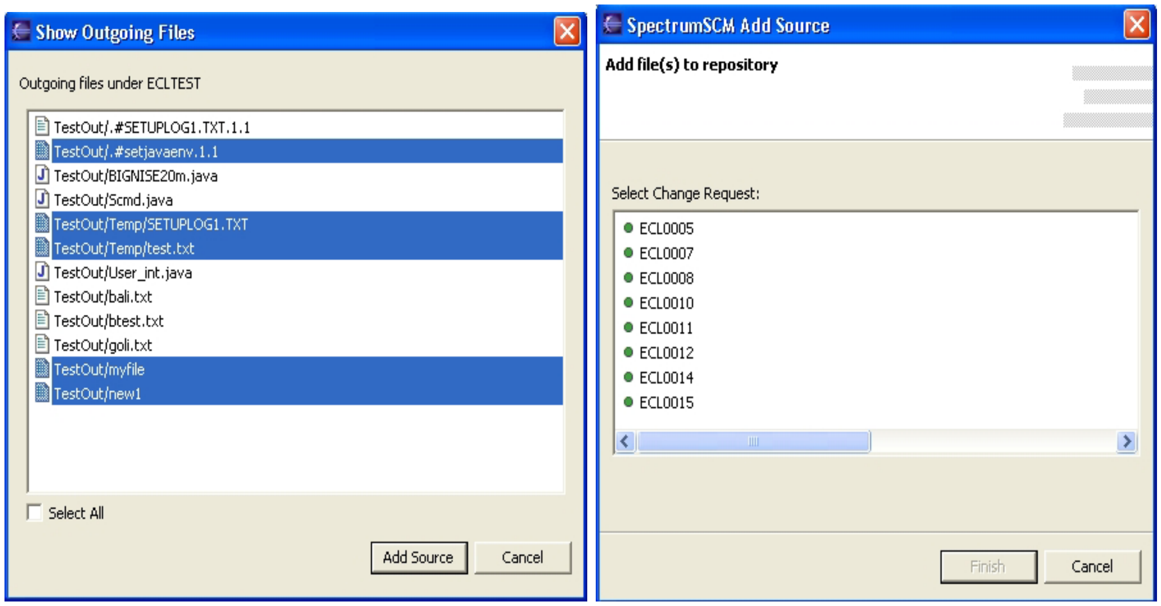
EclipseTest\common\RowEditor.java
EclipseTest\common\ScmAboutWindow.java
EclipseTest\common\ScmActionHandler.java
EclipseTest\common\ScmActionListener.java
EclipseTest\common\ScmBaseFrame.java
EclipseTest\common\ScmButton.java
EclipseTest\common\ScmButtonBox.java
EclipseTest\common\ScmChitFactory.java
EclipseTest\common\ScmComboBox.java
EclipseTest\common\myfile
EclipseTest\mycommon\CalendarPanel.java
EclipseTest\mycommon\filteredtext\date\ScmDateFormat.java

Figure 17: Show Incoming Files

Files under the selected project will be compared with the files in the repository and the user will be presented with the list of all files that exist in the repository but not under the selected project. The local root directory setting for the project will be used for determining the relative path for a file. The user can multi-select files from the list or select all the files using the **Select All** option. Click the **Extract** button to extract the selected files into the project. A result dialog showing the details of the extract operation will be presented once all the files have been extracted.

6.3 Show Outgoing Files

This operation shows the set of all files that are present in a project or folder but not in the SpectrumSCM repository with which the project is shared. To invoke this feature, select **Team→Show Outgoing Files …** from the context menu of a project or folder shared with a SpectrumSCM Repository Reference. Similar to the Show Incoming Files operation, you will be presented with a table listing all the files that need to go into the repository. Select the required files and click Add Source to add the selected files to the repository. A Select CR dialog will be presented to allow the user to select a Change Request for the operation. A result dialog showing the details of the add operation will be presented once all the files have been added.



Outgoing files under ECLTEST

TestOut/.#SETUPLOG1.TXT.1.1
TestOut/.#setjavaenv.1.1
TestOut/BIGNISE20m.java
TestOut/Scmd.java
TestOut/Temp/SETUPLOG1.TXT
TestOut/Temp/test.txt
TestOut/User_int.java
TestOut/bali.txt
TestOut/btest.txt
TestOut/goli.txt
TestOut/myfile
TestOut/new1

Add file(s) to repository

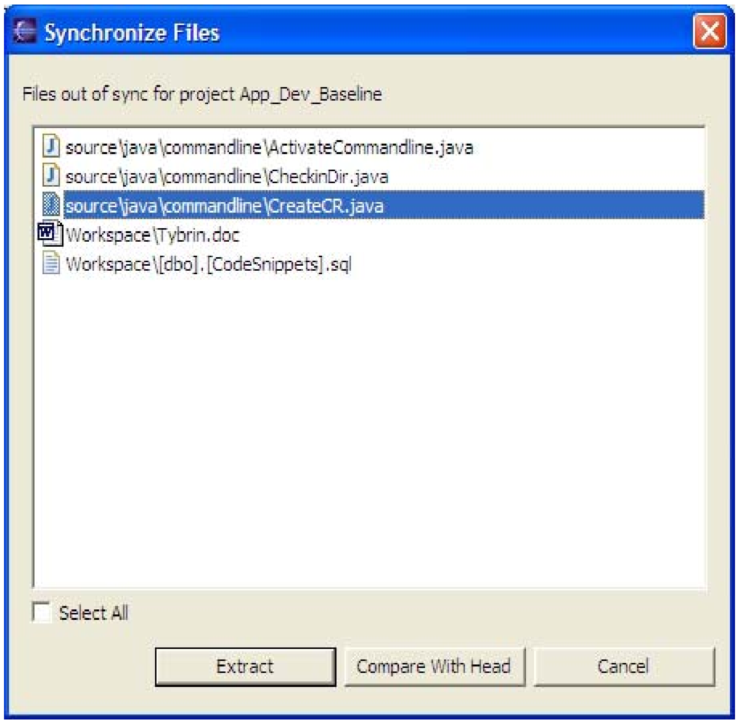
Select Change Request:

ECL0005
ECL0007
ECL0008
ECL0010
ECL0011
ECL0012
ECL0014
ECL0015

Figure 18: Show Outgoing Files

6.4 Show Out of Sync Files

This operation shows the set of all files that exist both under an Eclipse project and the SpectrumSCM repository, but are not in sync (i.e. they are different in content). A file will be marked as being out of sync if it has been modified by the user in the Eclipse workspace or if the file contents have changed in the SpectrumSCM repository due to modifications made by other team members. To list all the out of sync files under a particular project or folder, select **Team→Show Out of Sync Files …** from the context menu for a project or folder shared with a SpectrumSCM Repository Reference. The user will be presented with a table listing all the files that are out of sync with the files in the repository. Select the required files and click Extract to extract the files to the local workspace. The Compare With Head will be enabled when a single file is selected from the list and can be used to view the differences between the file in the local workspace and the file in the repository.



Files out of sync for project App_Dev_Baseline

source\java\commandline\ActivateCommandline.java
source\java\commandline\CheckinDir.java
source\java\commandline\CreateCR.java
Workspace\Tybrin.doc
Workspace\[dbo].[CodeSnippets].sql

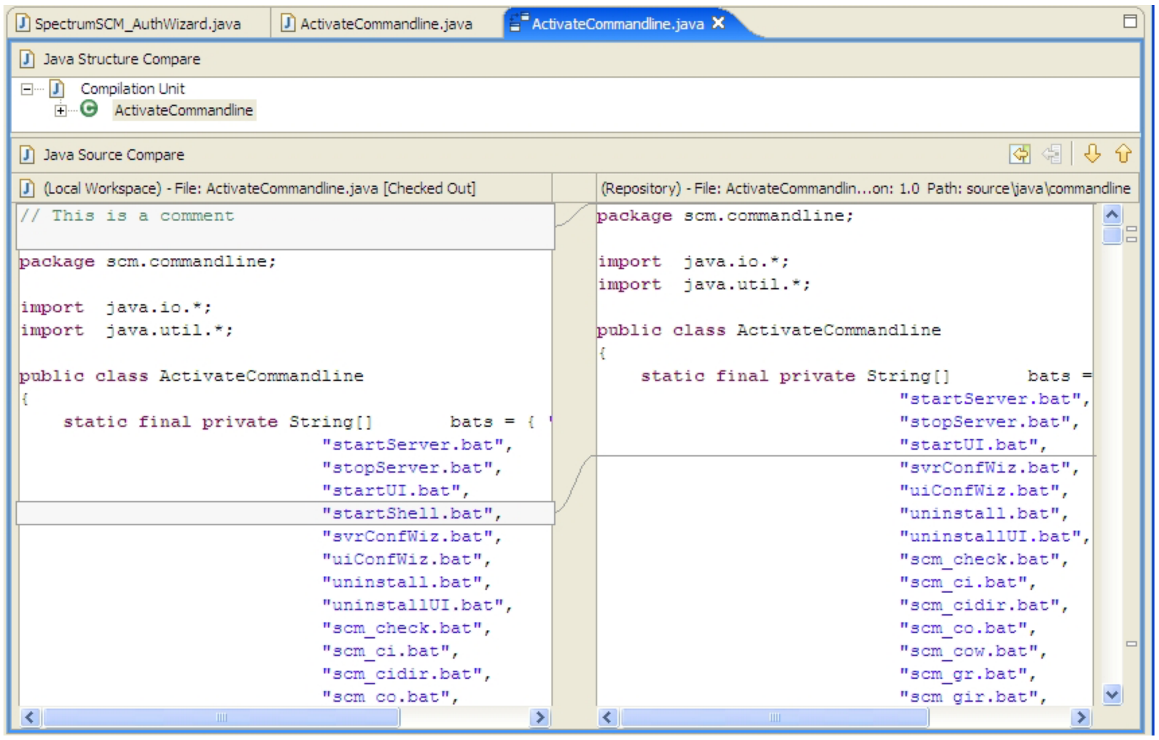
Figure 19: Show Out of Sync Files

6.5 Compare and Merge Operations

In a collaborative environment where multiple users work on the same project, other users can modify files in the repository in parallel to your work. You can find out the changes made by other users and merge these changes into your workspace using the compare and merge operations. Files can be compared with the head revision of the file in the repository or with a previous version.

To compare a file with the head revision, select **Compare With→Head Revision** from a file's context menu. If there are differences between the two instances of the file, the differences will be displayed using the standard compare editor provided by the Eclipse platform. If the file on the workspace has been checked out for edit, the compare editor will have additional buttons for merging the changes into the local file.

In a similar manner, a file can be compared and merged with a previous version of the file in the repository by selecting **Compare With→Previous Version …** from the file's context menu. By default, Eclipse provides features for comparing and merging a file with different instances of the file in the file's local history. It is always a good practice to synchronize your workspace and verify that your work is compatible with the changes made by others before you check your work in.



(Local Workspace) - File: ActivateCommandline.java [Checked Out]

```
// This is a comment

package scm.commandline;

import  java.io.*;
import  java.util.*;

public class ActivateCommandline
{
    static final private String[]       bats = { '
                                   "startServer.bat",
                                   "stopServer.bat",
                                   "startUI.bat",
                                   "startShell.bat",
                                   "svrConfWiz.bat",
                                   "uiConfWiz.bat",
                                   "uninstall.bat",
                                   "uninstallUI.bat",
                                   "scm_check.bat",
                                   "scm_ci.bat",
                                   "scm_cidir.bat",
                                   "scm_co.bat",
```

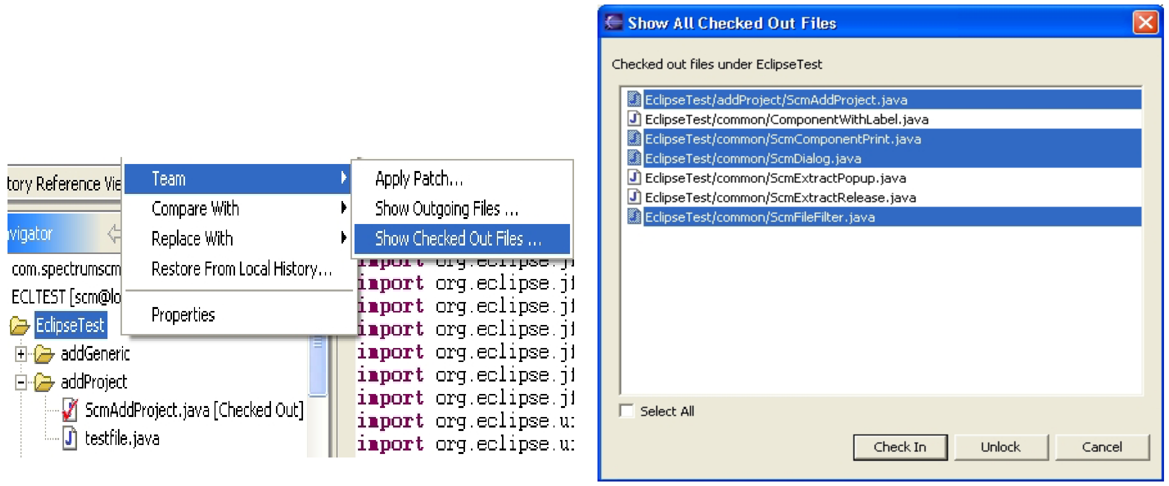
(Repository) - File: ActivateCommandlin...on: 1.0 Path: source\java\commandline

```
package scm.commandline;

import  java.io.*;
import  java.util.*;

public class ActivateCommandline
{
    static final private String[]       bats =
                            "startServer.bat",
                            "stopServer.bat",
                            "startUI.bat",
                            "svrConfWiz.bat",
                            "uiConfWiz.bat",
                            "uninstall.bat",
                            "uninstallUI.bat",
                            "scm_check.bat",
                            "scm_ci.bat",
                            "scm_cidir.bat",
                            "scm_co.bat",
                            "scm_cow.bat",
                            "scm_gr.bat",
                            "scm_gir.bat",
```

Figure 20: Compare-Merge Editor

6.6 Show All Checked Out Files

This operation can be used to check in or unlock multiple checked out files in one shot. To display all the checked out files under a project or folder, select **Team→Show Checked Out Files …** from the context menu for the project or folder. You will be presented with a table listing all the checked out files under the selected resource. Select the required files and click **Check In** or **Unlock** to check in or unlock all the selected files in one go. A result dialog showing the details of the operation will be presented at the end of the operation.



```
import org.eclipse.j
import org.eclipse.j
import org.eclipse.j
import org.eclipse.j
import org.eclipse.j
import org.eclipse.j
import org.eclipse.u
import org.eclipse.u
```

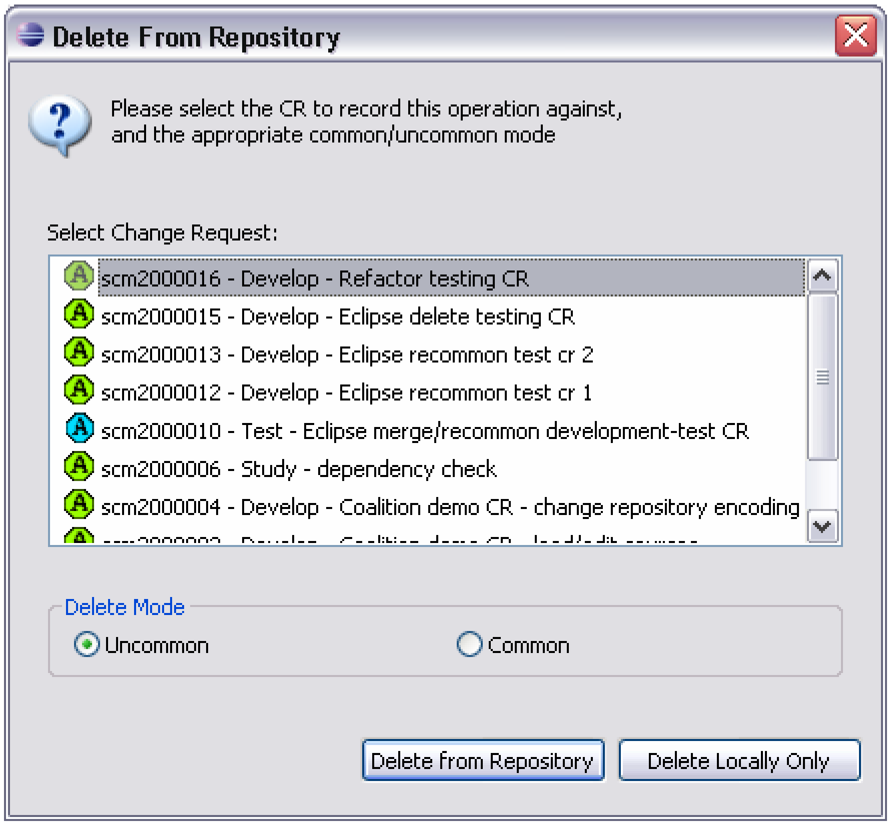
Checked out files under EclipseTest

- EclipseTest/addProject/ScmAddProject.java
- EclipseTest/common/ComponentWithLabel.java
- EclipseTest/common/ScmComponentPrint.java
- EclipseTest/common/ScmDialog.java
- EclipseTest/common/ScmExtractPopup.java
- EclipseTest/common/ScmExtractRelease.java
- EclipseTest/common/ScmFileFilter.java

Figure 21: Show All Checked Out Files

6.7 Deleting Files

If a user attempts to delete an Eclipse workspace file that is shared through a SpectrumSCM repository provider, a dialog will be triggered. First, though, the Eclipse platform itself asks for confirmation of the delete, and if the file is read-only a second confirmation will be presented. After that, you will be asked whether you wish to delete the file locally only, or whether to delete it from the repository too. If a repository delete is to be performed you will need an appropriate CR and permissions to complete this operation.



Please select the CR to record this operation against,
and the appropriate common/uncommon mode

Select Change Request:

- scm2000016 - Develop - Refactor testing CR
- scm2000015 - Develop - Eclipse delete testing CR
- scm2000013 - Develop - Eclipse recommon test cr 2
- scm2000012 - Develop - Eclipse recommon test cr 1
- scm2000010 - Test - Eclipse merge/recommon development-test CR
- scm2000006 - Study - dependency check
- scm2000004 - Develop - Coalition demo CR - change repository encoding

Delete Mode: [✓ Uncommon] Common

Local deletes will delete the file(s) from the Eclipse workspace but not from the repository. The repository would need to be synchronized at a later time, possibly by use of the workspace synchronizer.

6.8 Refactor/Rename

If a user attempts to delete an Eclipse workspace file that is shared through a SpectrumSCM repository provider, a dialog will be triggered.

First as a general note on refactoring, the Eclipse refactoring functionality works in conjunction with the compiler/build engine, as such if your module does not compile/build the refactor operation might not function successfully. This is independent of any SpectrumSCM behaviour.

A refactor operation that would involve the SpectrumSCM repository provider would be one (as an example) where the Java class name was being changed. This would involve changing the file name on this particular file AND updating all other source files that refer to this class.

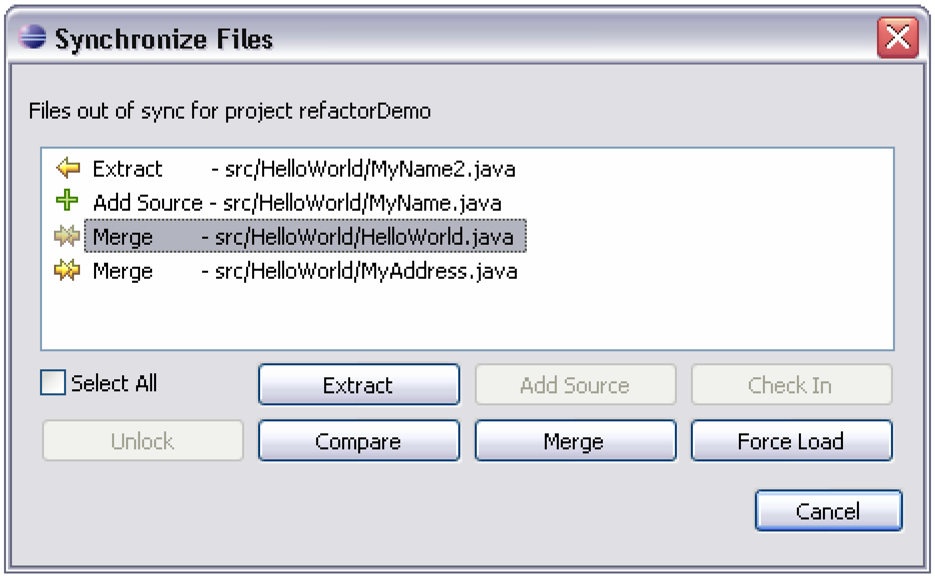
Upon initiation of such a refactor operation the user will be asked to check-out/select a CR against which to record this action. All of the files involved in the refactor will be checked-out if they are not already out for edit. The choice is also given to complete the refactor locally, in which case the repository will need to be synchronized at a later time.

Since such a refactor operation involves moving and/or renaming the file under SpectrumSCM, the main file will need to be checked-in before the actual move. This action is performed automatically by the SpectrumSCM plugin but the user is prompted if there are any unsaved changes as a safety measure.

Once the refactor completes, all the side-effect files will be checked back in (if they were automatically checked-out) but the main file will be left out-for-edit so that the main edit can be confirmed/continued.

6.9 Workspace/Repository Sync

The workspace synchronizer is a consolidation of all the above features (6.2 thru 6.8), and also allows the user to work offline from the SpectrumSCM server, and then synchronize after the fact.

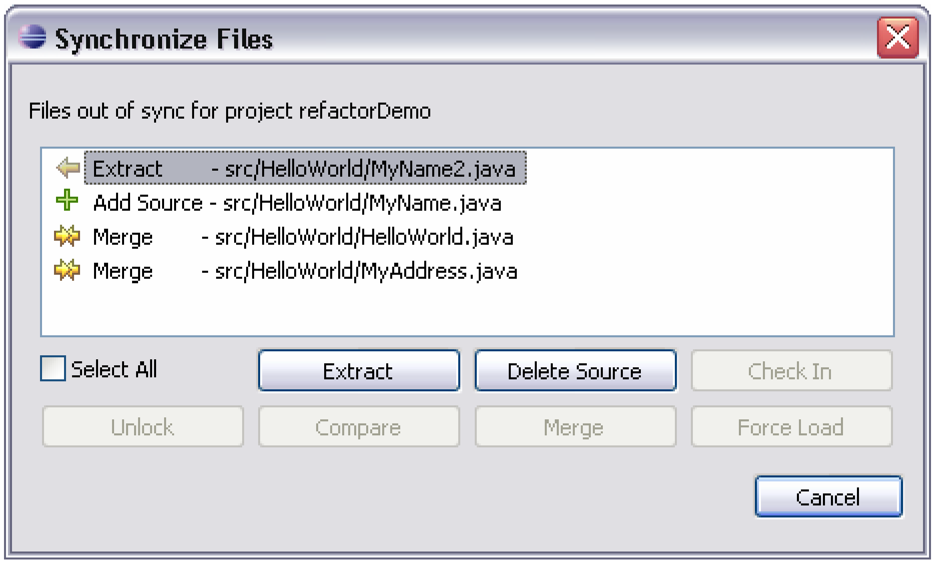

Synchronize Files

Files out of sync for project refactorDemo

Extract - src/HelloWorld/MyName2.java
Add Source - src/HelloWorld/MyName.java
Merge - src/HelloWorld/HelloWorld.java
Merge - src/HelloWorld/MyAddress.java

To initiate the workspace synchronizer, select a folder or project in the navigator or package explorer views, right-click and access the team provider menu options. The dialog presented will show the "out-of-sync" files and why they are deemed to be out-of-sync. The icon and keyword gives a quick indication as to the reason for the out-of-sync situation and the tooltip will give a more detailed explanation.

Selecting an item (or multiple of the same type) will enable the appropriate action buttons in the lower part of the screen. Certain buttons are also multi-function in that they switch modes based on the selections active. For example, if an "Extract" item is selected, the "Add Source" button switches to "**Delete Source**" to enable that source file to be removed from the repository.

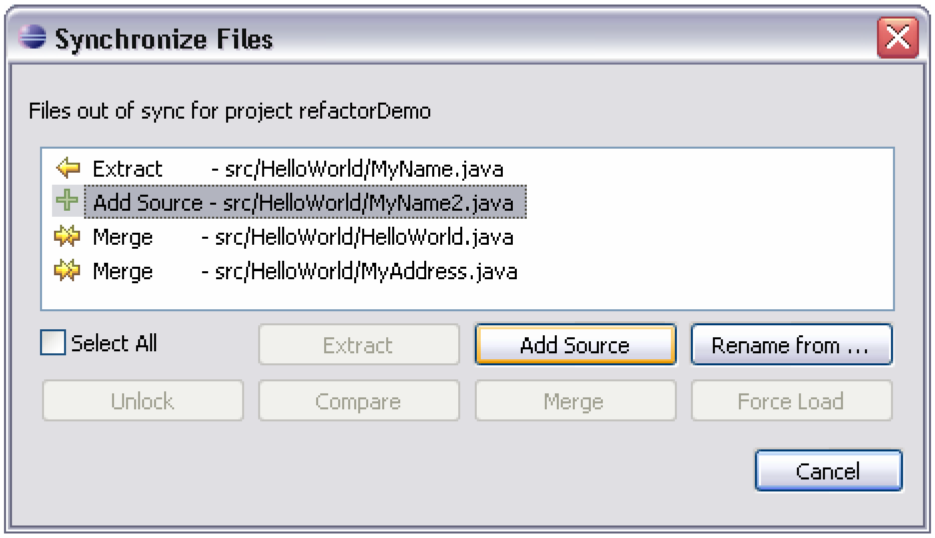


Synchronize Files

Files out of sync for project refactorDemo

```
Extract      - src/HelloWorld/MyName2.java
Add Source - src/HelloWorld/MyName.java
Merge       - src/HelloWorld/HelloWorld.java
Merge       - src/HelloWorld/MyAddress.java
```

If an "Add Source" item is selected, the "Check In" button switches to a "**Rename From**" button to enable an offline rename or move to be re-synchronized with the original repository source. Selecting the "Rename From" action will display a secondary dialog requested which repository file this Eclipse file was renamed from. Selecting the file completes the move/rename relative to the selected CR.



Synchronize Files

Files out of sync for project refactorDemo

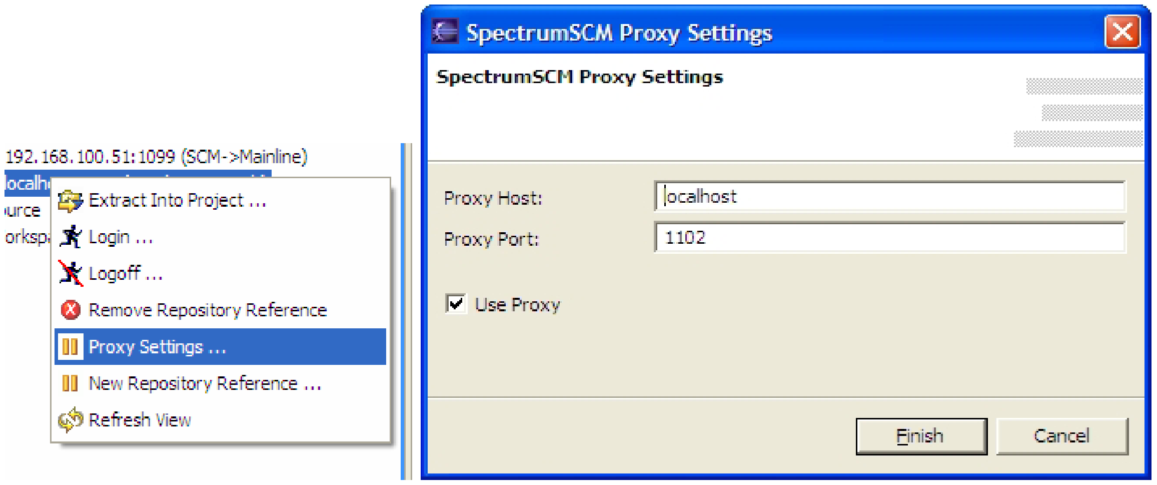
```
Extract      - src/HelloWorld/MyName.java
Add Source - src/HelloWorld/MyName2.java
Merge       - src/HelloWorld/HelloWorld.java
Merge       - src/HelloWorld/MyAddress.java
```

7.0 SpectrumSCM Proxy for Eclipse

The SpectrumSCM Proxy provides enhanced performance for file-based operations like checkout and extract in bandwidth constrained network environments. By maintaining a passive cache for frequently checked out revisions of e-Assets under source control, the SpectrumSCM proxy reduces the response time for file based operations and thus provides a better user experience for distributed development teams. The SpectrumSCM proxy also reduces the network traffic across the WAN and the load on the SpectrumSCM server. See Chapter 12 in the SpectrumSCM User Guide for more details.

The SpectrumSCM plugin for Eclipse is fully integrated with the SpectrumSCM proxy. Users can define and modify their proxy settings for a particular SpectrumSCM server from within the Eclipse IDE. Users can also dynamically enable or disable proxy usage for file based operations. All repository references that refer to the same SpectrumSCM server will share the proxy settings defined for that server.

To define or modify the proxy settings for a server, right click on a repository reference and select Proxy Settings from the context menu. The user will be presented with a dialog that can be used to specify the address and port number for the SpectrumSCM proxy. The Use Proxy option can be used to enable/disable use of the proxy. Click Finish to save the settings. If the Use Proxy option is enabled, all checkout requests are routed through the specified proxy and the file will be retrieved from the proxy cache if there is a hit. If there is a miss, the proxy will download the requested file from the server and cache it for future use.


192.168.100.51:1099 (SCM->Mainline)
Extract Into Project ...
Login ...
Logoff ...
Remove Repository Reference
Proxy Settings ...
New Repository Reference ...
Refresh View
SpectrumSCM Proxy Settings
Finish
Cancel

SpectrumSCM Proxy Settings

Proxy Host: localhost

Proxy Port: 1102

[✓] Use Proxy

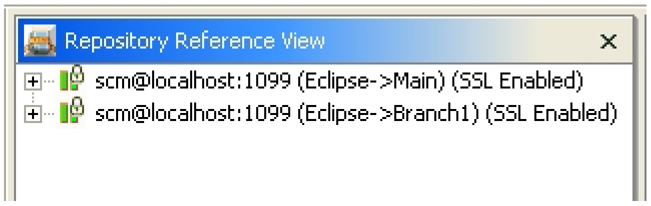
Figure 23: SpectrumSCM Proxy Settings

The SpectrumSCM Proxy is a separate software entity and needs to be installed and started separately. See Chapter 12 in the SpectrumSCM User Guide for proxy installation and administration details.

8.0 Security

SpectrumSCM supports the use of *Secure Sockets Layer (SSL)* to encrypt communication between the client and the server. This allows secure access to proprietary information across untrusted networks. Assuming that the required SSL parameters have been properly configured, the SpectrumSCM plugin for Eclipse will automatically detect and establish a secure connection with an SSL enabled server. If the communication is secured, the decoration for the Repository Reference will change to reflect the fact that the client and server are exchanging information over a secure layer. In addition, if the server uses LDAP authentication, the plugin can take advantage of the secure, centralized authentication system provided by LDAP. Thus the SpectrumSCM plugin for Eclipse supports different levels of security ranging from simple password protection, LDAP authentication to SSL protected communication.

The reader is referred to the http://www.spectrumscm.com for documents related to SSL and LDAP support for SpectrumSCM.



scm@localhost:1099 (Eclipse->Main) (SSL Enabled)
scm@localhost:1099 (Eclipse->Branch1) (SSL Enabled)

Figure 24: SSL Enabled Repository Reference

9.0 Summary

Since its introduction in 2001, Eclipse has enjoyed tremendous visibility and popularity as a robust, commercial-quality platform for the construction of seamlessly-integrated tools for application development. SpectrumSCM acts as a team repository provider for Eclipse and provides a tight integration with the platform through the SpectrumSCM Plugin for Eclipse. The plugin has been designed keeping the end developer in mind and provides a rich set of features for day-to-day development activities. However, SpectrumSCM is more than a simple version control and issue tracking tool, and includes process management, release management, advanced branching, workflow control and much more, all integrated into one tool. The above-mentioned features and several administrative features can be accessed through the client interface that comes bundled with SpectrumSCM.

For additional information on SpectrumSCM please visit our website at www.spectrumscm.com.
Or contact Spectrum Software at 770.448.8662