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

Subject: **Integrating SpectrumSCM with Visual Studio .NET**

Issue Date: **Dec 29th, 2004**

William C. Brown
corey@spectrumsoftware.net
(770)448-8662

1.0 Introduction:

While it is possible to use most standalone CM tools in parallel with an IDE (*integrated development environment*), managing both system together can be inconvenient. Modern IDEs silently modify a large set of supporting files as the user goes about their normal work. For instance, when a new resource is added to a Visual Studio project, the IDE automatically updates several project related support files. In order to use a standalone CM system in parallel with Visual Studio, the user would have to be aware of all of the supporting files that the IDE may need to create/update, and to have these file already checked out from the CM environment. Because of these constraints, it is much more practical to have the CM solution tightly integrated with the IDE itself so that supporting files can be automatically checked out and modified when necessary.

The integration between an IDE and a CM solution should be as clean as possible in order to reduce the amount of unnecessary interactions with the user. For example, when checking out a file through a user initiated activity, or from an IDE initiated background operation, the process should execute without presenting the user with additional popup boxes and other dialogs. The user should be able to setup the working environment before hand so that subsequent check outs and other operations can proceed without the IDE repeatedly prompting for additional information.

This paper describes the integration between Microsoft Visual Studio .NET[1], which will simply be referred to as Visual Studio throughout the rest of this document, and Spectrum Software's source configuration management solution, SpectrumSCM.

2.0 SpectrumSCM and the MS-SCCI:

The Microsoft SCCI (Microsoft Common Source Code Control Interface Specification) is the standard around which most CM-IDE integrations are formed. The SCCI is a fairly old interface specification that was originally written strictly as an interface specification between Microsoft's development environments and Microsoft's VSS (Visual Source Safe). The specification is available from Microsoft through an NDA[2] and is not officially supported by Microsoft. All Microsoft development tools, and most other IDE provider's tools, make use

[1] www.microsoft.com

[2] Microsoft requires implementers of the SCCI to sign a non-disclosure agreement in order to obtain the specification itself.

of the SCCI as the gateway to the users CM solution. CM tool vendors must implement the SCCI in order to provide a gateway from SCCI enabled development environments to their CM solution. The CM tool vendor must implement the SCCI exactly as it is defined, but are also free to enhance the implementation with additional popup dialogs and other necessary windows. While the vendor is free to add additional popup dialogs where necessary, it really is best to keep the number of extra popup dialog boxes to a minimum. Most IDEs already have a set of dialogs that are going to be presented to the user during normal operations anyway, so it would be an annoyance to the user to be presented with any other additional dialogs.

3.0 The SpectrumSCM SCCI Implementation:

The SpectrumSCM SCCI implementation (**SpectrumSCCI**) is a complete implementation of the Microsoft SCCI interface specification. This implementation attempts to abide by the tenets mentioned in the introduction and has been made to be as unobtrusive to the user as possible. For the most part, the SpectrumSCM implementation of the SCCI is almost invisible to the user and as such does not present additional popup dialog windows to the user unless absolutely necessary. All SpectrumSCM specific options are controlled through the SpectrumSCM Dashboard (see section 3.1 for details on the Dashboard) and as such do not need to be prompted for each time a CM operation is accessed through Visual Studio.

The SpectrumSCM implementation of the SCCI allows the user to use many of SpectrumSCM's advanced features, including the use of Change Requests[3], advanced branching patterns[4], reports and visual differencing. The implementation is a native implementation of the SCCI and was done by creating a Microsoft Windows DLL that Visual Studio uses to communicate with the SpectrumSCM Dashboard. The Dashboard completes the connection to the SpectrumSCM server itself, thus creating a 3-tiered system that is extremely flexible. The user is free to switch between multiple projects in Visual Studio without having to change settings on the Dashboard or make any other adjustments to the CM system. The user is also free to start as many instances of Visual Studio as they need. Each separate instance of the Visual Studio application will be accompanied by a separate instance of the SpectrumSCM Dashboard.

3.1 The SpectrumSCM Dashboard[5]:

The SpectrumSCM Dashboard is a unique concept in SCCI integrations and Spectrum Software is proud to be the first SCM vendor to offer this unique integration to Visual Studio users. The Dashboard is delivered as part of the standard SpectrumSCM UI installation, only the Visual Studio DLL plug-in must be downloaded separately from the Spectrum web site to complete the installation. (Please see the section 4.0 on installation). The SpectrumSCM Dashboard is a unique feature of the SpectrumSCM SCCI implementation that allows the user to use many of SpectrumSCM's advanced features and helps to eliminate annoying popup windows. The Dashboard itself is a very low profile strip that can be placed anywhere on the users screen, much like a detachable toolbar would be

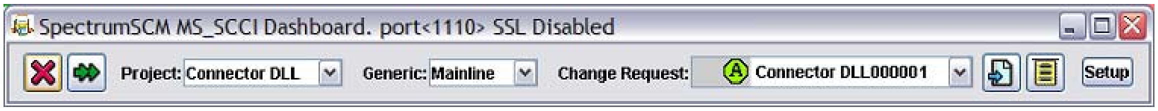
[3] Change Requests are also know as Change Lists or Modification Requests in other systems.

[4] http://www.spectrumscm.com/WhitePapers/BranchingDesignPatternsupdate.pdf

[5] Patent Pending.

placed just above the Window's Start Menu Taskbar, or anchored to the top of the users screen. The User simply needs to adjust the size of the Visual Studio main screen to just slightly below full screen status to allow the Dashboard a small viewing area. The Dashboard can also be minimized into the taskbar as it is not necessary to access the Dashboard unless the user wishes to select a different change request, or operate against a different branch of their product. An illustration of the SpectrumSCM Dashboard is given in Figure #1.

Figure #1



SpectrumSCM MS_SCCI Dashboard. port<1110> SSL Disabled

Project: Connector DLL

Generic: Mainline

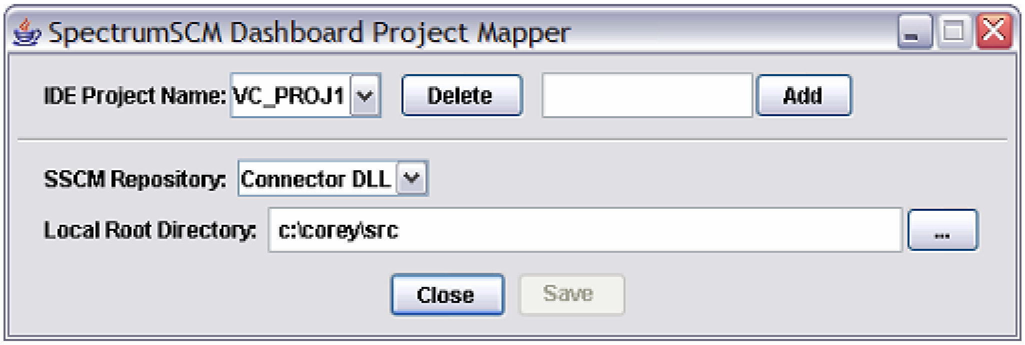
Change Request: Connector DLL000001

In this figure, the SpectrumSCM Dashboard is working with project **Connector DLL**, on the **Mainline** branch and the user is currently using CR# **Connector DLL00001** for his/her CM activities. Also note in the title bar that this instance of the Dashboard is listening for activity on port **1110**, which is the starting port number for a series of Dashboards that the user may start if they choose to work with multiple instances of Visual Studio. A single Dashboard instance can be used to work with a series of Visual Studio projects as long as the user opens those projects within a single instance of Visual Studio. Also note that in this particular case secure socket layers (SSL) is not being used.

3.2 Dashboard Configuration (Project Mapper):

The SpectrumSCM Dashboard must be configured using the Project Mapper configuration screen before the Dashboard can be used with Visual Studio or other IDE projects. The Project Mapper screen is accessed by pressing the Setup button, which is located on the far right of the Dashboard display. The Project Mapper screen is shown in Figure #2

Figure #2



SpectrumSCM Dashboard Project Mapper

IDE Project Name: VC_PROJ1

SSCM Repository: Connector DLL

Local Root Directory: c:\corey\src

The SpectrumSCM Dashboard Project Mapper allows users to map Visual Studio or other IDE projects to SpectrumSCM repository names. Each IDE project can also have a separate working directory (Local Root Directory) from other projects. During CM operations the SpectrumSCM Dashboard will examine the directory information provided by Visual Studio and will determine the proper working directory to use for each file level operation. This is extremely important for Visual Studio web based projects or for any higher level solution or project that contains multiple sub-projects.

Tip: Visual Studio solutions with sub-projects should all share the same SpectrumSCM repository (SSCM Repository) value. This must be done since Visual Studio and other IDEs do not treat sub-project files separately from the other files in the higher level solution.

The user must configure the Dashboard first before any other IDE operations are performed. Each Visual Studio project name must be entered into the Project Mapper tool and each IDE project must be associated with a SpectrumSCM repository and a Local Root Directory. Once the project information has been entered into the Mapper screen, the user must either press the ***Save*** button or, when prompted by the ***Close*** button action, choose to save the current configuration information to disk. Once saved, the configuration information will be persisted for future use.

In the Visual Studio environment, the Dashboard can be manually started by starting Visual Studio directly and then using the ***File->Source Control->SpectrumSCM*** menu item to start the SpectrumSCM Dashboard. When the Dashboard starts, login and then press the Setup button to access the Project Mapper screen. Once the configuration information has been added to the Dashboard, the user can proceed to open a Visual Studio project and begin SCCI related activities.

Other SCCI compliant IDEs may start the Dashboard on initialization but may proceed to immediately start issuing SCCI commands. In this case, errors may be encountered before the Dashboard configuration can be completed. Acknowledge any error messages that the IDE may produce and proceed to configure the Dashboard. Once configured, close and restart the IDE to begin work.

Dashboard configurations can be easily shared between users. The Project Mapper configuration information is stored in the user's home directory in a file called ".SCCIdata". See section 3.3 for a complete description of all of the control files and log files used by the SpectrumSCM Dashboard.

To share configuration information between users, simply place the ".SCCIdata" file in the home directory of each user that will be using the IDE solution. The .SCCIdata file can also be e-mailed to other users as an attachment.

3.3 Control Files and Log Files:

The SpectrumSCM Dashboard recognizes and works with three control files that are located in the user's home directory. The <**specSCCI.ini**> file controls the path to the Java virtual machine, and the starting port number for Visual Studio to Dashboard communications. The User can elect to change either setting in the specSCCI.ini control file. The format of the file is as follows:

```
C:\j2sdk1.4.2_06\bin\javaw | 1110
Debug=yes
```

Note that the separator between the Java virtual machine specification and the port number is the vertical pipe symbol "|". The second line of the file controls the debugging/logging feature of the Dashboard and the IDE. When enabled, both systems will write out communication information, errors and other useful information to the following two files:

```
C:\tmp\specSCCI_<port>.log
C:\tmp\Dashboard_<port>.log
```

The port number is part of the log file name itself. This allows the user to discriminate between the log files of multiple running instances of the Dashboard and the IDE. Normally, the debug feature is turned off as it generates a fair amount of information. The feature need only be turned on when requested by Spectrum Software in order to resolve any problems encountered by the user.

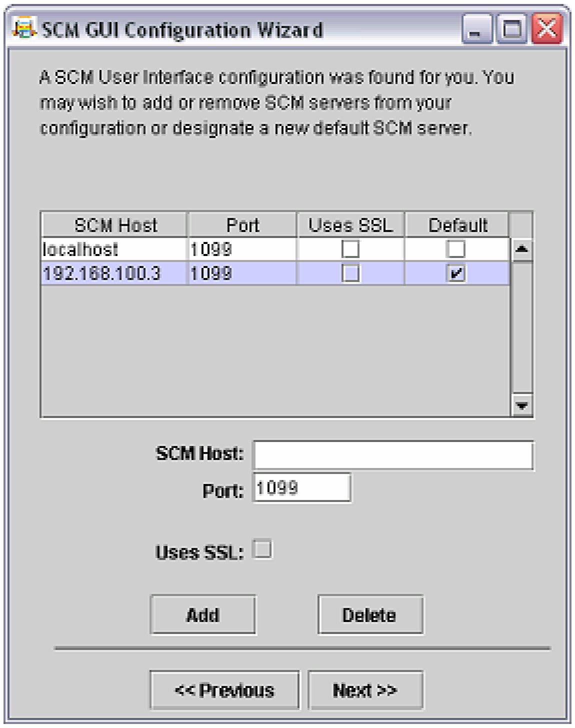
The second control file is the <**.SCCIData**> file. This file contains the information entered into the Project Mapper configuration screen. This file is binary and thus cannot be edited directly. Use the Setup button on the Dashboard to review and change information contained within this file.

The last control file is the <**.SCMui**> file. This file contains information regarding the installation directory for SpectrumSCM and the IP address or name of the default server. The format of an example <.SCMui> file is as follows:

```
c:\SCM
192.168.100.3:1099::*
```

The first line of the file contains the path to the actual SpectrumSCM installation directory. The second line and possibly other lines, contain information about where the SpectrumSCM server is running, the port number it is running on, whether the server is SSL enabled and finally, which server is the default. The asterisk denotes which server is the default server. **The .SCMui file should not be edited by hand**. The content of the file is controlled through the SpectrumSCM UI configuration wizard. The configuration wizard can be started by navigating through the Window's Start menu to the configuration wizard: ***Start->SpectrumSCM UI->UI Configuration Wizard***. Figure #3 illustrates the main screen of the UI configuration wizard:

Figure #3



SCM GUI Configuration Wizard

A SCM User Interface configuration was found for you. You may wish to add or remove SCM servers from your configuration or designate a new default SCM server.

SCM Host	Port	Uses SSL	Default
localhost	1099	□	□
192.168.100.3	1099	□	☑

SCM Host:

Port: 1099

Uses SSL: □

Note that the default entry on this screen is the 192.168.100.3 server, just like in the contents of the .SCMui file listed above. When the user changes this information, it is reflected in the .SCMui file. The Dashboard uses the information in the .SCMui file to locate the running server on a particular machine and particular port number.

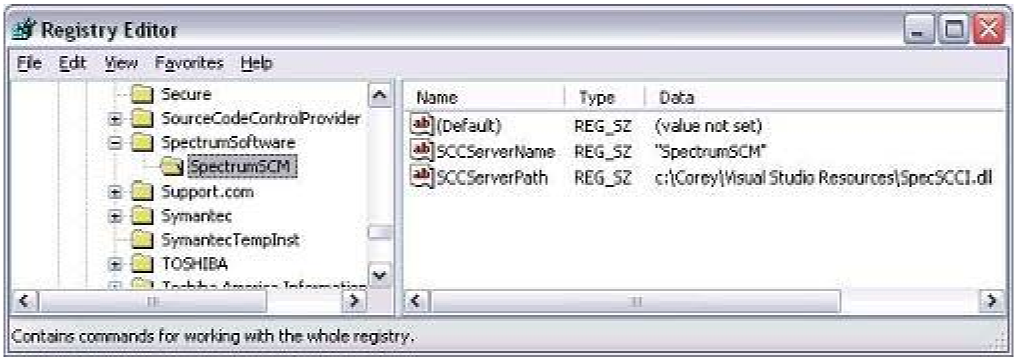
4.0 Installation:

Installation of the SpectrumSCCI product into a Windows machine is very simple. The user simply needs to download the SpectrumSCCI_install.jar from the SpectrumSCM web page www.spectrumscm.com and then double click the jar file to begin the installation. The installation needs to be done with the same user-id that was used to install the SpectrumSCM system itself. **Note that only the SpectrumSCM UI portion of the system needs to be installed on the user's workstation and not the entire SpectrumSCM server portion**. The installation will automatically detect the SpectrumSCM installation directory and adjust the registry entries to match. The installation will also install all of the necessary DLLs and other supporting libraries in the existing SpectrumSCM installation directory.

4.1 Registry Keys:

The SpectrumSCCI installation will install several keys into the Windows registry automatically. These entries are depicted in figures #4, #5 and #6. **Please note that this section is provided as reference only and requires no action by the user/installer.**

Figure #4

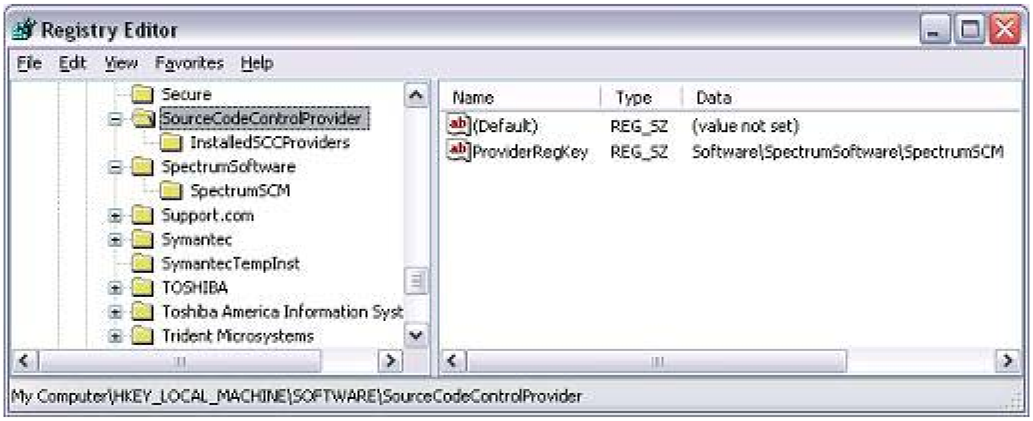


Name	Type	Data
(Default)	REG_SZ	(value not set)
SCCServerName	REG_SZ	"SpectrumSCM"
SCCServerPath	REG_SZ	c:\Corey\Visual Studio Resources\SpecSCCI.dll

Contains commands for working with the whole registry.

These registry entries define Spectrum Software as a software vendor that has the software product **SpectrumSCM** installed on this machine. The two keys (SCCServerName and SCCServerPath) identify the name of the SCCI provider (SpectrumSCM) and the path to the actual DLL that Visual Studio and other products will use to communicate with the Dashboard. Note that the installation path will be different for each machine, depending on where the actual SpectrumSCM installation directory is.

Figure #5

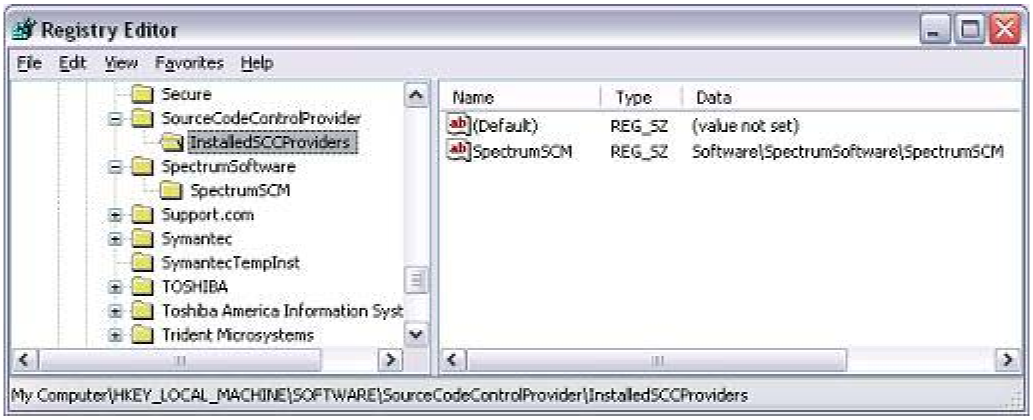


Name	Type	Data
(Default)	REG_SZ	(value not set)
ProviderRegKey	REG_SZ	Software\SpectrumSoftware\SpectrumSCM

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\SourceCodeControlProvider

This figure illustrates the second key installed during the SpectrumSCCI installation. This key installs SpectrumSCM as a possible SourceCodeControlProvider instance. Note that this key points to the key that was added in Figure #4.

Figure #6



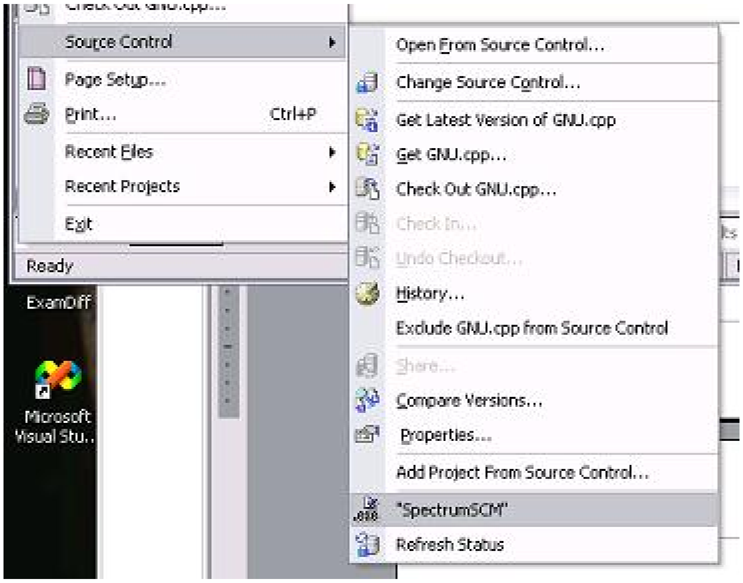
Name	Type	Data
(Default)	REG_SZ	(value not set)
SpectrumSCM	REG_SZ	Software\SpectrumSoftware\SpectrumSCM

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\SourceCodeControlProvider\InstalledSCCProviders

This figure illustrates the third and last key that is installed during the SpectrumSCCI installation. This key provides specific information about the SCCI provider and again points to SpectrumSoftware as a possible provider.

With these keys installed, Visual Studio will automatically detect the existence of SpectrumSCM as a CM solution provider and will add an entry to the Source Control sub menu of Visual Studio's File menu. This is depicted in figure #7.

Figure #7



This figure illustrates that Visual Studio has successfully located the SpectrumSCM registry keys and has enabled SpectrumSCM as a CM provider. If other providers are registered on the machine, they will be listed here as well.

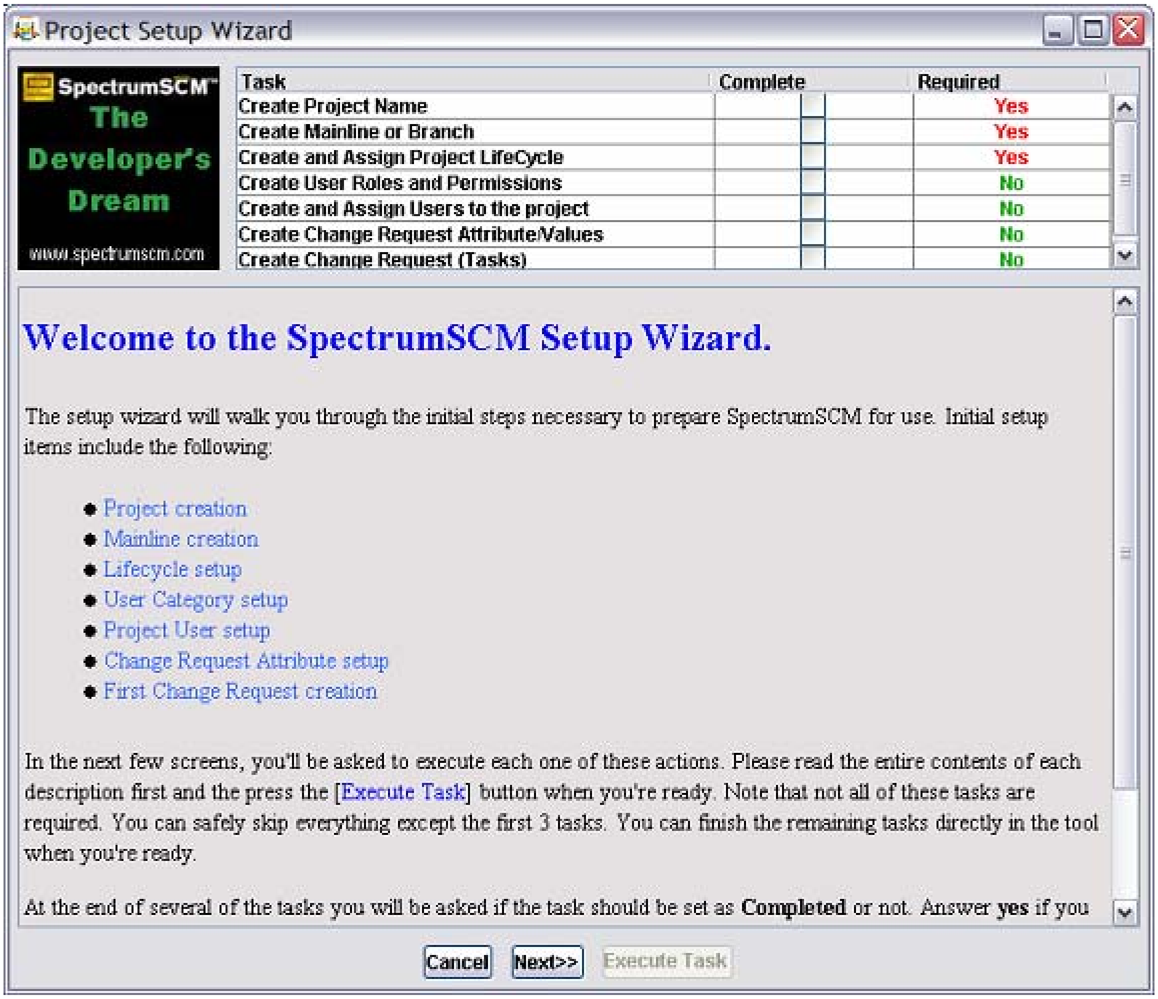
5.0 Setting up a SpectrumSCM Project for the First Time:

This subject really falls outside of the scope of this document, but a brief summary of the steps necessary to get a project up and running in SpectrumSCM can't hurt either.

While a Visual studio project name can be mapped to any SpectrumSCM project name, it is a whole lot easier to synchronize the two right off the bat. Determine what the Visual Studio project name for your new project is going to be and then proceed to create the exact same project in SpectrumSCM. Project creation in SpectrumSCM is very simple. Follow these steps to complete the task:

- Start an instance of the SpectrumSCM client software and login as a user with the appropriate privileges to create a new project.
- Pull down on the "Administration" menu and select the "Create Project Wizard" menu item (Figure #8)

Figure #8



Task	Complete	Required
Create Project Name	☐	Yes
Create Mainline or Branch	☐	Yes
Create and Assign Project LifeCycle	☐	Yes
Create User Roles and Permissions	☐	No
Create and Assign Users to the project	☐	No
Create Change Request Attribute/Values	☐	No
Create Change Request (Tasks)	☐	No

Welcome to the SpectrumSCM Setup Wizard.

The setup wizard will walk you through the initial steps necessary to prepare SpectrumSCM for use. Initial setup items include the following:

- Project creation
- Mainline creation
- Lifecycle setup
- User Category setup
- Project User setup
- Change Request Attribute setup
- First Change Request creation

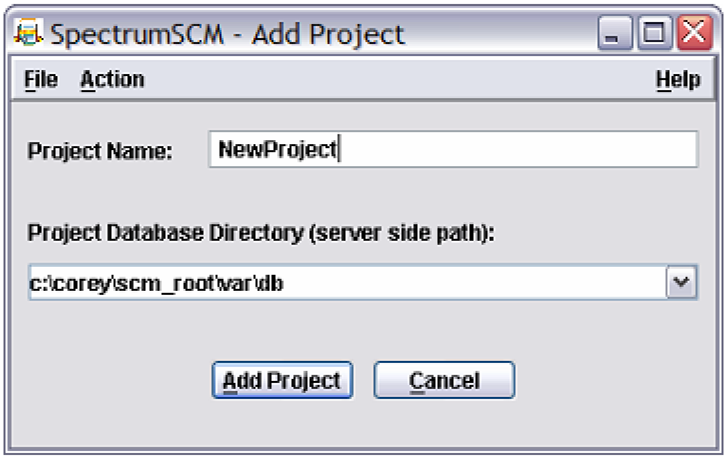
In the next few screens, you'll be asked to execute each one of these actions. Please read the entire contents of each description first and the press the [Execute Task] button when you're ready. Note that not all of these tasks are required. You can safely skip everything except the first 3 tasks. You can finish the remaining tasks directly in the tool when you're ready.

At the end of several of the tasks you will be asked if the task should be set as **Completed** or not. Answer **yes** if you

Step through the seven items in the Project Setup Wizard and execute each step as you go. When complete, a new project will have been created and completely configured. The following are the individual steps presented by the Project Creation Wizard.

- Create Project Name: Enter the name for the new project and modify the repository location if necessary.

Figure #8



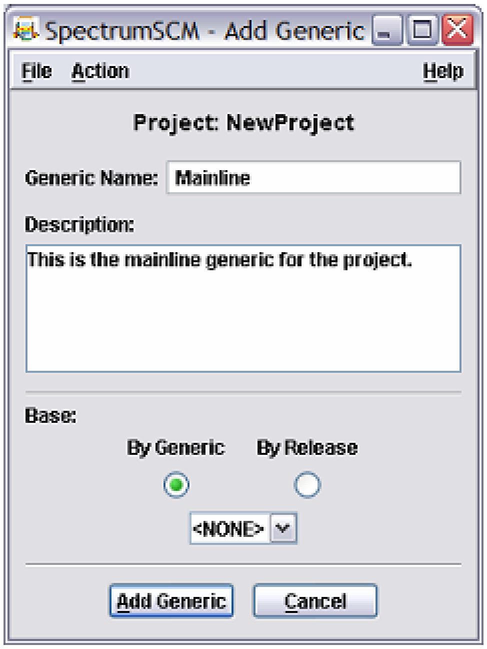
Project Name: NewProject

Project Database Directory (server side path):

c:\corey\scm_root\var\db

- Create Mainline or Branch: Add the first generic[6].

Figure #9



Project: NewProject

Generic Name: Mainline

Description:

This is the mainline generic for the project.

Base:

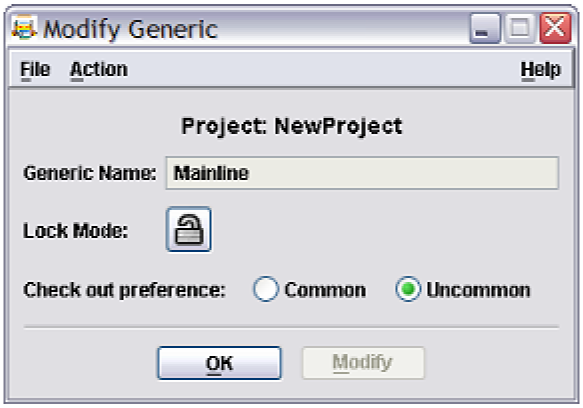
● By Generic ○ By Release

<NONE>

[6] The word Generic and branch are synonymous in SpectrumSCM

Press the “Add Generic” button and then click “OK” when the popup comes up. Set the check out preference and the locking mode on the generic:

Figure #10



Project: NewProject

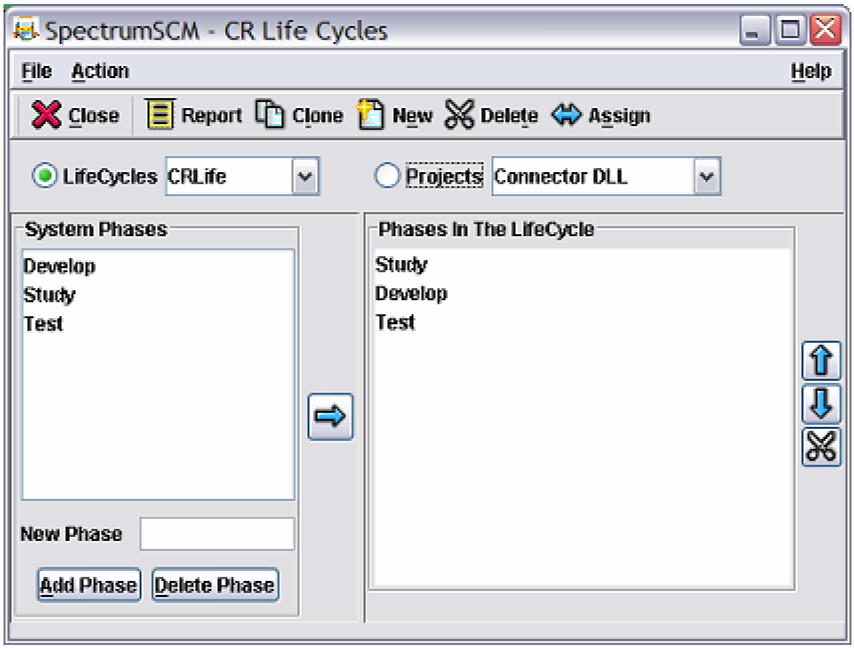
Generic Name: Mainline

Lock Mode:

Check out preference: ○ Common ● Uncommon

- Create and Assign Project Lifecycle: Assign the project’s life cycle and select the last development phase:

Figure #11



● LifeCycles CRLife

○ Projects Connector DLL

System Phases

Develop
Study
Test

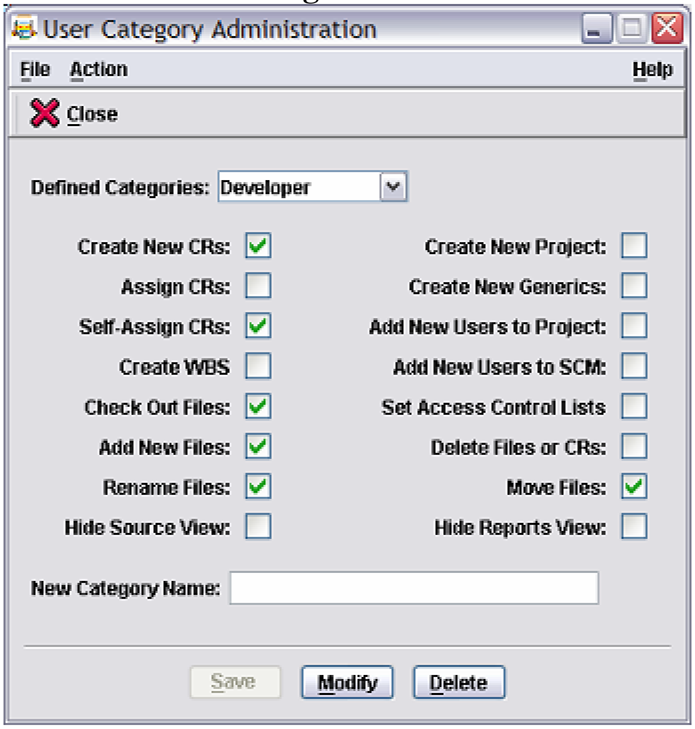
New Phase

Phases In The LifeCycle

Study
Develop
Test

- Create User Roles and Permissions:

Figure #12



Defined Categories: Developer

Create New CRs: ☑

Assign CRs: □

Self-Assign CRs: ☑

Create WBS □

Check Out Files: ☑

Add New Files: ☑

Rename Files: ☑

Hide Source View: □

Create New Project: □

Create New Generics: □

Add New Users to Project: □

Add New Users to SCM: □

Set Access Control Lists □

Delete Files or CRs: □

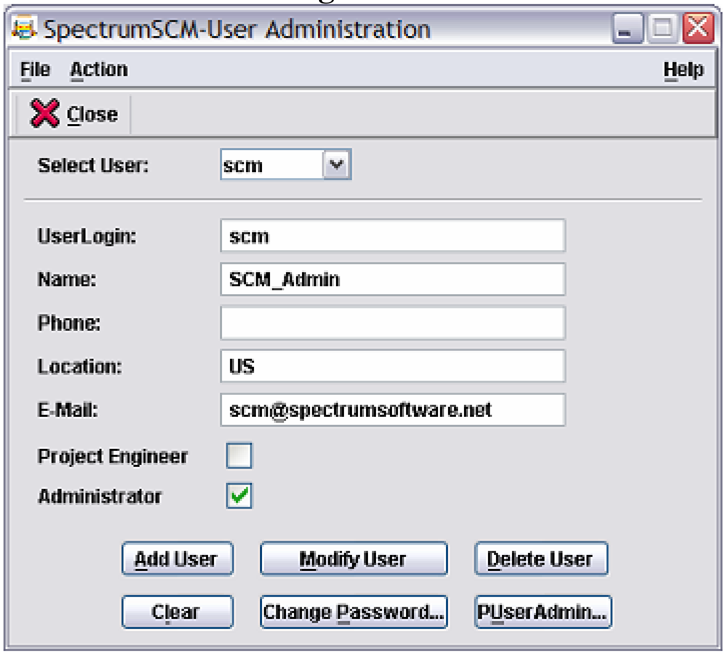
Move Files: ☑

Hide Reports View: □

New Category Name:

- Add Users to the System:

Figure #13



Select User: scm

UserLogin: scm

Name: SCM_Admin

Phone:

Location: US

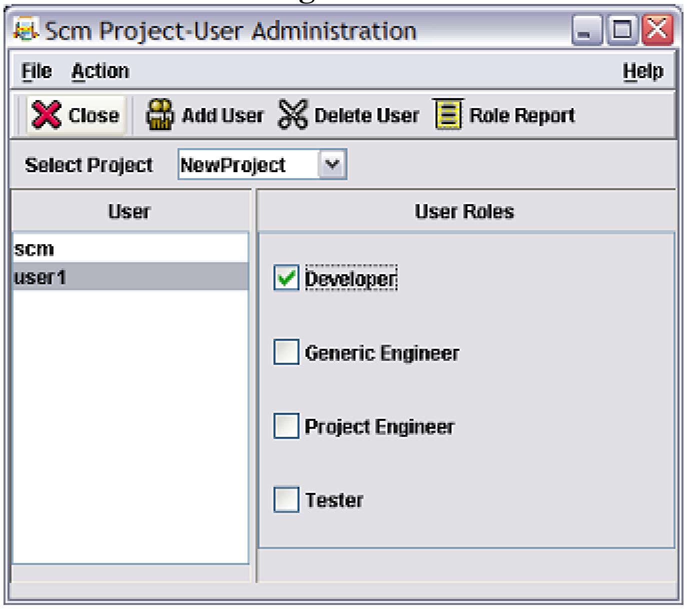
E-Mail: scm@spectrumsoftware.net

Project Engineer □

Administrator ☑

- Assign users to the Project: Assign users to the project and associate each user with one or more roles.

Figure #14



Select Project NewProject

User
scm
user1

User Roles

☑ Developer

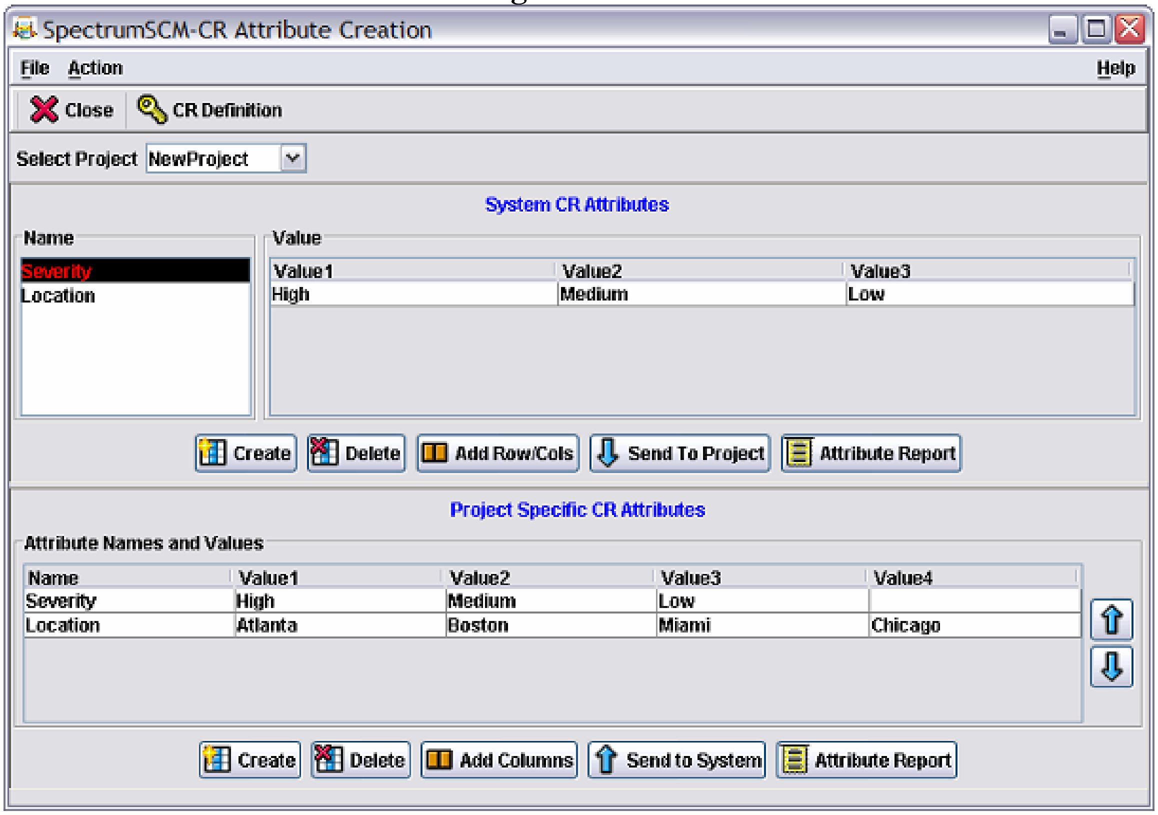
□ Generic Engineer

□ Project Engineer

□ Tester

- Create additional Change Request Attributes: Add additional CR attributes.

Figure #15



Select Project NewProject

System CR Attributes

Name

Severity
Location

Value

Value1	Value2	Value3
High	Medium	Low

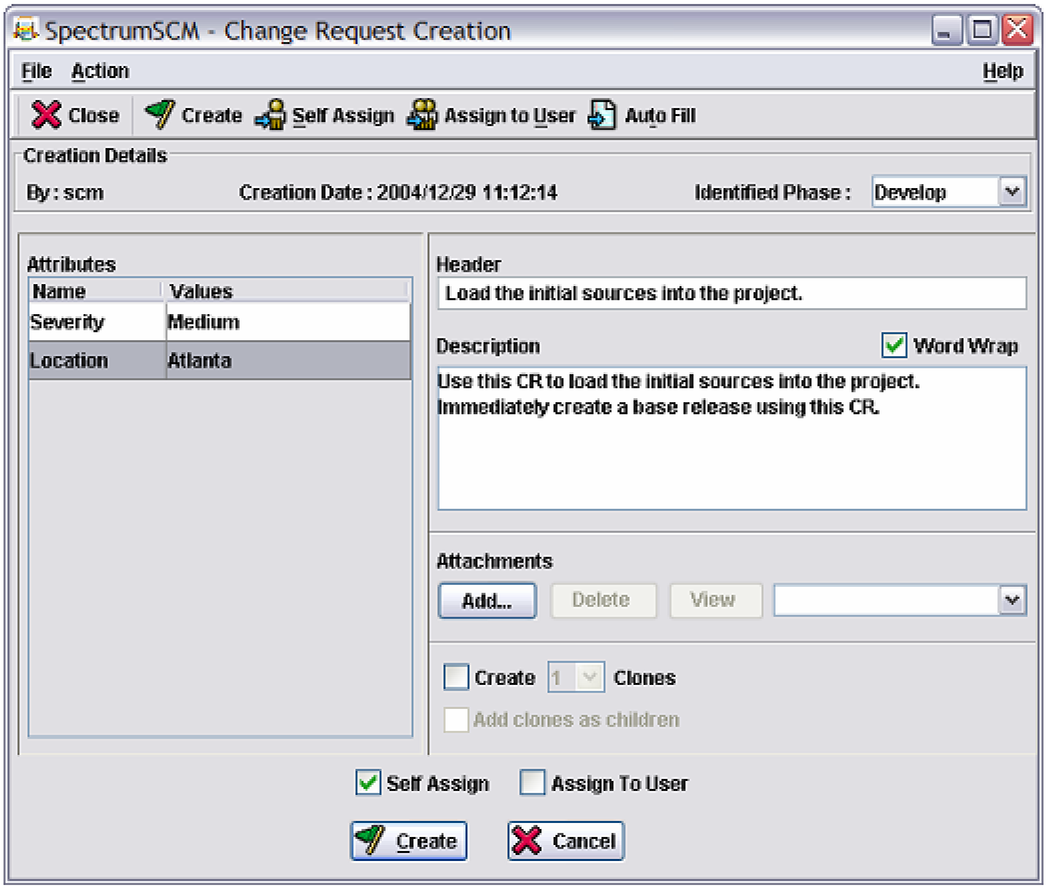
Project Specific CR Attributes

Attribute Names and Values

Name	Value1	Value2	Value3	Value4
Severity	High	Medium	Low	
Location	Atlanta	Boston	Miami	Chicago

- Create the Initial CR: Now add the first CR to the new project so that a CR will be available to add the new sources into the system from Visual Studio:

Figure #16



Creation Details

By : scm Creation Date : 2004/12/29 11:12:14 Identified Phase : Develop

Attributes

Name	Values
Severity	Medium
Location	Atlanta

Header

Load the initial sources into the project.

Description ☑ Word Wrap

Use this CR to load the initial sources into the project.
Immediately create a base release using this CR.

Attachments

□ Create 1 Clones

□ Add clones as children

☑ Self Assign □ Assign To User

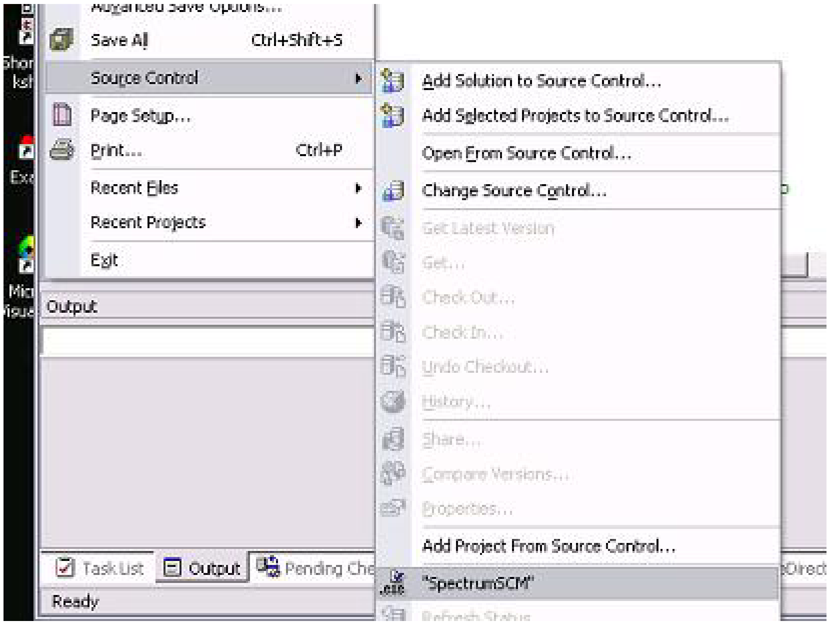
After following the directions outlined above, the new project is ready for use within SpectrumSCM. The next step will be to actually create a Visual Studio project and to bind that project to the same project name within SpectrumSCM. This was discussed in section 3.2 (Project Mapper Configuration)

5.1 Adding a Visual Studio Project to SpectrumSCM:

To add a new Visual Studio project to SpectrumSCM, start Visual Studio and create a new project as usual. Once the project has been created, it can be added to source code control right away, or can be deferred until a reasonable amount of code has been added to the new project. Of course, if the project must be shared right away, then the project must be added to source control immediately.

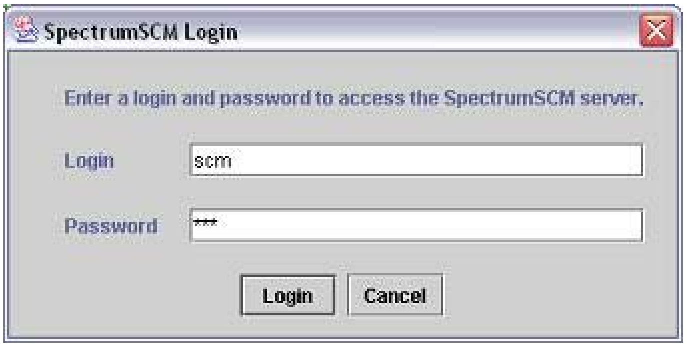
Because the new Visual Studio project is not yet bound to a source code control system, the SpectrumSCM dashboard must be started manually. Once the Dashboard has been started, and the new project added to source code control, the Visual Studio project will be permanently bound to SpectrumSCM and the Dashboard will not need to be started manually again. To start the Dashboard for the first time, use the *File->Source Control->SpectrumSCM* menu item to launch the Dashboard. Figure #17 illustrates this action:

Figure #17



When the Dashboard starts, it will immediately post a login dialog box to the screen. Login to the system using the user-id of the user that was assigned the CR created in the previous section, e.g. if the new CR was assigned to user Joe, then login as user Joe:

Figure #18



Enter a login and password to access the SpectrumSCM server.

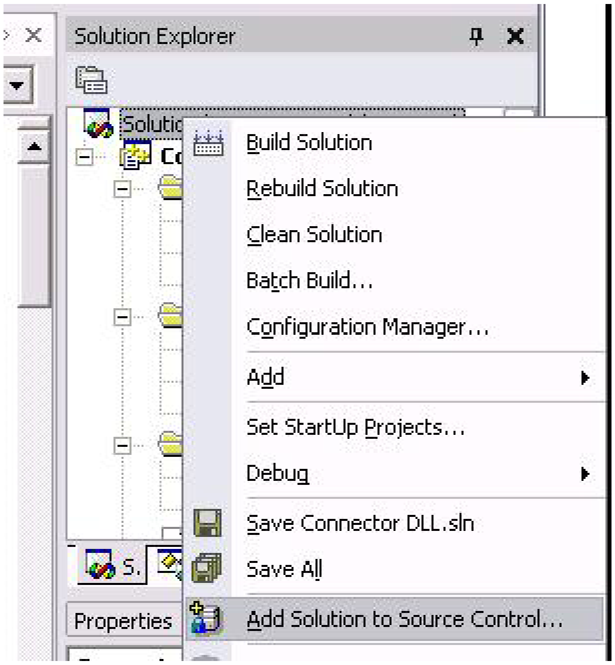
Login scm

Password ***

Because the Visual Studio project is not bound to a particular SpectrumSCM project at this point, the user will need to use the Project Mapper to configure the association between the project IDE and the SpectrumSCM repository project. See section 3.2 for details on how to do this. Once the project association has been setup properly, the Dashboard project will automatically switch to reflect the SpectrumSCM repository associated with the current Visual Studio project.

Right click the "Connector DLL" solution in the solution browser, which is in the upper right hand corner of the screen, and then pull down to the "Add Solution to Source Control…" menu option:

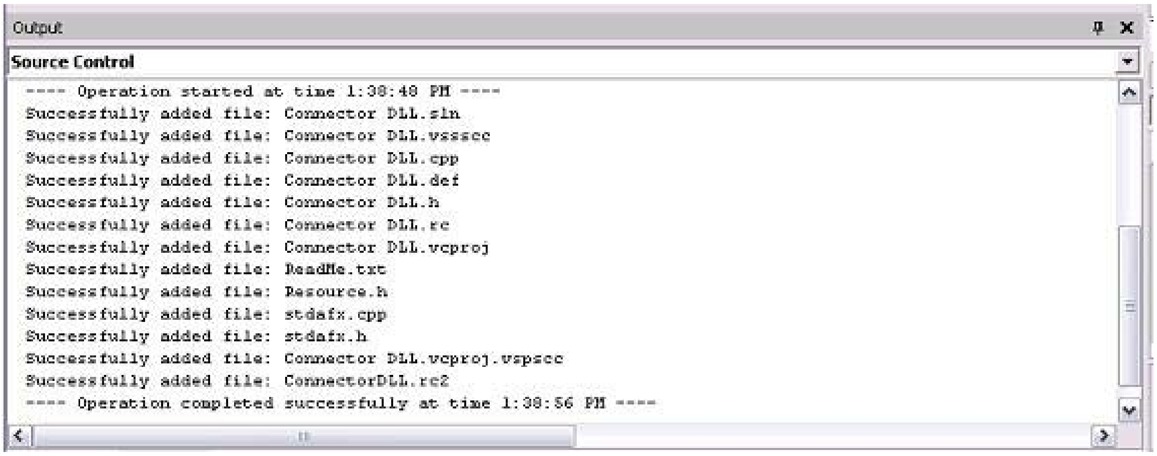
Figure #19



When the user adds the solution, the Dashboard will automatically switch to the proper SpectrumSCM repository and use the Local Root Directory entered in the Project Mapper configuration to determine how much of each project path should be added to the SpectrumSCM repository itself.

The next two figures illustrate some of the messages that will be visible during the add solution action:

Figure #20

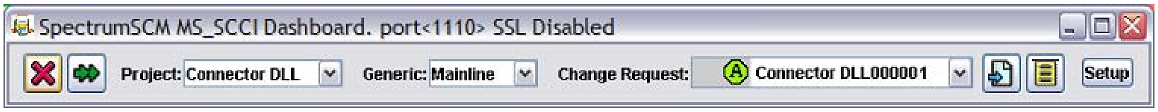


Source Control

```
 ---- Operation started at time 1:38:48 PM ----
 Successfully added file: Connector DLL.sln
 Successfully added file: Connector DLL.vsscc
 Successfully added file: Connector DLL.cpp
 Successfully added file: Connector DLL.def
 Successfully added file: Connector DLL.h
 Successfully added file: Connector DLL.rc
 Successfully added file: Connector DLL.vcproj
 Successfully added file: ReadMe.txt
 Successfully added file: Resource.h
 Successfully added file: stdafx.cpp
 Successfully added file: stdafx.h
 Successfully added file: Connector DLL.vcproj.vspscc
 Successfully added file: ConnectorDLL.rc2
 ---- Operation completed successfully at time 1:38:56 PM ----
```

Note that in the this figure, all of the files associated with the solution, including the project file and the solution file itself we're successfully added to source code control. The Dashboard will now reflect the correct synchronization between itself and Visual Studio:

Figure #21

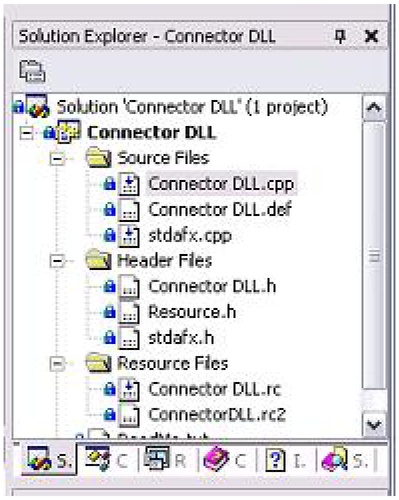

SpectrumSCM MS_SCCI Dashboard. port<1110> SSL Disabled

Project: Connector DLL Generic: Mainline Change Request: Connector DLL000001

This boot strap activity only has to happen once. Now that the Visual Studio solution is bound to SpectrumSCM as the CM provider, the Dashboard will be automatically started by Visual Studio each time the user opens the solution. Also note that the Project selection combo box on the Dashboard will be reduced to a single entry, which is the project that Visual Studio is active against. When a new project is selected in Visual Studio, the Dashboard will automatically switch over to the new project and reset the available generics and change requests.

Once the solution has been checked into SpectrumSCM, the solution browser in Visual Studio will change the icons associated with the resources in the project to reflect the state change:

Figure #22



Note the <lock> symbol next to each resource in the solution, including the project and the solution itself. Visual Studio interviews the CM system from time to time to determine the status of the files in the project. If Visual Studio can't contact the CM solution for whatever reason, there will be no icons associated with the resources. When a resource is checked out by the current user or another user, the icon next to that particular resource will change to reflect that status.

6.0 Visual Studio Standard CM Operations:

SpectrumSCM is a true SCM tool and as such uses an issue tracking system to associate code changes with particular issues. When a user is working with a Visual Studio solution, he/she will need to select a Change Request (CR) to work against. Each user may have one or more CRs assigned to them at any time. Select the CR that corresponds to the current work activity. Use the Change Request combo box on the Dashboard to select the proper CR. Once a CR has been selected, the user is free to carry out their work as usual in Visual Studio. As check-outs and check-ins are done in Visual Studio, the Dashboard will automatically associate that work with the selected CR.

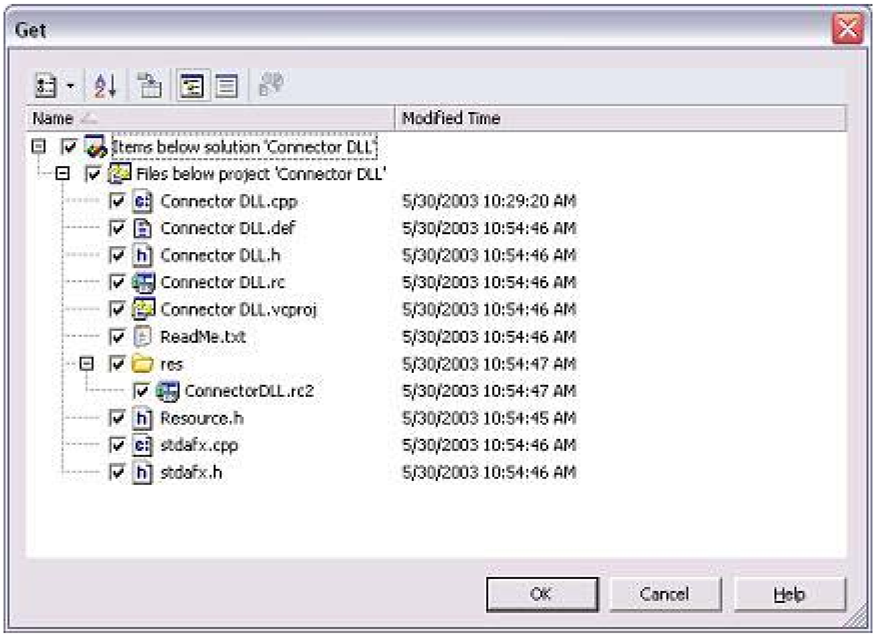
SpectrumSCM also supports concurrent work on multiple branches. Check to make sure that the correct branch of code has been selected before beginning your work. Most large scale development projects utilize multiple parallel code branches in order to allow work to continue on several projects in parallel. Use the "Generic" combo box to select the proper branch to work against. Note that the list of active CRs can and will change based on the selected branch of code.

6.1 Synchronizing the Workspace:

When multiple users work together, the CM repository will be updated from several different sources simultaneously. Users must periodically refresh their local workspace in order to be able to see new additions or changes that have been made to the system. Individual files can be refreshed simply by right clicking on the file and pulling down to the "Get Latest Version" menu selection. Entire directories, projects and solutions can be synchronized by right clicking on the item and then selecting the "Get Latest Version (recursive)" menu item. Visual Studio will automatically refresh all the files contained in the upper level directory and all sub-directories.

To better control the synchronization process, select the directory, project or solution that should be synchronized, then press the *File->Source Control->Get…* menu item. Visual Studio will respond with a popup dialog which allows the user to select individual files for synchronization:

Figure #23

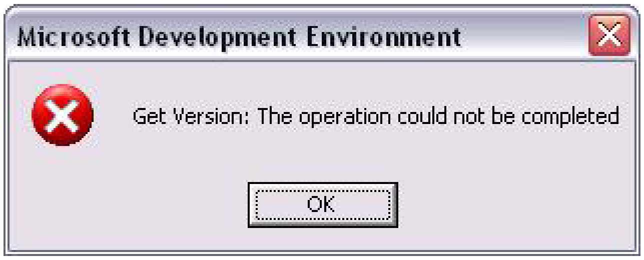


Name	Modified Time
☑ Items below solution 'Connector DLL'	
☑ Files below project 'Connector DLL'	
☑ Connector DLL.cpp	5/30/2003 10:29:20 AM
☑ Connector DLL.def	5/30/2003 10:54:46 AM
☑ Connector DLL.h	5/30/2003 10:54:46 AM
☑ Connector DLL.rc	5/30/2003 10:54:46 AM
☑ Connector DLL.vcproj	5/30/2003 10:54:46 AM
☑ ReadMe.txt	5/30/2003 10:54:46 AM
☑ res	5/30/2003 10:54:47 AM
☑ ConnectorDLL.rc2	5/30/2003 10:54:47 AM
☑ Resource.h	5/30/2003 10:54:45 AM
☑ stdafx.cpp	5/30/2003 10:54:46 AM
☑ stdafx.h	5/30/2003 10:54:46 AM

The user can select the individual files they want to synchronize or unselect any files that they do not want synchronized.

If the user has a file already out for edit, and chooses to include that file in the synchronization process, either intentionally or by accident, Visual Studio will complain that the over-all action has failed and will post in the output window which files it succeeded in synchronizing and which files failed. The next two figures illustrate this situation:

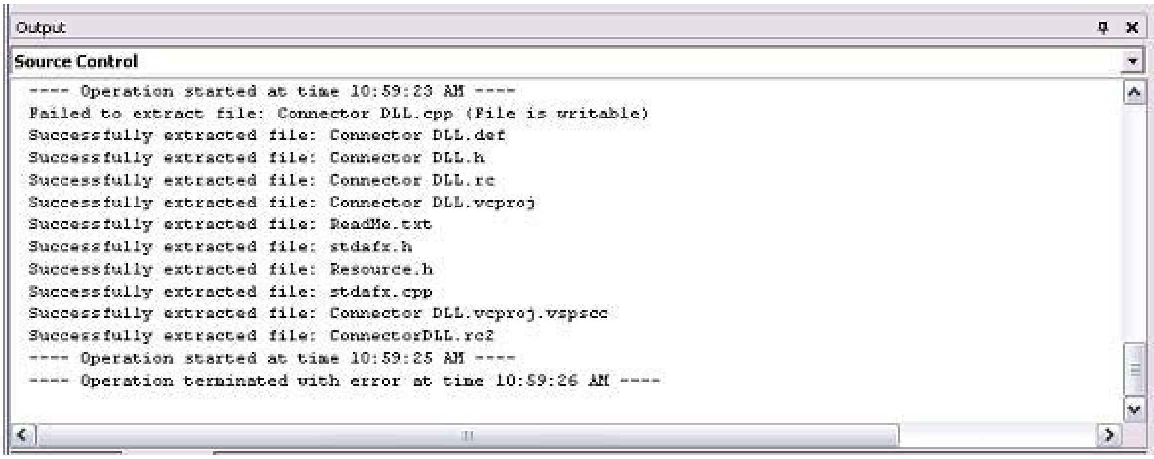
Figure #24



Microsoft Development Environment

Get Version: The operation could not be completed

Figure #25



```
 ---- Operation started at time 10:59:23 AM ----
 Failed to extract file: Connector DLL.cpp (File is writable)
 Successfully extracted file: Connector DLL.def
 Successfully extracted file: Connector DLL.h
 Successfully extracted file: Connector DLL.rc
 Successfully extracted file: Connector DLL.vcproj
 Successfully extracted file: ReadMe.txt
 Successfully extracted file: stdafx.h
 Successfully extracted file: Resource.h
 Successfully extracted file: stdafx.cpp
 Successfully extracted file: Connector DLL.vcproj.vspscc
 Successfully extracted file: ConnectorDLL.rc2
 ---- Operation started at time 10:59:25 AM ----
 ---- Operation terminated with error at time 10:59:26 AM ----
```

Note that the file "Connector DLL.cpp" failed to extract because it is writable. The rest of the files were synchronized properly. The Solution browser also reflects the fact that the connector file is out for edit by the user.

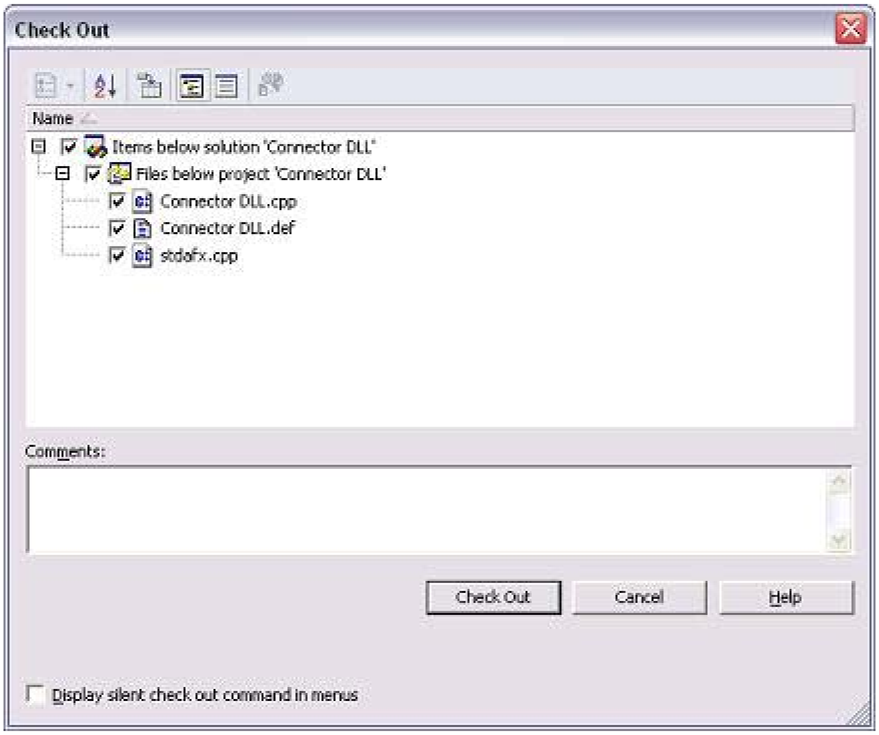
When synchronizing the workspace from the Solution or project level, Visual Studio will automatically add new files to the workspace or remove old files, based on the changes in the project file itself.

Tip: Synchronize your workspace only after you have finished a unit of work, or just before you check your work in. This will help to minimize any unnecessary churn while you are completing a unit of work. Synchronize your workspace just before you check in your work so that you can verify that your work is compatible with the changes that others have made in parallel to your work. Merge the work of others into your work, recompile the project and if everything is working to your satisfaction, check in all of your work. See section 6.7 on visually differencing your workspace files against the files in the CM repository

6.2 Check Out a File:

Individual files can be checked out by right clicking on the selected file and choosing the "Check Out…" menu option. Multiple files may be checked out simultaneously by selecting a range of files and then right clicking one of the selected files and choosing the "Check Out…" menu option. When checking out multiple files at once, Visual Studio will respond with a check out dialog that allows the user to customize the file selection:

Figure #26

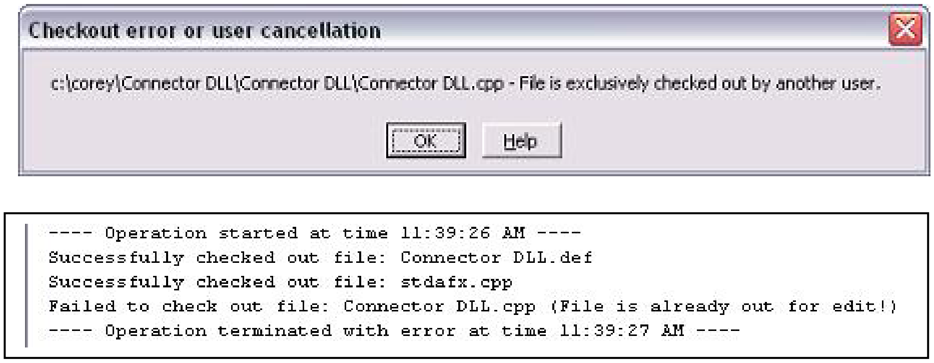


☑ Items below solution 'Connector DLL'
- ☑ Files below project 'Connector DLL'
  - ☑ Connector DLL.cpp
  - ☑ Connector DLL.def
  - ☑ stdafx.cpp

□ Display silent check out command in menus

If one or more of the files in the selection is already out for edit by another user, Visual Studio will respond with a notification informing the user why the files cannot be checked out. The Output window of Visual Studio will also reflect the status of the operation:

Figure #27



Checkout error or user cancellation

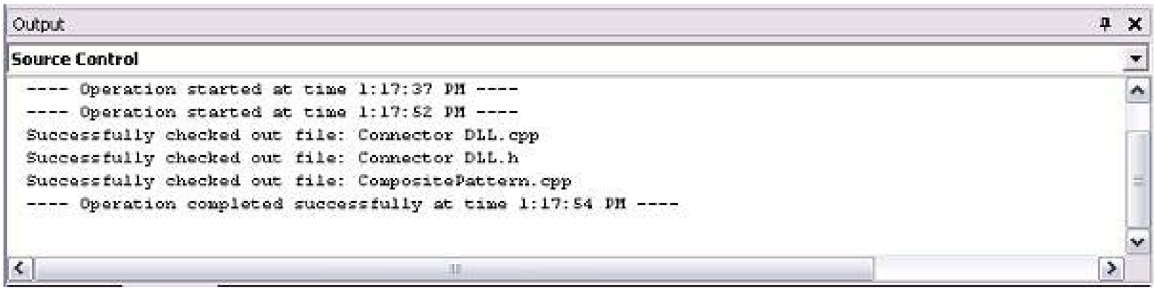
c:\corey\Connector DLL\Connector DLL\Connector DLL.cpp - File is exclusively checked out by another user.

```
---- Operation started at time 11:39:26 AM ----
Successfully checked out file: Connector DLL.def
Successfully checked out file: stdafx.cpp
Failed to check out file: Connector DLL.cpp (File is already out for edit!)
---- Operation terminated with error at time 11:39:27 AM ----
```

6.3 Check In a File:

Checking in a single file or a selection of files is basically the same as checking out a single file or a selection of files. Select a single file or selection of files, right click and select the "Check In…" menu option. When multiple files are selected, Visual Studio will respond with a dialog box which allows the user to fine tune the check in. Or, in the case of a single file, Visual Studio will proceed to check in the file.

Upon successful completion of the operation, the Visual Studio output window will reflect the status of the check-ins and the Solution Browser will change the icons on the resources back to "locked" status.

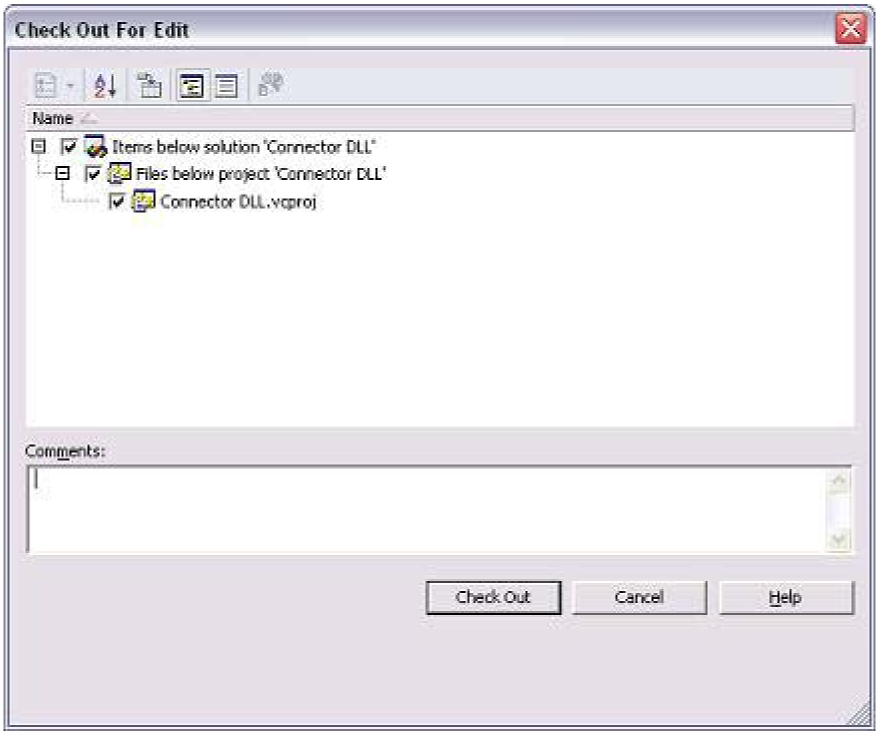


```
 ---- Operation started at time 1:17:37 PM ----
 ---- Operation started at time 1:17:52 PM ----
 Successfully checked out file: Connector DLL.cpp
 Successfully checked out file: Connector DLL.h
 Successfully checked out file: CompositePattern.cpp
 ---- Operation completed successfully at time 1:17:54 PM ----
```

6.4 Automatic Check Outs:

Some Visual Studio operations will result in the automatic check out of some files from SpectrumSCM. In particular, adding a new file to a project will result in Visual Studio automatically checking out the project file in order to record the fact that a new resource is being added to the overall solution. This happens in two phases. First Visual Studio will post a popup so that the user can confirm the check out of the project file:

Figure #28



Check Out For Edit

- ☑ Items below solution 'Connector DLL'
  - ☑ Files below project 'Connector DLL'
    - ☑ Connector DLL.vcproj

Then, once the project file has been successfully checked out (it might already be out for edit by another user), the user can add the new resource to the solution. Remember to add some content to the new resource before checking it into source control. SpectrumSCM will reject check-ins of zero length files.

The user can also check the project file out directly first, and then add the new resource as usual. This will prevent the user from having to enter the name of the new resource twice.

> Tip: Always check the project file back in immediately after adding or deleting a resource. This will allow other users to synchronize their work spaces and to add new resources of their own.

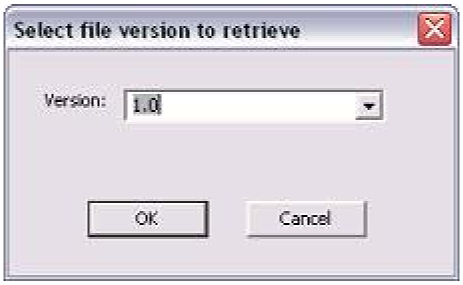
6.5 Extracting the Latest Version of a File:

The latest version of a file can be extracted by selecting the file, right clicking and then selecting the "Get Latest Version" menu option.

6.6 Extracting a Previous Version of a File:

A previous version of a file can be extracted by using the *File->Source Control->Get <filename>* menu option. Visual Studio will respond with a popup that allows the user to customize the selection even though only one file has actually been selected. On pressing "OK", a second popup will be displayed which contains a selection of all of the selected file's previous versions. Select one of the previous versions and press "OK" to get that version of the file:

Figure #29



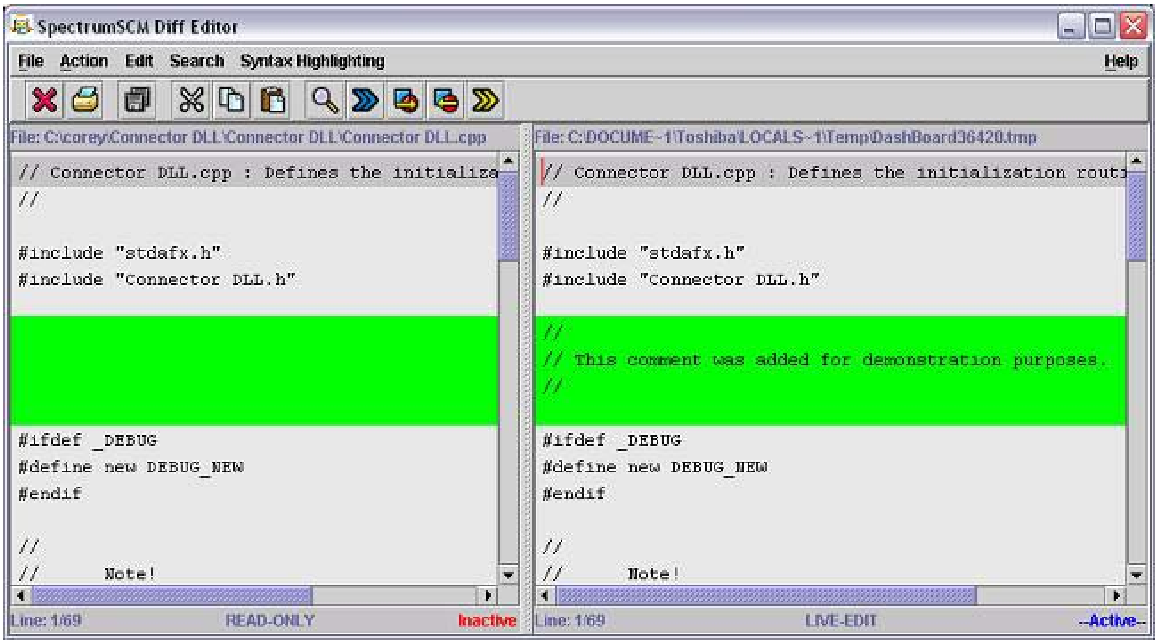
Version: 1.0

Previous versions of files cannot be edited. If the user needs to roll back to a previous version of a file, the file needs to be checked out and the contents of that file need to be replaced with that of the earlier file.

6.7 Visually differencing the Workspace File:

The SpectrumSCM SCCI integration includes a 2-way file differencing engine that can visually display the differences between files in the SpectrumSCM repository and files in the user's local working directory. To access this feature, simple select the file, right click and hold to get the secondary menu and then select the "Compare Versions…" menu item. Visual Studio will retrieve the repository file and start the SpectrumSCM Diff/Merge tool automatically:

Figure #30



File: C:\corey\Connector DLL\Connector DLL\Connector DLL.cpp

```
// Connector DLL.cpp : Defines the initializa
//

#include "stdafx.h"
#include "Connector DLL.h"




#ifdef _DEBUG
#define new DEBUG_NEW
#endif

//
//      Note!
```

Line: 1/69 READ-ONLY Inactive

File: C:\DOCUME~1\Toshiba\LOCALS~1\Temp\DashBoard36420.tmp

```
// Connector DLL.cpp : Defines the initialization routi
//

#include "stdafx.h"
#include "Connector DLL.h"

//
// This comment was added for demonstration purposes.
//
#ifdef _DEBUG
#define new DEBUG_NEW
#endif

//
//      Note!
```

Line: 1/69 LIVE-EDIT --Active--

In this example, the file on the left is the file in the users local work space and the file on the right is a representation of the file in the repository. The green color bar indicates that 4 lines have been added to the file in the repository that don't exist in the file from the local work space. The diff/merge tool can be instructed to diff the two files in one of two directions. When running the diff from left to right, the tool will display operations that need to take place on the file on the left to make it look like the file on the right. When running the difference engine from right to left, the same thing happens but in reverse. In this case the green bar changes to a red bar when the direction of the difference changes. When lines in the diff/merge tool are in direct conflict, they will be displayed in yellow. This list summarizes the colors the tool can display and what each color indicates:

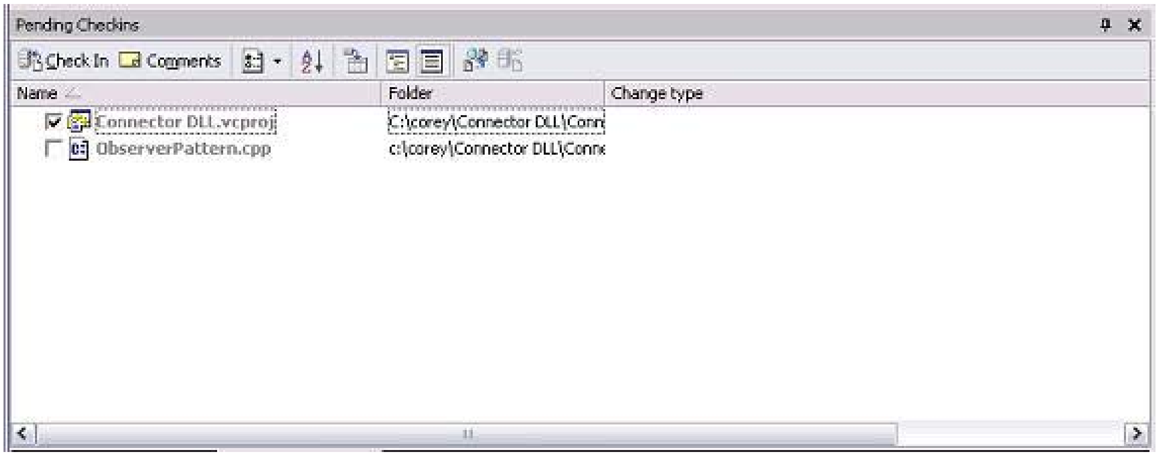
- Red: These lines do not exist in the other file and should be deleted.
- Green: These lines do not exist in the other file and should be added.
- Yellow: These lines are in direct conflict and should be changed.

The right hand editor always represents the file from the repository. It is not the actual repository file but a temporary file that can be edited, saved or discarded. The diff/merge tool also contains a subset of standard editor functionality including cut, copy, paste, search and replace, save and print.

6.8 Adding a New Resource to Your Project:

As previously documented in section 6.4 on automatic check out, adding a new resource to a Visual Studio project operates in two distinct phases. In the first phase, the project related files must be checked out. In the second phase, the new sources can be added to the system. It is a best practice to check the project related files back in immediately after the new resource has been added so that other users will have access to those files. Use the "Pending Checkins" Visual Studio window to view and select the files for check-in, then press the check in button to activate the check-in operation.

Figure #31



Name	Folder	Change type
☑ Connector DLL.vcproj	C:\corey\Connector DLL\Conn	
□ ObserverPattern.cpp	c:\corey\Connector DLL\Conn	

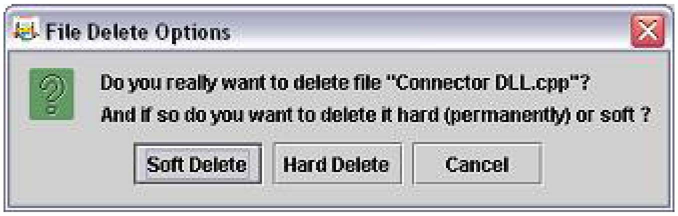
In this example, the project file has been selected for check-in. The new resource file, ObserverPattern.cpp has been de-selected so that work can continue on this file. When the user presses the "Check-In" button at the top left of the screen, the project file will be checked back into the repository and the new resource will be left checked-out.

6.9 Removing a File:

Removing a file is very similar to adding to new file. When a resource is selected for removal by the user, Visual Studio will inform the user that the project related files must be checked out first. Once the project file has been checked out, the user will be able to successfully delete the resource. Files that are removed from a Visual Studio project are not automatically deleted from the SpectrumSCM repository. Deleting files from the source control repository is a privileged operation, which ensures the sanctity of previous builds and releases. Only a SpectrumSCM user with the appropriate permissions can perform this operation. To remove a file permanently from the repository, the user must use the SpectrumSCM client and either hard or soft delete the file in question. Files that are involved in a previous release cannot be permanently deleted from the repository. Files that have never been associated with a release can be permanently deleted. In either case, files can be "soft" deleted in the repository. Soft deletion removes the file from the users view but can also be retrieved at a later date, if necessary.

To soft or hard delete a file from the SpectrumSCM repository, open a SpectrumSCM client, select the file in question and then click the *Administration->Delete…* menu item. The system will respond with a popup that allows the user to choose the type (hard or soft) of deletion.

Figure #32



File Delete Options

Do you really want to delete file "Connector DLL.cpp"?
And if so do you want to delete it hard (permanently) or soft ?

Files that have been soft deleted can be retrieved by using the "Delete Log" facility on the SpectrumSCM client main screen. Delete log presents to the user a list of items that have been deleted, their status (whether they can be retrieved or not) and the user-id of the person that actually deleted the resource. Soft deleted resources can be retrieved by selecting the resource and pressing the "UnDelete" button.

6.10 Switching Projects:

Visual Studio allows the user to have several projects open at one time, either through the same instance of Visual Studio or by allowing the user to start separate instances of the product. To open another project within the same instance of Visual Studio, navigate back to the "Start Page" and simply select the next project to work on. Visual Studio will automatically tell the SpectrumSCM Dashboard to switch the user to the selected project

6.11 Output Messages:

There are several different ways that Visual Studio communicates with the end user. The Visual Studio Output window, located on a tabbed frame at the lower left of the primary window, is the primary output message area for all CM related activities. An example of this is illustrated in section 6.2 of this document.

Hard errors and warnings are presented to the user as standard Visual Studio error or warning message dialogs. The Dashboard can also present error message dialogs directly to the user, but only under the most serious of circumstances. For instance, if the Dashboard becomes disconnected from the SpectrumSCM server, the Dashboard will post an error dialog describing the problem.

7.0 Working with the Dashboard:

The primary advantage of working with the SpectrumSCM Dashboard, is that it helps to reduce the number of extra popup windows that the user must navigate through during the course of their normal development activities. The Dashboard is essentially a reduced functionality version of the main SpectrumSCM client program. The Dashboard allows the user to work directly with Change Requests and it also allows the user the ability to quickly switch between code branches.

7.1 Logging in:

The Dashboard will automatically prompt the user for a login and password if one has not been entered for the current working session. The user also has the ability to temporarily release the server side license when it is not needed. For example, when the user needs to run both the Dashboard and the standard SpectrumSCM client program at the same time, but only wants to consume a single license, the license for the Dashboard can be temporarily released. Any new activities issued against the Dashboard will automatically cause the Dashboard to reinstate the suspended license. See section 7.8 on releasing the server side license.

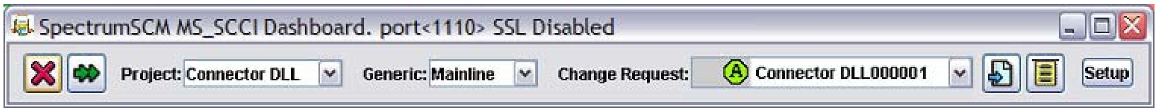
7.2 Selecting Branches:

Users often switch between branches of code during the normal work day. The Dashboard makes switching between code branches a simple operation.

> Tip: Always check to make sure your current work is checked back in before switching to a new branch. If files are left checked out when a new branch is opened, the status of the files in the solution browser will not match the actual file status.

Use the "Generic:" combo box to select a branch to work on:

Figure #34



SpectrumSCM MS_SCCI Dashboard. port<1110> SSL Disabled

Project: Connector DLL

Generic: Mainline

Change Request: Connector DLL000001

In this example, the "Mainline" generic[7] has been selected for work. When the user selects a new generic, the list of change requests assigned to the user automatically switches to the list of CRs assigned to the user, on the selected generic. The user should immediately synchronize their workspace with the repository since the project file and the contents of the branch may be different from what's in the user's workspace. See section 6.1 on synchronizing the workspace to the repository.

[7] The word "Generic" means "Branch" in SpectrumSCM.

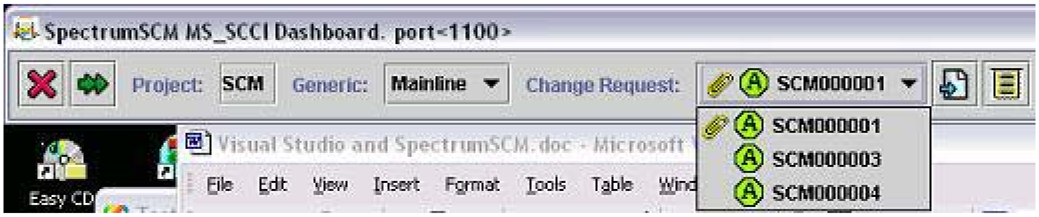
7.3 Selecting Change Requests:

All work done against a SpectrumSCM repository must be done against a Change Request. This is done for accountability reasons and, more importantly, so that releases can be formed around sets of issues and not just labels applied across the CM repository. In essence, a list of CRs in a release forms a bill of materials (BOM) for that release. Working with change requests is very easy. Simply select the CR to work against and the Dashboard will do the rest. Once a CR has been selected, all the work done by the user will be automatically associated with the selected CR.

> Tip: Users should group their work together into a series of CRs instead of placing all of their work into a single CR. This becomes much easier in the later phases of development as bugs or new features can be broken down into separate CRs.

Use the Change Request combo box to select the CR to work against:

Figure #35



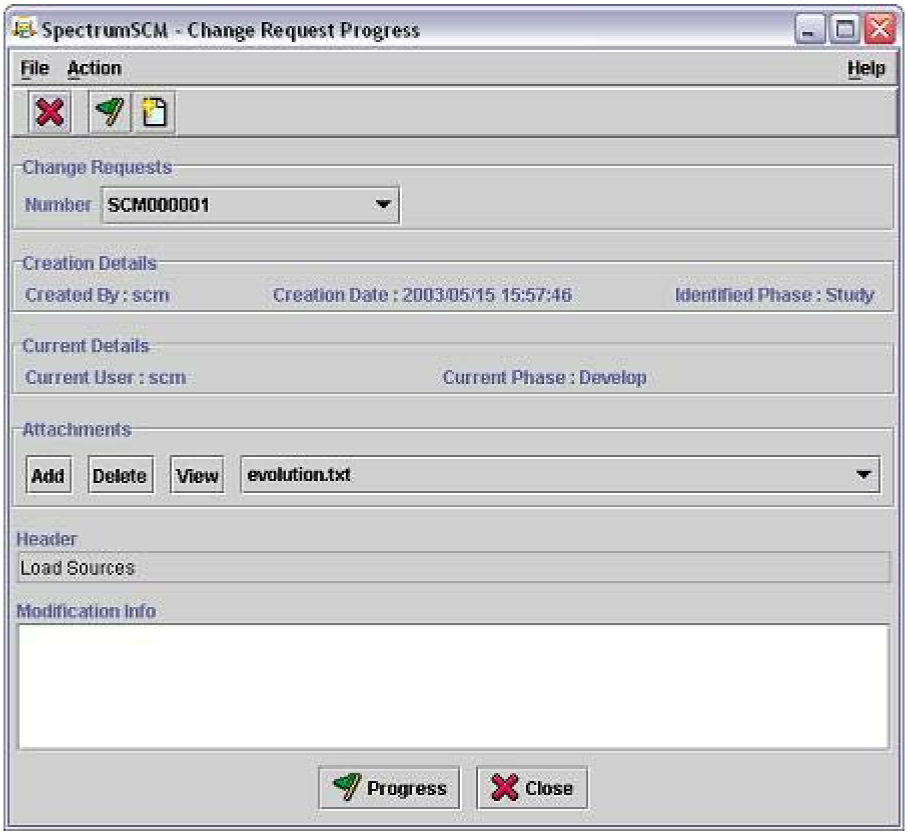
Change Requests can have several icons associated with them in the combo box. The "Gem Clip" icon informs the user that "attachments" have been added to this CR. The "Green A" icon simply tells the user that this CR is assigned to them. On the SpectrumSCM client main screen, the green "A" icon can also appear as a yellow "T" for Generic Engineers and Project Leaders. The Yellow icon indicates that the CR is in the TBA[8] state and needs to be assigned for work.

[8] TBA = To Be Assigned state. When users progress a CR out of a state the CR is automatically sent to the TBA state and e-mail is sent to the appropriate users. It is the responsibility of the Generic Engineer or Project Leader to assign the CR to the next person in the appropriate phase. If automatic workflow is enabled all of this will happen automatically behind the scenes.

7.4 Progressing Change Requests:

Users can move change requests out of their change request work list by pressing the progress icon, which is located to the right of the change request combo box. The progress icon invokes the change request progression screen illustrated in the next figure:

Figure #36



SpectrumSCM - Change Request Progress

Change Requests

Number SCM000001

Creation Details

Created By : scm

Creation Date : 2003/05/15 15:57:46

Identified Phase : Study

Current Details

Current User : scm

Current Phase : Develop

Attachments

evolution.txt

Header

Load Sources

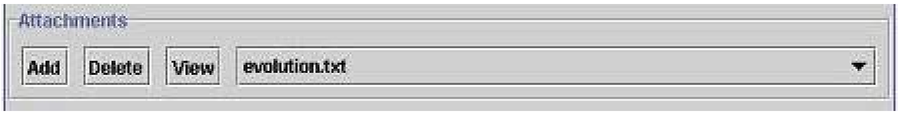
Modification Info

The progression screen allows the user to add additional notes to the selected CR, add, view, delete CR attachments and add modification information to the CR. Notes and attachments can be added to the CR at any time without having to progress the CR out of the users list of active CRs.

To add a new note to the selected CR, use the "Add Note" icon located in the tool bar at the top of the window. The user will be presented with a small popup that allows them to add a note of any length to the CR. When the popup is closed, the note will be permanently associated with the CR. CR notes are immutable.

To add an attachment to a CR, use the attachment portion of the progression screen:

Figure #37



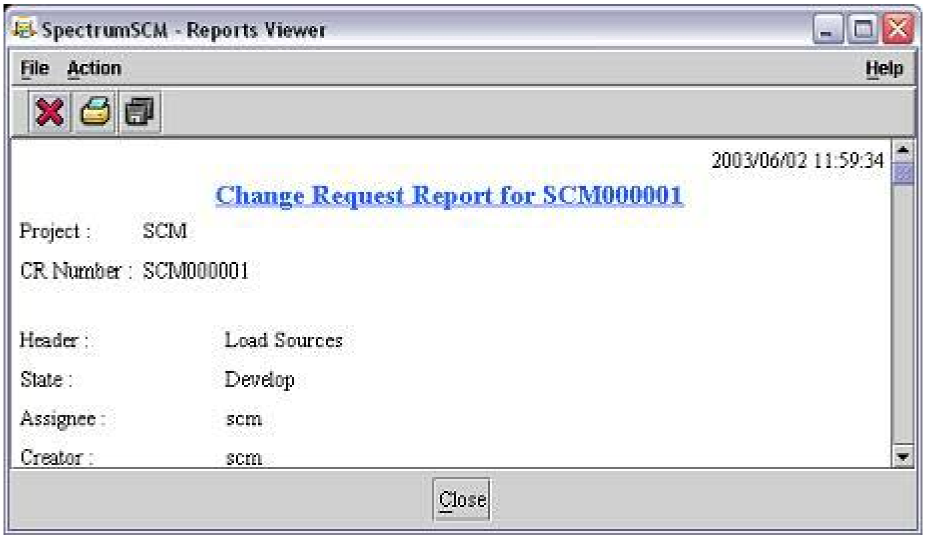
Attachments are not versioned items in SpectrumSCM, they are simply items associated with a CR that may or may not be plain text. In this example the attachment is an ASCII text attachment, but attachments are not limited to text files. Screen shots, Word documents and other types of binary files can also be attached. The user's custom editor preference will dictate how these attachments are viewed by the user. On the Windows platform, users should elect to use the default "regassoc" custom editor. Regassoc is not really an editor, it is a hook into the Windows registry that allows Windows to decide how a document should be opened and presented to the user.

Finally, when the user has added any additional notes, modification notes or attachments, the user can press the "Progress" button located at the bottom of the screen to invoke the progress action. Once the CR has been progressed, it will disappear from the users CR list.

7.5 Executing a Change Request Report:

The ☰ icon, when selected, executes the SpectrumSCM Change Request report against the selected CR. The CR report dumps all the associated information for the selected CR into HTML output text. The report can be viewed through the standard SpectrumSCM HTML report viewer, or can automatically displayed in Internet Explorer, Netscape or any other HTML viewer. To use a viewer other than the default viewer, select *Edit->Preferences->Custom Report Viewer* on the SpectrumSCM main display and enter the path to the viewer of choice. The following example illustrates part of the output for a CR report displayed in the standard report viewer:

Figure #38



2003/06/02 11:59:34

Change Request Report for SCM000001

Project : SCM

CR Number : SCM000001

Header : Load Sources

State : Develop

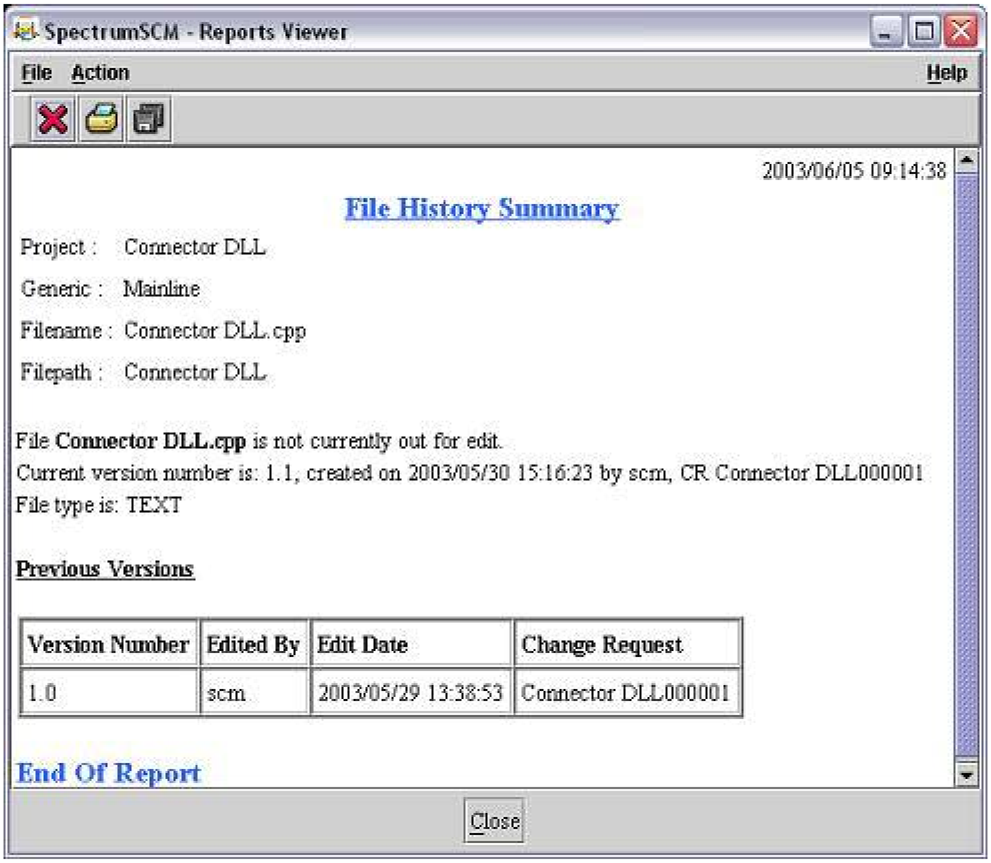
Assignee : scm

Creator : scm

7.6 Executing a File History Report

The File History Summary report details all the previous activities for the selected file. The detail includes the files previous version numbers, information about the editor for each particular version of the file, the date the change was made and the change request used to make the change. To run the summary report, select a file in the Solution Browser and then select *File->Source Control->History…* The report viewer will display the report in a separate window. Figure #35 illustrates and example of the report output window contents:

Figure #39



2003/06/05 09:14:38

File History Summary

Project : Connector DLL

Generic : Mainline

Filename : Connector DLL.cpp

Filepath : Connector DLL

File **Connector DLL.cpp** is not currently out for edit.
Current version number is: 1.1, created on 2003/05/30 15:16:23 by scm, CR Connector DLL000001
File type is: TEXT

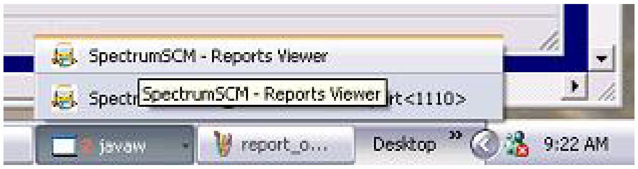
Previous Versions

Version Number	Edited By	Edit Date	Change Request
1.0	scm	2003/05/29 13:38:53	Connector DLL000001

End Of Report

The report window, like all the other Dashboard windows run separately from Visual Studio and thus can sometimes become obscured by Visual Studio main window. After executing this report, if the window does not automatically pop to the surface, check the Windows task bar and bring the report forward. This is illustrated in figure #36:

Figure #40


SpectrumSCM - Reports Viewer
Spectr SpectrumSCM - Reports Viewer rt<1110>
javaw
report_o...
Desktop
9:22 AM

7.7 Adding Notes to a Change Request:

See section 7.4 on CR progression for a detailed description of how to add notes to a change request.

7.8 Adding/Viewing/Deleting Change Request Attachments:

See section 7.4 on CR progress for a detailed description of how to add attachments to a change request.

7.9 Releasing the Server License:

Because a SpectrumSCM Dashboard user might also need to use the full SpectrumSCM client software, the Dashboard has the ability to temporarily release the server side license. The ☒ icon allows the user to temporarily drop the server side license. The license will be automatically reinstated when the user activates a CM related request from Visual Studio.

7.10 Refreshing the Project:

The SpectrumSCM Dashboard operates just like a web browser when it comes to accepting updates from the SpectrumSCM server. All updates to the Dashboard must be requested by the user by selecting the refresh ⇔ icon. When the refresh action is selected, the dashboard will contact the SpectrumSCM server and download the following items:

- Additional projects, or new project assignments
- New generics
- New or reassigned Change Requests

This operation is under the control of the user because it is a fairly expensive operation to run from a bandwidth perspective. In a large organization with more than a handful of users attached to the SpectrumSCM server at one time, it would be prohibitively expensive to automatically "push" this type of information to the client on a regular basis.

The user should use the refresh capabilities of the Dashboard periodically to refresh their CR and generic lists. E-mail notifications are always sent to a user whenever CRs are assigned. The user can use the e-mail notification as a prompt to refresh the project.

8.0 Summary:

The SpectrumSCM SCCI Implementation (SpectrumSCCI) is a fairly unique implementation of the Microsoft Common Source Code Control Interface Specification. Specifically, because of the tight integration between the SCCI and the SpectrumSCM Dashboard, users can use almost all of the features of SpectrumSCM that make it a truly integrated SCM system, directly through Visual Studio.

- The ability to associate work directly with a traceable change request document not only allows the developer to organize their own work, but it also allows project engineers and development managers the ability to track and control the progress of the total development effort.
- Releases of the product are based on collections of known units of work and not just collections of files with various revision numbers.
- Previous releases of products can be easily extended to create patches or customized versions.
- Users can easily work in parallel on separate branches of the code and can switch between development branches through the selection of a menu item.
- Users can quickly switch between entire projects or can run multiple instances of Visual Studio in parallel, without having to reconfigure their CM integration parameters.

9.0 Known Issues:

Currently the SpectrumSCM SCCI integration does not support the following Visual Studio menu selections:

- **File->Source Control->Add Project From Source Control…** This option is supposed to allow the user to pull resources from one Visual Studio project into another. This operation is not directly supported. To perform the same functionality, use the SpectrumSCM client to download the required project items and then add the items to the project as usual.
- **File->Source Control->Open From Source Control…** This option allows the user to open a Visual Studio project directly from source control. The option is not supported at this time. To perform the same functionality, use the SpectrumSCM client to download the project and then double click the solution file.