SpectrumSCM® Version 3.0

SpectrumSCM User Guide

Spectrum Software Inc.

5400 Laurel Springs Parkway
Suite 1004
Suwanee, GA 30024
(770) 448 8662
e-mail info@spectrumsoftware.net
Website: www.spectrumsoftware.net
www.spectrumscm.com

For all enquiries please contact:

Spectrum Software Inc.
5400 Laurel Springs Parkway
Suite 1004
Suwanee, GA 30024
Phone no:(770) 448-8662

e-mail to: support@spectrumscm.com

Purpose

The **SpectrumSCM User Guide** provides general and procedural information needed to make effective use of SpectrumSCM. It is intended to be used in conjunction with the Online Help, which contains detailed information about each of the SpectrumSCM screens, functions and commands.

Scope

This issue of the User's Guide applies to Version 3.0 of SpectrumSCM..

Intended Audience

This guide is intended for all users of the **SpectrumSCM system.**

Organization

Chapter 1, Introduction, provides an overview of the SpectrumSCM system.

Chapter 2, Installation, describes the installation and configuration of the SpectrumSCM components (server, UI client, Applet mode for web access), how to check server status, and how to start and stop the UI and server.

Chapter 3, SpectrumSCM Server and UI Configuration, describes the SpectrumSCM Server Configuration Wizard and the SpectrumSCM UI Configuration Wizard, and the scm.properties file.

Chapter 4, SpectrumSCM Main Screen, describes the SpectrumSCM Main Screen areas and functions, the toolbar, menus, and icons.

Chapter 5, User Management, describes how to setup SpectrumSCM user ids and passwords, define user roles / categories, and assign users to projects with their proper roles for that project. Users will also learn how to customize preferences for screen look and feel, fonts, and editors.

Chapter 6, Process Management, describes how to set up a project environment to support your development process, including setting up a project under SpectrumSCM, establishing the project life cycle, and creating a generic.

Chapter 7, Change Requests and Issue Tracking, describes the various fields in a change request form, how to create change requests, assign change requests to an individual for work, manage change requests, and use change requests to manager the work of a development effort. You will also learn how to customize change request attributes for each project.

Chapter 8, Version Control and Source File Management, describes how version control is managed by SpectrumSCM, how to check-out or extract files for read only or edit purposes, check-in files that were checked out, unlock a file, load an entire source tree, check-in a new file, do bulk check-in / check-out, and edit files using the default SCM editor or custom editor.

Chapter 9, Release Management, describes how describes how CRs are selected to create a release, how SpectrumSCM ensures the integrity of a release, and its automatic CR dependency checking. Additionally, Interim Releases (phase based development or testing type informal builds), and Package Management is also discussed in this chapter.

Chapter 10, Reports, describes how to execute and view predefined reports and how a user can create and save customized reports.

Chapter 11, Branching, Merging, and Recommoning, describes how to manage branching in SpectrumSCM, how to merge two different versions of a file, how to recommon two different versions of a file, and how to use the merge editor to merge and recommon files across branches.

Chapter 12, SpectrumSCM Administrative Functions, describes the SpectrumSCM system administration functions and some project-level administration functions accessible via the Administration menu option on the Main Screen.

Chapter 13, Command Line Interface, describes command-line functionality that allows simple tasks to be performed over ASCII terminals or via command prompt. It also allows reports and builds to be performed automatically using scripts.

Chapter 14, SpectrumSCM API Concepts and Usage, describes the event triggers and interfaces implemented by the SpectrumSCM API, which allows users of the system to construct automated workflow engines and external plugins

Chapter 15, LDAP Support for SpectrumSCM, describes the LDAP support in SpectrumSCM that allows users to authenticate against and import users from external LDAP server

Chapter 16, Graphs and Charts, describes how to execute and view predefined graphs and how a user can create and save customized graphs & charts.

TABLE OF CONTENTS

Chapter 1

1 Introduction

Congratulations on choosing SpectrumSCM™ – A Total Solution to all your configuration management needs

SpectrumSCM is the most economical full-featured, unified **S**ource **C**onfiguration **M**anagement (SCM) and Change Management application available. SpectrumSCM provides version control, issue tracking , change management, process management, release management, branching, and work flow management **all integrated in one tool - no bolt-on additions!**

- Integrated support for issue/problem tracking /change management.
- Flexible support for a development process that suits the culture of your organization, or the new culture that you wish to introduce.
- Support for teams, not just individuals.
- Support for distributed developers, including taking advantage of the Internet, Intranet and the only tool to offer full CM functionality over the web.
- Management of complex projects over multiple platforms, with multiple implementation strategies.
- A common configuration management system for Multiple Platforms.
- Simple migration from existing environment.
- Simple, quick and intuitive to install and get started.

SpectrumSCM is not just a Software Configuration, Source Control or Version Control system. **SpectrumSCM** provides all these functions in the context of comprehensive product life cycle tracking of product source, whether it is *software code, requirement docs, training docs, test plans, test scripts, build scripts, models, training docs, engineering drawings, images, web pages, documents, excel, powerpoints, contracts, proposals, curriculum material, SAS program files*, etc. All source is tracked from the day it is placed under SpectrumSCM control through any and all product builds or releases. Any product release is fully reproducible at any time in the future. Multiple releases (by customer or operating system, for example) can be developed concurrently with both shared and unique components.

1.1 Why is Software Configuration Management Required?

In today's world, software is often a company's most valuable asset. Far too often the source code and documentation are left unprotected in various directories on some multi-purpose machine. This is hardly considered safe and secure considering the time, money and effort spent producing that software.

Fixing a problem in a previous release can cause mass confusion as developers scramble to find the source of the problem and the related components. As files have been changed over time,

numerous versions of each file may exist. Which files to fix? Which files to deploy to the users? Much valuable time and effort is spent looking for the right versions.

Building a new release of the product may involve developers each having a different set of files and/or making changes to the same files with no control. Much testing time is spent finding that the wrong version of a file is being tested. How does the project team reconcile multiple versions of the same file? How do they assure that all required changes have been included? Are there dependencies among the files, and how are they communicated?

Traditional configuration management systems focus on keeping track of file versions. A definite move in the right direction, but not nearly enough. Some require so much additional work that developers resist using them.

SpectrumSCM is designed to bring order from chaos, providing a complete source configuration and management system that provides major benefits without adding overhead to the development process. It is flexible enough to work within an already established process, and it can help establish a process model where one does not already exist.

1.2 The SpectrumSCM Paradigm

Key concepts in the SpectrumSCM Paradigm include:

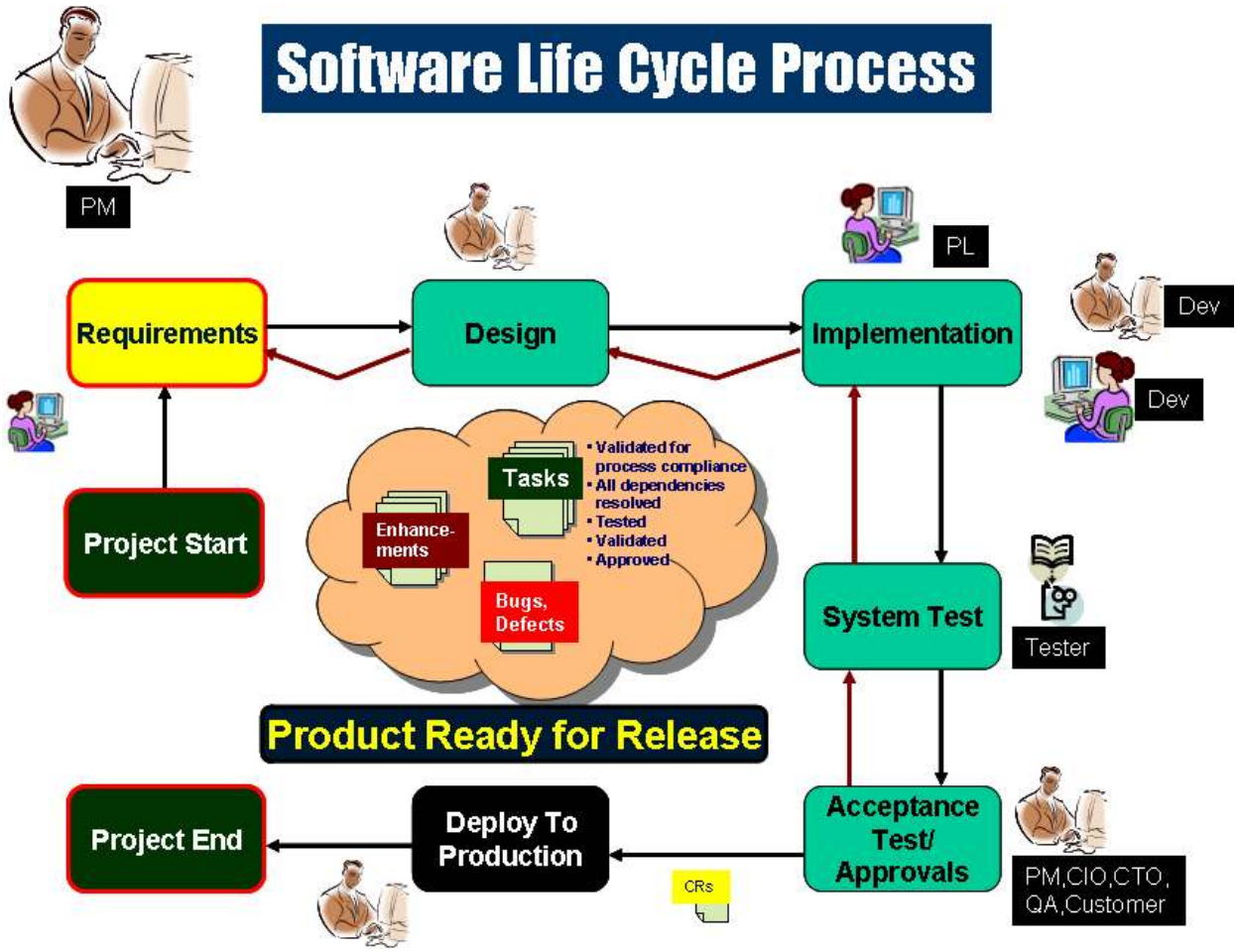
- Projects
- Branches (Generics)
- Project Life Cycle Phases
- Project Team
- Roles
- Change Requests
- File Versions
- Releases

Projects can be entire applications or systems or any set of tasks (technical tasks, business tasks, bugs/defects, infrastructure related, approvals tasks, enhancements, new features etc). A group of people working on a project is the Project Team. When a project starts to use the SpectrumSCM system, it is important that the system is set up to support the team and the desired process that the team uses or wishes to use. The project team members are added to the system and assigned roles. Each person on the project team may have such specific roles as developer, tester, project leader, etc. The project's life cycle phases have to be defined to the system. They should mirror the phases of the development process or work flow used by the team. Phases can be as simple as "study, develop, test, complete" or as specific as the phases required by many government contracts.

SpectrumSCM is a **Task/Change-based configuration management system**. SpectrumSCM uses the Change Request (CR) as the basic element of work to track the changes as you had mentioned. Files of any type are logically associated with the CR and move through the life cycle with the CR. Users no longer have to worry about file labels, version labels, release labels, etc. They no longer have to worry about overlooking a file when applying labels, error in which can

significantly impact builds and releases. Releases are composed of collections of CRs that have reached an approved state in the life cycle. CRs are simply dragged from the available pool into the associated release group in one easy step and all associated elements (i.e files) move into the release automatically. Only those CRs/Tasks that have been validated, approved, and have reached an acceptable state are allowed into a release.

The picture below shows an example of a simple software development life cycle using SpectrumSCM. In SpectrumSCM you can customize to a process that fits your business/project needs.



As work comes in, it is defined in the SpectrumSCM system by adding a new **Change Request** that defines the feature, fix or other reason why the work needs to be done. As the team members work, files are only added, changed and deleted in connection with a CR.

Traditional configuration management systems focus on keeping track of file versions. SpectrumSCM **Change Requests (CRs)** are the glue that assures that features, changes and fixes are connected to the file changes that implement them. CRs are the mechanism that allow management to track the history, activity, and status of particular issues or units of work. Using SpectrumSCM, files are managed by CRs that document why the work is being done. All changes, additions and deletions are traceable to a CR and user.

The project team may be working on various components of a large release or they may be working simultaneously on parallel versions of a release (designed for different customers or operating systems) or there may be fixes to a previous release while a new release is being developed. SpectrumSCM manages work within a project by defining generics. A **generic** is a branch of work that contains one or more features and therefore one or more files. A generic can be the first release of a system, a long-lived branch of continuing development work based on a previous release, or a short-lived branch created to fix a problem in an existing system. A generic contains files. The files can be source code, documentation, design, test plans, test results, or any type or file that must be managed. **Generics** can also be used to define/organize **sub projects** within a project. For example if different components are being built which are related to the same larger project but individually the desire is to have its own separate containers and change requests that makes its work items.

A project team can be working on multiple generics at any point in time. There are many ways a project team can define and divide work using various branching patterns. Some files may be the same as in another branch ("common"); some may be new or changed. ("uncommon") As files are changed over time, different **versions of a file** exist.

A release is a set of files, each at a particular version, that when extracted from the system make up a single version of the product. Managing release formation can be a tedious, time-consuming job on some CM systems. To create a release with the correct versions of individual files, a system needs to be able to properly track the file changes that make up each file version and present that information in some meaningful form. In the **SpectrumSCM** system this is easily accomplished with the built-in issue tracking system that ties each file change to a specific CR that describes what the changes were made. With SpectrumSCM a release is built from a set of CRs which automatically define the versions of each source file. Nothing is left to chance. CR dependency checking is done to assure release integrity.

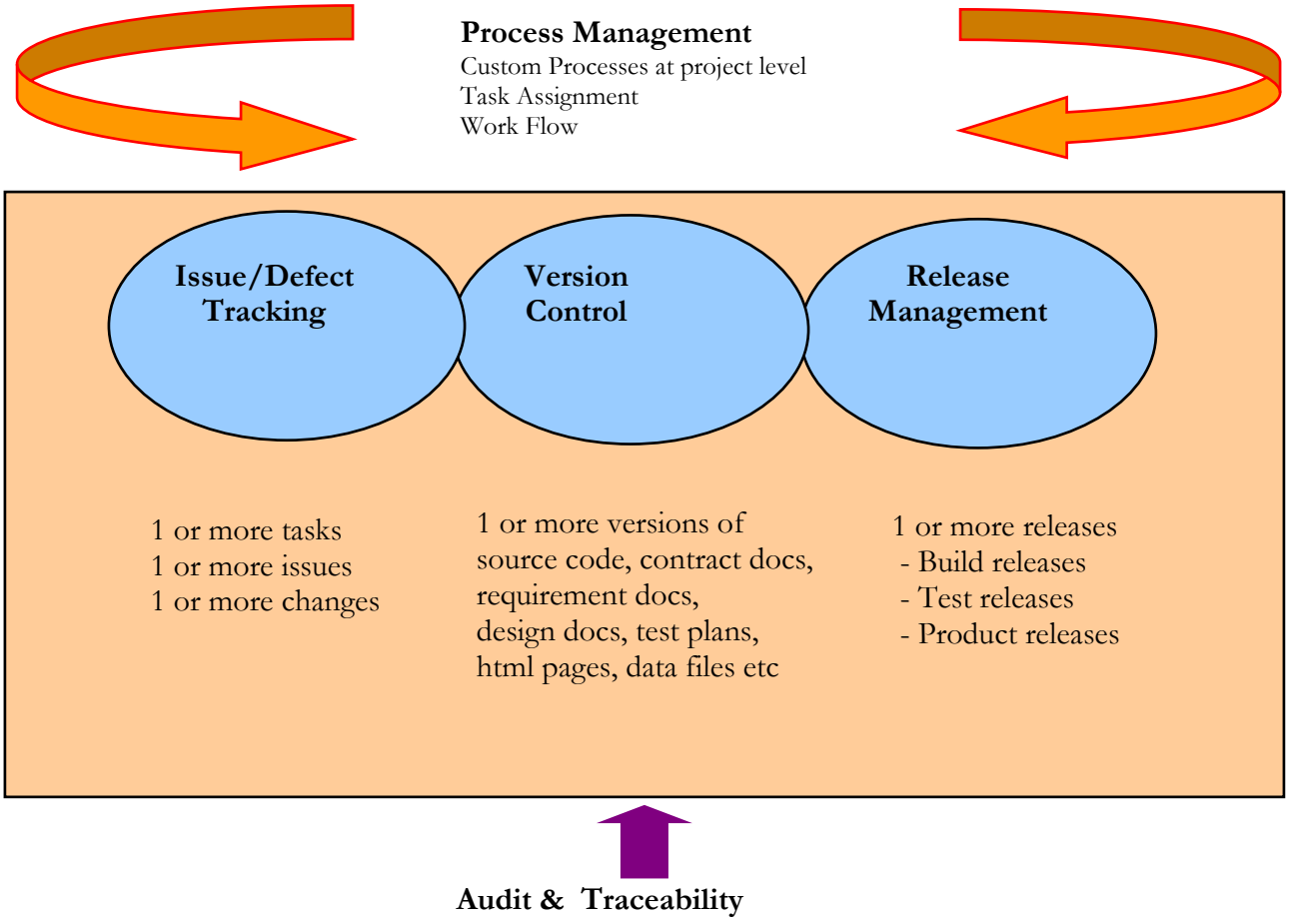
A release (a specific version of the product) can therefore be easily re-created at any time by automatically extracting the relevant versions of the files associated with the CRs in a release. SpectrumSCM was designed to make release management a simple task.

SpectrumSCM assures that the correct file versions are pulled for the release by extracting only the files associated with the CRs assigned to the release. If another CR not in the release has also changed the file, or the work on a CR has not been completed, the SpectrumSCM release management process will flag those CRs and report on the dependencies.

1.3 SpectrumSCM Process Management

SpectrumSCM supports a process that suits the culture of your organization or the new culture that you wish to introduce. Project life cycle phases and milestones are defined at the project level and can be as simple or as detailed as necessary. Government projects, for example, specify required life cycle phases that must be followed and documented. SpectrumSCM easily supports these requirements.

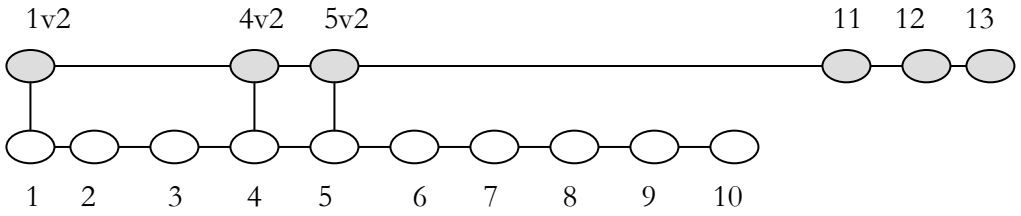
SpectrumSCM assures the quality of the product. All components are progressed through specified testing and quality assurance life-cycle phases. Some processes require a QA phase after each development step (requirements review, test plan review, test results review, etc.). Others simply require that testing be completed. SpectrumSCM allows for total customization of your process at the project level, provides the ability to verify that process was followed, and handles the storage of related process documentation associated with all phases of work All process tracking and documentation is tied to the associated change request.



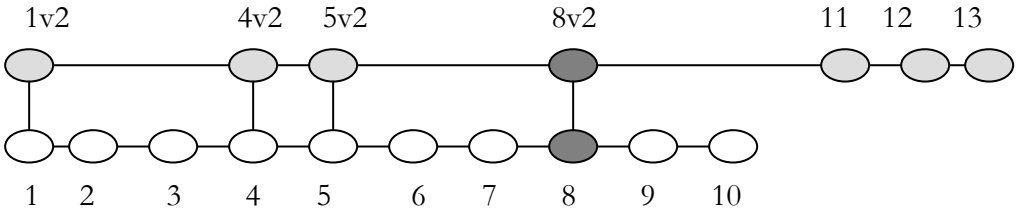
How SpectrumSCM manages Multiple Releases

For example, if a project team has developed and released version 1.0 of a system and they are currently developing version 2.0 (set up as generic 2.0), generic 2.0 will be based on the files in generic 1.0.

All files that are changed (1, 4 and 5) or added (11, 12 and 13) during the 2.0 development effort will be "uncommon" - changed only for generic 2.0. The rest of the files will be the same files as in 1.0. They are "common".



If, however, a problem is discovered in release 1.0, the fix for the problem might be made common to both generics. In this example, the problem is in file 8. The code is edited, the problem fixed and the fix is made common to both generic 1.0 and generic 2.0. A new release can be created to fix the problem (release 1.1) and deployed without disturbing the work that is going on in generic 2.0



In overview, checking a file out for edit "uncommon" will mean any file changes will only be made against that specific generic. If a check-out is performed "common" then the file changes will be made against ALL the generics that that file is currently in common with.

- **Common:** Versioned files that are physically the same across generics
- **Uncommon:** The act of physically separating versioned files from multiple generics

Checking out "common" is a powerful feature since it can be used to apply a single "fix" to multiple generics in one edit, however the developer would have to be careful of side-effects.

1.4 SpectrumSCM Technical Features

- **Powerful client-server architecture with an intuitive easy-to-use Graphical User Interface.**
  - *Works in LAN, WAN and WWW environments.*
  - *The client can even be run through your browser*
- **100% pure Java™ for complete platform independence**.
  - *Client and server can run on any platform supporting the appropriate Java Virtual Machine.*
  - *Absolutely no dependency on the underlying OS.*
- **Easy product creation and re-creation, any release, any time.**
  - *Build by Change Requests (CRs), not by file version*
  - *Build by Release (Release is a collection of CRs). It is much easier to remember a release number than it is to remember the CR number(s) that refers to a specific release.*
  - *Build by packages (a set of releases, folders or CR states) possibly from multiple projects or branches.*
  - *Reproduce, enhance or fix any release at any time.*
- **An industrial strength Object-Oriented database with full transactional integrity.**
  - *No need for database administration.*
- **Extremely scaleable and provides various levels of customizable security**
  - *Leverages the Java™ Security Model and Security Extensions with flexible Enhanced Access Control.*
  - *Projects occupy their own individual database.*
- **Source Code Extensibility**
  - *Ability to change files without disturbing previous or concurrent work.*
  - *Native file move and rename capabilities.*
  - *Workspace Analysis and Synchronization.*
- **Controlled Access to Shared Resources**
  - *Safe file sharing*
  - *All file changes traceable to user and Change Request*
  - *Versions can be compared side by side with differences high-lighted*
  - *Merge and recommon capabilities.*
  - *Access control lists (ACLs) for role based management of resource permissions*
- **Distributed Team Development**
  - *Single repository client-server model.*
  - *Remote access through Java WebStart, applet or regular client*
  - *Proxy server for bandwidth constrained environments*
  - *Work Space Analyzer and Work Space Synchronizer Utilities*

1.5 SpectrumSCM Management Features

- **Inclusion of management features as a key element of the product, not an afterthought**
  - *Full featured issue tracking and change management support*
  - *Customizable to fit the tool to your development process instead of changing your development process to fit the tool!*
  - *Pre-defined and customizable online reporting, graphs and charts*
  - *Powerful Project Performance and Metrics Dashboard*
  - *Work-flow management and tracking*
  - *Complete history of each issue and assignment*
  - *Supports meeting SEI CMM, ITIL, 21 CFR Part 1 etc. objectives.*
- **Issue Tracking / Problem Management / Change Management and Control**
  - *Ability to relate actual source changes to a living, track-able problem statement.*
  - *Parent child and peer to peer relationships between issues.*
- **Work-flow Management**
  - *Powerful Graphical workflow editor*
  - *Ability to create, assign and progress work through project life-cycle changes automatically, with real-time email notifications.*
  - *Business process rule automation*
  - *Significant cycle time reduction.*
- **Limitless versatility and total customization at the project level.**
  - *Supports the use of a process that suits the culture of your organization, or the new culture that you wish to introduce.*
  - *Enforces process without impacting development work.*

1.6 Getting Started with SpectrumSCM is Quick!

Follow the steps in the tutorial that is included on your installation CD or available for download at http://www.spectrumscm.com. The tutorial walks though the steps required to set up a project, generic, and project team and teaches the basics of using SpectrumSCM. This will enable a new user to get started in using all the basic features quickly and easily.

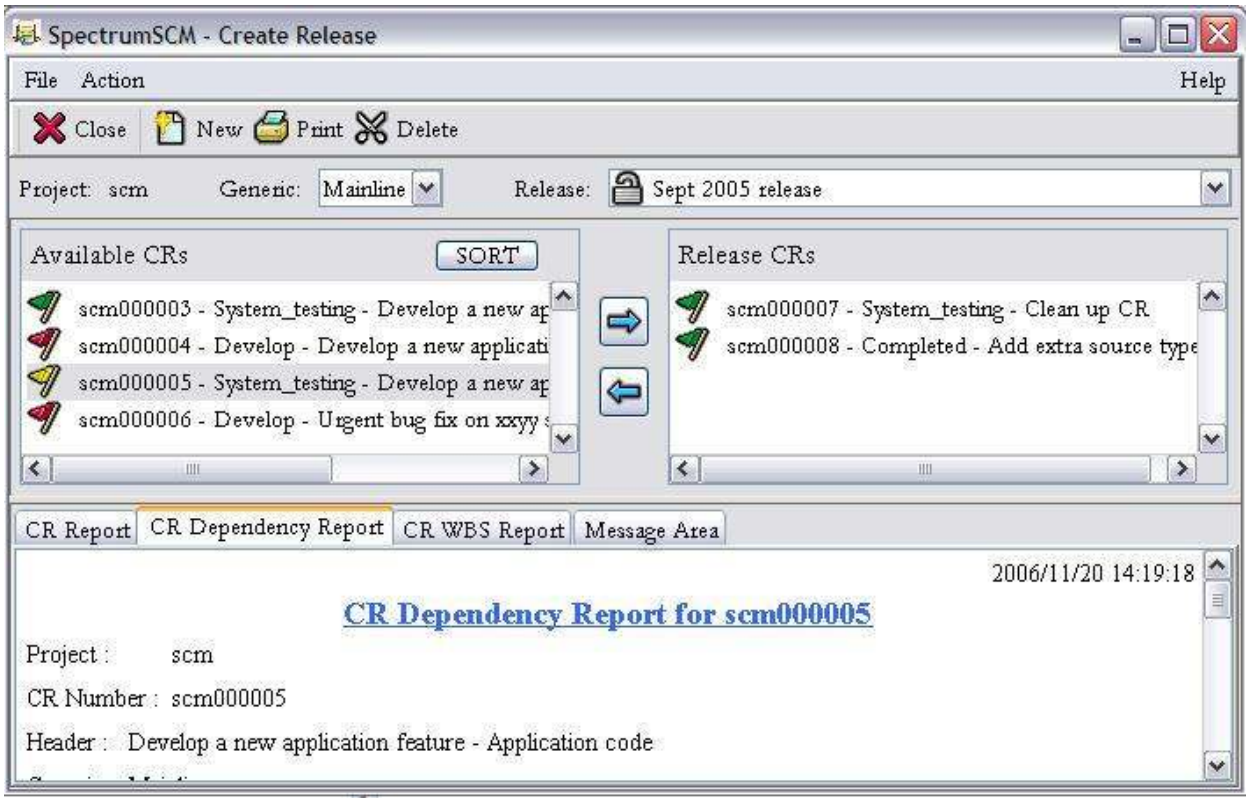
- **Installation** is automated and takes 5 - 15 minutes.
- **Create a Project**
- **Create a Generic**
- **Set up Project Life Cycle Phases**
- **Set up User roles and permissions**
- **Add project team members** as SpectrumSCM users and associated with the project.

1.7 Working with SpectrumSCM is Easy!

- **Create CR** - No work can be done without a Change Request (CR) Some projects create CRs at the feature level "develop feature 2" for example. Others create CRs at a more detailed level. A CR can also describe a problem that needs to be fixed
- **Assign CR** – A project leader will assign the CR to the first person who needs to do work on it. Depending on the size of the team and the phases in the life cycle, it might be assigned to one person to study, another to develop, and another to test. In a small team, one person might handle everything.
- **Establish local root directory** to define the location on the local hard disk where files to be checked in are found and files extracted to the hard drive are placed.
- **Load Directory Tree, Add Source, Check out files, Check in files** – the member of the project team to whom the CR is assigned might create and add files to the SpectrumSCM system or check out files, make changes, and check them back in. You can even use simple DragNDrop features to move files in and out from anywhere on your desktop.
- **Progress CR** – when a member of the project team finishes his or her work on a CR, it is "progressed", alerting the project leader that it is complete or ready to be assigned to the next person who needs to work on it.
- **Branching, Merging, and Recommoning files** as needed.
- **Create a release** – To create a release (nightly build, release for testing, or release for deployment), the Release Management feature of SpectrumSCM makes it easy to see which CRs (and their associated files) are ready.
- **Review Reports** at any point to see status of each file, CR, assignment, etc.
- **View Graphs and Charts** at any point to track, analyze, measure project, process and change request trends.

No CR goes out before its time!

SpectrumSCM assures that all components are complete and all dependencies are identified.
The Available CRs area displays the CRs that are in this generic. Note that CRs are flagged.



Only green CRs can be moved into a release.

Green – the CR's life cycle is complete. You can include this CR and its associated files in the release.

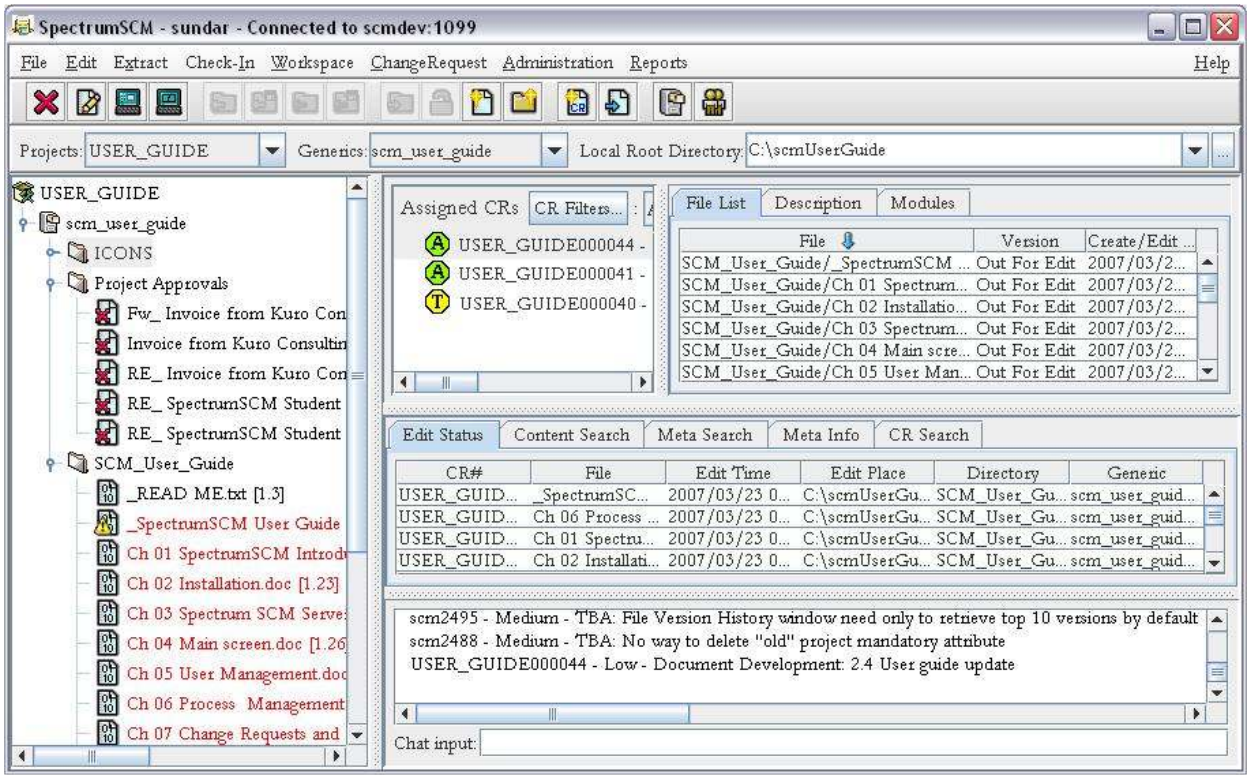
Yellow - the CR has completed development, but it is dependent on other CRs that have not yet been completed.

Red – the CR is not complete.

Dependency checking assures that all related CRs are complete before the release can be created..
The cycle of work continues until all CRs are completed and the release is ready to go!

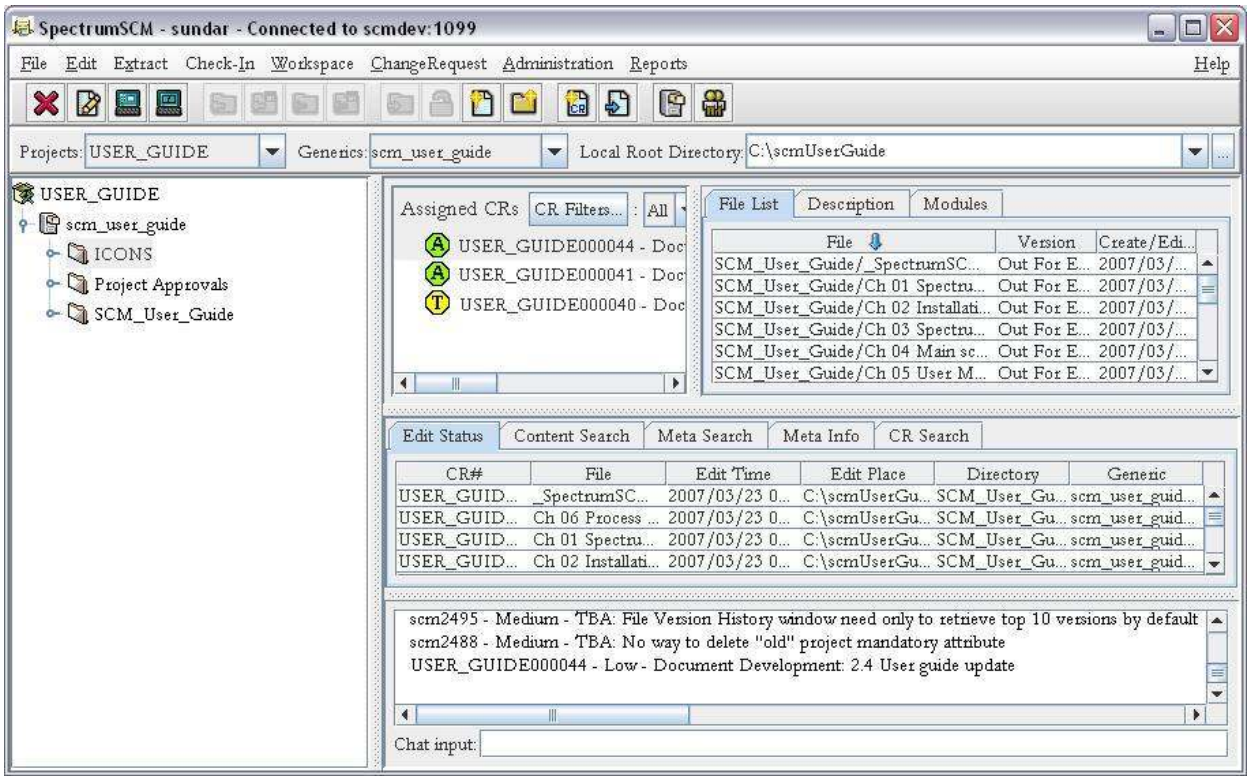
SpectrumSCM can be used to manage any type of product development, not just software. For example, it can be used to manage document creation or controlled storage of and updates to documents.

The creation of this user guide was controlled with SpectrumSCM!



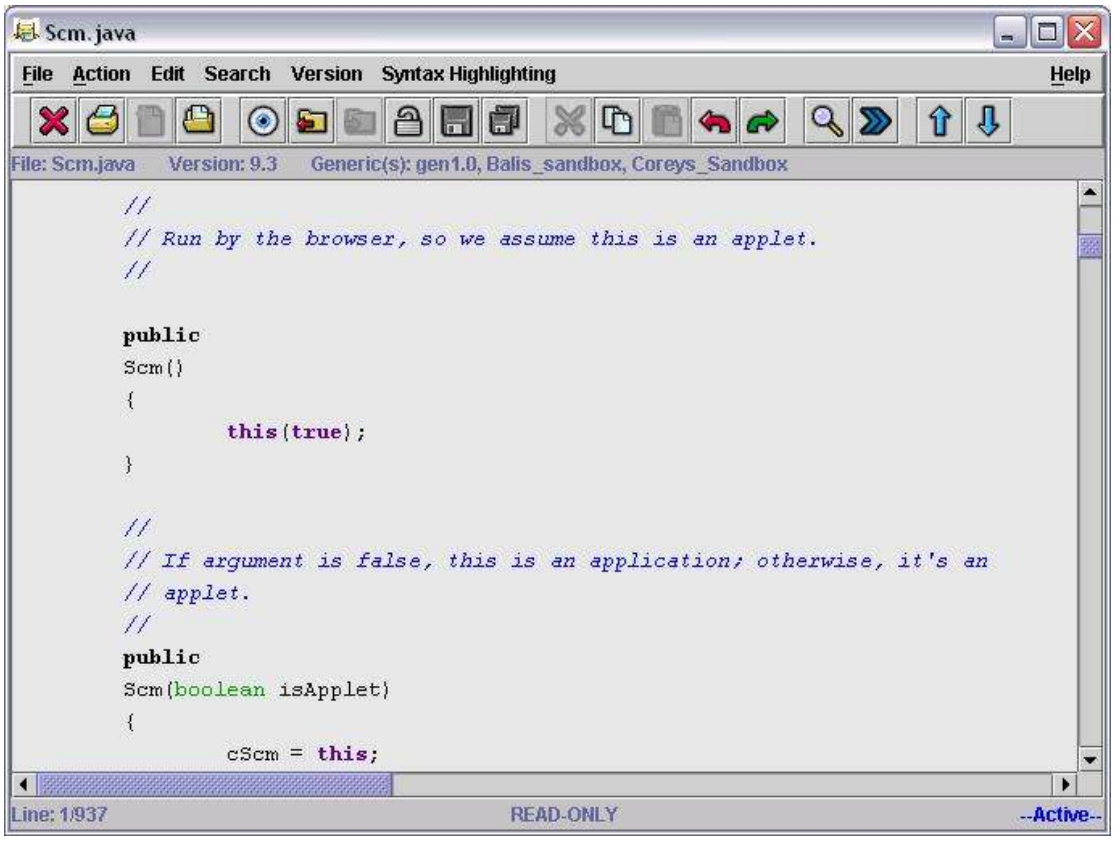
SpectrumSCM's Main Screen provides access to all functionality and a clear view to each user of his or her assignments and current work status. When each user logs on, his or her assignments are clearly visible.

- If the CR is assigned to the user for work, a green A is displayed.
- If the CR is assigned to the user for work and has supporting attachments, a green A with a "gem clip" is displayed.
- If the CR is assigned to the user for review or testing where no edits are allowed, a blue A is displayed
- If a CR has been progressed and is in a TBA (to be assigned) state, it is shown to all users in the generic engineer role and those with assignment capabilities, with a yellow T.

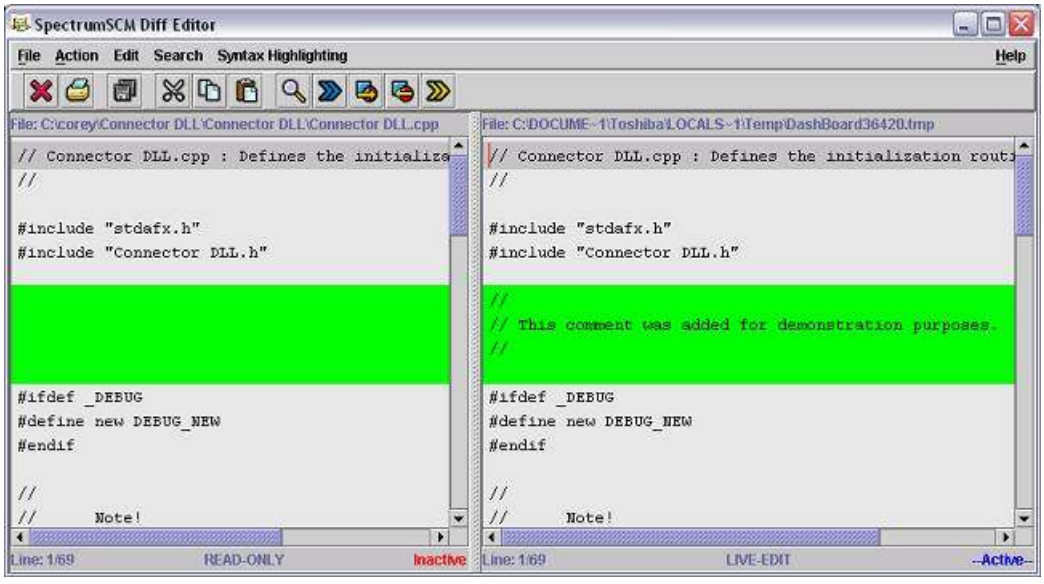


The file development process is supported and enhanced by

- A full featured graphical editor that supports syntax highlighting, the ability to walk backwards and forwards in time (file versions) and can handle enormous text documents with ease.



- A color-enhanced, side-by-side difference viewer / editor. Side-by-side difference viewer can also be used to aid merging two versions of code or text.



Even more exciting are the merging and recommon capabilities of SpectrumSCM. As members of the project team, work on different branches, multiple versions of a file are created and must be synchronized.

Merging in SpectrumSCM is the act of copying the changes made in one version of a file into another file. **Recommoning** makes the files identical; changes made in one are copied into the other and vice-versa.

Both merge and recommon use the **SpectrumSCM Merge Editor.** The Merge Editor highlights the differences between two versions of a file. Inserts show up in green, changes/differences in yellow and deletes in red. Differences can be generated and changes can be applied in either direction.

To learn about these and the many other features of SpectrumSCM, download the **SpectrumSCM Tutorial** (or access the tutorial from your installation CD) and follow the instructions step-by-step to create your first project in SpectrumSCM. Then read through the User Guide and review the examples. Many of your questions and even concerns will be answered as the power and flexibility of the SpectrumSCM system becomes apparent.

Stay current with new features and ways to use the product. The SpectrumSCM web site is constantly updated with new topics. Download and read the white papers available at www.spectrumscm.com.

1.8 Glossary

Term	Definition
SCM	*Source Configuration and Management* - Configuration Management for ALL of your product source whether it be software, web pages, requirements documents, whatever ...
Configuration Management	Process control that includes version control, audit trails, release management, and issue management working together to produce solid products that are both testable (what is in this release) and maintainable (what was in a previous release).
CR (Change Request)	An issue to be recorded and tracked. SpectrumSCM uses change requests (CRs) to drive its system. Changes made to files are tied to a specific CR, which lets developers quickly find all changes resulting from any CR. CRs allow files to be managed by issues, features, or fixes.
Branch	A separate version of a project that supports different feature sets or bug fixes from other branches within the same project.
Generic	A branch of work that contains one or more features and therefore one or more files. These files may or may not be branched (see common/uncommon files). Can be used to define sub projects or components within a project as well.
Life-cycle	A connected set of phases applied to a project change request. For example a CR might be created in the *TBA* state or assigned to *specific* state (phase). The CR might then progress through the *defined life-cycle* phases for that project before being considered complete. This set of phases is a life cycle.
Issue Management	The capability to record problems or issues and track them through the development process to their final release. This includes relating the actual source that fixes the issue to the issue itself. The capability to annotate the issue with progress notes is also significant.
Problem Tracking	See *Issue Management*
Version Control	Tracking changes to every file, and providing the capability to access any version of the specific file.

Release	A release is defined as a set of files, each of a particular version, which when extracted from the system together make up a single version of the product. SpectrumSCM ties each CR to specific versions of one or more files, so generating a release is just a matter of selecting the appropriate CRs to include. When it creates a release, SpectrumSCM displays a list of all CRs from which users can select specific (completed CRs). The user creating the release selects the CRs to include in each release.
Interim Release	A phase based informal release which extracts all the file versions at the specified workflow/life-cycle phase. This is most useful to perform automatic development/testing builds before formal release management gets involved.
Release Management	The ability to easily establish and maintain control over product releases at any time in that release's lifetime. This includes the ability to recreate it at any time and also the ability to extend it. Specifically a release is made up of a set of specific versions of the product's source files.
Package/Component Management	The ability to easily establish and maintain control over a complete product even if its components are maintained within separate projects or generics within the SpectrumSCM repository. Once defined a package can be recreated at any time. Specifically a package is made up of a set of specific releases, interim releases or folders.
Audit Trail	All changes to the product must be recorded so that control can be maintained. Who changed what, when and why?
Parallel Development	Supports development of multiple separate features from the same source base.
Uncommon File	A file that has been established in a separate generic AND has been edited (made different) under that generic (i.e. branched).
Common File	A file that shares its current version and version history with other branches.
Merging a file	The action of merging the contents of two uncommon versions of a file to create an updated version of one of those files.
Recommoning a file	The action of merging the contents of two uncommon versions of a file to create a single new version.

Chapter 2

2 Installation

In this chapter you will learn about the SpectrumSCM system requirements, how to install and uninstall SpectrumSCM components, and you will become familiar with the basic security features.

Typically, the SpectrumSCM server is installed on a shared machine (server / host) and the users install client/UI components on their machines. Users may access SpectrumSCM functionality **via six** interfaces:

- The SpectrumSCM GUI client application
- The command line interface (CLI),
- The SpectrumSCM applet (web access, browser)
- The SpectrumSCM WebStart interface (web access, no browser)
- SCMlite (CRs and reports only through the browser)
- IDE plugins (Eclipse and Microsoft SCCI)

The SpectrumSCM UI and the command line interface are "standalone" (they function independently of a Web browser). They are installed on a client machine and access the SpectrumSCM application that is hosted on the shared server.

The SpectrumSCM applet and SCMlite utilize a Web browser and require Web pages to be installed on a webserver that has access to the SpectrumSCM server.

The SpectrumSCM WebStart interface takes advantage of Sun Microsystem's WebStart functionality to download and run a SpectrumSCM client directly across the web. The advantage of using this mode is that it frees the user (or administrator) from having to install the client locally. WebStart can automatically detect when the client application has been updated and will automatically download the new application version to the users local machine, when necessary. Also, WebStart runs independent of the browser and thus a browser is not required once WebStart has been added to the client's machine.

The SpectrumSCM application and applet provide the same full graphical user interface functionality. SpectrumSCM is the only CM product that provides a web access to the tool's complete CM functionality.

The SpectrumSCM command line interface provides most SpectrumSCM functionality (and adds a few other abilities, such as starting/stopping the SpectrumSCM server) from the command line.
SCMlite provides a limited subset of SpectrumSCM functionality directly through a web-browser using HTML. Currently, only Change Request Creation and Reports can be accessed via SCMlite.

2.1 Minimum System Requirements

SpectrumSCM can be installed on any platform that supports Java™ Runtime Environment (JVM).

2.1.1 Standalone

CPU	Memory	Disk Space
Pentium (Min: 500MHz, Recommended: 800MHz)	256MB RAM	50MB + project storage space.

2.1.2 Multi-user Server

No. Of Users	CPU	Memory	Disk Space
<5 users	Pentium 800MHz	256MB RAM	50MB + project storage space.
6-20 users	Pentium 1GHz	512MB RAM	50MB + project storage space.
>20 users	Pentium (Min: 1GHz, Recommended: 2.5GHz +)	1GB RAM	50MB + project storage space.

2.1.3 Client

CPU	Memory	Disk Space
Pentium 800MHz	256MB RAM	30MB + project storage space.

2.2 Basic Installation Instructions

SpectrumSCM requires a Java Runtime Environment Version 1.4 or higher on both client and server machines. It is also recommended that all OS-specific patches be installed; check the vendor's web page for patches and updates.

Install Java 1.4 or the latest JVM. The SpectrumSCM CD contains a recent JVM bundled on the CD for MS-Windows, Unix, and Linux platforms. Alternatively, download the latest JVM from your OS/JVM provider (version 1.4 or later).

A typical enterprise installation:

- **Step 1: Install and configure the SpectrumSCM server on a shared host.** This is usually done by the server administrator, using the default id/password "scm" "scm". The SpectrumSCM server administrator should immediately change the password for this id and

retain it for any server administration that has to be performed, such as changes to the system files or security setting.

- **Step 2: Create SpectrumSCM system ids for the users who will be installing client software.** *See details on creating ids in Chapter 5, User Management.*
- **Step 3:** If users will be accessing the server via the web, **install the web pages.**

2.3 Installing the Server

Unix / Linux installation instructions

1) Insert the SpectrumSCM installation CD
2) Mount the CD (if your version does not auto mount CD drives
3) Change directory to the CD drive.
4) Run the install.sh script from the CD.

Windows installation instructions

1) Inserting the SpectrumSCM CD should trigger the auto-installer. Please wait while the installer initializes.
2) If the auto-installer does not trigger run the **install.bat** script from the CD.

The basic installation process is automated. and it is the same on all platforms.

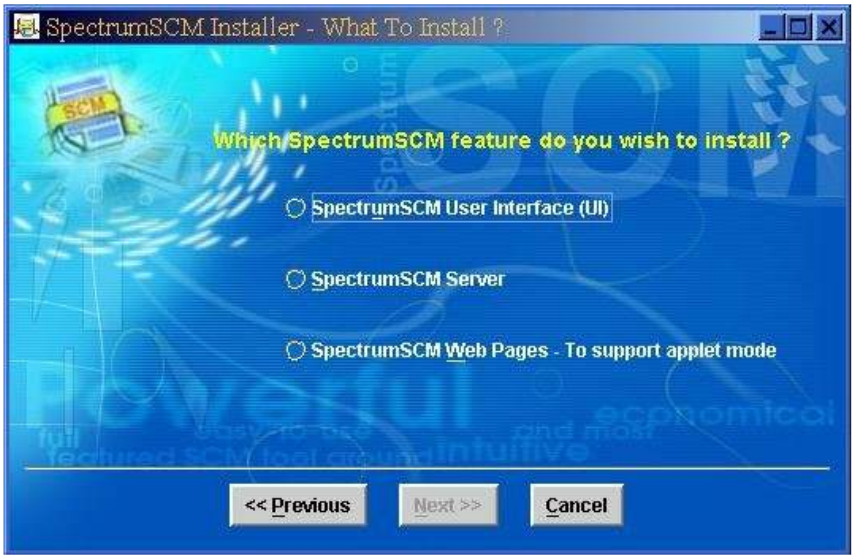
NOTE: Most examples in this user guide use MS-Windows directory structure and notation. If you are working on a Unix or Linux platform, remember that the directory structure uses forward slashes (/) instead of the back slashes (\) used in Windows.

On all systems, once the installation process has started, the **SpectrumSCM Installer** screen will be displayed. Click **Next** to proceed.

The End User License Agreement will be displayed.. Read it and click **I Accept** to continue.

2.3.1 What to Install?

Select Install **SpectrumSCM Server** – this is typically done first, by the server administrator, to install the SpectrumSCM server on a shared host. This option also installs an instance of the Graphical User Interface and command line functions onto the host machine.



Click **Next** to proceed. Clicking on the **Cancel** button will exit the installation.

2.3.2 Where to Install?

A prompt for the directory to install the SpectrumSCM tool is displayed. Type in or select a directory in which to install the SpectrumSCM components. If the directory does not exist, the system will prompt if you wish to add it. Click Next to proceed.

Browse button. You can browse the system to select a directory using the browse button

2.3.3 E-mail Configuration (simple setup without authentification)

SpectrumSCM supports full e-mail notifications of Change Request creation and transitions. In fact by setting this up, automatic email notifications happen in real time for CR/Task creations, assignments, re-assignments, progressions, workflow transitions etc. You do not have to manually send an email when you create/assign/progress an incident or CR.

These emails are sent to the person to whom it is assigned and to all stake holders (based on the roles/workflow rules) instantaneously in real time. In addition the Task/CR shows up on the individuals CR list on the main SpectrumSCM screen when they log in as well.

Most organizations that use SpectrumSCM use the email notification to facilitate workflow. The email option notifies users when CRs are assigned to them so they can immediately begin to work on them. All users set up with the **"generic engineer"** designation are also notified each time a CR changes state, is progressed, is assigned, or is in the TBA (to be assigned state) and needs to be assigned *(see Chapter 5 for details on user setup)*. To set up email notification, enter mail server contact

information (for example, smtp.mindspring.com) for notification and the e-mail "from" address. The "from" address will be the address used when mail is sent out from SpectrumSCM. If you do not wish to use the email feature, select "**Do not use e-mail notification"** (but you will miss out on a major workflow feature of the SpectrumSCM product!).

2.3.4 E-mail Configuration With Authentification

Please refer to **Sec 12.6 of Chapter 12 Administrative Functions** for more details on setting this up.

2.3.5 Standard or Custom Installation?

The custom installation option allows specification of the database directory and a custom port number. The standard installation uses the default database directory SCM_VAR inside your overall installation directory and the default port number 1099.

2.3.6 SpectrumSCM Installer

Review the Parameters given. If they are correct, click **Perform Install**.

2.3.7 Installation Successful Screen

The desired result is the Installation Successful Screen. This should complete your installation.

A list of files installed is placed into <installation directory>\scmfilelist.

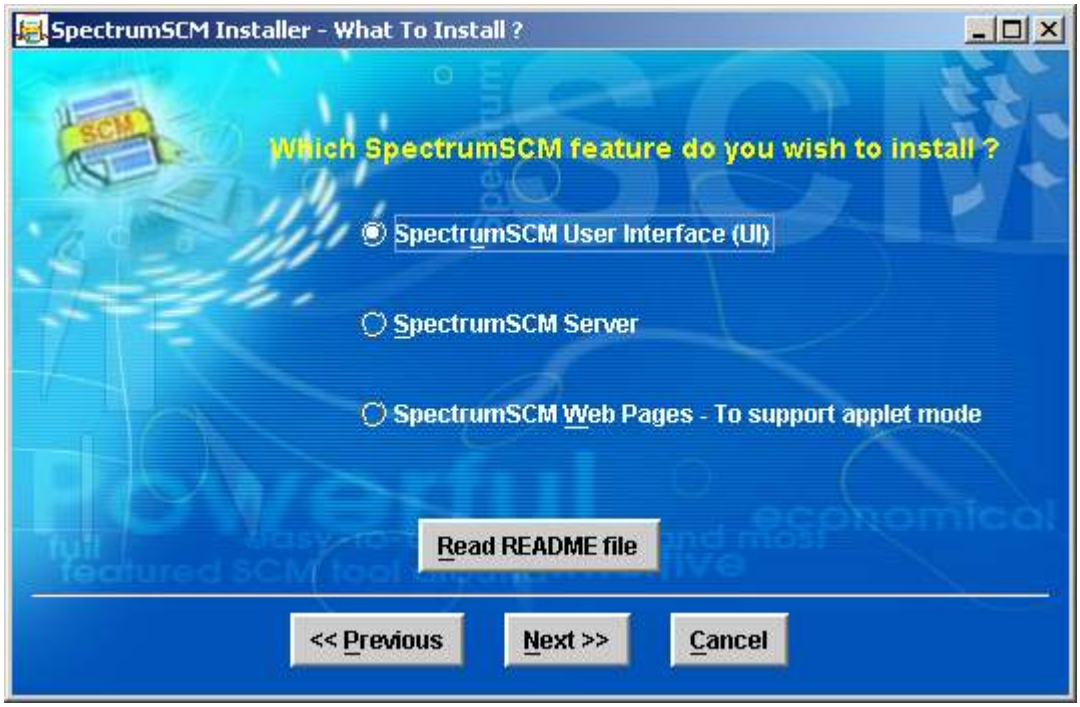
2.4 Install the SpectrumSCM User Interface (UI)

The **End User License Agreement** will be displayed. Read it and click **I Accept** to continue.

NOTE: Most examples in this user guide use MS-Windows directory structure and notation. If you are working on a Unix or Linux platform, remember that the directory structure uses forward slashes (/) instead of the back slashes (\) used in Windows.

2.4.1 What to Install?

Select Install **SpectrumSCM User Interface (UI).** This option installs an instance of the Graphical User Interface.

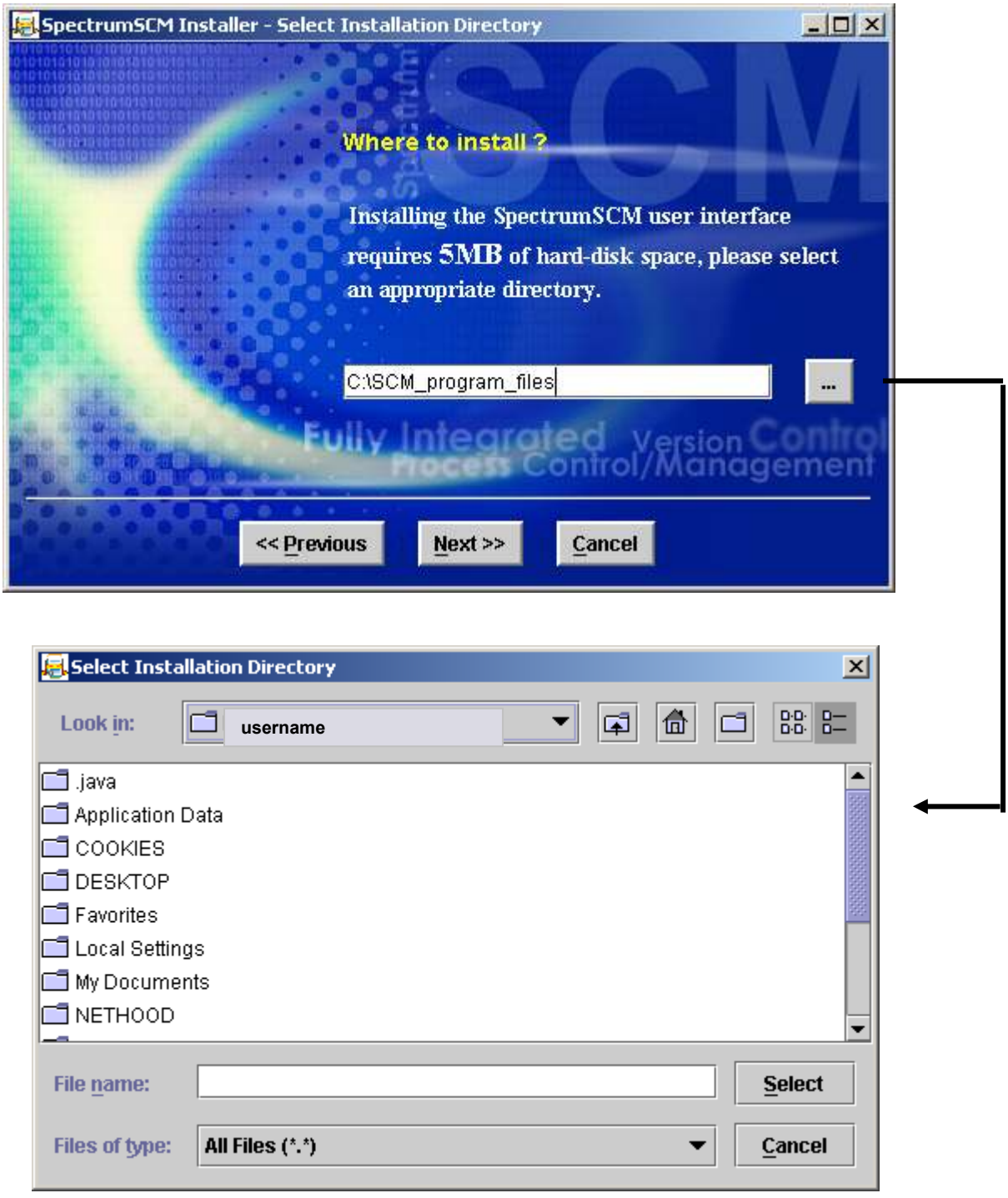


Click **next** to proceed. Clicking on the **Cancel** button will exit the installation.

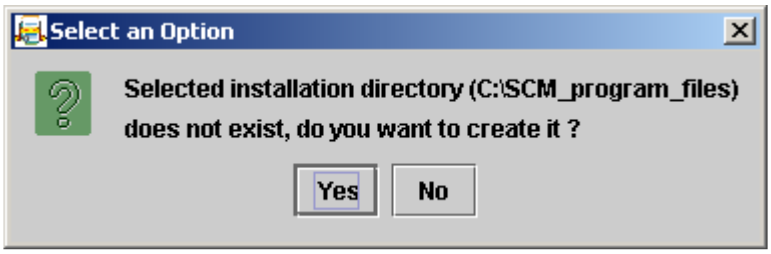
2.4.2 Where to Install

Choose an appropriate directory to install the SpectrumSCM UI components.

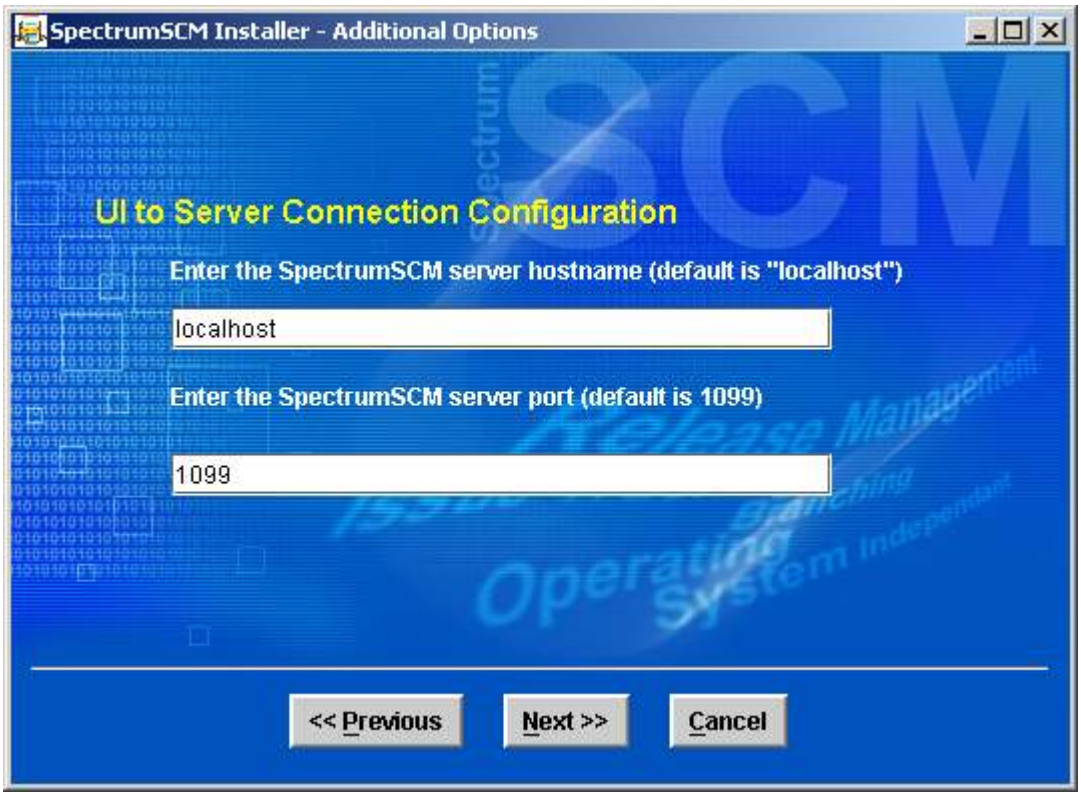
Type in a directory name or use the Browse button ... to select an existing directory. If the directory does not exist, the system will prompt if you wish to add it.



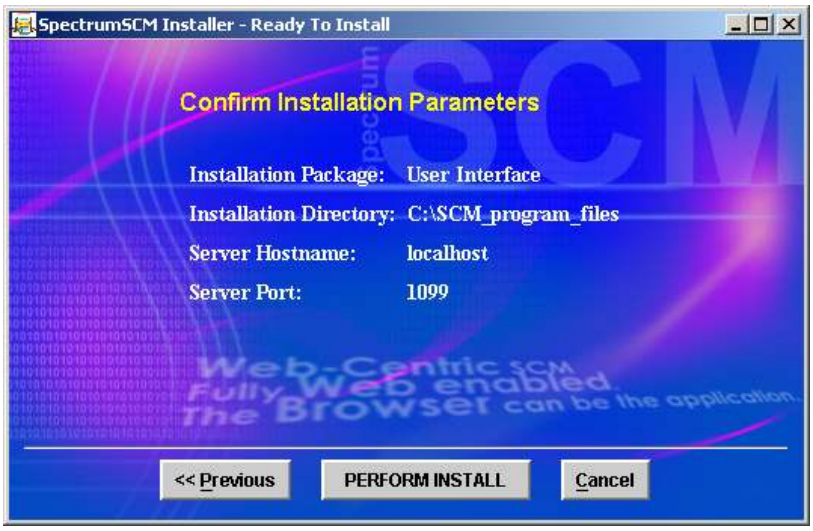
If you choose to create a new directory for the installation, you will confirm by clicking YES:



Your SpectrumSCM Server administrator or project leader will provide the appropriate server hostname or IP address and port number. You can add to or change these later using the SpectrumSCM UI Configuration Wizard *(see Chapter 3)*.



Confirm Installation Parameters. If correct, click **PERFORM INSTALL**. If incorrect, click **Previous** to make corrections.



To see a list of the files installed, see *scmfilelist* in the installation directory.

2.4.3 To start the SpectrumSCM UI

- In a Windows environment click the Start → Programs → SpectrumSCM server → **START UI** or execute **<folder where UI is installed>\bin\ startUI.bat** from the command prompt window
- In a Unix or Linux environment, **cd <directory where the UI is installed/bin>** and execute the command **startUI**

You will be asked to provide your SpectrumSCM login ID. Initial password is "default".

2.5 Install the SpectrumSCM web pages

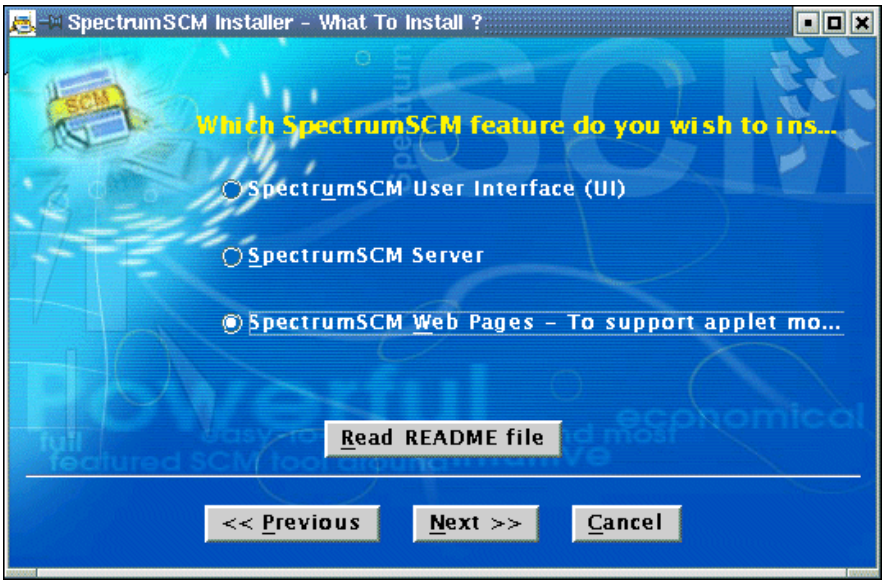
The SpectrumSCM web-pages install supports 2 modes of working, the Java™ Web-Start and the Applet modes. Both start-up methods are similar in that the user selects on the appropriate icon on a web-page. Java Web-Start however starts the actual application instance so that it runs independent from the browser, while the applet version runs through the browser itself. The Web-Start method is generally more popular because accidental closing or redirection of the browser will not close the SpectrumSCM connection, as would occur with the Applet mode.

The primary advantage a web-install has over a direct client install is that the user interface is only installed or updated once by the SpectrumSCM administrators. Users would then automatically get the (possibly updated) application without further action. Sometimes the direct client install is required, for example to use the client side commandline routines or to use the Microsoft Visual Studio (SCCI) IDE integrations.

To use the Web modes of the application, the SpectrumSCM Web Pages must be installed on a system running a web server, specifically a fully operational web server configured on the same network where the SpectrumSCM Server is located.

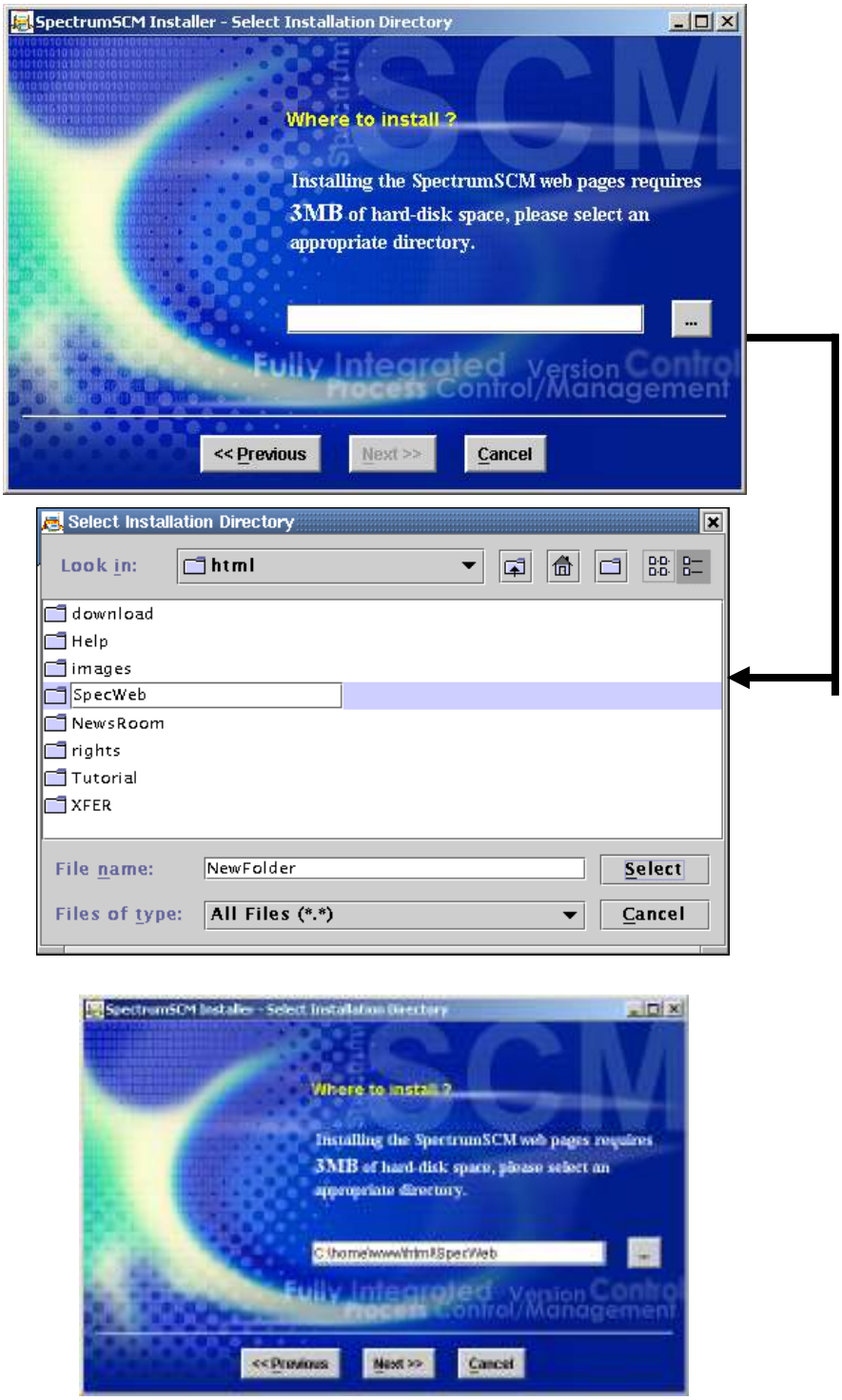
The directory in which the web pages are installed will need to be accessed by your web server. Some OS specific work might be required to set this up. For example, if using RedHat Linux with Apache, the root directory for the SpectrumSCM Web home must also be defined in the Apache config file. If using Microsoft IIS, the default web root is C:\Inetpub\wwwroot, so you could install under the C:\Inetpub\wwwroot\SpectrumSCM folder as an example.

The SpectrumSCM Web pages must be installed in the configured directory (specify it when prompted for the installation directory).



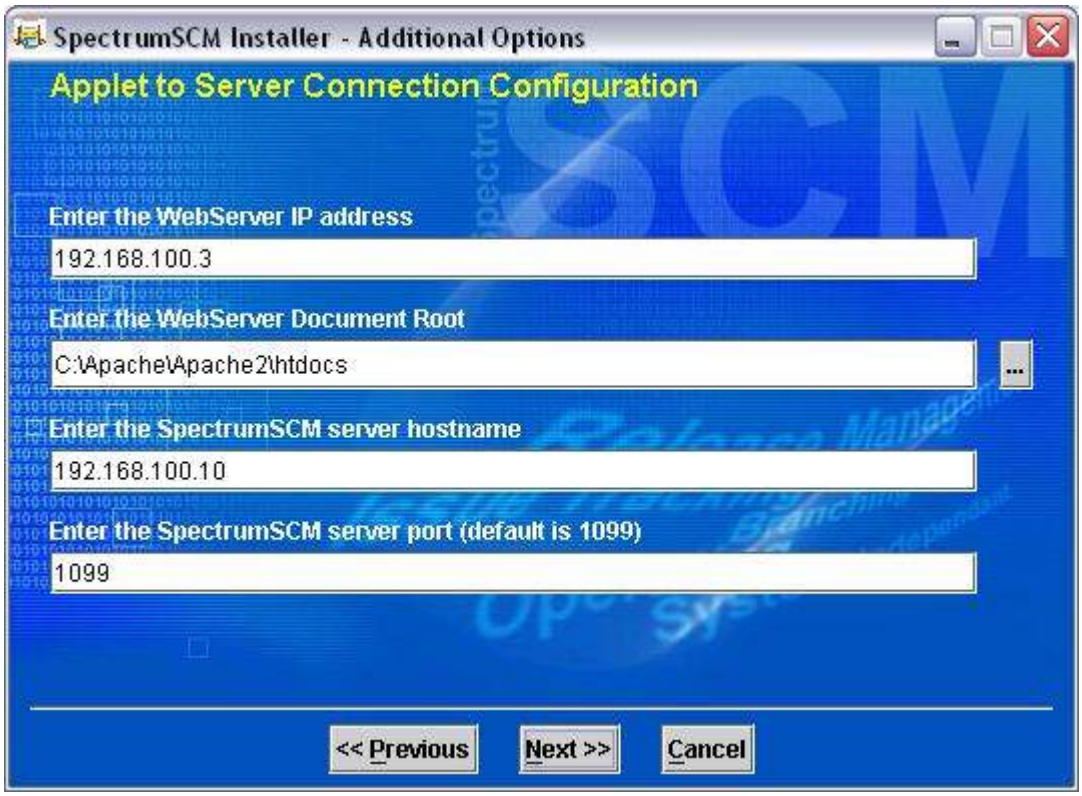
2.5.1 Where to Install

Type in the name of the root directory for the SpectrumSCM Web home directory name or use the Browse button  to select it.



Enter the Web server IP address, the WebServer document root directory, the SpectrumSCM server hostname or IP address and the server port (default is 1099). Click NEXT.

If you're not sure what your webserver's Document Root directory is, contact your web master for assistance.



Confirm installation directory, hostname, and server port and click **Perform Install**. If there are errors, click **Previous** to correct.

NOTE: Most examples in this user guide use MS-Windows directory structure and notation. If you are working on a Unix or Linux platform, remember that the directory structure uses forward slashes (/) instead of the back slashes (\) used in Windows.

A successful installation screen will be displayed., confirming the location of the web pages. Make a note of the Web Pages installation directory.

2.5.2 Applet specific items

To use the Web Applet mode of the application, additional, OS specific work might be required. This is because the applet runs as part of the browser and so is restricted by the browsers security controls.

For example, if using RedHat Linux 7.1 with Apache, the root directory for the SpectrumSCM Web home must also be defined in the Apache config file. The SpectrumSCM Web pages must be installed in that directory (specify it when prompted for the installation directory).

```
<Directory "/home/www/html/SpectrumSCM">
Options Indexes Includes FollowSymLinks
AllowOverride None
Order allow, deny
Allow from all
</Directory>
```

2.6 Accessing the web pages

The SpectrumSCM applet and WebStart interface allow access to the SpectrumSCM server via an Internet Browser. The WebStart interface only requires the use of a browser one time, to download the JNLP file. Once the JNLP (WebStart file) is installed in the local system, a browser is not required to start the SpectrumSCM client across the web.
To use the Web Applet mode of the application, the SpectrumSCM Web Pages must be installed on a system running a web server, specifically a fully operational web server configured on the same network where the SpectrumSCM Server is located.

To use the SpectrumSCM via WebStart or the applet:

Open a browser (IE or Netscape) and enter the URL to access the server that has access to the desired instance of SpectrumSCM server. Your SpectrumSCM Administrator should be able to provide the URL that is based on the directory in which the web pages have been installed.

If the web server pages are installed in directory X on the web server Y relative to the web server main page, then the URL to access SpectrumSCM would be http://www.Y/X/scmIntro.html. Note that if the SpectrumSCM pages are installed in the main page directory then X will be empty, resulting in the URL www.http://www.Y/scmIntro.html.

For example, if the webserver address is **http://www.spectrumscm.com** and the web pages are installed in a sub-directory off the main page (SCMWeb), the direct path is **http://www.spectrumscm.com/SCMWeb/scmIntro.html.**

The hostname supplied in the applet installation is the name that is used in the HTML to contact the server from the client. Therefore the client will need to be able to ping that name/address. Modifications to the **hosts/lmhosts** files might be necessary to achieve successful communications.

Due to Java applet security controls, for SpectrumSCM to access the local hard disk for operations such as check-out or read in from disk, a Java security popup will be presented. Accept this to allow SpectrumSCM to function fully. If this certificate is not granted accesses to the local disk will be forbidden and errors will occur whenever such an operation is attempted.

2.7 SCMlite

SCMlite provides a more limited subset of SCM functionality (Currently, Change Request Creation and Reports). SCMlite provides lightweight access to SCM tailored for casual users and is NOT intended to replace, only to complement, the full-featured interfaces required by developers and testers.

Why SCMlite? Some users may wish to access SCM over a low-bandwidth connection or their relationship to the project may not require access to the full SCM feature set. A customer may need access to SCM just to input CRs to file issues or request improvements; or a manager may wish to generate reports. Both need quick and easy access to limited functionality.

SCMlite is a pure HTML interface that does not require any active content or components (such as Java). There is no overhead associated with starting Java or downloading the SCM applet. In addition, there is a great reduction of training required for the casual SCM user.

SCMlite provide two SCM features:

- Change Request Creation
- Reports

SCMlite communications can be secured using SSL (https).

2.7.1 Enabling SCMlite

SCMlite is installed on the server with the WebPages. It is not turned on. To enable SCMlite, changes to the ***scm.properties file*** SCMlite section are required:

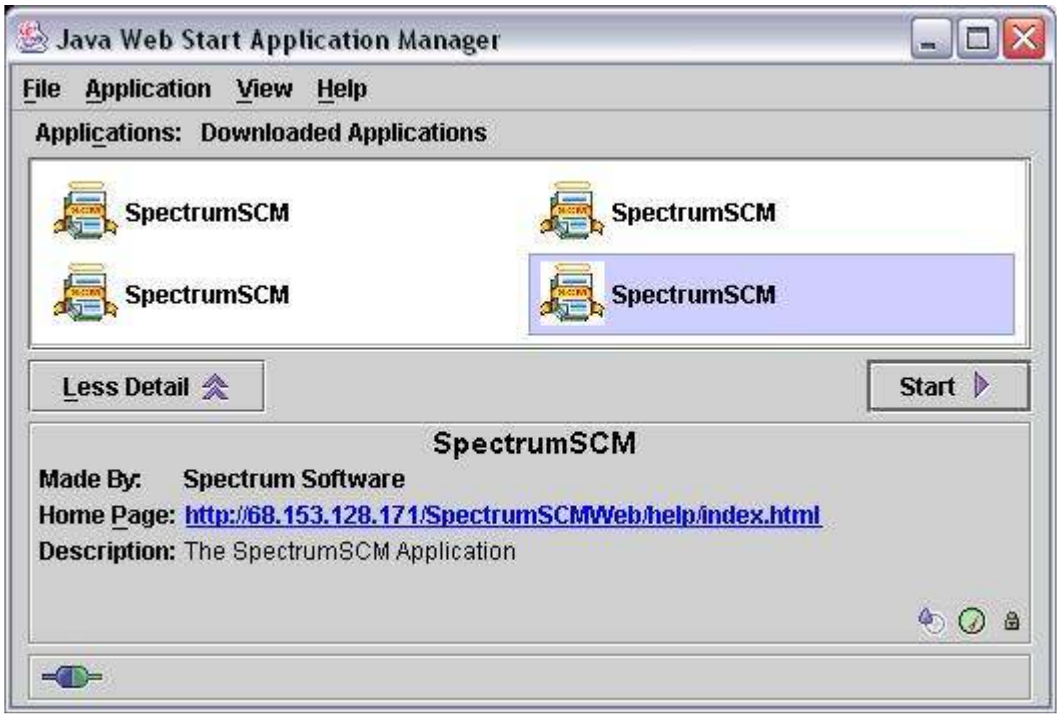
```
## SCMlite can utilize SSL to provide secure HTTP (HTTPS). Please refer
# to the second half of SSL properties, found below.   <== If SCMlite will use SSL, refer to
                                                            Communications Security above and
                                                            Chapter 3 for details on the SSL
                                                            section of the scm.properties file

## Indicates whether SCM should provide SCMlite (the HTTP interface).
## scm.http.inUse                  true    <== copy, remove #  to allow access to SCMLite

## The SCMlite interface port.
# Defaults to 1 greater than the scm.port, 1100.
## scm.http.port           12346                   <== copy, remove # and change port number if
required

## The maximum number of HTTP interface clients.
# If set to zero, any number of clients may connect.
## scm.http.maximumConnections     1       <== copy, remove # and change port number if required
#
# ^
```

(See Chapter 3 for details on making these changes directly or via the server configuration wizard)

2.7.2 Accessing WebStart, the Applet or SCMlite

To access the application via WebStart, the Applet or SCMlite, go to the same URL set up for accessing the SpectrumSCM system via the web. The web page will provide the option to logon to SpectrumSCM via WebStart, the applet or SCMlite.

If Webstart is not currently installed on your machine, use the supplied link to navigate to Sun's web pages and then download and install WebStart. Once installed, the WebStart console can be invoked by selecting WebStart from your desktop or by navigating through the Start menu options. Normally there is no need to start the WebStart console manually. The console will be automatically activated each time you access the SpectrumSCM client via WebStart. On your second attempt at starting SpectrumSCM via WebStart, the WebStart console will prompt the user to install a SpectrumSCM short cut on the desktop or in the start menu hierarchy. You can choose to do this or simply use the WebStart console to start SpectrumSCM instead of going through the browser:



In this example WebStart console, there are four (4) separate instances of SpectrumSCM installed. Each instance points to a separate instance of the SpectrumSCM server. Simply select the instance of the client to use and then press the "Start" button to activate the client. WebStart will automatically detect whether the jar files have changed and, if they have, automatically update the cached jars to the latest versions. This is an excellent solution for large group installations as the client and server software can be installed and updated in a central location. Users will automatically see the updates as they become available.

2.8 Un-install, re-install, and updates

B*efore attempting an uninstall, re-install or update, always back up the entire SpectrumSCM directory and all the project database folders.*

WARNING - A complete uninstall or re-install will remove the entire contents of the SpectrumSCM installation folder and the system and project databases. All users and projects will be deleted. Be sure this is really what you want to do. Just in case, it is wise to perform a complete backup before an uninstall/reinstall operation.

In SpectrumSCM, an install/update can be used to

- Completely re-install the same version of the product on a machine where it has previously been installed. To do a re-install, you uninstall the previously installed product, then reinstall, either into the same directories as used before OR into a different set of directories.
  NOTE: Always back up the entire SCM directory and all the project database folders in case any issues arise.
- Install an updated version of SpectrumSCM (a new release, fix, etc.) that does not involve database changes. **An update only updates the program executables and libraries, updating does NOT overwrite the SpectrumSCM system and your project databases.**
  NOTE: Always back up the entire SCM directory and all the project database folders in case any issues arise.
- When a new release of the SpectrumSCM software (for example, release 2.0) requires changes to the SpectrumSCM data base schema, the **Evolve** function is included to update the schema, leaving the data base contents intact. When this is required, detailed instructions will be included in the release's README file.
  NOTE: Back up the entire SCM directory and all the project database folders before beginning this process, just in case!

If you are having problems with an install, update or evolve, contact SpectrumSCM technical support. You may have to uninstall and then re-install if all fails. **This is why you always back up the entire SCM directory and all the project database folders before you begin such an operation.**

2.8.1 Update / Evolve the SpectrumSCM Server, UI or Web pages

Update
To make changes to the installed packages, you can use the configuration wizards described in Chapter 3 or rerun the installation process. This is useful when installing a new release or to restore the SpectrumSCM files to their original state.

- Rerun the installation from download or CD.
  - Progress thought the opening screen and licenses agreement.

- On **"What to Install"** select the component you wish to upgrade (server, UI, or web pages) and click **Next**

If the installation program detects that the component is already installed on the system, the "Verify Product Update" screen will be displayed. At this point the administrator can choose to update the component and all files associated with that component will be updated. With an update, no database changes will be made – the database will remain unaffected.

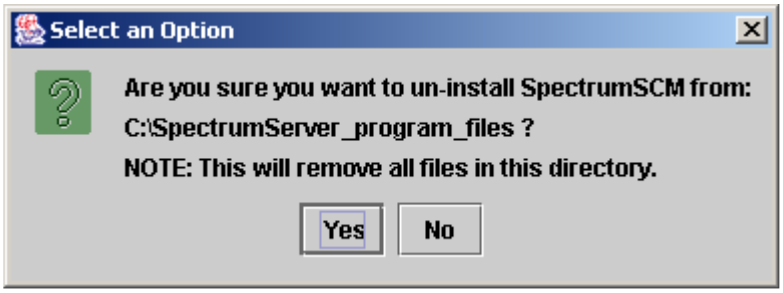
Evolve
When a new release of the SpectrumSCM software requires changes to the SpectrumSCM database schema, the **Evolve** function is used to update the schema, leaving the database contents intact. When this is required, detailed instructions will be included in the release's README file.

Uninstall the SpectrumSCM server
- In a Windows environment, from the command prompt window
  - execute the command **<directory in which the server was installed>\bin\uninstall.bat**
- In a Unix or Linux environment,
  - **cd** <directory in which the server was installed>/bin
  - execute the command **uninstall**

WARNING: ***This will delete all files associated with SpectrumSCM, including the database of project information. Before doing so, always back up the entire installation directory and all project database folders if you do not want to lose this information.***

You will be prompted to confirm that you want to uninstall the server.

WARNING: This will uninstall ALL FILES in the directory, all server and UI components from Click Yes to continue.



The uninstall is confirmed:

2.8.2 Uninstall the SpectrumSCM UI

- In a Windows environment, **In a Windows environment**:
  - START – Programs – SpectrumSCM UI – Uninstall

  or
  - from the command prompt window execute the command <directory in which the UI was installed>\bin**uninstallUI.bat**
- In a Unix or Linux environment,
  - **cd** <directory in which the UI was installed>/bin and
  - execute the command **uninstallUI**

  or
  - <UI install directory>/bin**/uninstall**

You will be prompted to confirm that you want to uninstall the UI. Click **YES** to continue. Confirmation will be displayed:

NOTE: Uninstalling the UI from a client machine will NOT impact the server or the database.

2.8.3 Uninstall the SpectrumSCM Web Pages / applet

- Delete the directory into which the web pages were installed.

or

- Rerun the installation from CD or download.
  - Progress through the opening screen and licenses agreement.
  - On **"What to Install"** select the SpectrumSCM Web Pages and click **Next**
  - If the installation program detects that web pages are already installed on the system, the "Verify Product Update" screen will be displayed. At his point the administrator can choose to uninstall the web pages.

2.9 SpectrumSCM Security Features

SpectrumSCM leverages the Java™ Security Model and Security Extensions to offer a variety of security features to protect the server, clients, and communications. These features fall into two groups, access control and communications security.

2.9.1 Access Control

SpectrumSCM employs a traditional account-based access model. The SpectrumSCM administrator role is responsible for creating user accounts. These accounts are protected by a login/password pair that must be provided when a user logs into the application. This is the default application security model, additional Access Control is configured through the server configuration wizard (or by editing the file **<directory in which the server was installed>**/SCM_VAR//etc/security/accessControl).

2.9.2 Location Control

SpectrumSCM provides an additional level of security based on the user's workstation's hostname (or IP address if unavailable). A user can be restricted to logging into the application from specific hosts. This feature is called **Location Control**; to enable it a "doAccess" line must appear in the accessControl file followed by "access" lines specifying allowed user/workstation combinations:

```
doAccess
        access  user1   workstation1
        access  user2   workstation2
etc.
```

With location control turned on only those users specified can log into the SpectrumSCM system. Even valid users attempting to login from workstations other than those specified in the accessControl file will be denied, *even if they supply a valid password*

2.9.3 Unauthenticated Commandline Access (single sign-on)

SpectrumSCM provides a UNIX-style command line interface for accessing many of its features, however as mentioned above the default security scheme would require a login and password for each server access. This may prove to be unnecessarily tedious when typing in a sequence of commands or incompatible for some activities (automated checking out of files by a nightly build script). This situation also comes up in the general world with people accessing many different IT systems and applications and has become known as **Single Sign-On**.

SpectrumSCM provides a feature called **Unauthenticated Commandline Access** to *relax* security, to allow unauthenticated command line access by specific users at specific workstations. Administrators should exercise caution when configuring this feature, and use it sparingly, if at all, considering all security ramifications.

Basically what single **"sign-on/unauthenticated access"** does is move the access control responsibility to the operating system/work-station level. Once a user has successfully logged in to the operatingsystem/work-station, that login information is what is then used to access the individual applications such as SpectrumSCM.

To enable this feature a "doUnauth" line must appear in the accessControl file followed by "unauth" lines specifying allowed user/workstation combinations:

```
doUnauth
    unauth user1 workstation1
    unauth user2 workstation2
etc.
```

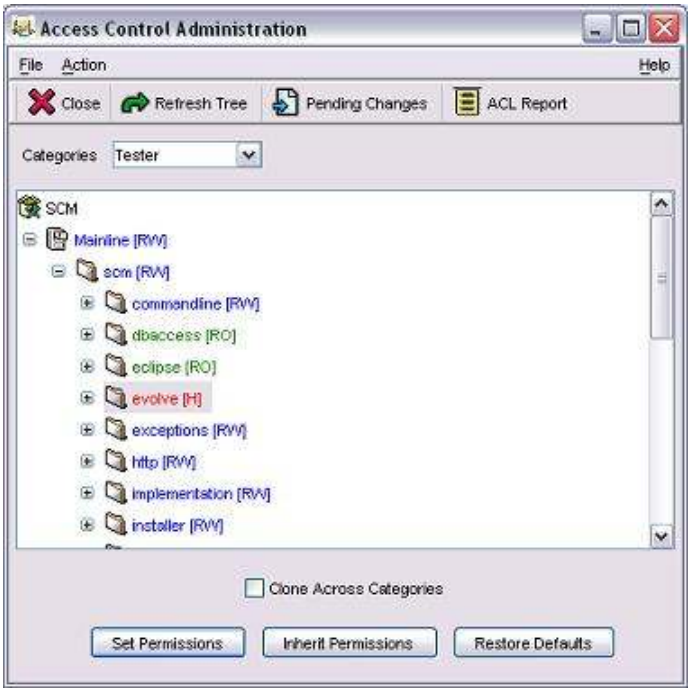
Users logging in to the UI or executing command line functions from their corresponding workstations as configured in the accessControl file will not be required to provide a password since that verification has already been performed by the operating system.

Administrators should exercise caution when configuring this feature, since if access to the workstation is not tightly controlled (ie access cards, screen locks etc) inappropriate accesses might occur.

NOTE/CAUTION !: Whereas Location Control tightens security, Unauthenticated Commandline Access could be viewed as relaxing security. In addition, Location Control takes precedence over Unauthenticated Commandline Access.

2.9.4 Access Control Lists

Access control in SpectrumSCM can be applied at the branch, directory and file levels. Controls are established for particular categories of users, instead of at the user level directly. This means that controls can be immediately established for all users assigned to a particular role. For instance, users assigned to a Development or Testing role may not have write access to the documentation directories or access to the requirements and design documents themselves.



In this example, users assigned to the Tester role have read-only access to the dbaccess and eclipse directories, and no access at all to the evolve directory. There are three modes that describe the level of access to repository resources:

- Read-Write [RW]
- Read-Only [RO]
- Hide [H]

When assigning Access Controls to a particular role, an already established set of controls can be used as a template for the new role. To use an already defined ACL template, select the "Inherit Permissions" button at the bottom of the screen and select an existing role to use as a template. Default access can be restored for any selected category by pressing the "Restore Defaults" button. All mode changes remain unimplemented until the "Set Permissions" button is pressed. Changes in ACL settings are highlighted with an "*" next to each repository element that has been changed.

2.9.5 Communications Security

SpectrumSCM provides a means to protect its communications so that assets may be checked out, modified, and checked in securely. By default, SpectrumSCM operates in an open mode, which is generally acceptable for use on a corporate intranet. This mode is the most efficient and appropriate for a majority of installations.

However, it may be necessary to use SpectrumSCM across an uncontrolled, insecure network (such as the Internet). In this situation, SpectrumSCM should be configured to protect its communications, which is accomplished via SSL (Secure Socket Layer, a standard developed by Netscape and approved by the Internet Engineering Task Force as a standard), or SSH (Secure Shell). Secured communication is not accomplished without a price - overall responses will be slower due to additional encryption/decryption processing at both ends of the connection. In addition, the administrator must obtain an SSL key and configure SpectrumSCM accordingly for SSL.

Once the administrator has obtained an SSL key, the configuration file must be edited to use it, this is done through the server configuration wizard or by directly editing *SERVER INSTALL DIRECTORY/SCM_VAR*/etc/scm.properties. The last section of the file pertains to SSL; scm.ssl.inUse should be set "true" and the other properties, particularly the SSL keystore (where the SSL key is stored) and the password that protects it must be provided.

Once the server has been properly configured, the SpectrumSCM clients may connect by supplying the SSL option. Turning on the SSL option is performed through the UI Configuration Wizard or by supplying **-ssl** on the command line. *See Chapter 3 for details.*

For secured communications using SSH, the server must be configured properly and the client side tunnel must be activated. To setup SSH on the server, modify the scm.properties file and set the following attributes as follows:

```
java.rmi.server.hostname            localhost
java.rmi.server.userLocalHostname   true
scm.port                            1099
scm.rmi.portnumber                  XXXX (where XXXX is some unused port)
scm.transport.portnumber            1101 (or any other unused port)
```

These entries tell the server to respond to client side connection requests using the machine name "localhost" and they also configure the port numbers to be used for the RMI channel, transport layer and registry port numbers.

On the client side, the SSH tunnel must be established via the command line and then the GUI will be able to connect to the server. The SSH command should look like the following:

```
ssh –N –v –L1099:server-ip:1099 –LXXXX:server-ip:XXXX –L1101:server-ip:1101
–l login server-ip
```

where "server-ip" is equal to the IP address of the host and XXXX is the port number assigned to the RMI channel described above.

3 SpectrumSCM Server and UI Configuration

Chapter 3

In this, chapter you will learn when and how to use the SpectrumSCM Server Configuration Wizard and the SpectrumSCM UI Configuration Wizard, and you will become familiar with various parameters in the scm.properties file.

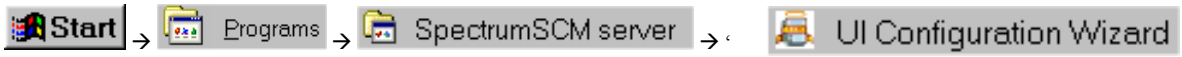
- UI Configuration Wizard
- Server Configuration Wizard
- scm.properties file

3.1 UI Configuration Wizard

The UI configuration Wizard is used if you are accessing your SpectrumSCM system via a client in a client-server mode through a LAN or WAN. The UI Configuration Wizard is designed to aid in configuring the SpectrumSCM User Interface client. Specifically, it manages the list of servers to which your organization allows access and on which SpectrumSCM server has been installed.

To start the SCM UI configuration wizard

In a windows environment

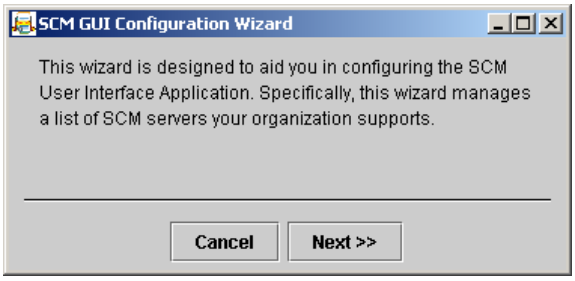


or

In a UNIX or LINUX environment the UI configuration wizard is started from the command line, nohup'ed, assuming you are running an X-server on the UNIX/Linux system.

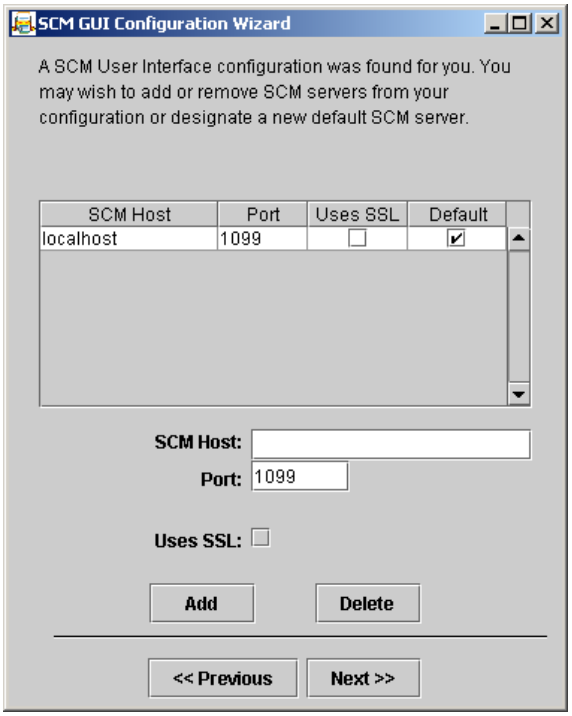
- change directory to the SpectrumSCM **install bin directory**
- execute the command ***uiConfWiz***

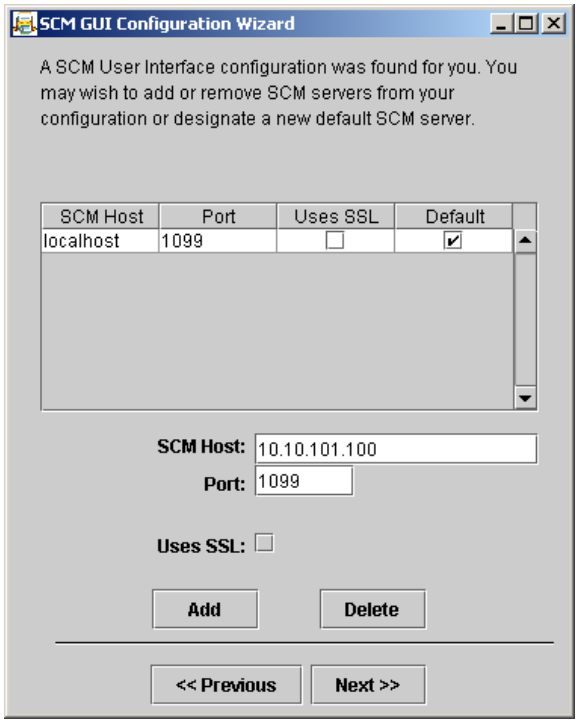
The UI Configuration Wizard will walk you through all the steps to access a SpectrumSCM Server. The server(s) do not need to reside on the same network as the local workstation; the GUI simply needs remote access to the network for the Server(s).



The UI Configuration Wizard displays the configured servers available for use by the Spectrum UI client. If a SpectrumSCM server has been installed on the same machine, it will appear as "local host".

In order to use the UI with a SpectrumSCM server via a LAN or WAN, SpectrumSCM needs to know the address for the server. You can enter either the IP address or the domain name assigned to the server. Contact your SCM administrator to get the host name / IP address and port number that is to be used.

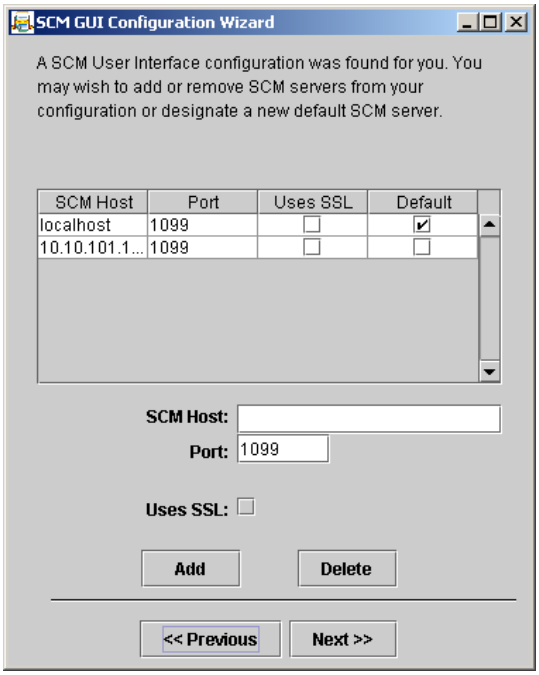




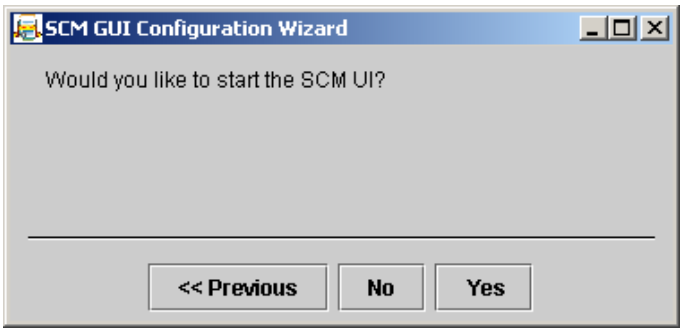
Click the **Add** button to save the new SCM host settings.

Review the settings. If an entry is incorrect, or if you need to remove a server from your list, highlight that server and click **Delete** to remove it from your list.

NOTE: If you have multiple servers listed, select a default server. This is the server that the SpectrumSCM UI will connect to when it is started. If you need to work on another server, you have to run the UI Configuration Wizard again, reset the default to the desired server and restart the UI client.



Verify that all settings are correct and click **Next.**
Now you may start the UI and connect to server by clicking the **Yes** button. If multiple servers are listed, you will be connected to the default server.



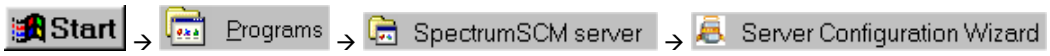
3.2 Server Configuration Wizard

The server is installed with default properties. It can be configured using the Server Configuration Wizard or by editing, the scm.properties file directly.

NOTES:

- **Only users with administrator authority can start, stop or reconfigure the server.**
- **Stop the server before changing its configuration and restart when done.**

To start the **Server Configuration Wizard**
In a WINDOWS environment click



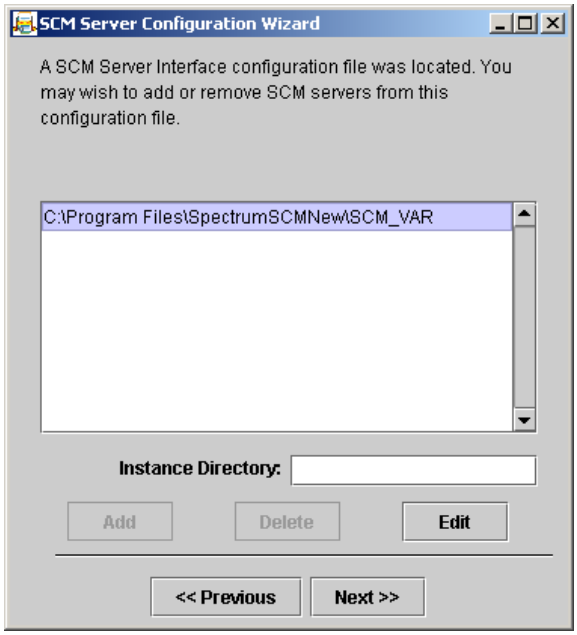
In a UNIX or LINUX environment the server configuration wizard is started from the command line, nohup'ed, assuming you are running an X-server on the UNIX/Linux system:

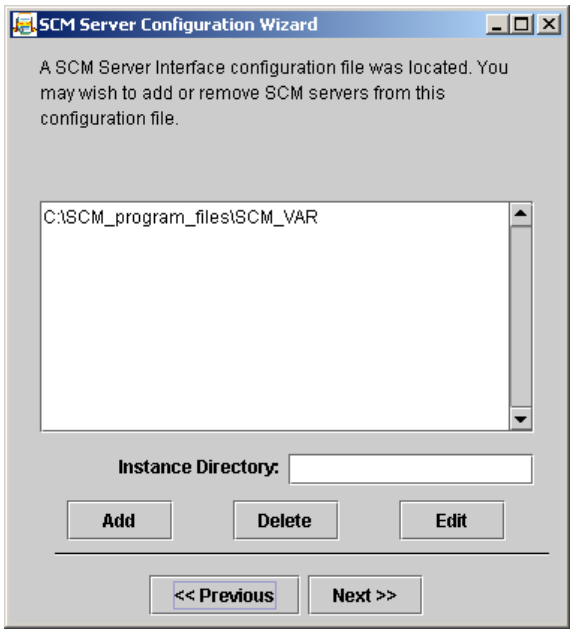
- Change directory to the SpectrumSCM install bin directory
- Execute the command **svrConfWiz**

The SCM Server Configuration Wizard will walk you through all the steps to set up or modify a SpectrumSCM Server accessible through your local network.

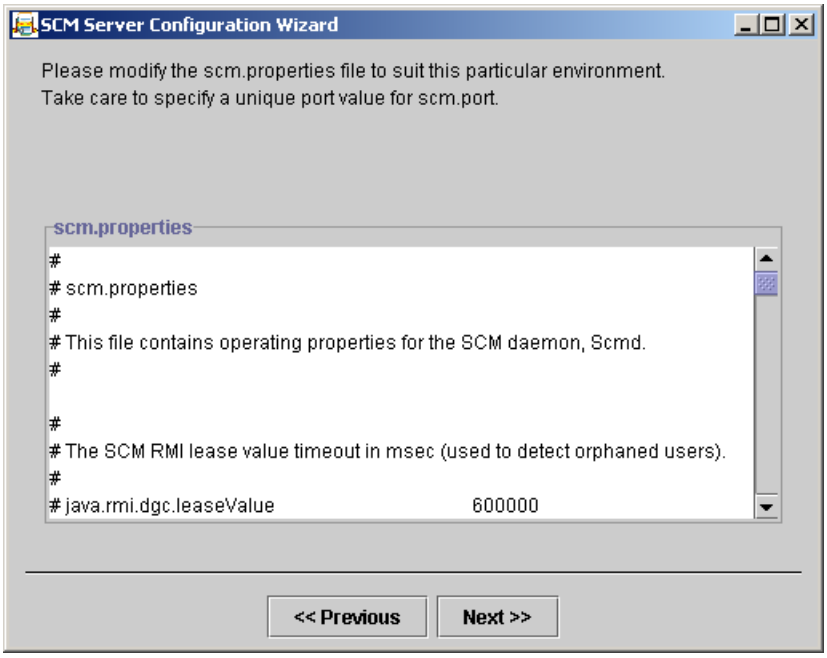
Click **NEXT** to proceed.

On this screen, you can select an instance of the SCM Server and edit a server's properties.





To edit an existing server's properties, select the instance of the server whose properties you wish to edit. Click **EDIT** to proceed. The selected server's **scm.properties** file is loaded into an edit screen.



Any content of the *scm.properties* file may be changed using the Wizard.

For example, to change the server's scm port, locate the following:

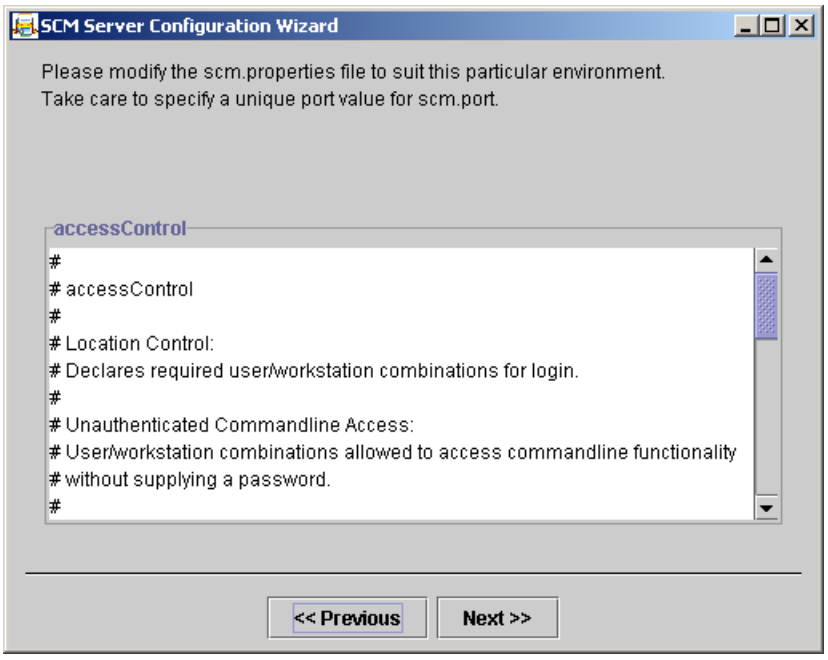
```
#
# The SCM network port.
# Defaults to 1099.
#
# scm.port                      12345
```

Remove the comment indicator # before SCM port and change 12345 to the desired port number.

```
#
# The SCM network port.
# Defaults to 1099.
#
scm.port                        1080
```

After making desired change(s), click **NEXT.** The access control window will appear, presenting the various security / access control options that can be modified.

See Chapter 2 for details about Access Control and Security features.



Click NEXT. A prompt will appear asking whether to start the server.

Click YES to restart the SpectrumSCM Server.

3.3 Start Server and UI

To use the SpectrumSCM UI, the server must be up and the SpectrumSCM UI client on your machine must be started.

To check to see if the server is up:

In a WINDOWS environment

Start-> programs -> SpectrumSCM UI ->CheckServer

In a UNIX or LINUX environment

- change directory to the SpectrumSCM install bin directory
- execute the command **checkServer**

If the Server is running, you can start the SpectrumSCM UI

In a WINDOWS environment

Start-> programs -> SpectrumSCM UI ->StartUI.

In a UNIX or LINUX environment

The UI can be started from the command line, nohup'ed, assuming you are running an X-server on the UNIX/Linux system. It can also be started from an xterm window.

- change directory to the SpectrumSCM install bin directory
- execute the command **startUI**

3.4 The scm.properties file

The scm.properties file contains operating properties for the SCM daemon, Scmd. It can be edited through the Server Configuration Wizard. Alternatively, since the properties are stored in the file system under <install directory>\SCM_VAR\etc. The **scm.properties** file can be edited directly with your favorite text editor.

The SpectrumSCM system is designed to run with preset defaults unless they are changed by modifications to the scm.properties file. The file itself is well documented with descriptions of the parameters that can be changed as well as those that are optional and can be enabled (for example, SCMlite and SSL).

When changing a value or enabling an option, be sure to remove the comment indicator (#) from the beginning of the line. It is recommended that you copy the commented line and edit this line to set the appropriate value or option. This way, in case you need to undo and redo the entry, you can always edit this line again.

For example, commented, the interface is not enabled.

```
#
# Indicates whether SCM should provide SCMlite (the HTTP interface).
#
#  scm.http.inUse                                true
```

Removing the comment indicator enables the SCMlite http interface.

```
#
# Indicates whether SCM should provide SCMlite (the HTTP interface).
#
scm.http.inUse                               true
```

The scm.properties file:

```
#
# scm.properties
#
# This file contains operating properties for the SCM daemon, Scmd.
#

#
# The SCM RMI lease value timeout in msec (used to detect orphaned users).
#
# java.rmi.dgc.leaseValue     600000


#
# Properties to control the behaviour of the SCCI interface.
#
# constrainGetToIDEProj applies to the Visual Studio OpenFrom functionality
# and specifically constrains the 'get' call to return only those projects
# or solutions that match the open IDE project name. If this is not set (the
# default) all projects/solutions in the SpectrumSCM project are returned.
# scm.scci.constrainGetToIDEProj    true


#
# The SCM runtime Directory
#
# scm.root  /scm/runtime


#
# The SCM network port.
# Defaults to 1099.
#
# scm.port  1099


#
#
# What character set should the system use as its default.
# Note that while human beings treat character sets as "isn't it obvious",
# unfortunately in the world of bits and bytes (computers) it is not.
# Specifically the issue becomes more complicated when sharing assets across
# multiple computing platforms/operating systems, which might have a
different
# interpretation of those bytes. The prime example of this is Microsoft
Windows,
# which in the default (Western European) settings uses a character set of
# Cp1252 or windows-1252. This is not an international standard but rather a
# variable proprietary extension of the ISO-8859-1 standard.
#
# Using non-standard characters in a platform portable environment will
possibly
# cause those characters to change, or even be deemed invalid on the second
```

```
# computing platform. In SpectrumSCM, invalid characters are sometimes seen
as
# character sized empty vertical rectangles.
#
# Note: From SpectrumSCM's perspective, "computing platforms" include all
# clients (particularly those doing file edits), and the server (particularly
# if the keyword expansion feature is enabled (see below)).
#
# WARNING: Changing this value on a system that already has assets stored
should
# be performed with great care. Changing the character-set can change the way
# that the system interprets the textual asset files. Contact SpectrumSCM
# if you have any questions.
#
# scm.charset     iso-8859-1

#
# The frequency (in hours) at which the reservations cleanup thread wakes
# up (Defaults to 24 hours)
#
# scm.reservations.cleanup_interval 24


# | SESSIONS SESSIONS SESSIONS SESSIONS SESSIONS SESSIONS
# V
# The concurrent_sessions attribute controls the number of concurrent read
# transactions that can be running against a "project" at any given time.
#
# Adjusting the value to a higher number will allow more read transactions to
# execute simulataneously but will also allow the server to consume more
# memory.
#
# Setting the value to a lower number will constrain the number of concurrent
# read transaction but will reduce the amount of server memory utilized.
#
# This attribute does not control the number of users that can be logged in
# at the same time.
#

concurrent_sessions      5

#
# The max_session_reuse parameter can be tuned in the cases where an
# application has a large amount of data that is accessed infrequently, or is
# more than can be held comfortably in memory. When a database session is
# completed, its database references are held via weak-references for future
# use. Setting the max_session_reuse parameter will dispose of that session
# after the specified number of times, thus allowing the garbage collector to
# free up the associated memory.

# max_session_reuse     200

# ^
# | SESSIONS SESSIONS SESSIONS SESSIONS SESSIONS SESSIONS
```

```
# | COMPRESSION COMPRESSION COMPRESSION COMPRESSION COMPRESSION
# V
# The compress_threshold indicates that high water mark for binary files
# where GZIP compression will be used to reduce the size of the file
# before transmission to the server
#

compress_threshold      10000

# ^
# | COMPRESSION COMPRESSION COMPRESSION COMPRESSION COMPRESSION


# | KEYWORD EXPANSION KEYWORD EXPANSION KEYWORD EXPANSION KEYWORD EXPANSION
# V
#
# SpectrumSCM supports the following keywords:
#
#     Keyword           Expands To
#    -------------    ----------------------------------------------------
#     $Author$    Authors login id
#     $Date$            Date of last modification
#     $Revision$  Version number of this file
#     $Source$    File name
#     $Path$            Directory path
#     $CR$        Change Request used for this modification
#     $State$           Change Request state at time of change
#     $Project$   Project name
#     $Generic$   List of generics that this file is common with
#     $Header$    Group containing $Path$ $Source$ $Revision$ $Date$ $Author$
$CR$ $State$
#     $Id$        Same as header less $Path$
#
# SpectrumSCM keywords are based on the RCS model. This means that keywords,
once
# expanded, can always be expanded again. For instance, the $Author$ keyword
will
# expand to the following: $Author: joe $
# The expanded version of the keyword allows for additional expansion during
future
# edits.
#
# Keywords must be enabled for them to be expanded. Also, keywords are
expanded
# during check-in and add-source activities only. Because the SpectrumSCM
server
# is character set agnostic, enabling keywords, which requires parsing text
files
# as ASCII text instead of binary data, requires that a character set
encoding
# be used. If the scm.charset option is specified above, that will be used.
Otherwise
# the platform default character set (file.encoding) will be used.
#
# The scm.keyword.expansion.maxdepth variable limits the number of lines
# searched in any given file to those found above this value. Adjusting
```

```
# the value to a lower number improves overall search times.
#
# Uncomment the following two lines to turn on keyword expansion.
# The maximum search depth for keyword searching defaults to 50 lines.
#
# scm.keyword.expansion.enable            true
# scm.keyword.expansion.maxdepth    50
#
# Note: keyword expansion will have a slight impact on check-in performance.
#       the closer the keywords are located towards the top of your source
#     files, the faster the server can locate and expand them.
#
# ^
# | KEYWORD EXPANSION KEYWORD EXPANSION KEYWORD EXPANSION KEYWORD EXPANSION


# | PASSWORD CONTROL PASSWORD CONTROL PASSWORD CONTROL PASSWORD CONTROL
# V
#
# The scm password expiry period (in number of days). This value is used
# when new users are added to the system. The expiry date is renewed
# whenever a user changes his password. Defaults to NEVER
#
# The minimum_length parameter controls the minimum length of the password
# entered by the user in the Change Password Screen. By default there is
# no minimum password length enforced.
#
# scm.passwd.expiry_period    NEVER
# scm.passwd.minimum_length   8
# scm.passwd.requires_mixed_case    true
# scm.passwd.requires_numbers true
# scm.passwd.requires_symbols true
#
# ^
# | PASSWORD CONTROL PASSWORD CONTROL PASSWORD CONTROL PASSWORD CONTROL


# | AUTO-LOGIN AUTO-LOGIN AUTO-LOGIN AUTO-LOGIN AUTO-LOGIN
# V
#
# SpectrumSCM autologins a user if:
#     1. The user name on the local machine (user.name property) matches the
#        ID of a valid SpectrumSCM user
#
#     2. The user has been granted unauth priveleges in the access control
#        file
#
# The disable_autologin parameter is used to disable auto login. Default is
YES
#
disable_autologin YES
#
# ^
# | AUTO-LOGIN AUTO-LOGIN AUTO-LOGIN AUTO-LOGIN AUTO-LOGIN
```

```
#
# | Mail Mail Mail Mail Mail Mail Mail
# V
#
# Set this next line to the SMTP mail host for your
# organization. ex: smtp.mycompany.com
#
# mail.smtp.host  mailhost
#
# Authentication ----------------------------------------
# If your smtp host requires authentication then
# you'll want to enable the following attributes:
#
# mail.smtp.auth  true
# scm.smtp.auth.login   SCM_LOGIN_ID
#
# The SpectrumSCM server will use the login and password associated
# with the supplied login as the SMTP server login and password combo.
#
# Authentication ----------------------------------------
#
# The SCM mail "from" address. This is the address that all the
# mail will appear to have come from. Make sure to use a valid
# e-mail address here.
#
# scm.from  scmAdmin@mycompany.com
#
# To control the format of the mail message sent out by the system
# Meta words are supported -
# $SHORT$       The default message 'Change Request $CR_id$ $Action$'
# $CR_id$       The Change Request Id
# $Action$      What action is causing this e-mail (create, progress, assign)
# $Assignee$    Who is the CR being assigned too ('N/A' for TBA, Completed or
Killed)
# $State$       What is the new state
# $Prev_State$  What was the previous state
# $Abstract$    The abstract/header from the CR
#
# Full example:
# scm.mail.subject_line_format      Change Request $CR_id$ was $Action$,
assignee $Assignee$, from $Prev_State$ to $State$: $Abstract$
# scm.mail.subject_line_format      $SHORT$
#
# If you would like the e-mail notifications sent as HTML
# you can enable the next line. Otherwise it will be sent as
# plain text
#
# scm.mail.content.type HTML
#
# If you want the e-mail notications to include all of the
# Change Requests previous state transitions and note, enable
# the next line
#
# scm.mail.content.allnotes   TRUE
#
# By default people assigned the Generic Engineer role will
```

```
# receive e-mails whenever a CR is re-assigned, unless otherwise
# specified by workflow rules. If you want them to only receive
# e-mails for To Be Assigned tasks then uncomment the following line
#
# scm.mail.sendAssignmentsToGenericEngineers    FALSE
#
# ^
# | Mail Mail Mail Mail Mail Mail Mail


#
# | Tutorial Tutorial Tutorial Tutorial Tutorial
# v
#
# The SCM Tutorial is accessed across the web by default.
# If access to the public network is not available, the
# tutorial path can be redirected to a local resource
# like in the following example for Windows:
#
# scm.tutorial    file:/C:/SCM_INSTALL_DIR/help/Tutorial/scmstart.htm
#
# or like this for Unix platforms:
#
# scm.tutorial    file:/SCM_INSTALL_DIR/help/Tutorial/scmstart.htm
#
# You do not have to use reverse slashes on the Windows platform.
# The tutorial materials are available on the SpectrumSCM installation
CD_ROM.
# Copy the entire Tutorial directory off of the CD_ROM to some location
# on a local or shared network drive, like in the examples above.
#

scm.tutorial      http://www.spectrumscm.com/Tutorial/scmstart.htm

# ^
# | Tutorial Tutorial Tutorial Tutorial Tutorial
#


#
# | SCMlite SCMlite SCMlite SCMlite SCMlite SCMlite SCMlite
# v
#
# The following three properties control the availability
# of the SCM HTTP interface, SCMlite, which provides access to
# a subset of SCM functionality:
#     * creation of Change Requests
#     * report generation
#
# SCMlite can utilize SSL to provide secure HTTP (HTTPS). Please refer
# to the second half of SSL properties, found below.
#

#
# Indicates whether SCM should provide SCMlite (the HTTP interface).
#
```

```
# scm.http.inUse  true

#
# The SCMlite interface port.
# Defaults to 1 greater than the scm.port, 1100.
#
# scm.http.port   12346

#
# The maximum number of HTTP interface clients.
# Leave commented out to allow any number of clients may connect.
#
# scm.http.maximumConnections 1

#
# ^
# | SCMlite SCMlite SCMlite SCMlite SCMlite SCMlite SCMlite
#



#
# | FIREWALL FIREWALL FIREWALL FIREWALL FIREWALL FIREWALL
# v
#
# The following four properties may be utilized to help SCM interoperate
# with firewalls, particularly those of the transport-level bridge variety.
# If either of these properties are used, be sure internal clients can
# route to the IP address or fully qualified hostname.
#
# As an example, in order to setup SpectrumSCM to operate through a NAT
# enabled firewall such as a LinkSys BEFSR41, the following steps will
# need to be implemented.
# 1. Install the server on a machine inside the firewall
# 2. Set the java.rmi.server.hostname to the name of the machine
# 3. Set the java.rmi.server.useLocalHostname to true
# 4. Set the scm.rmi.portnumber to an unused port number.
# 5. Set the scm.transport.portnumber to an unused port number.
# 6. Open the port numbers from steps 4 and 5 in the firewall and
#    port forward those ports to the server machine
# 7. Open port number 1099 on the firewall and port forward that port
#    to the server machine.
# 8. On client machines, edit the IP hosts table (hosts file) and
#    set the servers name equal to the IP address of the firewall
#    that will be used to do the port forwarding.
#    example: server(spongebob) = 192.168.100.1  firewall = 68.134.128.169
#             set spongebob = 68.134.128.169 on all clients that need
#          to use the SpectrumSCM server.

#
# Specifies the IP address SCM should encode into its interfaces.
# In most cases, this would be the firewall's external IP address.
#
# java.rmi.server.hostname    localhost

#
# Directs SCM to encode the server's fully qualified hostname into
```

```
# its interfaces. In most cases, this would be the firewall's externally
# known hostname.
#
# java.rmi.server.useLocalHostname  true

#
# This next argument directs the RMI framework to use a particular
# port number for all RMI socket level communications. Leave the
# value of the arugment set to (0) to instruct RMI to use any valid
# random port number.
#
# scm.rmi.portnumber    0

#
# This next argument directs the transport layer to use a particular
# port number. The transport layer will default to port number 1101
# unless directed to use another port number.
#
# scm.transport.portnumber    1101

#
# ^
# | FIREWALL FIREWALL FIREWALL FIREWALL FIREWALL FIREWALL
#


#
# | SSL SSL SSL SSL SSL SSL SSL SSL SSL SSL SSL SSL SSL
# v
#
# SpectrumSCM supports SSL for secured communications between the server and
# clients. SSL in Java requires a Provider that supports the SSL protocols.
# If you plan to use SSL with SCM or SCMlite, you must uncomment the line
# below or supply your own.
#

#
# The SSL provider.
#
# scm.ssl.provider      com.sun.net.ssl.internal.ssl.Provider

#
# SSL also utilizes a certificate signed by an entity all parties
# trust. This certificate is installed in the appropriate directory and
# the following eight properties are required to utilize it.
#

#
# Indicates whether SCM should use SSL.
# If set to true, be sure to supply the following seven properties.
#
# scm.ssl.inUse   true

#
# The protocol SSL should use.
#
```

```
# scm.ssl.protocol      TLS

#
# The algorithm used to encode the keystore.
#
# scm.ssl.keymanager.algorithm      SunX509

#
# The implementation of the keystore.
#
# scm.ssl.keystore.type JKS

#
# Name of the SSL keystore under etc/security/ssl.
#
# scm.ssl.keystore.filename   sslKeys

#
# Password of the SSL keystore.
#
# scm.ssl.keystore.password   passphrase

#
# Alias of the key to use for SSL.
#
# scm.ssl.key.alias     scmkey

#
# Password of the key.
#
# scm.ssl.key.password  keypass

#
# Indicates whether SCMlite should use SSL.
#
# SCMlite uses a separate certificate (usually in a separate keystore)
# installed in the appropriate directory. The following eight properties
# are required to utilize it.
#

#
# Indicates whether SCMlite should use SSL.
# If set to true, be sure to supply the following six properties.
#
# scm.http.ssl.inUse    true

#
# The protocol SSL should use.
#
# scm.http.ssl.protocol TLS

#
# The algorithm used to encode the keystore.
#
# scm.http.ssl.keymanager.algorithm SunX509
```

```
#
# The implementation of the keystore.
#
# scm.http.ssl.keystore.type  JKS

#
# Name of the SSL keystore under etc/security/ssl.
#
# scm.http.ssl.keystore.filename    httpsKeys

#
# Password of the SSL keystore.
#
# scm.http.ssl.keystore.password    passphrase

#
# Alias of the key to use for SSL.
#
# scm.http.ssl.key.alias      httpskey

#
# Password of the key.
#
# scm.http.ssl.key.password   keypass

#
# ^
# | SSL SSL SSL SSL SSL SSL SSL SSL SSL SSL SSL SSL SSL
#


#
# | LDAP LDAP LDAP LDAP LDAP LDAP LDAP LDAP LDAP LDAP LDAP LDAP LDAP
# v

# Indicates whether SCM should use LDAP Authentication, that is whether
# SpectrumSCM verifies its login & password information with the LDAP
# server before allowing a user into SpectrumSCM.
#
# LDAP.useAuth    YES


# Indicates whether SCM should use LDAP Import Facility i.e. Is SpectrumSCM
# allowed to import user data from the LDAP server into the SpectrumSCM
# database. The information imported is the display name, phone number.
# e-mail address and/or location attributes as specified below.
#
# LDAP.useImport  YES


# Indicates whether SCM should use SSL support for LDAP
#
# LDAP.useSSL     YES


# The LDAP server's address
```

```
#
# LDAP.server     192.168.100.51


# The LDAP network port (defaults to 389)
#
# LDAP.port DEFAULT


# Determines how aliases are dereferenced when the LDAP server is queried.
# Possible values are:
# always    - Always dereference aliases (default)
# never     - Never dereferences aliases
# finding   - Dereferences aliases only during name resolution
# searching - Dereferences aliases only after name resolution
#
# LDAP.dereferenceAlias always


# The distinguished name (DN) used for LDAP authentication
# $UU$ is a placeholder for the User ID (uid) used during the
# authentication process. The place holder is replaced with the
# login string entered by the user when he/she attempts to login
# to SCM. If the 'useNameAsUU' option is set, then the users name
# as recorded in the SpectrumSCM database is used as UU instead of
# the user id.
#
# Also note: multiple DNs can be specified, LDAP.dn would be the
# primary, LDAP.dn2 the secondary, LDAP.dn3 the tertiary, etc
#
# LDAP.useNameAsUU      true
# LDAP.dn   uid=$UU$,dc=SpectrumSoftware,dc=net
# LDAP.dn2  uid=$UU$,ou=Development,dc=SpectrumSoftware,dc=net


# The base for LDAP search operations, i.e. the top most point of
# query into the LDAP database.
#
# LDAP.searchbase dc=SpectrumSoftware,dc=net


# The LDAP attribute used for mapping the LDAP login name to the
# Spectrum SCM user ID. The value of this attribute in the LDAP
# directory MUST match the user ID used in Spectrum SCM.
# Upon successful authentication, this value of this attribute
# will be retrieved and used as the SCM User ID.
#
# LDAP.uid.mapping      uid
#
# Additional mapping parameters useful for the import facility
#
# LDAP.name.mapping     cn
#
# LDAP.phone.mapping    telephonenumber
#
# LDAP.mail.mapping     mail
```

```
#
# The location attribute is by default a general information text
# field. If, however your organization has many disparate organizational
# units within the LDAP database this can be in-efficient to search.
# Instead, SpectrumSCM can use the location attribute/field to cache the
# users specific distinguished name (DN).
#
# LDAP.useLocationForDN true
# LDAP.location.mapping distinguishedName


# ^
# | LDAP LDAP LDAP LDAP LDAP LDAP LDAP LDAP LDAP LDAP LDAP LDAP LDAP
#


# The following parameter controls whether the old linear life-cycle
# screen is still enabled.
scm.lifecycles.linear_enabled true
```

Chapter

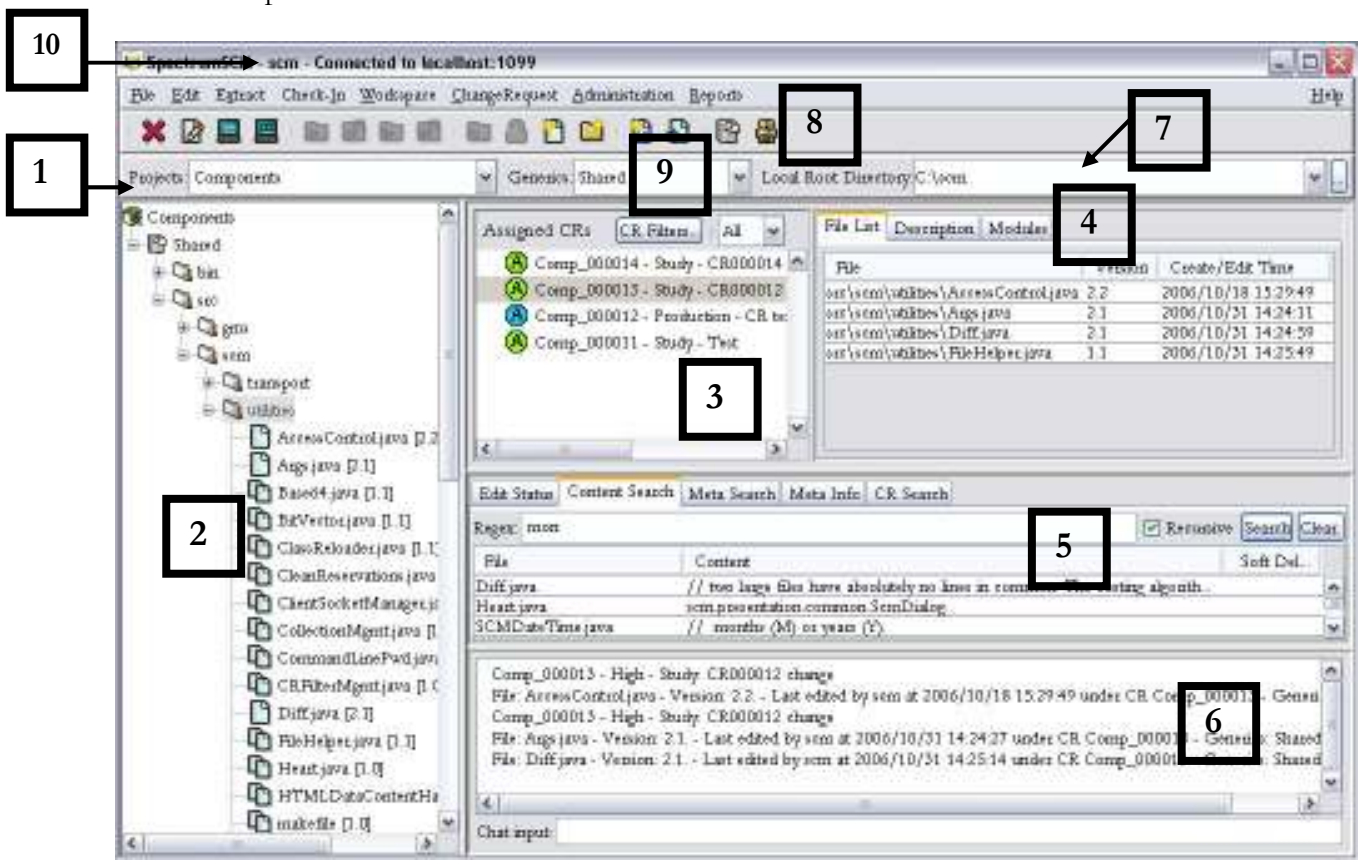
4

4 SpectrumSCM Main Screen

The SpectrumSCM system is designed to provide complex functionality while being as easy to use as possible. The Main Screen is uncluttered, easy to learn, and easy to use. All features are only a click or two away. The look and feel of the SpectrumSCM screens can also be customized by each user. In this chapter, users will also learn how to customize preferences with respect to screen look and feel, fonts, and editors.

4.1 Areas and Features of the Main Screen

This is the main screen of the SpectrumSCM application. The main areas and features are identified by numbers and described below: Detailed explanations for these features are described in the relevant chapters



1. The Project Selection Box and the Generic Selection Box.

The **Project Selection Box** will display all projects that the user has permission to access (is a member of the project team). Select which of your projects you wish to work with. On the screen above, the project "Components" has been selected.

To the right of the Project selection box is the **Generic (Branch) Selection Box**. From this pull down you can select the code branch you want to work with. All generics for a project will be available in this pull-down menu. The SpectrumSCM directories for the selected generic are displayed in the project tree.

When the SpectrumSCM client is closed and re-opened, the last selected project, generic and local root directory will be displayed.

4.2 Project and Generic (Branch) Views/Filters

New If you have a significant number of projects and/or configuration items (generics/branches and/or folders), you can specify views to reduce those to just the sets that you wish to work on at any particular point in time.. For example, if you have 100 projects to which you need access in general, but on a day-to-day basis you are working on maybe 5. You can define a project view to display only those 5.

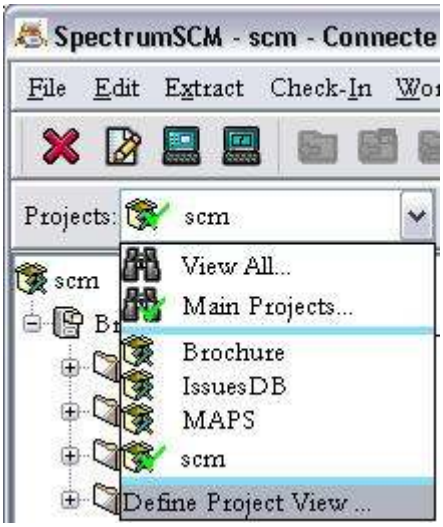


Figure (1)

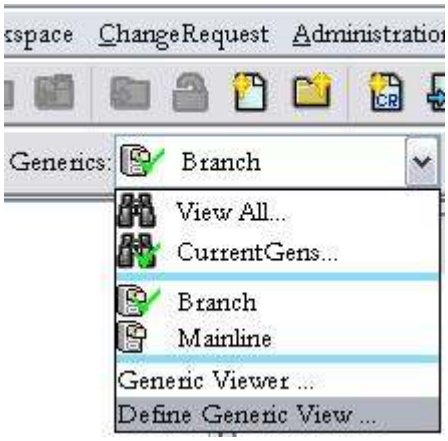


Figure (2)

Similarly with generics/branches and folders, you can define a view to constrain the display to just those frequently used items. Multiple views can be defined and saved so that the user can select the

appropriate view at the appropriate time. A visible icon (a pair of binoculars) indicates when a view is active, so the user knows that they might be seeing a restricted set of data.

To define a project or generic view, select the selector at the bottom of the appropriate toolbar list, as shown in figure (1) and/or figure (2) above. The definition screen as shown in Figure (3) or Figure (4) will be presented.

Figure (3)

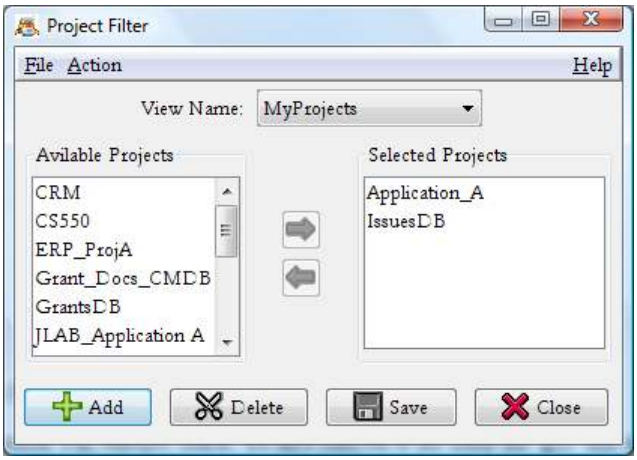
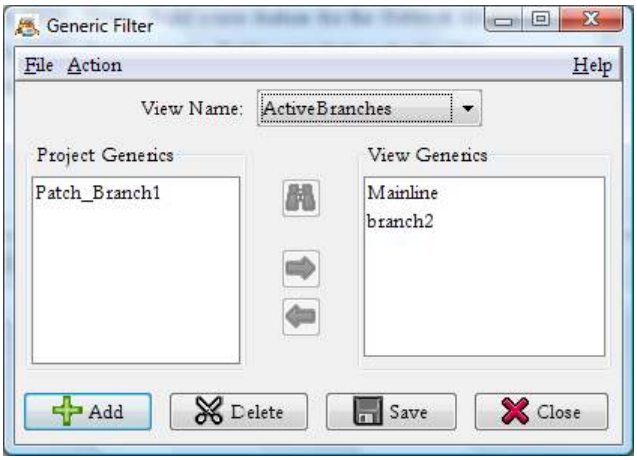


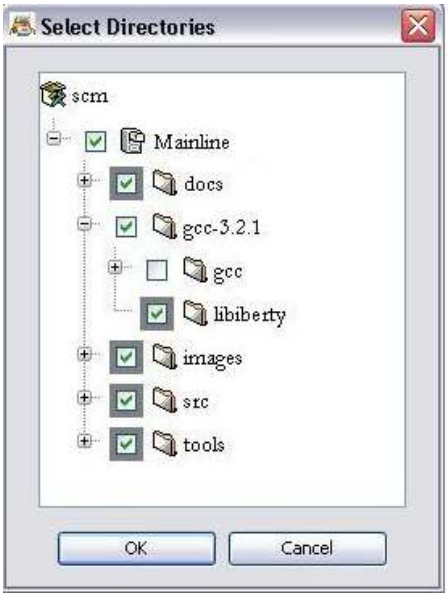
Figure (4)



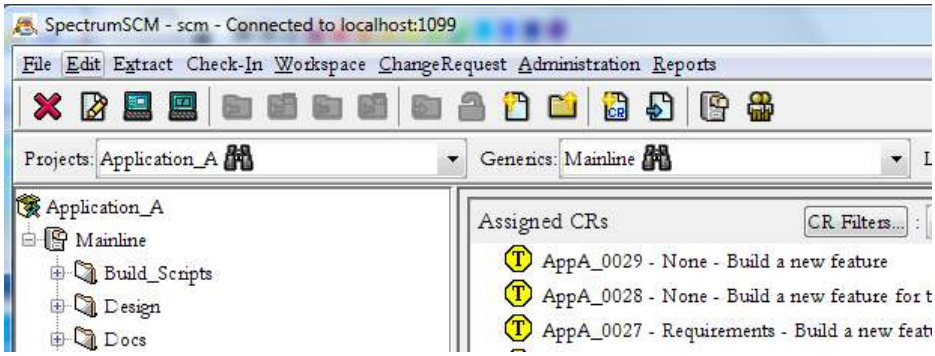
For project and generic views, you have the capability to create **new** views (the **Add** button), and **rename**, **clone** and **delete** existing views. These action items are available from action menu items and/or on the screen.

For generic views you have the additional capability to be able to filter directories. For example, in Figure (5) below, we have selected to not show the "gcc" sub-directory when this view is active. Dark checkboxes (like "docs") imply that all that directories contents will be displayed. Light checkboxes (like "gcc-3.2.1") imply that partial contents will be displayed.

Figure (5)



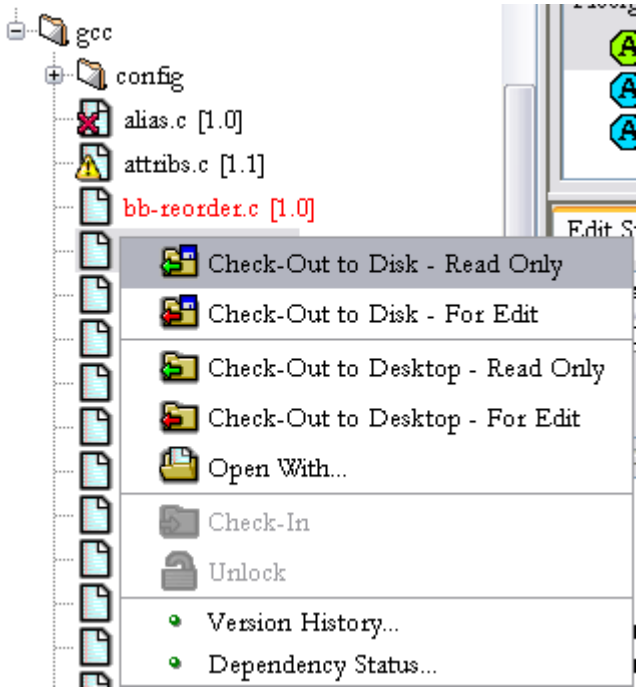
Once a view is defined it can be saved so that it can be easily re-used. When a view is made active it shows with a green tick mark when the project or generic choice boxes are open (as shown in Figure (1) and (2)). When the choice boxes are closed, the fact that a view is active is indicated by the **binoculars icon** show in Figure (3).



To de-activate a view simply select the **"View All"** option (see Figure (1)) . In this way all the project or generic information will be displayed.

2. The Project Tree

The project tree shows the structure of the project, including the generics (baselines, branches) and all their related files and directory structures. If you single select an item, its brief description will be shown in the Messages Area. If you double click on a file, it will be opened into the default editor or the alternate editor if one has been defined. Right clicking an item in the project tree will open a context sensitive menu system with actions that can be applied to the selected item.

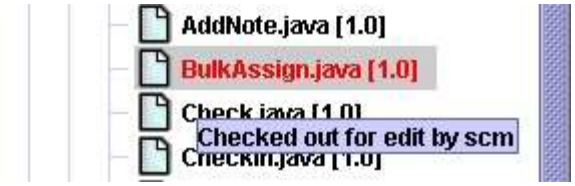


The context sensitive menu system provides quick access to the major file oriented functionality found in the main screen menu systems and tool bars. Directories have context sensitive menus to allow quick access to extract functionality. The contents of this popup menu are sensitive to the user preference, the type of item selected as well as the state of the item selected. Specifically, the context menu presents a single pane in its default "regular" mode. In the "advanced" mode, extra options and the indirect panels are added. In regular mode, this keeps the essential features quickly to hand. In advanced mode, more options are provided but at the expense of a slightly more complicated menu structure.

With the **Workspace Analyzer** turned on (can be toggled on/off from the main screen *workspace* menu), you might also see red 'X' marks () and yellow caution symbols () as indicated against the files *alias.c* and *attribs.c* respectively. The 'X' indicates that while the file is stored in the repository, it is not currently present in the current local root directory. The yellow caution symbol indicates that the local workspace file contents are different from those in the repository. These meanings are also explained in the tool tips shown when the mouse is held over these symbols.

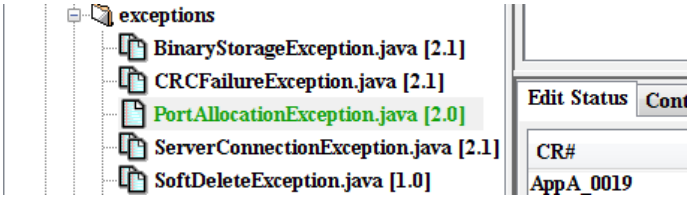
If a file item is already out for edit, only the check-in and unlock functionality will be selectable. When a directory is selected in the project tree only the *Extract Files, CreateDirectory* and *Paste* (move) functionality will be available.
Additional symbols and colorizations:

- The single sheet of paper as shown against the file *Args.java* indicates that this file is not currently common with any other generics.
- The double sheet of paper as shown against the file *Base64.java* indicates that this file is currently common with at lease one other generic. If you single click on this file the message area will show the file details including the list of generics this file is currently common with.
- If the "paper" shown contains ones and zeroes, then this indicates that the file type is binary. If the paper is blank then the file type is textual.
- A file node will be colored **RED** if it is currently checked out for edit by some one else and the tool tip will provide information regarding the user who has the file out for edit.

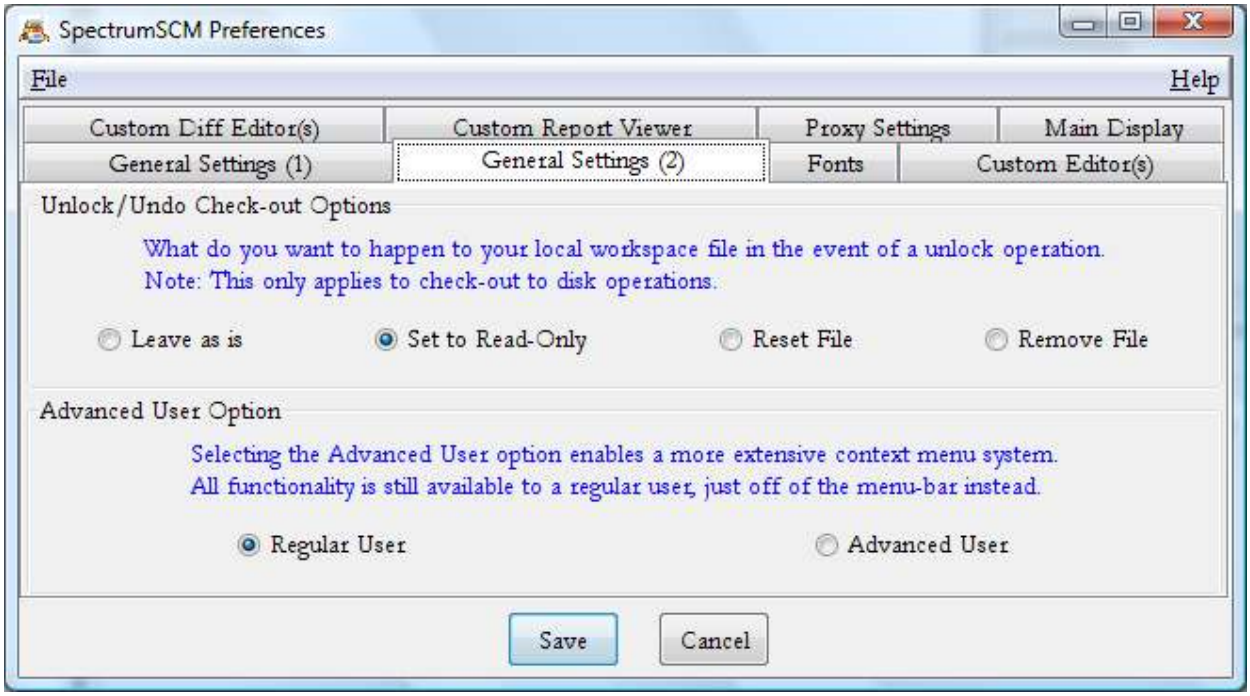


- A file node will be colored **GREEN** if it is currently checked out for edit by yourself.

New



There are two sets of context sensitive menus, depending on the option chosen in the **Edit->Preferences->General Settings (2)** pane. The default provides a simpler direct access to the main CM functions such as check-out and check-in. The **Advanced User** option includes more options and sub-menus, such as move, rename and common/uncommon choices. Note that even with the **Regular User** option selected, all the functionality is till available, it just needs to be accessed off of the menu bar.



File Rename/Move

If the *Advanced User* option is selected under the user preference, then rename and move functionality will be available on the context menu as well as under the File menu.

Rename – Right clicking on a file in the resource tree opens the context sensitive menu for the selected file. To rename a file, choose the Rename option from the menu and specify the new name for the file.

Cut – To move a file to another directory, choose the Cut option from the menu. This copies the file to the clipboard. Select the target directory and choose the Paste option from its context sensitive menu to move the file.

A file can also be moved by selecting it and dragging it into the target directory. Renaming or moving a file does not destroy its version history. The rename and move features cannot be accessed from the menu bar or tool bar.

Open With

The Open With context menu option works with your custom editor preferences.

With the *"Matching executables"* option selected only those editors matching the file type of your selected file will be displayed. Also, if the selected file is a text file, the SpectrumSCM editor will be displayed as an option.

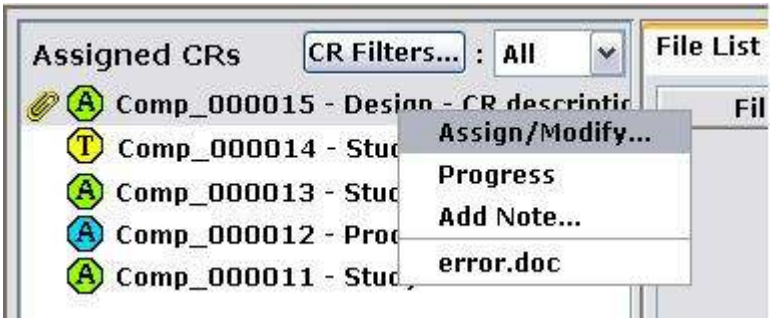
With the *"All executables"* option selected all of your custom editors will be displayed. The SpectrumSCM editor will also be displayed as an option.

Select the editor you wish to use and then select the actual edit operation type you wish to perform from the buttons on the right.

3. Assigned CRs

Change Requests that are assigned to the user or that need user attention.

- If the CR is assigned to the user, a green **A** is displayed  . The user can perform development (or requirements etc.) work relative to this CR.
- If a CR is assigned to the user, but not editable a blue A  is displayed. The user can perform testing or deployment activities for this issue/task, but the repository files can not be changed. If a file needs to be changed relative to this CR, then the CR needs to be moved back to an editable phase in the life-cycle (ex: development, requirements etc.)
- If a CR has been progressed and is in the TBA (to be assigned) state, it is shown to all users with assignment permissions with a yellow **T**  . Users without assignment permissions will not see the TBA CRs in their Assigned CRs panel.



Select a CR (one click) and its brief description will be shown in the Messages area (Severity, State and Header, if it is checked out and by whom). If you double click on a CR, its CR report will be run and displayed in the Reports Viewer.

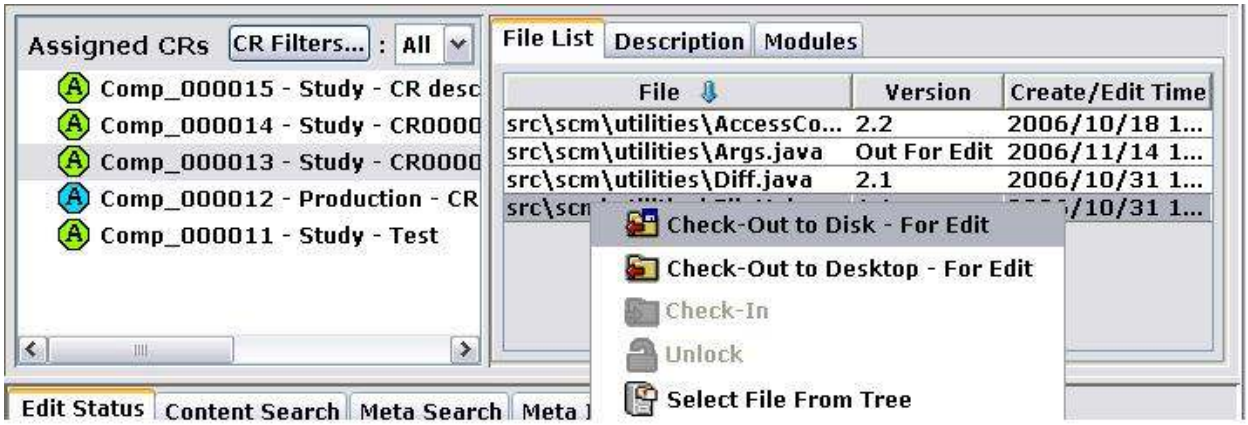
By right clicking on a CR the context sensitive menu system for Change Requests will be posted to the screen. The context sensitive menu allows the user to quickly open the CR Assign/Modify screen as well as the CR Progression screen and also displays a list of attachments that are currently associated with the selected CR.

In this example the selected CR has a single file attachment, *error.doc*. The menu also allows the user to quickly access the Assign/Modify and Progress screens.

Clicking on a CR attachment will open the attachment operation screen. This screen allows the user to open the attachment directly to the desktop, or to save the attachment to a file on the local file system.

4. File List/Description/Modules Panel

File List tab lists the files associated with a CR. Select a CR and click on **File List** tab, the files that have been edited related to the selected CR will be displayed on this panel with file version and create/edit time details. If files are currently out for edit, they are shown with the "Version" column marked as "Out For Edit". The Create/Edit Time column shows when the file was last touched relative to this CR. For a check-in, the check-in time will be shown. For an ongoing edit, the check-out time will be shown.



Right-click context sensitive menus give access to check-out, check-in and unlock functionality. The *"Select File From Tree"* option will open up the repository tree to select and show the corresponding file, this can be most useful when working with a large, diverse repository tree.

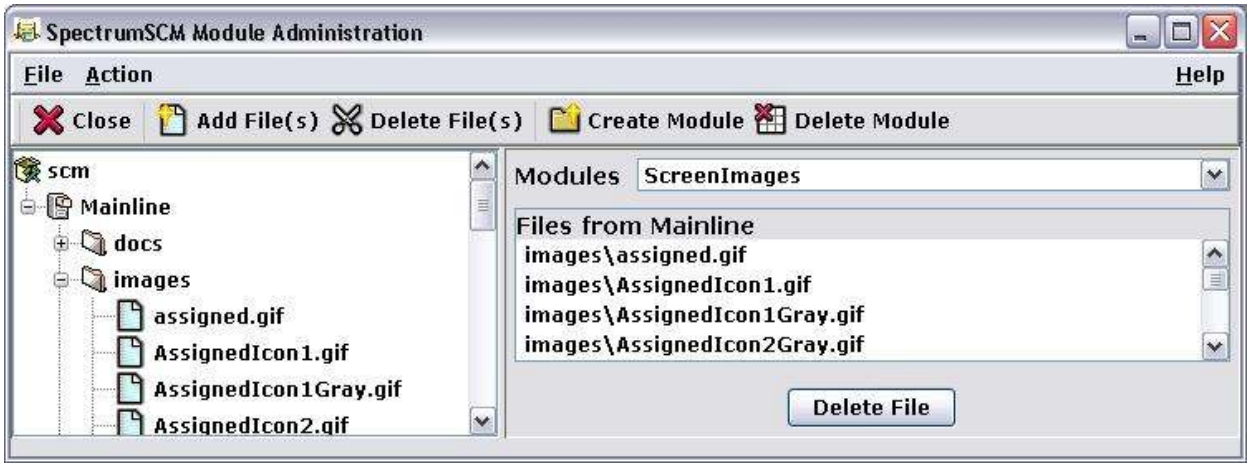
Description tab displays CR description. Select a CR and click on Description tab, full description of the selected CR will be displayed on this panel.

.

Modules tab lists the modules currently defined by this user. Modules can be used to operate on a group of files with one button selection. Modules can be defined for any grouping of files that the user desires. Once a module has been defined, it can be used to check-out to disk (or check-in from disk) all of the related files at the same time. Modules are created using the **Administration->Module Admin** menu option available via the Main Screen. To define a module select the **New**

tool-button and enter the desired module name. Then select the files from the project tree on the left that you want in the module.

Files can be deleted from the module using the Delete File button. A module can be deleted by selecting the **Scissors** tool-button. A file can be removed from the module by selecting the file and using the **Delete File** button. To extract or check in all files associated with a module, select a CR, the module and the function.

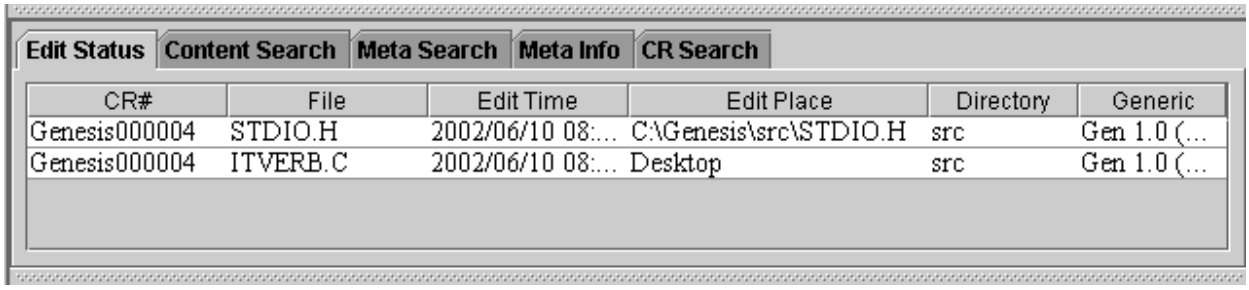


(See details on using modules for file management in Chapter 8).

5. The Middle Panel

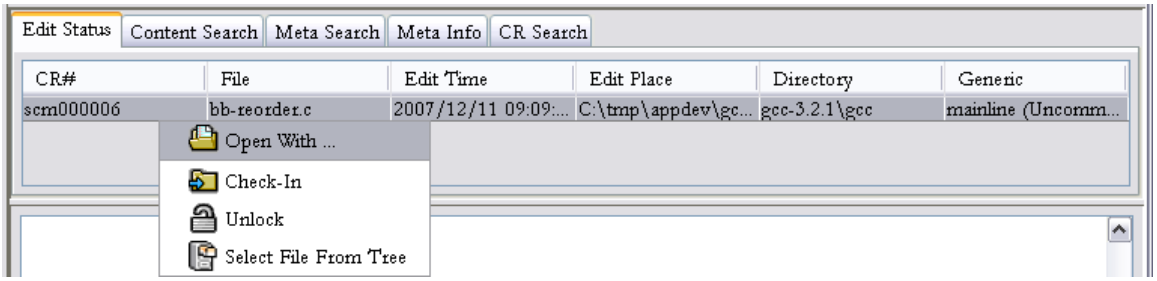
The middle panel has five tabs to show the status of files currently checked out for edit, and to allow various search capabilities.

On the **edit status** tab, files that are currently out for edit by you are indicated. The display includes the date and time that the file was checked out, which change request and generic are associated, and where the edit is being performed (desktop or file-system).

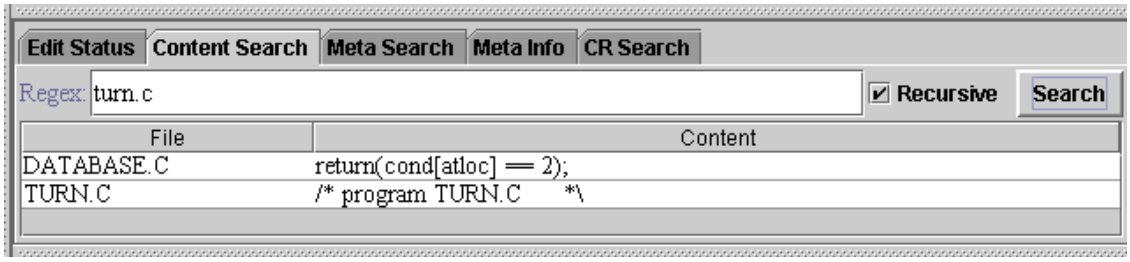


Double-clicking an item will open that file in your selected editor. If the edit was "to the desktop", the option to continue that edit will be presented. If the file is out for edit "to the disk", then the right-click menu will also have an "Open With" option to allow you to pick one of your custom editors.

Check-in and unlock functionality is also available by right clicking with the mouse. Multiple files may be checked-in or unlocked simultaneously.



The **Content Search** tab and the **Meta Search** tab enable searching of the projects source files. The search will cover all the appropriate files contained in the selected directory in the SpectrumSCM system project tree. This feature is highly useful for finding text within the sources themselves.



A content search will search the text files in the repository and report on those that contain matches against the requested search pattern. Only the first match is displayed against each file.

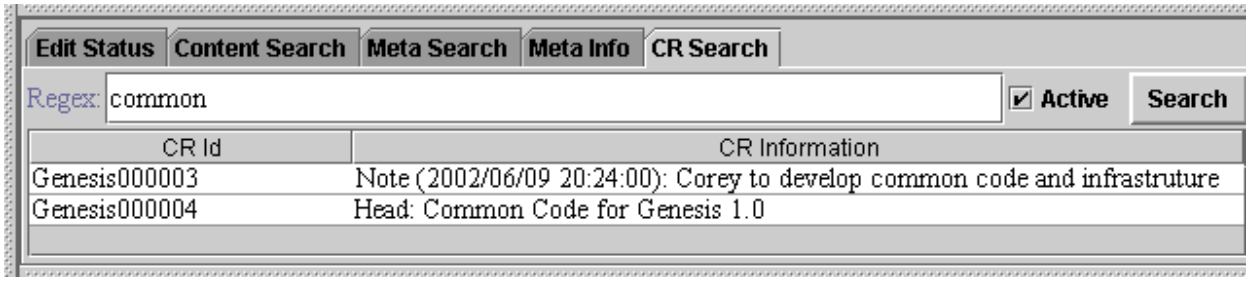
Double-clicking a selected entry will open your default editor against the repository version of this file. Right-clicking an entry will open the repository tree view to the selected file, this then enables quick and easy access to all of the extended file operations such as check-in/out, rename etc.

A **meta search** is controlled the same way, but searches the meta information associated with a file instead of its contents. Meta information is most useful when considering binary files such as drawings or pictures that would otherwise not be searchable. The meta information can be established for any file with a description of what that file contains or any other useful text. Another example would be if SpectrumSCM was being used as a documentation control library, then the meta information could be used to store key words. Double-click and right-click operations function the same as under the content search panel.

Meta information is maintained using the Meta Info tab, based on the selected project tree entry. When a file is added into SpectrumSCM its meta information is initialized to its filename and path. This enables files to be found by name in even the largest of projects simply by searching the meta information.

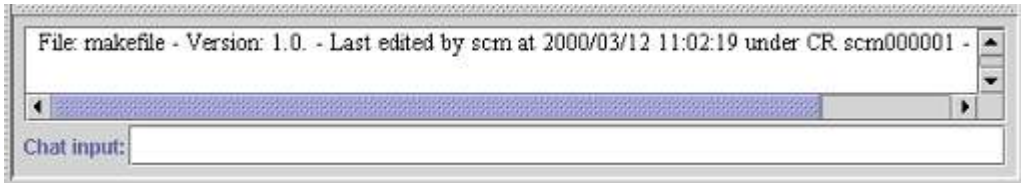
CR search provides the capability to search the change requests for any text matching the requested search pattern. Control is defaulted to only search for active CRs but can be expanded to search all CRs. Note however, that on a large project, searching all CRs could involve a lot of work and therefore a response might be a while in coming.

Double-clicking an entry will present the CR report for the selected change request. Right-clicking an entry will provide easy access to the Assign/Modify screen if you have the appropriate permissions.



6. The Messaging Area

System and chat messages are displayed here.



7. The "Local Root Directory"

The root directory defines the location on the local hard disk where files to be checked in are found and files extracted to the hard drive are placed. This is needed because files stored under SCM have only relative path names. An example "root directory" would be the directory in which you perform your local product builds or compiles. Another example would be a directory that you use when you are loading source into SCM from the directory.

Subdirectory names in all local root directories need to match the SpectrumSCM file structure in the project tree. Note that you can have multiple root directories if needed. Select the one you wish to work with via the Local Root Directory pull-down menu selection box.

An example local **work** directory in a Windows environment might be C:\GenesisBuild\src. In this case, C:\GenesisBuild would be the local **root** directory. In a Unix or Linux environment, a local **work** directory might be /Genesis_build/src and the **root** directory would be /Genesis_build.

The Browse button can be used to select the root directory from the file system.

8. The SpectrumSCM Main Screen Toolbar

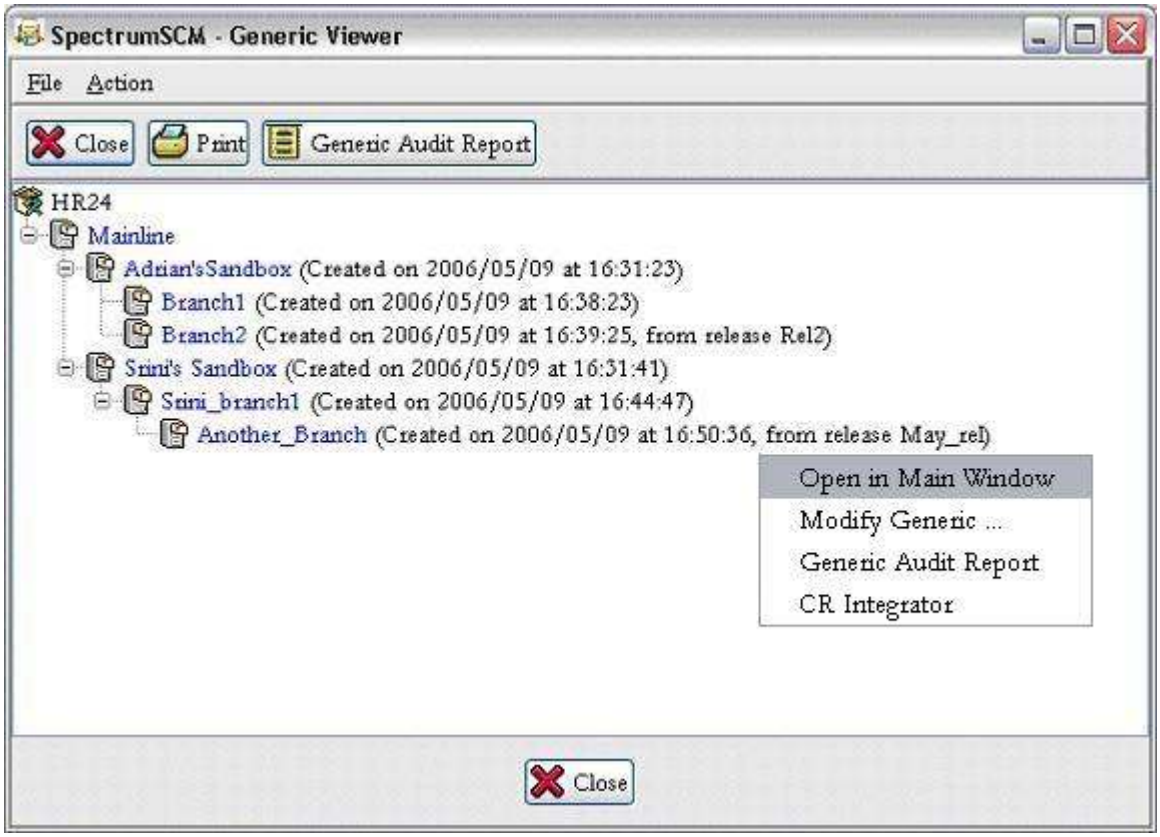
All of the toolbar buttons have explanatory tool tips, which appear as the cursor is passed over each icon. Toolbar functions, in order as they appear on the toolbar from left to right, are:

Exit		Exit the application.
New Editor		Startup a new empty editor panel.
Single Editor		Startup an editor panel on the file selected in the Project Tree.
MultiEditor		Startup a dual editor panel on the file selected in the Project Tree.
Checkout Read Only To the Desktop		Start an editor panel on the selected file in Read Only mode.
Checkout Read Only To the Disk		Write a Read Only version of the selected file to the local disk. The full path is determined from the Local Root and the directory the file resides in, in the project.
Checkout for edit To the Desktop		Start an editor panel on the selected file in Live Edit mode. Note that an Assigned CR must be selected for this operation to proceed. The toolbar button always performs an uncommon edit, for a quick common edit use CTRL K
Checkout for edit To the Disk		Check out a writable version of the selected file to the local disk. Note that an Assigned CR must be selected for this operation to proceed. The toolbar button always performs an uncommon edit, for a quick common edit use CTRL Q
CheckIn		Check in the selected file. The file selection can be made in the Project Tree, the Module window or in the Edit Status middle panel. Note desktop edits must be checked in from the desktop editor.
Unlock		Unlock the item selected on the Edit Status panel.
Add a File		Add a small set of files (individually) into the SCM system.
Add a set of files		Add a set of files identified by a directory and some filters. This feature can be used to load a whole project tree into SpectrumSCM.
Create a New CR		Create a new Change Request.
Progress a CR		Progress the selected Change Request.

Generic Hierarchy		View the current Generic Hierarchy
Active Users		Show the users currently online.

9. Generic Viewer

Generic Hierarchy can be viewed by selecting *Generic Viewer* pull down option in **Generics Selection Box.** The generic viewer can be used to view the details about the existing generics and their relationships to one another. Specifically, when they were created, if they are locked, which generic they were based off of (if any), and if they were based off of a release.



Right-click operations can be performed to switch the main screen generic, open the generic modification window, or perform generic comparisons. Generic comparisons can be run either in the report format or interactively through the CR Integrator.

Pointing at a particular generic will also display the generics description via its tooltip (if a description has been populated). Additionally, if the user does not have "Modify Generic" permissions, that popup menu item will be replaced with a "View Generic Description" option.

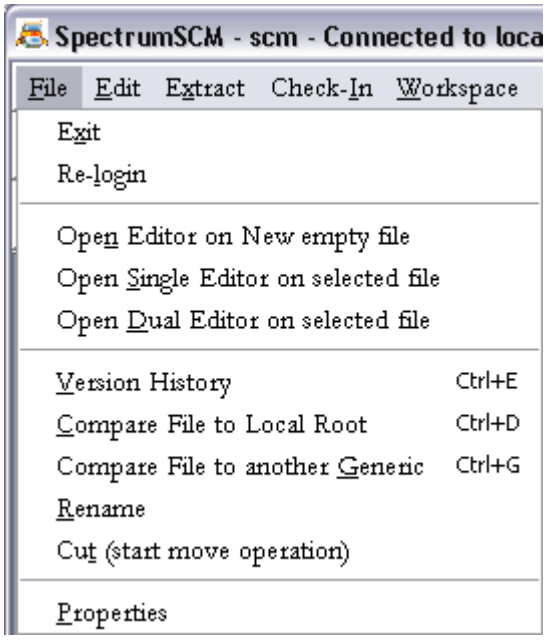
10. Current User

At the very top of the screen, the current user's name is displayed. The title bar also displays the server address and port number the client instance is connected to

In this example, "scm" is the current user.

4.3 Menu Items

4.3.1 File

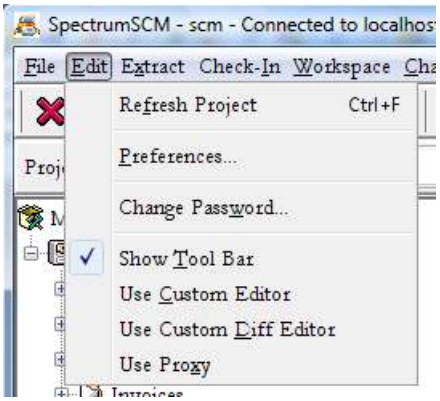


- **Exit** – Exit the application
- **Re-login** – Close the current session and login as another user. The application does not exit during this action
- **Open Editor on New empty file** – Opens the default editor on a new empty file *(See Chapter 8 for details on using the editor).*
- **Open Single Editor on selected file –** Opens the default editor or custom editor on the Selected file
- **Open Dual Editor on selected file** – Allows the user to view and/or edit two versions of the same file at the same time on a split screen.
- **Version History** – Displays file version history with related CR information
- **Compare File to Local Root** – Opens the merge/dual editor on the selected file. The left-hand panel with contain the repository version of the file. The right-hand panel will contain the local root file system version of the file. The difference buttons can then be used to view and browse any differences.
- **Compare File to another Generic** – First presents a popup to request which other generic you wish to compare to. Then will open the merge/dual editor on the selected file in this generic (left) and the other (right). The difference buttons can then be used to view and browse any differences.
- **Rename** the selected file. File history is maintained such that older file versioned will still be extracted under the old name. Newer versioned of the file will be extracted under the new name. In this way your release build will be maintained and reproducibly correct.
- **Cut File** – Starts a file move operation. To complete the operation, select the target directory, right-click and select the **paste** option. File history will be maintained in that older file versions will be

extracted in the old directory, newer versions will be extracted in the new directory. In this way your release build will be maintained and reproducibly correct.

- **Properties** - Shows the currently selected file(s) character-set and end-of-line expansion properties. The dialog also can allow for the modification of these properties if the user has edit permissions. The meaning and use of these properties are discussed in more detail under Chapter 8 – Source File Management.

4.3.2 Edit



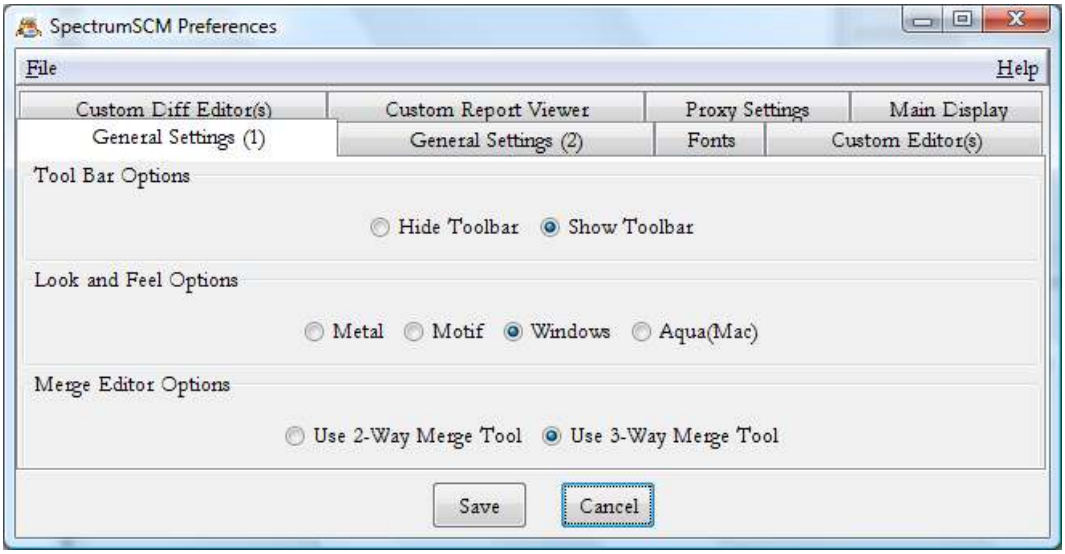
Various options are available under the Edit Menu option

Refresh Project

Refreshes the data displayed on the main screen with any updates from the server. Otherwise, changes made to the server, will not automatically show up on the Main Screen.

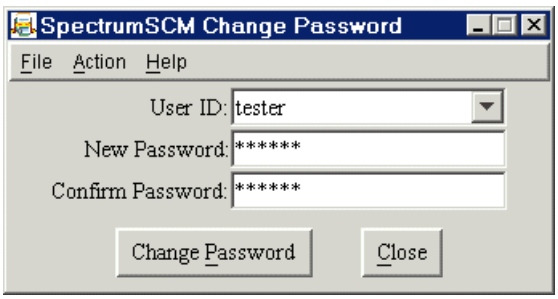
Preferences

Set your personal preferences. This includes fonts, alternate editor preference, look-and-feel and alternate report viewer.



(See Chapter 5 for details on user preferences.)

Change Password



A user can change his or her SpectrumSCM password. An administrator can reset any user's password.

Show Tool Bar

Quick button for turning on or off the display of the toolbar.

Use Custom Editor

Quick button for turning on or off the use of your custom editor. If the custom editor feature is turned off, the SpectrumSCM editor will be used instead. Note that this can also be toggled via the keyboard quick keys of **`Alt+EC`**.

Use Proxy

Quick access button for enabling or disabling the use of the SpectrumSCM Proxy

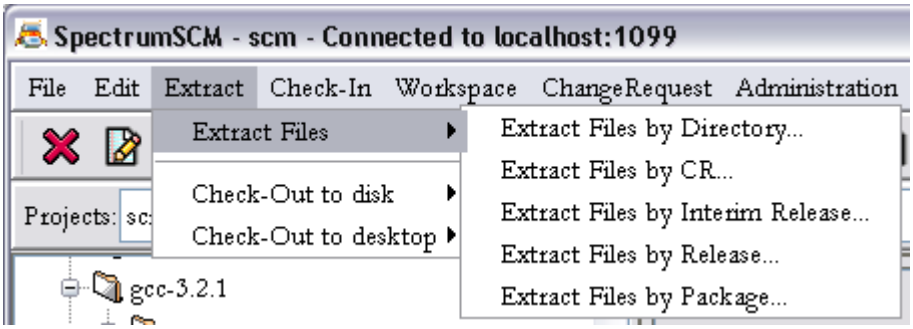
4.3.3 Extract

Extract Files

Files can be extracted en masse or checked out individually. You can also simply **Drag and Drop** files and folders onto your Desktop or any where on your file system.

New

(See details on Extract and Checkout in Chapter 8)

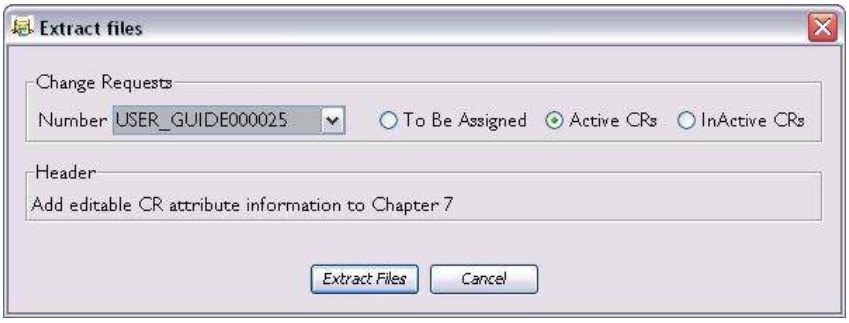


Extract Files by Directory

Select the required directory to extract from the Project Tree. That directory structure will be placed on your local hard disk under your local root directory.
This function allows files loaded into **SpectrumSCM** to be extracted as a group into a specified directory for development and building.

Extract Files by CR

Extract files by CR allows a user to extract to the local hard disk just those files that are associated with the selected CR. The CR extract window gives the user the opportunity to choose which CR to extract and whether that CR comes from the active or inactive list.



Extract Files by Interim Release

Select CR's that have passed a particular phase in your life-cycle and extract the set of files based on that. This feature allows you to extract informal test builds during the later stages of development, or the early phases of testing, before you want to build a formal release.

Extract Files by Release

Select the release to be extracted from the presented window. The contents of that release will be placed on your local hard disk under your currently selected local root directory. Releases and their contents are defined under the Release Management screen, this is accessed off of the Administration menu and is described in Chapter 9 of the user guide.

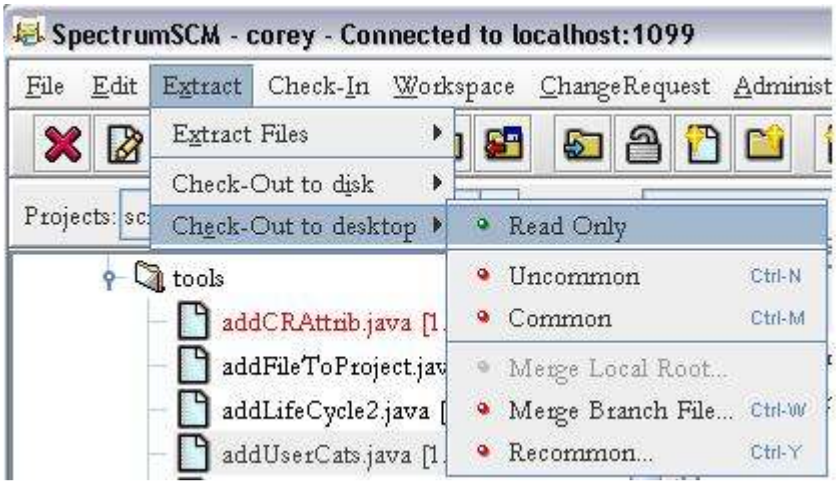
Extract Files by Package

Select the package to be extracted from the presented window. The contents of that package will be placed on your local hard drive under your currently selected local root directory. Packages and their contents are defined under the Package/Component Management screen, this is accessed off of the Administration menu and is described in Chapter 9 of the user guide.

Check-out to Disk

Options to check-out the selected file by version (read-only) or common/uncommon (default mode is "uncommon") for edit, and common concurrent. Concurrent editing allows multiple users to edit the same file on the same branch at the same time.

Check-out to Desktop



Options to check-out the selected file by common, uncommon, for merge or recommoning to the desktop. (Default mode is as defined in the generic set-up process, see common/uncommon overview below). The file will be extracted but will not be placed under your local root directory. Rather the file will be presented in either the standard SpectrumSCM editor or the appropriately specified custom editor.

Uncommon / Common and Common concurrent

Unless multiple Generics are being used in parallel (for example to maintain 2 similar source bases for 2 different customers) the default check-out mode (uncommon) is all that is required and the developer need not be concerned about these options.

(See Chapters 6 and 8 for details on branching, merging and recommon)

Checking out **"uncommon"** will mean any file changes will only be made against that specific generic. If a check-out is performed "common" then the file changes will be made against ALL the generics with which that file is currently in common with. The list of generics a file is currently common with can be seen by selecting the file and then viewing the message area.

Checking a resource out **"Common Concurrent"** allows for multiple developers to edit the same file at the same time. Upon check-in the system checks to see if the file must be merged or can simply be checked in. If the file requires merging, the multi-paned merge editor is automatically started and the user should then merge the contents with the contents of the repository.

When multiple generics are to be used in parallel the Generic Engineer must decide who is going to make the branching decisions for file check-outs. If it is to be left to the Developer then the generic should be left in the *unlocked* state.

Once the mode of the Generic has been established, if the commonality control is still with the developer, then he/she can choose the appropriate option. Checking out "Common" is a powerful feature since it can be used to apply a single "fix" to multiple generics in one edit, however the developer would have to be careful of side-effects.

- **Common:** Versioned files that are physically the same across generics
- **Common concurrent:** Multiple user, write access to the same file on one generic
- **Uncommon:** The act of physically separating versioned files from multiple generics

Merge Local Root

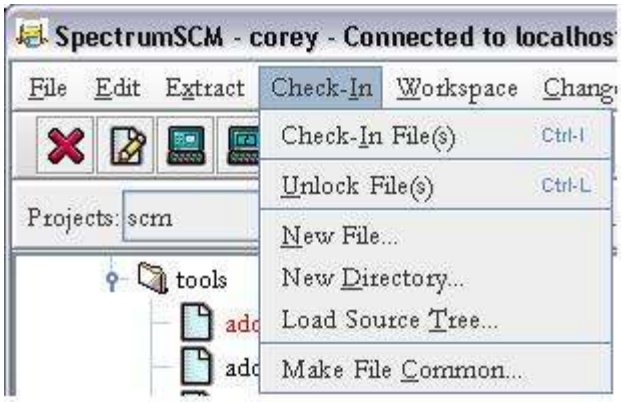
Merging changes from your local root up into the repository can be performed with this option. This option is enabled if the file on the local hard-drive is different than the repository version and is writable.

Merging allows the synchronization of files across two branches or the repository and file system versions. Merging in SpectrumSCM is the act of copying the changes made in one version of a file into another file using the SpectrumSCM Merge Editor. Merging two branch files allows the developer to copy/merge differences but retains the separate development paths.
In both the Merge and the Recommon processes, the files selected will be put in the split screen Merge Editor to allow the changes between the two versions to be identified and reconciled.

Merge Branch File & Recommon

Merging changes from your current branch/generic to another. Recommoning brings the two versions of the same file (in two different generics) back into one version shared between the two generics (makes them common again). This is useful when a parallel development effort on a project is brought back together to create one code path.

4.3.4 Check-In



File(s) can be selected in the Project Tree, File List/Module Window or in the Edit Status middle panel. *(See Chapter 8 for details on file management)*

Check-In

Check in the edits from the currently selected file(s). The check in operation can b performed from the main menu as shown, and can also be performed from the context sensitive menus found in the file tree itself and the edit status panel.

Unlock

Unlock the selected file from edit.

New File

Add a new source file into SpectrumSCM.

New Directory

Create a new directory under the selected repository location.

Load Source Tree

Add a directory structure into SpectrumSCM. You can also simply **Drag and Drop** files and folders onto your Desktop or any where on your file system.

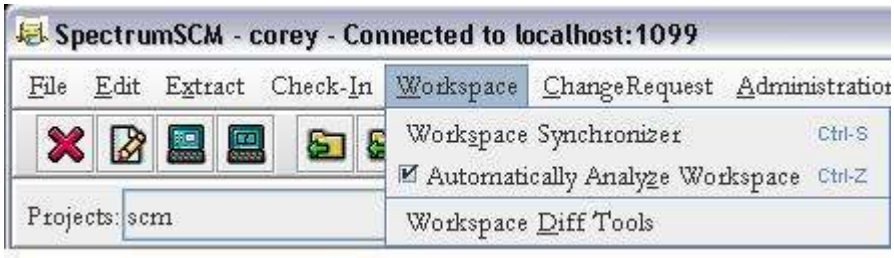
New

(See details on Extract and Checkout in Chapter 8)

Make File Common

Make a file common across several generics at one time. This feature is useful for adding a file commonality relationship across multiple generics, where the file did not exist before. Note, if the file does exist in the other generic(s), then the *Extract->Checkout to Desktop->Recommon* option should be used instead.

4.3.5 Workspace



Workspace Synchronizer

The Workspace Synchronizer enables the user to easily keep their local workspace up to date with respect to other user's repository changes. It also enables a user to synchronize or merge their offline work with the repository.

Automatically Analyze Workspace

Toggle the Workspace Analyzer on and off. When the analyzer is turned on, the repository view of the project files will include additional icon information concerning the difference between files in the local root directory and those in the repository. Additional information includes whether file is in-sync (no icons), out of sync( ) or missing altogether( ).

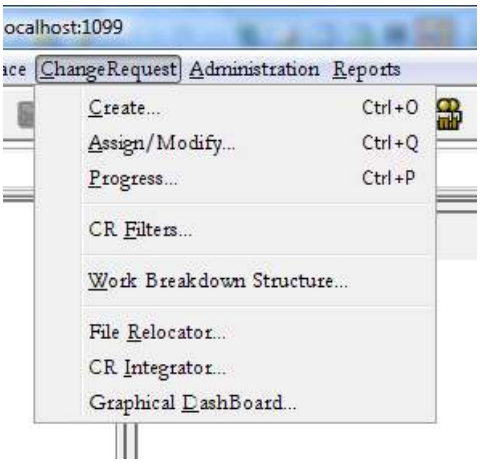
Workspace Diff Tools

Provides the ability to compare any two directory structures or any 2 or 3 files. Directory comparison includes by date, size or checksum. File comparison is performed using the SpectrumSCM 2-way or 3-way diff editor for text files.

4.3.6 Change Request (CR)

CR stands for Change Request. Similar acronyms are MR for modification request and WR for work request. Change Requests are the glue that binds the SpectrumSCM system together. Users are required to make all changes to the system through one or more CRs to track the reasons for and history of the changes. Releases are composed of units of work that are defined by CRs.

See Chapter 7 for details on Change Requests and how they are used for issue tracking, source configuration management and project management.



Create CR

Create a new Change Request, either brand new or based off an existing one by using the Auto Fill feature.

Assign/Modify CR

Assign the Change Request to someone for work or Modify the Change Request attributes, header or description. Note any such changes are tracked through the CR notes for auditability purposes.

Progress CR

Progress a Change Request into the next life cycle phase.

CR Filters

Define and manage your Change Request filters – used to reduce the number of CRs displayed on the Assigned CRs list and/or the CR Assign/Modify screen. (*see section 7.3.4 for a full description of CR filters and their use*). Multiple filters can be defined to present views based on what information you want to see at any point in time.

Work Breakdown Structure

Opens the Work Breakdown Structure main screen. *See chapter 7.5 for a full description of how work breakdown structures work.*

File Relocator

Allows file edits to be relocated from one Change Request to another.

CR Integrator

Compare, Merge and Recommon Generics from a CR and File perspective.

Graphical Dashboard:

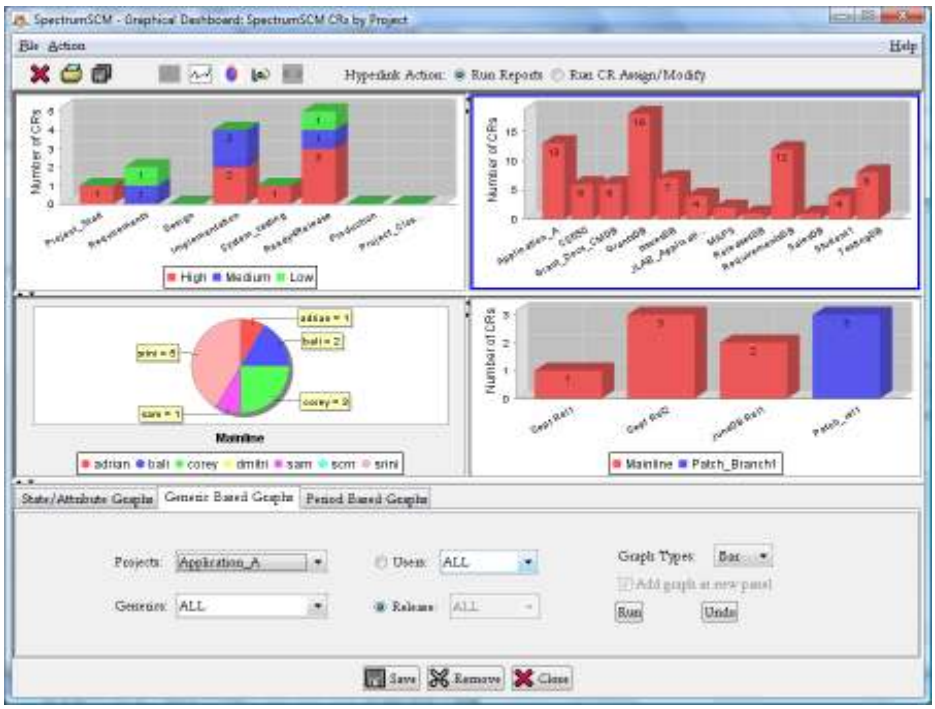
This feature is available as an additional optional module (SpectrumSCM Graphical Dashboard 1.0). The SpectrumSCM Graphical Dashboard 1.0 works in conjunction with the SpectrumSCM 3.0 version. The **Graphical Dashboard menu item will be activated** when you purchase this module.

The graphical dashboard is a project **performance and metrics** dashboard which gives software managers, team leads, QA engineers, release managers, Configuration managers, and Auditors quick insight into project risk, status, and trends by presenting the Change request/Tasks progress and status in the form of powerful Graphs and Charts. These displays are available individually or as up to 4 panels within a single Graphical screen. The Dashboard provides an array of views with color-coded status to be able to quickly spot trends and take corrective actions.

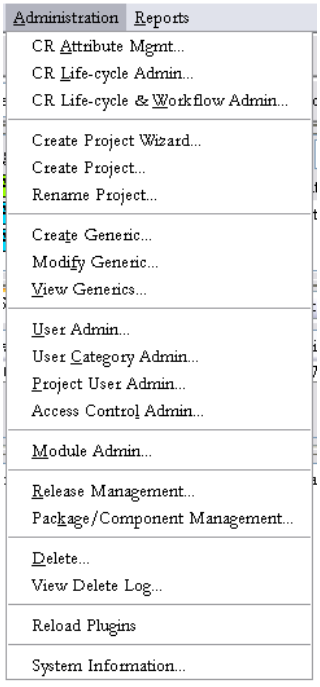
The Dashboard provides solutions for measurement and metrics, collecting data from one or more projects, one or more branches/generics, one or more users, one or more releases, or based on CR attributes. Data can also be collected over periods of time. The charts and graphs present intuitive summaries of key information in a single view so that users can analyze trends, perform management by exception, and drill down for more information when necessary.

Furthermore, the Dashboard can display up-to-date project status information in a graphical multilayer format, allowing managers to focus on decision making rather than manually gathering data and compiling reports to help ensure the success of their business. Project Managers and

Stakeholders need not deal with an increasing amount of data, sorting through hundreds of reports and project communications in search of bottlenecks or areas not meeting project expectations.



4.3.7 Administration

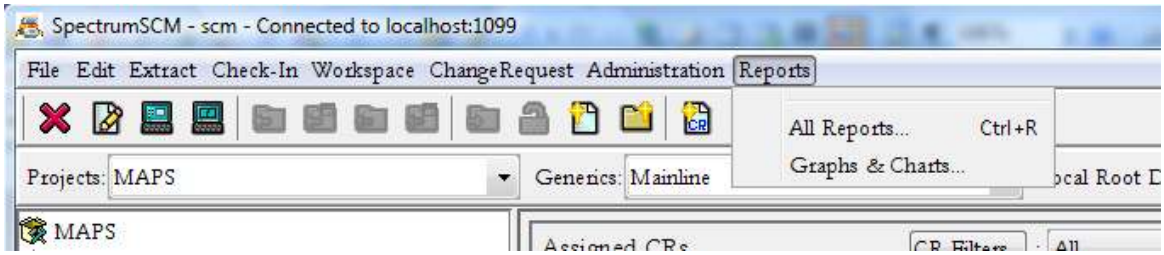


Except for module administration, these functions can only be performed by a user with appropriate permissions. Generally these would be Administrator or Project Engineer authority for the setup items, and Generic Engineers for the "Generic" and Release/Package Management items. Delete would either be controlled (generally) by the Generic Engineer or possibly the senior developers/team leads. *(See Chapter 5 for details on User Set-up) (See Chapter 12 for details on Administration functions).*

CR Attribute Management	Allows customizing attributes associated with CRs, defining and managing system-wide and project specific change request attributes.
CR Life-cycle Admin	Allows managing the linear life-cycle definitions and their assignments to projects. Note, use of this screen/functionality is mutually exclusive with the Workflow Administration screen. This is because the linear screen defines the life-cycle in the form of a list, whereas the workflow defines it in a 2-dimensional graphical form.
CR Life-cycle & Workflow Admin	Manage the graphical workflow definitions and their assignment to projects.
Create Project Wizard	Create a new project using the interactive project creation wizard
Create Project	Add a new project.
Create Generic	Add a new generic (baselines, branches) to the current project.
Modify Generic	Modify a generic, specifically to manage its commonality lock i.e. whether developers are allowed to perform common edits.
View Generics	View the current generic relationships and operate upon them.
User Admin	Create SpectrumSCM application login-ids. Maintain the system-wide user list, including names, default password and contact information.
User Categories	Define and maintain the set of user categories/roles.
Project-User Admin	Assign Users to a project team and manage the Project-User relationship. I.e. manage which users are assigned to work on which projects and with what roles.
Access Control Admin	Define access permissions for resources within the repository. Access permissions can be assigned at the Generic, Directory and File levels against particular user roles.
Module Admin	Define and Maintain your module definitions.
Release Management	Define a release set, manage the association of change requests with that set.
Package/Component Management	Define a package of components to extract for build or release. Manage the association of components with their packages.
Delete	Delete an item or items (project, generic, files, etc).
View Delete Log	Tabular view of resources that have been deleted from the repository. Items are marked as soft (restorable) or hard deletes, when the delete occurred, which CR was used and who executed the action.
Reload Plugins	Dynamically reload user created custom plugins. Please see chapter 14
System Information	View a graphical representation of the space available in the repository.

4.3.8 Reports

This function lists all reports available to the user. A report or reports can be selected and run. Frequently run reports can be personalized and saved as Personalized Reports that can be rerun by the user who created them. This saves the effort of having to create the report again.



(See Chapter 10 for details on reports)

All Reports

Produce a list of all pre-defined reports.

Personalized Reports

Execute personalized reports such as the "File History Summary" shown above. These are set up through the "All Reports" reports manager window, "Action" menu.

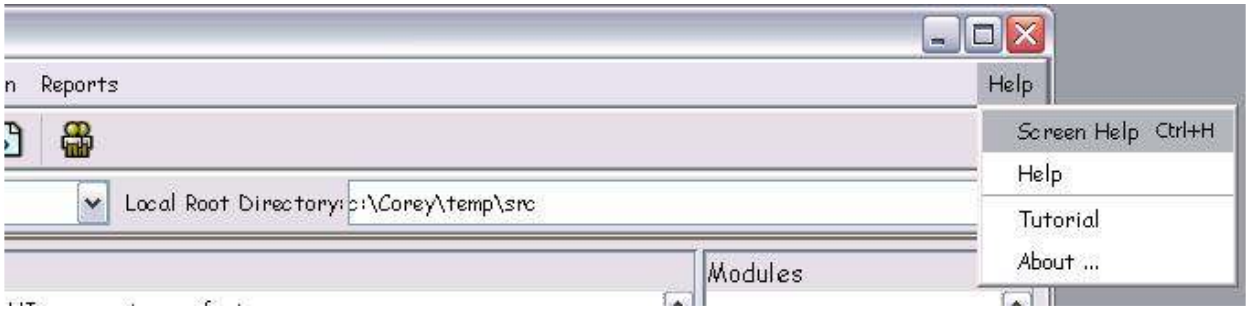
4.3.9 Graphs and Charts

New This function lists all the graphs and charts available to the user. A graph can be selected and run. Frequently run graphs can be personalized and saved as Personalized graphs that can be rerun by the user who created them. This saves the effort of having to create the graph again.

This feature is available as an additional optional module (SpectrumSCM Graphical Dashboard 1.0). The SpectrumSCM Graphical Dashboard 1.0 works in conjunction with the SpectrumSCM 3.0 version. The **Graphs & Charts menu item will be activated** when you purchase this module.

(See Chapter 16 for details on Graphs and Charts)

4.3.10 Help



Screen Help

Each screen in SpectrumSCM has screen help available for that screen. It can be accessed from within that screen.

Help

This menu item gives access to all of the help screens available by topics. These are HTML format and hence you can navigate easily by hyperlink to any topic of interest.

Tutorial

There is a simple "Getting Started" Tutorial to aid in learning SpectrumSCM and to get the latest most up-to-date information. The tutorial is accessed via the Help menu. It can be viewed via an Internet link to Spectrum's website for access to the most current version. The tutorial is also provided on the installation CD and can be installed locally for sites where web access is not available.

The tutorial covers the following topics:

- Installation
- Create a project
- Modify Generic
- Create a User
- Create CR
- Progress CR
- Adding Source
- Check out files
- Branching files
- Recommoning files
- Quick Start
- Create Generic
- Assign life cycle Phases
- Add User Roles/Access Authorization
- Assign CR
- Establish local root directory
- Load Directory Tree
- Check in files
- Merging files
- Review Reports

About - Provides information about the installed version of SpectrumSCM (version number, Service Pack and license data and expiration date) and provides the telephone phone number and email address of the SpectrumSCM support team.

Maximum users indicates the number of users who can concurrently access the SpectrumSCM server.

If you have purchased the graphical dashboard feature (or running an evaluation copy) then this will also reflect the number of graphs user licenses.

4.4 Quick Keys

Quick Keys operate from any screen.

4.4.1 By Function

Menu	Menu Item	Mnemonic Alt+ Key	CTRL+ Key
File			
	Exit	FX	
	Login	FL	
	Edit New	FN	
	Open single editor	FS	
	Open dual editor	FD	
	Version History	FV	E
	Compare Local Root	FC	D
	Compare Generic	FG	G
	Rename	FR	
	Cut	FT	
	Properties	FP	
Edit			
	Refresh	EF	F
	Preferences	EP	
	Change Password	EW	
	Show Toolbar	ET	
	Custom Editor	EC	
	Custom Diff Editor	ED	
	Use Proxy	EX	
Extract			
	Extract files by Directory	XXD	
	Extract files by CR	XXC	
	Interim Release	XXI	
	Extract files by Release	XXR	
	Extract files by Package	XXP	
	Disk – Read – Only Top	XIRT	B
	Disk – Read – Only - Version	XIRV	(use CTRL-E to access the version History)
	Disk - Uncommon	XIU	U
	Disk - Common	XIC	K
	Disk - Concurrent	XIT	T
	Desktop – Read Only	(Use file double-click)	
	Desktop - Uncommon	XEU	N
	Desktop – Common	XEC	M
	Merge Local Root	XEL	
	Merge Generic	XEG	W
	Recommon	XER	Y
Check-In			
	File	II	I
	Unlock a file	IU	L
	Add a New File	IN	
	Add a New Directory	ID	
	Load Source Tree	IT	
	Make File Common	IC	

Workspace			
	Work-Sync	WS	S
	Work-Analyzer	WZ	Z
	Workspace Diff	WD	
Change Request			
	Create	CC	O
	Assign/Modify	CA	Q
	Progress	CP	P
	CR Filters	CF	
	WBS	CW	
	File Relocator	CR	
	CR Integrator	CI	
	Graphical Dashboard	CD	
Administration			
	Attribute Management	AA	
	Life-cycle Management	AL	
	Workflow Management	AW	
	Create Proj Wiz		
	Create Proj		
	Create Generic	AT	
	Modify Generic	AF	
	View Generics	AV	
	User Admin	AU	
	Category Admin	AC	
	PUser Admin	AP	
	ACL	AL	
	Module Admin	AM	
	Release Management	AR	
	Package Management	AK	
	Delete	AD	
	View Delete Log		
	Reload Plugins		
	Database Information		
Reports			
	All Reports	RR	R
	Individual Personalized Reports/Graphs		1-9 (Top 9 personalized reports/graphs)
	Graphs & Charts	RG	
Help			
	Screen Help	HS	H
	Help	HH	
	About	HA	

Chapter

5

5 User Management

This chapter describes how to setup SpectrumSCM user ids and passwords, define user roles, and assign users to projects with their proper roles for that project. Users will also learn how to customize preferences with respect to screen look and feel, fonts, and editors.

NOTE: There are system level ids that provide users access to the SpectrumSCM system. There are project-level roles assigned to each user that give him or her specific permissions and responsibilities for each project. It is important to understand how system-level and project-level permissions interact. System-level permissions take precedence.

5.1 System-level roles

There are three distinct system-wide levels of permission and authority:

- **Administrator** - The overseer of the SpectrumSCM system for an application or a set of applications. This task might fall to a server administrator or a specific person within the project team. An administrator can perform all functions within the SpectrumSCM system. Only an Administrator can stop and start an SCM server and make changes to the server configuration. An administrator adds new users to the SCM system and assigns project engineer permissions. Specific users who will perform the Administrator role can also be assigned administrator level of authority via the User Administration Screen.

Administrator authority is very powerful. Consider carefully to whom this role will be assigned. Some installations allow only system administrators to be SpectrumSCM administrators. Others give administrator access to project engineers or even generic engineers to allow them to control and manage adding new users to the system, stopping and starting servers, etc. This is reasonable if there is only one project using a SpectrumSCM server. It can be problematic if multiple projects share an SCM server instance. There are pros and cons to each choice.

- **Project Engineer** –The Project Engineer role is responsible for creating new projects, for assigning the generic engineering role for that project, for assigning people to work on that project and for assigning roles to those people. The PE is the only login with the power to delete a project. The role of PE is a system-level permission, assigned by the administrator when the user is added to the system. This role can be handled by a Senior Developer, Project Leader, Project Manager, or the Technical Manager of a project.

The Project Engineer is responsible for defining how the project team will use the SCM system to manage its work. The project engineer defines and assigns the project's life cycle phases, as well as the project's CR attributes and values. While these tasks can be significant, they would only be performed once at the start-up of a new project. SpectrumSCM provides the flexibility to define and associate different life cycles and CR attributes for different projects.

- **User:** The user role is the most common user role model. A user has limited access to the system; a user has no administrative permissions other than the ability to change his or her password. A user can create, assign and work CRs, add and modify new source files and use all the regular features of the SCM tool that do not fall into an administrative category. The user role can be configured differently per user based on the assignment of project-specific roles.

In addition to the system-wide levels of authority, user roles are defined more finely at the project level. This allows a user to be assigned different permissions within a project depending on his or her role on a project team. For example, a user may be a developer on one project and a generic engineer on another project.

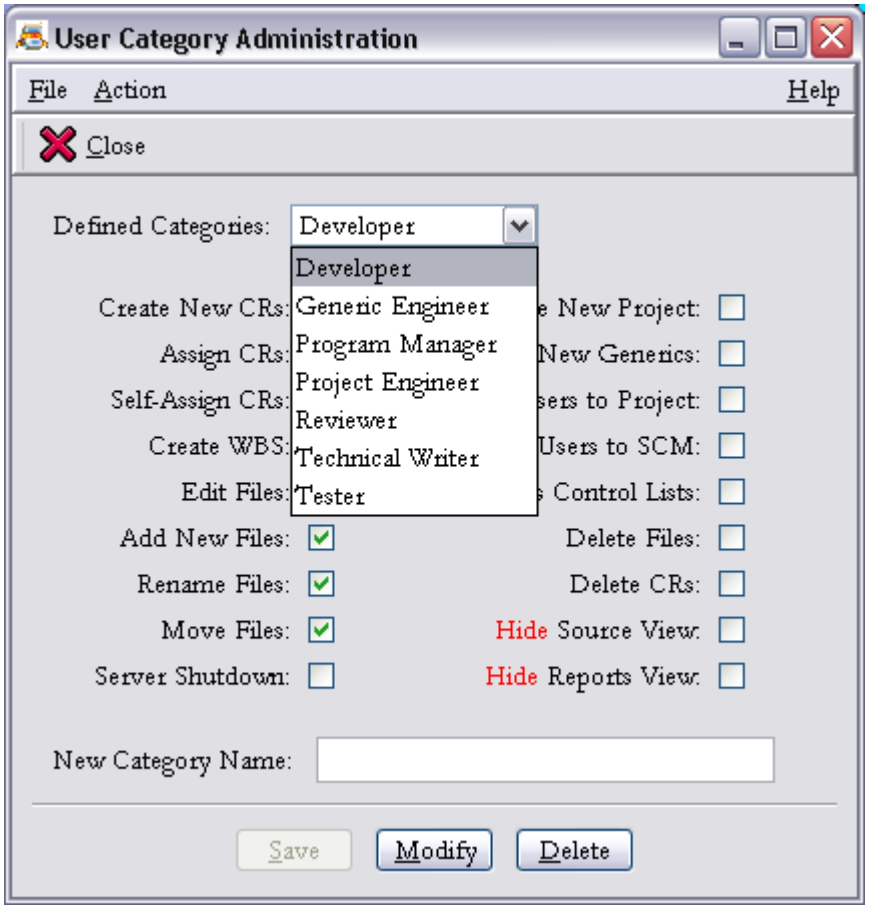
5.2 Project-level Roles

Except for system administrators, users of SpectrumSCM are members of project teams. Even the simplest project team has specific roles, for example, team leader, developer(s), and tester(s). Each member of the team performs the functions of one or more roles.
SpectrumSCM provides basic pre-defined roles and allows project leaders to define additional roles as needed. A role has specific authorities and permissions pre-defined within the context of the SCM system. A role can be assigned to a distinct individual, an individual can be assigned multiple roles, or multiple individuals can be assigned the same role (developer for example).
The Administrator and Project Engineer roles are system-wide roles that are assigned to users via the User Administration screen. Either or both roles can be assigned to a user when the user is added to the system or later via the Administration / User Administration menu options, using the Add User or Modify User buttons on the User Administration screen
Three pre-defined project-level roles are:

- **Developer -** The developer reviews CRs assigned to him or her, checks-out or creates the source files needed to address each CR, makes changes or performs any other work required to completing the CR, checks in source that is modified or added, progresses the CRs to the next state and awaits CR assignments.
- **Generic Engineer** - The day-to-day project leader or supervisor responsible for assigning work-items (CRs) to those within the project team. The Generic Engineer is responsible for creating and managing the project's generics (branches) and forming releases.
- **Tester** – The tester has the ability to create CRs when problems are found during testing. He/she does not have the ability to assign work or modify source.

NOTE: A project team leader is free to select or create the roles that will be used by his/her project, but note that all roles are defined at a system level. *Be aware that permission changes for a role will affect all projects using that instance of the SpectrumSCM server.*

It is recommended that the permissions for predefined roles NOT be changed. If a project wishes to makes changes to permissions for developer or tester, for example, the project can define a project-specific role ("GENESIS Developer", for example) and add or delete permissions. Roles and their permissions can be viewed via the Administration / User Categories / Category Administration screen:



Remember, all roles are available system-wide and can be used by any project sharing the SpectrumSCM server instance. Changes to a role will affect all projects using that system role.

When setting up a project, carefully consider the process your project team will follow and define roles and permissions accordingly.

For example, a project team might include one or more of the following -

- A **Project Manager** would generally have Project Engineer authority set at the system level.
- A **Generic Engineer** would have similar permissions as a project engineer but perhaps without the *create project* or *add new users to a project* capabilities.
- A **Senior Developer** might have all development permissions and the capability to assign work.
- A regular **Developer** might only have permissions to create CRs, check-out files and add new-files.
- A **Tester** or **Customer** might only have the capability to create new CRs.

Example Project

In our example Genesis project, project team roles have been assigned as follow:

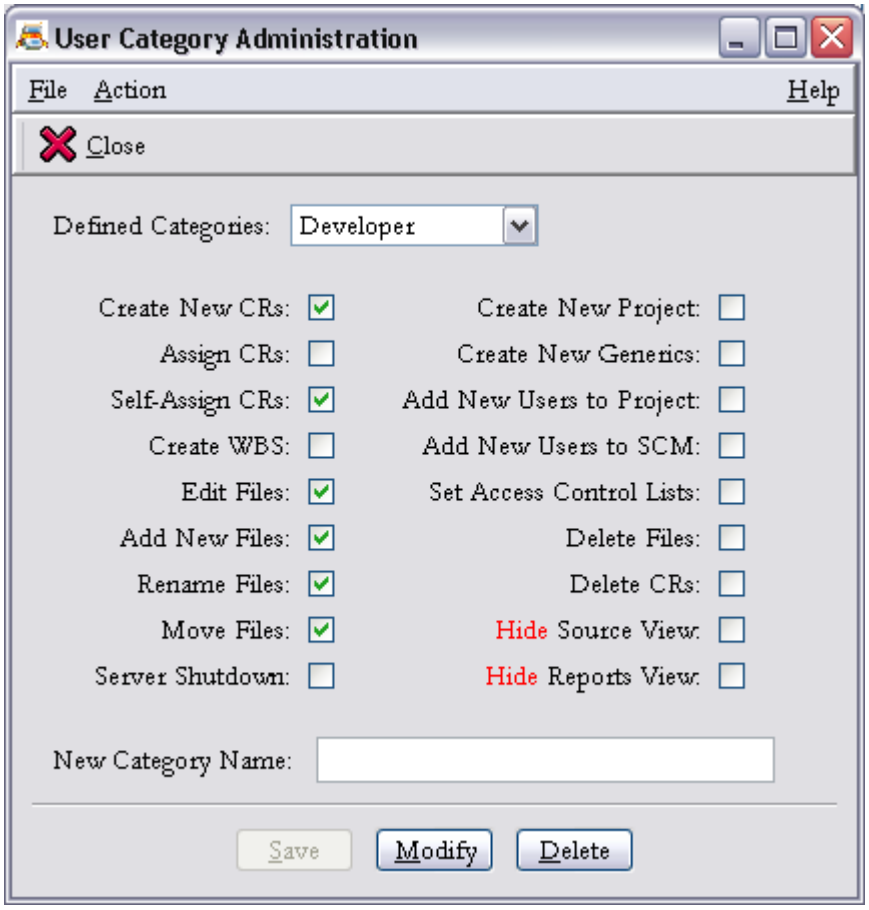
- Project Manager – Lisa
- Generic Engineer – Gene
- Senior Developer - Corey
- Developers – Debbie, Sundar
- Tester - Rich
- Customer – Patma
- Supervisor – Srini

We will follow their progress in our sample screen shots as the Genesis project is set up and they develop and release generic 1.0.

5.3 Defining project-level user categories and access permissions

User categories or "roles" are defined via the User Categories Screen. Roles for a project must be defined before users can be assigned to a role.

New roles can be added by an Administrator or Project Engineer. Roles can be added via the Administration - User Categories / Category Administration screen.

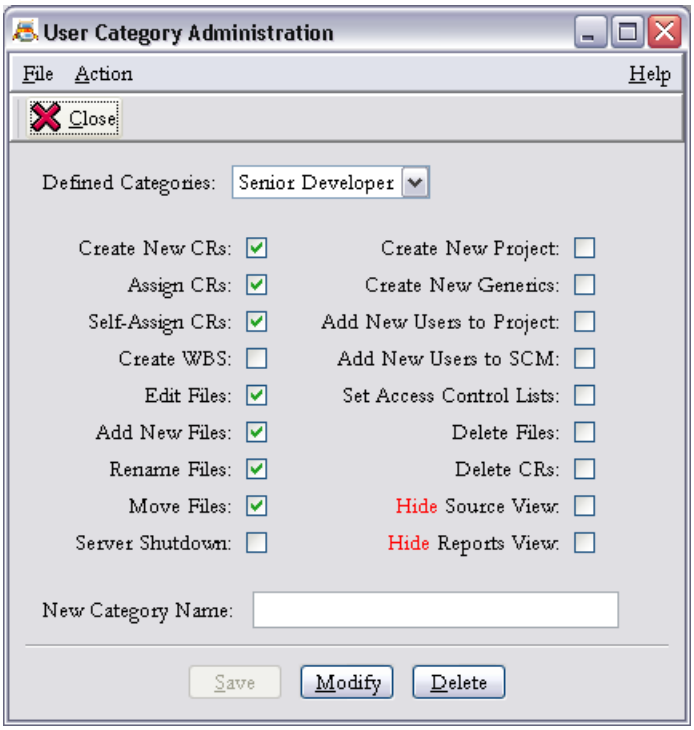


For example, to add the new role "Supervisor":

1) Define the role in project terms
Supervisor – Tracks Issues that are associated with his/her projects specifically by looking at the Reports. The supervisor may also create CRs.

2) Enter the new role in the 'New Category Name' field, select the appropriate permissions, and click Save to add the new role to the system.

When adding the role "Senior Developer", the project engineer might include all permissions granted to developers plus the ability to Assign CRs and Self-Assign CRs.



5.4 Setting up users in SpectrumSCM

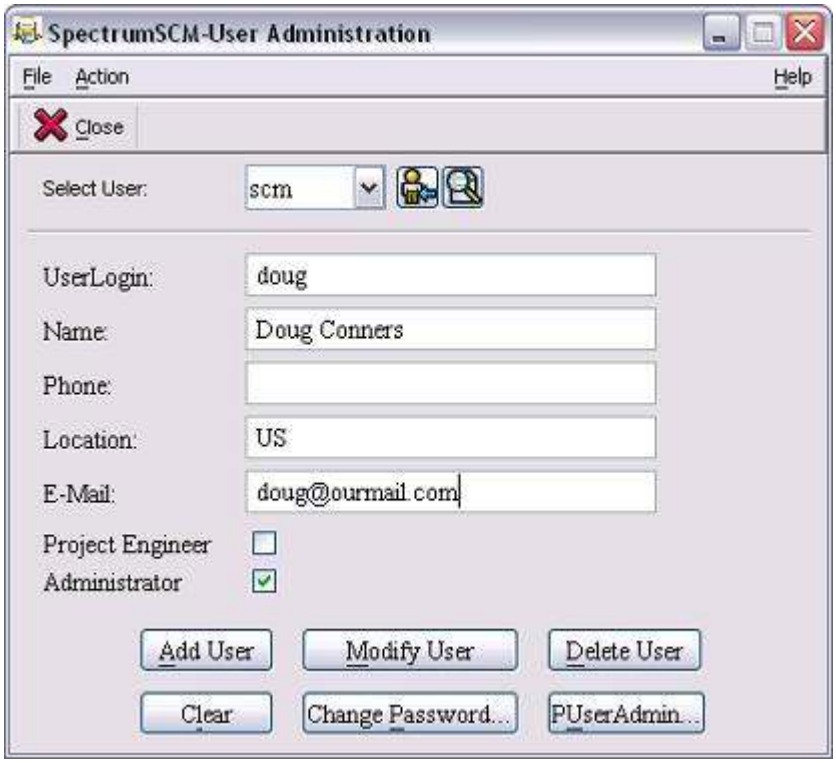
Users must be added to the SpectrumSCM system before they can be assigned to any project running on that instance of the SCM Server. User information can be added, deleted or modified here.

5.5 User Administration screen

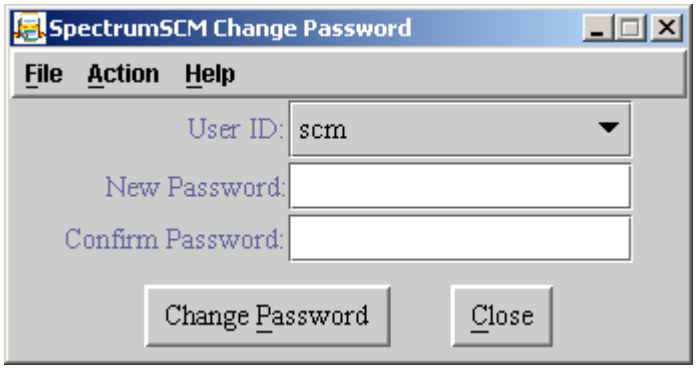
Users are added to the SpectrumSCM system via the User Administration screen accessible via the Administration / User Admin menu options.

User information can also be modified and users deleted via this screen.
E-mail information is required if the project will use e-mail notifications. E-mail notifications occur when a CR is assigned to that user or, if the user is a Generic Engineer, when a CR has been progressed.

Administrator and Project Engineer permissions are granted or revoked here because these permissions levels are project independent.



This icon will bring up the screen for a user to change his/her password or an Administrator can reset any user's password.



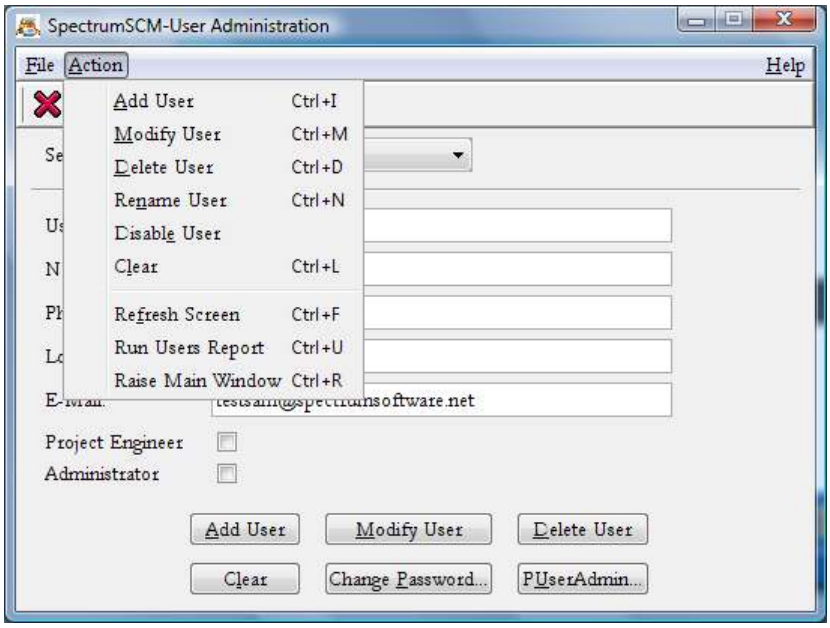
This icon will bring up the Project-User Administration screen used to assign users to projects.

NOTE: All members of a project team must be added to the SpectrumSCM system and User Categories and Access Permissions must be defined before users can be assigned to a project.

5.6 Renaming Users and Disabling Users

New

You can rename an existing user-id while maintaining full tracability of all the CRs that would transition from the old user-id to the new one.



5.7 Disabling Users

New

You can disable a user so that their full identification is maintained but yet they would not be able to login to the SpectrumSCM system. You can re-enable this user at a later date if you choose do so.

5.8 Project-User Administration

When a new project is started or a new user joins the project, each person should be added to the SpectrumSCM system and then assigned to a role in the project team.

Assigning roles is done via the Project User Admin screen, which can be accessed from the Administration / Project User Admin menu selection from the main screen or via the PUserAdmin button on the User Administration screen.

On the Project-User Administration screen, notice that all users currently assigned to the project are listed via the pull-down User menu area on the right. All user roles defined to the system are displayed to the right.

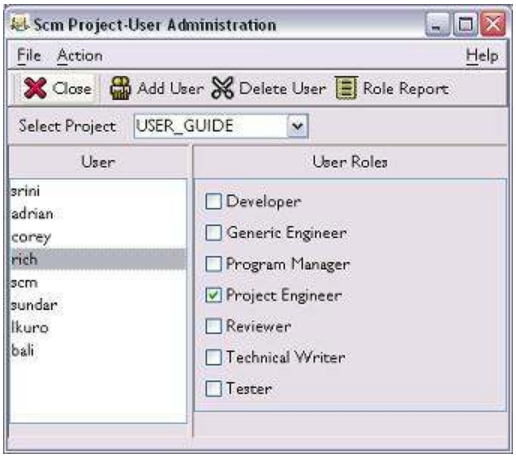
5.8.1 To assign a user to a project:

Select the project.
Select the user to be assigned to the project (each user defined in the system can be accessed via the Add User popup dialog.

Select the user's role(s) within that project team.

Roles assigned to a user can be added or deleted by selecting the user in left "User" window, which will display the user's current role(s). Make the changes in the right window and select
Action / Save PUser from the menu bar

5.8.2 To delete a user from a project

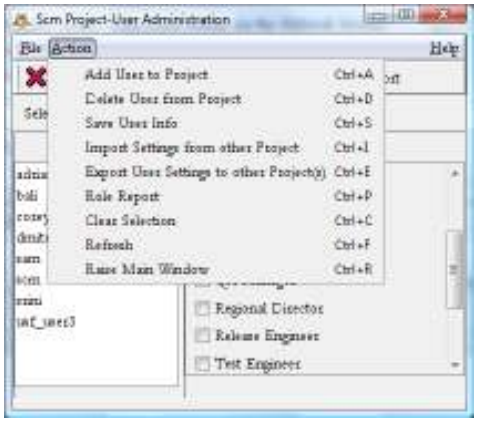


Select the project and the user.

Use the  button or Action / Delete PUser menu option to delete. This will only delete the user from the project team, not from the SpectrumSCM system. For example, to remove user rich from the USER_GUIDE project team, select "rich" in the User window

and use the  icon or Action / Delete PUser to delete him from the project team.
"rich" will no longer be a member of the USER_GUIDE project team, but he is still defined as a user in the system.

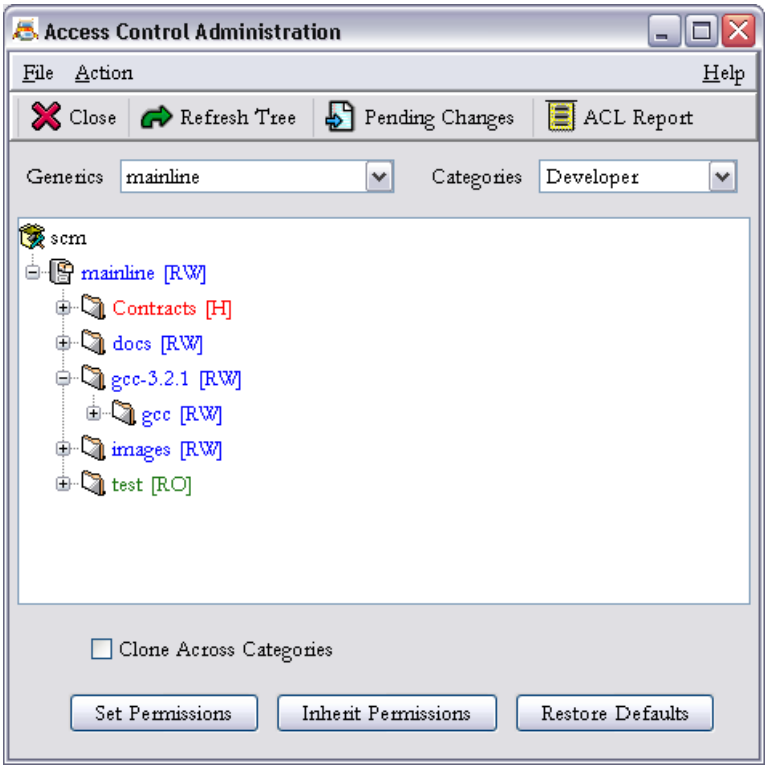
5.8.3 To clone user settings from a project / To add a set of users to other projects



Use Action / Import Settings from other project menu option to clone user settings from an existing project and Action / Export User Settings to other projects menu option to add a set of users with current user settings into other projects. The above two menu options enable projects to inherit the users with their roles and permission settings from other projects. It is very useful particularly when the the team size is large or the new team member (or members) joins a significant number of projects.

5.9 Role Based Access Control Lists (ACLs)

SpectrumSCM implements a Role Based Access Control (RBAC) model which allows project managers to enable access permissions on all configurable items in the project repository (branches, directories and files) based on the role that a user plays in a project. To launch the Access Control Administration screen, select the **Administration → Access Control Admin** option from the main menu.



The Access Control Admin screen presents three views of the access permissions for all the resources in the repository tree for a generic. The Categories combobox indicates the selected user category. The color coding represents the end user view of the access permissions. **RED** indicates that the resource is hidden, **GREEN** indicates that the resource is read-only and **BLUE** indicates that the resource has read-write permission.

The information within square brackets [] represents the database view of the access permissions for the resources in the tree. **[H]** indicates that the resource has been marked hidden in the database, **[RO]** indicates that it is read only and **[RW]** indicates that the resource has been marked read-write. Notice that a resource can be marked read-write in the database but can be read-only or hidden in the end user view if its parent resource has a lower permission.

A ***** besides an item indicates that it is a pending change that needs to be saved. The list of all pending changes can be obtained by clicking on the **Pending Changes** button.

The access permission for a resource can be changed by clicking on the resource in the tree view. The user view immediately changes to reflect the new permission and the resource is marked with a * to indicate that the change is pending. Subsequent clicks change the permissions in a cyclic order based upon the permissions that are currently available for the resource. Once the permissions have been changed for the resources, the pending changes can be saved by clicking on the **Set Permissions** button. If the Clone Across Categories has been selected, a dialog is presented prompting the user to choose the categories across which the access rules need to be cloned. This option can be used to clone the new permissions across different user roles with a single click.

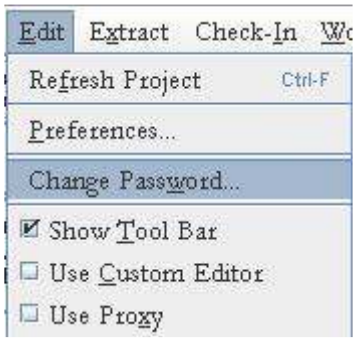
The **Inherit Permissions** button can be used to inherit the permissions that have been defined for another user category. A dialog is presented prompting the user to select a category for importing the access rules. Combined with the Clone Across Categories option, this feature can be used to easily create hierarchical permission models.

The **Restore Defaults** button can be used to reset the permissions for all resources in the tree to the default value (Read-Write). The reset operation is specific to the selected category and the Clone Across Categories option can be used to reset the permissions for multiple categories at a time.

The **Refresh Tree** button can be used to load the tree for another generic once a new generic has been selected in the main screen project bar. The Categories combobox can then be used to load the permissions for the selected category. The **ACL Report** displays the current access permissions defined for all configurable items under a particular generic and for a particular user category.

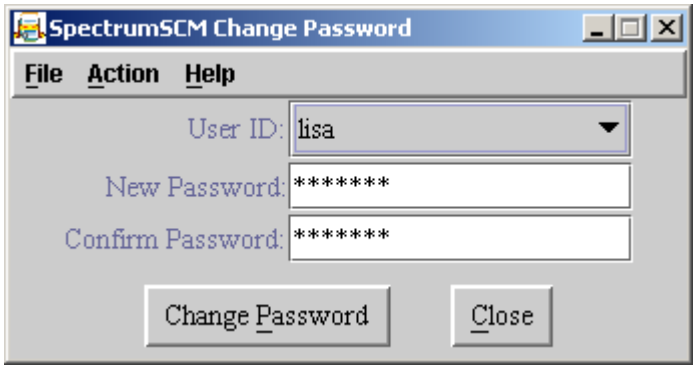
5.10 User Password Administration

All users are added with the initial password "default". This should be changed by the user as soon as possible. A user can change his or her own password or the SpectrumSCM administrator can change any user's password (for example, a forgotten password).



Access the password change screen from the EDIT selection on the main menu or via the button on the User Admin screen.

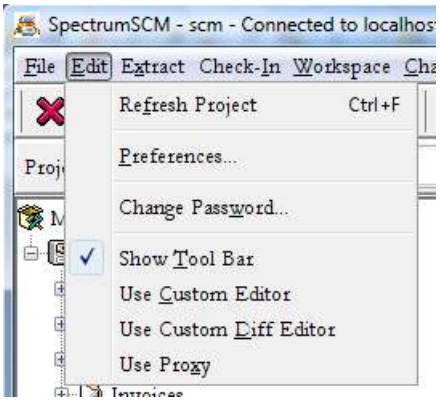
Either will bring up the Change Password screen for a user to change his or her password or an Administrator to reset any user's password.



Select the user and enter the new password twice, then click Change Password to confirm the change. Close the window when finished.

5.11 Setting User Preferences

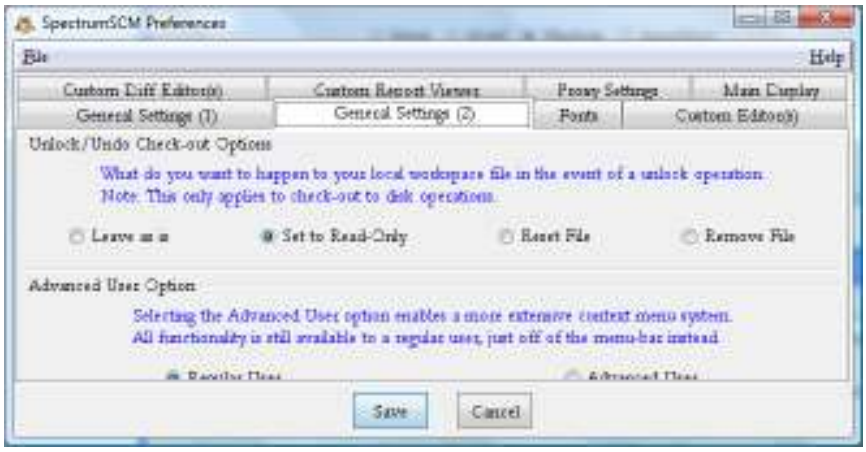
User can set their specific preferences by selecting *Preferences* option from the Edit menu item on the main screen as shown below.



Each user can set individual preferences for -

- **Main Toolbar** - Whether or not to show the toolbar..
- **Look & Feel** - Metal (the Java default), Motif, Windows, and Aqua (for the Apple Mac).

- **Merge Editor preferences –** Whether you prefer to use the 2-way or the 3-way merge editor.
- **General Settings (2)** - Unlock preferences & Advanced User Setting.
- **Unlock preferences** – How do you want an unlock from disk operation to be completed in terms of the file under your local root directory.
- **Advanced User Setting** – Alters the options available in the main tree context menu system.
- **Fonts** - To select a font of your choice.
- **Custom Editor** - Whether to use the provided SpectrumSCM editor or whether you prefer your own..
- **Custom Diff Editor** - Whether to use the provided SpectrumSCM diff editor or whether you prefer your own.. You can use this feature to define the use of any third party diff editors including binary diff tools. See screen help for specific examples. New
- **Custom Report Viewer** - Whether to use the SpectrumSCM HTML viewer or your own
- **Proxy Settings** – The IP address and port for the SpectrumSCM proxy



Unlock Options –

a) **Leave as is** – The disk file will be left writeable and with its file contents unchanged by the unlock operation.

b) **Read-Only** – The disk file will be set to read-only but the file contents will be left unchanged by the unlock operation.

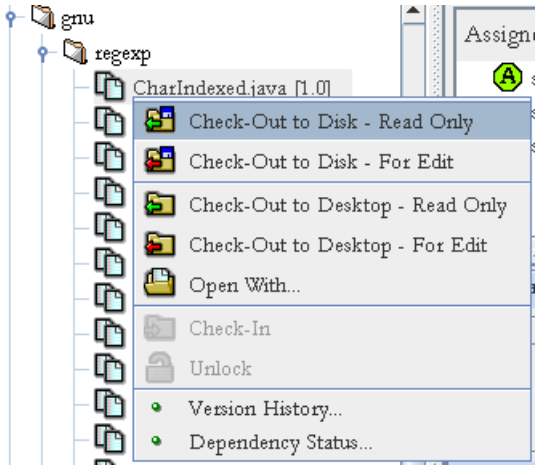
c) **Reset** – The disk file will be reset to the current head version from the repository, this includes making the file read-only.

d) **Remove** – The disk file will be deleted.

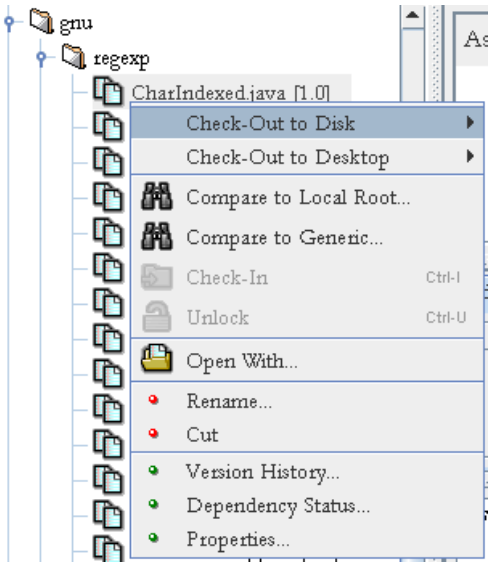
Advanced User Option –

Under the default "Regular User" option, the main screen repository tree will have a short, direct context menu. This gives direct access to the needed CM functions. Under the "Advanced User" option more choices are provided on the popup menu such as Local Root comparison and Cut and Rename operations. Under the regular option these advanced functions are still available under the menu bar.

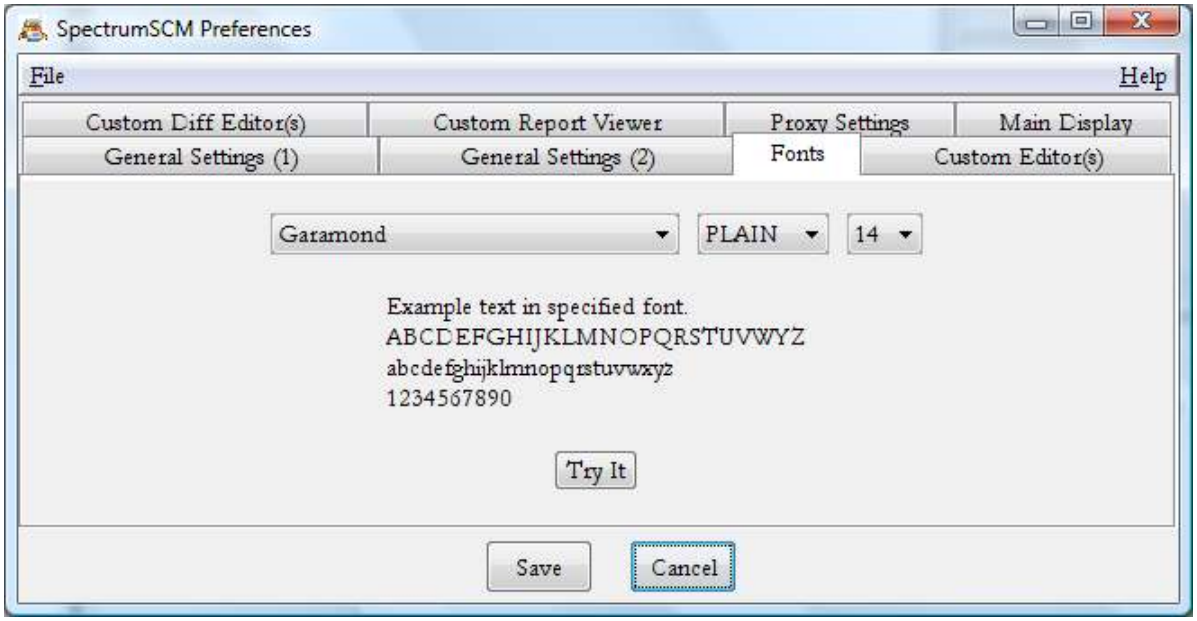
File Context Menu - Regular Option



File Context Menu - Advanced Option

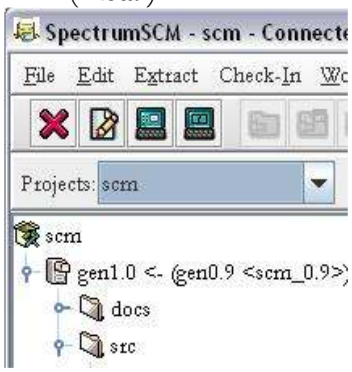


Fonts – Select from the system defined fonts. Font families and their sizes are OS dependent. The Fonts preferences panel will display all fonts installed on the client OS.

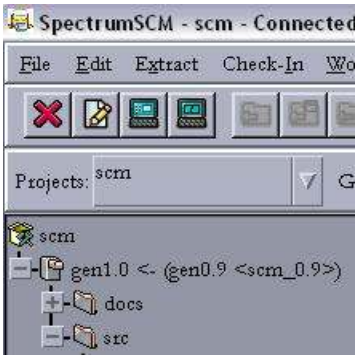


Look & Feel - Metal (the Java default), Motif, Windows, and Aqua(Mac).

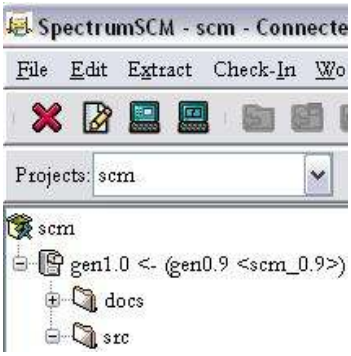
(Metal)



(Motif)



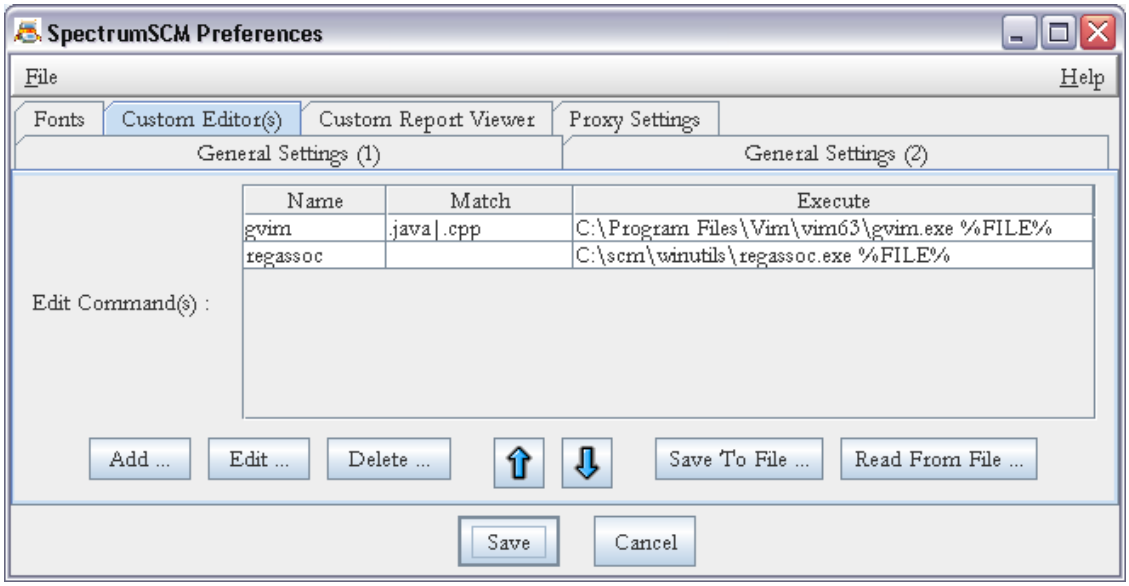
(Windows)



Main Toolbar – Sets whether or not the main toolbar is displayed. On the SpectrumSCM Main Screen.

Custom Editor – The default SpectrumSCM editor can be used in any environment, but only for text files. The Custom Editor option allows the user to specify other preferred editors. For example, to use MS Word, enter as the edit command:

```
C:\Program Files\Microsoft Office\Office\WINWORD.EXE %FILE%
```



If the project involves multiple file types supported by the Microsoft Windows registry, the "regassoc" command will automatically open the file using the native application based on the on the file suffix (for example, opening a .XLS file will launch MS Excel). This option is only available in the Windows environment. Enter as the edit command:

<client install directory>\winutils\regassoc.exe %FILE%

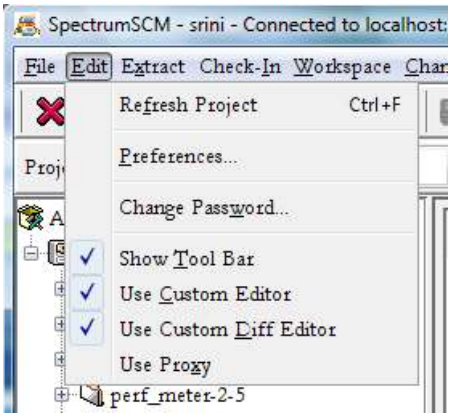
The custom editor in Linux can be set to the editor of choice and the desktop interface of choice for the user. The user can also choose whether to edit in a window or use an editor with an integrated window.

For example, in RedHat using the Gnome desktop and the gvim editor, the edit command should be set to:

/usr/X11R6/bin/gvim %FILE%

To use standard vi in an xterm session:

/usr/X11R6/bin/xterm -e vi %FILE%



Though multiple custom editors can be specified, on a plain 'open file' request only the top-most matching entry will be used. The order of entries can be adjusted by using the arrow buttons.

The match field is used to specify which types of files this entry can be used for. A blank match field indicates that it can be used for any file type. Multiple file types can be matched with a single custom editor entry by using the pipe symbol separator ("|") as shown above.

The Save To File option allows the user to save their current settings to the local disk. This is so that the settings can be shared with other users.

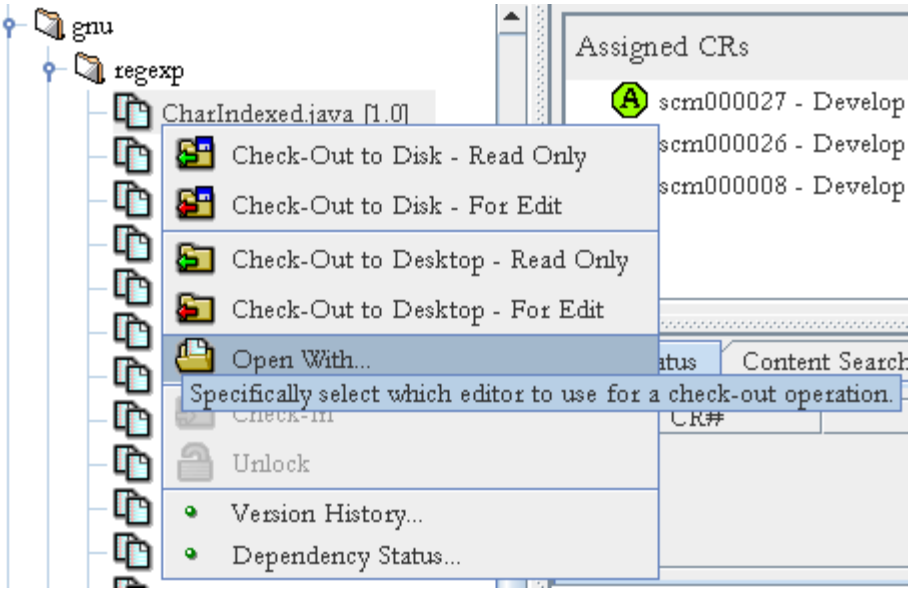
The Read From File option reads a previously saved file and merges the entries into the current settings.

Enable Custom Editor

Once set, use of the custom editor can be disabled/re-enabled.
This is done by check box "Use Custom Editor" via the Main Screen Edit menu option.
To use the custom editor, select "Use Custom Editor".
To use the SpectrumSCM editor, deselect "Use Custom Editor".

Open With

If a file is selected in the main screen file tree with a mouse right-click, the context sensitive menu will appear. The "Open With" option works with your custom editor selections and allows you to choose between them. For example if you have a couple of appropriate editors defined for a particular file type (HTML to be viewed through Internet Explorer or Netscape).



With Matching executables selected (the default), each of the custom editors that match the file type you have selected will be displayed for selection. If the file is textual the SpectrumSCM default editor will also be displayed.

With All executables, all of the custom editor entries and the SpectrumSCM default editor will be displayed.

Select the editor you wish to use and then the specific edit operation you wish to perform from the buttons on the right.

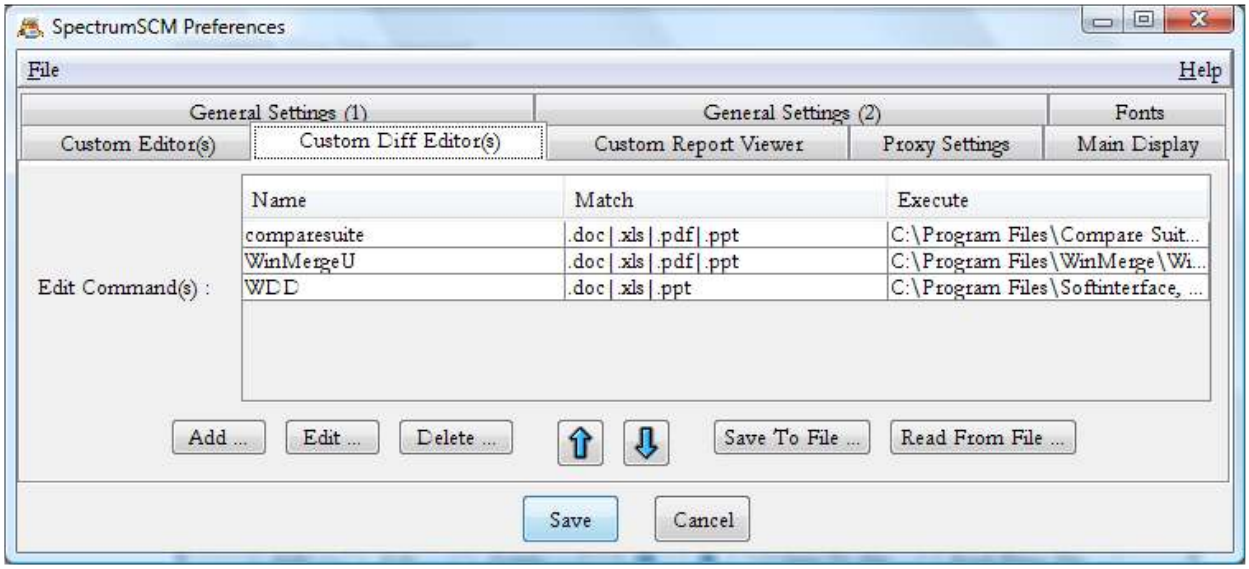
Custom Diff Editor – The default SpectrumSCM editor can be used in any environment, but only for text files. The Custom Diff Editor option allows the user to specify other third party differencing/comparison editors.

New

For example, to use WinMerge, an open source diff merge editor for comparing .doc,.xls,.pdf, .ppt etc. , enter the following command in your diff editor settings:,

C:\Program Files\WinMerge\WinMerge.exe "%FILE1%" "%FILE2%"

NOTE: if use "…." Button to select the WinMerge.exe above the FILE1 / FILE2 text will be included automatically.



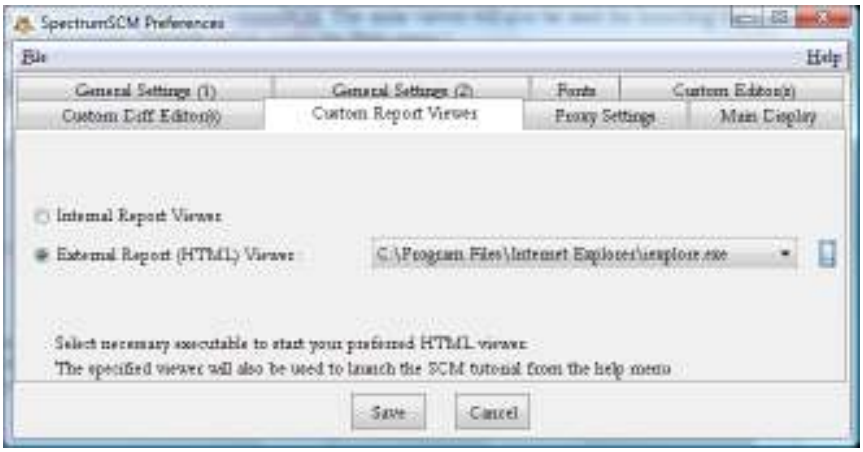
See screen help for more specific examples.

Enable Custom Diff Editor

Once set, use of the custom diff editor can be disabled/re-enabled..
This is done by check box "Use Custom Diff Editor" via the Main Screen Edit menu option.
To use the custom diff editor, select "Use Custom Diff Editor".
To use the SpectrumSCM diff editor, deselect "Use Custom Diff Editor".

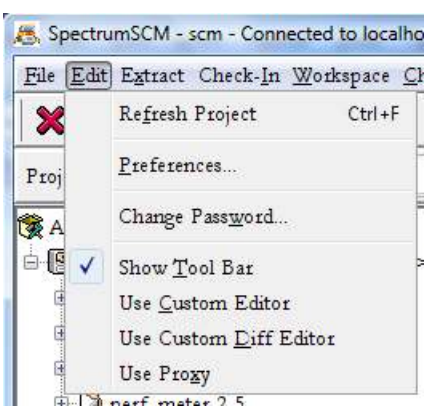
5.11.1 Custom Report Viewer

The custom report viewer allows you to specify a preferred HTML viewer like Netscape or Internet Explorer for viewing reports in SpectrumSCM. The same viewer will also be used for launching the SpectrumSCM tutorial (Tutorial option under the Help menu.). Furthermore, the most recently used 5 custom report viewers are maintained and selectable to help if you are in an environment where you need to switch between different viewers.



5.8.3 Proxy Settings

The SpectrumSCM Proxy provides enhanced performance for distributed development teams in bandwidth constrained network topologies. The Proxy Settings panel allows users to specify the IP Address and Port Number for the proxy server to be used. The **Use Proxy** option under the Edit menu can be used to enable or disable proxy usage.



See *Chapter 12 – Administrative Functions* for details on what the Proxy feature is, its benefits and how to install, configure and use it.

Chapter

6

6 Process Management

This chapter describes how to set up the project environment to support your development process, including setting up a project under SpectrumSCM, establishing the project life cycle, and creating a generic. It also describes how to set up the project directory structure and load existing files and file structures into the SpectrumSCM system.

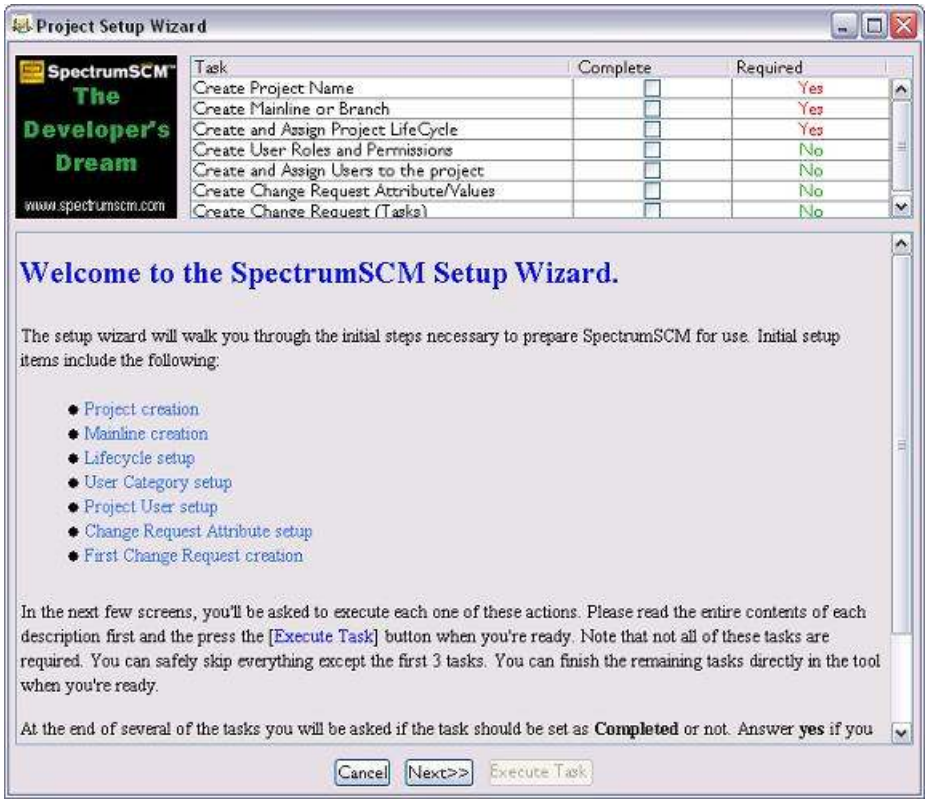
SpectrumSCM Process Management

- Ensures that components are progressed through chosen life cycle phases before being released. For example, to verify that testing and quality assurance occur before a component is cleared for release.
- Allows for total customization of your process at the project level.
- Allows the use of a process that suits the culture of your organization or a new process that you wish to introduce
- Change Management is integrated tightly into the entire product life cycle.

Work through this chapter and follow the steps to set up a sample project. SpectrumSCM provides complete flexibility to map your current process or customize the system to handle specific project and process needs.

6.1 Setting up a New Project under SpectrumSCM

To add a new project to SpectrumSCM, click on the **Create Project Wizard** menu item under the **Administration** menu on the SpectrumSCM main screen to access the **Project Creation Wizard.**

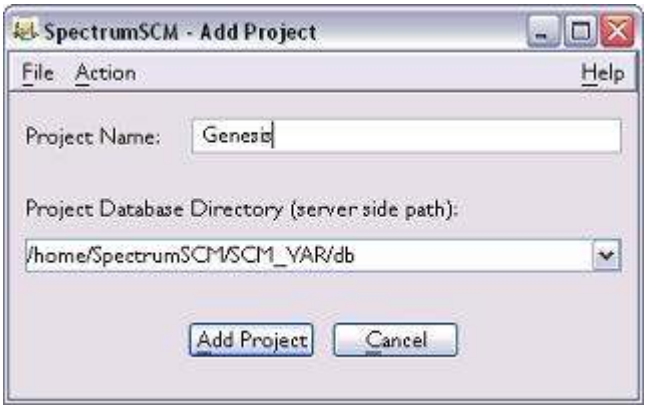


The Project Creation Wizard is an aid to creating new projects in SpectrumSCM. The wizard guides the user through all of the steps necessary to create a new project, set up a new branch, and all of the other steps required to get the project ready for first use. The Project Creation Wizard screen is broken up into two separate sections. The table area at the top contains the list of activities that the user will execute in order to setup the new project. Note that some of the activities are required and some are not. The steps that are not required can be skipped during execution. The bottom portion of the screen is a text area that describes the step that the user is about to execute. It also provides screen snapshots of each individual screen that will be presented during execution. At the bottom of the screen are three buttons, **Cancel**, **Next>>** and **Execute Task**. These buttons allow the user to navigate through the tasks and to execute them when instructed. The user can press the **Next>>** button at any time to move to the next task. This allows the user to read through all of the instructions first, and then return to actually execute the tasks.

As tasks are completed, a check mark is added to the table entry for the completed task. Tasks that have been completed cannot be executed again. Tasks that have not been completed can be revisited and executed at any time.

6.1.1 Project Creation

In this step the user will be prompted to create a new project. The text area at the bottom of the screen will contain information about the project creation screen itself and provides instructions concerning the project creation screen.



In the project creation screen the user must enter the name of the project that they want to create, and the server side path to the database directory. The database directory path will always default to the SCM_VAR/db directory, located under the server side installation directory.

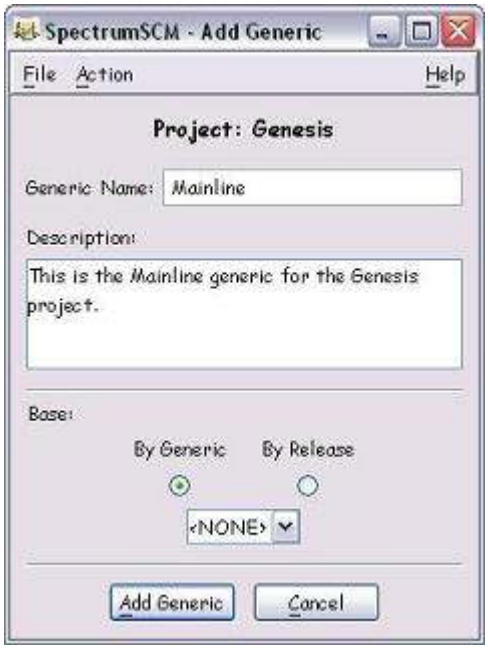
NOTE – The directory entered into the Project Database Directory path needs to be backed up on a regular basis. If a backup strategy is already in place for this server, make sure to extend it to include this new directory. If no backup strategy is effect for this server, make sure to create one right away and start backing up the files located under this directory on a nightly basis.

Once the project has been successfully created, the user will be presented with a confirmation dialog indicating that the project has been created.

6.1.2 Mainline Creation

Projects can contain many generics (branches). Upon creation, a new project does not contain a generic. Before work can begin on the project, a new generic must be created. The Generic (branch) creation screen prompts the user to enter a name for the new generic as well as a short description of what the generic should be used for. For new projects the radio buttons at the bottom "By Generic" and "By Release" have no meaning, since there are no other generics or releases for this project at this time. Enter a name for the new generic and a short description and then press the "Add Generic" button.

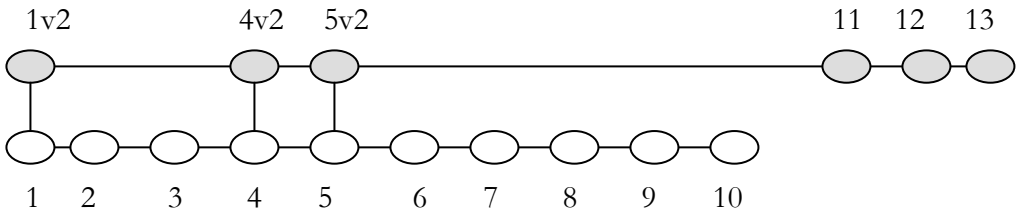
Once the new generic has been created, the system will respond with the Modify Generic screen.



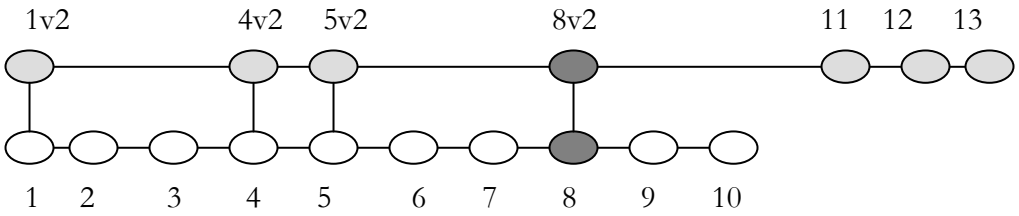
This screen allows the user to set the check out preferences for the buttons on the main screen and to mark the generic as "Locked" or not. Locking a generic fixates all of the files in the new generic to their current version number. Edits done on other generics, against files that are common with the locked generic, will not affect the files on the locked generic. The files will be automatically specialized (uncommoned) into the locked generic thus preserving the content of the files on the locked generic. The locking mechanism is most useful when a generic is created from a previous release. The files on the newly created generic will always contain the same content from the release that they were branched from, regardless of any actions that happen to common files on other branches. The exception to this rules comes when files are edited directly against the locked generic. Edits performed common to the locked generic are not automatically specialized into the generic.

Common Checkout vs. Uncommon Checkout – The checkout preference determines how the two checkout buttons on the main screen tool bar operate. If "Common" is set, edits always default to common and if "uncommon" is set, the default behavior is to uncommon, but the user is presented with a pop-up that prompts the user to "confirm" that an uncommon check out is really desired. The generic engineer would decide how these functions should be setup.

If a check-out for edit is performed "common" then the file changes will be made against all generics with which the file is currently common (as determined during project set-ups). Sometimes there are good reasons for this to be done, for example, fixing a problem in multiple generics.
For example, if a project team has developed and released version 1.0 of a system and they are currently developing version 2.0 (generic 2.0), all modules that are changed (modules 1, 4 and 5) or added (modules 11, 12 and 13) during the 2.0 development effort will be "uncommon" - changed only for generic 2.0.



However, if a problem is discovered in release 1.0, the fix for the problem might be made common to both generics. In this example, the problem is in file 8. The code is edited, the problem fixed and the fix is made common to both generic 1.0 and generic 2.0.



When multiple generics are to be used and developed in parallel (for example to maintain 2 similar source bases for 2 different customers), the Generic Engineer must decide who is going to make the branching decisions, who will determine which of the modules will be common or uncommon with other generics. If it is to be the Developer then the generic should be left in the *unlocked* state. If the Generic Engineer will make the decisions, the generic should be set in the *locked* state and can be unlocked on a case-by-case basis.

Once the mode of the Generic has been established, if the commonality control is still with the developer (the generic is "unlocked"), then he or she can choose the appropriate option
In overview, checking out "uncommon" will mean any file changes will only be made against that specific generic. If a check-out is performed "common" then the file changes will be made against **ALL** of the generics with which this file is currently common.

Checking out "common" is a powerful feature since it can be used to apply a single "fix" to multiple branches in one edit, however the developer would have to be careful of side-effects. *This topic will be covered in more detail in Chapter 8.*

Create a Generic from a previous Generic. A new Generic can be created as a branch from another generic or can be created from a previous release. On the Generic Creation screen select the radio button on the bottom "By Generic". Then use the combo box to select the generic to branch from. When the new generic is created all of the files on the branched generic will become common with the new generic.

Individual specializations may then be applied by checking files out as either common or uncommon to the new generic.

Create a Generic from a previous Release. A new Generic can be created as a branch from a previous release. On the Generic Creation screen select the radio button on the bottom "By Release". Then use the combo box to select the release to branch from. When a generic is created from a previous release, all of the files in the new generic will match the content of the files as they appear in the release itself. The version numbers of the files may or may not match depending on whether the files have been extended since the release was formed. Files that have been extended will be specialized into the new generic and those that have not been extended will remain common with the current branch. This gives users the opportunity to immediately extend release patches into the current release on the previous or mainline branch.

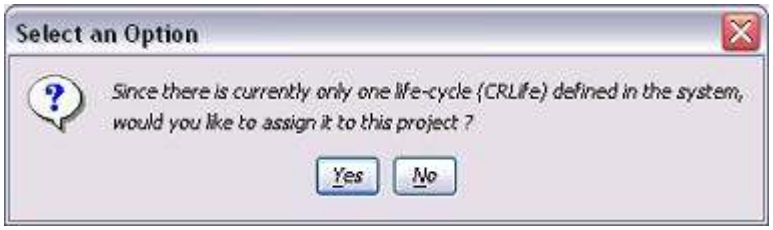
6.1.3 Life-Cycle Setup

Once the new project and its mainline generic have been created, the next step is to add a lifecycle to the project. There are two options to assigning a lifecycle to a new project. Either an existing lifecycle can be used, or a new lifecycle can be created and assigned.

If the system has any life cycles with workflow created using the SpectrumSCM LifeCycle graphical Editor , the Project Creation Wizard brings up LifeCycle/Workflow administration Editor.. Otherwise, the system brings up the linear life cycle creation screen.

6.1.3.1 Linear Life-Cycle

If only one lifecycle exists in the system at this point, the

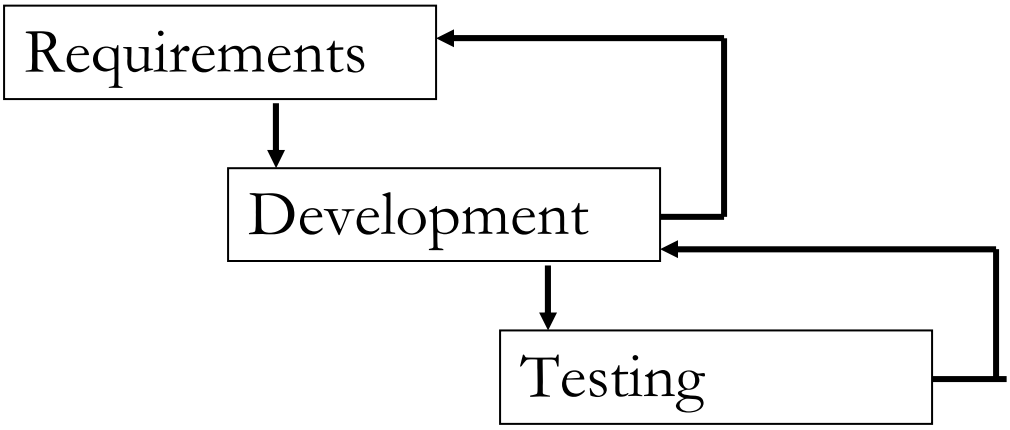


Project Creation Wizard will post the following screen to the user asking whether the system pre-defined lifecycle "CRLife" should be used or not. The user may choose to use this pre-defined lifecycle by pressing the "Yes" button, or the user may press the "No" button and the system will allow the user to create their own lifecycle and phases

A life cycle defines a set of phases, where entry and exit from each phase is well defined.

An SCM system is traditionally used to manage some form of Product Development lifecycle, for example, a software development life cycle for IT projects. Some SCM systems define and enforce a strict adherence to a pre-defined life cycle, while others can fit comfortably into an existing life cycle.

A trivial example of a software development life cycle might look like this:



In this very simple life cycle, the requirements phase is the starting phase followed by development and finally testing. The phases follow the standard waterfall model with feedback loops, which enables backtracking to an earlier phase.

More advanced life cycles are generally found in large software producing organizations where more focused phases may include several layers of testing and possibly even phases for hardware testing/profiling.

SpectrumSCM is designed to be complimentary to the process, and not an obstacle to be worked around. It is

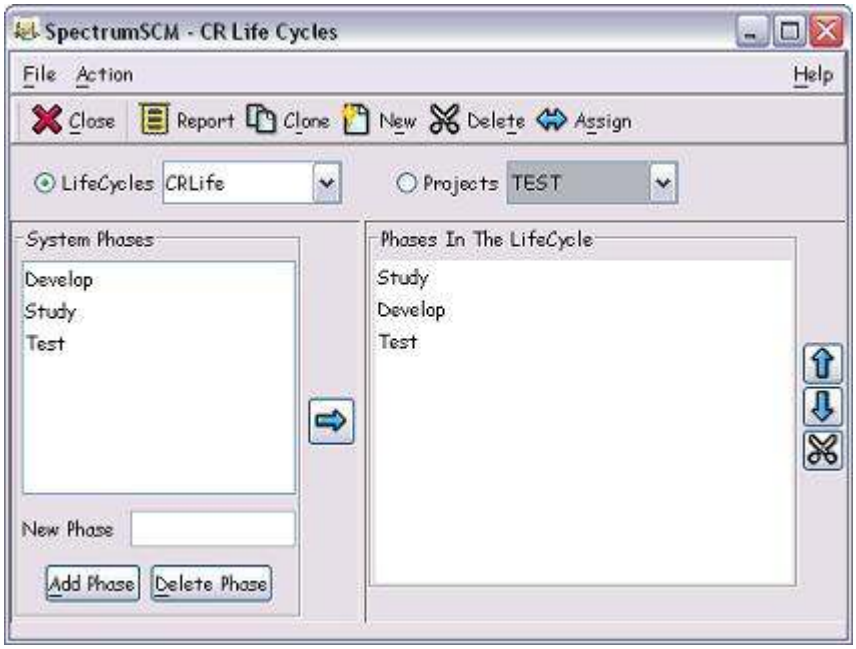
- flexible enough to work within an already established process
- able to help establish a process model where one does not already exist

For a non-software project, life cycle phases might include development, review, revision, and approval.

6.1.3.2 Creating a new life cycle

To define the life cycle phases for a project, the **CR Life Cycles** screen is accessed via the **Main Screen, Administration / CR Life Cycle Admin**:

This screen can be approached two ways depending on the choice of the **LifeCycles** or **Projects** radio button just below the tool bar.

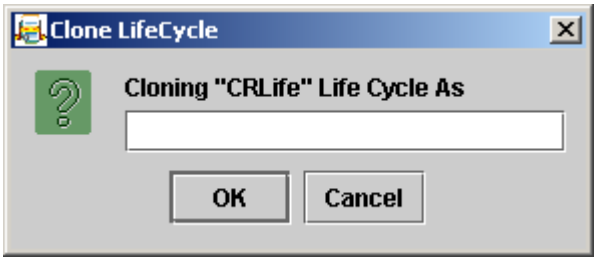


If the **LifeCycles** button is selected (the default) then the pull-down allows the user to choose from the life cycles that have been created in the system, or to create a new life cycle. The simple life cycle "CRLife" has been pre-defined. If there are any other life cycles that have been created in the system, one can be selected via the pull-down and its phases will be displayed.

Life-cycles can be administered by using the other buttons on the screen –

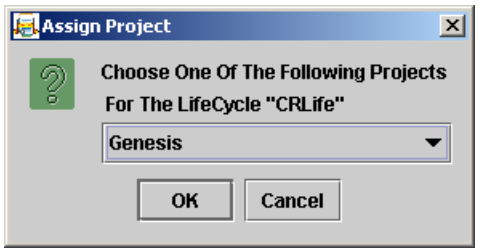
- **New** - to create a brand new life cycle from scratch.

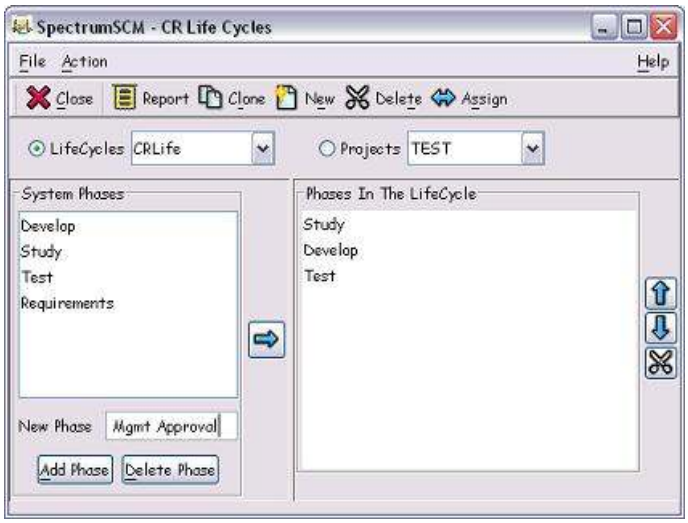
- **Clone** - to create a new life cycle from an existing life cycle.



- **Delete** - to delete a life cycle. This can only be performed if no projects are using this life cycle.
- **Add phase** - to add a phase to the list of available phases.
- **Delete phase** - to delete a phase from the list of available phases.
- **Right arrow** - to assign a phase into a life cycle.
- **Up or Down arrows** - to re-sequence a phase within the life cycle.
- **Scissors** - to delete a phase from a life cycle.

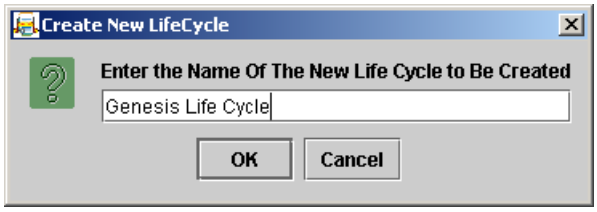
The **Assign** button is used to assign a life cycle to a particular project and can be used regardless of whether the Lifecycles or Projects radio button is selected.





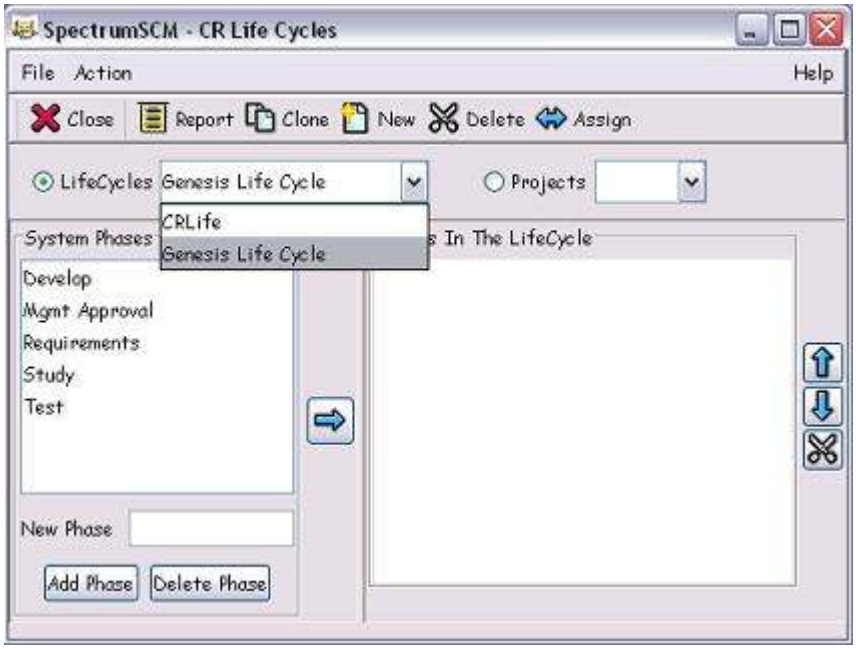
As an example, we'll set up the life cycle for the Genesis project. First, we will add two additional phases, **Requirements** and **Mgmt Approval**. To add a phase, type the name into the New Phase text field and click the **Add Phase** button, or simply hit the enter button when done typing.

Then create a new life cycle (we'll call it "Genesis Life Cycle"). Click **New** to add a new life cycle.

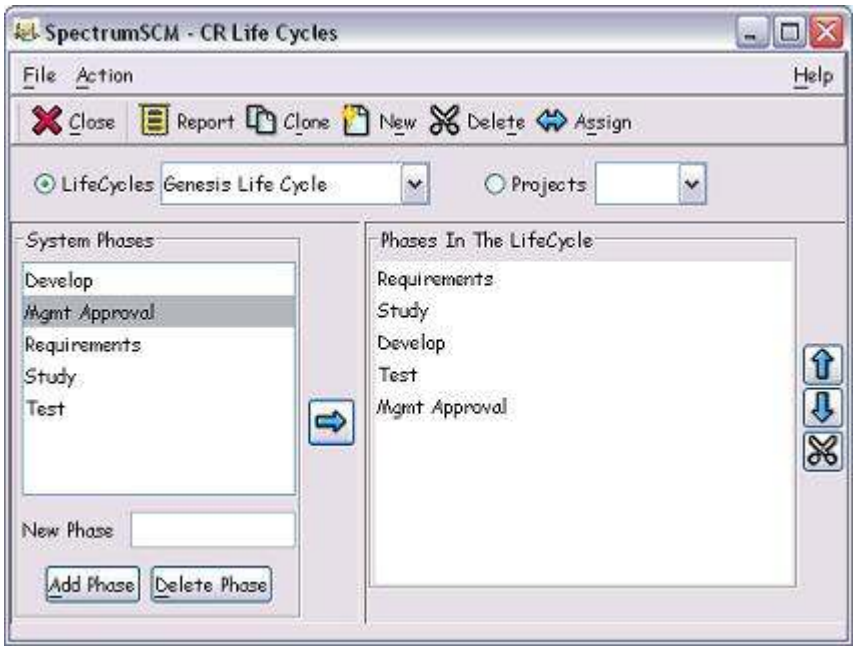


** NOTE **: Be careful modifying life cycles that already exist – they may be in use by other projects!

Select the life cycle to set up its phases.

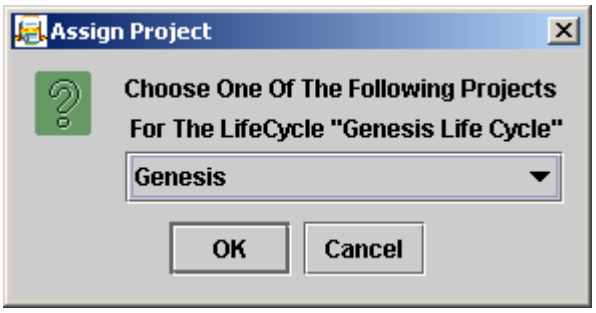


Select the phases you wish to add to this life cycle and click the  to move them into the **Phases In The LifeCycle** pane**.**



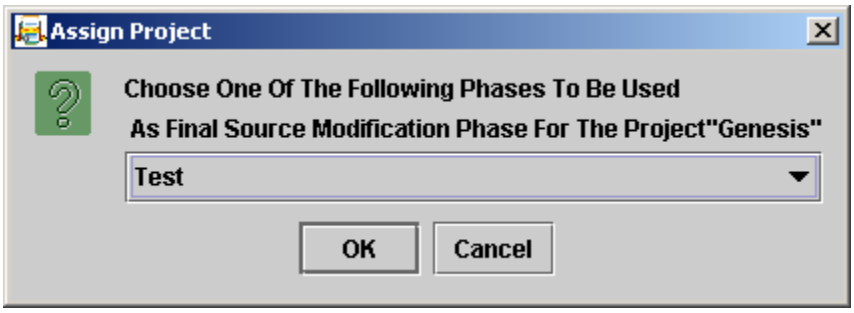
Use the  icons to order the phases and the scissors  icon to delete until you are satisfied with the phases and order.

Use the  button to bring up the Assign Project screen and assign the chosen life cycle to the Genesis project.



You will then set the **Final Source Modification Phase**, the last phase during which source can be modified. It is not the last phase of the project; it may be next-to-last, but completing this phase signifies that the component is ready for inclusion in a release. **Generally, you select the phase that you want any issue, at the minimum, to have progressed past in order to allow a release to be built.**

This is an important decision. Once a component has been progressed past this phase, it is assumed to be done, tested, and ready for inclusion in a release. It will be "green-flagged."

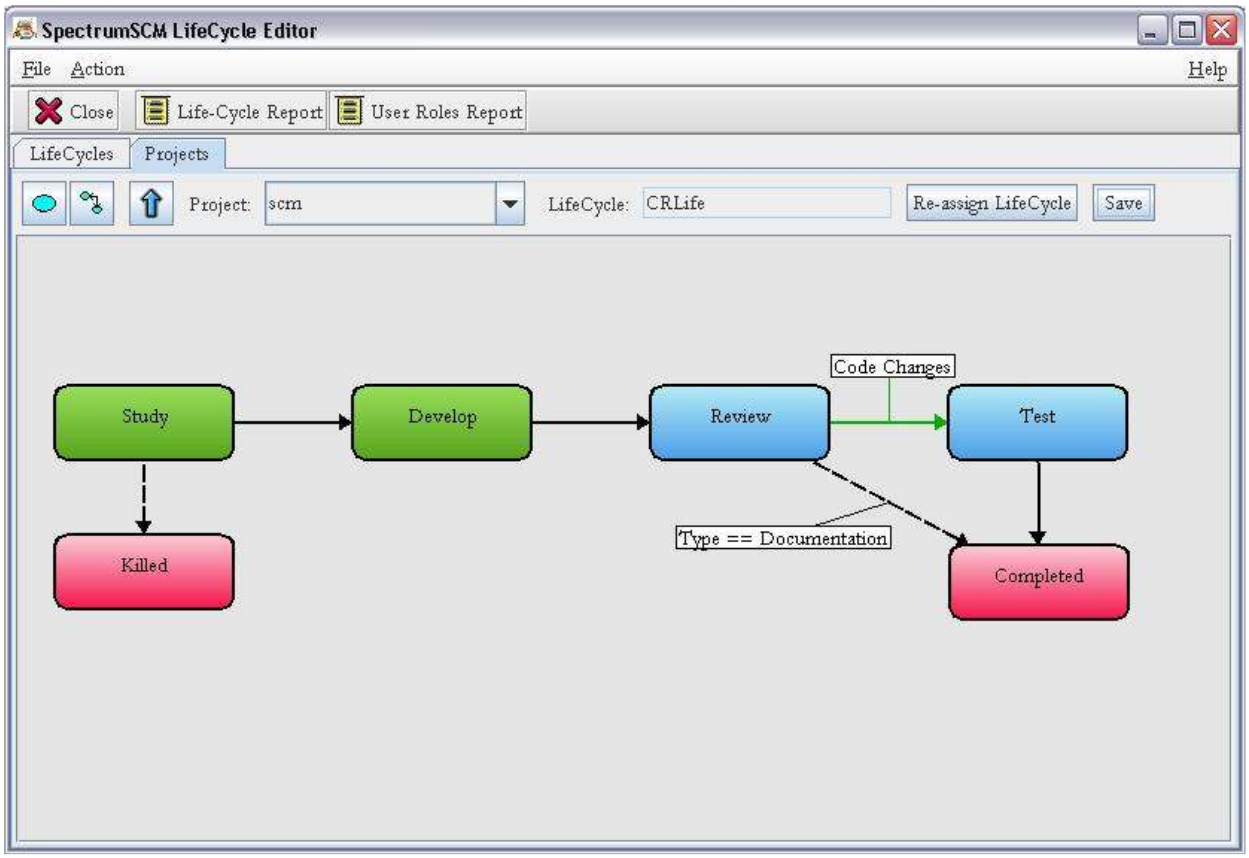


6.1.3.3 Life-Cycle/Workflow Administration

The workflow administration screen is an enhancement and replacement for the Linear Life-Cycle administration described above. All the Life-Cycle administartion functionality is available under the Workflow screen. However, once the workflow functionality has been activated you cannot make changes in the Linear Life-Cycle administration screen. If you wish to revert to using the Life-Cycle administration functionality, you can, but the workflow data items will be archived.

The key functionality points for workflow administarion screen are –

- The capability to specify valid transition paths, so that normal assignment tasks are focused and not complicated with unneeded options. For example, when assigning into a testing phase, only the testing users can be provided as options.
- The capability to automate transitions by specifying who is responsible for certain phases of the life-cycle.
- The capability to specify multiple transition paths, so that different types of Change Request can have different life-cycle paths.
- The capability to register callouts so that external functions/business rules can be implemented.
- Register custom e-mail triggers so that specific people or roles will receive mail about certain CR transitions but not others.
- The capability to state that particular phases are "Approval" phases, where the approver is prompted to approve (or reject) the CR.

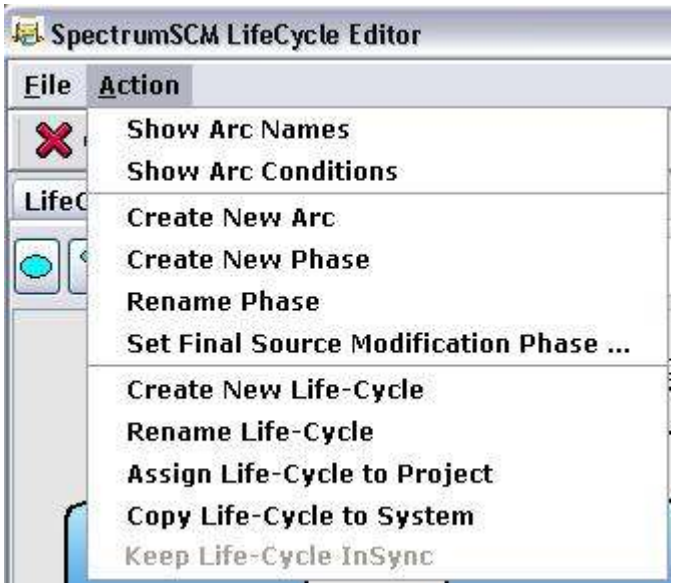


Life-Cycle/Workflow Administration Screen

Action menu item provides the following options –

- **Show Arc Names** – This is a toggle. If turned on, any arc names (like "Code Changes" above) will be shown. An arc name can be specified during arc creation or by using the right-click "Manage Arc Name" option.

- **Show Arc Conditions** – This is a toggle. If turned on, any arc conditions (like "Type == Documentation" above) will be shown. An arc's conditional can be specified during arc creation or by using the "Manage Conditional" option on the arc right-click popup menu.
- **Create New Arc**
- **Create New Phase**
- **Rename Phase**
- **Set Final Source Modification Phase** – See below for Final Source Modification details and implications.
- **Adjust Phase Linear Order** – Since the graphical workflow can essentially define a 2 dimensional graph but yet certain application choices such as during CR creation or Assign/Modify present a linear choice-box, this option allows the ordering of the workflow to be adjusted. In general this should not be needed, but the functionality is provided in case it is needed or desired.
- **Create New Life-Cycle**
- **Rename Life-Cycle**
- **Assign Life-Cycle to Project**
- **Copy Life-Cycle to System**
- **Keep Life-Cycle InSync** – When a project change is made, a popup will ask whether this change is to be made "Common" i.e. for all the projects using this life-cycle, or "Uncommon" i.e. only for this specific project. This choice is "stored" through the "Keep Life-Cycle In-Sync" menu item, and can be changed (toggled) at a later time (if desired). **Note** – If the "uncommon" option is chosen, the life-cycle will be made specific to this project by creating a new life-cycle with the original name appended with the project name. For example: Life-cycle "CRLife" for project "abc" would become "CRLife_abc" when specialized.



There are 2 tabs on the Life-Cycle/Workflow Administration screen -

- The **"LifeCycles"** tab - This is where your system-wide life-cycle templates are defined. Projects can be assigned to life-cycles from this set. Toolbar buttons are available to create new life-cycles, delete life-cycles and clone existing life-cycles into new ones.

- The **"Projects"** tab - Where the assignment of a specific life-cycle to a specific project is made. Note that projects can use the system-level template definition as-is, or they can customize/extend it as appropriate for that specific project. If the project life-cycle is extended (made different from the template) then it will be given a new name to differentiate the project level from the template.

The two tabs together have a total of 6 toolbar buttons, 5 on the LifeCycles and 3 on the Projects tab.

 Create New Life-Cycle.

 Delete Life-Cycle.

 Clone Life-Cycle – Create a new life-cycle from an existing one.

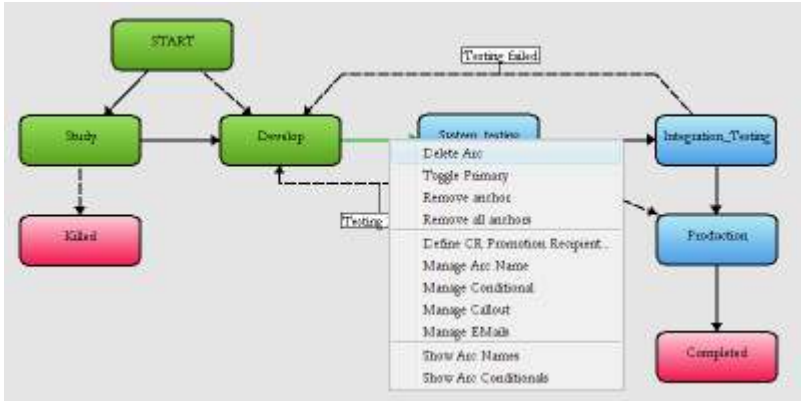
 Create a new life-cycle phase (for bothLifeCycles and Projects).

 Create a new life-cycle arc/transition (for bothLifeCycles and Projects).

 Copy the current project specific life-cycle up to the system/template level thus making it available for other projects to use.

6.1.3.4 Phase Customization

Once a phase has been created (named) you can right-click on it to specify other attributes (or modify existing ones) if desired.



- **CR Assignment User Category** – To specify which user roles are to be available to be assigned tasks in this state.
- **CR Assignment Users** – To specify which individuals are to be available to be assigned tasks in this state. This works in addition to any assignment user categories specified above.
- **CR Promotion Recipient** – To specify who is essentially responsible for this phase i.e. if this is the "Testing" phase, then the probable promotion recipient would be the test team leader. The promotion recipient is who would be assigned tasks automatically coming into this state.
- **Approval Attribute** – If this phase is an approval phase, which CR attribute is to be used to record the approval state. See below for more details.
- **Final Source Modification Phase** - see below

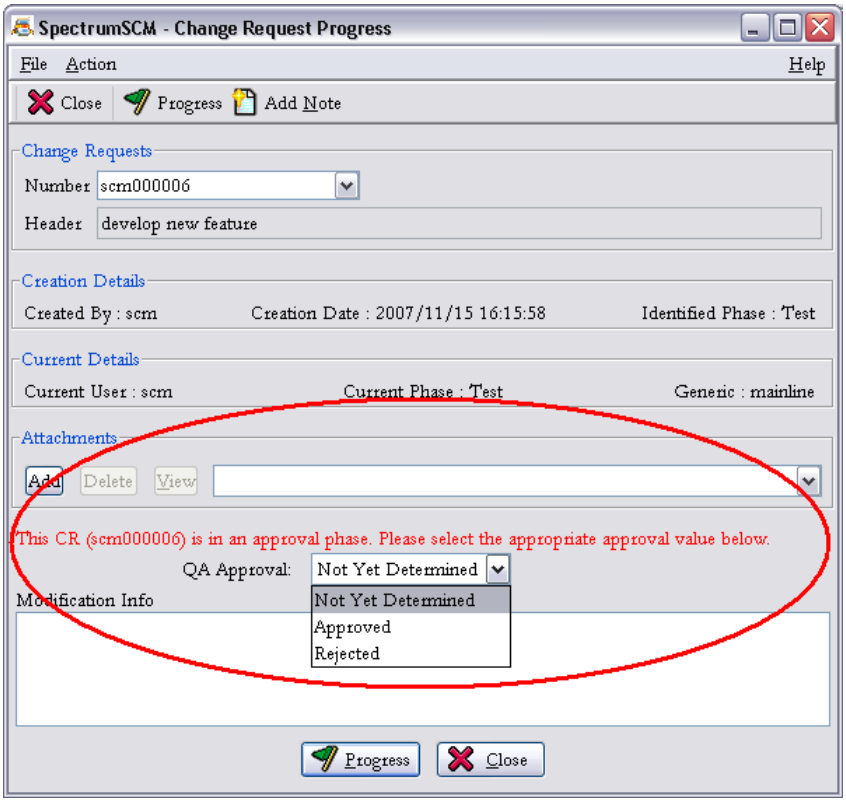
Note that if there are no CR Assignment entries made then all the current project users will be presented as assignment choices.

6.1.3.5 Approval Attributes

The approval mechanism is implemented by prompting the assigned user to update the approval state on the Change Request. This update, whether it be to a "passed" or "failed" state is recorded against a selected CR attribute such that it can be reported on, tracked and audited in a normal manner. Indeed, since attributes can be used to conditionally switch to different phases, such "passed" and "failed" conditions can be automatically applied to the workflow. In this way the CR can automatically transition to the appropriate success or failure phase.

As an example usage, a CR attribute called "QA Approval" could be created with values "Not Yet Needed", "Approved", and "Rejected". Define the approval attribute for the "Test" phase to be "QA Approval". Then whenever the CR is to be progressed from the "Test" phase the assignee will be prompted to set the attribute to the "Approved" or "Rejected" values. The workflow could then conditionally switch on this attribute to direct the CR forwards or backwards as appropriate.

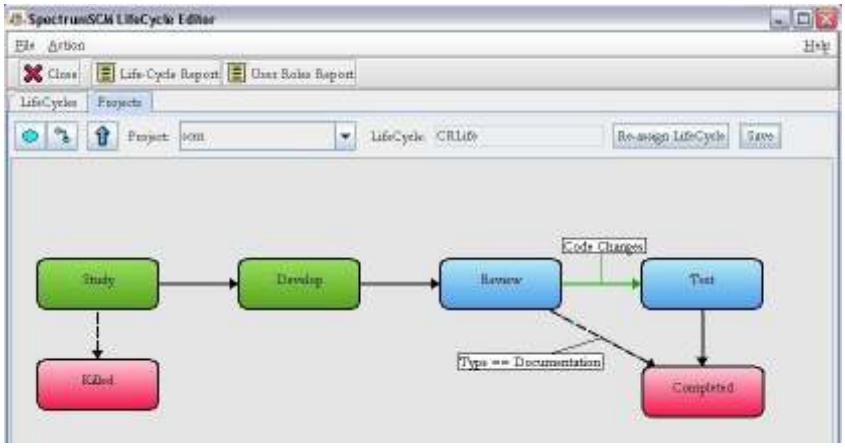
Select and "Set" the desired Approval Attribute as appropriate for the particular phase/state.



Since the approval is input through the progression screen, regular progression/ approval notes can be input at the same time before hitting the "Progress" button.

6.1.3.6 The Final Source Modification Phase

The Final Source Modification (FSM) Phase controls when Change Requests are considered eligible to be placed into a release. The final source modification phase must be on the primary path (shown with the solid line). All phases before the FSM phase will be shown in **green** to show that these CRs are OK for editing. All phases after the FSM phase will be shown in **blue**, CRs in these phases will be considered good to be placed into a release (subject to dependency checking).

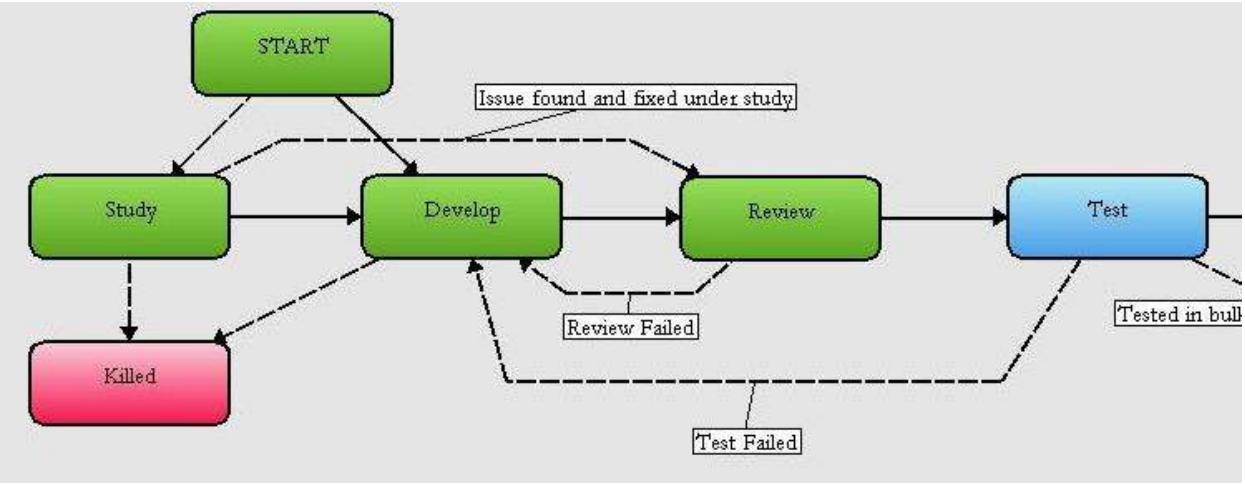


So for example in the LifeCycle/Workflow Administration screen shot above, the Final Source Modification phase can be seen to be "Develop".

If phases on secondary paths (shown with dashed lines) need to be considered releasable then the "Set Past Final Source Modification Phase" option can be used.

6.1.3.7 The "START" Phase

If you want to constrain the CR creation process then you can define a phase with the name "START" (case sensitive). Arcs from this phase define the valid assignment/promotion paths and can even specify conditionals (if appropriate).



If such a phase is defined then the CR creation assignment will follow the assignment user and assignment roles subject to any conditionals that have been specified.

If no assignment is specified on the CR create screen, then the promotion rules will be evaluated to see if the newly created CR can be automatically assigned. If no assignment is specified and no automatic path is appropriate the CR will be placed in the 'TBA" state.

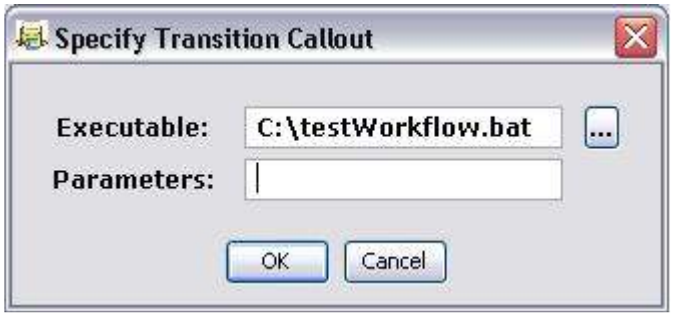
6.1.3.8 Arc definition and customization

The solid arrowed lines indicate the primary transition path through the workflow. This is essentially the normal path work-items will flow. The dashed lines indicate secondary paths which can be taken if certain conditions (manual or automatic) are met.

To define a transition you can select the toolbar button. You can also select the source phase, or the source and target (use the shift key), and right-click to "Add Arc". Any of these will present the following screen –

This is where you can specify if this transition only applies to certain types of Change Request based on the conditional constraints. By selecting the ... button to the right on the *Conditional* line, attribute options will be presented for specifying transition constraints. Only Change Requests whose attributes match the specified conditional expression will be considered valid for this transition.

External callouts can be made by specifying the program name and any parameters. By selecting the ... button to the right on the *Callout* line, external callouts can be specified. Note that callouts will be executed on the SpectrumSCM server as this is the central control point, not on an individual users work-station. Also note that the callouts are executed with an additional parameter added, a file-name containing all the details of the change request and the transition being made.

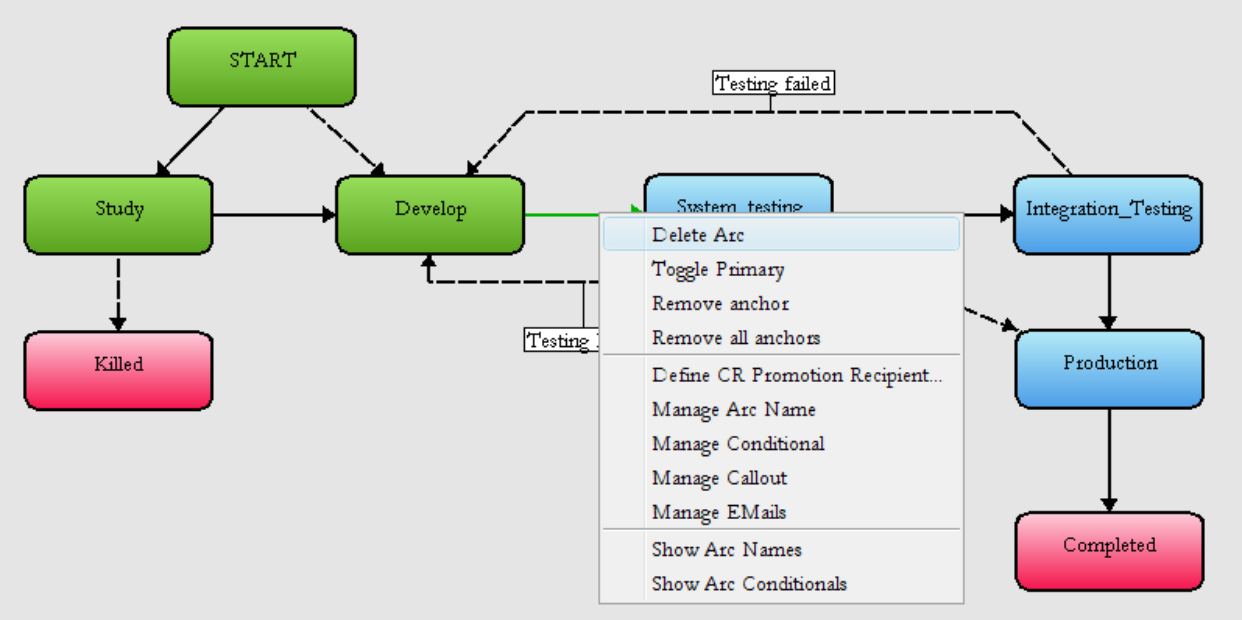


Selecting the ... button to the right of on the *EMails* line allows browse and control of who should receive e-mail for this transition. Note that e-mail recipients can be selected by role or as an individual but they must have an entry under the "User Administration" screen (which gives their actual e-mail address).

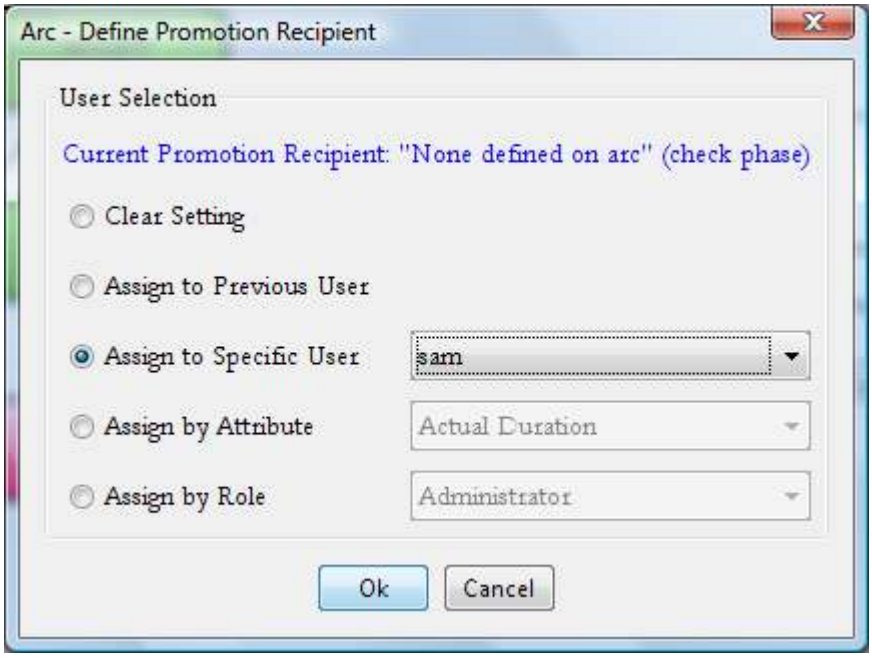
Once an arc has been created you can right-click on it to update any of these values. In addition to the functionality above, you can also -

a. Toggle **labelling** on (or off) to show (or hide) either the arc name or the arc conditional expression.

b. Specify an **Arc Promotion Recipient**, to further specify your automation rules. New



When specifying an **Arc Promotion Recipient,** be aware that it will override the Phase based promotion recipient (if set). In this way the phase promotion recipient can be considered the default setting for the target phase, with the arc promotion recipient overriding that default, subject to the arc's conditional expressions.



New

In setting the Arc Promotion Recipient there are 5 options -

1. **Clear Setting** - Clears any existing Arc Promotion Recipient setting so that only the target phase promotion recipient will apply (if any).
2. **Assign to Previous User** - Determine which was the most recent user assigned this CR under the target phase and assign the CR back to that person. For example, if the CR is being rejected from a testing state and being moved back into development, the task probably wants to get assigned to the person who developed this item. Note - if no previous user is found for this target state, the CR will default to the phase promotion recipient.
3. **Assign to Specific User**
4. **Assign by Attribute** - This option is meant to allow pre-specification of an issues desired workflow routing. By setting up CR attributes such as "Responsible Tester", "Requirements Engineer", "Project Manager" etc (as appropriate in your business environment). These attribute values should either have the appropriate user ids specified or be editable (or both). The workflow will retrieve the value for the specified attribute and if it is a valid user for this project, the CR will be assigned to that person. If the attribute does not exist or its value does not specify a valid user id the CR will be defaulted to the phase promotion recipient (if any).
5. **Assign by Role** - If there is currently one person assigned to the specified role for this project then assign this CR to that person. If there is more than one person or there are no applicable user ids, then the CR will be defaulted to the phase promotion recipient.

6.1.3.9 Automation

If, when a CR is progressed, there is one and only one appropriate target phase as defined by the workflow arcs and any conditional expressions, AND the promotion recipient has been specified,

then that CR will be automatically progressed and assigned to the promotion recipient.in the appropriate life-cycle phase.

Any callouts associated with the transition will be automatically executed. Any e-mails specified will also be sent. If this condition is not met, the CR will be placed in the To Be Assigned (TBA) state to await appropriate guidance/assignment.

6.1.4 User Category Setup

This topic is covered in more detail in Chapter 5.

6.1.5 Project User Setup

This topic is covered in more detail in Chapter 5.

6.1.6 Change Request Attribute Setup

This topic will be covered in more detail in Chapter 7.

6.1.7 First Change Request Creation

This topic will be covered in more detail in Chapter 7.

6.2 Setting up the local root directory or local work area

The **SpectrumSCM** system stores files internally in a directory structure that duplicates that of the native OS directory structure. A local work area is an area on the client workstation into which files can be extracted and extended without interfering with the files in the rest of the system or other workspaces. When files are extracted from the SpectrumSCM system into client workspaces, they are written into the receiving file system using that same directory. If the necessary directory structure does not exist at the time of the extraction, the directories are created as necessary.

The Local Root Directory defines the location on the client hard disk where files to be checked in are found and files extracted to the hard drive are placed. This is needed so that files stored under SCM have only relative path names. An example "root directory" would be the directory in which you perform your local product builds or compiles. Another example would be a directory that you use when you are loading source into SCM from the directory.

Subdirectory names in all local root directories need to match the SpectrumSCM file structure in the project tree. Note that you can have multiple root directories if needed. Select the one you wish to work with via the Local Root Directory pull-down menu selection box.

For example, in a Windows environment for the file path C:\temp\src\test.C, C:\temp would be the local root directory. In a Unix or Linux environment, for the file path /temp/src/test.C, the local root directory would be /temp.

For our example project, we have chosen to set the local root directory to c:\temp. Each "scm" project team member will create a folder by the same name on his or her client machine. Additional local root directories can be created by each user as needed and selected for use via the local root directory pull-down menu.

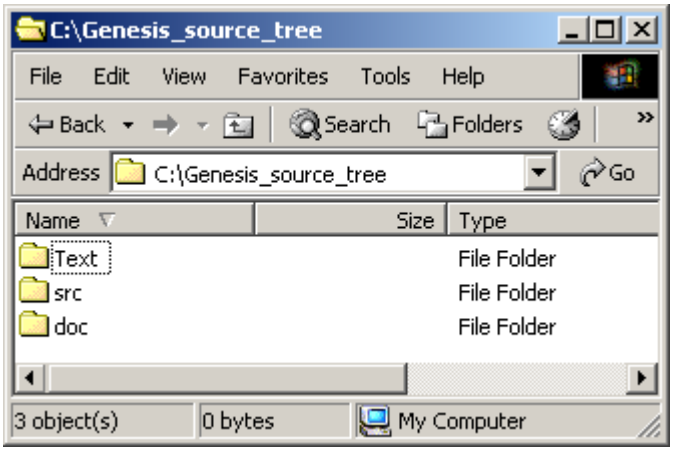
6.3 Setting up the project directory structure in SpectrumSCM

It is a good idea for the project engineer or generic engineer to set up the basic file structure for the project before development begins.

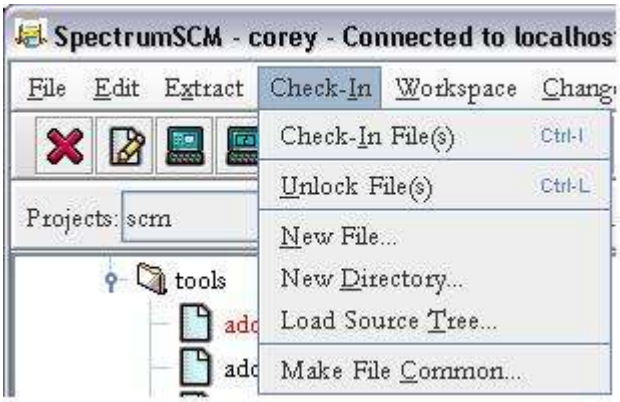
This can be done using the **Add Source Tree** screen. The Add Source Tree screen is accessed via the Main Screen **Check-In / Load Source Tree** menu option.

A CR has to be created to load the source tree (in our example, this is done under the "set up project in SCM" CR). *See Chapter 7 for details on creating, assigning, and progressing CRs*

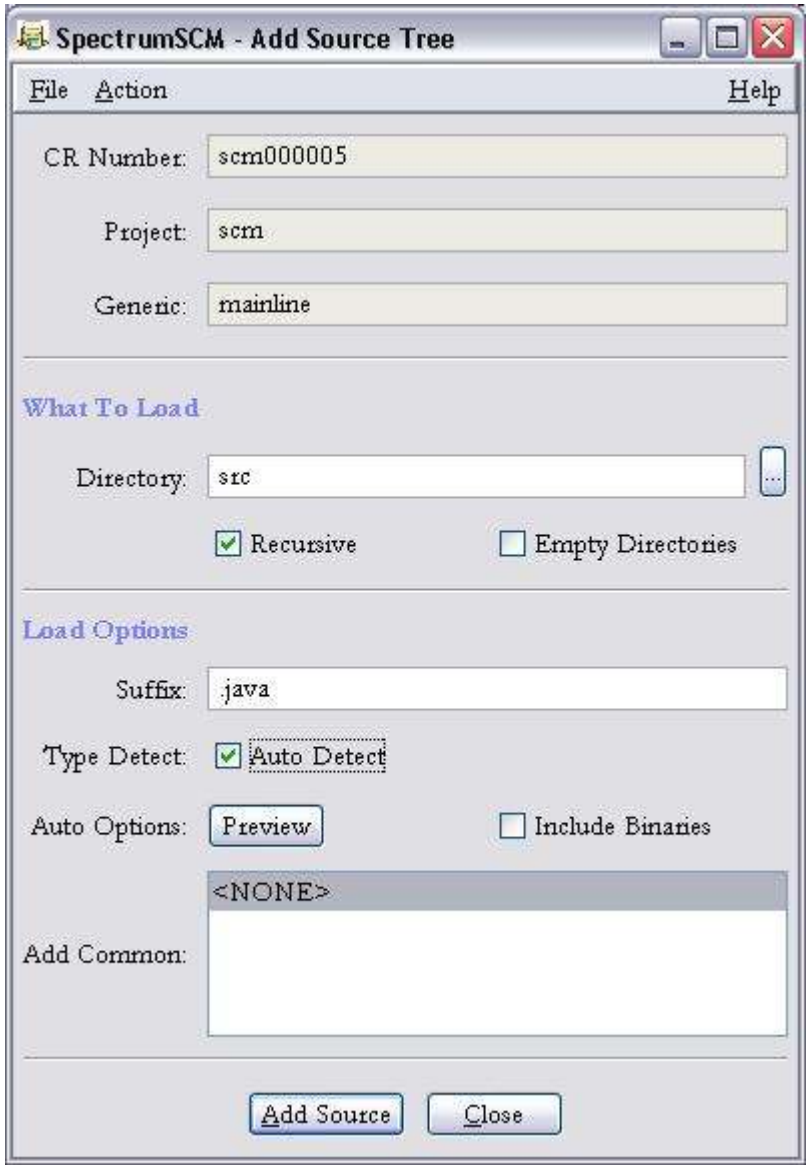
The structure to be loaded must be in a local root directory (in this case, C:\Genesis_source_tree).



On the main screen, select the CR, the project, and the local directory containing the project structure. Access the Add Source Tree screen via the **Check-in / Load Source Tree** menu option.



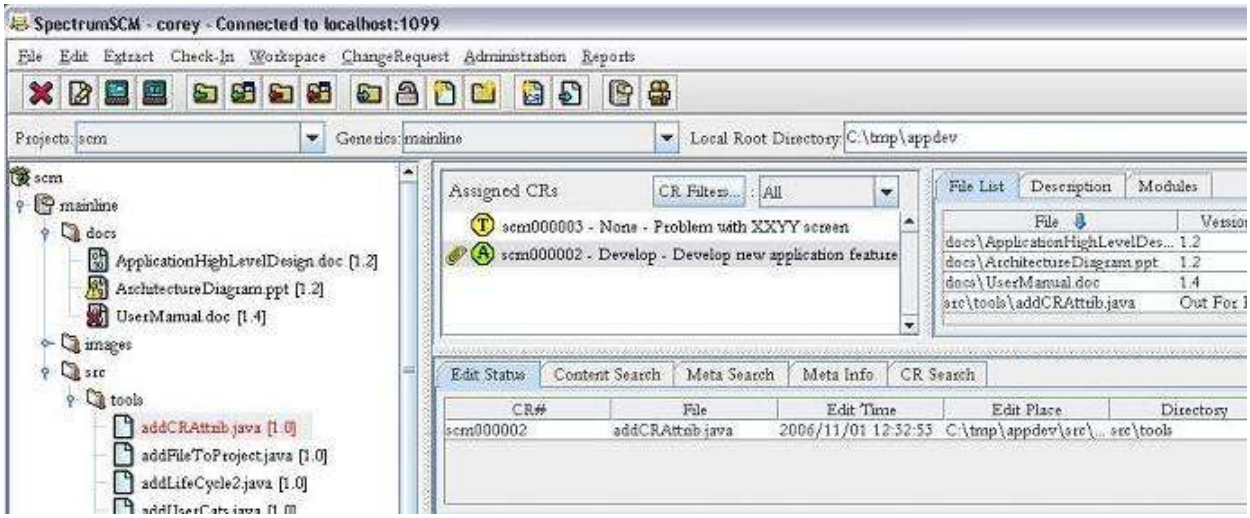
The **Add Source Tree** window will be displayed.
The SpectrumSCM system will look in the specified local root directory and move its contents into the SpectrumSCM system into the corresponding project and generic directory (in this case, Mainline under the project Genesis).



Specify **Suffix** to load only those files with a specific suffix (for example .java, .doc, .txt)
The system will automatically detect the "type" of a file (binary or text) as it is being loaded into the system. This decision can be overridden by the user by toggling the **Auto Detect** checkbox, which then enables the **Load As** (ASCII, BINARY) check boxes. By using these check boxes the user can force the system to load files of any type as another type. For instance, binary files can be checked in as text and large text files can be checked in as binary.

Specify **Recursive** if there are subfolders to be loaded. Specify **Include Binaries** if there is any non-ascii source to be loaded as part of the initial set-up. Specify **Empty Directories** if the system should add empty directories to the repository during the load.

The **Add Source Tree** function is also used to move an existing baseline source tree into SCM if you are migrating to SpectrumSCM from another CM tool or manual file versioning. To establish the structure for a new project, create a source tree of empty directories with appropriate folder names and move those into SpectrumSCM using this function. When the load is completed, a confirmation is displayed. The loaded project tree structure can be seen via the Main Screen Project tree.



Alternatively you can use the **DragNDrop feature** available in SpectrumSCM to check-in or load a single folder or an entire folder structure by simply **Dragging** the files or folders that are of interest directly from desktop or file system and **Dropping** it into appropriate folder location on the project repository window panel. This mechanism can also be used to check in individual files as well.

This feature basically automatically populates load source tree or the add source file screen with the proper source and target location without the user having to set the local root directory or screen settings.

7 Change Requests and Issue Tracking

Chapter 7

In this chapter you will learn about the various fields in a change request form, how to create change requests, assign change requests to an individual for work, manage change requests, and use change requests to manage the work of a development effort. You will also learn how to customize change request attributes for each project.

7.1 Change Requests (CRs)

Change Requests are the glue that binds the SpectrumSCM system together.

- Releases are composed of units of work that are defined by CRs
- Users are required to make all changes to the system through one or more CRs.
- CRs are the only mechanism that allows management to track the history, activity, and status of particular units of work or issues.

Similar acronyms are MR (modification request) and WR (work request).

A change request is usually created by an authoritative source on a project and assigned for work to be done in a particular phase, by a particular developer. Once the requested work for the current life-cycle phase has been completed, the CR is "progressed" to the next phase by that developer. Additional notes or other work requirements can then be added to the CR as it continues through the project's life-cycle towards completion. At any point, the generic engineer can re-assign the CR for work in the next phase or any other defined phase. In essence, the management of CRs is workflow management, with the work being the development of software and related products and services.

A Change Request can be created at any time by any project team member who has permission to create change requests *(see Chapter 4, User Management)*. Creating CRs is a means of reporting new tasks, bugs, issues, and defects in the current release. These CRs could then become part of the current or subsequent releases.

From a **management perspective**, CRs allow for the easy tracking and resolution of problems as well as the progress of feature sets and or releases. Ownership and status of each work product is clear and easily obtained by any member of the project team.

From the **developer perspective, SpectrumSCM** provides workflow management without hindering the natural flow of work. The assignment of CRs to specific individuals clarifies "who's doing what" and the progression of a CR clearly marks the work as ready for the next life-cycle phase (for example, from development to testing) with no need for additional communication or the chance that a component was overlooked.

Testers can easily communicate problems by adding a CR to describe a problem or by sending a CR and its components back to the previous phase for correction.

7.2 Creating and Managing Change Request Attributes

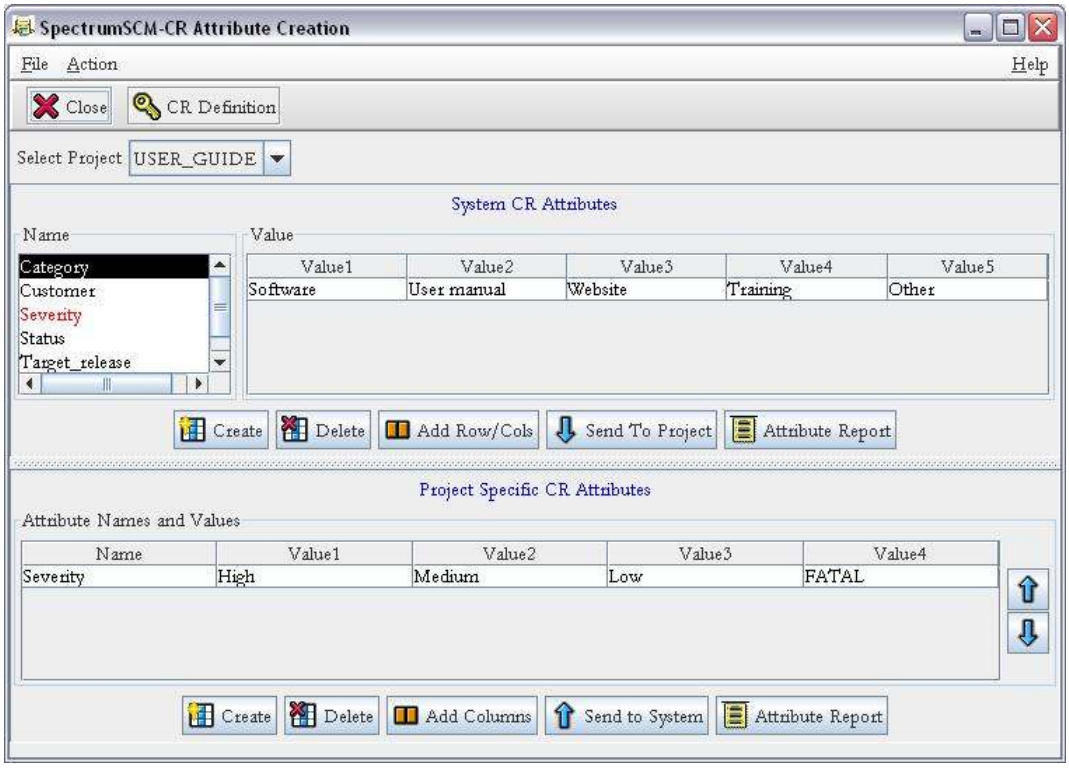
CR attribute-value pairs should be added before the very first CR is created for a project. This typically is undertaken when the project environment is set up *(See Chapter 5)*. Attributes can be added later, but once a CR is using an attribute, that attribute value will not be deleted from the CR even if the attribute itself is deleted from the attribute management screen (otherwise traceability will be lost).

SpectrumSCM allows for a completely customizable Change Request, just as it provides for the customization of project life cycles and change request attributes at the project level. Most software development shops have their own ideas on what fields or attributes should be found on a Change Request form. Even within a shop, project teams with different needs might disagree. SpectrumSCM gives each project team the freedom to set up SCM for large projects with a complex life cycle and CR support, but also to support and manage very simple systems with minimal CR and life cycle support.

7.2.1 Define a project's CR Attributes

When setting up a project, the CR attributes need to be defined. The project engineer should define these during project start-up. This is done via the CR Attribute Creation screen, accessible from the main screen via the Administration menu.

The CR Attribute Creation screen is used to maintain Change Request Attributes.



On the top of the screen are the system-wide attributes; on the bottom are the attributes assigned to the selected project. In the far left panel in the top pane is the list of all the system attribute names that have been defined. When one of these is selected, its attribute value set(s) are displayed. Note that an attribute name can have more than one set of values at the system level. When an attribute is assigned to a project, a value set must be chosen. This means that Project A can have a Location value set of "Atlanta", "Boston" etc, while Project B could use the values "Houston", "Orlando" etc.

7.2.2 Creating an attribute

To create an attribute, select the **Create** button. If you want this attribute only in a specific project then select the project side **Create** button, otherwise select the system side **Create** button.

A panel is presented requesting the attribute name, a brief description of its purpose (used for tooltips), and the number of values the attribute can have.

The **Editable** check-box allows the attribute creator to specify whether this particular attribute can be manually edited by the CR creator. As an example,

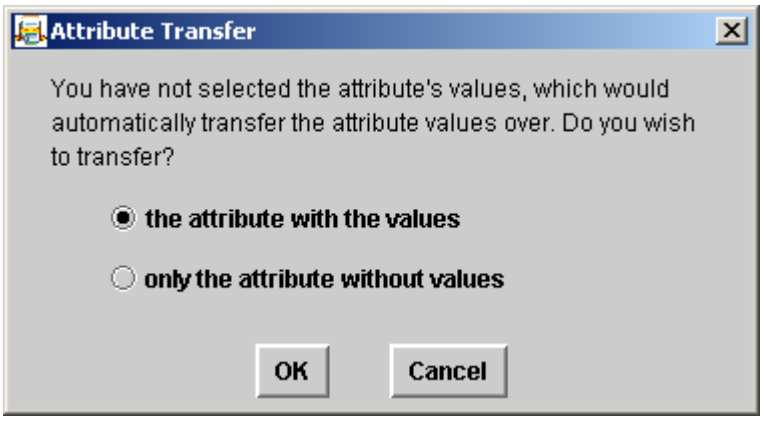
when the CR creator selects the **Location** attribute, they can either use one of the pre-defined values provided, or they can manually enter a completely new attribute value. The editable property of any CR attribute can be changed by using the pull down menu system located on the CR Attribute Creation Screen:

When the attribute creator presses the **Next** button, a second panel is presented which allows the user to enter the default values for this attribute.

The **Force User Selection** check-box sets the mandatory status of the item to be turned on. An attribute with mandatory status means that the user is not allowed to finish the creation of a new CR without picking a value for the selected attribute. The default value for a mandatory attribute is always the string "Select A Value", which can be seen when creating a new value set for a mandatory attribute.

7.2.3 Assigning attributes to a project

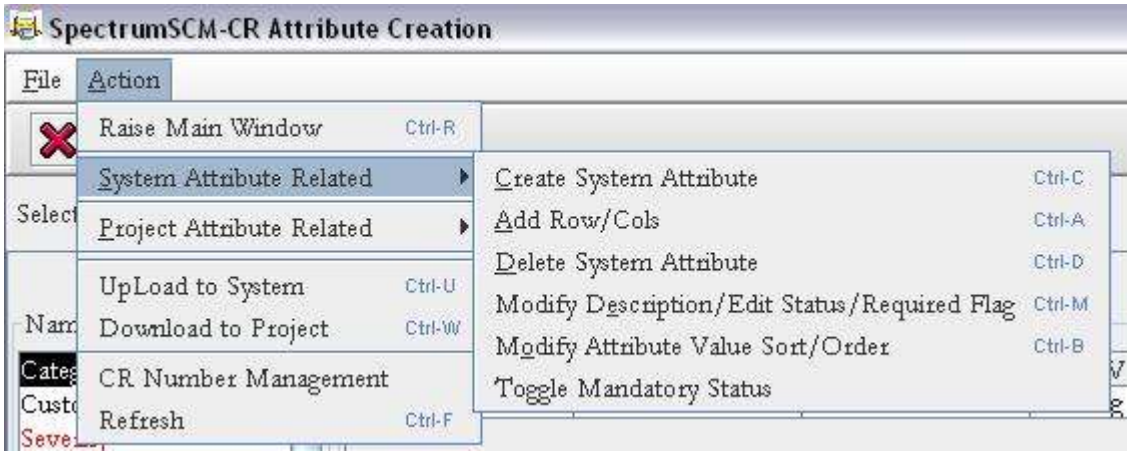
If attributes have been created at the system level, they can be assigned to a project simply by selecting the appropriate attribute name and value set, and then selecting the down (Send to Project) arrow button. Attributes can similarly be copied from a project to the system level (for use in other projects) by selecting the appropriate attribute and the up (Send to System) arrow button. If only the attribute is selected, the user is presented with a choice to include a value set or create one specific for the project.



7.2.4 Adding and Deleting Attributes and Values

An attribute can be deleted from the system or project by selecting it and the appropriate **Delete** button. Values can be added to an attribute value set by selecting it and the system-side **Add Row/Cols button**  (adding rows adds a new value set; adding columns adds values to the existing value set) or the project-side **Add Cols** button, via the **Action** Menu for either System or Project Attributes, or using the Add Rows/Cols menu option.
Specify the number of columns that are to be added (either to this row or in the new row (if that option was chosen)). The system then prompts for new values.
To change a value in an attribute set simply select the value in the table cell and edit it. Use the TAB or Enter keys to complete the change. To delete a value from an attribute set simply select the value and edit it to be blank.

For our example Genesis project, we chose **Action / System Attribute Related / Create System Attribute** to add the location attribute and its initial value set.



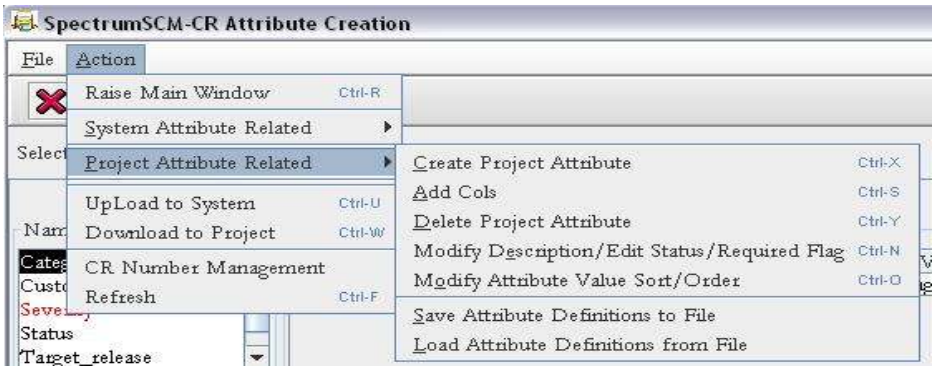
The **Add Rows/Cols** feature is used to add a row (a new value set) to a system attribute or to add a column or columns (additional value(s)) to a value set.

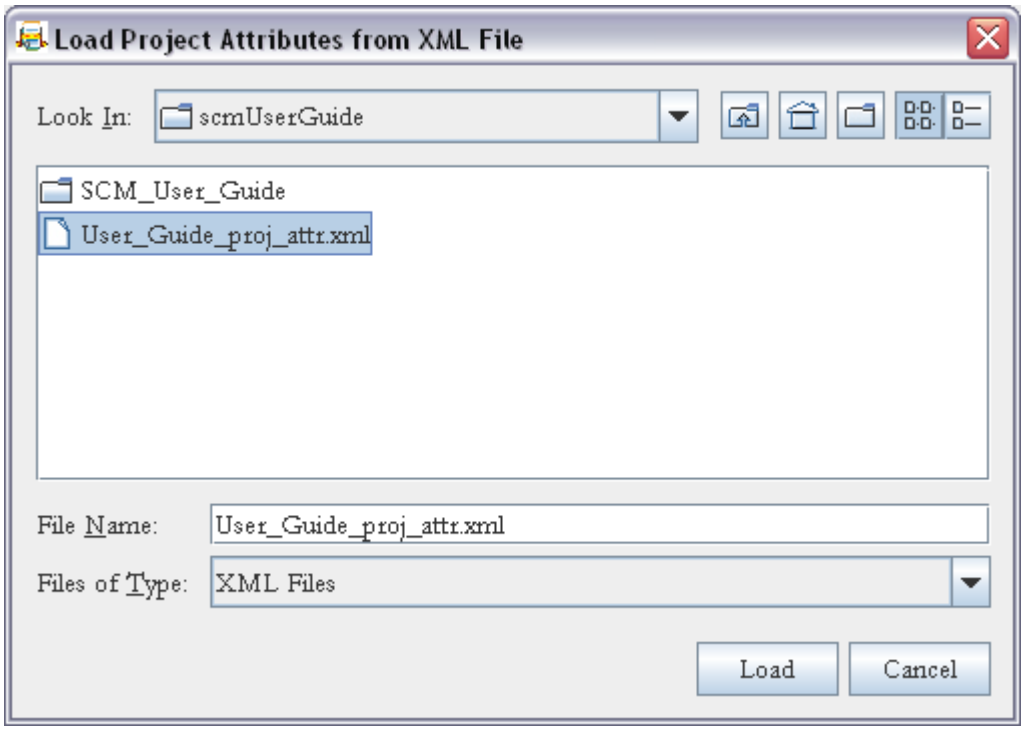
As an example, one could use the **Add Row** feature to create a new value set for the Severity attribute containing the values "bad" and "real bad". The **Add Column** feature could then be used to add two more values ("worse" and "worst") to that value set. This value set might be useful for a project that is only correcting problems. Projects would be able to choose which of the value sets they want to use.

For Genesis, we chose to use the Severity attribute and the value set (High, Medium, Low) to move to the project side. We also selected the Location attribute and its value set to move over. The result is that Genesis CRs will have two attributes, Severity (using the value set = High, Medium, Low) and Location (using the value set = Atlanta, Chicago, Boston, Miami). Additional locations can be added at any time using the **Add Cols** feature.

7.2.5 Saving and Loading Project Attribute Definitions

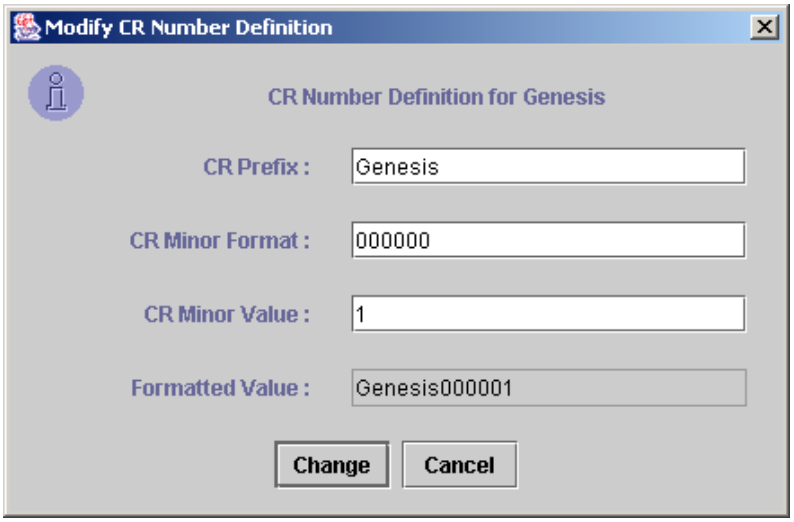
Project attributes can be saved to a file by selecting "**Save Attribute Definitions to File**" option from "**Project Attribute Related**" option on CR Attribute Creation window. Saved project attributes can be applied to new projects by selecting "**Load Attribute Definitions from file**".





7.3 Change Request Number Definition

It is very important that the CR numbers be unique. The CR number is made up of a string followed by a number; by default, this is the project name and a 6-digit number. The string, the numbering scheme, and/or the "minor" value (the starting number for the CRs) can be changed on the **CR Attribute Creation Screen, via** the **Toolbar** or via **CR Number Management** from the **Action** menu. Using this facility, you can select any string prefix to be followed by any number format.
For our example project Genesis, we choose to take the default prefix and numbering scheme.
Now that our project CR attributes and value sets have been established, we can begin to add CRs.

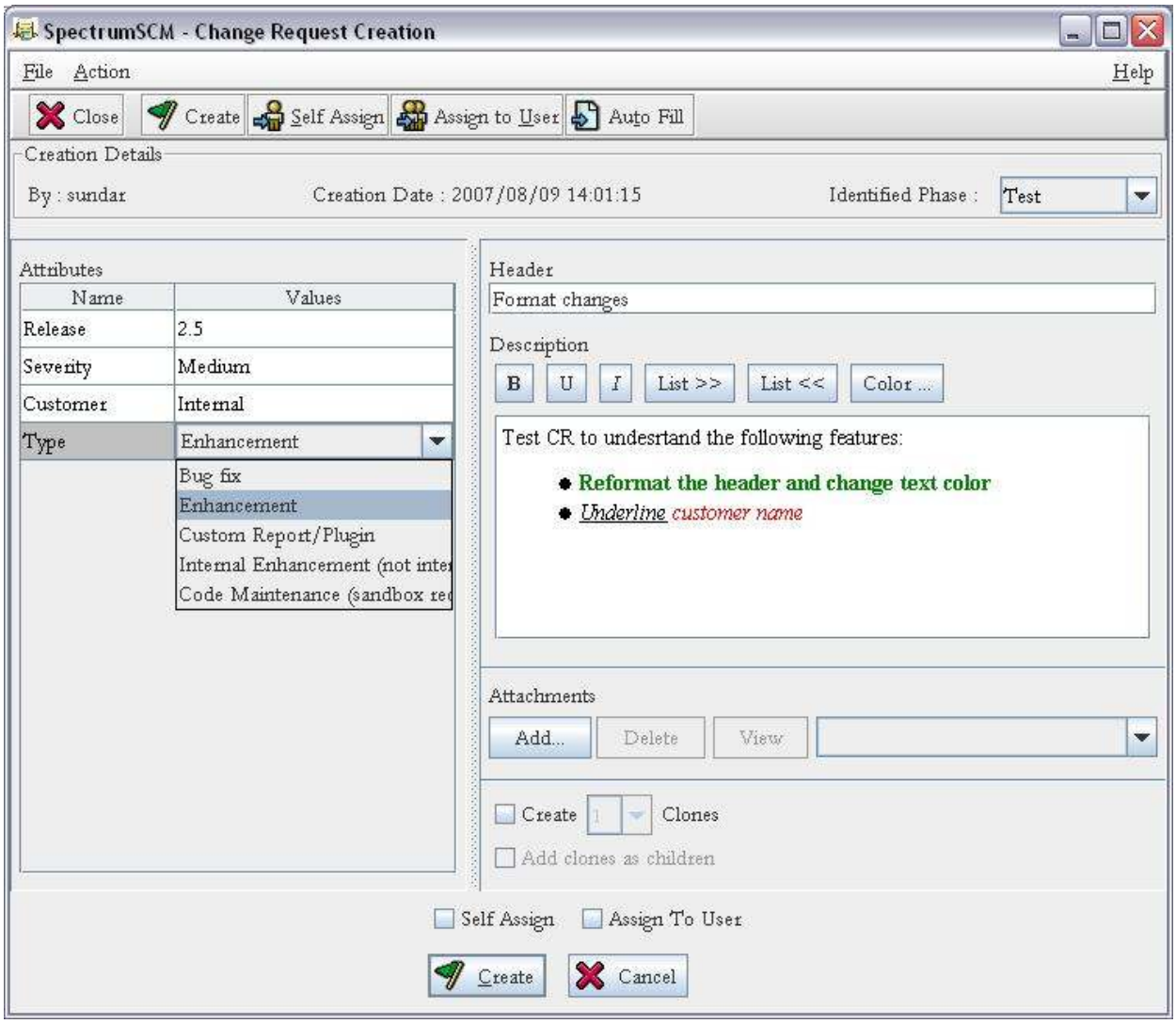


7.4 Creating and Managing Change Requests for a Project

Creating and managing CRs is typically the work of the generic engineer.

7.4.1 Creating a Change Request

The CR Creation screen is used to create a new CR for the project and assign that CR to a member of the project team. CRs can only be created by users whose roles allow them "Create New CRs" permission *(see Chapter 5 User Management - where user roles and permissions were defined and assigned to users).*



On the left are four attributes for this CR and their possible values. The value fields are actually pull-down choice-boxes(double-click to activate the pulldown), which contain all the possible values for these attributes. Editable CR attribute values are shown in blue can be manually entered or the default pull down values can be chosen. Mandatory attributes are shown in red, an appropriate value selection must be made before the CR can be created.

On the right is the header field for entering a single line description and below that the description area for entering larger amounts of information pertaining to the CR. The options provided by buttons in CR description section (shown below) enables the CR description to have color, bold, underline, italics and list formatting.

Attachments can also be added to the CR at this point, attachments are supporting documents such as screen snapshots, test plans and/or requirements snippets. Any form of supporting information can be added.. Simply select the "Add…" button in the Attachments area and the system will respond with a file selection dialog box. Select the item to add as an attachment and press OK. The attachment can be viewed immediately by selecting the "View" button or the attachment can be deleted from the system by selecting the "Delete" button. The "View" option uses the editor selections to open the appropriate viewer for binary files (see Custom Editor preference settings under Chapter 5). Text files can be viewed using SpectrumSCM's editor, or using a custom viewer as desired. Attachments are not version controlled but updates are recorded against the Change Request in terms of who performed the update and when. If full version control of these items is desired, then check the items in to the repository against this change request instead of using the attachment mechanism.

Drag-N-Drop functionality can also be used to directly add external files such as requirements documents or e-mails, as attachments. Textual information/files can also be added to the CR description field. Note, drag-N-drop works best with files in to CR attachments, there are a wide variety of drag-N-drop sources and it is therefore difficult to guarantee the correct processing of all forms of drag operations. If your drag operation does not work as desired, save the item as a file and drag that into the CR attachment field.

In the upper right-hand corner of the screen, the CR is being marked as being identified in the *Test* phase i.e. where in your life-cycle is this issue coming from, is the issue coming from production, test, development etc. By default the "**Identified Phase**" selector lists all of your life-cycle phases in order. If this is not appropriate or you other wish to customize this list, you can do so by defining a CR Attribute called "Identified Phase" (case sensitive) and giving it the values you desire. You can also make this attribute (and hence the "Identified Phase") a forced/mandatory item or give it a default value.

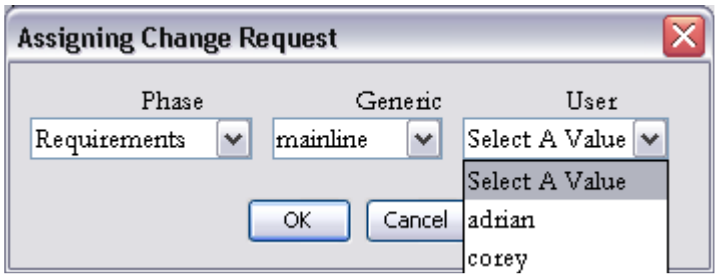
Clones of a CR can be immediately created using the CR cloning feature. The cloning feature allows a user to create a set of related CRs that can be tied together in a parent child relationship. All of the original CRs attributes, header, description and initial phase information will be transferred to the clones upon creation. *See section 7.5 in this document for additional information on CR Work Breakdown Structures and how to create and manage parent/child and peer/peer relationships.*

If you want to create a "clone" of an already existing CR, use the **Auto-Fill** button off of the toolbar. This lets you select the base CR you wish to clone from, and when you hit the "Create" button will copy in the original CRs information (header, description, attributes etc) into the new

CR. Note, both cloning and auto-fill annotate the CRs in their respective description sections as to the origin of any cloned information i.e. yyyy/mm/dd HH:MM:SS: Auto filled using cr00123.

Finally, depending on the users roles (and therefore permissions) the user performing the CR creation can decide whether this issue needs to be assigned to someone directly for work, or not. Select "Self Assign", "Assign To User" or leave them un-selected as appropriate. If the user does not have the appropriate permissions, the respective option(s) will be grayed out.
If **Self Assign** has been chosen, selecting the Create button will present the Assigning Change Request window but with only the target phase and generic choices available. When the CR is created it will be immediately assigned to you the user creating the CR.

If **Assign To User** had been chosen, selecting the Create button will present the Assigning Change Request window but with the additional list of project team members. One needs be chosen to receive the assignment. If in doubt, assign the CR to the generic engineer; it is his or her responsibility to examine and assign "To Be Assigned" (TBA) CRs.



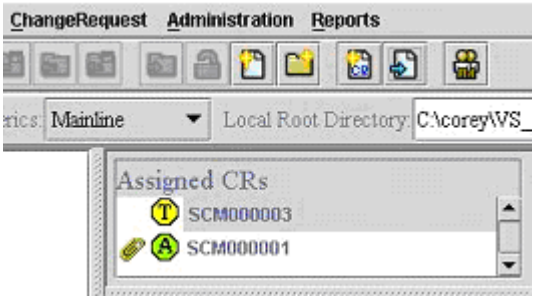
The first Genesis CR has been created and assigned:

If a workflow has been defined for this project with a "START" phase to constrain/control CR creation assignments, then the above dialog options will be appropriately restricted. For example, if a newly created CR should only be assigned either directly to developers or to the review board for acceptance, then only those options will be presented (and only the valid users for those phases will

be presented in the "Assign To User" dialog). See Chapter 6 on process management and defining workflows for more details.

7.5 Automatic Notification of CRs needing Attention

When each member of the project team logs into the SpectrumSCM system, he or she is presented with the CRs that require their attention.
In our example team, both the generic engineer (Gene) and the senior developer (Corey) have generic engineer authority. Corey is also a developer, so he sees CRs assigned to him for work as well.



For example, when user "corey" logs into the SpectrumSCM system, he sees that one CR has been assigned to him. The CR is shown on the screen when he logs to alert him that it requires his attention. The user also sees a CR in the TBA state, meaning that it needs to be assigned.

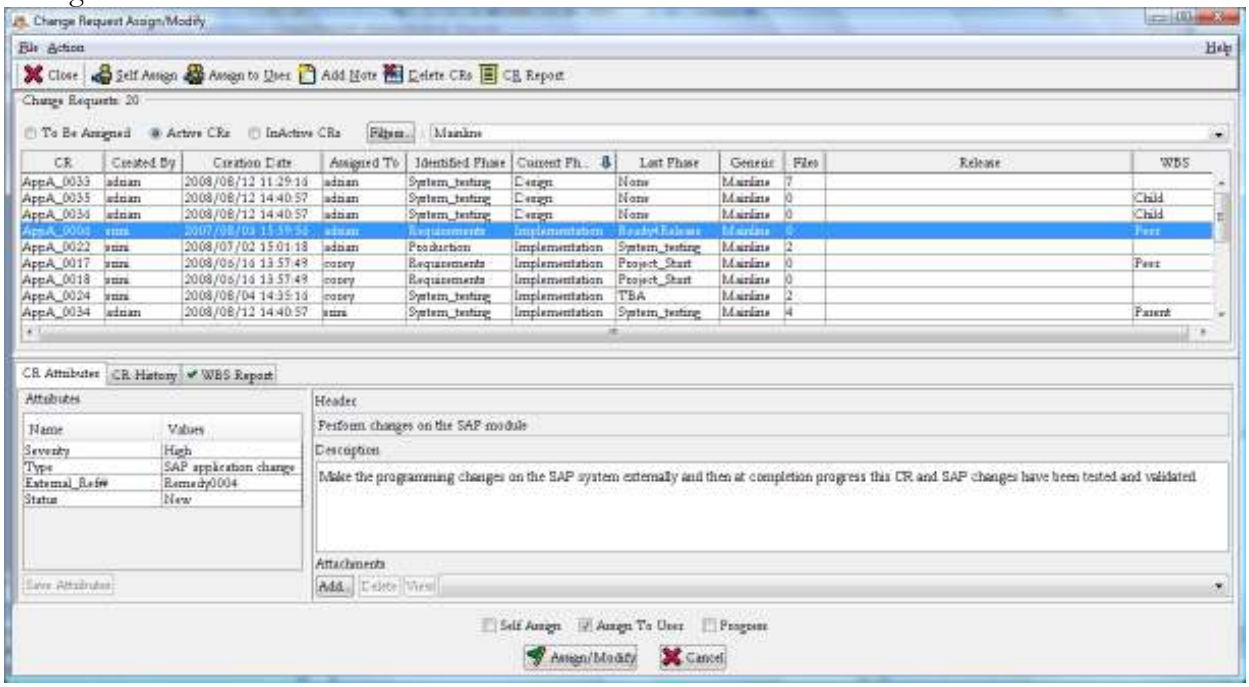
- The (green A) indicates that the CR has been assigned to him.
- A (yellow-T) indicates that the CR is in the TBA state and that someone with the Generic Engineer role or assignment privileges needs to assign it onto the appropriate team member for the next phase in your life-cycle. All users who have permissions to assign CRs will see TBA CRs displayed on their main screen. This allows all appropriate project leaders to be made aware of TBA tasks so they can re-assigned/managed expeditiously.
- A paper clip next to either of the icons indicates that attachments are associated with the CR. The user can "Right Click" the CR to see a listing of the available attachments and select one for viewing or to save the image to the local disk.

Additionally, if the e-mail notification option is configured, users will be automatically notified when CRs are assigned to them to work. If the user is assigned the Generic Engineer role (or has assignment privileges), he/she will also receive e-mails for CRs as they transition into the TBA state.

7.6 Assign / Modify a Change request

The Change Request Assign/Modify screen is accessed via the main screen **ChangeRequest / Assign/Modify** menu option. This screen is most often used by the Generic Engineer or other project leads with the authority to assign work. The screen is also accessible off the the main screen active CRs list by right-clicking a Change Request.

The display is controlled by the three radio buttons to the top left and then the filter selections to the right.



The initial selection is related to how the screen is activated. If the screen is accessed off of the menu system then the default activation mode will be **Active CR**s for the current **Generic**. If accessed off of the Active CRs list, then only that CR will be presented (under the "**selected CR**" filter).

Those CRs that need Generic Engineer input are presented under the **To Be Assigned** selection. Those CRs currently assigned for work are shown under the **Active CR**s selection. CRs that have finished their active life in SpectrumSCM are archived under the **InActive CR**s selection, specifically these are the CRs in the **Completed** and **Killed** states.

The arrow indication shows the current sort key and direction. If you select on a different column, that column will become the sort key. If you select a second time, the sort order will be reversed as will the arrow displayed in the header.

Users with the appropriate permissions can assign single, or multiple CRs simultaneously, simply by selecting those CRs and then self-assigning or assigning to another user.

In the bottom half of the screen, in **the CR History tab**, each of the state transitions that the CR has been through is shown (CR History). This includes who made those changes and when. If an entry here is selected any notes associated with that transition will be displayed in the lower panel. If required, the value of the CRs attributes can also be modified. This is done by selecting the attribute value displayed (choosing the **CR Attributes tab**) and selecting the new value from the presented choice box. Once the change has been made the "Save Attributes" button will be enabled. Please press this button to save the previously selected changes.

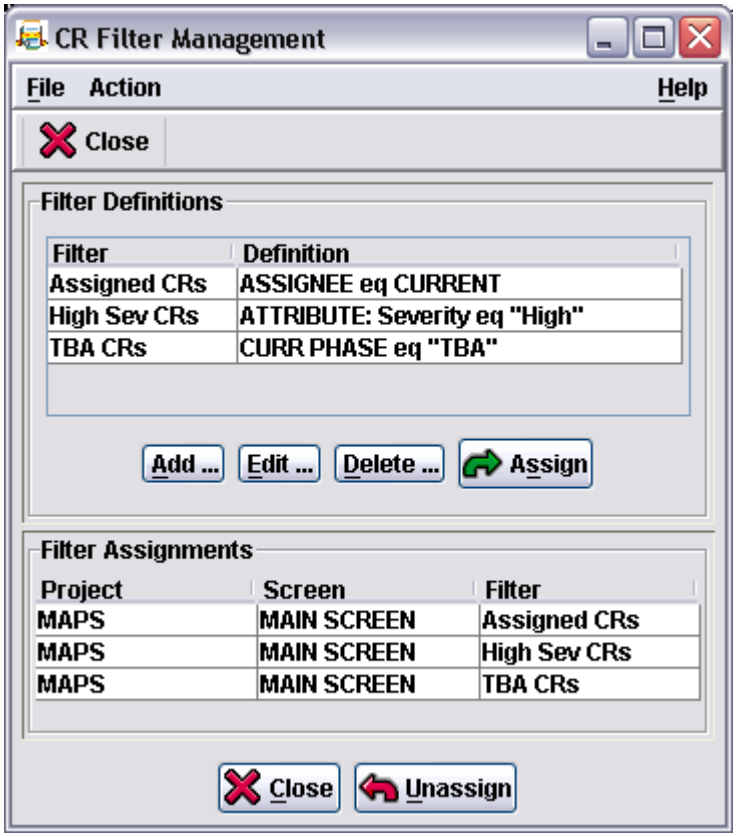
WBS tab added to Assign/Mod – so the full details of the work-breakdown-structure can be viewed in report form. In addition, a **WBS column** indicates whether a CR has WBS

relationship(s) or not. Further, in the main screen message area – a simple single click on a Change Request will display whether it has any WBS dependencies.

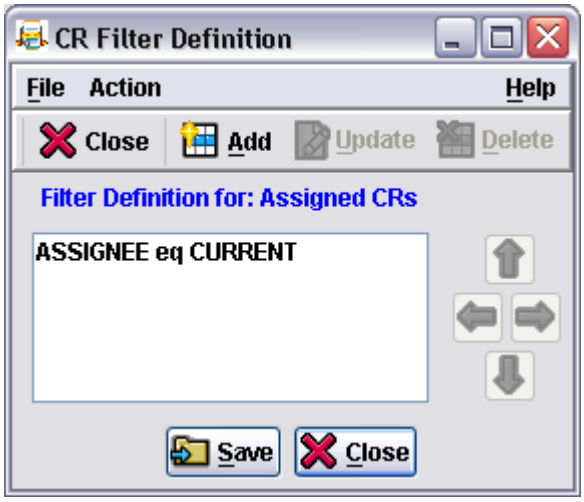
The user can switch back and forth between **CR Attributes, CR History and WBS Report Tab** by choosing the appropriate tabbed pane at the bottom of the screen.
Note that at this point, CR attachments for the selected CR can be added, deleted or viewed.

7.7 CR Filtering

To help in managing large numbers of issues you can define CR filters. When selected, a filter will then apply to the TBA, Active or InActive CR selection and only those CRs that match will be displayed. By default the system will present "**All**", "**Selected CR**" and a list of the current generics, in the filter choice box.
Selected CR will restrict the view to the single CR selected in the main screen assigned CRs list.
To define a filter (or modify an existing one), select the "**Filters…**" button, or select "**CR Filters**" off of the main screen Change Request menu.

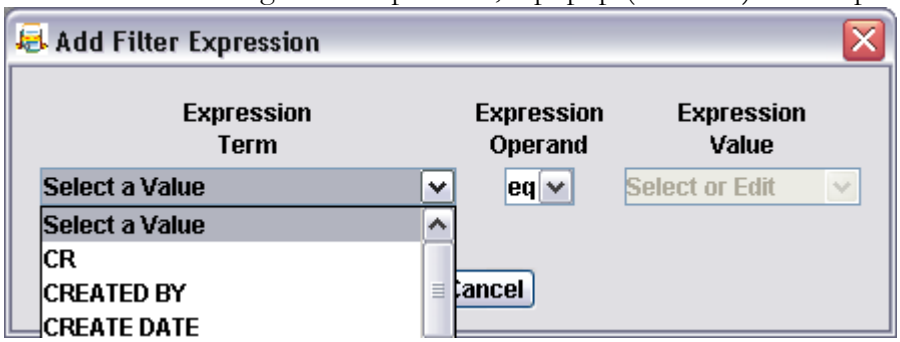


You can add, edit or delete a particular filter definition. You can also assign a particular filter definition to a project/screen combination. In this way, you can define a filter once and use it many times, in a quick, easy but controllable fashion.

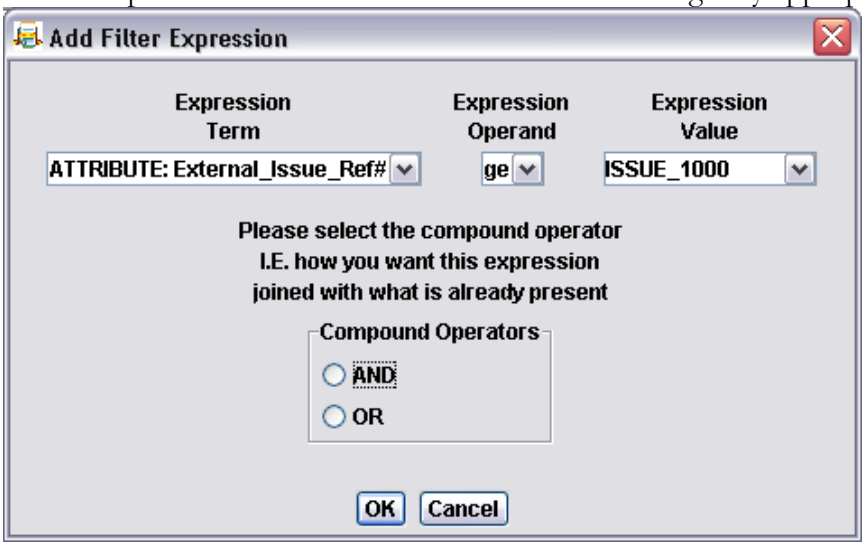


To create a new filter, select the "**Add…**" button, enter the desired filter name. The screen will come upblank for a new filter, or will show the existing filter definition if one exists. From here you can **Add** new rows to the definition, **Update** existing rows, or **Delete** unneeded rows.

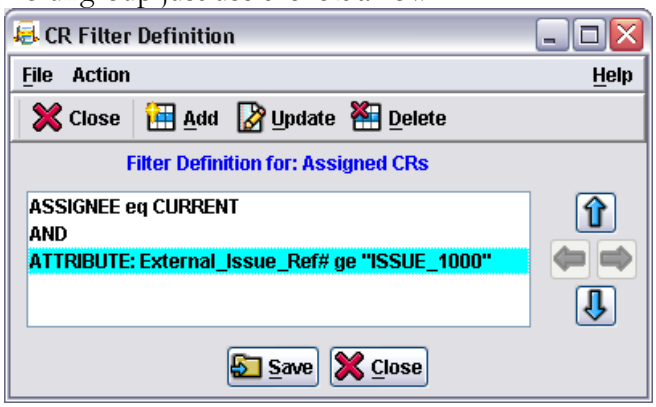
To add or edit a single row expression, a popup (as below) will be presented. Select the expression term (the attribute to be compared). The value (what to compare the attribute against). And the operand i.e. how to do the comparison.

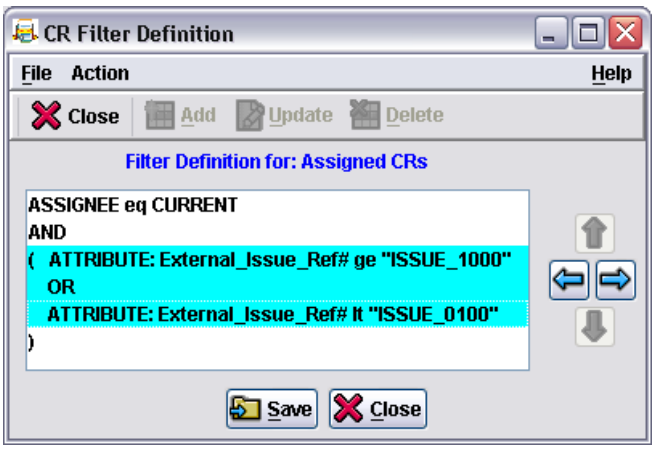


If this is a second or subsequent row you will also be asked how this expression is to be joined to the first expression row. Select either **AND** or **OR** as logically appropriate.

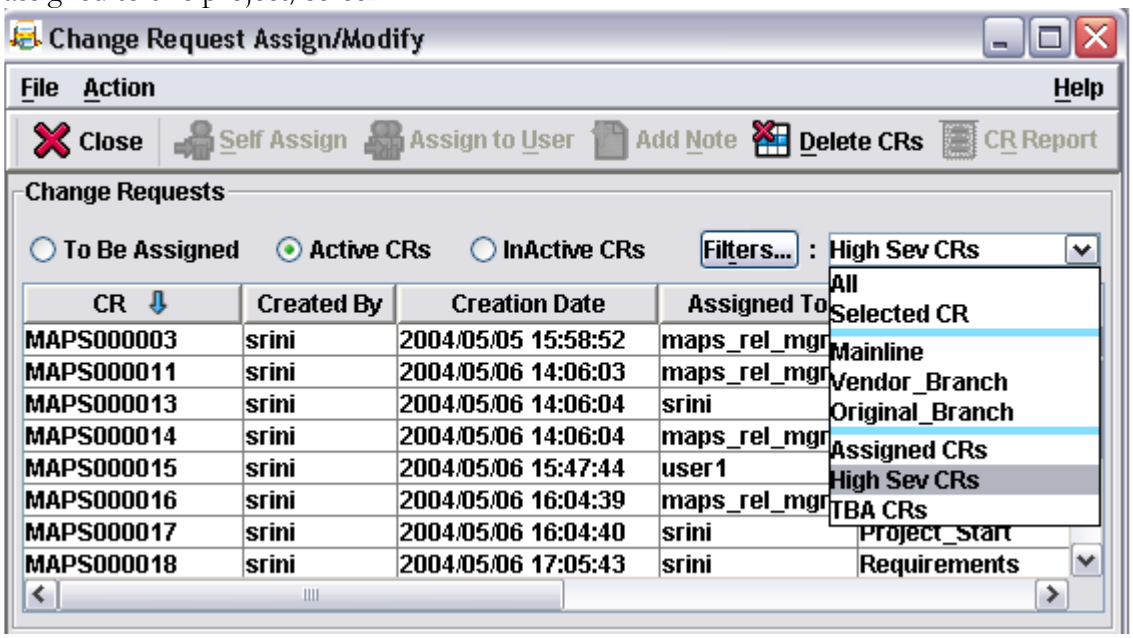


If you have multiple rows defined in your expression you can also re-order the rows by simply selecting the appropriate row and then the up or down arrow. If you want to group certain parts of your expression you can do so by selecting those rows to be grouped and selecting the right arrow. To ungroup just use the left arrow.





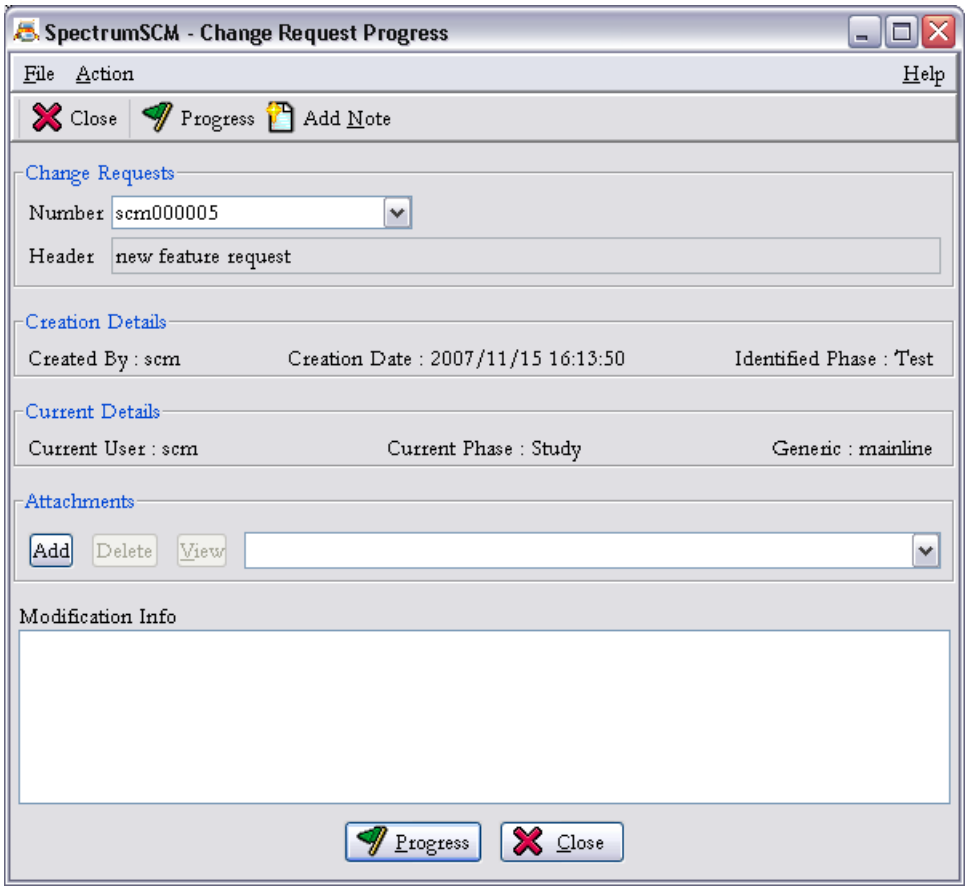
When your edits are complete save them, or you can cancel out and not save them if you so desire. Remember that you also need to assign this filter to a project/screen before you will be able to use it. Currently the CR Filters are supported on the CR Assign/Modify screen and the main screen Active CRs list. On the CR Assign/Modify screen, in the filters choice box there are three sections delimited by a light-blue line. The first section is the pre-defined filters, "All" and "Selected CR", the next section is the list of generics for this project, and the third section is the list of filters assigned to this project/screen.



7.8 Progressing Change Requests

The power of SpectrumSCM in managing workflow is in the Change Requests and the ability to move, manage, and monitor them through the defined process phases. When a user has finished with an assigned task, he or she will ***progress*** the CR to notify the generic engineer (or the system, if an automatic workflow has been set up) that the CR is ready to be assigned to the person responsible for handling the next life-cycle phase (for example, when development is complete, the CR is progressed to testing and assigned to a tester).

When a CR is progressed, information or notes can be added to describe what has happened during the phase. For example, if the CR was in a study state then the notes should include the results of that study, what action is recommended, and what documentation was produced. If the CR was in a development state then the notes could include some description of what has been changed and any areas of specific interest or difficulty. Any new or updated attachments can also be recorded (test reports, screen snapshots for user guide updates etc).

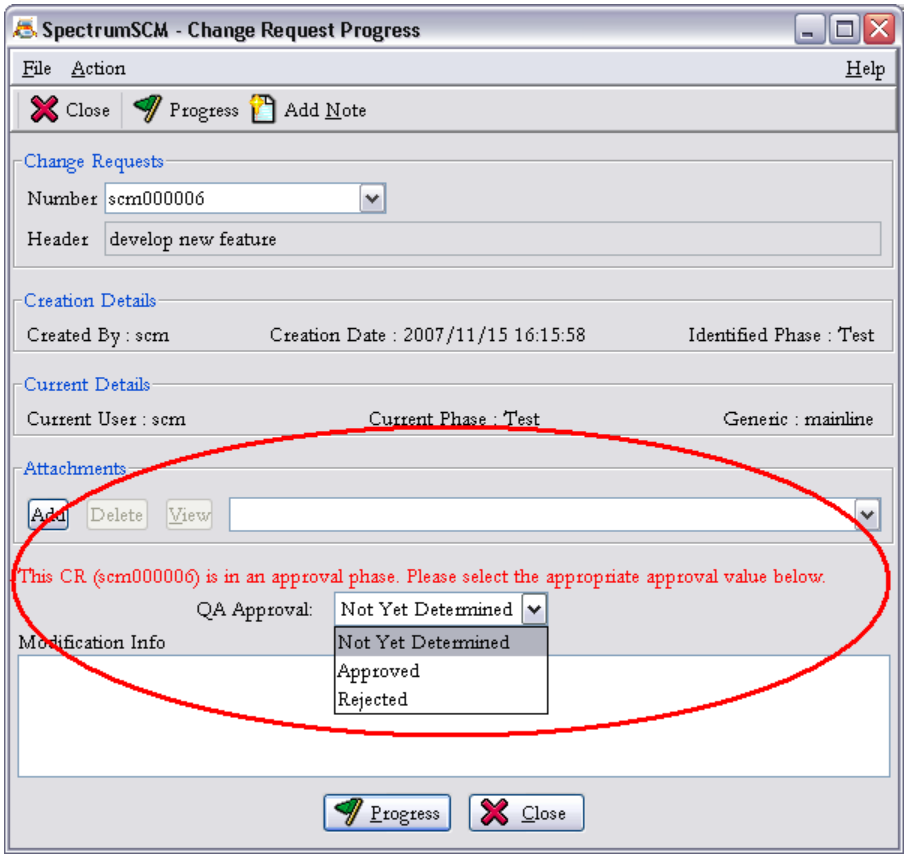


Once a CR has been progressed, what happens depends on whether an automatic workflow is in-place or not. By default the CR will be placed in the **To Be Assigned (TBA)** state and will show up on the Generic Engineer's screen with a yellow T indicator  . If an automatic action is specified for this transition the CR will skip the TBA state and be directly assigned to the user responsible for the next phase in the workflow i.e. the test team leader for testing CRs etc.

If the CR is in the TBA state the Generic Engineer will then assign the CR to the next phase and to the appropriate project team member. Depending on the roles and permissions, the project manager, project lead or lead developer might also have the ability to assign CRs. (*See Chapter 5 for details on user management, roles, and permissions).*

NOTE: If a CR is assigned in error, it can be reassigned and notes recorded about the error, but the erroneous assignment will remain in the history records for the CR. Start and Stop dates (from assignment to "progress the CR") are kept for workflow analysis. Dates will show if the error was quickly detected and corrected or if it caused a delay.

If the CR is being progressed from a phase that is marked as an Approval Phase (*see Chapter 6 for workflow definition and approval attributes*), the user will be prompted for the approval/rejection. This will then be recorded against the CR in the history section.

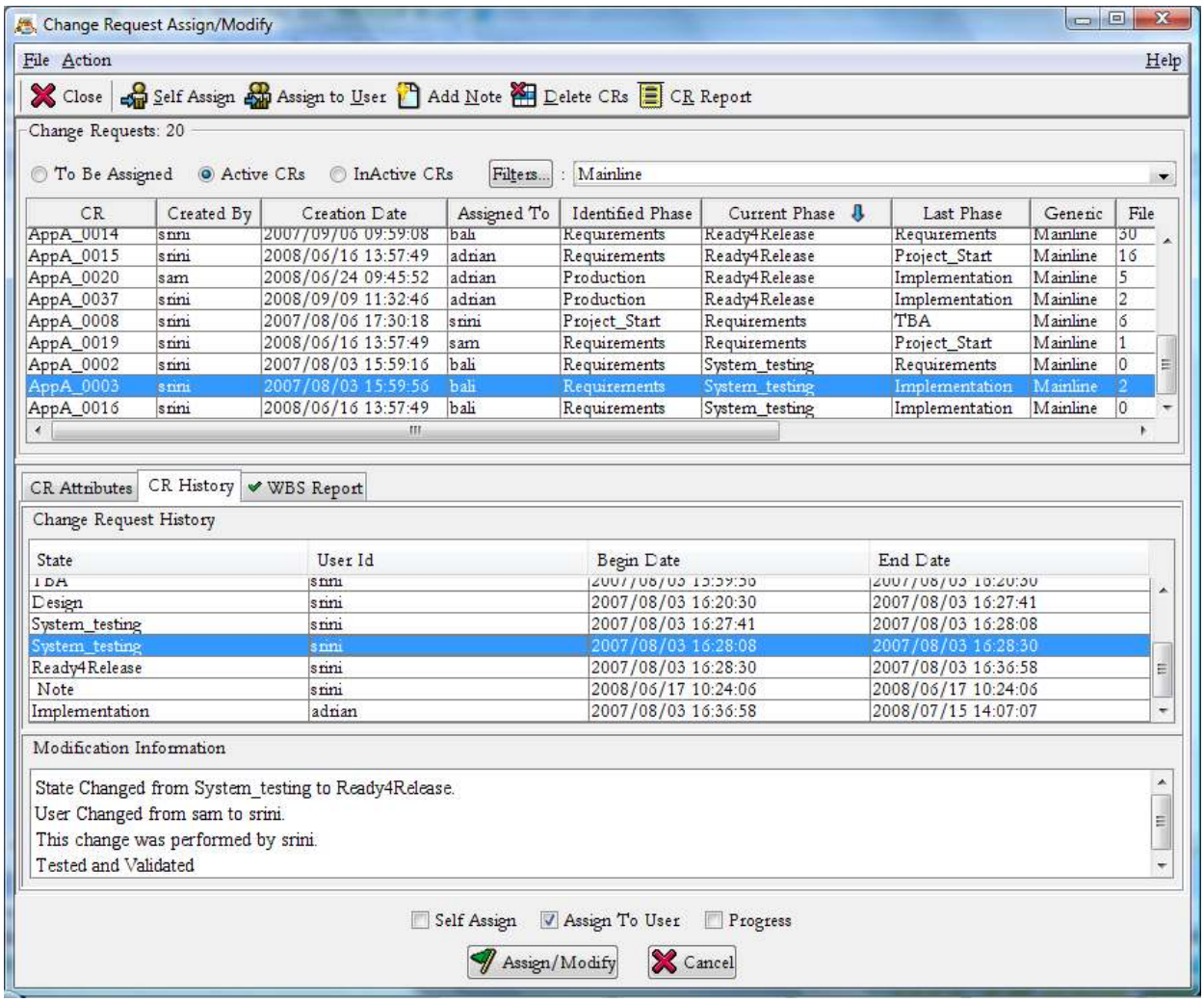


7.9 Display CR Details and History

The details and history of a CR can be displayed by running a Change Request Report by double-clicking on the CR or via the Main Screen Reports / All reports menu option or from the Change Request Assign/Modify screen.

Through the Assign/Modify screen details can be viewed interactively in the lower part of the screen. The attributes, header, description, and any attachments are available under the primary "CR

Attributes" tab. The state transitions and any notes made are available through the "CR History" tab.



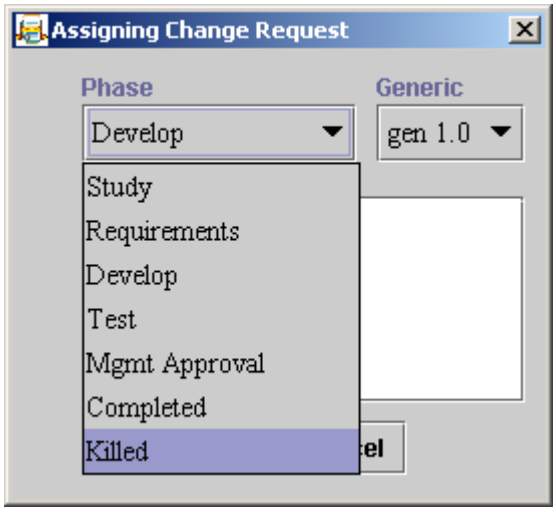
Reports can be run on TBA, Active, or inactive CRs. Reports can be viewed online, printed or saved to a file. *See Chapter 10 for details on Reports.*

7.10 Killing Change Requests

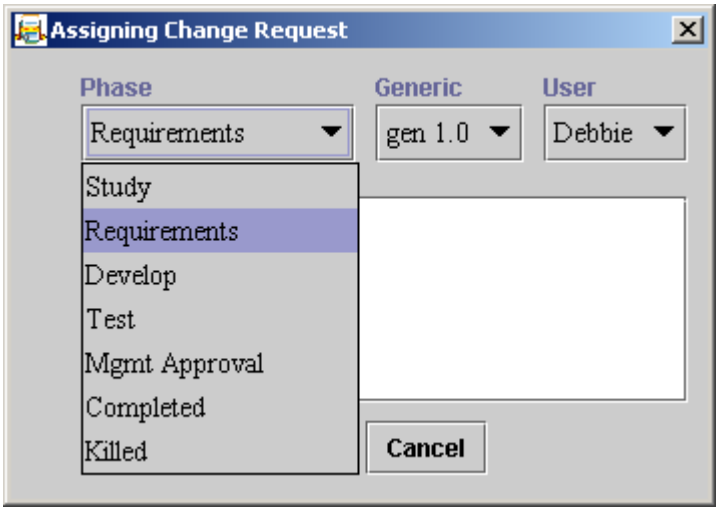
CRs that are added in error, duplicates, or those deemed to be inappropriate for any reason can be "killed". "Killed" is just a special state defined and aligned with the InActive category. Modification information or notes can be added to describe why the CR was killed.

A CR cannot be killed if there are any file level modifications associated with the CR. If the CR is really to be killed, those file modification would either have to be moved to another CR through the File Relocator (Main screen -> CR menu), or the file modifications would have to be deleted (if permissible, using the File Version History -> Rollback and Delete option).

Killed CRs are inactive and are therefore not assigned to a user. All killed CRs can still be viewed and modified via the Change Request Assign/Modify screen.



A killed CR can be revived (in case it was a mistake or you change your mind). On the Change Request Assign/Modify screen, select inactive CRs. Select the CR to be revived and click Assign/Modify. Select an active phase and assign the CR to a generic and a member of the project team.
Click OK and the CR becomes active again.

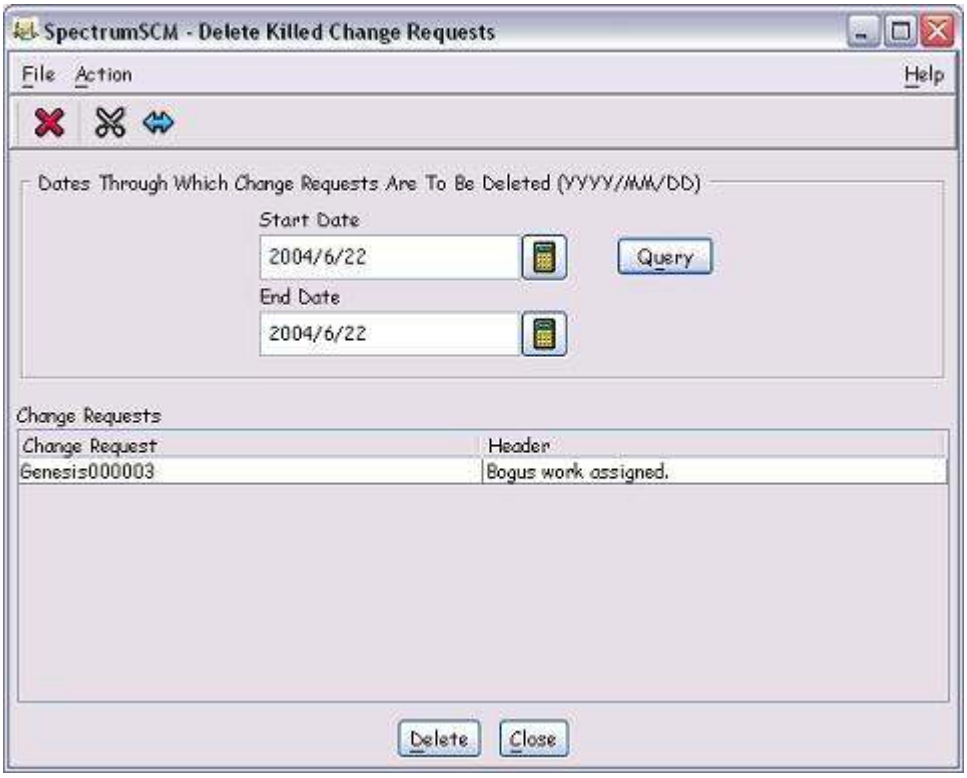


7.11 Deleting Killed CRs

Eventually, killed CRs pile up and you will want to clean them up.

Note: A CR cannot be killed if there are any modifications associated with the CR; therefore, a CR with modifications associated with it cannot be deleted.

To delete killed CRs, bring up the **Delete Killed Change Requests** screen using the ***Action -> Delete Killed Change Requests*** menu option off of the CR Assign/Modify Screen.



Select the date range for the Killed CRs that you want to delete (last year's for example). The specific date that is queried is the date that the CR was killed (inclusive). The format for the date is "YYYY/MM/DD" or it can be chosen from the calendar selector buttons.

Select a date range, and then use the **Query** button to display the killed CRs matching the date range. Then you can select specific Killed CRs to delete, multiple selections are allowed. Select the CR(s) to delete and press the **Delete** button. Be careful - this function permanently removes the CRs from the system and they cannot be re-activated.

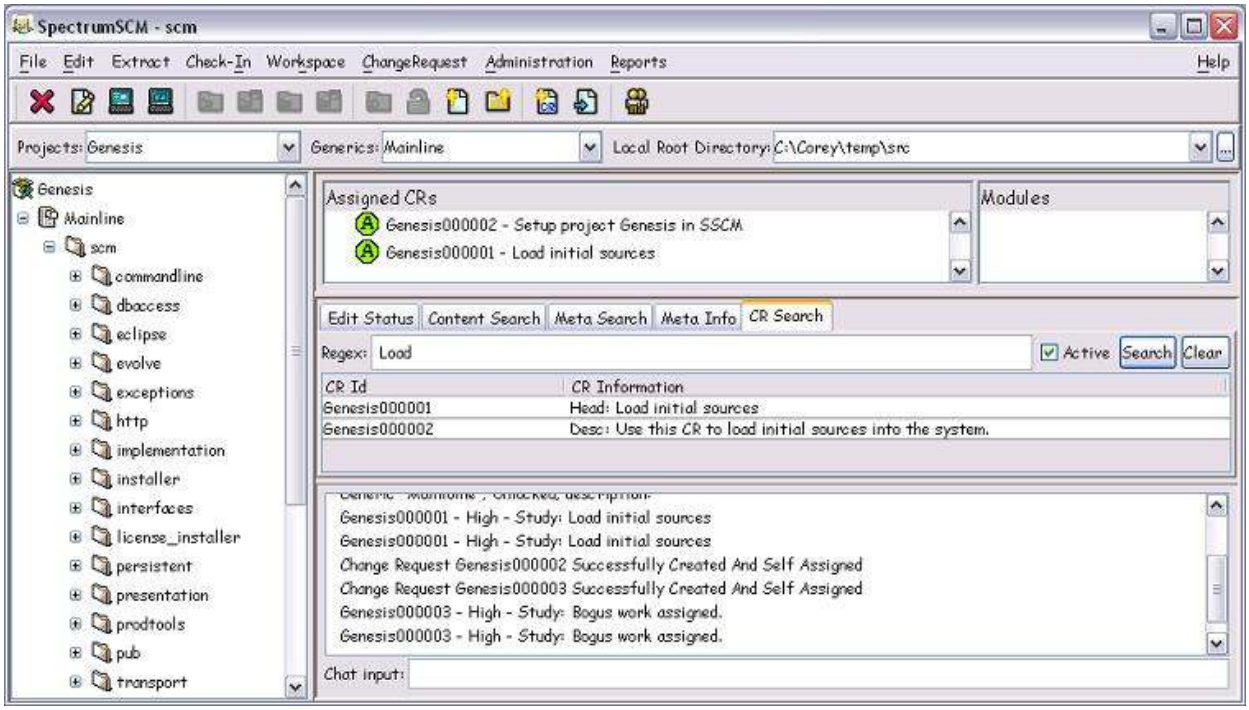
7.12 Searching Change Requests

In the Middle Panel of the SpectrumSCM main screen, the CR search tab can be used to search for CRs by either CR number or content.

CR Search provides the capability to search the change requests for any text matching the requested search pattern. Control is defaulted to only search for active CRs but can be expanded to search all CRs (active and inactive).

NOTE: On a large project, searching all CRs could involve a lot of work and therefore a response might take a while. Be patient!

Once a search has been run, you can select a result line and double-click to run the full CR report for that change request. Alternatively, you can right-click and bring up the Assign/Modify screen if you have the appropriate permissions.



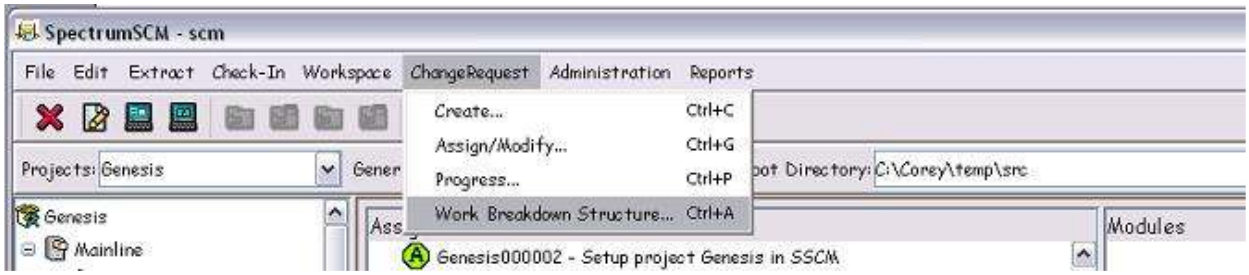
7.13 Change Request Work Breakdown Structures

Change Requests in SpectrumSCM can be grouped together in parent/child or peer/peer relationships. This allows project leaders to developer high level CRs as complete features, requirements or bug fixes that can then be broken down into a cascade of smaller specific CRs for assignment to individual users. The relationships between the CRs are enforced at the release level. This means that a high level CR in a parent/child relationship cannot be added to a release unless all of the CR's assigned children are either in the release or are ready to be placed in the release.

Peer to peer relationships are similar in nature. Peers are tied together so that any one of the CRs cannot be placed into a release without the others.

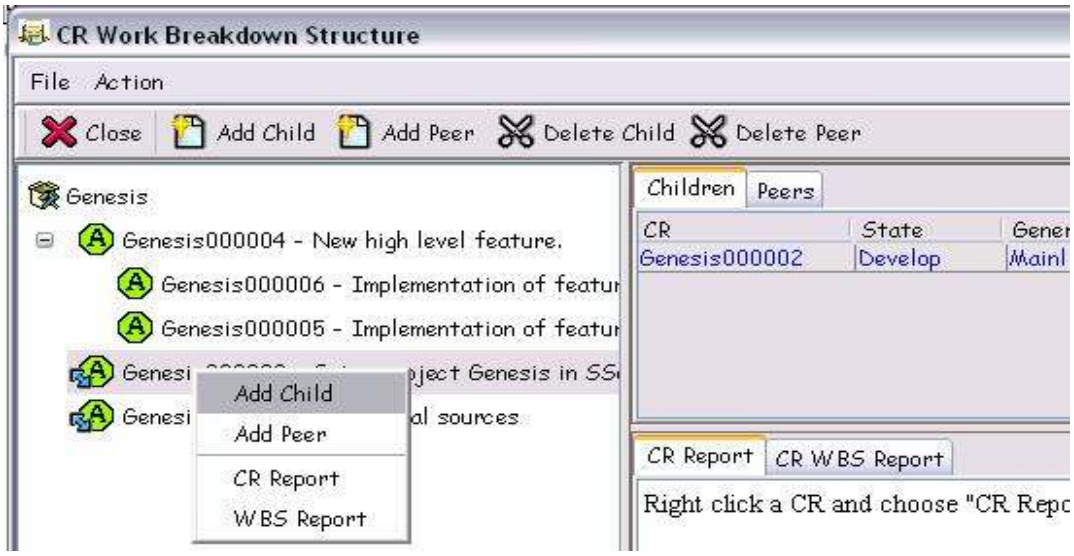
New You can also establish **parent-as-peer WBS option** to further tighten work-breakdown structure definition. This relationship means that you get the strength of the **peer-to-peer** relationship and maintains the hierarchical organization of the **parent-child** relationship.

The Change Request Work Breakdown Structure screen provides a project level admin user the ability to make and break Change Request relationships. To access the WBS screen, use the ChangeRequest / Work Breakdown Structure… menu item.



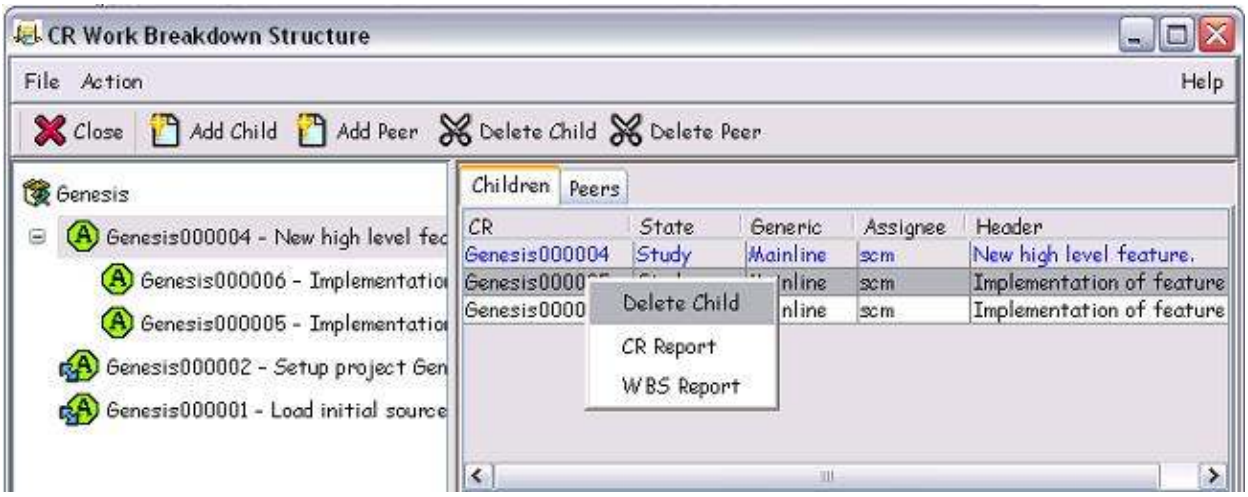
The WBS screen has three main sections. On the left is a tree view of the CRs and their relationships. On the right are two panes, one displays the WBS relationships for the selected CRs in a top-right table view. The lower-right portion of the screen displays CR reports output.

In this example, CR #4 is the parent of CRs #5 and #6. CRs #1 and #2 are tied together in a peer to peer relationships. CRs involved in peer to peer relationships have a small blue left right arrow icon associated with their normal icon. CRs involved in parent/child relationships will be displayed as a tree view with an expansion + or – sign. Peer CRs can only be defined at the top most level. That is, CRs already involved in a parent/child relationship cannot also be involved in peer/peer relationships.
To add children to a parent or to add a peer to another CR, right click on the parent or peer CR to bring up the context sensitive menu system.



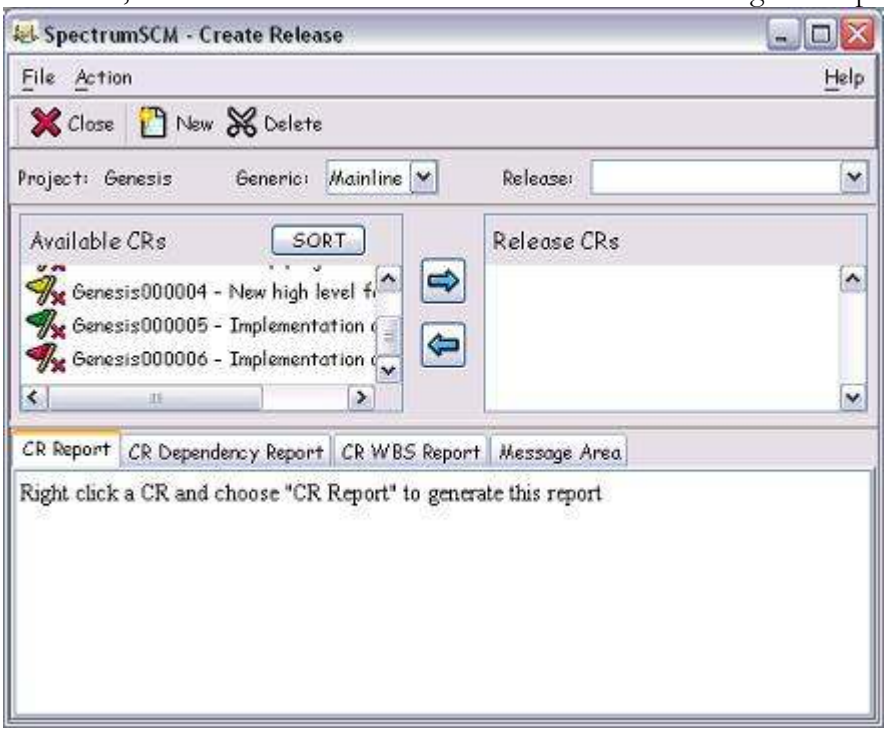
The CR and WBS reports can also be run from the context sensitive menu. When the reports are executed, the report output will show up in one of the two tabbed panes at the lower right of the screen. If a custom report viewer is defined (*See chapter 4 for details on user preferences*) the report output will be displayed in the custom report viewer instead.

Relationships between CRs can be broken at any time. To delete a child from a parent child releationship, select the parent CR in the tree view on the left and then click the "Children" tab on the right hand side of the screen. Now individually select the children to be removed from the relationship and then press the "Delete Child" button on the top of the screen, or right click to get the context sensitive menu.



Work Breakdown Structures are enforced in the release management screen (see Chapter 9 – Release Management). Just like file level dependency checking within CRs, the release management screen also does CR level dependency checking. The color coding is exactly the same for CRs regardless of whether a dependency is formed at the file level or at the CR level. However the tooltip will indicate which type of dependancy exists against a yellow flagged CR.

In this example CRs #4, 5 and 6 are involved in a WBS with CR #4 acting as the parent. CR #5 is



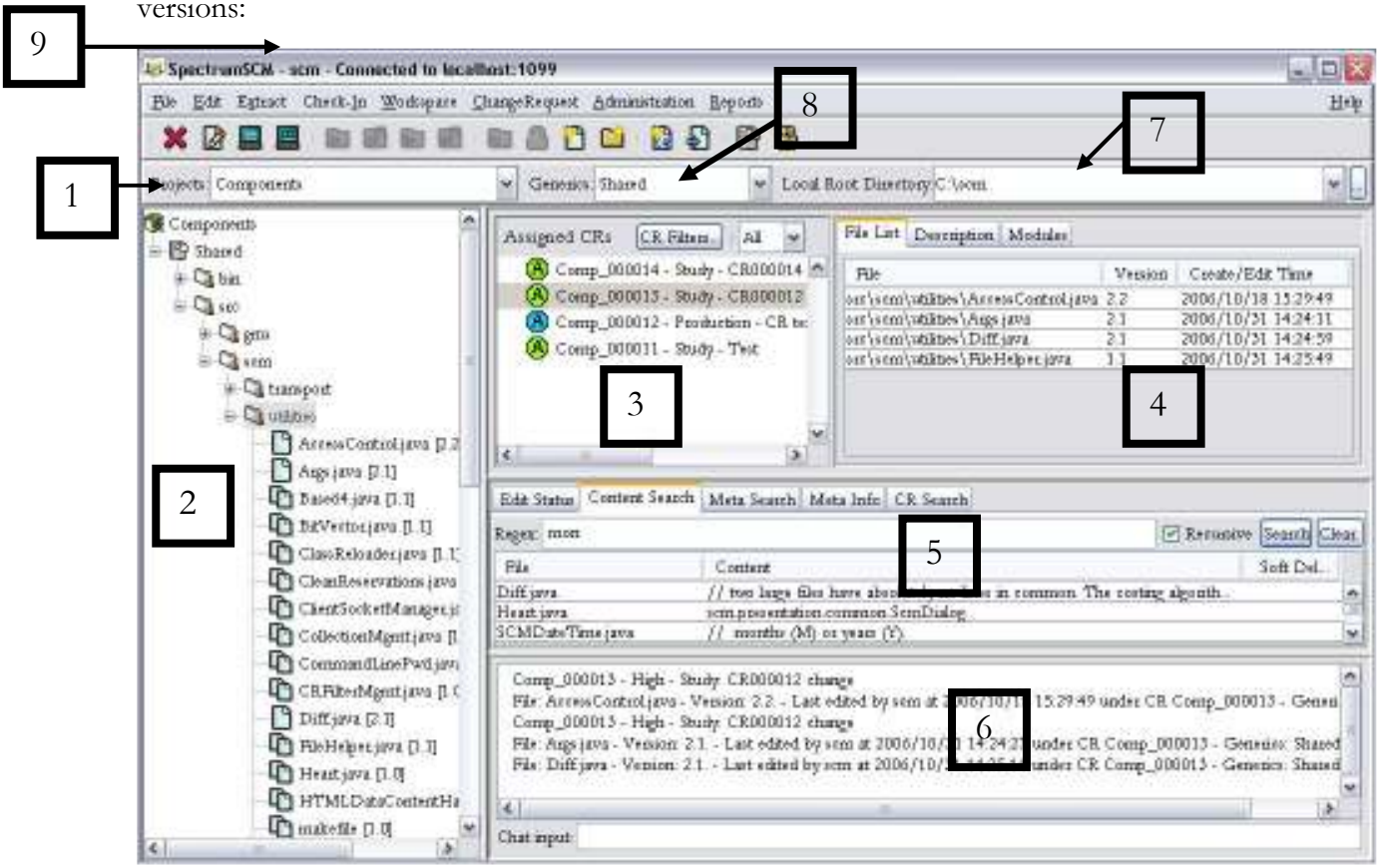
complete and ready to be assigned to a release while CR # 6 is red-flagged, because it is incomplete, and not ready for assignment. The parent CR has been marked as completed but is yellow flagged due to the CR level dependency between it and CR # 6. The parent CR #4, cannot be placed into the release until all of the children CRs are green-flagged. CR level dependencies can intermix with file level dependencies too. To distinguish between the two, run both the WBS report and the CR Dependency report against any yellow flagged CR to determine the reason why it is yellow flagged. The WBS and CR Dependency reports can be accessed by right clicking the CR to activate the context sensitive menu system.

8 Version Control and Source File Management

Chapter 8

This chapter describes how file version control is managed by SpectrumSCM, how to check-out or extract files for read only or edit purposes, check-in files that were checked out, check-in a new file, unlock a file, load an entire source tree, do bulk check-in / check-out, and edit files using the default SCM editor or a custom editor.

The Main SpectrumSCM screen provides the functions for managing source files and versions. Be sure you are familiar with the areas and functions that are used for managing source files and versions:



Below the icon bar is the project bar (1). The project bar contains the project selection box, the Generic Selection box (8) and the root directory box (7). The local root directory is used to control where files are checked out / extracted to and from where new files are taken to load into

SpectrumSCM. The contents of the root combo box are modifiable by the user and the content of the box is saved from session to session.
The file tree (2), located on the left-hand side of the main screen, contains the current view of the entire project.

Right of the file view is the change request display window (3) and right of that is the file list/Description/module display window (4). Only change requests that are assigned to the current user and project are displayed in the change request display area at any time. The module feature is used to group sets of files together so that they can be operated upon with a single-click.

Directly in the middle is the current edit display window (5). This window displays all files that are currently checked out. In this example, one file is out for edit to the desktop.

On the very bottom of the screen is the chat component and system message area. All system related messages are displayed in the message area (6).

At the top right is the local root directory (7). Always be sure the local directory is set properly for the work you are doing. Files are checked into the specified directory, uploaded to SpectrumSCM from the specified directory, and loaded from the specified directory. A user may have multiple local root directories for different purposes (the 5 most recently used are remembered).
See Chapter 4 for details on all areas and functions available on the main screen

The Toolbar provides many functions you will use when working with source files:

- **New Editor** - Startup a new empty editor panel.
- **Single Editor** - Startup an editor panel on the file selected in the *Project Tree*.
- **Multi-Editor** - Startup a dual editor panel on the file selected in the *Project Tree*.
- **Check-out Read-Only To the Desktop** - Start an editor panel on the selected file in Read-Only mode.
- **Check-out Read-Only To the Disk** - Write a Read-Only version of the selected file to the local disk. The full path is determined from the *Local Root* and the directory the file resides in, in the project.
- **Check-out for edit To the Desktop** - Start an editor panel on the selected file in Live-Edit mode. Note that an Assigned CR must be selected for this operation to proceed.
- **Check-out for edit To the Disk** - Check-out a writable version of the selected file to the local disk. Note that an Assigned CR must be selected for this operation to proceed.
- **Check-In** - Check-in the selected file. The file selection can be made in the *Project Tree*, the *Module* window or in the *Edit Status* middle panel. Note desktop edits must be checked in from the desktop editor.
- **Unlock** - Unlock the item selected on the *Edit Status* panel.
- **Add a File** - Add a small set of files (individually) into the SCM system.
- **Add a set of files** - Add a set of files identified by a directory and some filters. This feature can be used to load a whole project tree into SpectrumSCM.
- **Create a new Change Request** – Add a new Change Request to the system and assign it either to yourself or to another user. The ability to create and assign CRs depends upon your role and access permissions.

- **Progress a Change Request** – Progress a Change Request into the TBA (To Be Assigned) state.
- **Generic Viewer** – View the current set of generics and their heirarchy. Also provides switching, comparison and modification functionalities.
- **User List** – View a list of users currently logged into the system.

The Middle Panel has 5 tabs to show the status of files currently checked out for edit, and to allow various search capabilities.

On the **Edit Status** tab, files that are currently out for edit by you are indicated. The display includes the date and time that the file was checked out, which change request and generic are associated, and where the edit is being performed (desktop or file-system).

The contents search panel and the meta search panel enable searching of the project source files. The search will cover all the appropriate files contained in the selected directory (from the project tree). A **contents search** will search the text files and report on those that contain matches against the requested search pattern.

A **meta search** is controlled in the same way but searches the meta information associated with a file instead of its contents. Meta information is most useful when considering binary files such as drawings or pictures that would otherwise not be searchable. Meta information can be established for any file with a description of what that file contains or any other useful text. In the case where SpectrumSCM is being used as a documentation control library, then the meta information could be used to store key-words. Meta information is maintained through the use of the Meta Info tab, based on the selected project tree entry. When a file is added into SpectrumSCM its meta information is initialized to its filename and path. This enables files to be found by name in even the largest of projects simply by searching the meta information.

8.1 Checking-in or Adding New Source Files

New files can be checked in or files that have been checked out for edit can be checked back into SpectrumSCM. Source files can be added into SpectrumSCM individually or en-mass (via load source tree). The file(s) to be loaded must be under the local root directory, in a subdirectory that either corresponds to one that already exists in the project tree or you want created in the project tree. All files checked in must be associated with a CR.

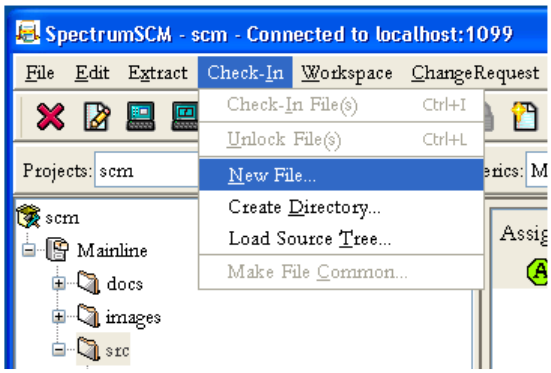
Alternatively you can use the **DragNDrop feature** available in SpectrumSCM to check-in or load a single file or multiple source files by simply **Dragging** the files that are of interest directly from desktop or file system and **Dropping** it into appropriate folder location on the project repository window panel.

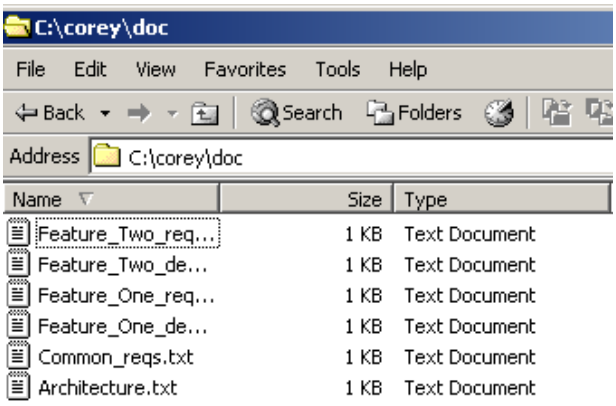
This feature basically automatically populates the add new source file screen with the proper source and target location without the user having to set the local root directory settings.

8.1.1 Check-In a new file

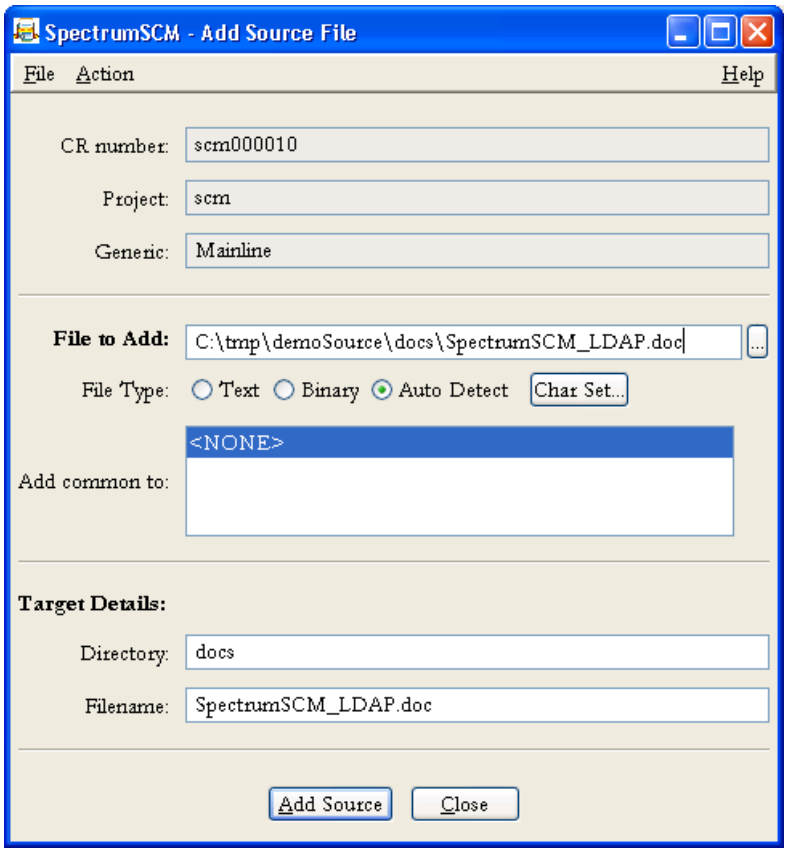
To check in a file created outside of SpectrumSCM,

- The file can be dragged from anywhere on your file system, must be in the local root directory, in a subdirectory that corresponds to one in the project tree (in the example below, the file is in the local root directory C:\corey, subdirectory \doc. It will be loaded into the project tree into the Genesis Gen 1.0 doc folder)
- Select a CR from the **Assigned CR** list.
- Select **Check In → New File**





The Add Source File screen will be displayed.



Click the file browser (. . .) and select the file to add. It must be in the specified local directory, under a file structure similar to that in SpectrumSCM. For example, to add the SpectrumSCM_LDAP document to the system. They are in the local directory C:\tmp\demoSource\docs folder. With the local root set to "C:\tmp\demoSource" this means that the file will be added into SpectrumSCM under the "docs" folder.

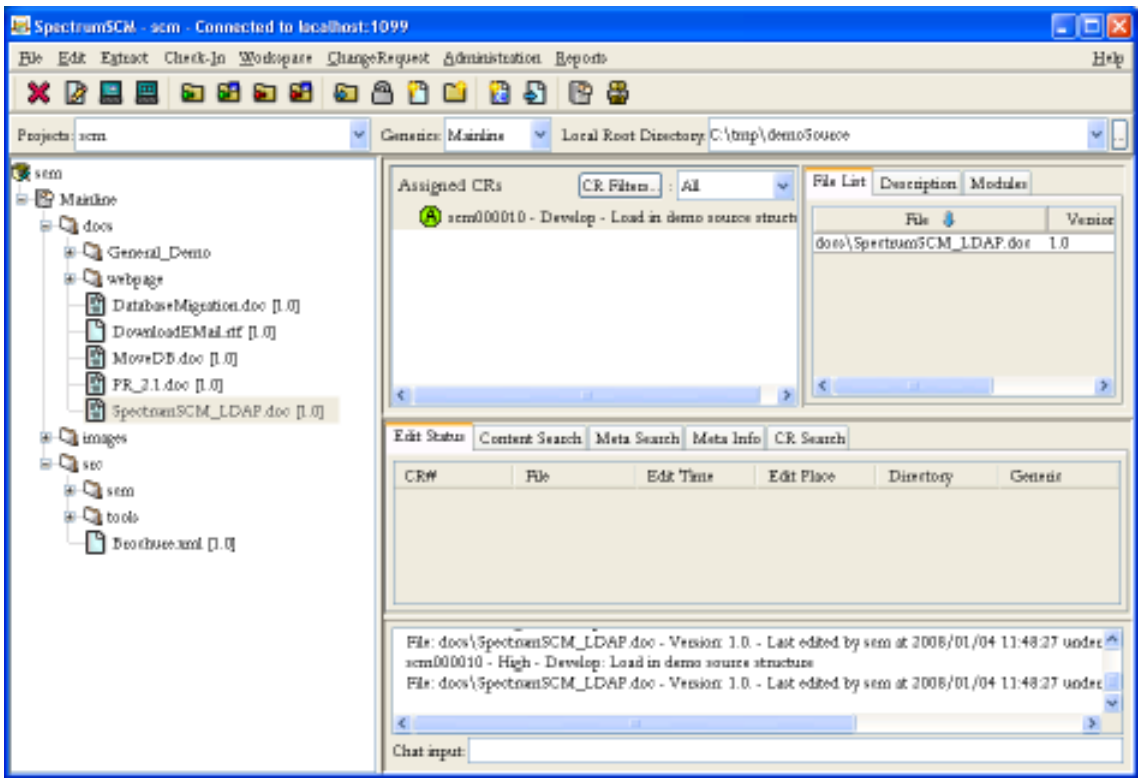
By default files are automatically evaluated to determine whether they are text or binary, relative to the SpectrumSCM system character set. The system character set is either the default operating system character set from the SpectrumSCM server, or the character set value specified through the server configuration wizard. Individual files can be stored under SpectrumSCM with different character sets simply by selecting the appropriate value under the "Char Set" button. In most situations users do not need to be concerned about character sets, however international and other technical situations can introduce this factor.

Select **<NONE>,** or the generic(s) to make this new file common to. If you want to add this file common to a number of generics, you can multi-select them from the list using the *shift* key.

Click Add Source.

A confirmation is displayed

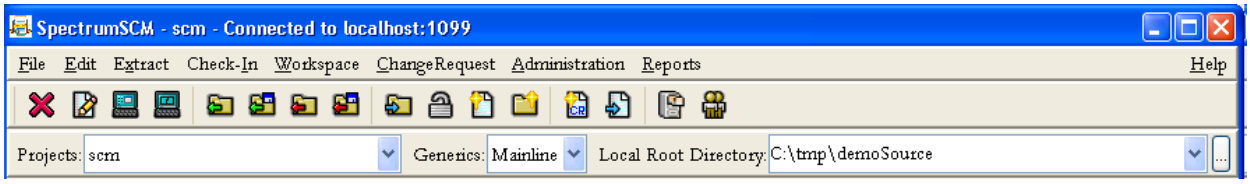
Multiple files associated with the same CR can be added, either one at a time using the **Add Source** button, or by multi-selecting the files through the browse button. To verify that the source file was added correctly and associated with the desired generic and directory, refresh the UI with updated data from the server (**Main Screen / Edit / Refresh Project**) and examine the SpectrumSCM project tree. When you close the Add-Source window this refresh is performed automatically.



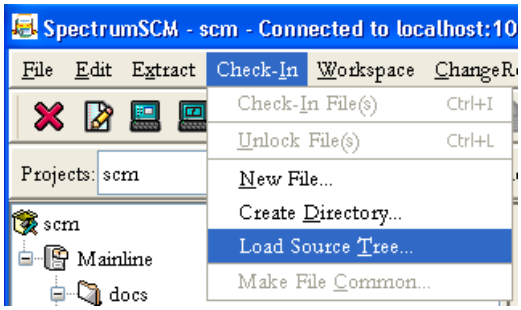
8.1.2 Load a Source Tree or an entire directory into SpectrumSCM

The **Load Source Tree** menu option is used to load an entire directory structure and its contents into SpectrumSCM. Subdirectories can also be recursively loaded if desired. The subdirectories can be empty, to establish the project tree structure, or they can contain the baseline files. The **Load Source Tree** function is also used to migrate structure and files from another CM tool or file versioning system.

On the main screen, select or create an appropriate CR for loading the source tree, make sure you are in the right project and generic and that the local root directory is set correctly (this is the directory (or above it) that contains the source tree you wish to load).

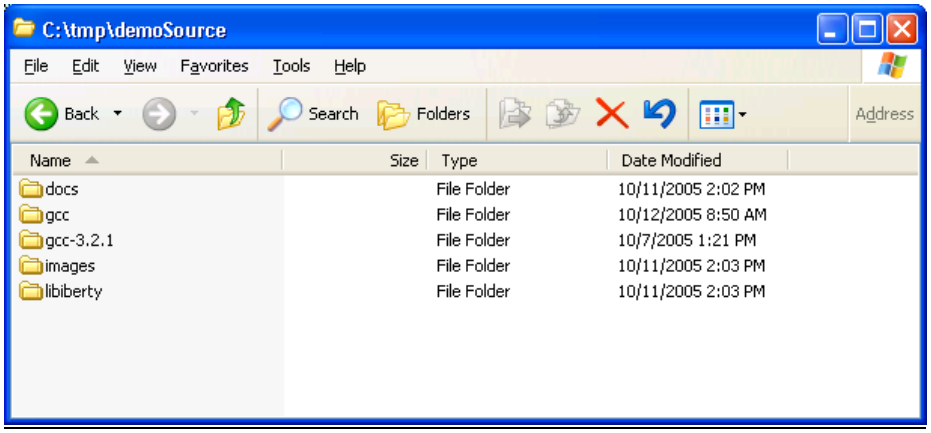


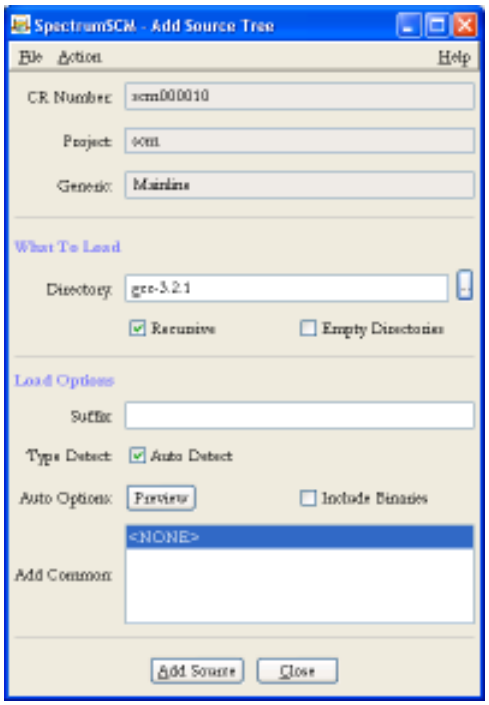
Then access the Add Source Tree screen via the **Check-In / Load Source Tree** menu option.

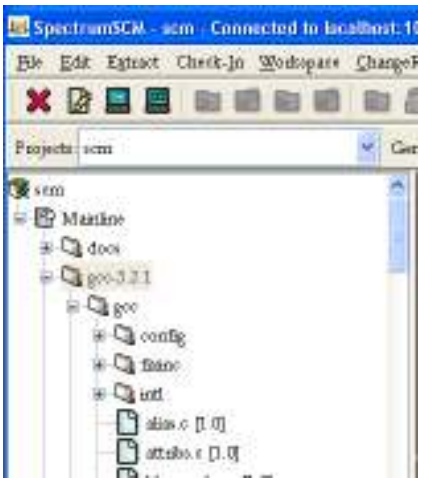


This will bring up the **Add Source Tree** screen. The **Add Source Tree** screen allows the user to add an entire directory structure of files into the system all at one time.

NOTE: The source tree directories and file must be in the local root directory. In the example shown, the local root directory is C:\tmp\demoSource. All subdirectories and files in them, if any, will be loaded into the project tree.







The directory to be loaded is specified via the *Directory* text field or chosen via the browse button (...). The directory must be under the local root directory. Using "/" (or "\", either works) loads the entire local directory.

The SpectrumSCM system will look in the local directory for the specified directory and copy its contents into the SpectrumSCM system under the corresponding directory, generic and project. The structure to be loaded must be under the current user's specified local root directory. If you need to load from another location, change your local root directory.

File selection can be delimited by suffix. If the **Suffix** field is blank, all ASCII files will be loaded. Binary files will also be loaded if the **Include Binaries** checkbox is selected. If a file suffix is specified (like ".java", ".C" or ".doc" for example) then only files of that type will be loaded.

After specifying the parameters of the load operation and clicking the "Add Source" button, a Load Source Tree Results window will be displayed indicating the success (or failure) of the file loads. To view the loaded files through SpectrumSCM close the Load Source Tree window and check the project tree for the desired results.

Click on the icon or + sign to open a directory and view the expanded project tree.
The icon or - sign indicates that the directory is open.

Alternatively you can use the **DragNDrop feature** available in SpectrumSCM to check-in or load a single folder or an entire folder structure by simply **Dragging** the files or folders that are of interest directly from desktop or file system and **Dropping** it into appropriate folder location on the

project repository window panel. This mechanism can also be used to check in individual files as well.

This feature basically automatically populates load source tree screen with the proper source and target location without the user having to set the local root directory settings.

8.2 Extracting or Checking-out files

A single file can be checked out or files can be extracted en-mass. Files can be checked out read-only or for editing. Only files checked out for edit can be checked back in directly.
Note: Files are checked out into the **local root directory** specified on the Main Screen. Be sure it is set correctly.

Simply double-clicking on a file in the repository view will open that file (read-only) for viewing. If you have a custom editor specified the file will be opened in the appropriate editor (See Chapter 5 for information on setting up user preferences which includes custom editor settings). If the file is a binary type such as a Microsoft Word document an appropriate custom viewer/editor will need to be established.

- **Check-out to Disk** - Presents options to check-out the selected file by version (read-only) or common/uncommon for edit. Files are checked out into the local directory specified on the Main Screen, ***the file remains checked-out until it is checked back in.***
- **Check-out to Desktop** - Presents options to check-out the selected by common, uncommon, for merge or recommoning. (See commonality discussion below). When the file is checked out, an edit session is directly opened. ***When the edit session is closed, the file is automatically checked back in to SpectrumSCM.***
- **Extract Files by Directory** - Select the required directory to extract from the *Project Tree*. That directory structure will be placed on your local hard disk under your local root directory specified on the Main Screen.
- **Extract Files by Release** - Select the release to be extracted from the presented window. That release will be placed on your local hard disk under your local root directory.

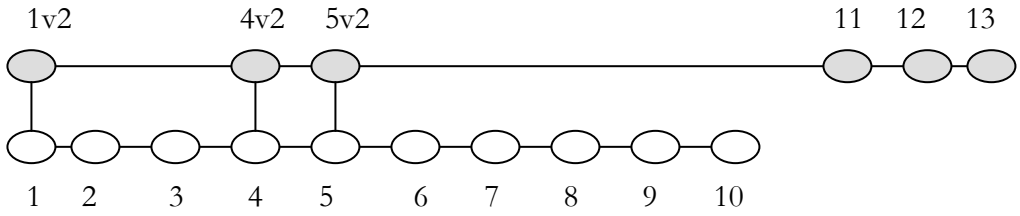
Alternatively you can use the **DragNDrop feature** available in SpectrumSCM to extract **(read-only)** files or folders by simply **Dragging** the files or folders that are of interest directly to the desktop or file system and **Dropping** it into appropriate folder location anywhere in your file system.

8.2.1 Common / Uncommon

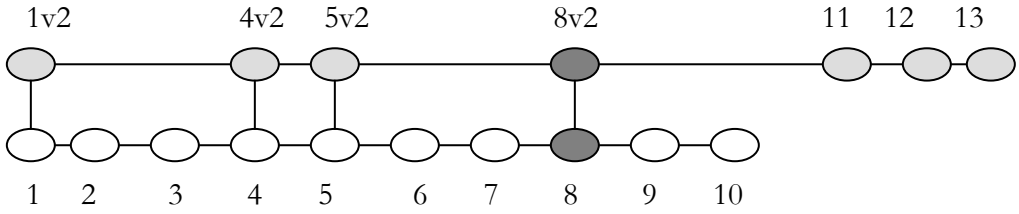
In overview, checking out "uncommon" will mean any file changes will only be made against that specific generic. If a check-out is performed "common" then the file changes will be made against ALL the generics that that file is currently in common with.

- **Common:** Versioned files that are physically the same across generics
- **Uncommon:** The act of physically separating versioned files from multiple generics

Checking out "common" is a powerful feature since it can be used to apply a single "fix" to multiple generics in one edit.
For a new development effort, the default check-out mode (uncommon) is all that is required and the developer need not be concerned about these issues. Checking out "uncommon" will mean any file changes will only be made against that specific generic.
If a check-out, edit, and check-in is performed "common" then the file changes will be made against ALL the generics that the file is currently in common with (as defined at project set-up; see Chapter 6 for details).
When a new release of the system is being developed, the changes and additions are based on the previous generic or release as defined on the Add Generic Screen". Changes and additions are made uncommon to the previous release. For example, if a project team has developed and released version 1.0 of a system and they are currently developing generic 2.0, all modules that are changed (modules 1, 4 and 5) or added (modules 11, 12 and 13) during the 2.0 development effort will be "uncommon" - changed only for generic 2.0.



However, if a problem is discovered in release 1.0, the fix for the problem might be made common to both generics. In this example, the problem is in module 8. The code is edited, the problem fixed and the fix is made common to both generic 1.0 and generic 2.0.



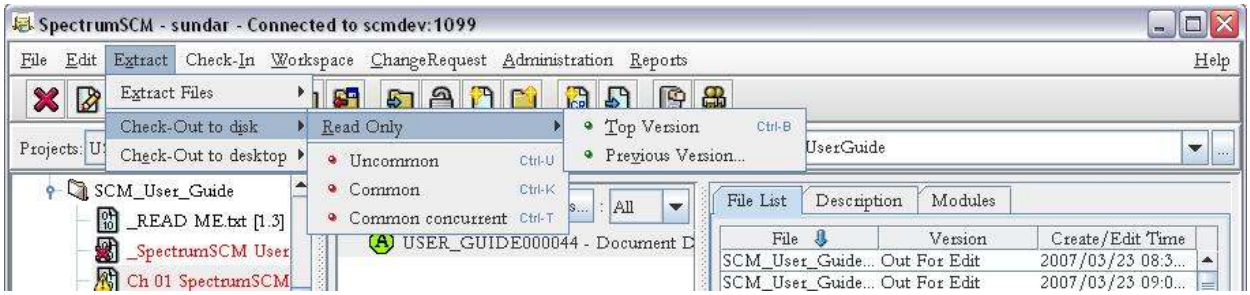
When multiple generics are to be used and developed in parallel (for example to maintain 2 similar source bases for 2 different customers), the Generic Engineer must decide who is going to make the branching decisions – who will determine which of the modules will be common or uncommon with other generics. The default mode (unlocked) leaves the choice with the developer to make as each module is checked out for edit, assuming the developer would best know the source issues. However to prevent inadvertent changes to other generics or to enforce process control issues, the Generic Engineer could lock the generic (via the Modify Generic Screen) which would make all subsequent edits be performed uncommon. The generic can be subsequently unlocked if a situation arose that required a common change.

Once the mode of the Generic has been established, if the commonality control is still with the developer (the generic is "unlocked"), then he/she can choose the appropriate check-out mode.

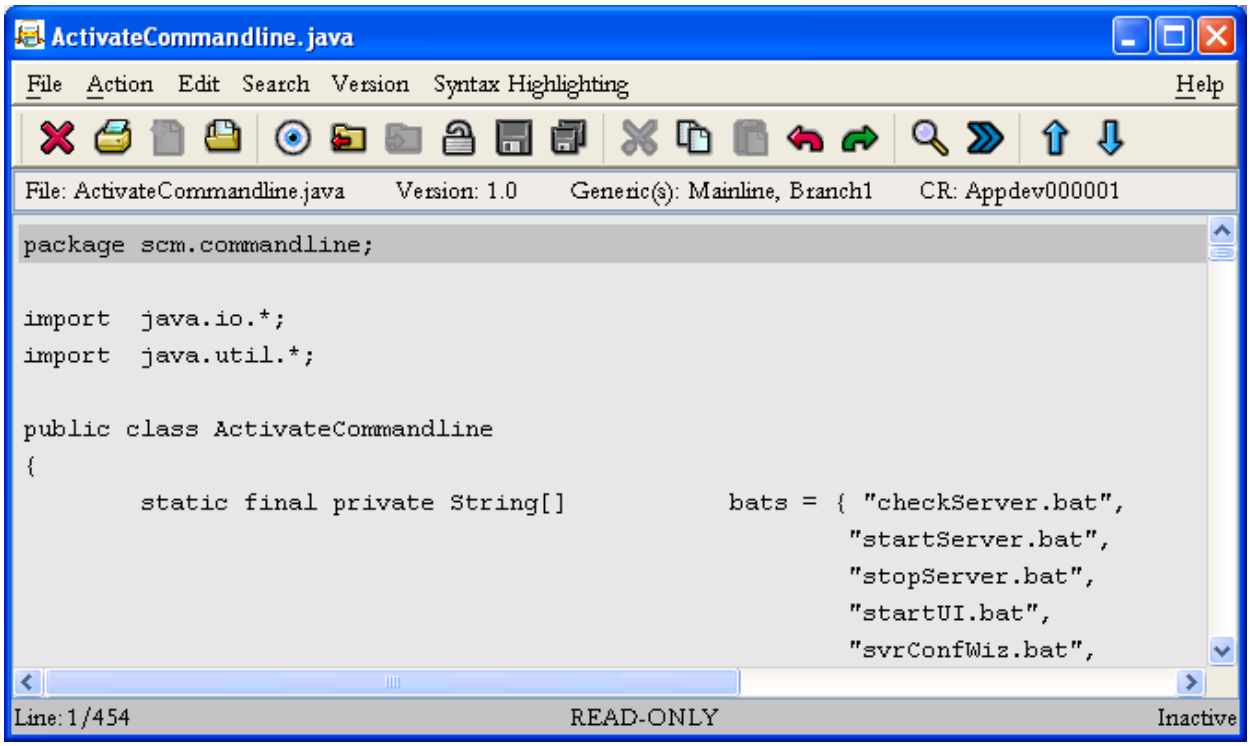
8.2.2 Checking out for read-only to desktop or disk

If you select a file without selecting a CR, read-only is your only option.
With read only, you can extract the top (most current version) or any previous version of a file. You might do this to compare the most current version with the previous version. You might extract a file read-only to use it as a model for a new file.

A file checked out read-only cannot be checked back in to the SpectrumSCM system.

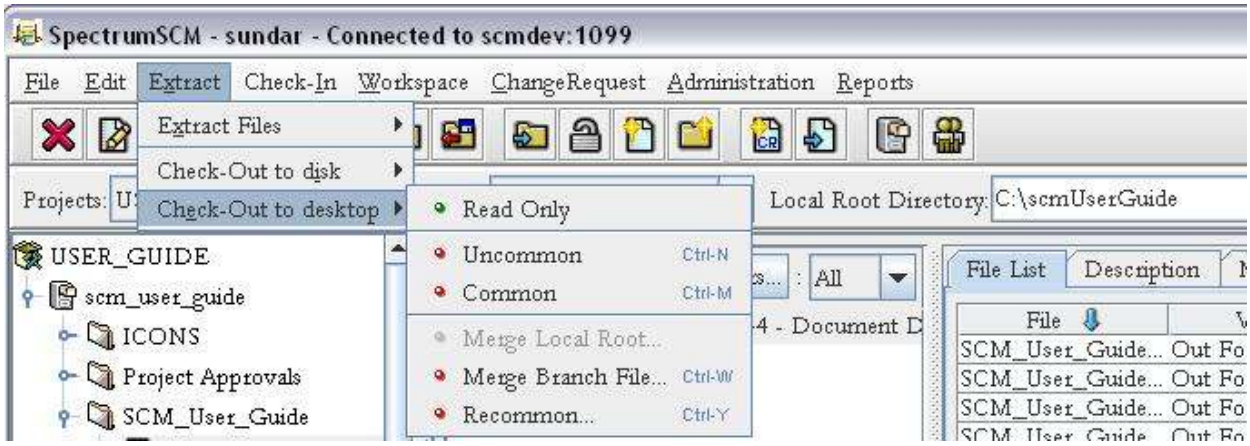


A file checked out read-only to disk is placed into the local directory and is marked as read-only. A file checked out read only to the desktop is brought up in the SpectrumSCM (or custom) editor. You can search the file, copy it, save it to the hard drive, and pull up another version. The file cannot be modified or checked back into the SpectrumSCM system.

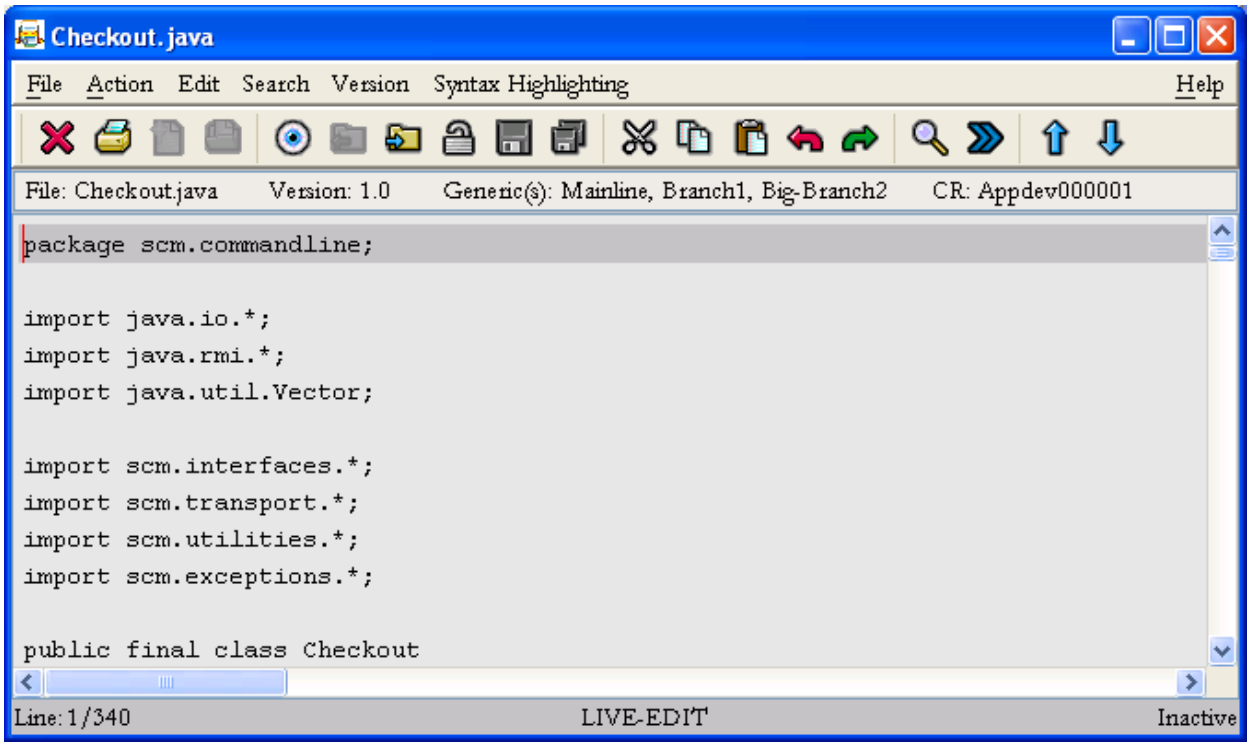


8.2.3 Check out to desktop for edit

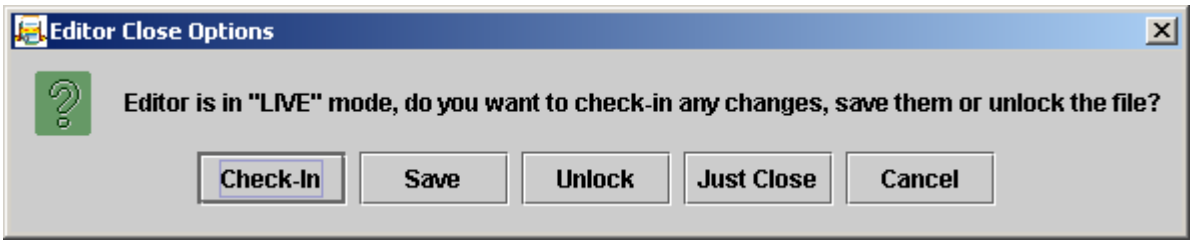
- Select **Checkout to desktop** -> **Common** or **Uncommon** to put the file in a write-able state in the editor. The file to be checked out and a CR must be highlighted on the Main Screen



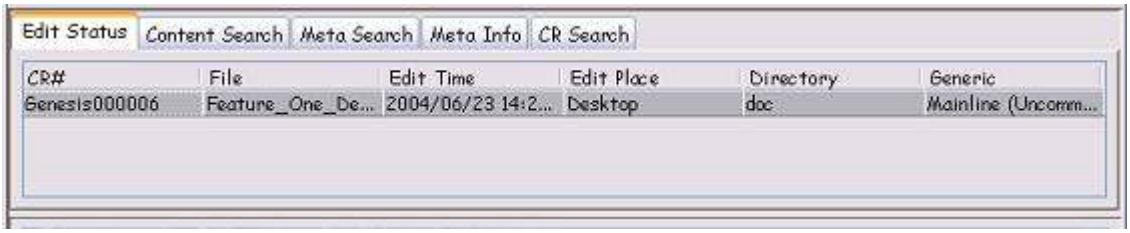
- The SCM Editor or custom editor will open the file for editing.



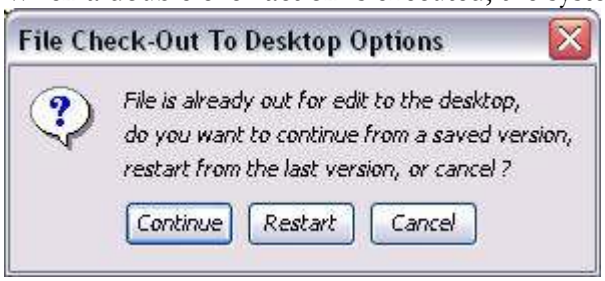
On closing the edit session, the system will present the user with options to check in, save to disk, unlock, or close.



If the file is checked in, a new version of the file is created in SpectrumSCM. Changes made to the file will be associated with the CR selected at checkout. The **Unlock** option causes the edit changes to be thrown out and the file will be unlocked from edit.
The **Just Close** option literally does that, it throws out the edit changes to the file, but leaves the file checked-out.
The **Save** option saves the edit changes back to the SpectrumSCM server. Once a file has been "saved", it will remain out for edit and the edit session can be continued or restarted by double clicking on the edit entry within the main screen. Note, if the edit is subsequently unlocked, the "save" will also be lost. This is because the "save" is associated with the edit session.



When a double click action is executed, the system will respond with the following popup window.



Selecting the **Continue** button will retrieve the file as saved off earlier, and load the file into the SpectrumSCM editor. If the **Restart** button is pressed, the previously saved changes are thrown out and the original content of the file will be loaded into the SpectrumSCM editor. The **Cancel** button simply dismisses the popup.

8.2.4 Merge and Recommon options

The "Checkout to Desktop / Merge" feature is used to merge changes among generics (branches).
The "Checkout to Desktop / Recommon" feature is used to recommon files among generics.
(See Chapter 11 for details on using the Merge Editor for Merge and Recommon functions)

8.2.5 Check out to disk for edit

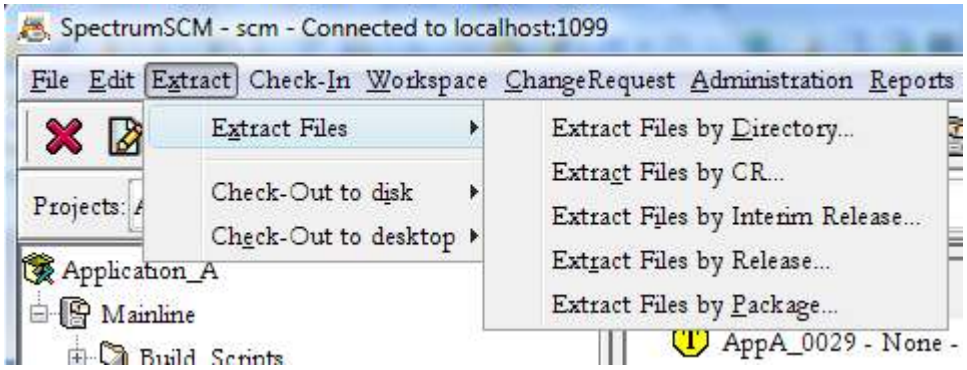
Select **Extract / Check out to disk / Common, Uncommon** or **Common Concurrent** to put the file in a write-able state on the local disk in the specified local root directory.

- Edit the checked-out file using an editor of your choice. Note that if you have your custom editors set up (see Chapter 5 – User Management), you can simply double-click on the entry in the edit status panel and your preferred editor will be directly opened on the disk file.
- Check-in the file when editing is completed. Changes made to the file will be associated with the CR selected at checkout.

8.2.6 Common Concurrent

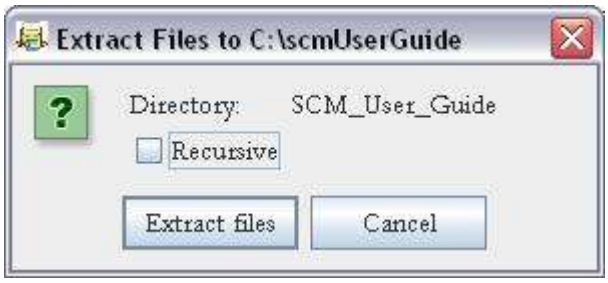
Checking out to desktop or disk as "**Common Concurrent**" will put a soft lock on the file. Other users can check out the file, make updates, and check the file back in. When a file checked out as common concurrent is checked back in, the system will check for modifications made by other users. If there have been other modifications, the file is hard-locked and the user who checked out common concurrent is required to merge the versions using the spilt screen merge editor. This feature allows concurrent editing, but adds some extra work to merge the changes.

8.2.7 Extract Files By Directory

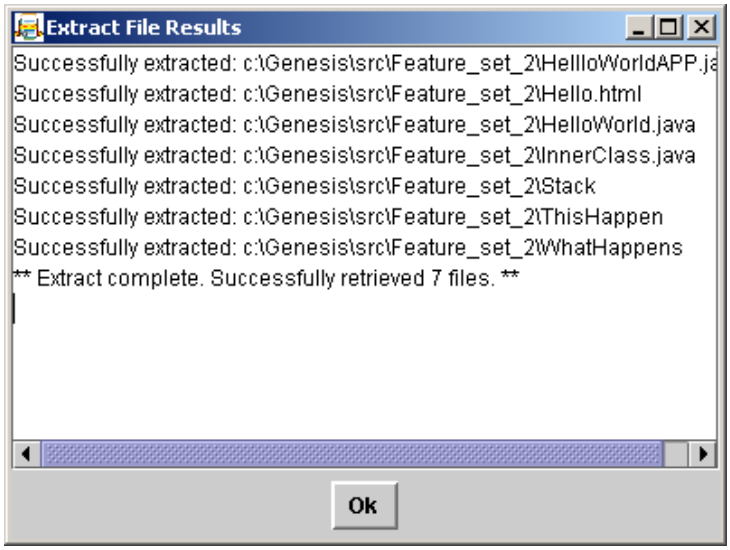


From the Main Screen, select the directory to be extracted from the project tree and then menu item.

Extract -> Extract Files by Directory.
When presented with the confirmation screen -
Choose **Recursive** to extract all subdirectories and files. Otherwise only the files directly inside the selected directory will be extracted.

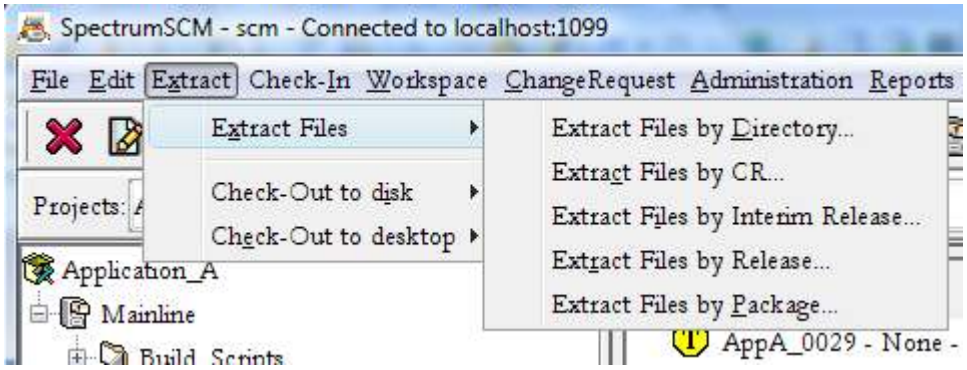


Files are extracted and placed into the user's local root directory.



8.2.8 Extract Files by Release

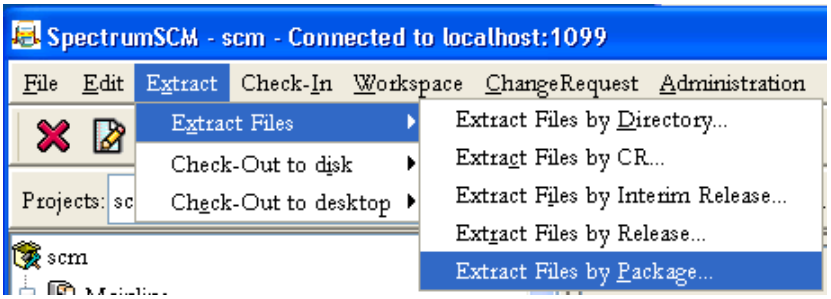
This option can be used if there is a previous release of the project (*See Creating a Release in Chapter 9*). The entire project tree associated with a release is extracted and placed into the local root directory.



This feature can be used to extract files to create a new release or re-create a previous release. A Bill Of Materials report (BOM) can be produced at the time of extract, this lists all the files and their versions that make up this release.

8.2.10 Extract Files by Package

This option can be used to extract previously defined packages (*See Creating a Package in Chapter 9*). The entire source set associated with the package is extracted and placed into the local root directory.



This feature can be used to extract files to create a new release package or re-create a previous packages. A Bill Of Materials report (BOM) can be produced at the time of extract, this lists all the files and their versions that make up this package.

8.2.10 Extract Files by Interim Release

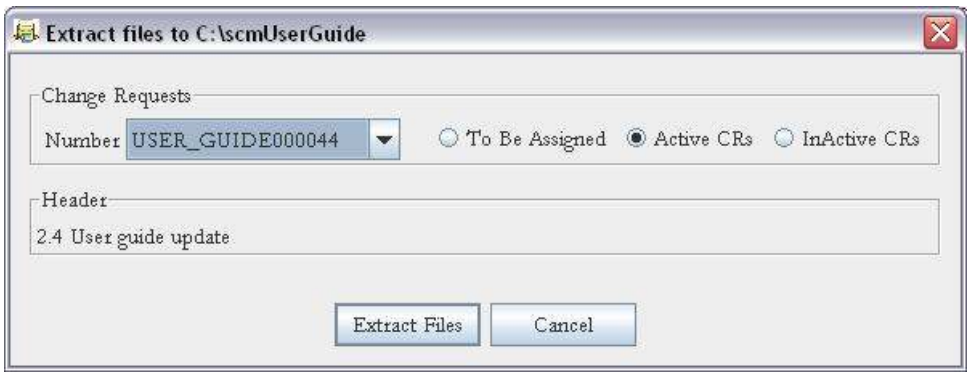
You use **Interim Releases** to extract a set of files based on a phase of your life-cycle. This feature is intend to allow you to form "test" releases where many but not all issues have been resolved, and so a formal release cannot be built yet. For example, if you have an "Integration Test" life-cycle phase as part of your process, but yet developers can still modify files under these CRs until things stabilize and become ready for the formal release.

See Chapter 9 for more details on this feature.

8.2.11 Extract Files by CR

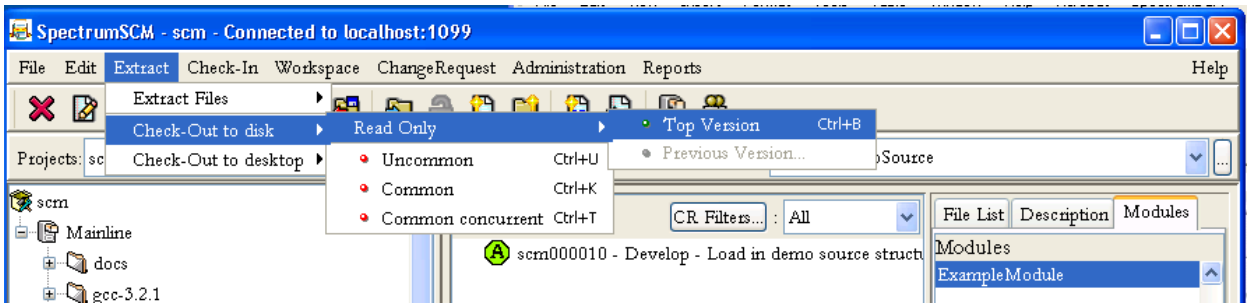
With this option you can extract just the files that were changed under the selected Change Request.

Select the CR that you are interested in and press the "**Extract Files**" buton. For example, if CR USER_GUIDE000044 changed 5 files, only those 5 files would be extracted (into your current local-root directory).

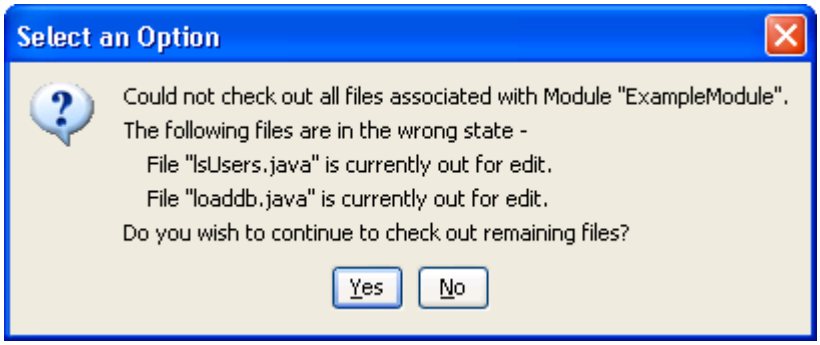


8.2.12 Extract by module

A module is a defined group of files. All files associated with a module can be extracted.
Select a CR with which the work is associated
Select a module from the module list
Select a function
In this example, the user is checking out read-only all files associated with module "ExampleModule", to disk. Note, this option (and the option to check-out for edit) is also available by right-clicking on the module name in the modules tab.



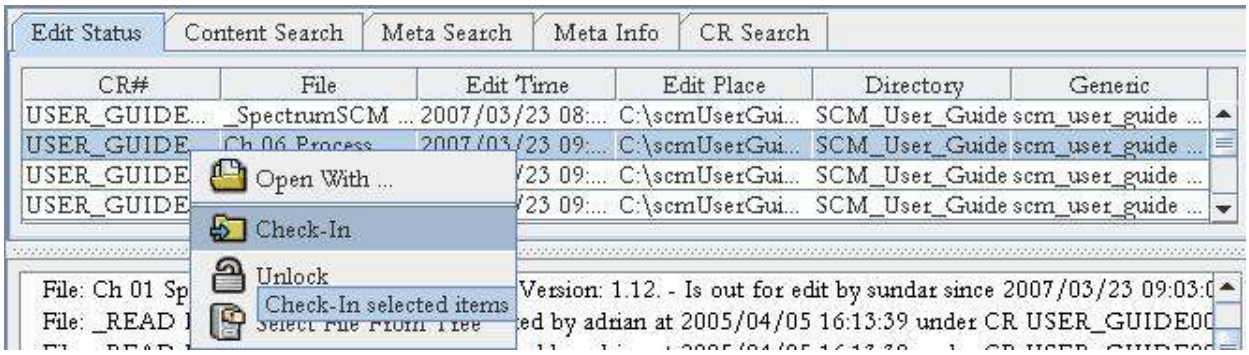
If the files are being checked-out for edit and they are not all available, an error will be displayed and the user given a choice whether to proceed.



8.3 Checking in files

8.3.1 Checking in a file that was checked out to disk for edit

If the file was checked out to disk, the system will check it in from the local directory.
There are three different ways to check in a file or set of files.
Check in from the menu – Select the file in the tree view and then select **Check-In** -> **Check In** from the main menu bar.
Check in from context menu – Right click the file in the tree view and select **Check-In.**
Check in from the Edit Status Panel – Single select or multi-select a series of files in the Edit Status panel and then right click to bring up the context sensitive menu. Choose **Check-In.**
All files selected for check in will be associated with the CRs that were used to check the files out.



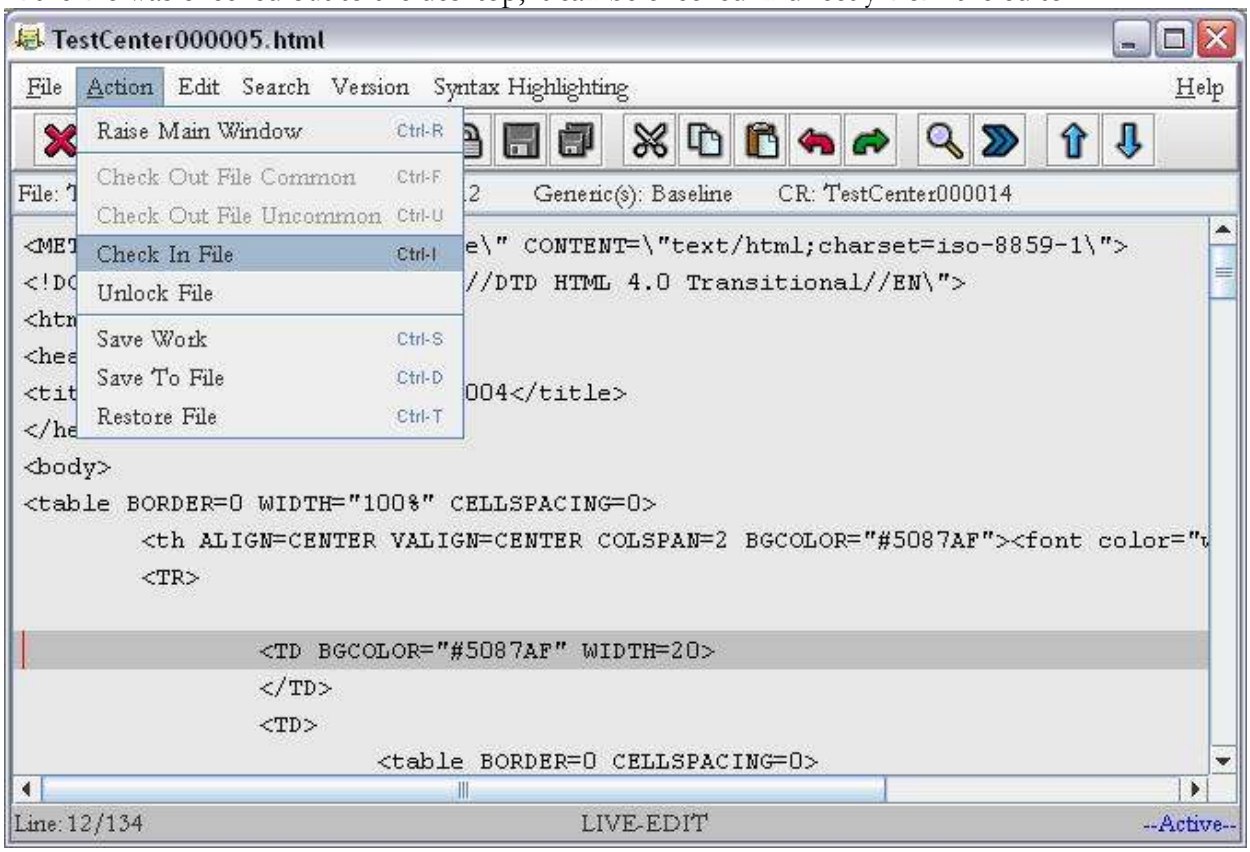
The user will be presented with a screen to confirm the check-in. Verify the file(s) and CR(s).

If this is correct, click **Yes.**

You will receive a confirmation screen when the process is complete. The completion message will show that a new version of the file has been created. If no changes were made, the file will be unlocked and no new version will be created.

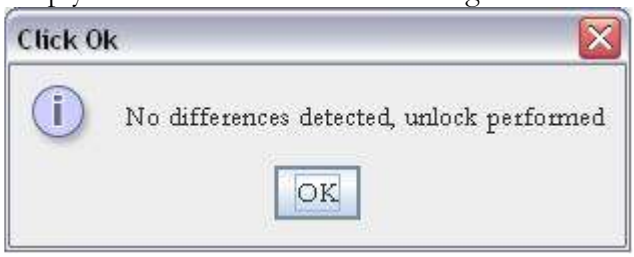
8.3.2 Checking in a file from the desktop

If the file was checked out to the desktop, it can be checked in directly from the editor.



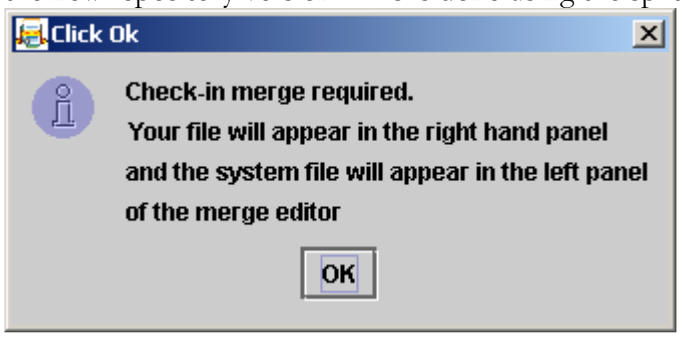
When the user closes a file that has been checked out to desktop edit, he/she is prompted to check it in, save it, unlock it (changes edit status to "read only"), or just close (lose changes made). Changes made to the file will be associated with the CR selected at checkout.

If you attempt to check in a file and there have been no changes made, the system will alert you and simply unlock the file without creating a new version.

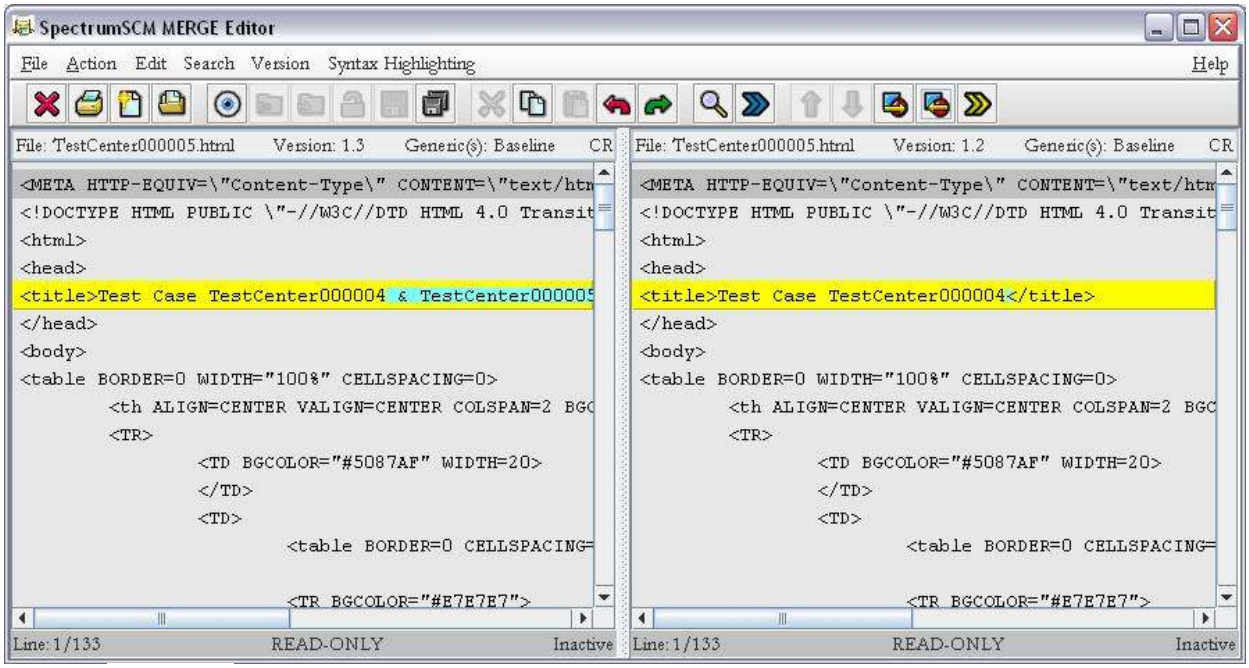


8.3.3 Checking in a file checked out common concurrent

If a file was checked out **common concurrent,** at check-in the system will check for modifications made by other users. If there have been none, the file is checked in as above.
If there have been modifications, then a merge operation is required to merge in your changes with the new repository version. This is done using the spilt screen merge editor.



The Split screen merge editor automatically appears. The file is now locked while the merge is being done to prevent any intermediate changes.
The merge is required before the file can be checked in.
The new, current repository file is presented in the right-hand panel. The left-hand panel is your file version.



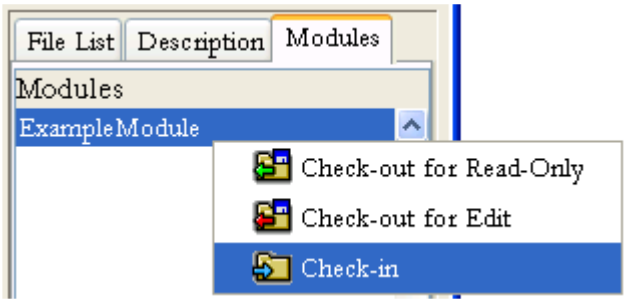
Use the  buttons to highlight the differences.

Move the mouse cursor over the change to be applied and right click. Changes can be applied from one editor to the other a block or a line at a time. Block and line level changes are applied from left to right or right to left depending on the direction of the original diff selection.

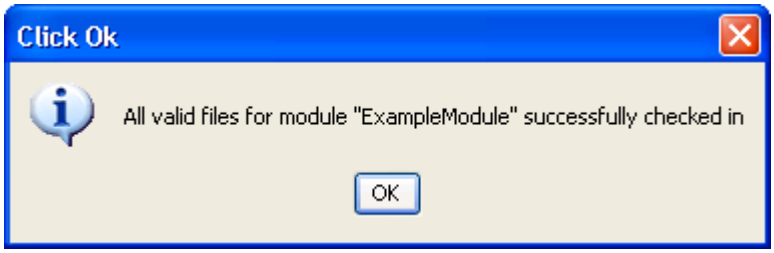
Selecting the right most editor button color codes changes to the files based on what has to happen to the right hand editor to make it look like the left hand editor. Selecting the left most diff button color codes the left hand editor so that applied changes will make it look like the right hand editor. Think of it this way, what do I have to do to this edit session to make it look like the other edit session?
Once merging is complete, check the file back in and a new version of the file will be generated. If there are a number of differences that you are sure all need to be applied, rather than going through and right-clicking each one, you can go to the **Action -> Apply all diff marks** menu option. Make sure that this is really want you want to do, if there are a large number of differences it might be difficult to spot that debug statement or some other code or statement that does not need to be in the submitted copy.

8.3.4 Checking in a module

To check in a module (a group of files) that was checked out, highlight the module name and **Check-In.**

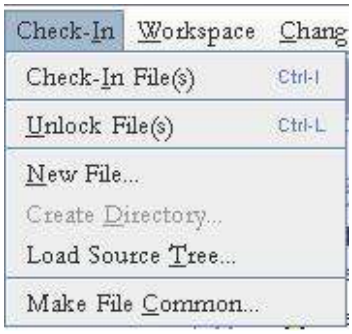


All files in the module that were modified will be checked in and associated with the CR selected when they were checked out. Files that were not changed will simply be unlocked, a new version is not created.



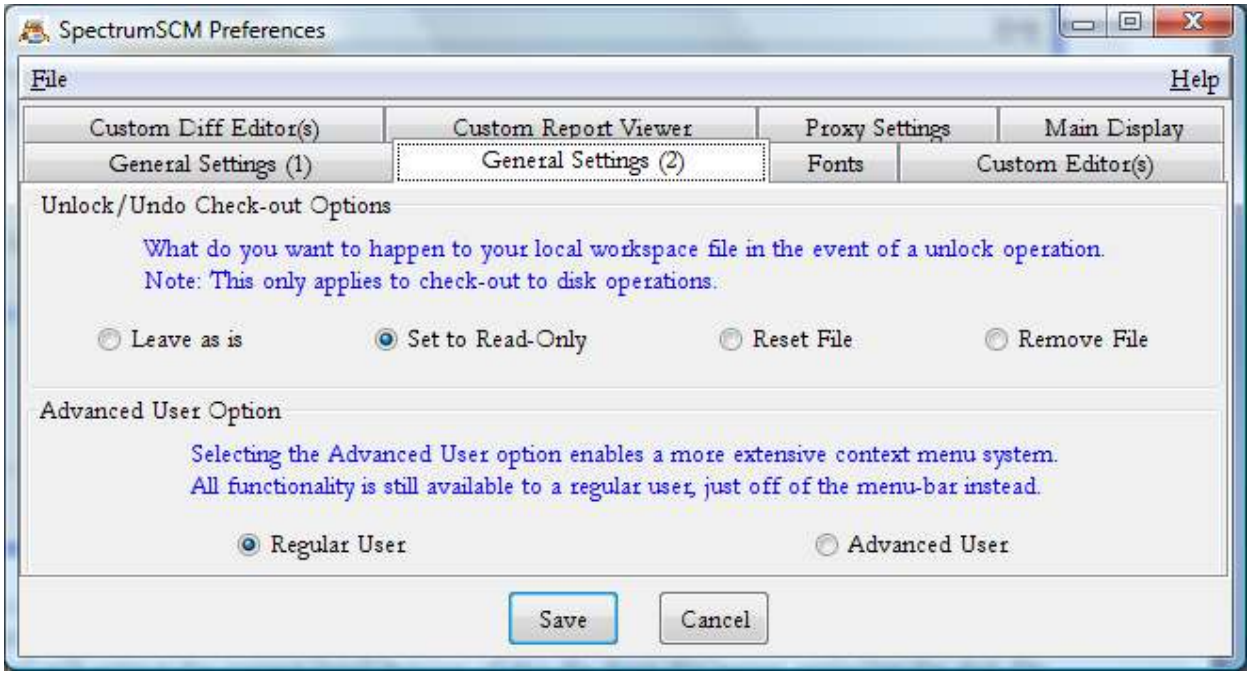
8.3.5 Unlocking a file from Edit

A file that has been checked out for edit can be unlocked. It is as if the file had not been checked out from the SpectrumSCM system. An unlocked file cannot be checked back in. If you had changed your mind, you would need to check out the file again.



Note: An administrator can unlock a file checked out for edit if the user is unavailable to do so (on vacation, in a meeting etc). However, if the edit has not been saved to the disk by the user, any changes will be lost. Note – if the e-mail notification option has been enabled, the user will be sent an e-mail notification of the administrator unlock. This is to act as a reminder in-case the user had edit changes that do need to be checked back in. The workspace synchronizer (Section 8.5 below) can be used to merge such changes back into the repository if needed.

When a desktop edit is unlocked the file and any changes associated with it will be thrown away and therefore lost. When a disk based edit is unlocked, by default, the disk file is left as it is but simply marked read-only. This is so that a subsequent extraction will not prompt to overwrite a writable file. This behaviour can be changed through the user preferences unlock options.

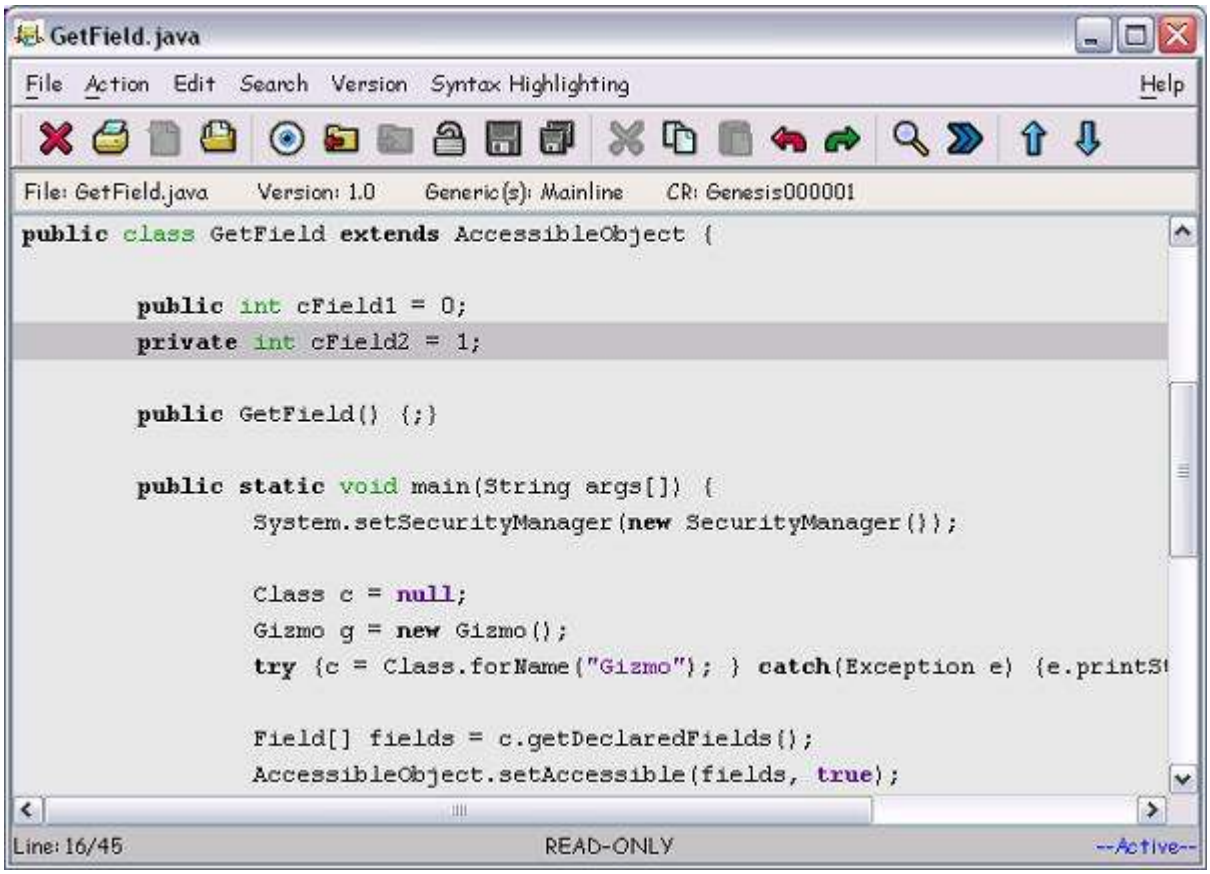


"Leave as is" will leave the file contents the same and in a writable state.
"Set Read-Only" will leave the file contents the same but mark the file read-only.
"Reset File" will extract the current head revision of this file from the server, replacing the disk file and its contents.
"Remove File" will delete the file from the local root hard-drive. The file will still reside in the SpectrumSCM server repository.

8.4 Using the SpectrumSCM Editors

8.4.1 Using the SpectrumSCM Single Screen Editor

The editor is a fully featured WYSIWYG (what you see is what you get) editor. It supports cut/copy/paste, undo/redo functionality, search/replace, syntax highlighting, printing and the unique ability to walk backwards and forwards through a file's version numbers. The user can also elect to jump directly to a particular version number. Just below the tool bar is the file information bar. This area contains information on the file in the editor, its current version number and to what generic(s) the file belongs and the CR number that was used to create the current version of the file. The bottom line of the editor pane contains useful auxiliary information such as the current line number and whether the file is open for edit or read-only.



The menu system supports the following activities.

File – File Menu

Close – Close the current file

Print – Print the contents of the current file

New File – Create a new file from typed contents

Open File – Open a file from the file system and replace the contents of this edit session with that content

Action – Action Menu

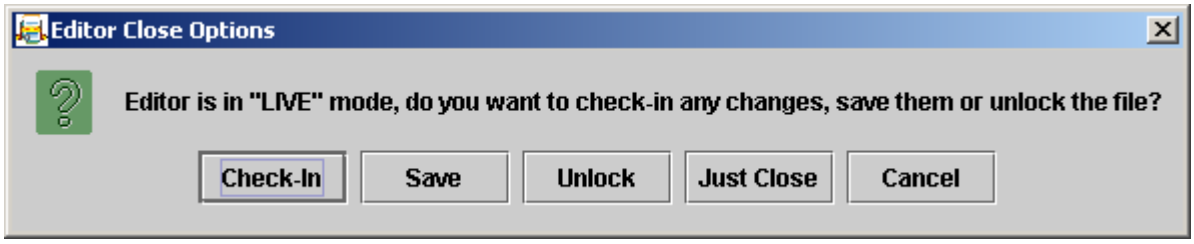
Raise Main Window – Bring the main window into the foreground

Check Out File Common – Check out the current file "common"

Check Out File Uncommon – Check out the current file "uncommon"
Check In – Check in the current file (if out for edit)
Unlock – Unlock the current file (if out for edit)
Save Work – Saves a copy of the edit off to the server
Save To File – Save the contents of this file to the file system
Restore File – Restore file contents back to original content
Edit – Edit Menu
Cut – Delete the currently selected text
Copy – Copy the currently selected text
Paste – Paste the text from the previous cut or copy operation
Undo – Undo the last edit action
Redo – Redo the last undo operation
Fonts – Select between large, medium and small monospaced fonts
Search – Search Menu
Search and Replace – Invoke the search dialog
Find Next – Find the next item matching the previously entered search criteria
Goto Line – Move to the specified line
Version – File version menu
Prev – Move to the previous version of this file
Next – Move to the next most recent version of this file
Goto – Specify a file version to switch to
Syntax Highlighting – Enable syntax highlighting for your preferred programming language.

The tool bar contains shortcuts for most of the items found in the main menu system.

If an edit session is in-progress and the window is closed a popup will request which of the "save" operations (if any) is required.

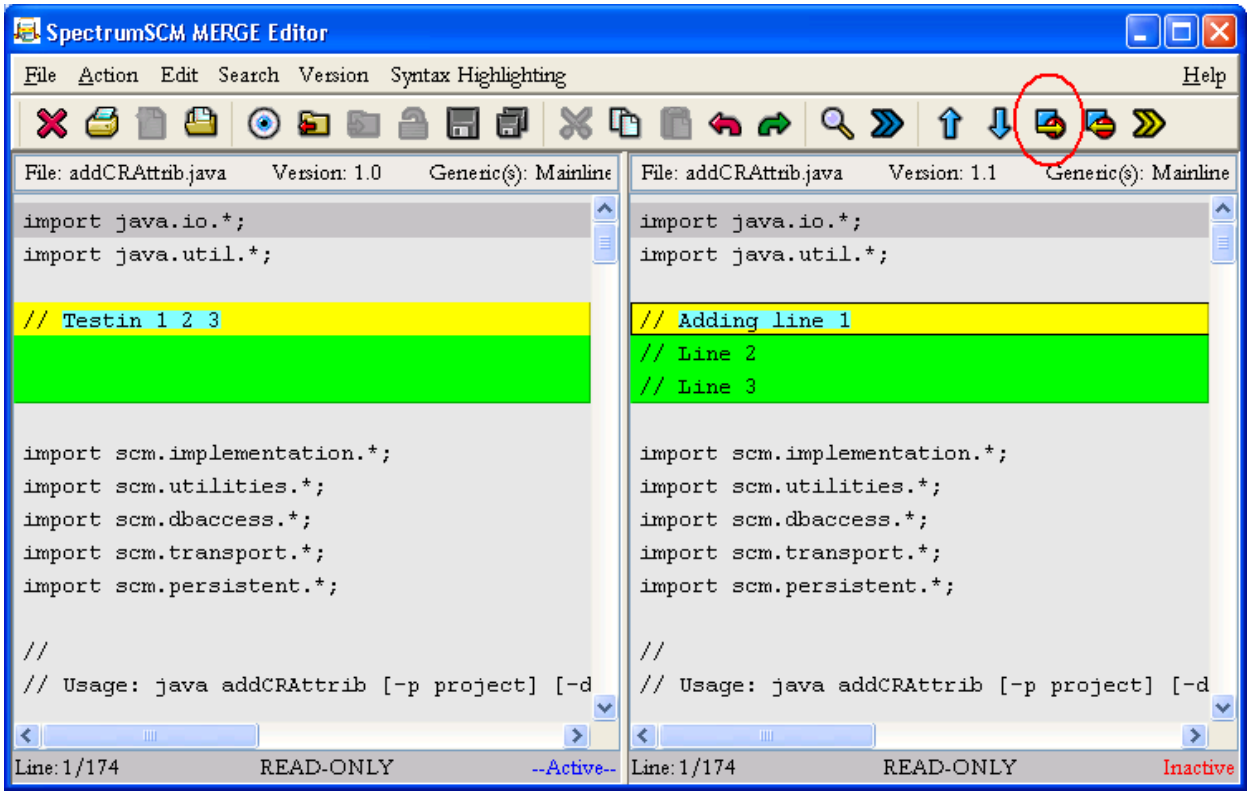


If the file is checked in, a new version of the file is created in SpectrumSCM. Changes made to the file will be associated with the CR selected at checkout. The Unlock option causes the edit changes to be thrown out and the file will be unlocked from edit.
The **Save** option saves the file contents back to the SpectrumSCM server. The edit session can then be continued at a later time by double-clicking on the file in the middle-panel edit-status tab.
The **Just Close** option throws out the edit changes to the file.
Neither option checks the file back in.

8.4.2 Using the SpectrumSCM Split Screen Merge Editor

The Split Screen Editor shares all the same attributes as the single pane editor and has the unique ability to visually show differences between files. In this simple example one line was changed and a couple of lines added to the right-hand window and the diff button (shown in the red circle) has been selected. The right-hand editor shows which lines must be inserted into the left-hand window in order to make it match. The blue highlight against the yellow change bar indicates what text has changed within the contents of the changed line.

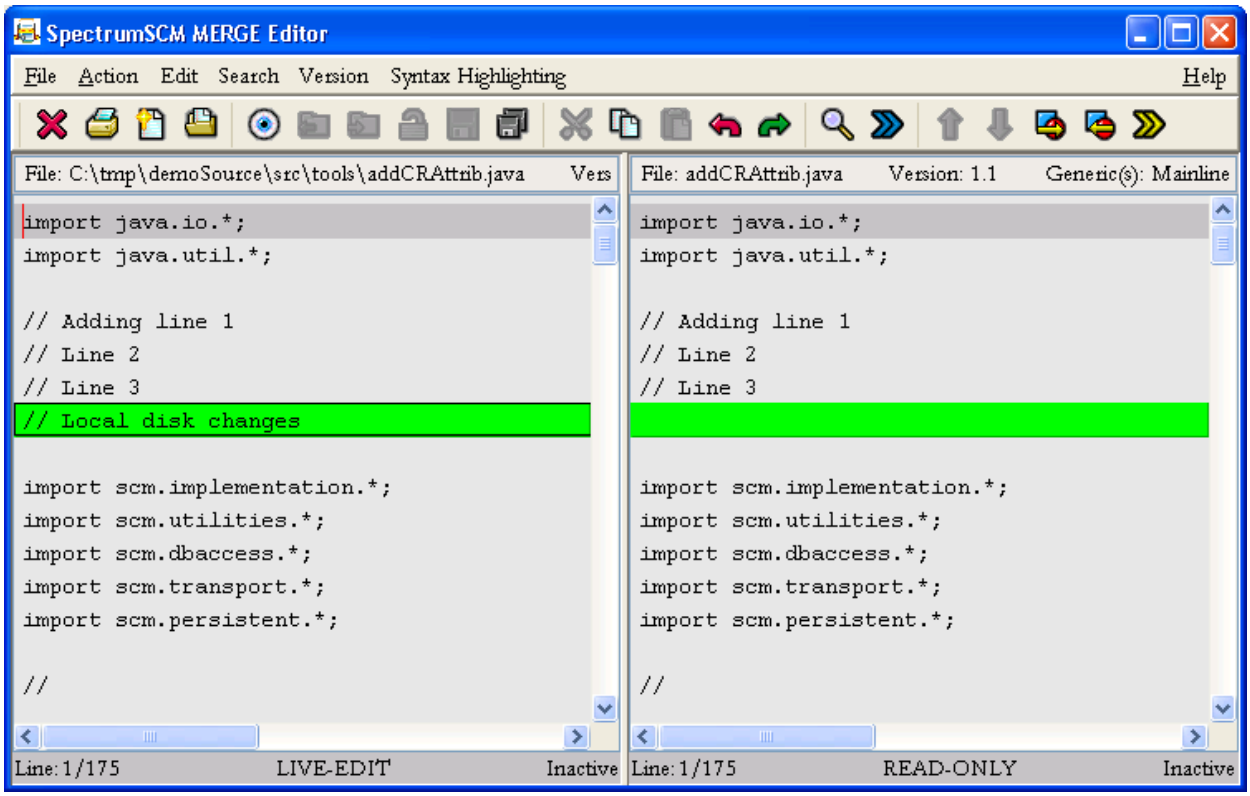
The difference action can also be triggered off of the Action menu, either "Diff left to right" (i.e. what has to be done to the left panel to make it look like the right), or "Diff right to left".



A file can be opened in the split screen editor via the File -> Open Dual Editor on selected file menu option. Select the file from the middle panel if it has been checked out for edit (live-edit mode) or project tree (read-only mode).

In the example below, addCRAttrib.java has been checked out for edit (to the disk), so it opens in live-edit mode. The repository version of the file will be opened in the other side of the split screen.

The user can then choose another file from the current local directory (with **File -> Open File)**, or via the **Version** menu options if desired. To open the other file, click on the right side panel to make that panel **Active** and then use the menu system to load the desired file contents into that active panel.



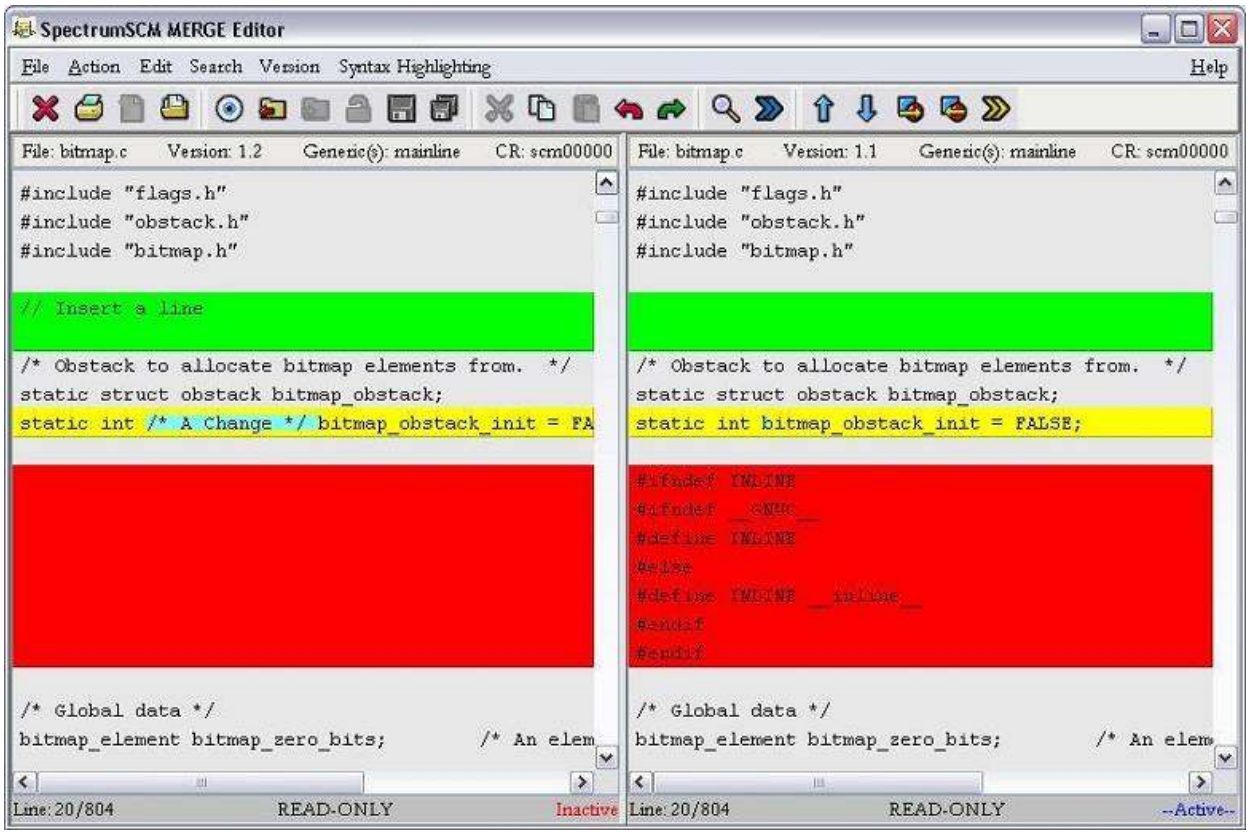
In this example, we have the disk file open on the left. The user clicked on the right-hand panel, and then Version -> Previous to bring up V 1.1 for comparison.

The  icons can be used to display a previous or later version of the file. The  icons can be used to copy selected source to and from the clipboard.

The Merge Editor has the unique ability to visually show differences between files and allow the user to copy those differences from one file to another. In the next example, several lines have been changed in the left-hand window and the diff button has been selected. The color bars show which lines must be changed in order to make it the panels match.

Use the  buttons run the difference from either right to left or left to right. The "difference" is that an insert in one file will be a delete in the other or vice-versa, depending on which "direction" the user chooses to run the diff.
Think of it this way: ask yourself, "How do I make the file on the left look like the file on the right, or how do I make the file on the right look like the file on the left?" Choose the appropriate button to execute the differencing engine, which will generate the colored highlights on applying edits from one screen to the other.

Inserts show up in green, changes/differences in yellow and deletes in red. For the yellow change bars, small light-blue inserts will show the area of change within the text. Sometimes it is tricky – in the example below, the red areas indicate blank lines deleted from one file and not from the other!



If diff marks have been inserted, use the right double arrow  to scroll to the next difference. Repeated button presses will move the selection one difference at a time (by default). If the SHIFT key is pressed at the same time as the double-arrow button, then the selection will move one difference block at a time. If the CTRL key is pressed the selection will be moved in the reverse direction i.e. up the file instead of down.

If the right mouse button is selected over a color-coded change, a pop-up will ask if the change should be applied to the other panel.
The user can select yes or no at this point.

Selecting **Apply Line** to apply a single line change, or press **Apply Block** to apply an entire block of changes all in one shot. Individual lines may be selected out of a larger block.

If there are a number of differences that you are sure all need to be applied, rather than going through and right-clicking each one, you can go to the **Action -> Apply all diff marks** menu option. Make sure that this is really want you want to do, if there are a large number of differences it might be difficult to spot that debug statement or some other code or statement that does not need to be in the submitted copy.

After the changes are applied, the live file(s) can be checked back into the SpectrumSCM repository.

8.5 Deleting Files

Deleting files, folder, generics and projects are generally considered protected functions because of the severity of the impact if something is done inappropriately. Deletion is covered by separate permissions levels under the User Category screen. The user must have "Delete Files" permissions to delete files and/or folders. To delete a generic or a project the user must have the corresponding "create" permissions level (create new project or create new generic respectively). Deletion activities are recorded in the deletion log for trackability purposes.

To delete a configuration item select it in the main tree and the corresponding CR against which the deletion is to be recorded. Then select the Admin menu -> Delete item.

There are 2 types of delete actions, hard and soft. A soft-delete is an item that is marked as deleted and will not therefore be displayed or extracted. However a soft-deleted item still resides in the SpectrumSCM repository and can therefore be recovered at a future time if needed. A hard-deleted item is literally removed from the repository and is therefore not recoverable from within the SpectrumSCM system. To maintain release reproducibility (a significant SCM system requirement) assets that have been placed in a release are not hard-delete'able, these must be soft-deleted instead. A soft-deleted item can still be accessed by release management to guarantee release reproducibility.

If a file or folder is being deleted that involves other common generics, the user will be prompted as to whether the delete action should involve these other generics (common) or be isolated to just this generic (uncommon).

8.6 Workspace Analysis and Synchronization

SpectrumSCM provides an easy way for users to keep files synchronized between their local workspace and the server side repository. The main display supports workspace analysis between the repository view and the users local files.
In this example the file Feature_One_Design.txt in the users local workspace is out of sync with the same file in the server side repository.
The tree view supports two icons that give the user visual clues about the state of the files in his local workspace.

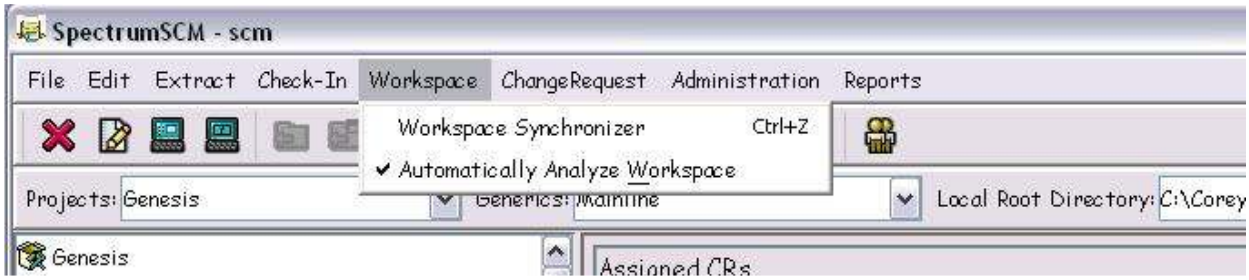
Caution  This icon tells the user that the file is out of sync with the repository

Missing  This icon tells the user that the file is missing from the users local workspace. Users can resolve workspace synchronization issues in several ways. Out of sync or missing files can be manually downloaded simply by right clicking the file and using the context sensitive menu to extract the file to the local hard drive.

The user can also use the Workspace Synchronization feature to analyze directory hierarchies for files that are out of sync or missing.

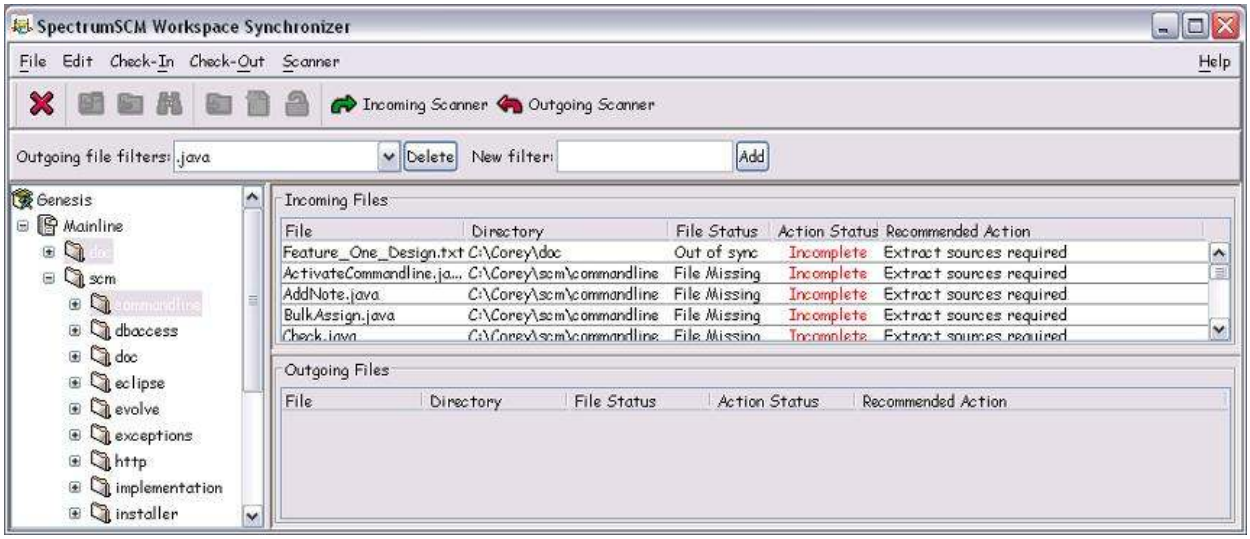
The Workspace Analyzer is responsible for adding the icon decorations outlined above to the regular file icons in the tree view. The Analyzer runs automatically when a user expands the tree view for a directory or when a project level refresh is activated. Project level refreshes can be started either by typing CTRL-F on the keyboard or by selecting Edit->Refresh in the main screen menu system.

The Workspace Analyzer can also be manually activated by toggling the Automatically Analyze Workspace menu item found in the Workspace menu.



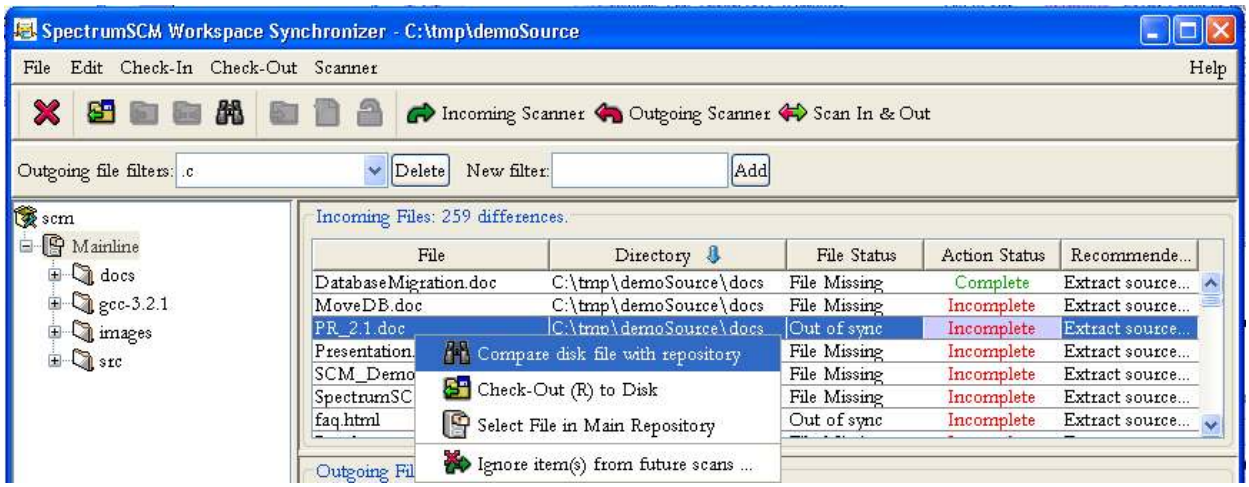
8.6.1 Workspace Synchronizer

The Workspace Synchronizer is a separate screen that can be used to detect file synchronization issues and can be used to act upon out of sync situations.



The Workspace Synchronizer can be used to resolve the following change types:

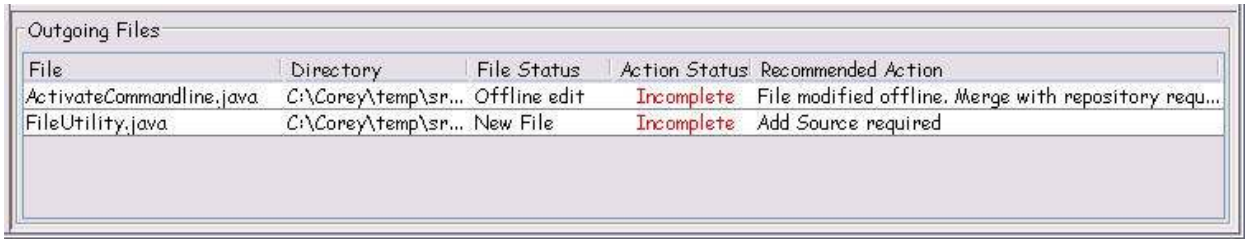
Incoming Changes – These are changes that have been made in the repository and need to come **in to** your workspace. The Incoming Scanner can detect this type of issue and can be run at any directory level. Multiple directories can also be chosen for scanning at the same time. The Incoming Scanner allows the user to take the following actions on files:
Compare differences (Visually)
Check out to Disk (Read only)
Select the file in the main repository view
Ignore this item from future scans



Checking out to disk read-only will overwrite the local file image with the version of the file from the repository.
Comparing the disk file with the repository will engage the diff/merge editor and display both versions of the files side by side.

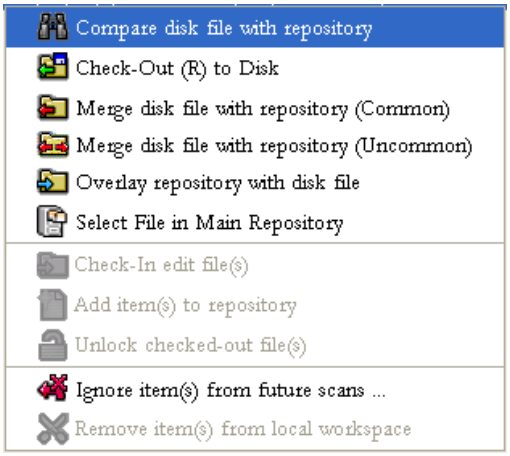
Outgoing Changes – These are changes that have been made locally that need to go **out** to be applied to the repository. These changes can include offline as well as online changes. Offline changes are those that the user might make while disconnected from the SpectrumSCM Server, i.e. work done at home at night or on the bus during the morning commute. Online changes include those that are done while connected to the SpectrumSCM Server and include regular check outs.

Outgoing scans are performed relative to the "**Outgoing file filters**" specified. These are used to pick up only the source files of interest and to ignore temporary or object/executable files that are not to be checked in to the repository. To set the outgoing filter, type the extension of interest ".java", ".c", ".vcproj" etc into the "New Filter" box and hit add. Multiple file types can be added in one entry by separating them with the pipe symbol (eg ".java|.cpp")



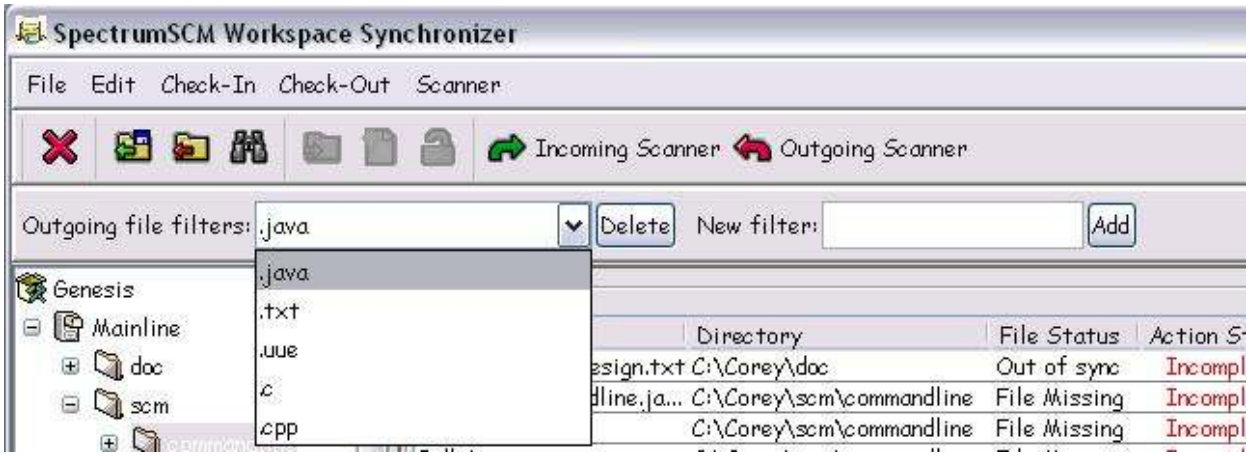
In this example the Outgoing Scanner has detected two files that need to taken care of. The first file, ActivateCommandline.java, was edited offline. The second file, FileUtility.java, is new and does not yet exist in the SpectrumSCM repository. There are many actions that can be taken on outgoing files, depending on their overall status:

These actions can all be accessed via the context sensitive menu associated with right clicking on the files. Most actions can be applied to multiple files at the same time, but some actions can only be applied to a single file at a time (Compare).



The Workspace Synchronizer can be customized to operate against different file modes and on different file types. Normally the Incoming scanner only operates on files that are marked at the OS level as read-only files. Also, the outgoing scanner normally only operates on files that are marked at the OS level as read-write. This can be changed by opening up the Scanner Options. Incoming scans can be configured to look at read-only files as well as files that are marked as read-write. Outgoing scans can be configured to take into account read-write files as well as read-only files.

The outgoing scanner **must** be configured to work against a particular set of file types. This is so that temporary files or compiled binaries are accidentally (and time-consumingly) included. On the Workspace Synchronizer main screen the user uses the Outgoing Scanner Filters section to configure the file types that should be considered in the scan.



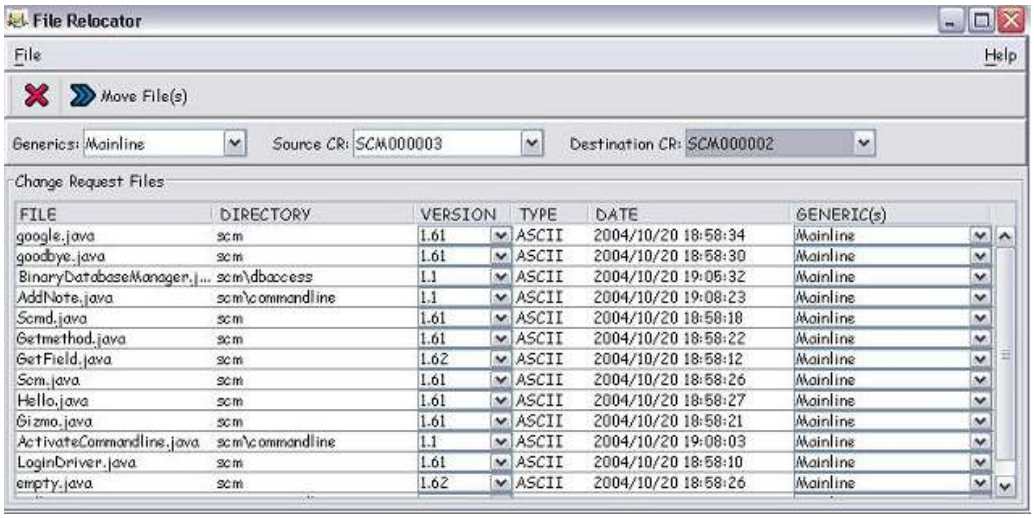
In this case the Outgoing file filter is set to operate on 5 separate file types. New types can be added by simply entering a file type extension into the New Filter section and either hitting Enter or by pressing the "Add" button. File types can be removed just as easily by selecting the type to remove and pressing the "Delete" button.

All Workspace Synchronizer functionality for incoming and outgoing scans is available in both the context sensitive menus associated with the files, or in the menu and tool bars located at the top of the screen.

One of the most outstanding aspects of the Workspace Synchronizer is that it allows the user to keep their local workspace in sync without having to download every file in the SpectrumSCM repository. Consider a remotely located repository with 35,000 files checked in. Downloading 35,000 files may take quite a while, depending on the size of the networking connection between the two locations. By using the Workspace Synchronizer the user can quickly and easily identify files that have changed and need to be downloaded. Also, working offline becomes a snap since the synchronizer knows exactly what was changed while working offline, and will allow the user to quickly and easily reconcile those offline changes with the SpectrumSCM repository.

8.7 CR/File Relocator

Should a file accidentally get edited under the wrong Change Request, the CR/File Relocator feature can be used to move the files around underneath a CR. Selecting the **File Relocator** option under the **Change Request** menu will present the following screen.



Select the **Source CR** from which you want to move one or more files. Select the **Destination CR**, where

you want the files to go. Both CRs must be assigned to you for editing; otherwise they will not show in the choice boxes.

Once the source CR is selected, the file list will be populated with all the files and their versions that are available for relocation. The version default to the oldest version edited for this CR i.e. ALL versions of this file that were previously edited under the source CR will be moved to the destination CR. If you only want to correct the last edit made, then select the last version number (the highest) in the version popup.

As versions of files are selected the generic field shows what generics that version of the file is common to. If a simple or uncommon branching model is being used then there should only be one entry in the generic field per file version.

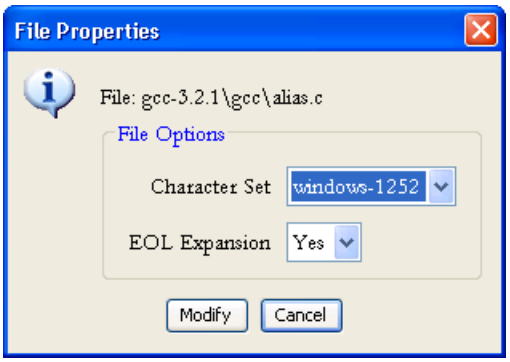
Once the desired file versions have been selected, then you can select all the files that you want to move, and press the "**Move File(s)**" button. Note this is also available on the right-mouse click context menu too.

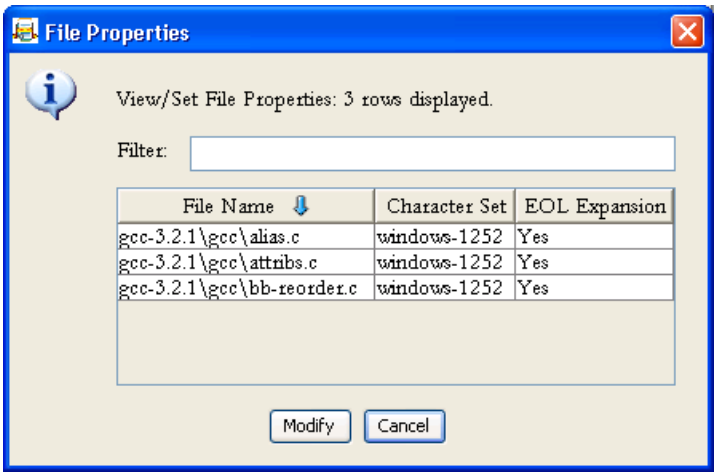
8.8 File Properties / Character-Sets

When a file is added into SpectrumSCM it is stored "as-is" on the server in the repository. When that file is subsequently extracted however, it could be modified to match the local platforms end-of-line characterization. For example, on Unix platforms the default end-of-line is just a single newline character ('\n' or 0x0a), while on Microsoft Windows platforms it is a carriage return followed by a newline ("\r\n" or 0x0d 0x0a). This automatic conversion is a convenience feature in most cases, however in other cases it can cause some complexities.

One way around any issues would be to check the file in as binary instead of text. In this way the file is always checked-in and out as a binary blob and nothing is changed. The end-user would be responsible for any platform specific changes necessary. However, a binary file cannot be easily differenced or compared and if the base file truly is textual then this might not be the best way forward.

Using the file properties options, 2 additional options are made available under the 2.5 release. Select the file(s) of interest and then menu items "File -> Properties" (or if you have the advanced menu item active, you can right-click and select "Properties" there).





The left-most screen is presented if a single file is chosen, allowing direct modification of its properties. The right-most screen is presented if multiple files are chosen, allowing further refinement of the selection either by sorting and re-selecting the table entries or by filtering on particular file types. If the filter field is blank all selected files will be shown. A filter such as "`*.c`" will show only the C files.

Setting the End-Of-Line Expansion option to "No" will mean that the text file will be extracted out to the local hard-drive the same as it was placed into the SpectrumSCM repository. The end-user would be responisble for any platform end-of-line differences.

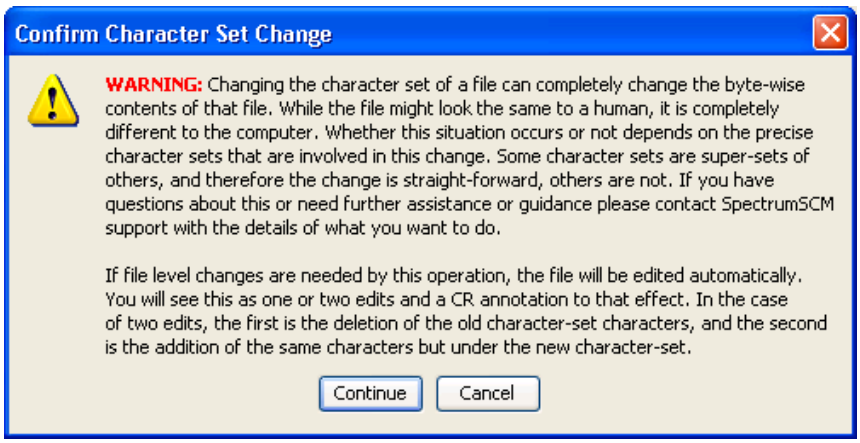
Setting the character–set might be another option/requirement, particularly when dealing with international files (such as those with Unicode, Multi-Byte, Chinese or Japanese characters). SpectrumSCM operates relative to the local operating system character sets, however in a cross-platform or international environment this might need to be further clarified. For example you don't want a user checking in a file in one character set that might not be understood by a second user operating under a different environment/country/character-set.

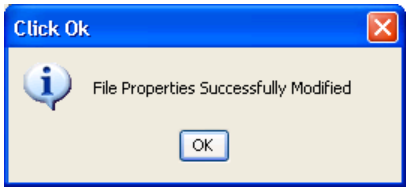
To override the default system character-set the SpectrumSCM administrator can set the **scm.charset** property through the server configuration wizard (See Chapter 3 – Server and UI Configuration). This value is also important if file version keywords (**scm.keyword.expansion** parameters) are going to be used since these are updated on the server-side as a file is being checked-in.

Additionally, if an individual file (or some subset of files) are desired to be recorded in a non-standard character-set then that can be done through the File -> Properties character-set option. An example of this is that some recent Microsoft Visual Studio control files can default to the Unicode character set of UTF-8 or even UTF-16 (multi-byte format). Simply select the appropriate character-set option, select the appropriate Change Request against which this operation is to be recorded and hit the "Modify" button. The CR will be annotated with the record of the property change.

State	User	Begin Date	End Date	Note
Note	scm	2008/01/07 13:23:53	2008/01/07 13:23:53	gcc-3.2.1\gcc\attribs.c: Character Set option changed from "windows-1252" to "ISO-8859-1"

Note: In changing the character-set of a file you might see the following warning message, this is normal but please contact SpectrumSCM Support if you have further questions on this subject.





Chapter 9

9 Release/Package Management

This chapter describes how files can be packaged to create a product release. You will learn how SpectrumSCM ensures the integrity of a release, its automatic CR dependency checks, and how to select CRs to create a release. Under SpectrumSCM version 2.6 the Package/Component Management feature has been introduced which allows product extract/builds to be assembled from multiple release/project components.

9.1 What is a Release?

A release is a set of files, each at a particular version, that when extracted from the system, make up a single version of the product. Managing release formation can be a tedious, time-consuming job on some CM systems. In order to create a release with separate versions of individual files, a CM system needs to be able to properly track the file changes that make up the individual file versions and present that information to the user in some meaningful form. In the **SpectrumSCM** system, this is easily accomplished with the built-in issue tracking system.

Specialized file changes, like small branches in the code, cannot be lost in the system because each change is officially recorded in the issue tracking system. Each release is a collection of active CRs in the system. Releases build on top of previous releases to simplify the entire release management job. As each CR is completed and the releases are created, the number of outstanding CRs that must be accounted for is reduced to the number of open CRs since the last release was created. This keeps the number of CRs for any given release down to a manageable few.

A release (a specific version of the product) can therefore be easily re-created at any time simply by extracting the relevant versions of the necessary files using the appropriate CRs. Outside of tracking changes to files, creating releases is the single most important job that a CM system must perform. The **SpectrumSCM™** system is designed from the ground up with the necessary features that make release management a simple task.

9.2 Releases and their Relationship to Generics

Releases are defined within a generic and a generic can contain a sequence of releases. The first release in a new generic is based on the last release in the predecessor generic. This is done in the **SpectrumSCM** system as a space saving measure and it keeps in line with the overall practice of basing one release on another.

NOTE: When a new generic is formed, the last release in the predecessor generic will become locked, and that release will be used as a basis for subsequent releases in the descendent generic.

9.3 The Release Management Process

By design, the **SpectrumSCM** system imposes no rigid or predefined release management process. The following sections define two example/possible release management process strategies. Either strategy can be used to produce good results while maintaining the ease of use of the overall system. The user of the system is free to use either one or a combination of the two strategies, depending on the needs of their organization.

Strict Release Management Process

A strict release management process is one that guarantees that any previous release can be directly extended to include new works or bug fixes. Recall that releases are built with change requests, and that change requests are made up of particular versions of files. Recall also that releases are based on previous releases, that is, release 2.0 is dependent on the particular versions of files in release 1.0 and so on. Once the 2.0 release has been formed (files have been extended from release 1.0 to form this new release), release 1.0 can be directly extended when a strict release management process is in place. **This strict release management process requires a new generic for each release.** This is also known as the **Branch on Release Pattern**[1]

Simple Release Management Process

A simple release management process is very similar to the strict process except that multiple releases can be formed in the same generic. When multiple releases are formed in a single generic, only the last release can be *directly* extended. Any one of the previous releases can be extended by forming a new generic with the files contained in that release. When a generic is formed from a release, the files in that generic will exactly match the versions of the files in the release.
When generics are created from releases in **SpectrumSCM™**, only the files that have actually changed from the baseline will get new disk images. All other files will simply be common object references to already existing files. This is done as a space saving feature considering that a good number of files in any system rarely change as new features are added.

Combinations of strict and simple release management

Combining the two release management process strategies involves using the simple strategy to produce patch releases of a product. When a new feature is being added, create a new generic (per the strict release management process) and do the new feature work in the second generic. The default behavior of the system is to check files out uncommon. This guarantees that the last major release of the product in any generic is always easily extensible in case the need arises. A fix applied to the current release (the last release of the previous generic) can be made common to the new generic by making the fix common.

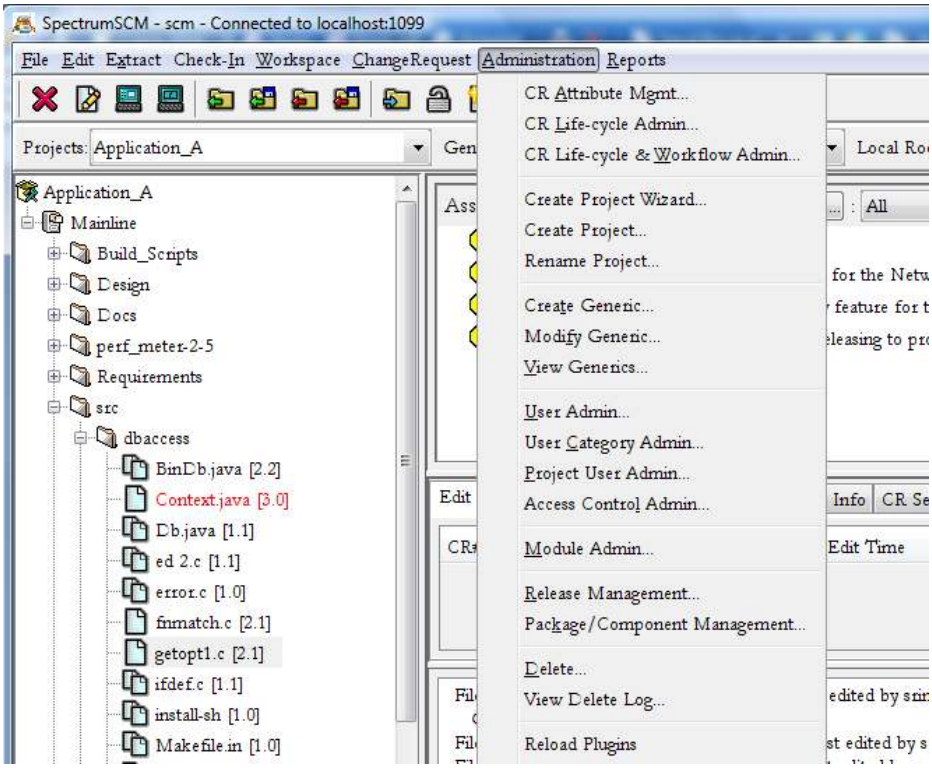
---

[1] Berczuk, Stephen P. and Appleton, Brad. Software Configuration Management Patterns. Effective Teamwork, Practical Integration. Addison Wesley 2003. ISBN 0-201-74117-2

9.4 Creating a Release

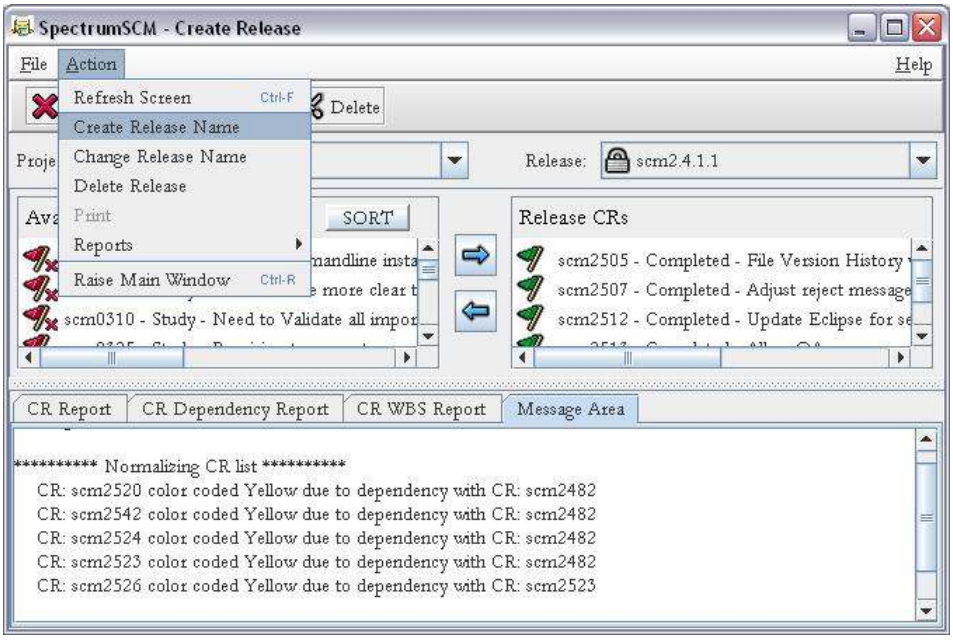
To create a release,

1) Select Release Management from the Administration menu on the Main Screen.

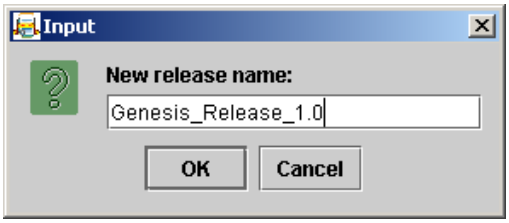


This will bring up the **Create Release Screen.**

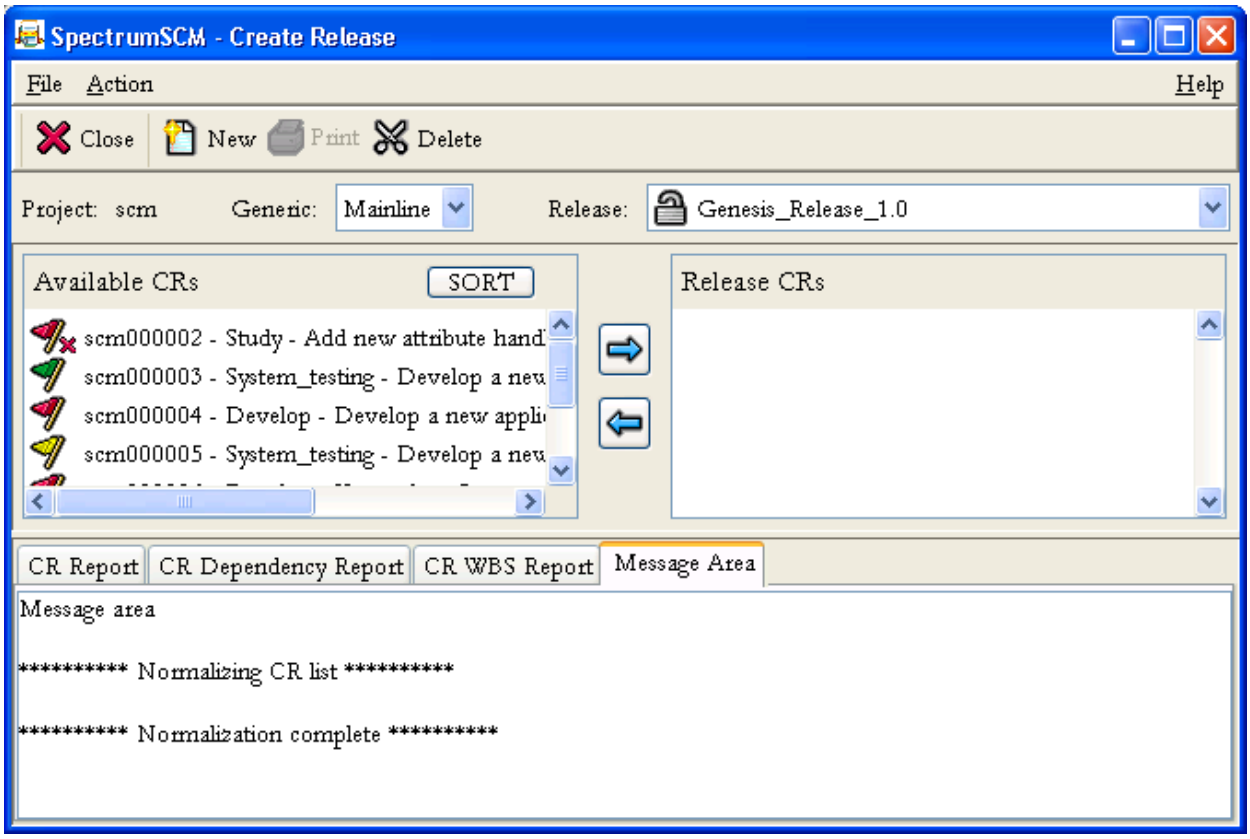
2) Select **Create Release Name** via the Action Menu on the Create Release screen.



3) This will bring up the **Input** window where you enter the new release name. Click **OK**.
This will take you back to the **Create Release screen.** Notice that the release name is populated.



9.4.1 Select the CRs to be included in the release



Available CRs – This area displays the CRs that are in this generic. Note that CRs are flagged and the flags can be green, red, or yellow.

Green – the CR's development life cycle is complete (but possibly pending testing, approval and deployment type phases). You can include this CR and its associated files in the release.

Yellow - the CR has completed development; it is dependent on other CRs that have not yet been completed.

Red – the CR is not complete.

Any CR with a red "**x**" to the right of the flag icon indicates that the CR has no file level changes associated with it.

Choose the CRs that you want to include in the release and use the right arrow  to move them into the **Release CRs** area. If you make a mistake, you can use the left arrow to remove a CR from the release.

Only green CRs may be selected for inclusion in a release. If a CR that you wish to include in the release shows up as red (), it is not finished "development" i.e. it is not past the final source modification phase defined in the project's life-cycle. If this CR is wanted to be included in this release then the current assignee will need to finish their work and progress the CR or you (or someone else with assignment privileges) will need to use the Assign/Modify screen to move the CR past the final source modification phase. Requiring CR's to be past the final source modification phase guarantees the stability and integrity of the release build process because those CRs cannot be edited.

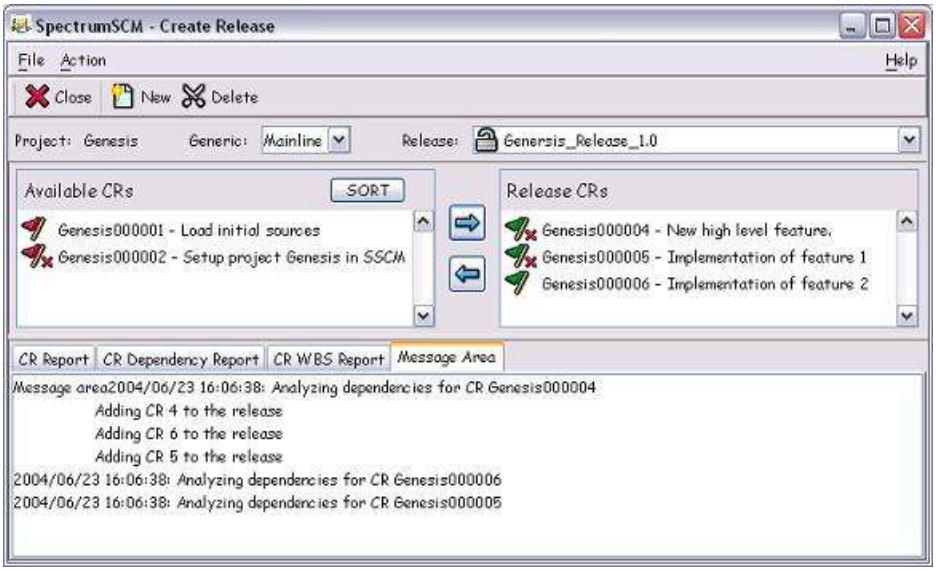
Note, it is perfectly valid for development to continue on CRs not needed by this release. These CRs can even edit files to be released in this release since they will be newer file versions and not dependent file versions.

If the CR is flagged in yellow (), it has one or more dependencies. Double-click the CR to show the dependency detail or right-click and run the specific **CR Report** or **Dependency** report. A yellow CR means that this CR has completed development, but it is dependant on other CRs that have not yet finished.

9.4.2 Dependency Checking

Dependencies are created when multiple CRs have made changes to the same file(s) or when CRs are involved in a work breakdown structure.

As you select CRs for the release, file level dependency checking is being performed. Only green (finished development) CRs and their dependants can be assigned. When a CR is placed into a release, any CRs that the original CR depends on will automatically be placed in the release, too. This action can be seen by opening up the message tab on the bottom of the release management screen.

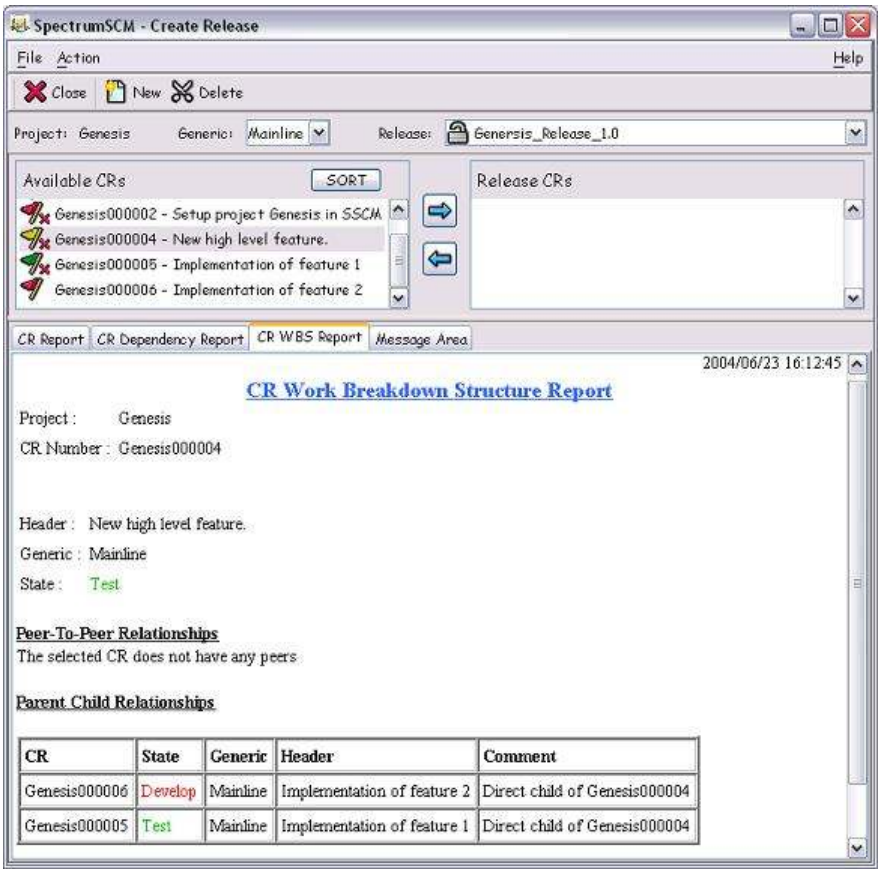


The above example shows that when CR Genesis000004 was added to the release that CRs 5 and 6 were automatically added to the release. This is the result of a parent/child relationship forming a dependency at the CR level. The same action will take place when CRs with file level dependencies are added to a release.

Requiring dependents to be added to the release guarantees the integrity of the build because nothing is being missed out. I.E. Feature A is dependent on Feature B but only Feature A gets placed into the release, this either breaks the build (or worse, the product at run time) because that dependency is unresolved – SpectrumSCM does not allow this situation to occur.

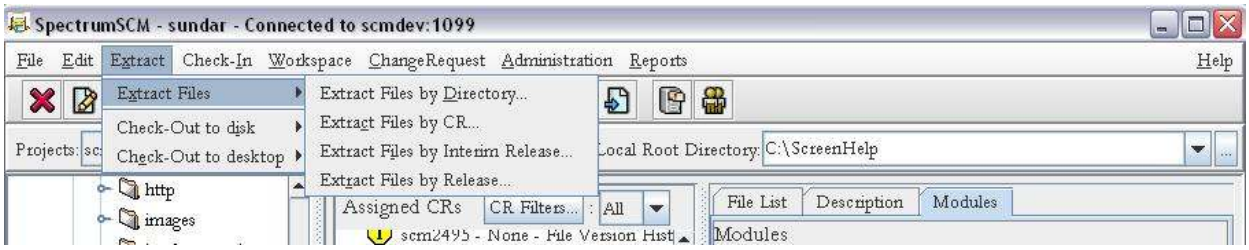
Once the set of incomplete CRs and dependant CRs have been identified, the developers responsible for completing these work items can be alerted and their progress monitored to ensure that they are going to complete the tasks on time. When the dependant CRs have been completed (past the last development phase), a refresh of the Release Management screen will show the CRs are now in a green state and they can be moved into the release.

In this next example, we are defining Genesis_Release_1.0. We see that CR Genesis000004 is yellow. When we right-click on Genesis000004 and look at the WBS Report, we see that it is involved in a parent/child relationship with CRs 5 and 6. CR Genesis000006 is still in the Develop phase. Therefore, CR Genesis000006 has to be completed before Genesis000004 can be moved into the release. CR Genesis000005 may be moved into the release since it is green flagged and has no dependencies.

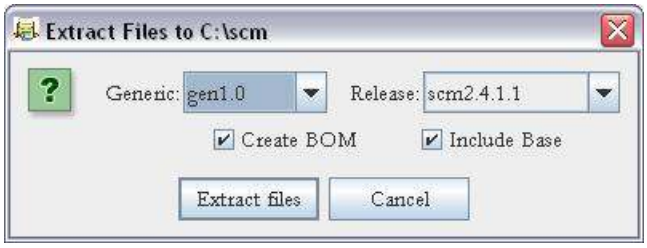


9.4.3 Extract Files to Build the Release

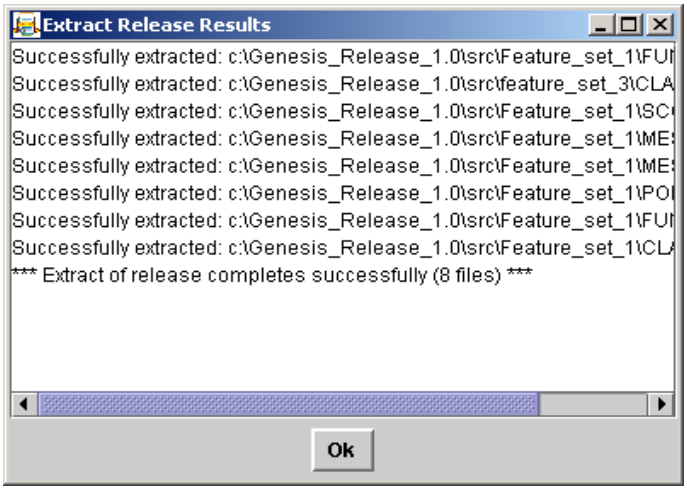
Once a release has been defined, an **Extract Files by Release** can be performed (using the **Main Screen Extract / Extract Files by Release** menu option or cmmand-line routine **scm_gr**) to extract the necessary files that can be used to build the release. The file versions associated with the CRs included in the release will be extracted to the Local Root Directory.



NOTE: Do not extract files for building a release into the same local root directory used for development since this would potentially overwrite (or at least warn about) any on-going development files. Set up a different directory (remember, you can have multiple local root directories). The **Extract Files** window allows you to verify or choose the generic and release to extract.



Click **Extract Files**. All files associated with the CRs included in the release will be extracted to the Local Root Directory specified. If the **Create BOM** and **Include Base** options are selected, a Bill of Materials for the release along with the files in the Base will be created under the local root directory. The BOM file uses a *SSCM_REL_BOM_Date_Timestamp* format for the file name. The results of the extract process are displayed.



9.5 Building the Release

SpectrumSCM gives users the flexibility to use their native environment and desired products to do their builds. Extract the files associated with a release to the local directory from which you would normally run your build. After extracting the files, use them as input into the desired build process to compile the release.

9.6 Adding CRs to a Release

Once a release has been created, CRs can be added via the **Create Release** screen. New files can be added to existing CRs. New CRs can be created and added to the release.

To add, a new feature or bug fix to an existing release, simply add the CR for that feature or bug fix to the **Release CR** list. When CRs or files are added, changed or removed from a release, you must extract the code and rebuild the release for the new code to be included.

NOTE: CRs cannot be added to or removed from a release that has been locked. Release 1.0 will be locked when Release 1.1 is created. Creating a new generic locks the last release in the previous generic.

9.7 Removing CRs from a Release

CRs can be removed from a release via **the Create Release** screen. Use the left arrow to remove a CR from the release. When CRs or files are added, changed or removed from a release, you must extract the code and rebuild the release for the new code to be included.

NOTE: CRs cannot be added to or removed from a release that has been locked. Release 1.0 will be locked when Release 1.1 is created. Creating a new generic locks the last release in the previous generic.

9.8 Re-creating a Past Release

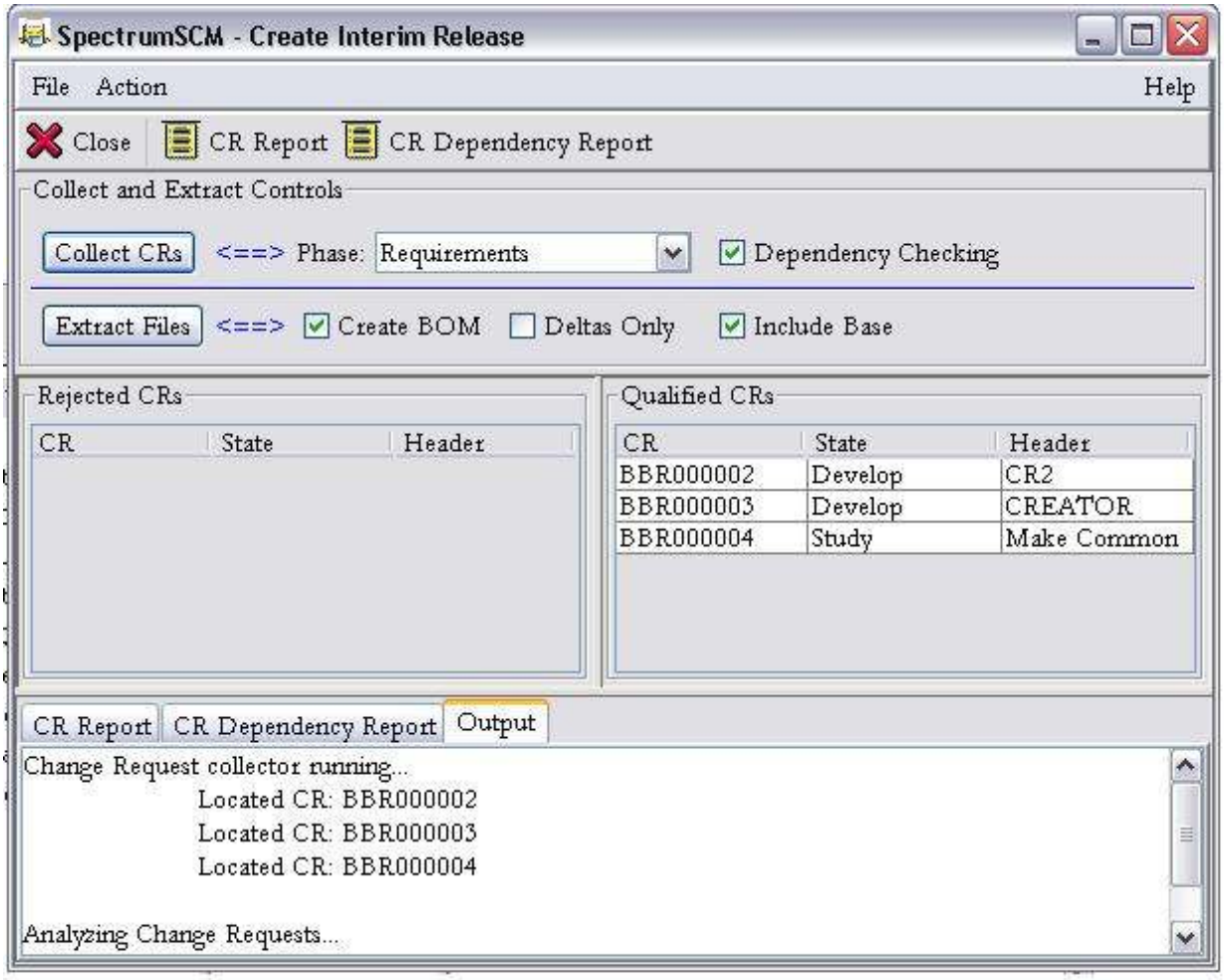
Any release defined in SpectrumSCM can be re-created by following the same steps used for extracting files in a new release. Use the **Main Screen Extract / Extract Files by Release** menu option to pull up the **Create Release** screen. Set up a local root directory specifically for the release so that files belonging to other releases are not corrupted (remember, you can have multiple local root directories). Choose the generic and release to extract using the pull down boxes for generic and release and click **Extract Files**. All files associated with the CRs included in the selected release will be extracted to the specified Local Root Directory.

The results of the extract process are displayed. The contents of the specified local root directory can then be used as input into the build process to recreate the release.

9.9 Interim Releases

You use Interim Releases to extract a set of files based on the phase of your life-cycle but before they are placed into a formal release. So for example, if you have an "Integration Test" life-cycle phase as part of your process, but developers can still modify files under these CRs until things stabilize and become ready for the formal release.

Since this is an informal process, you also have various options to override the dependency checking if needed. Specifically you can turn-off dependency checking altogether (via the checkbox at the top of the screen) or via a right mouse click context menu, include or exclude specific CRs as necessary. Note that by including CR's with dependency issues you might get undesirable end-results. Changes from the dependant edits could affect compilation or execution of the end product. Select the phase in your life-cycle for which you wish to create this IR and then press the "**Collect CRs**" button. The qualified and rejected CRs will be displayed in the right and left panels respectively.



The "**Create BOM**" checkbox will create a Bill of Materials report as part of the extraction. This lists all the files and their version numbers that make up this extraction. Additionally, by selecting the "**Deltas Only**" checkbox, if you are performing a series of extractions and builds, only those

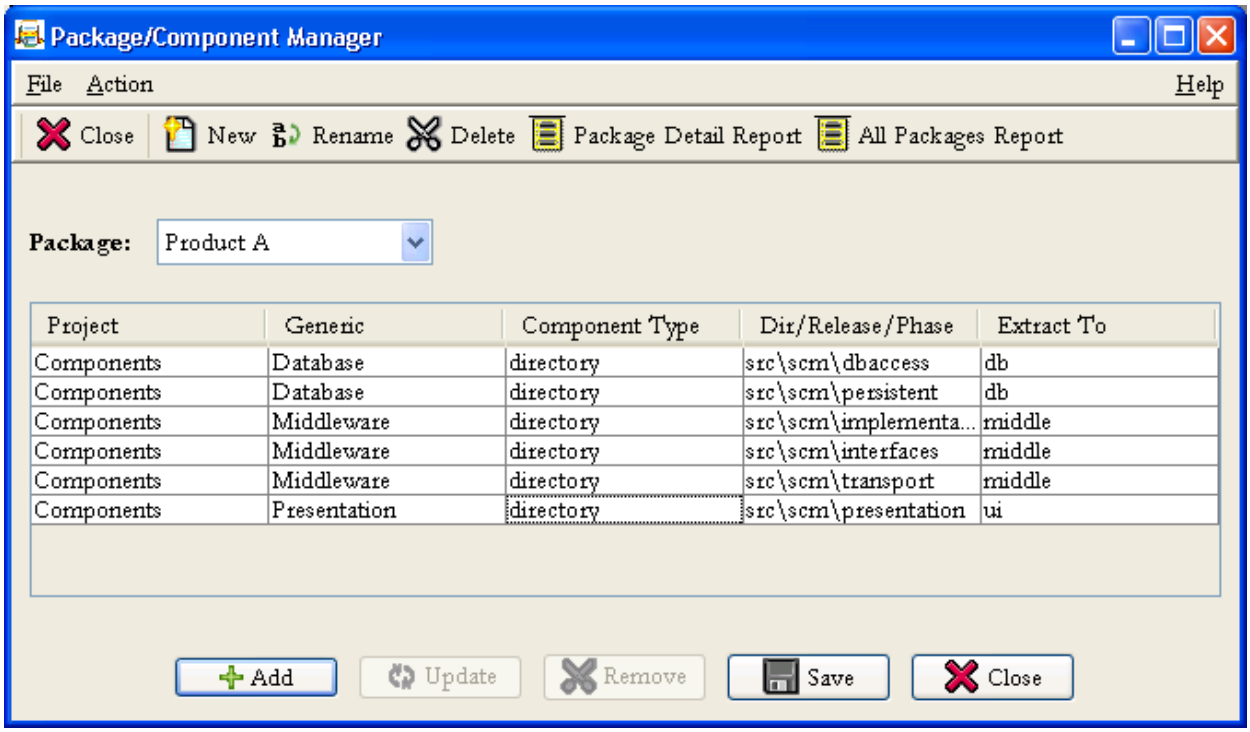
files that have changed since the previous extraction will now be pulled. This could be a significant performance improvement, depending on your process.

The interim release command line command can also be used to automate nightly-build or continuous development environments. Either nightly or as soon as a CR is progressed from the development phase the **scm_gir** command could be triggered to extract and build the target environment.

9.10 Package/Component Management

The Package/Component management feature allows the product manager to organize the SpectrumSCM controlled assets in a flexible manner within the overall repository. For example if your product consists of 3 separate components (it could consist of many more), a database component, a web servlet/ASP component and a main heavy-weight user interface component. These 3 components could be controlled within SpectrumSCM as separate folders, generics or even projects. Depending on the dependencies within these components or their independence, how to store them might vary from one organization to another.

Release Management (as described above) allows the Generic Engineer/Release Manager to define release sets based on Change Requests within a particular project and generic. If you want to organize your components above this level then package management can be used to bring the overall product together in an appropriate manner.

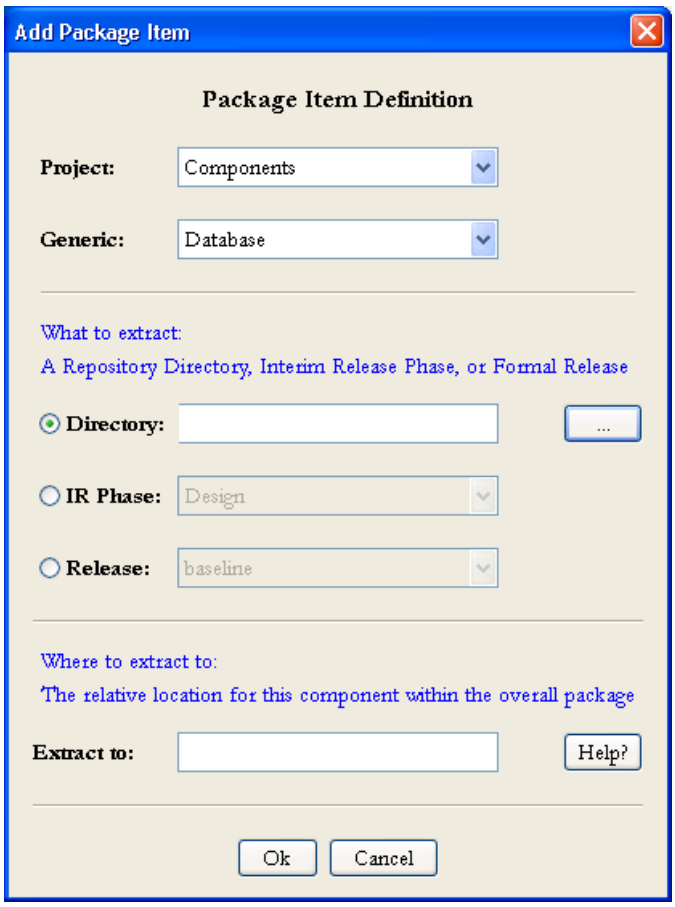


Once a package has been defined it can be extracted in a reproducible manner via the "Extract -> Extract Files -> Extract Files by Package" menu option or the **scm_gpack** command-line.

A Package can be defined by hitting the "New" button or the "File -> Create Package" menu item. This defines an empty package to which you can "Add" the appropriate components (or Action menu -> Add Package Item). This will present the Package Definition window as shown below.

Existing packages can be updated or even removed, however all such actions will be logged (in the server logs directory) for auditability purposes.

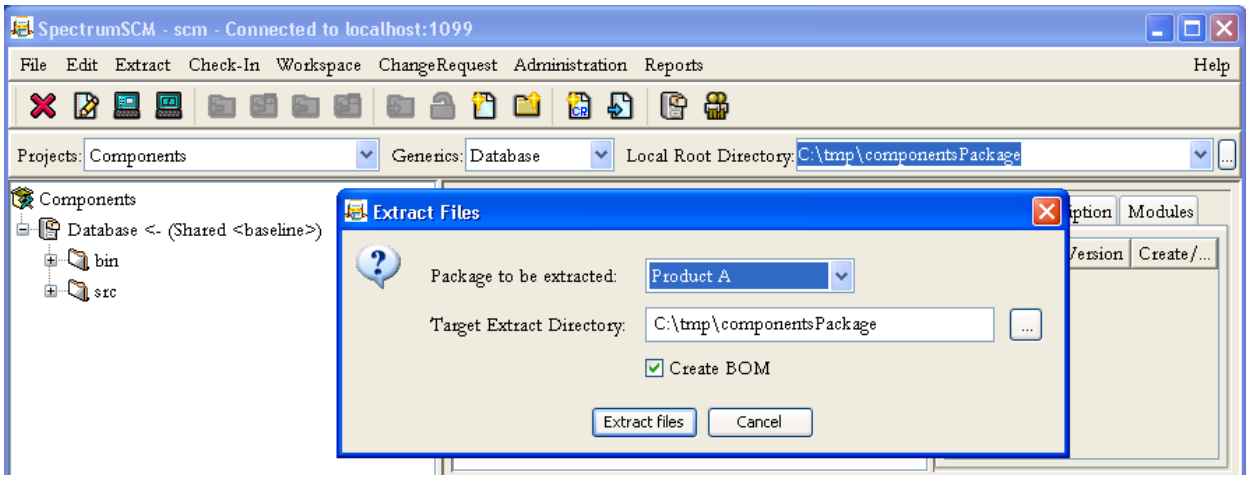
A package item or component can be any release, directory or even an interim release (for testing builds). Select the appropriate project and generic, and then the appropriate radio button (on the left-hand side) to indicate which type of component this package item is to be selected from.



For a release or interim-release, the selection is a simple choice-box. Select the appropriate release or IR phase in these cases. For a directory component, the directory path can either be typed in, or the "…" button can be used to browse for the desired repository folder(s). If more than one directory folder is selected, each will be represented as their own component in the overall package definition (for example the 2 database folders or 3 middleware folders shown above).

Once a package is defined it can be extracted through the Extract menu. In performing a package extract a "**target extract directory**" is specified. This defaults to the current local root directory but can be changed. When performing an extract each component is extracted to the target extract directory **PLUS** the "**Extract to**" location (if any is specified). Thus if the target extract directory is specified as below, then the database components in the example above will be extracted to the `C:\tmp\componentsPackage\db` folder.

The "**Extract to**" location for a particular package component is the **relative** directory into which it should be extracted.



When extracting a package a Bill of Materials report will be produced (unless the "Create BOM" option is un-checked) just like the other extraction methods (release and/or IR). The BOM includes a textual report (file list) with the name "SSCM_PKG_FL_*date_time*.txt" and the full HTML report with the name "SSCM_PKG_BOM_*date_time*.html". The BOM reports detail all the files extracted and their specific version numbers and locations.

The BOM is also available on-demand directly from the package management screen "**Package Detail Report**" button. The "**All Packages Report**" produces a summary report of all the packages and their components as they are currently defined.

Package Management is aligned with Release Management and the Generic Engineer role from the perspective of permissions. Therefore only users with "Create New Generic" permissions will be able to access the Package Management screen. The user will then only be able to access the projects to which they have the "Create New Generic" permissions in that project. Administrators and system-level Project Engineers have this permissions level for all projects.

Chapter

10

10 Reports

In this chapter you will become familiar with all the SpectrumSCM reports - how to create them and the meaning and use of the information in these reports. You will also learn how to personalize your reports and save personalized versions as well has how to create a custom report and install it in the system

The Spectrum system provides comprehensive set of pre-defined reports. SpectrumSCM provides all the reports necessary for the system to be as useful as possible right out of the box. The following list describes the basic reports that are packaged with the **SpectrumSCM** system.

- **Change Request Report**: Show the details of the specified change request.
- **CR Dependency Report:** Show the dependencies of the specified change request.
- **CR WBS Report:** Summarizes the parent-child and peer-to-peer dependencies for a selected CR
- **CR by State, Severity and/or Date**: Summarize CRs by State, Severity and/or creation date.
- **CR by their Attribute Values - Tabular:** Summarize CRs by their attribute values and list the summary information in tabular format.
- **CR by their Attribute Values - Detailed:** Summarize CRs by their attribute values and list their detailed information in CR Report form.
- **Outstanding Change Request Report:** Show the list of all active change requests.
- **Specific Creator and Period Change Request Report:** Show the change requests created by a user within specific period.
- **User Change Request Report:** Show the change requests currently assigned to a user.
- **CR Assign History:** Show the change requests assigned to the specified user within a specified period.
- **Change Requests Assigned to Release:** Show which CRs are assigned to a release.
- **Change Requests Not Assigned to Release:** Show all CRs that are not currently assigned to a release.
- **Files Touched in a Release:** Show which files were modified in a release
- **Files Version Numbers in a Release:** Show the version number of all files in a release
- **File History Summary:** Report all file changes including who made them and when.
- **File History Detail:** Report all file changes including the summary info and the actual code changes.
- **CR File Diff Report:** Show the file changes made under the selected Change Request.
- **File Status Report:** Report the status of all files under a specified directory.
- **User Checked Out Files:** Report all the files currently out for edit by a user or all users.
- **Generic Audit Report:** Compares two generics and reports the differences (common, uncommon, outgoing and incoming files).

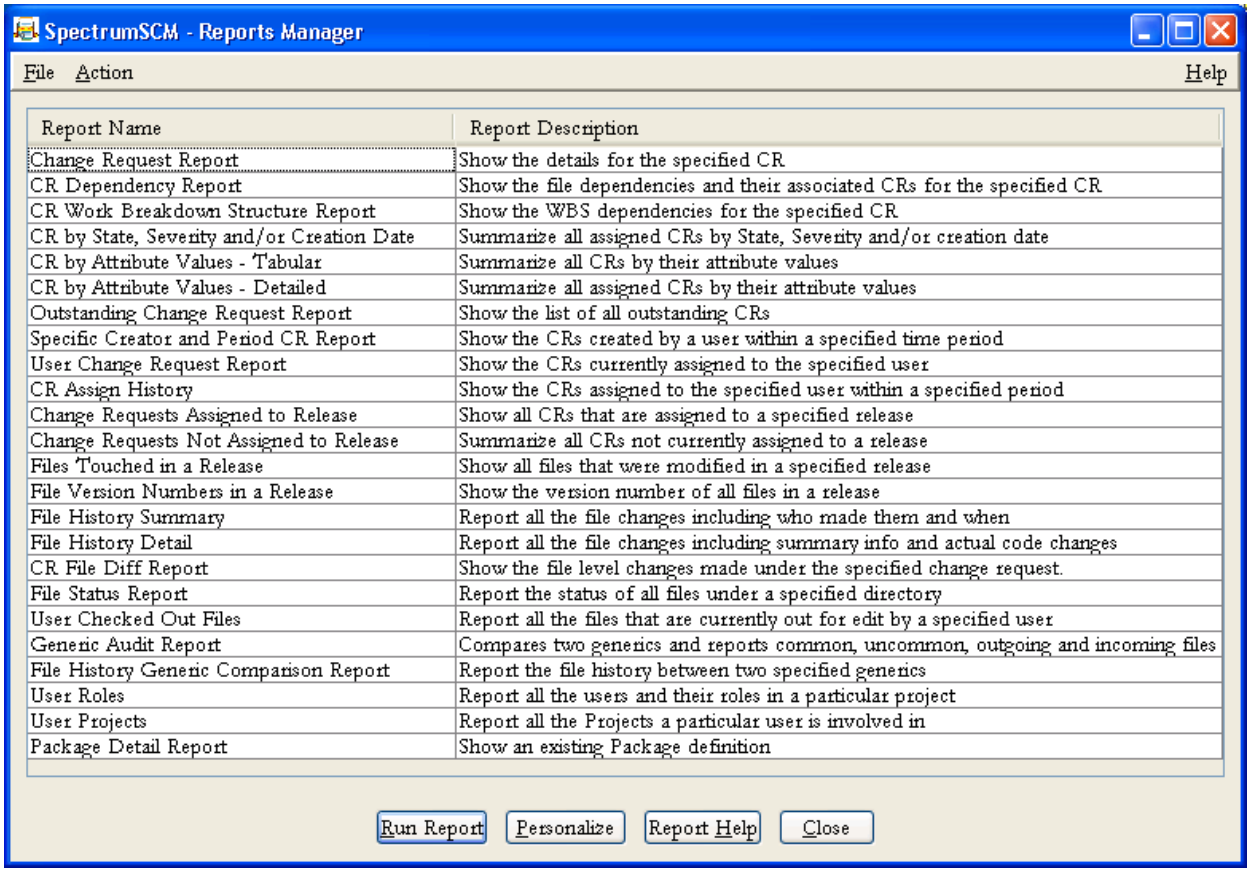
- **File History Generic Comparison Report:** Report the branching activity on a file with repsect to its source generic and one other.
- **User Roles:** Report all the users and their roles in a particular project.
- **User Projects:** Report which projects a particular user (or all) is involved in and what roles they hold within those projects.
- **Package Detail Report:** Show the components currently associated with a package and optionally report the file versions that would be extracted.

10.1 Running a Report

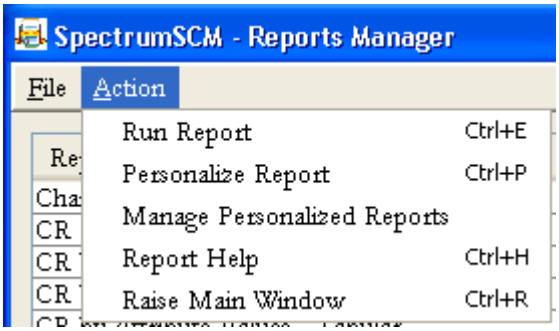
Access the reports via the **Reports** option on the **Main Screen** menu

Select Report

Select the report that you desire, a brief description is listed to the right. More report specific help can be displayed by selecting the help button.

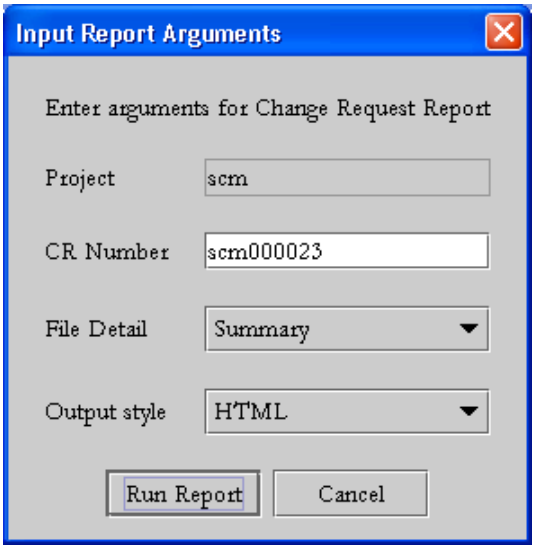


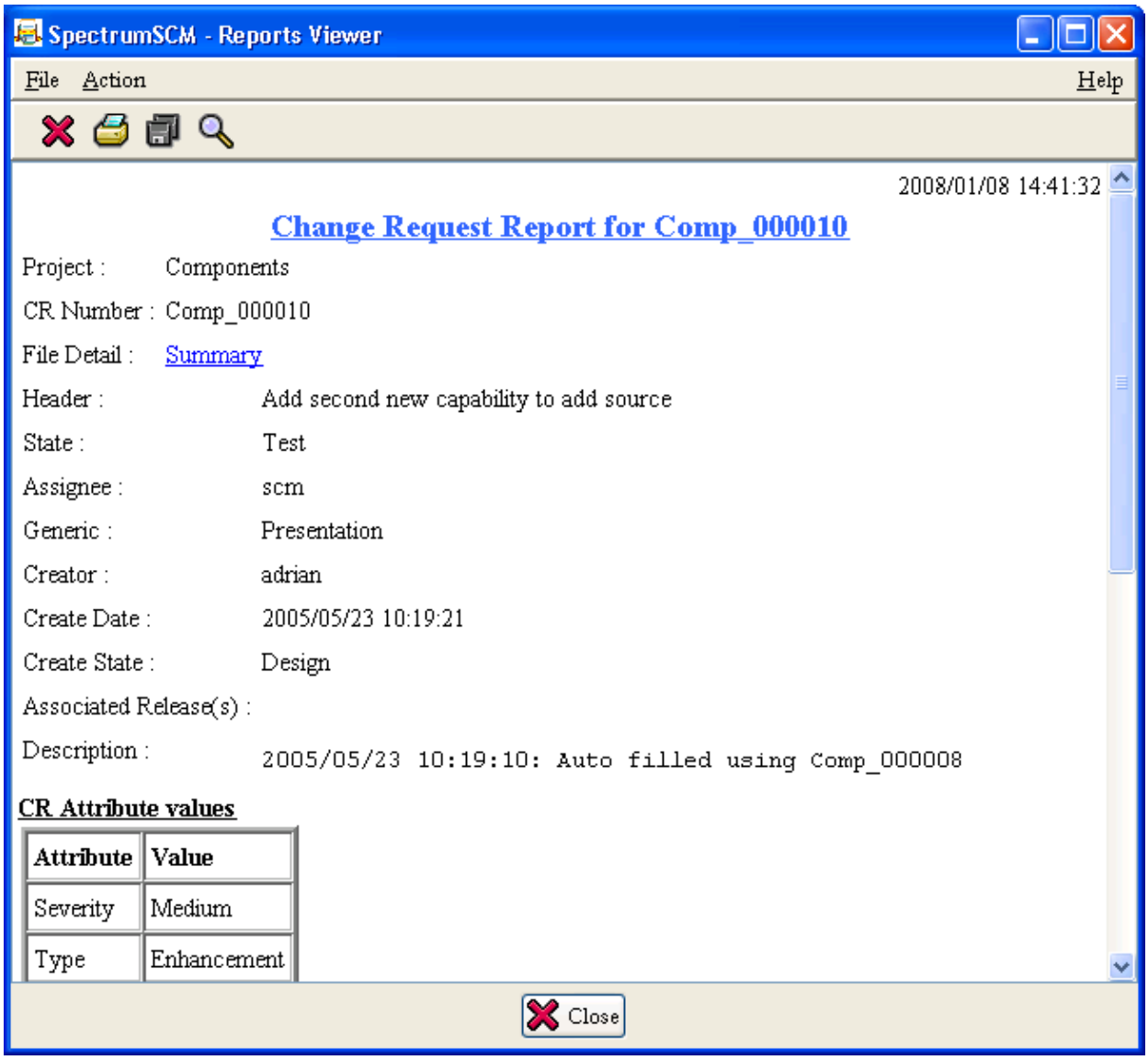
The **Action** menu item offers the same options as on the screen plus an additional option to Manage Personalized Reports (see below for details).



Run a Report

When the **Run Report** button is selected a panel will be presented requesting all of the required input parameters for the chosen report. Some of these will be defaulted to values selected on the main screen. Enter any changes or refinements to the report parameters and run the report. The report will be presented in the Report Viewer. You can output the report either in **HTML or CSV format.**



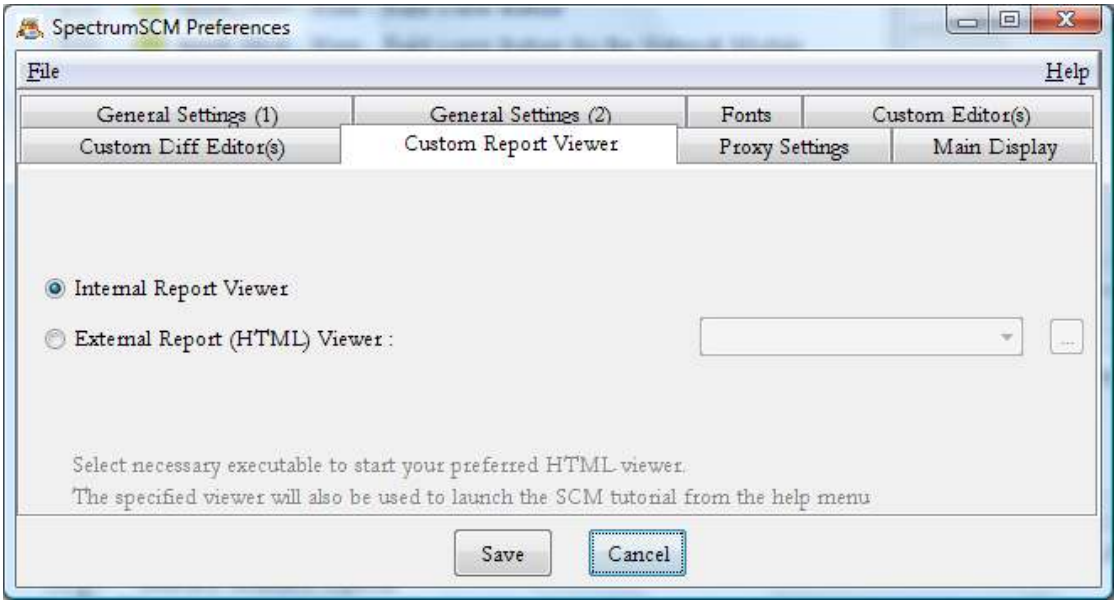


The SpectrumSCM Report Viewer has standard print, save and find capabilities. Under the SpectrumSCM release 2.6 hyperlink functionality was also added to the reports and the report viewer to allow "zoom-in" on items of interest. In the report example shown above, the standard CR Report is run in "Summary" mode which only shows the summary of the files edited under this CR. Selecting on the "summary" hyperlink will switch the report into the "Detail" mode which will show all the file versions created and all the generics/branches affected by those edits.

10.2 Customize the Report Viewer

The default report viewer works fine for all reports, but is limited in functionality outside of printing and simply formatting the HTML output of the reports. The user can elect to set and use a custom report viewer for viewing, printing, mailing or other actions that are outside of the capabilities of the standard report viewer.

To set a custom report viewer, open the Preferences popup (**Edit->Preferences…**) and select the "**Custom Report Viewer**" tab:



In this example, the custom report viewer has been set to Internet Explorer. It could also have been set to Netscape Navigator or some other HTML/web browser. Microsoft Excel and other spreadsheet programs that understand HTML and also table formats (for sorting, filtering and extended data processing) can also be useful choices. On the Windows platform the browse button automatically opens up the browse dialog window in "C:\Program Files\Internet Explorer". Once the customer viewer is set, all report output is sent to the chosen custom report viewer.

To set the viewer back to use the SpectrumSCM report viewer simply set the custom report viewer field to blank.

10.3 Printing and Saving a Report

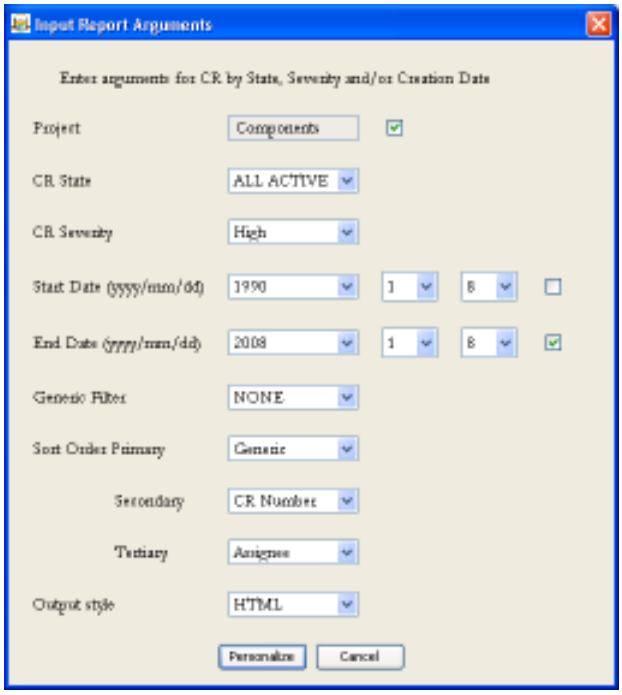
If the default report viewer is used for displaying the report, the report can be **printed or saved** as a HTML, text, or Comma Separated Values (CSV) file through the report viewer.

10.4 Personalizing a Report

Reports can be personalized by selecting the **Personalize Report** button. Personalization allows the user to pre-specify popular reports and assign them to the main screen reports menu. This is very useful for frequently executed reports i.e. for that Monday/Friday weekly status meeting.

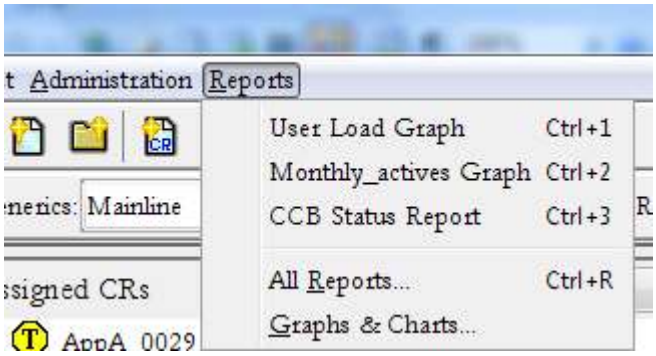
The personalize report button presents an argument panel similar to the one for entering the regular report arguments. If a value is provided directly as a report argument, the value is hard-coded in the personalized report. If a **check box** is available and the user selects it, then the argument value will be retrieved from the main screen selection at the time that the personalized report is executed.

For example, to personalize the CRs by State/Severity report, select the report and (in this case) select the Project and End-Date check-boxes. Then click the **Personalize Report** button.



Click the **Personalize** button to enter the name that you wish to assign to this report and click OK.

This report and all other personalized reports saved by the user can now be executed directly from the main screen reports menu without re-entering parameter values. Quick keys (Control + number 1 thru 9) will execute the first 9 personalized reports.



By checking the Project and "End-Date" during the personalization of this example report, it means that the report will find data on the current project and to the current date when the report is executed. As opposed to always running to the "Components" project and the specified end-date.

10.5 Custom reports

Until the Reports API is available for general use, the only way to create a custom report is to request such a report from Spectrum Software. Report creation is part of your maintenance agreement and most reports can be created and shipped to the customer within 24 hours of receiving the request. Once the report has been delivered to the customer, it's simply a matter of starting the report installer and the custom report will be automatically installed:

```
$ java –jar ReportInstaller.jar
```

The installer automatically installs the report in the custom reports directory and the report itself is immediately available for use on the reports screen. There is no need to restart the server after installing the report. On some operating systems you may be able to simply double click the ReportInstaller.jar file to invoke the installation.

11 Branching, Merging and Re-common

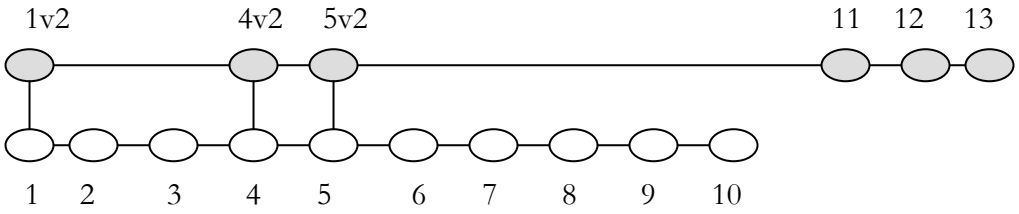
Chapter 11

SpectrumSCM provides a very simple but extremely powerful way to make and manage branching decisions. Branching in the SpectrumSCM system is handled very differently from most other CM systems, providing many more options in how branching is handled. SpectrumSCM supports numerous branching patterns, allowing simple and effective management of activities, source or project artifacts during the various scenarios that could occur in any project's life cycle.
In this chapter you will learn how to manage branching in SpectrumSCM, how to merge two different versions of files, how to recommon two different versions of files, and how to use the merge editor to merge and recommon files across branches.

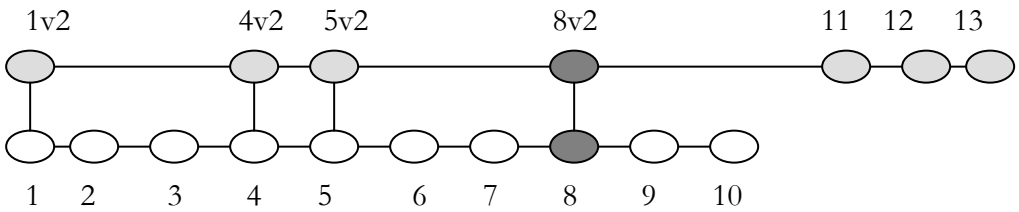
11.1 Branching

The concept of branching in an SCM system can be one of the hardest concepts for users of the system to understand. Simply put, branching is a deviation from a main line of file changes. Branching is often used to aid in parallel development and for creating new project feature sets or file content in support of some special needs.
In the SpectrumSCM system, a branch is known as a ***generic***. A generic is a special form of branching that is highly visible to the user and is not limited to single file branching. That is, multiple files may join the same generic "branch" in order to form a specialization of a product or project.
A generic then, is a single branch structure that can accept one or more files off of the main line development stream or another generic. Each generic contains the file changes necessary to customize the product for different platforms. Creating an editor for each OS platform then is as simple as creating a release, or multiple releases, on each generic.
Generics can be thought of as containers for custom work and pointers back to common components. Except for the customizations needed in particular files, all files in a generic share the same physical disk space and physical instances of files with the main line or previous generic upon which it is based. Files that need to be different for the generic are uncommoned. If a file has been uncommoned among generics, it can be recommoned (made the same).
Uncommoning a file is creating two physical disk images of a single file thus creating divergent images of the same file. **Recommoning** is the act of bringing the divergent files back together into the same physical disk image.

As an example, consider a part of the development team being charged with developing advanced features for the editor, in parallel with the current work. The new work could be done in a separate generic on uncommon files. Once the work has been completed and tested, the new features for the editor can be rolled up into (recommoned back) with the main development code stream.
As an example, if a project team has developed and released version 1.0 of a system and they are currently developing generic 2.0, all modules that are changed (modules 1, 4 and 5) or added (modules 11, 12 and 13) during the 2.0 development effort will be "uncommon" - changed only for generic 2.0.

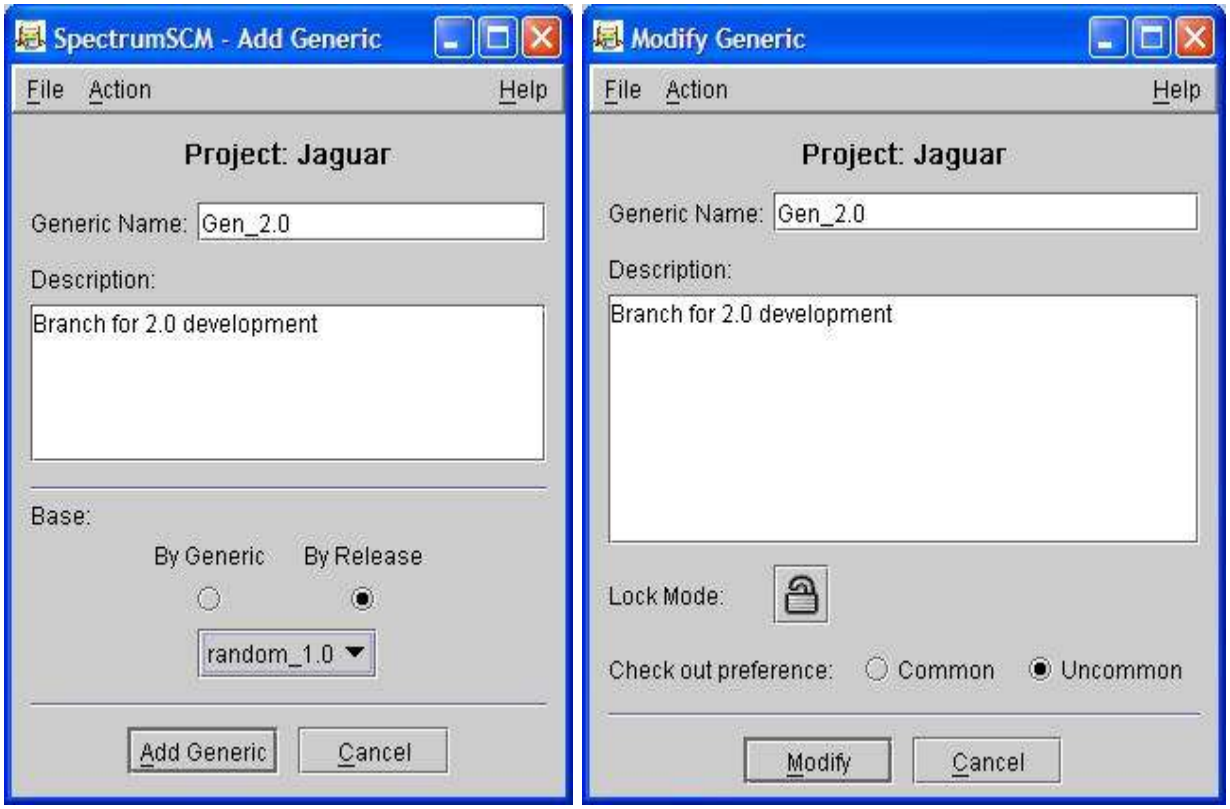


However, if during the development of generic 2.0, a problem is discovered in Release 1.0, the fix for the problem might be made common to both generics. In this example, the problem is in file 8. A CR is created, the code is edited, the problem fixed and the fix is made common to both generic 1.0 and generic 2.0. If the fix needs to be distributed, Release 1.1 can be created, containing the contents of release 1.0 and the fixed file 8. Development of Generic 2.0 can proceed, knowing that the fix will be carried forward.



As another example, there are times when multiple generics are developed in parallel (for example, to maintain 2 similar source bases for 2 different customers). The Generic Engineer must decide who is going to make the branching decisions – who will determine which of the modules will be common or uncommon with other generics.

When a new release of the system is being developed, the changes and additions are based on the previous generic or release as defined on the Add Generic Screen.



(See chapter 6 for details on creating a new generic)

Changes and additions are typically made uncommon to the previous release. Notice that the default checkout mode (checkout preference) is set on the Modify Generic Screen. If the generic is locked, all changes made on related generics will uncommon the associated files on this generic. If the generic is unlocked, the developer can choose to make a specific change uncommon or common. The Generic Engineer can lock a particular generic to prevent inadvertent changes to other generics or to enforce process control issues. The generic can be unlocked at a later stage, if a need arises to make a common change.

In overview, checking out "uncommon" will mean that any file changes will be isolated to that specific generic. If a checkout is performed "common" then the file changes will be made against ALL the generics that that file is currently in common with.

- **Common:** Versioned files that are physically the same across generics
- **Uncommon:** The act of physically separating versioned files from multiple generics

Checking out "common" is a powerful feature since it can be used to apply a single "fix" to multiple generics in one edit, however the developer would have to be careful of side-effects.

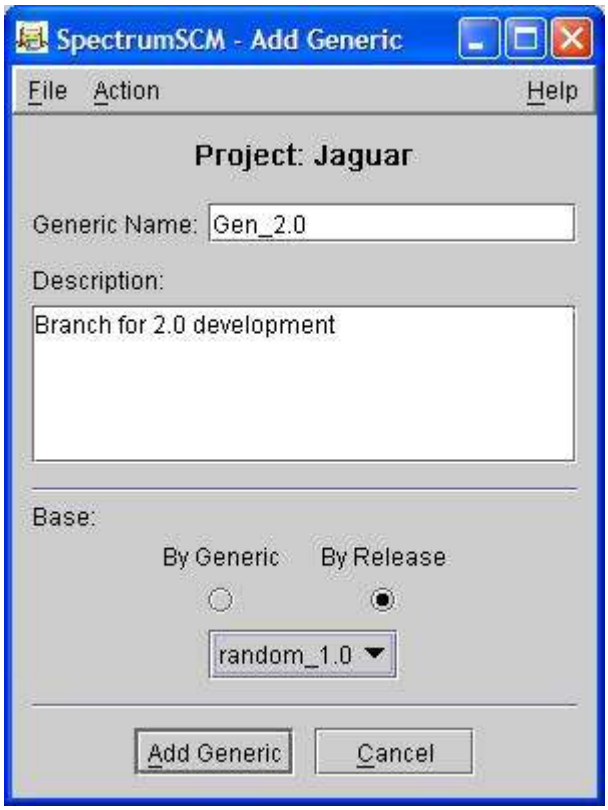
The ability to create generics solves the basic file branching problems. A generic is, in essence, one large branch. Files may be added to the branch or remain common with the other files in a previous or concurrent generic. By creating a separate generic to represent a unit of work it becomes much easier to keep track of all the changes necessary to create a new product release. Consider the example of developing an operating system. The initial generic might be developed to create an OS

for the Intel platform. By creating a new generic where all the files are initially common between the original and new generic, we can easily break the commonality between generics for specific files in order to create an OS for the Sparc platform. Generics allow us to more easily organize, manage and track our source files for the purpose of creating releases and managing current and future work.

11.2 Creating a Branch (Generic)

(See chapter 6 for details on creating a new generic)

1. Determine the generic or the release upon which the new generic will be based. When a new release of the system is being developed, the changes and additions are based on the generic or release selected on the Add Generic Screen.



2. Create the new generic by selecting the **Administration/Create Generic** menu option. Select the radio button, **By Generic** or **By Release,** based on step 1 above and click on the ***Add Generic*** button.
3. Select the **Administration /Modify Generic** menu option. This will take you to the **Modify Generic** Screen.

4. Determine whether you want to leave individual common/uncommon edit decisions to the relevant developer or mandate by clicking on the **lock icon** on this screen. The default mode (Unlocked) leaves the choice with the developer since they would best know the source issues.
5. The new branch is now created. Be aware that creating a new generic will lock all releases on the predecessor generics.

To start editing files under this branch, create the CR (s) under this generic by selecting on the Generic in the Project Tree window and clicking on the **Change Request / Create** menu option or by assigning existing CRs for work under this generic. Note that CRs that have already been worked under one generic cannot be reassigned to another generic.

11.3 Merge and recommon

Merging allows two branches to synchronize the contents of the files and still retain the separate development paths. Merging in SpectrumSCM is the act of copying the changes made in one version of a file into another version of the file using the SpectrumSCM Merge Editors. All changes need not be made identical in both files.

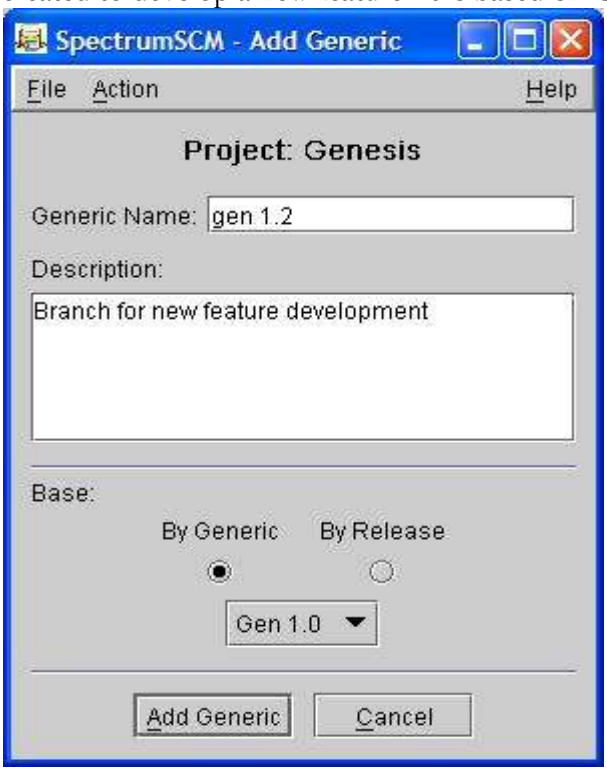
For Example, Team 1 may have made fixes to a file while Team 2 was making different changes to the same file as part of the mainline development. The changes made by Team 1 need to be incorporated (merged) into Team 2's version of the file.

Using the SpectrumSCM Merge Editor for Merge and Recommon
The Split Screen Editor is used for the Merge and Recommon functions. For these functions, modules must be checked out for Merge or Recommon.

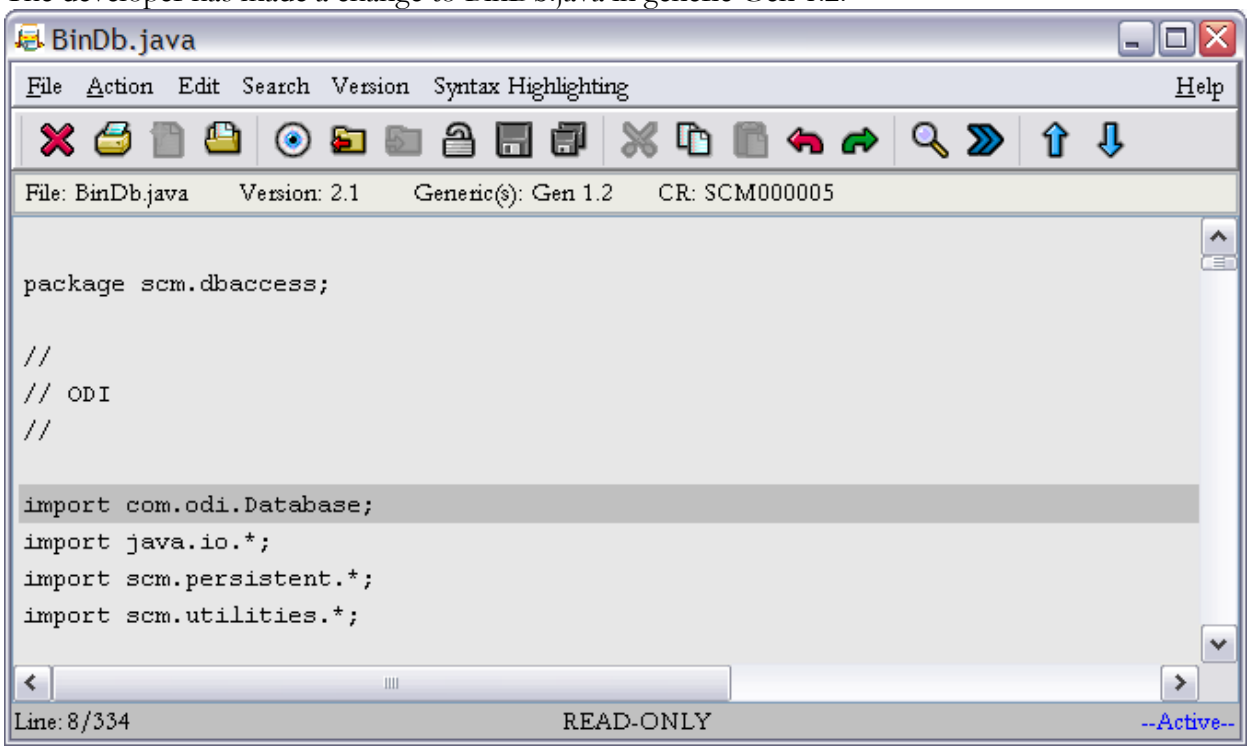
In both the Merge and the Recommon processes, the files selected will be put in the split screen **Merge Editor** to allow the changes between the two versions to be identified and reconciled. This will produce a single version of the file that is then made common to the two generics in the case of a Recommon or reconciled and kept separate in the case of a Merge.

NOTE: When doing a recommon, the contents of both files in the editor must match when the check-in button is pressed. The editor will tell you if this is not the case. Merging does not have the same restriction.

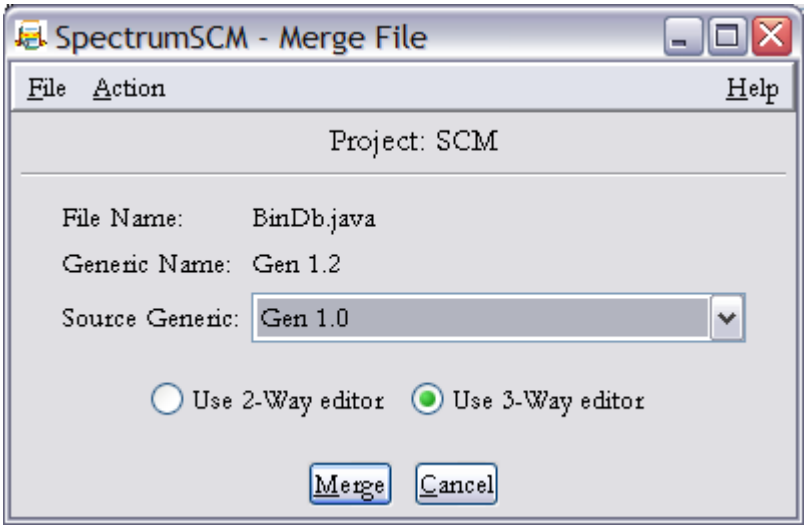
Merge
To start the merge activity, from the SpectrumSCM Main Screen use the **Extract/ Check out to desktop** menu item and select both files to be merged and check them out with the **Merge** option. Open one on each side of the Merge Editor. This allows use of the SpectrumSCM Merge Editor to see and reconcile the differences between the two files and complete the merging. In this example, Genesis generic gen1.2 is created to develop a new feature. It is based on Gen 1.0.



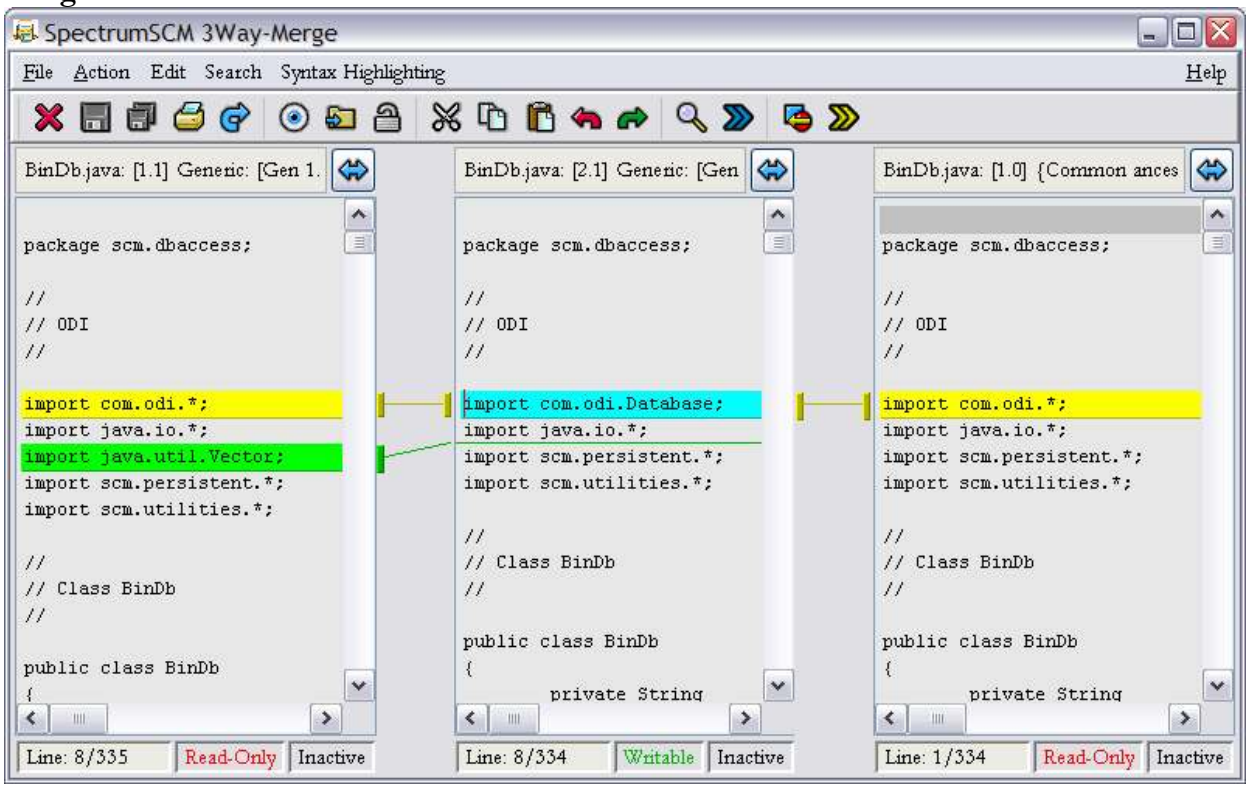
The developer has made a change to BinDb.java in generic Gen 1.2:



The same developer has changed the same file in generic Gen 1.0. He would like to merge the changes between the two generics. To do so, he selects the file BinDb.java from the later generic (1.2) and checks it out to the desktop to merge options using **Extract / Checkout to desktop / Merge** menu options. A confirmation screen is shown. Click Merge to continue.



The choice of merge editors will default to the merge editor type selected by the user in the user's preferences options. Once the editor has been started both versions of the file are displayed in the **Merge Editor**.

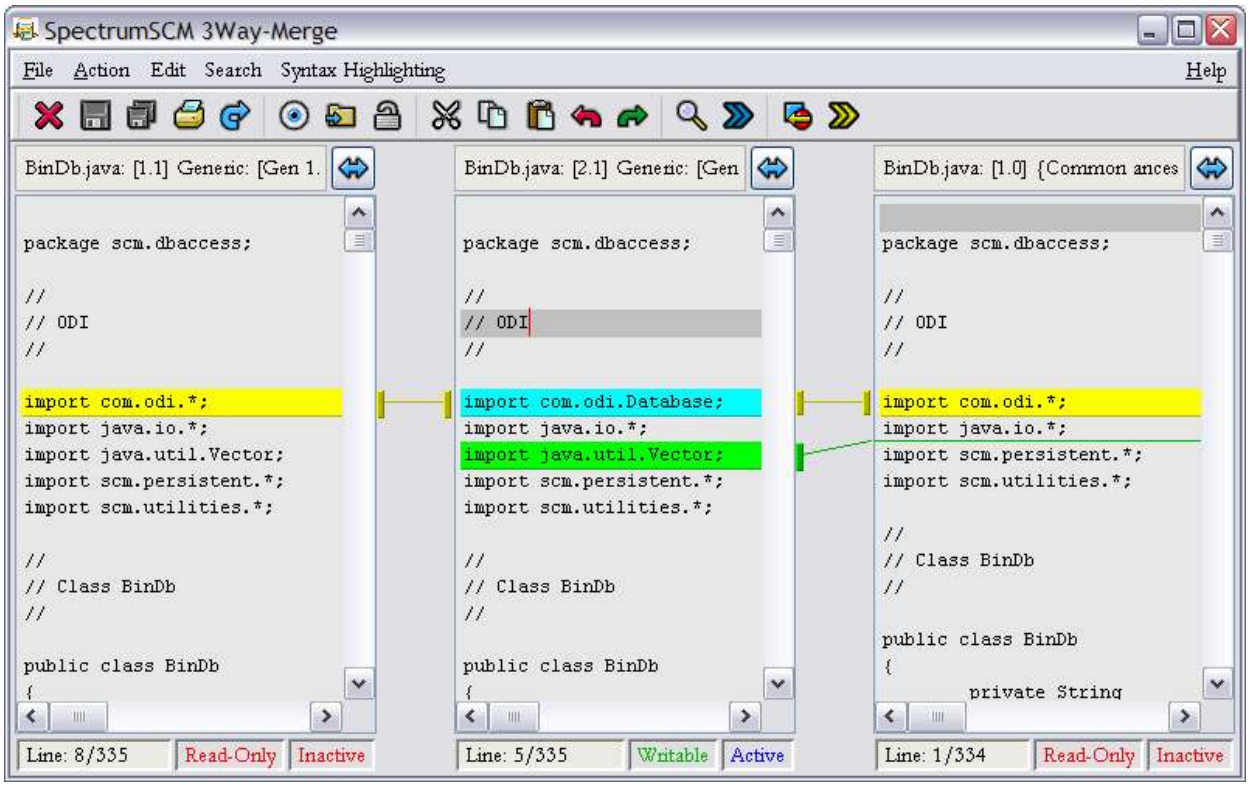


In the three way diff/merge tool (displayed) all diffs are done left to right relative to the common ancestor file on the far right hand side. That is, the branched file on the far left is compared relative to the second branch file in middle pane and then finally the second branch file is compared to the common ancestor on the far right. The user can choose to swap the position of the two branch files so that the comparison can be done against the first branch file and the common ancestor.

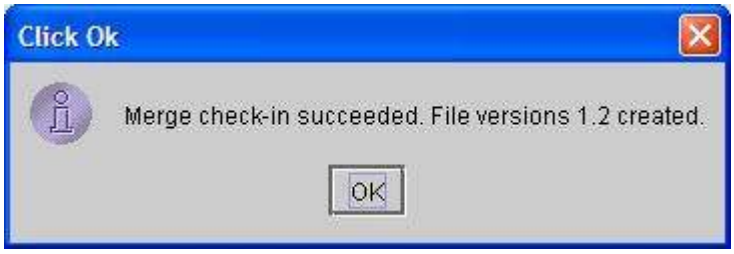
In the 2-way diff/merge tool, the user can run the diff analysis in either direction, which will have an affect on how the actual difference is displayed to the user. For example an insert in one file will be a delete in the other or vice-versa, depending on which "direction" the user chooses to run the diff.

The Merge Editor(s) highlight the differences between two versions of a file and the common ancestor (3-way). Inserts show up in green, changes in yellow and deletes in red.

In this case the first branch file has added a single line relative to the second branch file and the second branch file has a single line difference (conflict) with the first branch file. The insert from the first file to the second is highlighted with a green bar and the conflict between the two editors is highlighted with a yellow change bar. Note that the change line in the second branch file is actually color coded cyan. This is an indication that this change is an actual conflict between the changes relative to the common ancestor and must be resolved by the user manually.



In this case the developer has resolved the change by selecting the change bar in the first branch file and applying that change to the second branch file with a mouse click. He then selects **Check-in** button and since only the file from gen 1.2 was changed, only one file was checked in.

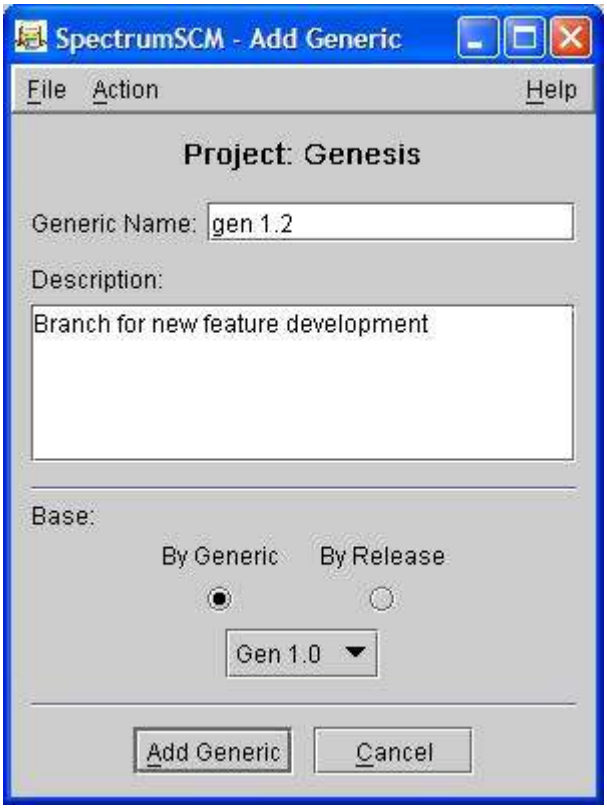


Generic 1.0 still has its version 1.1 file unchanged.

Recommon

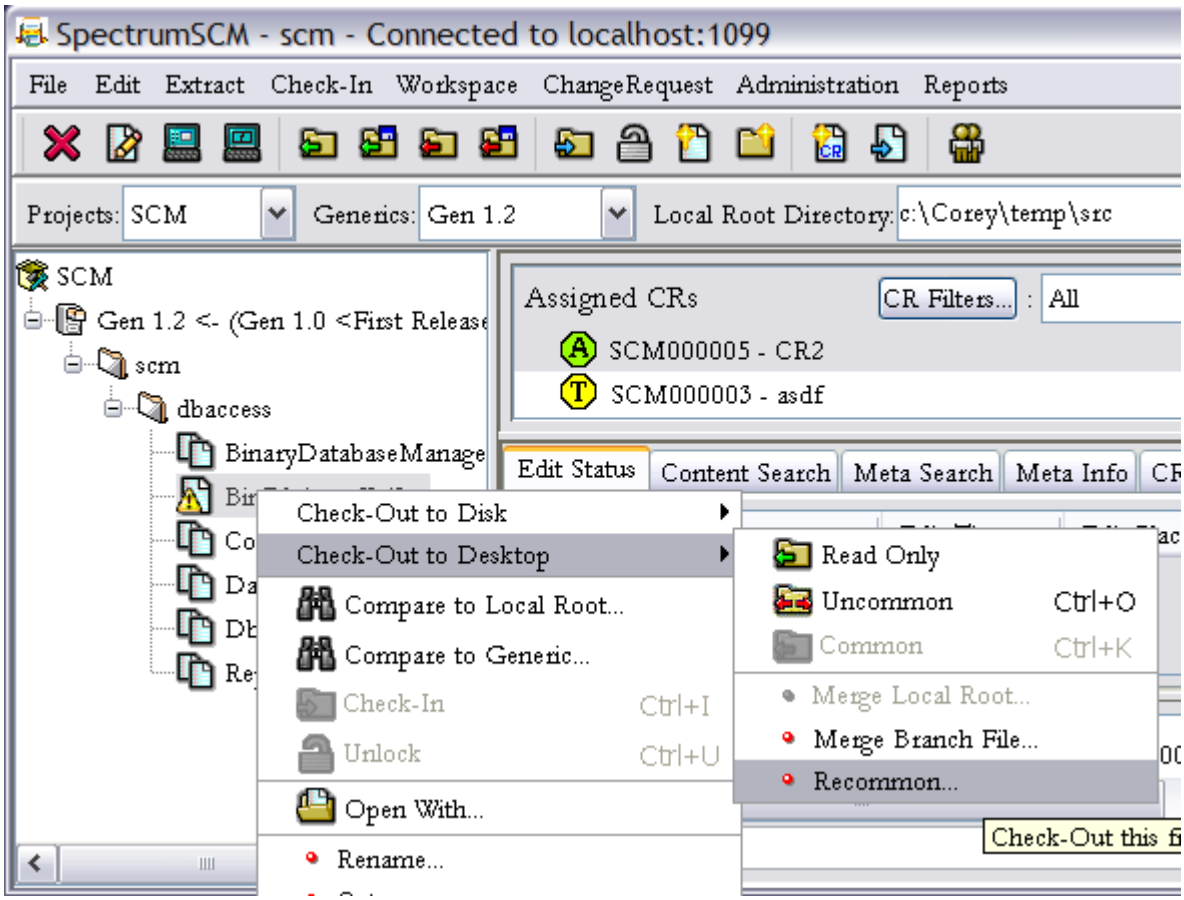
Recommoning brings two versions of the same file (in two different generics) back into one version shared between the two generics (makes them common again). This is useful when a parallel development effort on a project is brought back together to create one code path.

Recommon requires that the later generic has been set up with a previous generic as a basis and that commonality to the previous generic is allowed. In this example, Genesis generic gen1.2 is created to develop a new feature. It is based on Gen 1.0.

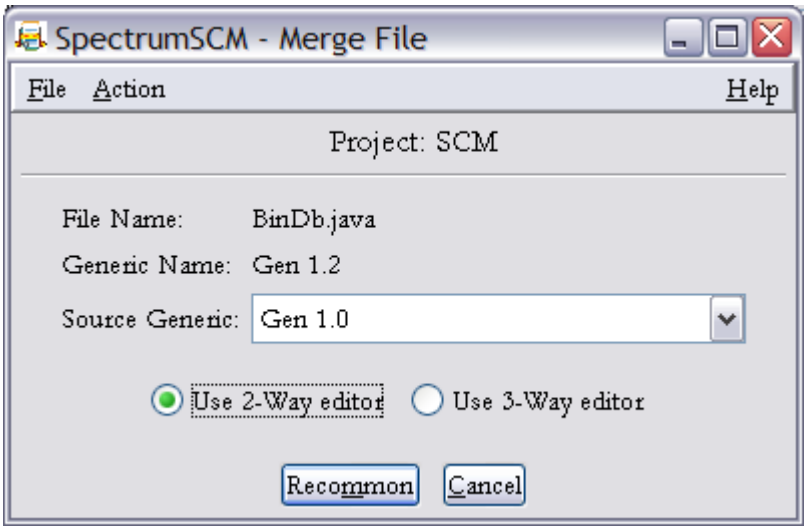


Recommoning is the act of bringing the divergent files back together into the same physical disk image. For example, recommoning could occur when part of the development team is charged with developing advanced features for the editor in parallel with the current work. The new work could be done in a separate generic on un-commoned files. Once the work has been completed and tested, the new features for the editor can be rolled into (recommoned with) the main development code stream.

Continuing from the previous merge example, the user has chosen to re-common the file BinDb.java in generic Gen1.2 with the same file in Gen1.0.


SpectrumSCM - scm - Connected to localhost:1099
File Edit Extract Check-In Workspace ChangeRequest Administration Reports
Projects: SCM
Generics: Gen 1.2
Local Root Directory: c:\Corey\temp\src
SCM
Gen 1.2 <- (Gen 1.0 <First Release
scm
dbaccess
BinaryDatabaseManage
Assigned CRs
CR Filters...
All
SCM000005 - CR2
SCM000003 - asdf
Edit Status
Content Search
Meta Search
Meta Info
Check-Out to Disk
Check-Out to Desktop
Compare to Local Root...
Compare to Generic...
Check-In Ctrl+I
Unlock Ctrl+U
Open With...
Rename...
Read Only
Uncommon Ctrl+O
Common Ctrl+K
Merge Local Root...
Merge Branch File...
Recommon...
Check-Out this fi

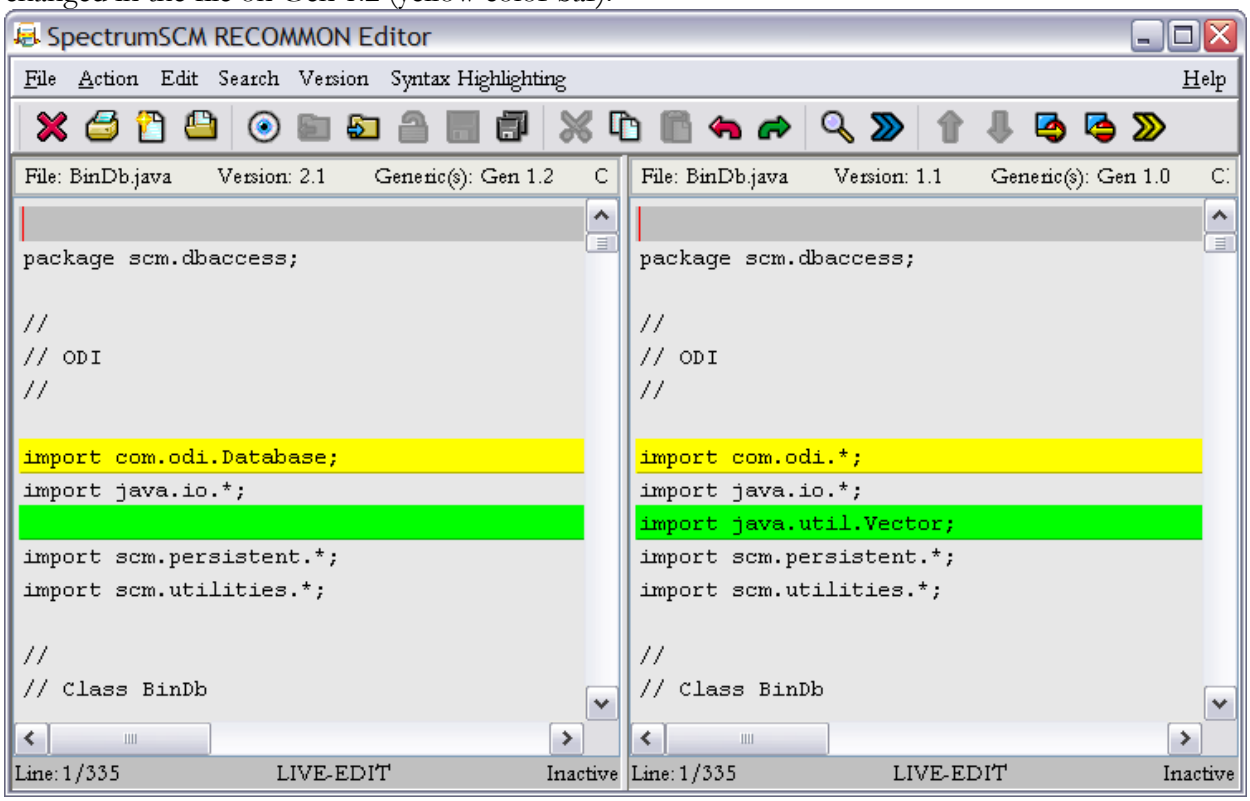
To start this activity, from the SpectrumSCM main screen use the context sensitive menu on the tree view and select **Check-Out to Desktop->Recommon…**. SpectrumSCM displays the source generic (gen 1.2) and the base generic (Gen 1.0). When the re-common action has been completed, the file from these two generics will be recommoned.



In this case the user has chosen to use the 2-way diff/merge tool to perform the re-commoning operation. The BinDb.java file from Gen 1.0 and Gen 1.2 are brought up in the Merge Editor.

The Merge Editor highlights the differences between two versions of a file.

Use the  buttons to base the difference from either right to left or left to right. Inserts show up in green, changes in yellow and deletes in red. Depending on which direction you base the differences, an insert (green) one way will be a delete (red) in the other.
In this case, a single line was added to the file in Gen1.0 (green color bar) and a single line was changed in the file on Gen 1.2 (yellow color bar).

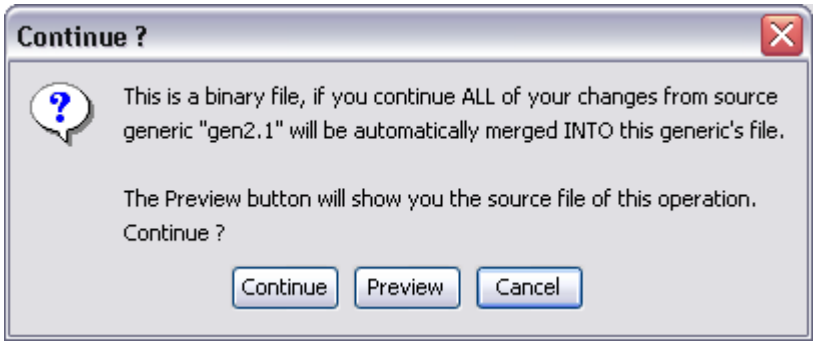


Use the mouse to select each change and then apply the change in the direction that the diff was executed. Use the  buttons to continue to check for differences in both directions. Then use the **Check-in** function to recommon them into both releases. When doing a recommon, the contents of both files in the editor must match at the time that check-in is selected. The editor will tell you if this is not the case and check-in recommon will be blocked until the files are identical.
When the files are identical, both will be checked into their respective generics.

Merging and Recommoning binary files

Since the SpectrumSCM dual editor can only be used with text files, the merge/recommon operations for binary files proceed differently.

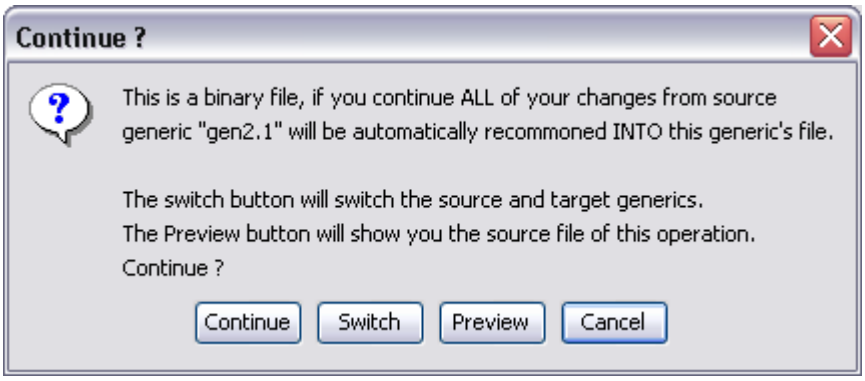
With a merge operation you will be presented with the following popup –



The source generic will be the one you selected at the start of the merge operation. Selecting the **Preview** button will open your custom editor on the version of the file under the source generic. *See Chapter-5 for details on how to define custom editors*. This will be the version that will be merged into the current file in the generic where you are performing the edit.

The **Continue** button will proceed and automatically perform the merge, overlaying the current version of the file with the contents from the other generic. Since all the changes are versioned, you can still step backwards through the version history if necessary.

With a recommon operation, the popup is slightly different –



Specifically, because the edit is occurring on both generics, there is a "**Switch**" button which allows you to choose which file should be the source of the recommon operation. Use the **Preview** button as before to see the version of the file that will become the head revision.

Once the **Continue** button is pressed, the recommon operation will complete automatically.

11.4 Using Branching Patterns for Configuration Management*

*Based on *Advanced Branching Techniques for SpectrumSCM, a White Paper written by William C. Brown, 05/14/2002*

The technical execution of creating a generic, creating, merging and recommoning files is simple compared to the task of determining how best to create branches to support the specific needs of the system development effort. Over the years, developers and system engineers have developed many unique branching techniques to solve difficult configuration management problems. The

purpose of this section is to describe several of the most common branching techniques and to illustrate how these techniques can be implemented using the SpectrumSCM system. SpectrumSCM approaches branching in a significantly different and more powerful manner than most CM systems. The SpectrumSCM system introduces the concept of *Product Level Branching* that is unique to the CM industry. Product level branching ensures that branches are well known (documented), controllable and use repository space as efficiently as possible.

Like design patterns used in programming techniques, the application of proper design patterns to configuration management will result in the development and evolution of systems that are more maintainable, understandable, extensible and scalable. The use of a CM system should not become a burden by adding to the workload of the development team. A properly used CM system should free the developer from the intricate details of branching and release management. The proper application of branching design patterns can result in systems that are as easy to use and maintain after years of activity.

SpectrumSCM does not impose any one branching design pattern on the users of the system. Users of SpectrumSCM are free to use many different branching design patterns, including all of the patterns outlined in this paper. Most developers are familiar with the most common branching technique, which involves branching single files during code development. For a short period of time, the code is extended in a branch to resolve a particular problem or to introduce a new feature outside the mainline development effort. The branched code is eventually *merged* back into the mainline after the fix has been verified or the new feature set has been implemented. While this is a common technique, and one that is supported by SpectrumSCM, it's not the best solution for every situation.

The following design patterns are supported by SpectrumSCM and, in some instances, are unique to SpectrumSCM.

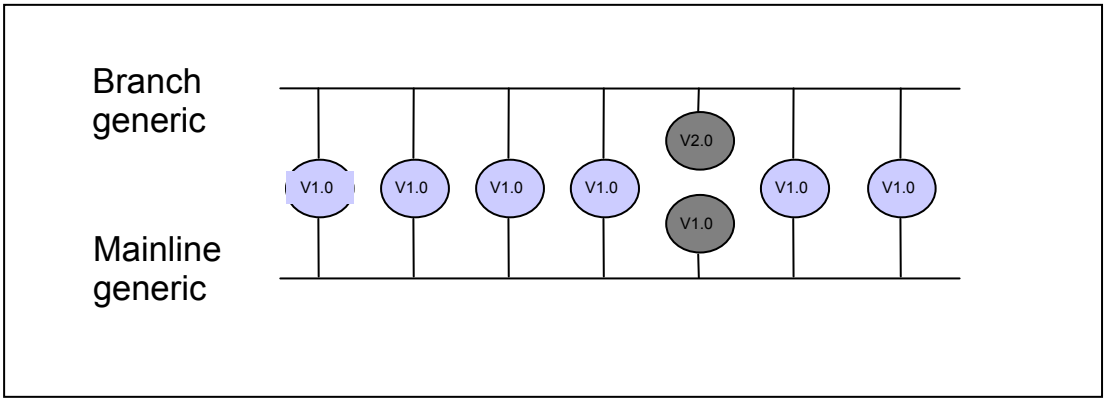
- **The Classic branching design pattern**
- **Parallel Development pattern**
- **The Sandbox pattern**
- **The Promotion (Repository) Pattern**
- **The Patch Pattern**

The Classic Branching Design Pattern

The classic pattern is the basic branching pattern outlined above. This pattern is the most recognized and most often used pattern for branching code. The classic pattern allows individual developers to individually create alternate branches of code extended from the mainline development stream. Without a strong CM system such as SpectrumSCM, the existence of such a branch is not immediately obvious to the other users of the system.

Traditionally, this type of branching is done at the file level and the branched files are only conceptually linked to a specific branch by a branch number embedded in the version number of the file, for example in systems based on RCS (Revision Control System) where the third digit in the file version number is greater than "0" (file version 1.3.1.2 might mean that a branch for this particular file was formed at version 1.3 and is now at version 1.2 of the new branch). Creating a release with the proper file versions can be difficult at best.

In the SpectrumSCM system, branches are first class objects in the system and their existence is readily apparent to the users of the system. To perform ***classic*** branching in the SpectrumSCM system, a generic is created to ***contain*** the branched files. Notes and other artifacts can be associated with the branch to assure that the purpose of the branch is known and available to every user of the system. When a generic is created from another generic or release, all files shared between the original code stream and the new generic are ***common*** to the two streams. This means that only one first class object for each file physically exists in the system and both branches point to that object. Actual branching is accomplished by ***uncommoning*** a file from the mainline into the branch. When a file is ***uncommoned***, there are two first class system objects, one for each version of the file. The following diagram illustrates the point:



In this example, the dark circles represent a single uncommoned file. Each branch contains a separate physical instance of this file and shared instances of all the other files. When the files are ***recommoned***, both branches will again share a single file instance.

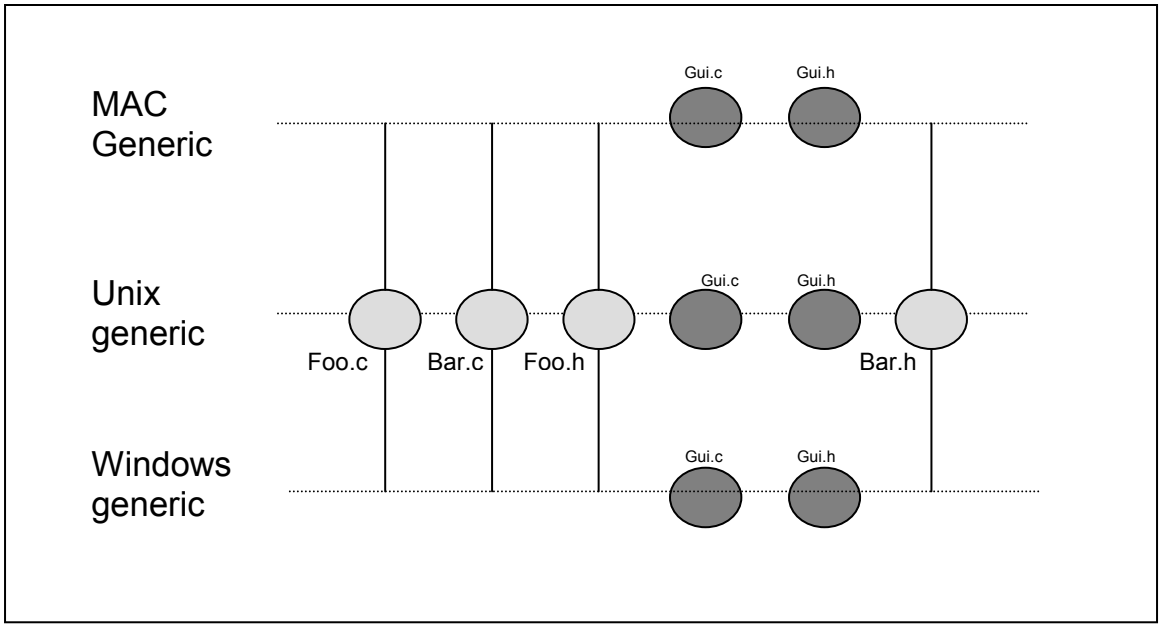
The objective of the classic branching pattern is to diverge one or more files from the mainline, usually for a short period of time, so that custom work or bug fixes can be applied outside the mainstream development effort. At a later date, the changes are merged or ***recommoned*** back into the mainline development stream.

In the SpectrumSCM system there is a significant difference between merging and recommoning. **Recommoning** makes one single source instance out of two independent entities. **Merging** combines the contents of the two files, but the two files remain physically separate.

The Classic branching pattern is used to diverge small numbers of files from the mainline code stream, for a short period of time, in order to fix a known problem or to implement a new feature. The diverged files are merged back into the mainline code stream when the work has been completed.

Parallel Development Pattern

The parallel development pattern is very similar to the classic pattern in that two or more branches are created, but in the parallel pattern, some files are never merged or recommoned back with the mainline. The parallel development pattern might be used during the development of a product for use on multiple operating systems. The vast majority of the functionality and source files are the same on all operating system, but some files must be unique to support the differences between the operating systems. For example, the direct video calls for any GUI components will most certainly be different and will thus require different code. The implementation of the second or third generic (branch) is exactly the same as in the classic pattern except that some files will never be recommoned. Each generic will become a platform-specific release of the product. The following diagram illustrates parallel development for an editor that will run on three different operating systems:

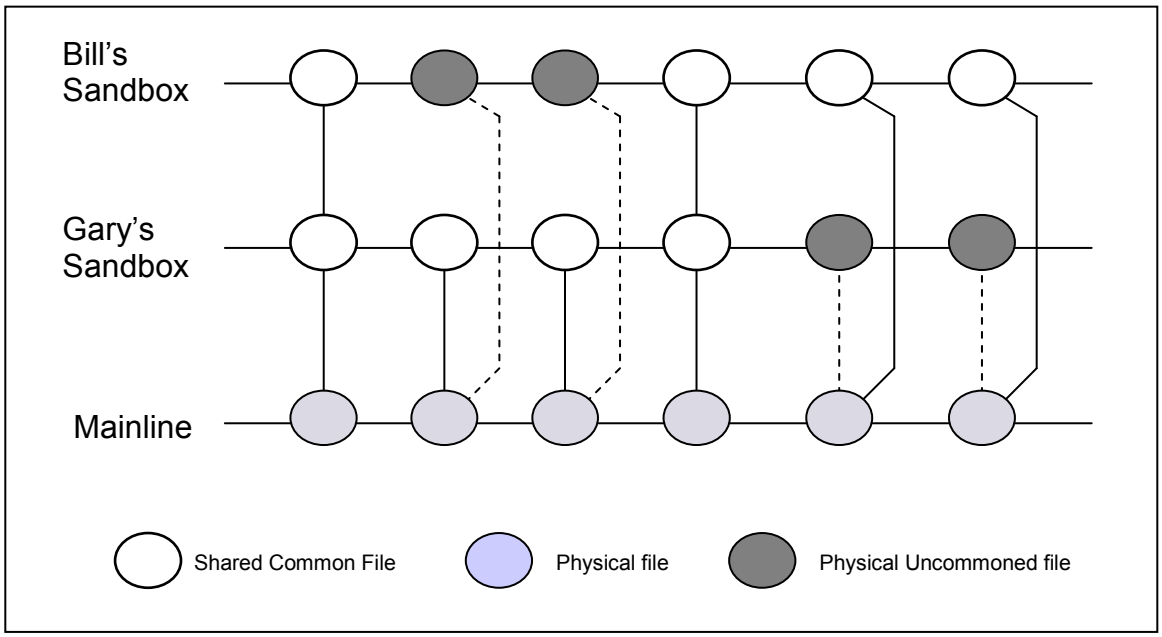


In this case, the files Gui.c and Gui.h are different for each operating system and must remain diverged. The MAC, Unix and Windows generics all share the vast majority of files and only the files necessarily different to implement the GUI on each OS are diverged.

This is where one of the strengths of the SpectrumSCM system becomes very apparent. There are three separate streams of work, one for each of the three supported operating systems. But the vast majority of the files that make up the product are common. As a result, when problems are fixed in these common files, all three generics get the fix at the same time. The CR (Change Request) that is used to resolve the issue is available to be included in a release on all three generics. *This feature, which is unique to SpectrumSCM, relieves the developer from fixing the same bug three times in three separate branches of the code.*

The Sandbox Pattern

In the Sandbox pattern, all work is performed in separate generics before being integrated back into the mainline. The most attractive feature of this pattern is that the mainline branch and the development branch are always in a known good state. New features are first developed in separate sandboxes and then integrated into the mainline only after the new features have been thoroughly tested and approved by the testing organization. When the features are recommoned into the mainline, the developers can extract the code from the mainline and build a system with a known set of working and tested features. This pattern assures that the mainline is never in a quasi-buildable state, which happens quite often in traditional development. The development branch, because there is one branch per developer, always matches what the developer has in her private work area on her machine. The developer is free to work independently on a separate branch without impacting other developers. Consider the following diagram:



In this diagram the transparent circles are the shared common files that are common to the mainline. The blue circles are the actual physical instances of the files that the shared common files point to. The green circles are unshared physical files that have been uncommoned into Bill and Gary's sandboxes. When these developers are finished with their parallel development work, the files will be recommoned with the mainline. After recommoning, the circles will become transparent like the others.

The sandbox pattern enables long-term parallel development. Most CM systems offer some form of parallel development in the form of concurrent editing. The problem with traditional concurrent editing is that it is file-based; as soon as a programmer checks in a concurrently edited file, it must be merged back to the mainline code stream. Sandboxes allow long-term parallel development by allowing the programmer to freely check in and out any amount of code, for any amount of time, without disrupting work that may be in progress on the mainline. Only after the entire new feature has been tested and verified will the new code for the feature be merged back to the mainline branch.

The only caveat to this pattern is that it is an "all or nothing" pattern. All developers on the project team must use this pattern or they cannot use it at all. If files are uncommoned into the mainline

and also into individual sandboxes, it becomes difficult to recommon the files back into all of the parallel generics. Consistent use of the sandbox pattern guarantees that the recommoning effort will be trivial, involving only a single merge of each file. The sandbox pattern closely resembles the parallel development pattern, except that all files will be recommoned into a single generic when development work is complete.

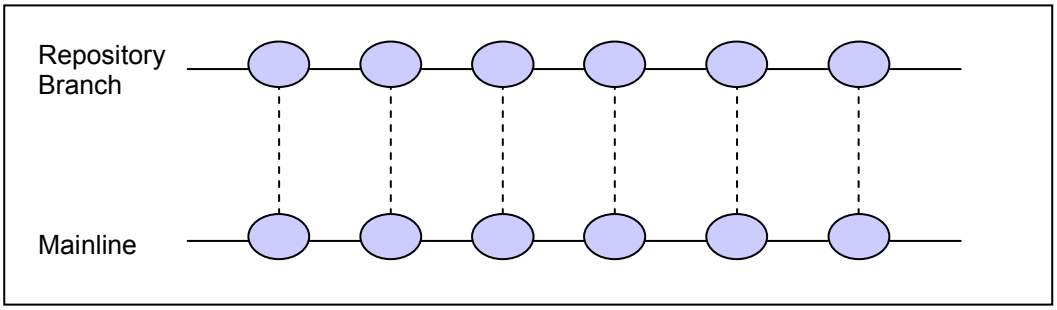
SpectrumSCM may be the only CM system that properly supports this pattern.

The Sandbox pattern provides each developer with a separate environment in which to work on new features or bug fixes. This pattern provides for long-term parallel development that is completely isolated from the mainline.

The Promotion (Repository) Pattern

The Promotion (Repository) Pattern is similar to the sandbox pattern, but it operates in reverse. Using the promotion pattern, all work is done in the mainline and only after a feature has been thoroughly tested and approved is the feature promoted to another branch. The promotion branch or **repository** branch is where all feature sets are included to create system releases. The mainline becomes the development sandbox for all developers on the team. The objective of the promotion pattern is to produce a code repository for all known good work. System releases are generated only from the repository branch. At any time a good system can be extracted and built from the contents of the repository branch.

This pattern depends on classic concurrent editing and at any time the mainline may be in a severe state of flux. Fortunately, developers don't often check unfinished code back into the mainline until it is done; thus the mainline should remain relatively clean, but that is a process issue. The following diagram illustrates the point:

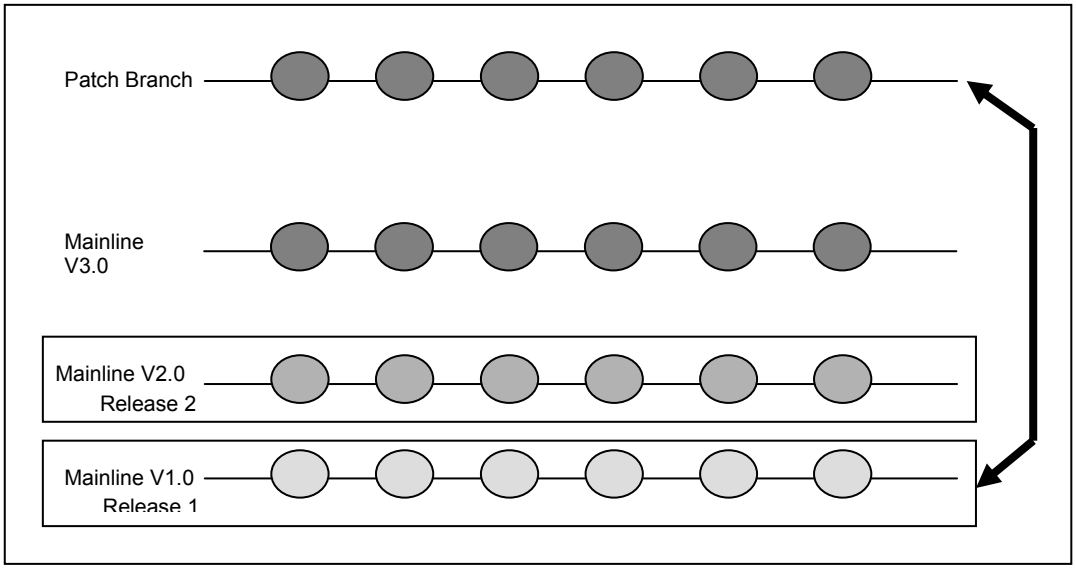


SpectrumSCM easily supports this pattern. After the repository branch is created, all files are checked out uncommon from the mainline. SpectrumSCM allows the project leaders to enforce this behavior by locking the repository generic. Locking guarantees that all files checked out or into the mainline will be uncommoned from the repository branch. Later, when the new features have been developed and tested, the new code is **merged** into the repository branch using the SpectrumSCM Merge Editor or by simply adding the new files to the repository generic. When features are rolled into the repository generic, the work is done by creating a new CR (Change Request) and that CR is used for adding or merging the files into the repository generic. Each CR in the repository generic represents a complete system feature or problem resolution. These CRs are easily included in new releases created from the repository branch. It is extremely easy to determine which feature sets and which bug fixes have been included in a release from the list of CRs associated with that release.

All work done by the development team occurs in the mainline branch. Only after features and bug fixes have been thoroughly tested and approved are they merged into the repository branch for creation of releases.

The Patch Pattern

The patch pattern is used to repair and re-release previously shipped releases. Typically this pattern is exercised when a customer calls to report a bug, in a particular release of the system. By using the patch pattern, the particular release that the customer is having problems with can be extracted and placed into a new working branch. The branch must be immediately locked so that common files are uncommoned during edit operations. The new branch allows developers to work on the exact file versions that were used to initially create the release. This allows the developers to faithfully reproduce the system as it was released to the customer and to debug and fix the problem. The problem files are extended directly from the released version numbers to add the fix. Once the fix(es) have been added to the patch branch, a new release can be generated, tested and released back to the customer and made available to other customers using that release. Consider the following diagram:



In this example, release 1.0 has been extracted into a new generic called the patch branch. The generic is locked and all of the files that are edited to resolve the problem are either already uncommoned or will become uncommoned as part of the edit operation.

The creation of the patch branch pattern allows developers to use the merge and recommon editors to apply bug fixes from subsequent releases into the newly created patch release. *SpectrumSCM easily supports this pattern and, again, may be one a few CM systems that supports this pattern correctly.* Any system can be used to dump out a previous release (one hopes) but very few systems actually allow the creation of a branch from a previously released system.

Sometimes previously released systems must be patched due to unexpected problems. End users of the previously released system may be reluctant to upgrade to the latest release due to testing issues and possible downtime. SpectrumSCM allows for any release to be recreated, and patch branches off that release to be easily created. Creating a new generic that is rooted in a previous release of the system does this. Files in the new branch are visible exactly as they were when the release was created. The calendar is essentially turned back to that time frame and the revision numbers for files in the patch generic are based on the revision numbers of the released product.

The patch pattern allows for a previously released version of a system to be easily extended and re-released without impact on other releases or current work. This pattern is not often needed, but

when it is, the ability to actually create a branch from a previous release results in a collective sigh of relief from the product developers and development managers.

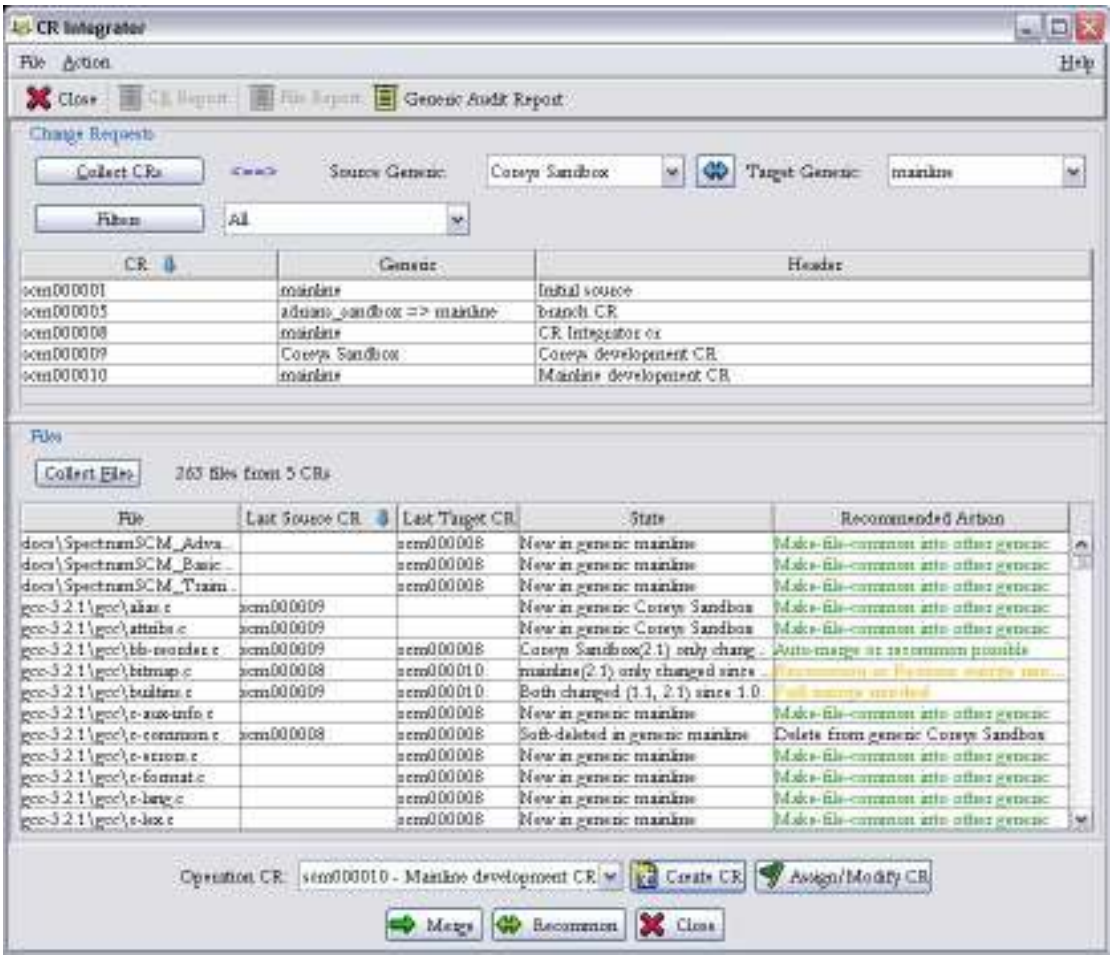
Conclusion

Several different branching techniques have been outlined above. The application of these patterns can result in systems that are easy to maintain, manage and extend. The application of the wrong pattern at the wrong time, or simply not applying any patterns to everyday development work, can lead to the development and evolution of systems that are confusing at best and extremely hard to manage at worst. Some shops avoid concurrent editing or any form of branching simply because their CM tools do not make branching and merging an easy process. Branching, merging and concurrent editing should not be difficult subjects that are only spoken about in soft whispers around the water cooler. If an organization's CM tools do not easily support these features, then it's probably time to think about some new tools. SpectrumSCM supports all of these patterns easily. Branching, merging, concurrent editing, and recommoning are all features of the Spectrum tool that are very easy to use and make difficult CM situations easier to manage.

11.5 CR Integrator

The CR Integrator is an interactive screen that can be used to compare, merge and recommon 2 generics or branches. The integrator is designed to help and automate (where possible) the process of merging or recommoning changes from one generic to another. This action frequently occurs as part of a number of branching patterns (e.g. the sandbox pattern). Under the sandbox pattern a developer (or team) perform their work in a sandbox generic, isolated from the main production line. Then when that work is complete they merge the work into the mainline ready for the next production release.

Some work-items can be automatically completed (seen in **green** in the screen-shot below) leading to huge productivity gains. For other items that require some scrutiny (seen in **yellow**), the user will be walked through the necessary steps. This in turn, not only gives you some productivity gains but also ensures the stability of your product.



11.5.1 Screen Flow

Select Generics - The first thing you want to do, is select your ***Source Generic*** and then your ***Target Generic***. Only generics that are related and can therefore be merged will be presented. The target generic is where the merge operation work is going to be performed.

For example: if you are merging your work from your sandbox back into the mainline, then the source generic will be your sandbox and the target will be the mainline.

Note: The  button can be used to swap the source and target generics.

Collect CRs - Once the source and target generics have been selected, the *Collect CRs* button should be pressed. This performs the generic audit to identify which CRs have caused files to be different between the 2 generics.

A double click on a CR entry will trigger the Change Request Report. This is also available off of the toolbar and Action menu.

Collect Files - Once the CRs list is populated, you can filter them if desired OR you can just present the file differences by pressing the *Collect Files* button. The file differences show which files that were edited by the selected CRs, are currently different between the source and target generics.

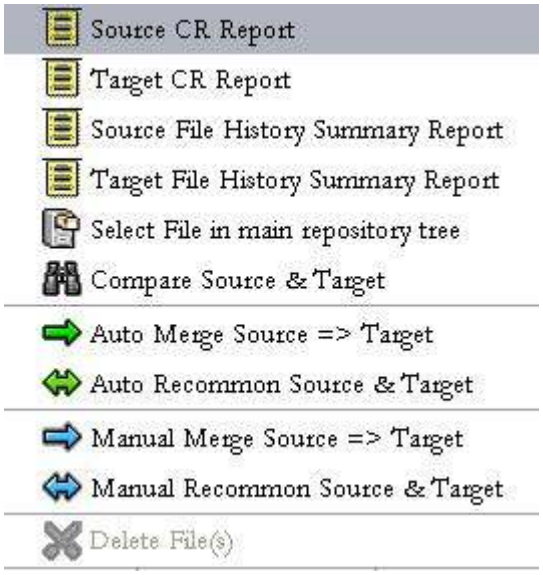
The Collected Files table presents the following information -

- The file.
- The last source CR - which CR was last used to edit this file in the **source** generic.
- The last target CR - which CR was last used to edit this file in the **target** generic.
- State - A summary line as to the state of the difference between the 2 generics for this file.
- Action - A recommendation as to which actions would generally make sense under this state.

At this point you can review the differences, or you can work the differences either individually or in bulk.

File lines shown in green are able to be automatically resolved.

File lines in yellow need manual assistance.



There are also a number of options available on the right-mouse button, these operations are performed relative to the selected lines. Double-clicking on a file line will run the **File History Generic Comparison** report, which shows how the file versions have progressed in the source and target generics, version by version.

11.5.2 Running an Auto-Merge

Select the CR for this operation - If you perform an edit operation, it needs to be performed relative to a change request that is *assigned to you*. If this is a merge operation the CR needs to be assigned under the target generic. If this is a recommon operation, since both generics are affected, the CR can come from either the source or the target generic.

The **Operation CR** choice box allows you to select this CR. Target generic CRs are shown in **black** and can therefore be used for either merge or recommon operations. Source generic CRs are shown in blue and can therefore only be used for recommoning. When a specific CR is chosen, then that will be used for all the operations until it is changed.

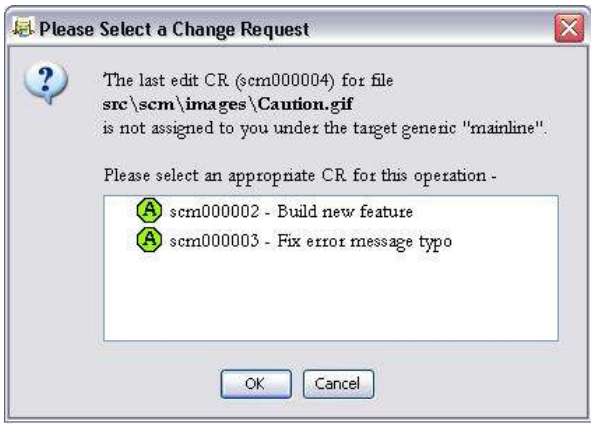
NOTE - Because merge operations have to be completed against a CR in the target generic, only **black** CRs will enable the merge button.

There are also a couple of special values ***Last Target CR*** and ***Last Source CR***.

If the *"Last Target CR"* option is chosen, the merge or recommon operation will be attempted against the CR shown in the *Last Target CR* column for each file. This is useful if you are merging the work of multiple CRs and you want the merge operation to be recorded against each of those CRs respectively.

Similarly, the *"Last Source CR"* will attempt to perform the requested operation against the CR shown in the *Last Source CR* column.

If there is no "Last ... CR", or it is no longer assigned to you, you will be presented with a warning and requested to select a valid CR.



If no file lines are selected and either of the merge or recommon buttons at the bottom of the screen are pressed then ALL lines will attempt to be processed. If one or more lines are selected then only those lines will be processed. You can select on a column header of the file table to sort by those values. So, for example, if you want to sort the actions into automatable and manual tasks you can do so.

Manual Merge and Manual Recommon operations are the same as if they were executed from the main screen file tree/menu structure.

Auto-Merge will update the target generic with the source file version contents but leave the files as separate instances. Auto-recommon literally makes the two generics point to the same file (with the source file contents). Recommoning enables future common edits.

If an automatic operation is selected upon a number of file lines, only those lines that are auto-capable will be automatically completed. The dialog will walk you through performing the manual tasks as appropriate.

When file lines are processed the **Recommended Action** box is set as such. If you want to reprocess a particular line, simply select it and select the required operation. A popup dialog will confirm the reprocessing action.

11.6 Generic/Branching Reports

There are 2 specific reports useful for managing multiple generic/branch situations. The "Generic Audit Report" compares the 2 selected generics and reports on the differences (uncommon or new files). The "File History Generic Comparison Report" reports on a particular files history with respect to its branching, commonality or uncommon situations.

The example below shows how the file was introduced common under CR 10. The file was then edited common under CR 11 before being edited uncommon into the "Branch" generic by CR 12. Work under CR 12 continued in the "Branch" until it was recommoned back into the mainline with version 2.2.

File History Generic Comparison Report

Project : scm
Source Generic : Branch
Compare Generic : Mainline
Filename : alias.c
Filepath : gcc-3.2.1\gcc

File type is: TEXT

Version History

Branch Version Number	Edit CR	Edit Date	Common	Mainline Version Number	Edit CR	Edit Date
2.2	scm000012	2008/01/08 16:18:41	Common	2.2	scm000012	2008/01/08 16:18:41
2.1	scm000012	2008/01/08 16:18:05				
2.0	scm000012	2008/01/08 16:17:54				
1.1	scm000011	2008/01/08 16:17:48	Common	1.1	scm000011	2008/01/08 16:17:48
1.0	scm000010	2008/01/04 12:01:19	Common	1.0	scm000010	2008/01/04 12:01:19

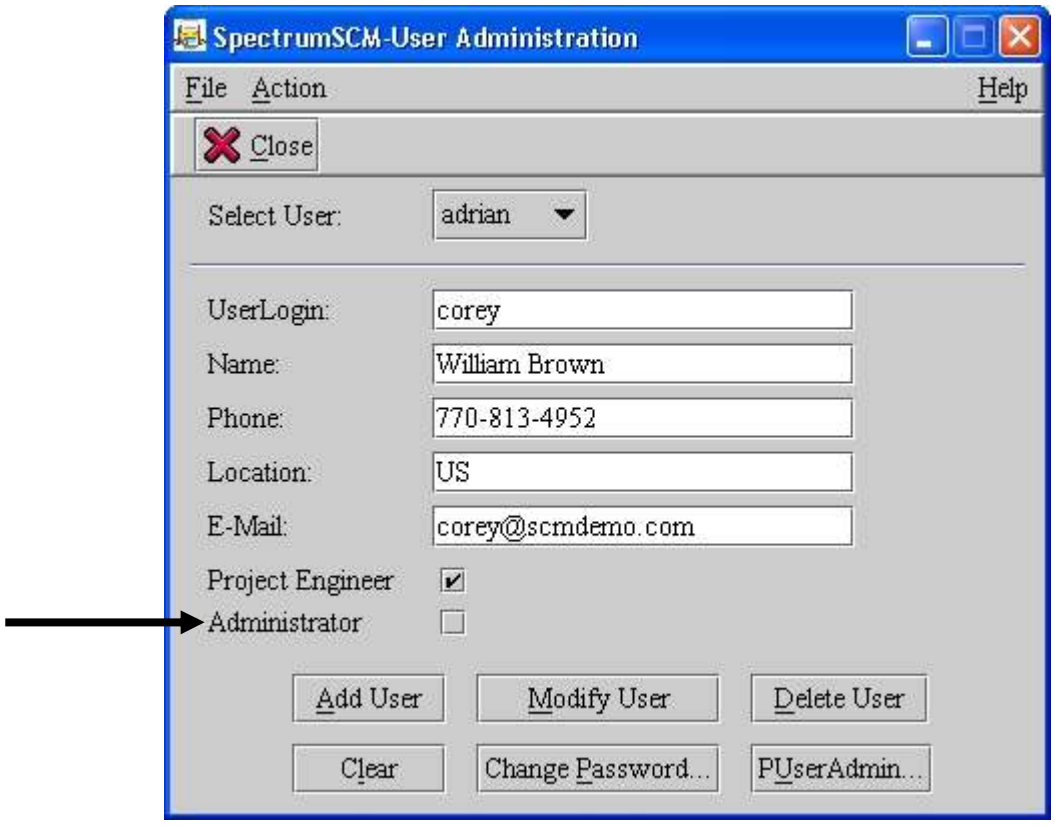
Chapter

12

12 SpectrumSCM Administrative Functions

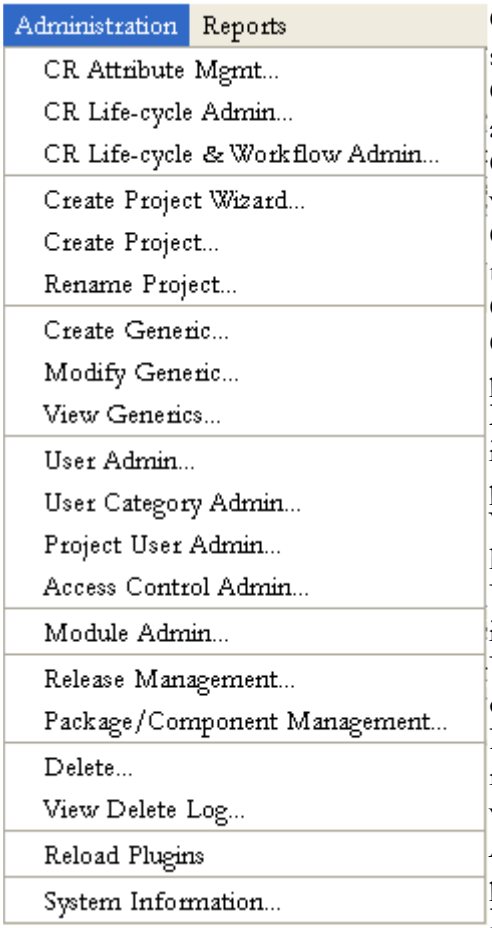
This chapter covers the SpectrumSCM system administration functions and some project-level administration functions accessible via the Administration menu option on the Main Screen.

The **Administrator** is the root/all-powerful user of the SCM tool. The "scm" administrator account is automatically added to the tool at build time. This may be removed at a later time but at a minimum you must always have at least one SCM administrator in the SpectrumSCM system. When a user is added to the SpectrumSCM system, he or she can be assigned Administrator authority level via the User Administration screen. *See Chapter 5, User Management, for details.*



12.1 SpectrumSCM Administration Menu

The Administration options are available via the Main Screen.



CR Attribute Mgmt - Manage the system-wide and project specific change request attributes.
CR Life-cycle Admin - Manage the life-cycle definitions and their assignment to projects.
CR Life-cycle & Workflow Admin – Manage the graphical workflow definitions and their assignment to projects.
Create Project Wizard – Add a new project using a wizard that runs through all the project creation steps
Create Project - Add a new project.
Create Generic - Add a new generic/branch to the current project.
Modify Generic - Modify a generic, specifically to manage its commonality lock i.e. whether developers are allowed to perform common edits.
View Generics – View the existing generics and their relative heirarchies.
User Admin - Maintain the system-wide user list. This includes names and contact information only.
User Category Admin - Maintain the set of user categories/roles.
Project User Admin. - Manage the Project-User relationship. I.e. manage which users are assigned to work on which projects and with what roles.
Access Control Admin – Define role based access permissions for generics, directories and files
Module Admin - Maintain your module definitions.
Release Management - Create and manage releases.
Package/Component Management – Create and manage packages and their components.
Delete - To delete an item (Project, Generic, Directory or File).
View Delete Log – View and possibly restore deleted items. Items that are marked as soft deleted can be retrieved. Hard deleted items cannot be restored.
Reload Plugins – Reload and restart user defined custom API plugins.
System Information – A "gas gauge" for the project repository. It shows you how much of the repository space is left. If System Repository Space is running low, contact Spectrum Software Support.

- CR Attribute Management is described in Chapter 7.
- CR Lifecycle Admin, Create Project, Create Project Wizard, Create Generic, and Modify Generic functions are used during project set-up and are described in Chapter 6, Process Management. Create Generic and Modify Generic are also used in Branching, which is described in Chapter 11, Branching, Merging and Re-common.
- User Admin, User Category Admin, Project-User Admin and Access Control Lists are used to control security and user access permissions. These are described in Chapter 5, User Management.
- Module Admin is a user level function that is used in managing source files and is described in Chapter 8.
- Release and Package Management functions are described in Chapter 9.
- Only Administrators and Project Engineers would generally have the ability to use the Delete function, however this is covered by the role based permissions and is configurable through the user category screen. File, Folder, Generic and Project deletion is described in Chapter 8 – Source File Management. CRs are deleted via the Change Request Assign/Modify screen described in Chapter 7.
- See Chapter 14 – API Concepts and Usage for more information on defining and using plugins.

12.2 Other Administrative Functions

The following functions can be executed from the command line or via the Windows START menu.

- **checkServer** - Check the status of the current SpectrumSCM server.
- **startServer –** Starts the SpectrumSCM server.
- **startServerAsService –** On Microsoft Windows platforms it might be desired to setup and run the SpectrumSCM server as a Windows service. Executing this script will install the Microsoft supplied ***AutoExNt*** service and set it to run automatically. If the server machine is restarted the SpectrumSCM server process will then restart automatically. See http://www.spectrumscm.com/FAQ.htm#_NTService for more details on this.
- **stopServer** - Stop the SpectrumSCM server. Note that this requires the user to input an administrator login and password for security reasons.
- **startUI –** Starts the SpectrumSCM graphical user interface client.
- **cloneActivation –** When the SpectrumSCM server is installed a control file is located in the users home directory and on Microsoft Windows platforms "Start" menu items will be inserted. If another user is requested to perform administration duties they would not have access to these control items. Simply executing the "cloneActivation" script (as that new user) will set up the appropriate control file and menu items.
- **uninstall –** used to uninstall all components of the server and UI. Do not use this when doing upgrades! It removes ALL components of SpectrumSCM, including all project databases.
- **uninstallUI –** removes the components of the SpectrumSCM user interface from a client machine.

To use the SpectrumSCM UI, the SpectrumSCM server must be up and the SpectrumSCM UI client on your machine must be started.

To check to see if the server is up:

In a WINDOWS environment
Start-> Programs -> SpectrumSCM UI ->CheckServer
In a UNIX or LINUX environment
change directory to the **<SpectrumSCM install>/bin** directory
execute the command **checkServer**

If the Server is running, you can start the SpectrumSCM UI
In a WINDOWS environment
Start-> Programs -> SpectrumSCM UI ->StartUI.
In a UNIX or LINUX environment
The UI can be started from the command line, nohup'ed, assuming you are running an X-server on the UNIX/Linux system. It can also be started from an xterm window.
change directory to the **<SpectrumSCM install>/bin** directory
execute the command **startUI**

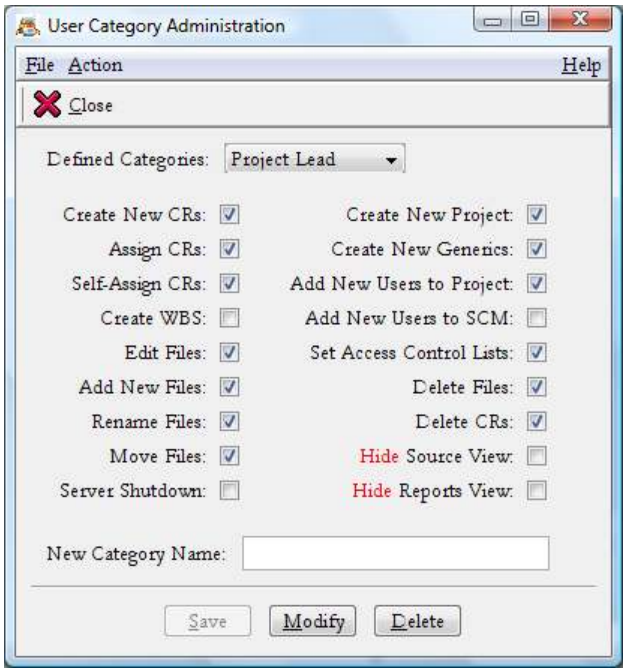
The UI is closed by exiting the graphical user interface (File / Exit) or by closing the SpectrumSCM UI window.

12.3 Who can execute Administrative functions

Access to the SpectrumSCM Administrative functions is limited and allowed by system and project level permissions.

As the default, when users are added to the SpectrumSCM system,

- Administrators have access to all functions.
- Project Engineers have access to most functions except User Admin. A project engineer at the system level is a project engineer across all projects in the system. Users can be granted Project Engineer permissions on a project-by-project basis through the Project-User Administration screen.
- Users would generally only have access only to Module Admin *(see Chapter 8, Module Admin for details)*



NOTE: These defaults can be overridden by project-level category permissions when categories are set up on the **Category Administration** Screen.

Any user assigned to a role that allows "Create New Project" permissions will have access to the system wide functions involved in setting up a new project. These include the attribute management and life-cycle management screens.

Users who have the permission to "Create New Generics" will also have access to the Generic Engineer aligned functions of Release Management and Package Management.

Permission to "Add New Users to SCM" allows access to the User Admin screen/function. Permission to "Add New users to Project" provides access to the Project User Admin screen/function. Permission to "Delete Files" provides access to the Admin menu -> Delete function.

NOTE: Pay attention to how the System-level and project-level permissions interact. Define and assign project-level roles carefully, giving users only the permissions they require to complete their work assignments.
See Chapter 5 for details on User Management, including system and project level permissions.

12.4 SpectrumSCM Security Features

SpectrumSCM offers a variety of security features to protect the assets it manages. These features fall into two groups, access control and communications security.

12.4.1 Access Control

SpectrumSCM employs a traditional account-based access model. The SpectrumSCM administrator role is responsible for creating user accounts. These accounts are protected by a login/password pair, which must be provided when a user logs into the application. This is the default application security model, an open mode that is generally acceptable for use on a corporate intranet. Additional Access Control can be configured through the server configuration wizard (or by editing the file *SERVER INSTALL DIRECTORY*/SCM_VAR/etc/security/accessControl).

12.4.2 Communications Security / SSL

By default, SpectrumSCM operates in an open mode, which is generally acceptable for use on a corporate intranet. This mode is the most efficient and appropriate for a majority of installations. If it is necessary to use SpectrumSCM across an uncontrolled, non-secure network (such as the Internet), SpectrumSCM provides a means of using **SSL** (**S**ecure **S**ocket **L**ayer) to protect its communications, assuring that source files may be checked out, modified, and checked in securely. SpectrumSCM can be configured to protect its communications using **SSL**, a security standard developed by Netscape and approved by the Internet Engineering Task Force as a standard).

The administrator must obtain an SSL key and configure SpectrumSCM. The configuration file must be edited to use SSL. This is done through the server configuration wizard or by directly editing *SERVER INSTALL DIRECTORY*/*SCM_VAR*/etc/scm.properties. The last section of the file pertains to SSL. The **ssl.inuse** property should be set "true" and the other SSL properties, particularly the SSL keystore (where the SSL key is stored) and the password that protects it must be provided.

Once the server has been properly configured for SSL, the SpectrumSCM clients may connect by using the SSL option. The SSL can be turned on via the UI Configuration Wizard or by supplying **-ssl** on the command line.

NOTE: Secured communications is not accomplished without a price - overall responses will be slower due to additional encryption/decryption processing at both ends of the connection.

12.4.3 Location Control

SpectrumSCM provides an additional level of security based on the user's workstation hostname (or IP address). A user can be restricted to logging into the application from specific hosts. This feature is called **Location Control**; to enable it a "doAccess" line must appear in the accessControl file followed by "access" lines specifying allowed user/workstation combinations:

EXAMPLE SCRIPT

```
doAccess
      access  user1   workstation1
      access  user2   workstation2
etc.
```

The accessControl file is located in the following directory: <SERVER INSTALL DIRECTORY>\SCM_VAR\etc\security

NOTE: With location control turned on, users can only log into the SpectrumSCM system from specified workstations. Even valid users attempting to login from workstations other than those specified in the accessControl file will be denied, *even if they supply a valid password!*

12.4.4 Unauthenticated Commandline Access (Single Sign-On)

SpectrumSCM provides a UNIX-style command line interface for accessing many of its features, however as mentioned above the default security scheme would require a login and password for each server access. This may prove to be unnecessarily tedious when typing in a sequence of commands or incompatible for some activities (automated checking out of files by a nightly build script). This situation also comes up in the general world with people accessing many different IT systems and applications and has become known as **Single Sign-On**.

SpectrumSCM provides a feature called **Unauthenticated Commandline Access** to *relax* security, to allow unauthenticated command line access by specific users at specific workstations. Administrators should exercise caution when configuring this feature, and use it sparingly, if at all, considering all security ramifications.

Basically what single **"sign-on/unauthenticated access"** does is move the access control responsibility to the operating system/work-station level. Once a user has successfully logged in to the operatingsystem/work-station, that login information is what is then used to access the individual applications such as SpectrumSCM.

To enable this feature a "doUnauth" line must appear in the accessControl file followed by "unauth" lines specifying allowed user/workstation combinations.

EXAMPLE SCRIPT

```
doUnauth
      unauth  user1   workstation1
      unauth  user2   workstation2
etc.
```

Users logging in to the UI or executing command line

functions from their corresponding workstations as configured in the accessControl file will not be required to provide a password since that verification has already been performed by the operating system.

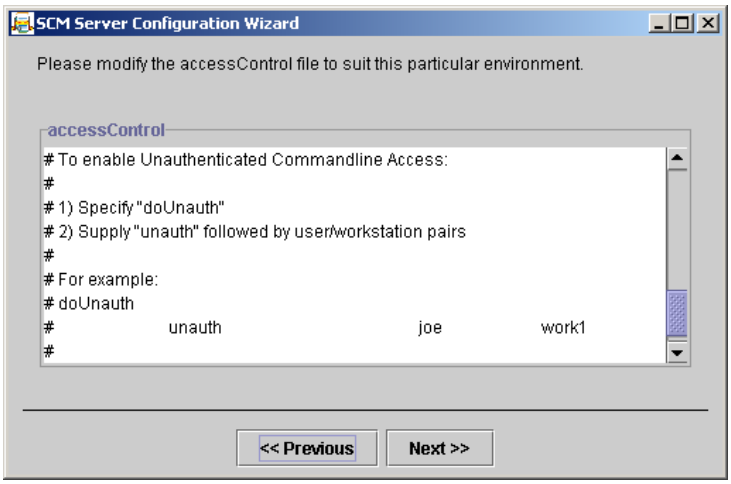
Administrators should exercise caution when configuring this feature, since if access to the workstation is not tightly controlled (ie access cards, screen locks etc) inappropriate accesses might occur.

The accessControl file (SERVER INSTALL DIRECTORY/ SCM_VAR/etc/security/accessControl) can be edited directly or via the Server Configuration Wizard.

To use the Command Line interface in a Windows environment, start the Command Prompt (cmd.exe) via Start / Programs/ Accessories / Command Prompt. In Unix or Linux, use the command line or an xterm window. Commands are executed from the <SCMUI install> bin directory:

Unix or Linux example: > cd /home/user/scm/bin
Windows example: > cd C:\home\user\scm\bin

See Chapter 13 for details on using Command Line Commands.



NOTE/CAUTION!: Whereas Location Control tightens security, Unauthenticated Commandline Access could be viewed as relaxing security. In addition, Location Control takes precedence over Unauthenticated Commandline Access.

12.4.5 Automatic Login Control

Setting up unauthenticated command line accounts also enables GUI automatic logins. Users that have an unauthenticated entry in the access control file will not need to enter a password into the initial GUI login screen. The system will automatically detect the user's login account from the operating system environment and attempt to use that information to automatically sign the user in the

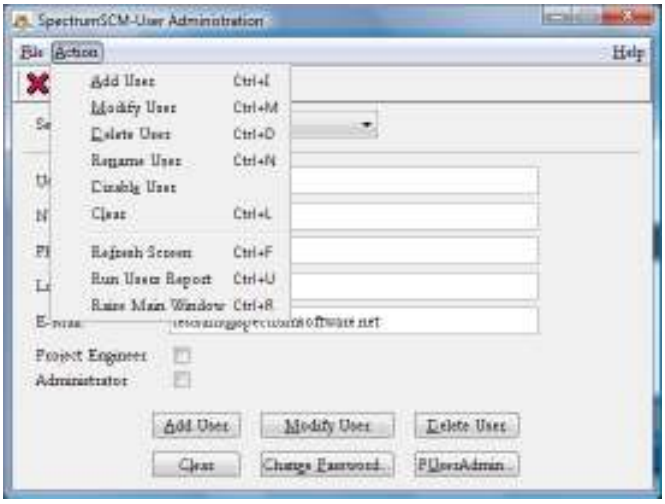
SpectrumSCM system. Failed automatic logins will present the user with the standard login/password screen for manual identification.

Once logged in, the user can use the **re-login** control, located in the File Menu Pulldown to log into the system as a different user.

12.5 Renaming Users

New

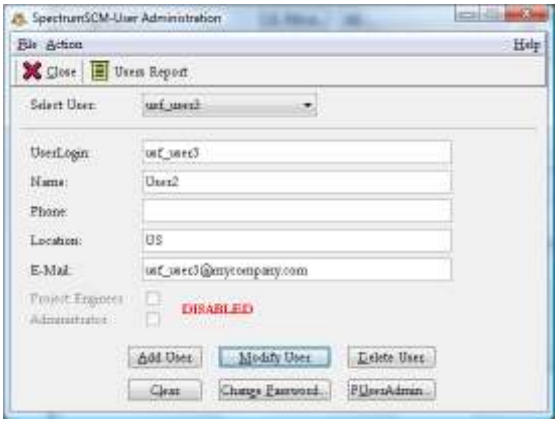
You can rename an existing user-id while maintaining full tracability of all the CRs that would transition from the old user-id to the new one.

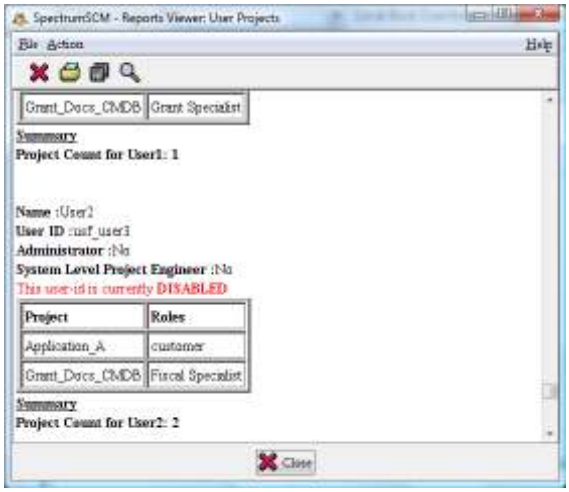


12.6 Disabling Users

New

You can disable a user so that their full identification is maintained but yet they would not be able to login to the SpectrumSCM system. You can enable a later date if you choose do so as well. Additionally you can click on the **"Users Report"** tool bar icon. This report displays the roles of all users across all projects. You can see the "DISABLED" status of "usf_user3" used in the example below.





12.7 Email Setup and Setting up email authentification

SpectrumSCM supports full e-mail notifications of Change Request creation and transitions. In fact by setting this up, **automatic email notifications happen** in real time for CR/Task creations, assignments, re-assignments, progressions, workflow transitions etc. You do not have to manually send an email when you create/assign/progress an incident or CR.

These emails are send to the person whom it is assigned and to all stake holders (based on the roles/workflow rules) instantaneously in real time. In addition the Task/CR shows up on the individuals CR list on the main SpectrumSCM screen when they log in as well.

The email notification feature can be configured at install time (you would have seen that screen during installation) or also at any time after installation as well.

STEPS:

So to set this up, stop your SpectrumSCM Server application. **(if it is up).**

The configuration information is available through the **Server Configuration Wizard.**

Since you have already installed the product, you can bring this screen up by going to **Start Programs->Server Configuration Wizard.**

Select the SCM_VAR entry that you see on the screen and hit the **<Edit>** button. This will bring up the **scm.properties** file.

This file holds all of the system configuration parameters

For using the email notification, in the scm.properties file under the MAIL section, uncomment the mail.smtphost line and set the **mail.smtp.host** parameter to your mailhost name or the appropriate smtp server name.

For using the **email authentification**,below the header which says "Authentication" uncomment the following as shown below. I have shown an example from one of my scm.properties file. You replace the appropriate values in the ones shown in red with your mail configuration details. Contact your Mail administrator to get these details.

```
#
# | Mail Mail Mail Mail Mail Mail Mail
# V
#
# Set this next line to the SMTP mail host for your
# organization. ex: smtp.mycompany.com
#
# mail.smtp.host mailhost
mail.smtp.host     smtpauth.earthlink.net
#
```

```
# Authentication ----------------------------------------
# If your smtp host requires authentication then
# you'll want to enable the following attributes:
#
mail.smtp.auth true
scm.smtp.auth.login johndoe@mindspring.com
#
# The SpectrumSCM server will use the login and password associated
# with the supplied login as the SMTP server login and password combo.
#
# Authentication ----------------------------------------
#
# The SCM mail "from" address. This is the address that all the
# mail will appear to have come from. Make sure to use a valid
# e-mail address here.
#
scm.from  johndoe@mycompany.com
```

****

After you make these entries in the properties file, restart the server.

Then you need to do one more step. You need to create a user_id and password in SpectrumSCM as well. So in the above example, the created user id is johndoe@mindspring.com and it had the password in the SpectrumSCM database its corresponding mail password. Ask your Mail administrator for the appropriate id and password.

Please refer to **Sec 3.4 of Chapter 3 SpectrumSCM Server and UI Configuration** in the User Guide (available at www.spectrumscm.com) for more details on the **scm.properties** file entries.

12.8 Backing up the SpectrumSCM data

Regular backups are recommended. The SpectrumSCM data (all the information about users, projects, everything else) is stored on the server in the directory specified at installation time, for example, **<SSCM_INSTALL_DIR>\SCM_VAR**. In addition, any project databases created in other directories would also need to be backed up.

Archiving these files regularly using any third-party backup mechanism is appropriate. If you do not remember where these directories are, you can pull up the *create project* menu item and select the project database directory's combo box, this will display all the directories that you are currently using.

It is recommended that the SCM server be properly shutdown and halted before the backup to guarantee that the data is stable. However, in general, running the backup at a quiet time (overnight) is perfectly adequate and does not require a server shutdown.

12.9 Setting Quick-start Tutorial under local environment

Included with the SpectrumSCM product is a quick-start tutorial. This is available via the SpectrumSCM Main Screen **HELP** menu option. As supplied, this will connect to the SpectrumSCM web site to get the most up-to-date information. If for some reason you prefer to have the tutorial information installed locally, it is provided on the installation CD, in the **tutorial** sub-directory. The tutorial is in HTML format for easy viewing with any standard browser.
Uncomment this line in the scm.properties file to allow Internet access to the tutorial.

scm.tutorial http://www.spectrumscm.com/Tutorial/scmstart.htm

To set up for a locally installed tutorial, replace the default entry for scm.tutorial in the scm.properties file. Use the Server Configuration Wizard or directly edit the scm.properties file and comment out the internet access and uncomment the local access, entering the path as described:

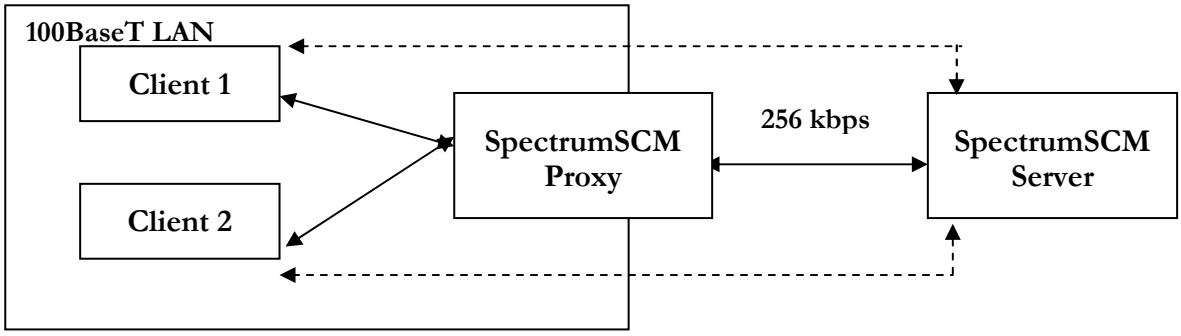
```
# The SCM Tutorial is accessed across the web by default.
# If access to the public network is not available, the
# tutorial path can be redirected to a local resource
# like in the following example for Windows:
#
# scm.tutorial  file:/C:/SCM_INSTALL_DIR/help/Tutorial/scmstart.htm
#
# or like this for Unix platforms:
#
# scm.tutorial  file:/SCM_INSTALL_DIR/help/Tutorial/scmstart.htm
#
# You do not have to use reverse slashes on the Windows platform.
# The tutorial materials are available on the SpectrumSCM installation
CD_ROM.
# Copy the entire Tutorial directory off of the CD_ROM to some location
# on a local or shared network drive, like in the examples above.
#
```

(See Chapter 3,SpectrumSCM Server and UI Configuration for more information on editing the scm.properties file.)

12.10 SpectrumSCM Proxy

The SpectrumSCM Proxy provides enhanced performance for distributed development teams in bandwidth constrained network topologies. The proxy acts as a bridge between the SpectrumSCM client and server and builds a local cache for frequently checked out revisions of e-Assets that are under source control. Files checked out by a client are used to serve similar requests from other clients, thus improving overall response time for check-outs and extracts. The proxy not only provides a better user experience for remote teams but also reduces the network traffic across the WAN and the load on the remote SpectrumSCM server. Load sharing can be achieved by using multiple proxies for different projects in SpectrumSCM. As opposed to multi-repository solutions for distributed development, the proxy uses a single repository model and thus removes the administrative overhead involved with maintenance and synchronization of multiple repositories. Also, the single repository architecture provides for a more reliable mechanism for version control activities while allowing the user to take full advantage of the integrated change management, process management and other advanced features in SpectrumSCM. The proxy updates its cache transparently as and when users check-out and check-in files and thus does not depend on the server to "push" updated information into its cache.

If the SpectrumSCM server is in a remote location and the available bandwidth between the local and remote ends is small, SpectrumSCM (like any other system) exhibits considerable lag for operations that require a file to be transferred across the wire. The proxy tries to address this issue by maintaining a "Local Cache" which reduces the number of file transfer operations across the WAN. Here is an example scenario:



Assume that the Clients C1 and C2 are on a 100BaseT LAN and the SpectrumSCM server is in a remote location, accessible through a 256 kbps WAN pipe (W). In the absence of the Proxy, the clients directly interact with the server and all file transfer operations happen over W. This can be slow and inefficient considering the fact that the bandwidth is small. In such situations, whenever the client C1 downloads a file from the server, other clients who need the same file can make use of the file downloaded by C1. Assuming that the file contents have not changed, the clients can bypass the server and get the file from within the LAN, which makes the operation extremely fast. Retrieving the file from within the LAN also reduces the network traffic across the WAN and the load on the remote SpectrumSCM server. Thus files downloaded from the server can be used to serve similar requests from different clients, provided that the file contents have not changed. This is the function of the SpectrumSCM Proxy.

In a proxy based configuration, whenever a client needs a file from the server, it routes the request through the SpectrumSCM Proxy. The proxy checks its cache to see if it has the file that is being requested. If it determines that the file being requested and the cached copy are the same, it serves the request without downloading the file from the server. Thus the file is retrieved locally. If it is a miss, it downloads the file from the server, passes it on to the client and caches it for future use. Over a period of time, the number of hits will surpass the number of misses making checkout operations considerably faster. The SpectrumSCM Proxy also receives checkin triggers from the clients and updates its cache when users check-in files. The proxy not only caches the head revision, but also stores previous versions of a file and thus provides fast response times during release extracts and extraction of files by version/CRs.

Installation

You need to install the SpectrumSCM client or server before you can install the proxy. Instructions for installing the server/client are available on our website at **www.spectrumscm.com.** To install the SpectrumSCM Proxy, download the **ProxyInstaller.jar** file to a machine with a SpectrumSCM client/server installation. On Windows systems, you can double-click the jar file to launch the installation wizard. On UNIX systems, open a terminal window and run :

java -jar ProxyInstaller.jar

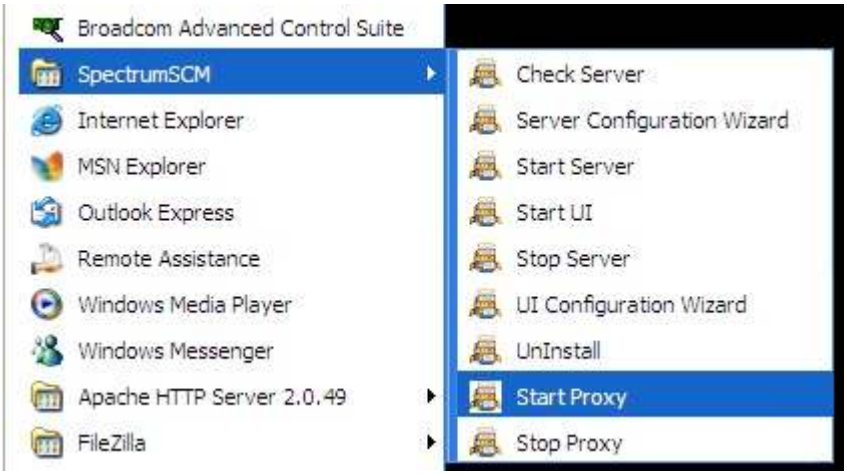
Follow the on-screen instructions to complete the installation. The proxy scripts and libraries will be installed under the SpectrumSCM install directory

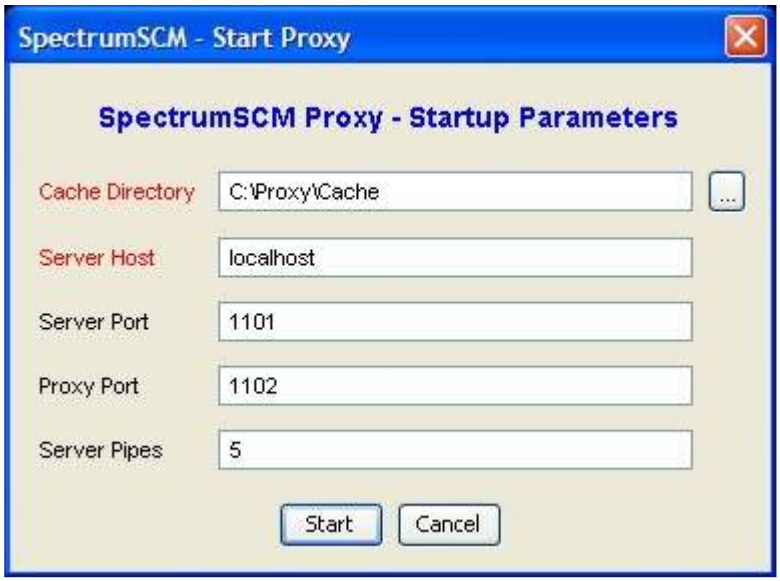
Starting the SpectrumSCM Proxy

The SpectrumSCM Proxy can be started using the GUI or command line modes. Users can specify the directory to use for the proxy cache. The proxy log files are maintained under <specified cache dir>/logs directory

Windows

Go to the SpectrumSCM menu under the Windows Start menu and choose the Start Proxy menu item or open a command prompt window and go to the <SpectrumSCM Install>/bin directory. Use the startProxy.bat script.





Unix/Linux/Solaris

Open a terminal window and go to the <SpectrumSCM Install>/bin directory and execute the startProxy script

The Proxy startup program uses the following arguments:

-cache	Cache Directory for the Proxy	REQUIRED PARAMETER
-port	Listen Port for the Proxy	Defaults to 1102
-serverhost	SpectrumSCM Server IP	Defaults to localhost
-serverport	SpectrumSCM Server Transport Port	Defaults to 1101
-serverpipes	Max no. of connections b/w proxy & server	Defaults to 5
-gui	Start in GUI mode (overrides other options)	

Stopping the SpectrumSCM Proxy

Windows

Go to the SpectrumSCM menu under the Start menu and choose the Start Proxy menu item or open a command prompt window and go to the <SpectrumSCM Install>/bin directory. Use the stopProxy.bat script

Unix/Linux/Solaris

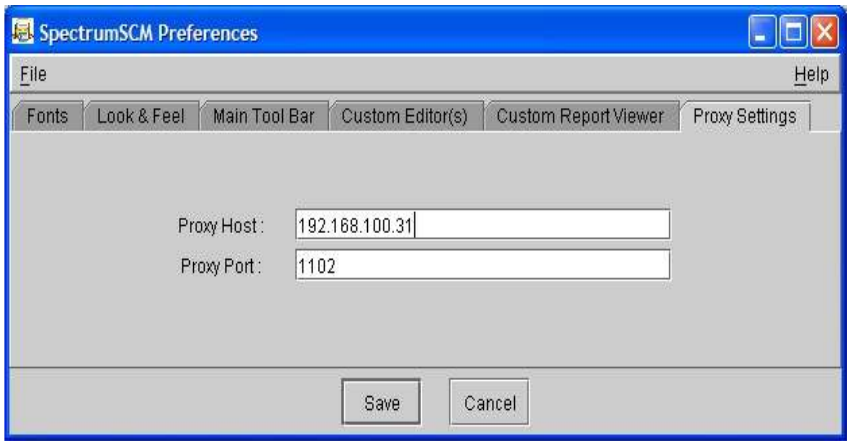
Open a terminal window and go to the <SpectrumSCM Install>/bin directory and execute the stopProxy script

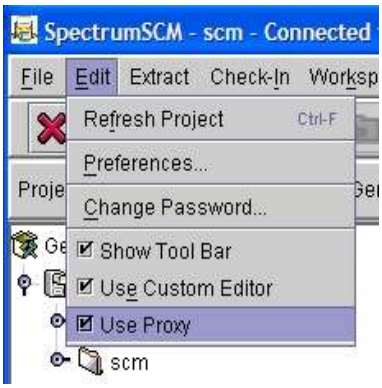
The proxy shutdown program uses the following arguments:

-host	Proxy IP Address	Defaults to localhost
-port	Listen Port for the Proxy	Defaults to 1102
-gui	Start in GUI mode (overrides other options)	

How to Use the SpectrumSCM Proxy

Start the SpectrumSCM client and choose the Edit-->Preferences menu item. Choose the Proxy Settings panel and specify the Proxy IP and Port Number. Save your settings and check the "Use Proxy" option under the Edit menu to route checkout requests through the proxy. Uncheck the option to communicate directly with the SpectrumSCM server.





Chapter 13

13 Command Line Interface

SpectrumSCM supports a Command Line Interface so that tasks can be performed over ASCII terminals and reports and builds can be performed using scripts. Command Line commands can be run from the Unix or Linux command line, an xterm window, the Windows Command Prompt Screen, or used within a script.

To use to the Command Line interface in a Windows environment, start the Command Prompt (cmd.exe) via Start / Programs/ Accessories / Command Prompt. In UNIX or Linux, use the command line or an xterm window.
Commands are executed from the <SCMUi install> bin directory:

Unix or Linux example:	> cd /home/user/scm/bin
Windows example:	> cd C:\home\user\scm\bin

To access the Command Line interface there are 3 security options, these basically cover how the user is verified and allowed access to the repository server and information. The command-line infrastructure defaults to use the users current operating system user-id, the "-login" option can be specified to change this. The password can then be provided via the "-password" option, supplied from the terminal using the "-prompt" option or thirdly via accessControl. AccessControl basically states that user X on workstation Y is a pre-verified (or trusted) user and therefore needs no further password (this is also sometimes called single sign-on, where a user has to logon to their workstation but then that information is passed to the applications as their sign-on information). AccessControl can be enabled through the Server Configuration Wizard. (*See details in Chapter 12, Unauthenticated Commandline Access).*

All the command line functions display detailed usage statements to the standard output when the command is specified without parameters.

Command	Usage
scm_addnote	Add a note to the supplied CR
scm_bulkassign	Bulk assign a set of CRs from one situation to another
scm_co	Check out a file for read.
scm_cow	Check out a file for edit.
scm_check	Check what we currently have out for edit.
scm_ci	Check in a file.
scm_cidir	Check in a set of files.
scm_diff	Initiate the standalone UI file or directory differencer
scm_getUserProjectInfo	Extract an XML statement of the specified users project view.
scm_gv	Extract the current version of the product or directory.
scm_mkdirs	Create the relative workspace directories.
scm_gcr	Extract only the files touched under the specified change request
scm_gr	Extract the specified release of the product.

scm_gir	Extract the intermediate release from the specified life-cycle phase
scm_gpack	Extract the specified package and its components.
scm_mngeMeta	Manage a files meta-information/notes.
scm_newcr	Create a new Change Request.
scm_newXMLcr	Create a new Change Request using an XML template
scm_progress	Progress the state of a worked Change Request.
scm_rpt	Reports interface.
scm_unlock	Unlock a file from edit.
scm_updateXMLcr	Update the specified CR via an XML template

13.1 Parameters

All command line arguments require some set of these parameters. Some arguments are mandatory; others are optional.

NOTE: If an argument value contains embedded spaces, the value must be quoted.

13.2 Commonly used parameters that require an argument.

If a command uses one or more among these parameters, an argument is required.

Parameter	Argument and Usage
-root <root directory>	This is the current local root directory for the project. i.e. the root of the disk location where operations should be performed.
-project <project Name>	The project name
-generic <generic >	The generic name
-filename <filename>	The file name
-release <release name>	The release name
-cr <cr>	The CR associated with an action. For example, if the user wishes to edit the file, the -edit argument and the associated CR should be specified
-host <server>	Specify the server, if you are not on the system containing the SpectrumSCM server
-port <port number>	If your server is not using the default (1099) port, specify the port number here.
-proxyhost <host>	If the proxy feature is being used, where is the proxy located
-proxyport <port>	If the proxy feature is being used, which port is it using on the host.
-login <login id>	your login id.
-password <password>	This option can be used to directly provide your server login password.

13.3 Parameters without Arguments

If any of these parameters are specified for a command, the parameters do not require arguments. Using the parameter with a command will result in the described action. .

Parameter	Action
-dirsOnly	Used with commands scm_gv and scm_mkdirs, this will either extract or create the directory structure for the project.
-binary | -text | -ascii	Used with the command SCM_CI to specify the type of file being check in. ***NOTE:*** -text and -ascii are synonymous. If no argument is specified for check in, system will determine file type automatically. If there are any characters outside of the normal ASCII range of printable characters, the file type is assumed to be Binary
-includeBinaries	Use to indicate that binaries are to be included
-overwrite	Use if the file is to be overwritten
-recurse	This argument will walk the entire directory tree from the point specified and perform the operation requested.
-ssl	Enable secure socket layer interface ***NOTE:*** The –ssl option cannot be used unless the secure socket layer (ssl), a security option, is enabled on the server. Security options must be enabled by an authorized administrator.
-edit	Used only in scm_co to flag a file to be cheeked out for edit.
-prompt	Requests that the server prompt for the users password.

13.4 Environmental Variables

Two environment variables are supported to ease use, allowing presets for Java Virtual Machine memory and common arguments such as server connectivity and project information. Setting environment variables at the beginning of a command line session or in a script will eliminate the need to set the parameters in each of the subsequent command line commands. This increases productivity and reduces the chance for typing errors.

SCM_CMDL_ARGS

Use SCM_CMDL_ARGS to reduce typing with respect to common command line options.

For example, setting SCM_CMDL_ARGS to "-project Genesis -generic gen1.0" will automatically supply those arguments to each command line function invocation without repeated typing.

Using the SCM_CMDL_ARGS option, this example sets the project name to Genesis and the generic name to gen1.0 for all subsequent commands and thus they need not be repeated. This increases productivity and reduces the chance for typing errors. The parameters are set once for the entire script or session.

```
$ export SCM_CMDL_ARGS="-project Genesis -generic gen1.0 –root /home/user/gen10"
. . .
$ scm_co  -filename Test.java
$ scm_cow -filename Test.java -cr TestPrj000005
$ scm_ci –filename Test.java –cr TestPrj000005
$ scm_gv
$ scm_gr –release baseline
```

Depending on your security settings, you could also specify your login and password (or prompt) options here too.

SCM_JAVA_ARGS

SCM_JAVA_ARGS are used to fine-tune the Java Virtual Machine.
Setting SCM_JAVA_ARGS to "-Xmx128m" can be used to control the server memory allowance available to the Java Virtual Machine (setting it to 128MB in this case). The default heap size for the VM is 64MB.

Using the SCM_JAVA_ARGS option, an example is:

```
$ export SCM_JAVA_ARGS=”-Xmx128m”
. . .
$ scm_gv -root /home/user/scm -project Genesis -generic gen1.0
```

Setting $ export SCM_JAVA_ARGS=”-Xmx128m” could be used when the user needs more memory when executing an extremely large report or extracting a very large directory structure.
Error messages such as "out of memory" or "stack overflow" indicate the need for more memory.

13.5 Commands

Command line commands can be run from the command line (or Windows Command Prompt Screen) or used within a script. **For help in the command line interface,** type in the command. All of the command line functions display detailed usage statements to the standard output if only the command name is input.

Command	Arguments
scm_co Check out a file	-root <root> -project <project> -generic <generic> (-filename <filename> | -filelist <filelist file> | -bomfile <BOM file>) [-edit] if the file is being checked out for edit [-cr <CR number>] [-overwrite] if the file needs to be overwritten [-common] [-uncommon] [-version <version>] [-host <SCM host>] [-port <SCM port>] [-proxyhost <proxy host>] [-proxyport <proxy port>] [-login <login>] [-password <password> | -prompt] [-ssl]

Example: To check out a file Test.java from the source directory src/java under a user's SCM directory /home/user/scm; the current directory should be /home/user/scm/src/java where the file Test.java resides.
The user should execute either:

$ scm_co -root /home/user/scm -filename Test.java -project Genesis -generic gen1.0

(to extract a read-only copy of the head revision of Test.java)

$ scm_co -root /home/user/scm -filename Test.java -project Genesis -generic gen1.0 -edit -cr TestPrj000005

(to extract for edit the Test.java file)

NOTE: If the user wishes to edit a file, either the **scm_cow** command or the scm_co command can be used with the -edit argument and the appropriate CR specified.

The ***filename*** option is the most frequently used. However, if you wish to check-out a significant number of files, place that list of files (one per line) into a temporary file. Then use the ***filelist*** option and supply that temporary file.

The ***bomfile*** option allows you to specify a Bill Of Materials report generated from a release extract (GUI or command-line, intermediate or full release). The scm_co operation will then fully reproduce that initial extract by pulling the specific file versions from the repository. Use the HTML BOM report here as opposed to the textual file.

Command	Arguments
scm_cow Check out a file for edit.	-root <root> -project <project> -generic <generic> -filename <filename> -cr <cr> [-overwrite] [-common] [-uncommon] [-host <SCM host>] [-port <SCM port>] [-login <login>] [-ssl]

Example: To check out a file Test.java from the source directory src/java under a user's SCM directory /home/user/scm;

Change directory to /home/user/scm/src/java where the file Test.java resides.
Execute:
$ **scm_cow -root /home/user/scm -filename Test.java -project Genesis -generic gen1.0**

Command	Arguments
scm_check Check what a user currently has out for edit. Default user is the user currently logged into SCM. Use –user argument to see what another user has checked out.	-project <project> [-user <userid>] [-host <host>] [-port <port>] [-ssl]

Example: To view a list of files that a user has checked out for the project Genesis, this command should be used.

$ **scm_check -project Genesis**

```
                                                        2002/11/21 08:34:39
                            User Edited Files
                           ------------------
--------------------------
| Project :  | Genesis   |
| User :     | rich      |
--------------------------
-------------------------------------------------------------------------------------------
|File Name    |Change Request| Generic | Version No | Edited on           | Creator     |
|                                                                                           |
|install.java | TestPrj000005| gen1.0  | 1.39       | 2002/05/19 08:56:19 | Gene        |
|                                                                                           |
-------------------------------------------------------------------------------------------
End Of Report
```

Command	Arguments
scm_ci Check in a file.	-root <root> -project <project> -generic <generic> (-filename <filename> | -filelist <filelist file>) -cr <cr> [-binary | -text | -ascii] ***Note*** *-text and -ascii are synonymous, if none are specified Checkin will determine file type automatically.* [-host <SCM host>] [-port <SCM port>] [-login <login>] [-ssl]

Example: To check in a file Test.java to the source directory src/java under a user's SCM directory /home/user/scm;

Change directory to /home/user/scm/src/java where the file Test.java resides.

Execute:

$ scm_ci –root /home/user/scm -filename Test.java -cr TestPrj000005 -project Genesis -generic gen1.0

Command	Arguments
scm_cidir Check in a set of files contained under the current working directory.	-root <root> -project <project> -generic <generic> -cr <cr> [-host <SCM host>] [-port <SCM port>] [-login <login>] [-recurse] [-includeBinaries] [-ssl]

Example: To check in a directory to the source directory src/java under a user's SCM directory /home/user/scm;

Change directory to /home/user/scm/src/java where the target files reside.

Execute:

$ scm_cidir -root /home/user/scm -cr TestPrj000005 -project Genesis -generic gen1.0

Command	Arguments
scm_gv Extract the current version of the product (get version)	-root <root> -project <project> -generic <generic> [-recurse] [-dirsOnly] [-host <SCM host>] [-port <SCM port>] [-login <login>] [-ssl]

Example: To obtain the contents of the src/java directory under the user's current SCM directory /home/user/scm, the user must:

Change directory to the /home/user/scm/src/java directory.

Execute:

$ **scm_gv –root /home/user/scm -project Genesis -generic gen1.0**

Command	Arguments
scm_mkdirs Create the relative workspace directories (Note – this is really just a wrapper for scm_gv –dirsOnly).	-root <root> -project <project> -generic <generic> [-recurse] [-host <SCM host>] [-port <SCM port>] [-login <login>] [-ssl]

NOTE: If the user wishes to obtain subdirectories, the -recurse argument should be appended.

Example: To obtain the contents of the src/java directory under the user's current SCM directory /home/user/scm, the user must:

Change directory to the /home/user/scm/src/java directory.

Execute:

$ **scm_mkdirs –root /home/user/scm -project Genesis -generic gen1.0**

Command	Arguments
scm_gr Extract the specified release of the product.	-root <root> -project <project> -generic <generic> -release <release> [-host <SCM host>] [-port <SCM port>] [-login <login>] [-ssl]

Example: To extract an entire release, to perform a release build for example, you can run this from the command line or from inside an automated build script.

$ **scm_gr –root /home/user/scm -project Genesis -generic gen1.0 -release scm2.6**

Command	Arguments
scm_gir Extract the specified interim release of the product based on all the CRs at or past the specified phase.	-root <root> -project <project> -generic <generic> -phase <phase> [-excludeBase] [-includePhase <comma-list>] [-excludePhase <comma-list>] [-nodeps] [-deltas] [-host <SCM host>] [-port <SCM port>] [-login <login>] [-ssl]

Example: To extract an intermediate release build for all issues ready for integration testing, you can run this from the command line or from inside an automated build script.

$ **scm_gir –root /home/user/scm -project Genesis -generic gen1.0 –phase "Integration Test"**

The ***nodeps*** option will turn OFF dependancy checking, this means that ALL CRs in the specified phase and later will be extracted. With dependancy checking ON (the default and safest option), only CRs that do not depend on other incomplete CRs will be included.

The ***deltas*** option will only extract files that have changed since the last *gir* command. This is controlled by the Bill Of Materials file that is produced into the root directory with each *gir* execution.

The ***excludeBase*** option will only extract the file versions relative to the non-released CRs i.e. only extract those files that are still being worked.

The ***includePhase*** and ***excludePhase*** lists are not normally needed as the interim release mechanism will automatically pick up the appropriate life-cycle phases based on the linear project life-cycle. However, in the case of an extensive graphical (non-linear) workflow specific phases might want to be included or excluded if the linear order is not quite correct for the desired extract.

Command	Arguments
scm_gcr Extract the set of source file versions that were edited by this CR.	-root <root> -project <project> -generic <generic> -cr <cr> [-before] [-createBOM | -bomOnly] [-host <SCM host>] [-port <SCM port>] [-login <login>] [-ssl]

Example: To extract just the files touched by CR TestPrj000004.

$ scm_gcr –root /home/user/scm -project Genesis -generic gen1.0 -cr TestPrj000004

The ***createBOM*** option will create a Bill of Materials report in addition to performing the file version extracts. The ***bomOnly*** option will only create the Bill of Materials report and will not extract the file versions.

The ***before*** option will extract (or report) what file versions existed in the repository BEFORE this CR. I.E. In a simple linear case where this CR edited version 1.3 of file X, then the –before option will extract version 1.2 of file X. This option is useful in conjunction with code review type procedures or tools to state exactly what the before and after situations were.

Command	Arguments
scm_unlock Unlock a file from edit.	-root <root> [-host <SCM host>] [-port <SCM port>] -project <project> -generic <generic> -filename <filename> [-ssl]

Example: To unlock a file that was checked out and locked, execute:

**$ scm_unlock -root /home/user/scm -filename Test.java -cr TestPrj000005 -project Genesis **
-generic gen1.0

Command	Arguments
scm_newcr Create a new Change Request.	[-project <project>] [-generic <generic>] [-header <header>] [-description<description>] [-host <SCM host>] [-port <SCM port>] [-login <login>] [-ssl]

Example: To create a new CR,

```
$ scm_newcr
Project: scm_utils
CR creation Phases:
      (1) Study
      (2) Develop
      (3) Test
Please select a CR creation phase: 1
Please select one value from each presented attribute.
"Severity"  values:
      (1) High
      (2) Medium
      (3) Low
Please select a value: 2
Attribute value selection completed.
Please provide a brief description of this CR:
Test of command line cr create
Please provide a detailed description of this CR; terminate the description
with a single period '.' on a line:
Test of command line cr create
.
Would you like to have this CR assigned to you? (y/[n]): y
Generics:
      (1) base
Please select a generic: 1
Assignment Phases:
      (1) Study
      (2) Develop
      (3) Test
Please select a phase: 1
You have provided the following information:
1) Project:          scm_utils
2) Create State:      Study
3) Attributes:
      Severity       Medium
4) Header:           Test of command line cr create
5) Description:
Test of command line cr create
6) Assigned User:      rich
  Assigned Phase:     Study
  Generic:          base
Are these correct? If so, enter <y>, otherwise enter the number of the incorrect field: y

CR scm_utils000014 successfully created.
$
```

Command	Arguments
scm_newXMLcr Create a new Change Request using an XML template.	[-project <project>] [-generic <generic>] [-host <SCM host>] [-port <SCM port>] [-login <login>] [-file <template>]

Example: To create a new CR,

$ scm_newXMLcr –project Genesis – generic Mainline –file c:\Work\template.xml –login scm

A sample XML template is shown below:

```
<CRS>
      <NEWCR>
            <HEADER     VALUE="This is the header"/>
            <DESCRIPTION>
                  <DESCITEM VALUE="Now is the time for"/>
                  <DESCITEM VALUE="all good programmers"/>
                  <DESCITEM VALUE="to come to the aid of"/>
                  <DESCITEM VALUE="their editors"/>
            </DESCRIPTION>
            <CREATOR VALUE="scm"/>
            <ATTRIBUTES>
                  <ATTRIBUTE_ENTRY NAME="Severity" VALUE="High"/>
                  <ATTRIBUTE_ENTRY NAME="Location" VALUE="Atlanta"/>
            </ATTRIBUTES>
            <CREATION_PHASE VALUE="Study"/>
            <ASSIGNED_USER>
                  <USER VALUE="scm"/>
                  <PHASE VALUE="Study"/>
                  <GENERIC VALUE="Mainline"/>
            </ASSIGNED_USER>
            <HISTORY>
                  <USER VALUE="scm"/>
                  <HISTORY_ENTRY VALUE="some info"/>
                  <HISTORY_ENTRY VALUE="some info"/>
            </HISTORY>
      </NEWCR>
</CRS>
```

Command	Arguments
scm_addnote Add a note to a Change Request.	[-project <project>] [-cr <Change Request No>] [-host <SCM host>] [-port <SCM port>] [-login <login>] [-ssl]

NOTE: This command will prompt for all necessary arguments.

Example: The entered information is in bold.:

```
$ scm_addnote –project SD_Demo – cr SDCR000031 –login scm

Please provide the note to be added to this change request.
Terminate the note with a single period '.' on a line
This is my note
.

You have provided the following information:
1) Project:              SD_Demo
2) Change Request:  SDCR000031
3) Note:
This is my note

Are these correct? If so, enter <y>, otherwise enter the number of the
incorrect field: y

CR SDCR000031 successfully annotated.
$
```

Command	Arguments
scm_mngeMeta Get, Set or Append meta-information to the specified file.	-project <project> -generic <generic> -root <root> -filename <filename> (-get | -set | -append) -meta "meta information" [-host <SCM host>] [-port <SCM port>] [-login <login>] [-ssl]

For example, to add meta-information to file Test.java from the source directory src/java under a user's SCM directory /home/user/scm; the current directory should be /home/user/scm/src/java where the file Test.java resides.

The user should execute:

```
$ scm_mngeMeta -root /home/user/scm -filename Test.java -project <project> \
    -generic <generic> -set -meta "new meta information"
```

Command	Arguments
scm_bulkassign Bulk assign a set of CRs based on specified criteria	[-project <project>] [-generic <generic>] [-cp <Current Phase>] [-ca <Current Assignee>] [-cr <Current Release>] [-np <New Phase>] [-na <New Assignee>] [-note <Annotation>] [-host <SCM host>] [-port <SCM port>] [-login <login>] [-ssl]

NOTE: This command will prompt for all necessary arguments.

Example: The entered information is in bold.:

$ **scm_bulkassign -project SD_Demo -generic Mainline -cp Test -ca ken -cr xyz123 -np Release -na eric -login scm**

Please provide the note to be added to these change request(s).
Terminate the note with a single period '.' on a line
This is my note
.

You have provided the following information:
1) Project: SD_Demo
2) Generic: Mainline
3) Current Phase: Test
4) Current Assignee: ken
5) Current Release: xyz123
6) New Phase: Release
7) New Assignee: eric
8) Note:
This is my note

Are these correct? If so, enter <y>, otherwise enter the number of the incorrect field: **y**

Found CR SDCR000011
Found CR SDCR000017
Found CR SDCR000005
Found CR SDCR000007
Confirm re-assignment of 4 CRs (y/n) > **y**

CR SDCR000011 successfully assigned.
CR SDCR000017 successfully assigned.
CR SDCR000005 successfully assigned.
CR SDCR000007 successfully assigned.
$

Command	Arguments
scm_progress Progress the state of a worked Change Request.	[-project <project>] [-cr <Change Request No>] [-host <SCM host>] [-port <SCM port>] [-login <login>] [-ssl]

NOTE: This command will prompt for all necessary arguments.
Example: The entered information is in bold.:

```
$ scm_progress

Please enter the Project: scm_utils
Please enter the Change Request no: scm_utils000002
Please provide some description of the changes performed.
Terminate the description with a single period '.' on a line
test progress
.

You have provided the following information:
1) Project:          scm_utils
2) Change Request:   scm_utils000002
3) Description:      test progress

Are these correct? If so, enter <y>, otherwise enter the number of the
incorrect field:  y

CR scm_utils000002 successfully progressed.
$
```

Command	Arguments
scm_rpt Reports interface	<-xmlinputfile <filename> || -project <project name> [- template] [-host <host>] [-port <port>] [-login <login>] [-width <width>] [-ssl]

The command line report functionality operates in one of two modes, interactive or XML driven. In the interactive mode the user is prompted by the report mechanism to choose a particular report to run, and is then prompted for each of the arguments for the chosen report. Alternatively, the user can execute the command line report mechanism in an XML driven mode. In XML mode, the user supplies an XML template which describes the report to execute and all of the necessary input parameters.

To drive the system in interactive mode, the user should supply the "-project" option along with the necessary "-host" and "-login" options like the following example:

$ scm_rpt –project MYPROJECT –host MYSERVER –login MYLOGIN

The report functionality will respond with a list of all available reports, including custom reports and the user will be prompted to choose one of the reports. Once a report has been chosen, the user will be prompted for the specific input parameters for the chosen report.

Note that specific options can also be specified on the commandline as parameters. For example:
$ **scm_rpt –project MYPROJECT –host MYSERVER –login MYLOGIN –report 1 –cr_number TestPrj000005 –output_style text**
Which shows the CR report for TestPrj00005, and by using SCM_CMDL_ARGS or scripting typing can be reduced considerably.

To generate a template for use with the XML input option, use the "-template" option to signal for the creation of an XML template:
$ scm_rpt –project MYPROJECT –host MYSERVER –login MYLOGIN –template
After the user has chosen the report and added all of the arguments for the report the command line report mechanism will prompt the user for a file location. The user should choose a file location and name that corresponds to the name of the chosen report, i.e. **ChangeRequestReport.xml**.

The following is an example of a completed CR Assignee report input file in XML format.

```
<Root>
  <Argument>
    <Name val="Report_Name"/>
    <Input_Value val="scm.implementation.reports.XMLCrAssignHistory"/>
  </Argument>
  <Argument>
    <Name val="Project"/>
    <Input_Value val="SCM"/>
  </Argument>
  <Argument>
    <Name val="Assignee"/>
    <Input_Value val="scm"/>
  </Argument>
  <Argument>
    <Name val="Start Date (yyyy/mm/dd)"/>
    <Input_Value val="1990/01/01"/>
  </Argument>
  <Argument>
    <Name val="End Date (yyyy/mm/dd)"/>
    <Input_Value val="2003/04/30"/>
  </Argument>
  <Argument>
    <Name val="Output style"/>
    <Input_Value val="CSV"/>
  </Argument>
</Root>
```

This XML control file can then be passed to the scm_rpt commandline to execute the report and produce the appropriate report. Once the XML control file has been defined it can be re-used many times, for example to automatically generate a weekly status report. In addition, the user could use scripting languages like SED, AWK or Perl to modify the XML input file before it is processed by the command line report function to update date (or other) fields as desired. The following example illustrates how to run the report functionality via an XML input file:

$ scm_rpt –xmlinputfile MYFILE.xml –host MYHOST –login MYLOGIN

NOTE*: See Chapter 10, Reports for a complete description pre-defined reports and how to develop and save customized reports.*

Chapter

14

14 API Concepts and Usage

SpectrumSCM is a process driven Source Configuration Management System that can be used to manage the life cycle of any electronic asset. Users of the system define workflows in the tool that correspond to the processes that are already in place within their organizations. By default, workflows defined in SpectrumSCM are considered ad-hoc, which means that work items can be assigned from any user defined phase into any other user defined phase[2]. The responsibility of moving work items from one phase to another falls on the shoulders of users that have been assigned that particular responsibility[3] (Project Managers, etc…).

With the advent of the graphical workflow system under release 2.4, Spectrum has introduced the capability to perform some automations directly through the UI. These include the "Promotion Recipient" and callouts. The API mechanism is still supported as a general way to extend the SpectrumSCM system and in particular, because it is Java based instead of shell based, it is more efficient.

The purpose of the SpectrumSCM API (Application Programming Interface) is to allow users of the system to construct automated business systems by implementing a well defined set of event triggers and interfaces. An automated business system relieves the burden of making particular users or team leaders responsible for performing certain tasks manually and adds the ability to automate decision making processes, including the automation of interfaces to external systems. External system integration allows issue tracking numbers and content from external systems to be easily integrated into and out of the SpectrumSCM system.

Custom programs that implement the SpectrumSCM APIs are written in the Java programming language and are known as *plugins* in SpectrumSCM. *Plugins* are easily compiled and can easily be added to a running SpectrumSCM server through an XML interface. Plugins are loaded dynamically into the server at run time. Dynamic loading also allows the plugins to be changed by the developer and reloaded without impacting the server. The SpectrumSCM plugin

---

[2] Only users with the proper permissions model can actually assign work items. Such ad-hoc transitions are still fully recorded to provide the required audit trail.

[3] See the SpectrumSCM User Guide Chapter 5 on User Management.

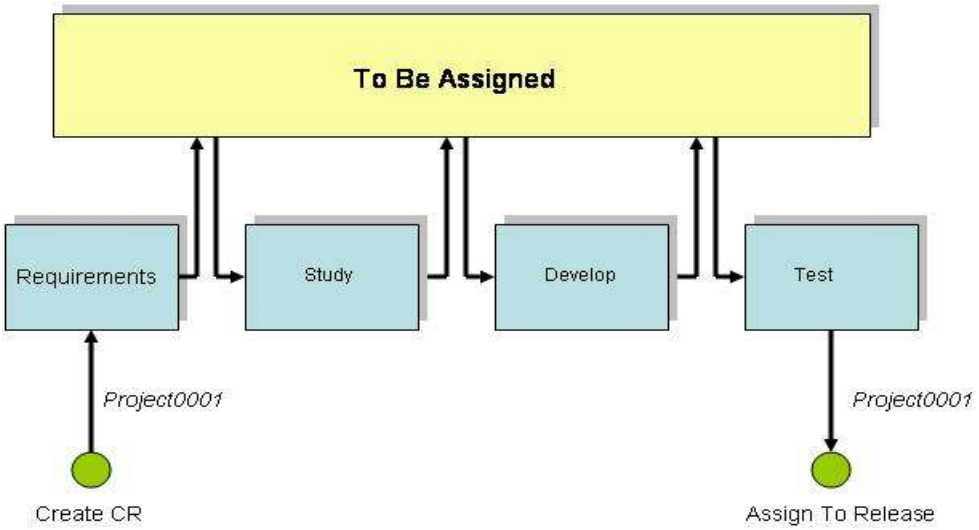
configuration allows for one or more plugins to be actively configured into the system. Individual plugins can be turned on or off by simply modifying the plugins XML definition file.

14.1 Manual vs. Automated Workflow

By default, the SpectrumSCM system provides for the manual ad-hoc workflow process. This decision was made because of the complexities involved with trying to configure and support an automated workflow system right out of the box. The manual system allows the majority of users to quickly and easily get started with SpectrumSCM. The automated systems, both in terms of the graphical workflow and the API, allow the more advanced environments the flexibility they need to configure/automate their needs. Every organization has a different definition of what a workflow system should accomplish and how it should work. Our goal was to be able to handle the common denominator of all workflow systems and then allow for complete customization through server extensions. SpectrumSCM has accomplished this goal with their base system, the graphical workflow and the API extensions.

14.2 A Typical Simple Workflow Process

The SpectrumSCM system allows the user to create completely customizable workflows. Through the use of API mechanism, users can tailor their business process workflow to be cognizant of overall business practices. This allows for external business rules to be executed as part of the transition of an issue or Change request. The following workflow diagram is an example of a trivial software development workflow.

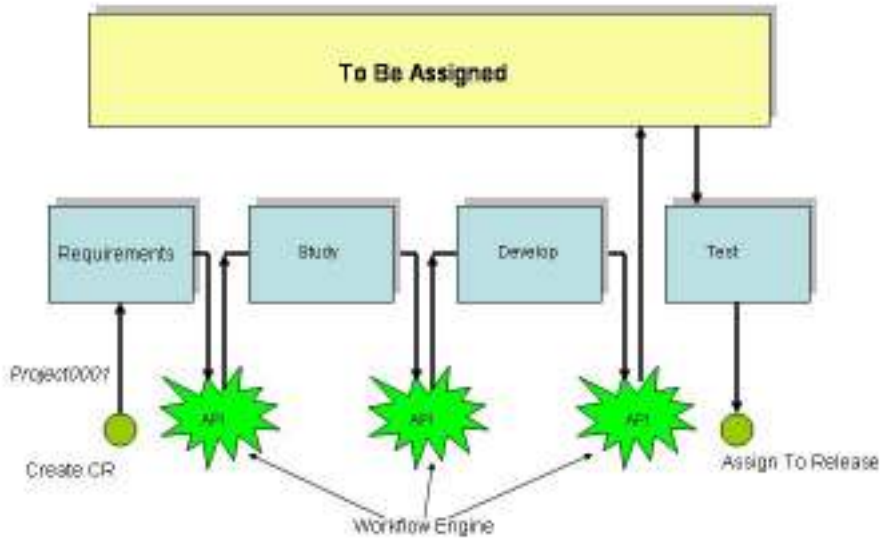


In this example, there are four user defined life-cycle phases *Requirements*, *Study*, *Develop* and *Test*. In the SpectrumSCM system, Change Requests (CRs) can be created and immediately assigned to any phase within the defined life-cycle. In this example, the CR (*Project0001*) was initially created and assigned to the *Requirements* phase. As Change Requests are manually

progressed from phase to phase by the end user, they are not immediately progressed into the next defined phase. Each Change Request is first progressed into the **TBA**[4] super phase. The TBA super phase is where all manually progressed change requests are assigned, unless they are directly assigned into another user defined state by a user with the proper permissions. This supports the working model where a developer or tester does not have assignment privileges, but the project/team lead does. The appropriate user (project/team lead in the previous example) would then manually assign the CR into the next appropriate phase. When change requests are promoted into the TBA phase, e-mail notifications are sent to the appropriate users so that process decisions about where the CR should go next can be made.

14.3 An Automated Workflow

In an automated workflow, progression decisions for manually progressed Change Requests are delegated to an automated workflow engine. The engine, in this case built using the SpectrumSCM API, can apply business process decisions to each individual CR and assign them to the next responsible user, or place the CR in a holding pattern until certain business rules have been satisfied. For instance, a CR may not be eligible for promotion from the *Develop* phase into the *Test* phase until some form of code review has been performed. This diagram depicts the flow of control:



In this workflow, Change Requests are manually progressed out of each phase. When the Change Request enters the TBA super phase, the user defined automated workflow engine is immediately activated and decides where the Change Request should go next, and to whom it

[4] TBA = To Be Assigned

should be assigned. In this example, when a CR is progressed out of the ***Develop*** phase it is not immediately progressed into the ***Test*** phase. But rather it is left in the TBA super phase by the workflow engine until certain business rules have been followed.

14.4 API Activation Points

The SpectrumSCM API provides four separate activation points. The activation points can be used to enable a single automated extension, or can be used separately to implement independent extensions. Plugins that are associated with the activation points are run in separate threads of execution. Running the plugins in separate threads guarantees that the basic responsibilities of the SpectrumSCM server are never blocked.

14.4.1 System Startup and Shutdown

There are two activation points specifically designed to work with system startup and shutdown. When the SpectrumSCM server is started or stopped, all registered plugins are searched to see which ones implement the *SystemListener* interface. Each plugin that implements this interface is called from a separate thread of execution. It is possible that, given the length of execution of any particular startup or shutdown transaction, several plugin transactions may run concurrently.

- Example Usage

Use the startup and shutdown activation points to connect the SpectrumSCM server to external issue tracking systems or other external business systems.

14.4.2 Change Request Transition

Change Request transitions define the third plugin activation point. When a Change Request transitions from a user defined life-cycle phase into the TBA super phase, all plugins that implement the *ChangeRequestListener* interface are executed in separate threads. The change request transition activation point is one of the most useful activation points. Fully automated workflow systems will use this activation point as an event trigger for applying custom change request routing logic and other business processes.

- Example Usage

Use the change request transition activation point to implement a business rule sensitive automated workflow system.

14.4.3 Change Request Creation

Change Request creation defines the last plugin activation point. When a new Change Request is created in the SpectrumSCM system, this plugin activation point will be called in a separate thread of execution.

- Example Usage

Use the change request creation activation point to communicate change request creations to downstream project management systems.

14.5 Implementing the Interfaces

In order to create a user defined plugin, API users must first implement one or both of the Java interfaces defined by the API described below.

14.5.1 The *SystemListener* Interface

This interface defines the methods that are used by the startup and shutdown activation points. Users can implement this interface to create long running processes that are integrated directly into the SpectrumSCM server itself. For example, this interface could be implemented in order to create an active interface with an external system.
The following code block illustrates how to implement the *SystemListener* interface:

```
import scm.pub.interfaces.SystemListener;
public class WorkflowEngine implements SystemListener {
        public void startUp() {…}
        public void shutDown() {…}
}
```

14.6 The *ChangeRequestListener* Interface

This interface defines the methods that are called during Change Request creation and transition. The *ChangeRequestListener* interface defines the following two methods:

- changeRequestTransition(String project, ChangeRequest_d cr_d)
- changeRequestCreated(String project, ChangeRequest_d cr_d)

These two methods are passed the name of the project as a String and the data structure ChangeRequest_d. The ChangeRequest_d data structure contains all of the current information for the CR involved in the creation or transition. Users would need to write code similar to the following snippet in order to implement the ChangeRequestListener interface:

```
import scm.pub.interfaces.ChangeRequestListener;
import scm.pub.transport.ChangeRequest_d;

public class WorkflowEngine implements ChangeRequestListener {
        public void changeRequestTransition(String project, ChangeRequest_d cr_d) {
                …
        }

        public void changeRequestCreated(String project, ChangeRequest_d cr_d) {
                …
        }
}
```

14.7 Interacting with the System

There are three first class objects defined in the SpectrumSCM API that can be used to interact with a running SpectrumSCM server. These objects implement the *Proxy* design pattern[5] as described in the GOF (Gang of Four) design patterns book. Each object is a proxy or stand in for the corresponding persistent object located in the SpectrumSCM server.

- **ChangeRequest:** The ChangeRequest object is a proxy object for a live ChangeRequest object located in the SpectrumSCM server. Calling the getInfo( ) method on this object will result in all of the latest information for this particular Change Request to be returned. This object can be used to promote the Change Request into another phase or add history elements and other notes directly to the Change Request.

- **Project:** The Project object, just like the ChangeRequest object, is a proxy object for the live Project object located in the SpectrumSCM server. This object defines methods that allow the caller to extract project related information directly from the server. This object also contains methods that allow for the creation of new ChangeRequests.

- **ScmSystem:** The ScmSystem object, just like the other objects, is a proxy for the actual ScmSystem object. This object implements both the *Proxy* design pattern as well as the *Singleton* design pattern. The getInstance() method is used to retrieve **the one and only instance** of this object. The ScmSystem object contains methods that allow the caller to retrieve a list of all active projects in the system as well as a list of all registered system users. The object also contains an interface into the SpectrumSCM e-mail system, which allows the caller to send E-mail messages to interested parties.

A workflow engine can be designed to use these objects transiently for short term operations, or the objects, once constructed, can be stored at a higher scope level for use at a later time. The decisions for the design of the workflow engine are left up to the implementer.

14.8 Transport Objects

Transport objects are used as simple data structures to pass large amounts of information into and out of the proxy objects. The following is a list of all of the Transport Objects defined in the SpectrumSCM API.

- **AttributeMap_d:** An AttributeMap_d object is returned from the method Project.getProjectChangeRequestAttributes( ). The AttributeMap_d object is a mapping of Change Request attribute names to a set of attribute values.
- **ChangeRequest_d:** An ChangeRequest_d object is returned from the method ChangeRequest.getInfo( ). This data structure contains all of the current and historical information for the given Change Request.

---

[5] Gamma, Helm, Johnson, Vlissides: Design Patterns, Elements of Reusable Object-Oriented Software. 1995 ISBN: 0-201-63361-2

- **ChangeRequestCreator_d:** This object is specifically used to create a new Change Request in the system. The contents of this object describe who the new Change Request will be assigned to, in what phase and on which Generic (branch).
- **ChangeRequestHistory_d:** The ChangeRequestHistory_d object is actually a sub-object that is returned as part of the ChangeRequest_d object. It contains historical information about the Change Request.
- **User_d:** An User_d object is returned from the method ScmSystem.getUserInfo( ) as well as Project.getUserInfo( ). In the case of calling the getUserInfo( ) method on the Project object, more information about the users current category assignments are returned.
- **CRFileDescriptor_d:** This data structure contains the version, generic and path information for a file under source code control. A set of CRFileDescriptor_d objects are returned from the method ChangeRequest.getFileDescriptors( )

14.9 Compiling the Code

In order to compile an API based plugin, the developer must have access to the SpectrumSCM server jar files. These jar files are located in the following directory:

<SCM_INSTALL_DIR>/lib

The developer's CLASSPATH environment variable must be extended to include the **scmServer.jar** file. The extended CLASSPATH variable should look like the following when complete:

CLASSPATH="$CLASSPATH:<SCM_INSTALL_DIR>/lib/scmServer.jar"
in Unix shell notation, or
CLASSPATH="%CLASSPATH%;<SCM_INSTALL_DIR>/lib/scmServer.jar"
in Windows batch notation.

Note that the path separators are platform dependent. Once the CLASSPATH variable is set properly, compile the code with the normal java compiler arguments.

14.10 Installing the Code

The developer's compiled code must be included in the SpectrumSCM server's CLASSPATH. A directory named *custom_plugins* exists in the SpectrumSCM server directory structure and default CLASSPATH, the developer's code can be placed in this directory.
<SCM_INSTALL_DIR>/SCM_VAR/custom_plugins

If the developer's code needs to reside in a jar file, the jar can be placed in the SpectrumSCM server lib directory. The script that is used to start the server must be modified to include this jar file. If the server is running on a Windows platform edit the file startServer.bat. If the server is running on a Unix or Mac platform, edit the file startServer.

14.11 Modify the Plugins XML file

In order to tell the SpectrumSCM server that a plugin has been added to the system, the plugins XML file must be modified. This file is located in the following directory:
<SCM_INSTALL_DIR>/SCM_VAR/etc/plugins.xml

The following is an example of a valid plugins.XML file:

```
<!-- This is an example of what the plugin file should look like -->
<!-- Note that the STATUS element can be set to either ENABLED OR DISABLED -->

<PLUGINS>
        <!-- Put your plugin declarations here -->
        <PLUGIN>
                <NAME VALUE="Coreys Plugin"/>
                <CLASS VALUE="com.scm.TestPlugin"/>
                <PROJECT VALUE="SCM"/>
                <STATUS VALUE="ENABLED"/>
        </PLUGIN>
</PLUGINS>
```

There are four (4) XML elements that must be in this file. Each element is described below:

- **NAME:** This is simply the name of the plugin and is used initially as a key to the plugin itself.
- **CLASS:** This is the actual class name of the class that implements the listeners described above.
- **PROJECT:** This is the Project name that this particular plugin should be associated with. The same plugin can be associated with separate Projects as long as the developer is careful not to include Java class attributes with global class scope in the plugin.
- **STATUS:** This determines whether the plugin should be used or not. Set this to DISABLED if the plugin needs to be turned off.

14.12 Example plugins

The basic plugin skeleton is trivial to construct. Here is a working plugin that implements both interfaces but doesn't really do anything:

```
import java.io.*;

import scm.pub.interfaces.*;
import scm.pub.transport.*;
import scm.pub.exceptions.*;

public class TestPlugin implements SystemListener, ChangeRequestListener {

        public void startUp() {
                System.out.println("Startup called..");
        }

        public void shutDown() {
                System.out.println("ShutDown called..");
        }

...
```

```
…
        public void
        changeRequestCreated(String project, ChangeRequest_d cr_d) {
                System.out.println("changeRequestCreated..");
        }

        public void
        changeRequestTransition(String project, ChangeRequest_d cr_d) {
                System.out.println("ChangeRequestTransition called...");
        }
}
```

Note that the only thing this plugin does is report to the standard output when the interface methods have fired. Compile and add this plugin to the system to see which user level actions cause these methods to execute. For instance, creating a new Change Request will cause the changeRequestCreated() method to execute. Progressing that CR into the TBA state will cause the changeRequestTransition() method to execute. The start() and stop() methods will execute when the server is started and stopped.

This next code snippet accesses all of the major first class objects and extracts some information from a Change Request:

```
        public void
        changeRequestCreated(String project, ChangeRequest_d cr_d) {
                java.lang.System.out.println(cr_d.toString());

                try {
                        ScmSystem       sys     = ScmSystem.getInstance();
                        Project         proj    = new Project(project);
                        ChangeRequest   cr      = new ChangeRequest(proj, cr_d.getCRId());

                        System.out.println(cr.getInfo().toString());

                } catch(Exception e) {
                        System.err.println("Caught: " + e.getMessage());
                }
        }
```

Note that this code is actually redundant. The ChangeRequest_d information that was extracted from the cr.getInfo() call was already handed to the enclosing method as an argument. The example is just to show how to access some of the more important objects. Also notice that the System object is accessed by simply calling the static method getInstance() on the System class. Projects and ChangeRequests can be constructed as often as necessary. All of these objects can be stored for later use once they have been constructed.

This next example is a more complete example of an automated workflow engine. In this code snippet, the ChangeRequest passed to the transition method is examined and automatically progressed into the next life-cycle phase:

```
public void changeRequestTransition(String project, ChangeRequest_d cr_d) {

        ScmSystem       sys    = null;
        Project         proj   = null;
        ChangeRequest cr       = null;
        Vector       phases    = null;
        try {
                sys    = ScmSystem.getInstance();
                proj   = new Project(project);
                phases = proj.getLifeCyclePhases();
                cr  = new ChangeRequest(proj, cr_d.getCRId());
        } catch(Exception e) {
                System.err.println("Caught: " + e.getMessage());
                return;
        }

        try {
                ChangeRequest_d          crObj = cr.getInfo();
                ChangeRequestHistory_d crh_d = null;

                Vector        history    = crObj.getHistoryInfo();
                String        lastPhase  = null;
                String        nextPhase = null;
                int           index      = -1;

                for(int indx = history.size() - 1; indx >= 0; indx--) {
                        crh_d = (ChangeRequestHistory_d)history.get(indx);
                        if(crh_d.getPhase().endsWith("note")) {
                                continue;
                        } else {
                                lastPhase = crh_d.getPhase();
                                break;
                        }
                }
                index = phases.indexOf(lastPhase);
                nextPhase = (String)phases.get(index+1);
                cr.assignToPhase(crh_d.getUser(), crObj.getCurrentGeneric(),
                                 nextPhase, "Here's some more work");
                sys.sendEMail("joe@x.com", "CR Status", "Assigned CR <" +
                                        crObj.getCRId() + "> to phase <" +
                                        nextPhase + ">");
        } catch(Exception e) {
                System.err.println("Caught: " + e.getMessage());
        }
}
```

Unfortunately, in order to get all of the code into this single example, some of the empty lines had to be removed from the text and the vast majority of the error handling code has also been removed. The last few lines in the example are the most important. The method assignToPhase() called against the ChangeRequest actually assigns this particular CR to the next phase in the life-cycle and adds a small note. The next line uses the e-mail interface to send mail to an interested party.

14.13 Summary

The SpectrumSCM API allows a developer to easily create fully automated business processes and external system interfaces. Currently the API is limited to this type of functionality. The developers of the API chose to exclude a file level listener interface from the current API implementation. The existence of such a listener has limited use in a fully integrated tool like SpectrumSCM. One of the basic tenets of SpectrumSCM is that individual files are worked or changed as part of a larger issue or change request. In this scenario, the need to know when a single file has changed, or to act upon a single file change is unnecessary. In other systems that are *interfaced* instead of *integrated,* this type of functionality may be necessary as individual file changes are not already associated with a traceable statement of work. File information can be retrieved through the corresponding Change Request.

15 LDAP Support for SpectrumSCM™

Chapter 15

15.1 Introduction: Over the past few years, the Lightweight Directory Access Protocol (LDAP) has gained wide acceptance as the directory access method for the Internet and organizational intranets. A directory is a specialized database that is used for storing information regarding various users, applications or resources on a network. A company-wide implementation of an LDAP directory service provides a one-stop solution for any application, running on any platform, that needs access to company specific information like employee details, customer information, licenses, security keys etc. Of late, LDAP is gaining popularity as a centralized authentication system for users on a network. Storing user names and passwords in an LDAP directory as opposed to a local application-specific database not only provides for enhanced security but also provides users with the convenience of using a single username and password for accessing company-wide resources and applications. Of course, the security benefit realized by using a centralized LDAP authentication scheme depends on how secure the centralized database is.

An LDAP directory service is based on the client server architecture in which an LDAP client sends requests to an LDAP server that processes the request and returns the appropriate response. The server hosts the LDAP information database that is used for storing and retrieving information. The directory service is generic in nature and can be used for storing and retrieving any kind of information. The basic unit of information in a LDAP directory is an "entry". Each entry may represent an organization, user, resource or any object of interest. Entries are composed of attributes which contain information about the object. Every attribute has a name and one or more values. Entries are organized in a hierarchical, tree like structure. Each entry is uniquely identified by a "Distinguished Name" (DN). The DN acts as a primary key for the entry and is derived from the DN of its ancestor in the tree. The DN of the highest node in the tree is called the Root DN.

An LDAP server may implement its own schema or a standard schema as defined in RFC 2252. Mainstream implementations of LDAP include Netscape Directory Server, Sun One Directory, Microsoft's Active Directory and Novell's Directory Service and Open LDAP. Though LDAP defines operations for updating the stored information, it is optimized for read operations. The most common LDAP operations can be classified into two categories:

Query - This includes operations for searching a particular entry in the directory or for bulk retrieval of a set of entries that satisfy a search criterion.

Authentication - This includes operations to establish and end a session between an LDAP client and server, and access control. LDAP provides different levels of security ranging from unauthenticated anonymous access, simple authentication to secure authentication using SASL/SSL.

15.2 LDAP Support for SpectrumSCM: SpectrumSCM (version 1.3.7 and above) is LDAP enabled and is fully compliant with LDAP v3. LDAP support for SpectrumSCM includes the following features:

LDAP Authentication: This allows users of SpectrumSCM to authenticate against a centralized LDAP database as opposed to the SpectrumSCM database. LDAP authentication provides enhanced security for user credentials (passwords) by storing them in a secure centralized database instead of a local application specific database. Thus the confidence level for user password becomes a function of how secure the LDAP database is. This is particularly useful for organizations whose security policies prohibit them from storing user credentials in application-specific databases. LDAP authentication also provides users with the added convenience of using a single user name and password for all LDAP enabled applications.

LDAP Import: This feature allows an administrator to import details regarding a particular user from the LDAP database. Users in SpectrumSCM are defined by the following attributes: user-id, name, phone, email, location. The LDAP import facility allows an administrator to import these details from an LDAP database while adding/modifying users.

LDAP Search: This feature allows a user (with the required privileges) to search the LDAP database for users satisfying a particular search criteria. Users can be searched based on their name, phone number, email address or location. The search result can be used for bulk addition of users into the SpectrumSCM database. The import and search features are useful in organizations that store user-specific details on a company-wide HR database or on a publicly available LDAP server.

Security: LDAP support for SpectrumSCM defines different levels of security ranging from anonymous binding, simple password protected binding to strong SSL based mechanisms. SSL support for LDAP provides confidentiality protection for authentication through the use of certificates and integrity protection by encrypting the data transmitted over the wire. SSL support for LDAP uses SSL v3.0 or TLS v1.0. The LDAP server should support the above mentioned mechanisms before they can be used with SpectrumSCM.

15.3 Using LDAP Support for SpectrumSCM: Configuring SpectrumSCM for LDAP support is quick and easy. The *scm.properties* file includes a few parameters that need to be configured before the LDAP related features for SpectrumSCM can be used. The parameters are as follows:

LDAP.useAuth - This parameter specifies whether LDAP authentication for SpectrumSCM should be enabled.

Keywords: YES, NO
Default: NO

LDAP.useImport - This parameter specifies whether the LDAP import and search features should be enabled.

Keywords: YES, NO
Default: NO

LDAP.useSSL - This parameter specifies whether SpectrumSCM should use SSL protection for communicating with the LDAP server.

Keywords: YES, NO
Default: NO

LDAP.server - This parameter specifies the LDAP server's address

Examples: ldap.xyz.com, directory.verisign.com, 192.168.100.7

LDAP.port - This parameter specifies the port number for the LDAP server

Default: 389 for LDAP, 636 for LDAP with SSL

LDAP.dn - This parameter specifies the distinguished name (DN) used for LDAP authentication and binding. UU in the DN string acts as a placeholder for the login string entered by the user when he/she attempts to login to SpectrumSCM The place holder is replaced with the login string entered by the user when he/she attempts to login to SCM. If the 'useNameAsUU' option is set, then the users name as recorded in the SpectrumSCM database is used as UU instead of the user id.

Also note: multiple DNs can be specified, LDAP.dn would be the primary, LDAP.dn2 the secondary, LDAP.dn3 the tertiary, etc

LDAP.useNameAsUU	true
LDAP.dn	uid=UU,dc=SpectrumSoftware,dc=net
LDAP.dn2	uid=UU,ou=Development,dc=SpectrumSoftware,dc=net

LDAP.useNameAsUU – As above. If false, the user-id is used to replace the UU tag. If true, the user name as specified in the SpectrumSCM database relative to the supplied user-id, is used.

Keywords: TRUE, FALSE
Default: FALSE

LDAP.searchbase - This parameter specifies the search base for the LDAP server i.e. the starting point for all searches. In most cases, it is the DN of the top-most entry in the hierarchy defined by the LDAP database

Default: NULL

The following parameters are used to map between the attributes defined in the LDAP server's schema and the information required by SpectrumSCM. *Ldap.uid.mapping* is a required parameter while other parameters are optional. The *LDAP.uid.mapping* attribute maps the LDAP login name to the SpectrumSCM user ID. The value of this attribute in the LDAP directory MUST match the user ID used in SpectrumSCM.

Parameter	**Example**
LDAP.uid.mapping	uid

LDAP.name.mapping	cn
LDAP.phone.mapping	telephonenumber
LDAP.mail.mapping	mail
LDAP.location.mapping	loc

The example attributes are based on the standard LDAP schema defined by RFC 2252. Once the above parameters are correctly specified, SpectrumSCM should be able to communicate with the specified LDAP server. The authentication and import/search features can be used separately or in tandem. In either case, the integrity of the information transmitted between the SpectrumSCM server and the LDAP server can be protected by enabling SSL.

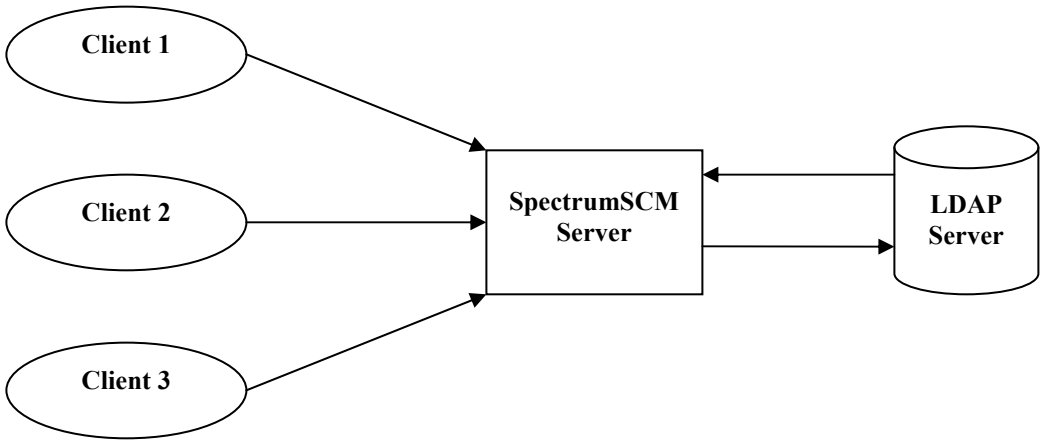
In general it is usually better to ensure that all the parameters are correct and functioning through the use of the import/search features BEFORE turning on the useAuth. If useAuth is turned on but the LDAP parameters are not correct, the login authentication will fail and you won't be able to log in to SpectrumSCM.

Additional note
The location attribute is by default a general information text field. If, however your organization has many disparate organizational units within the LDAP database this can be in-efficient to search. Instead, SpectrumSCM can use the location attribute/field to cache the users specific distinguished name (DN).

LDAP.useLocationForDN	true
LDAP.location.mapping	distinguishedName

With this setting, an import operation will import the users distinguishedName attribute from the LDAP database and place it in the location field in the SpectrumSCM database. If this user then attempts to perform LDAP operations such as authentication (login with useAuth on), the DN from the location field will be used ahead of the LDAP.dn parameters.

15.4 Architectural Diagram:



15.5 Frequently Asked Questions:

How do I use LDAP authentication ?

Configure the LDAP parameters in the scm.properties file. Make sure that the user IDs can be mapped to entries on the LDAP server. Turn LDAP authentication ON (with or without SSL). Restart the SpectrumSCM server. Use your LDAP user name and password to login

How do I use LDAP authentication with SSL ?

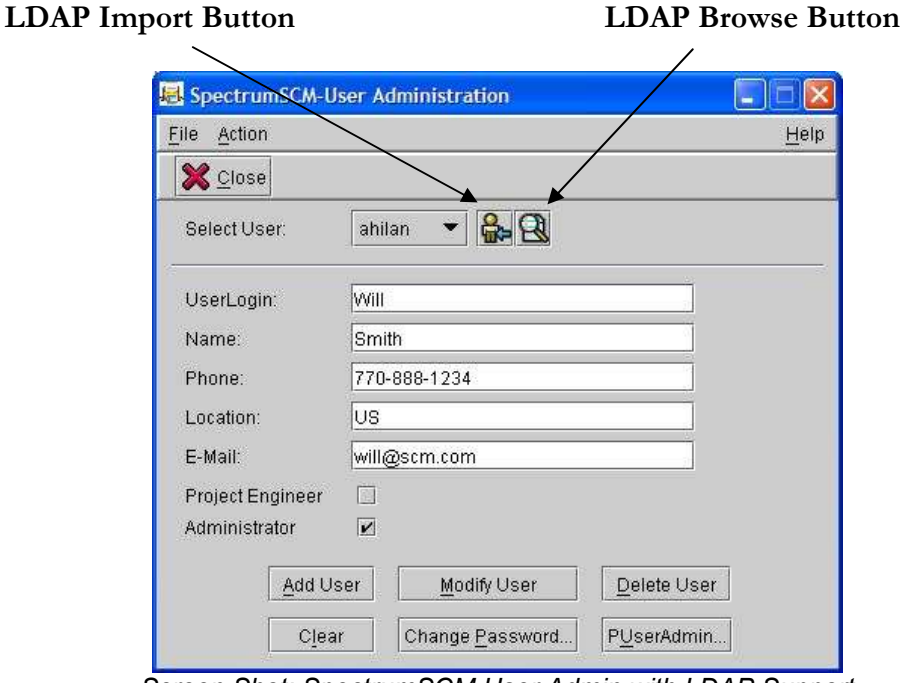
Make sure that the JRE on the client machine will accept certificates signed by the Certificate Authority used by the LDAP server's certificate. Turn on SSL support by specifying YES for the LDAP.useSSL parameter. CA certificates can be installed using the 'keytool' program provided by JAVA as follows:

 # cd JAVA_HOME/lib/security ...
 # keytool -import -file ca.cert -keystore cacerts

JAVA_HOME is the JRE install directory. *ca.cert* is the CA certificate file. The *cacerts* (or similar) keystore can be found in the /lib/security directory under JAVA_HOME

How do I use LDAP import ?

The name field acts as the main input box for the import feature. Use the LDAP import icon to import user details from the LDAP server. Details are imported based on the mappings specified in the *scm.properties* file



Screen Shot: SpectrumSCM User Admin with LDAP Support

What part does the mapping play ?

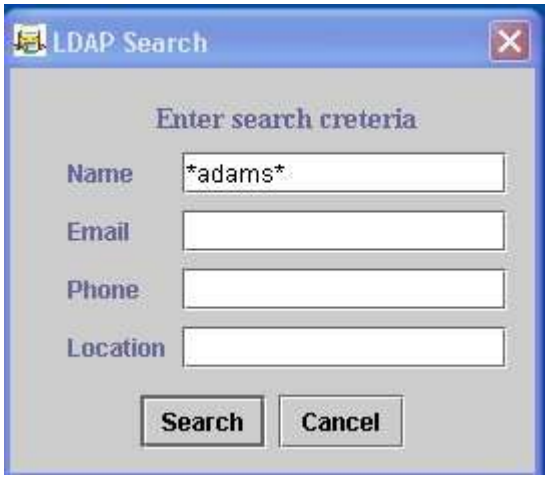
The mapping parameters are used to map between the attributes defined in the LDAP server's schema and the user details in the SpectrumSCM database. SpectrumSCM users are defined by five attributes (uid, name, mail, phone, location). The uid mapping is important for LDAP authentication. Each user on the LDAP server must have a unique ID which can be identified by some attribute name say 'user_id'. This attribute must map to the uid attribute used to identify a user in SpectrumSCM. Thus the mapping will be configured as:

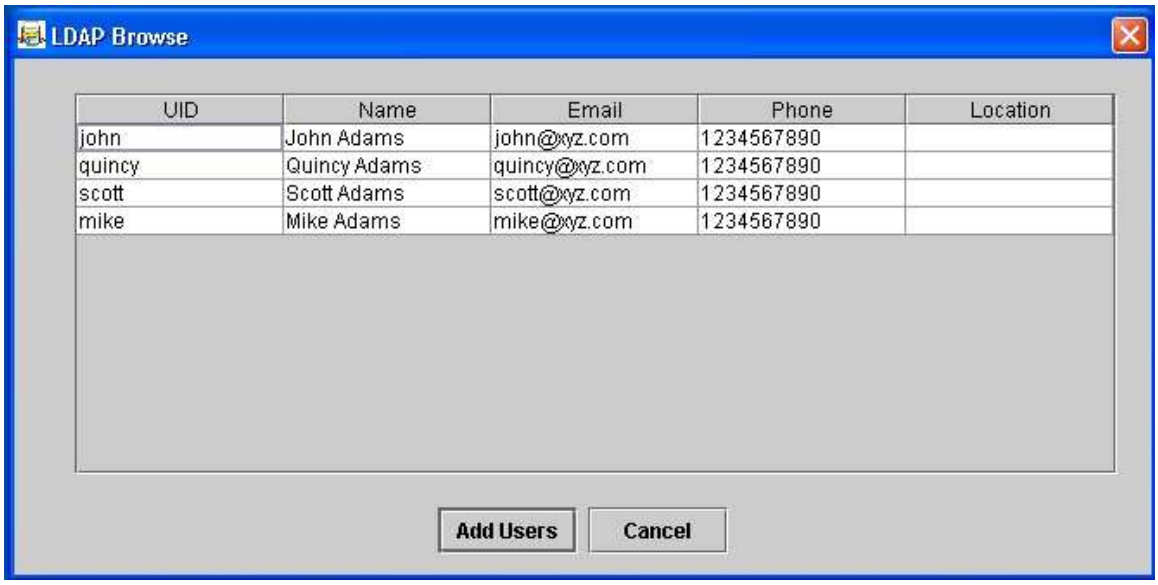
LDAP.uid.mapping user_id

Other attributes are mapped in a similar manner. However these are attributes are optional.

How do I use the search feature ?

Click the LDAP search icon. If you are not yet authenticated, you will be prompted for authentication. Enter the search criteria. The results table will allow you to add users in bulk. If you are using a public server with thousands of entries, there is a possibility of exceeding the server limit if your search criterion is not refined.





Screen Shot: LDAP Search Panel and Results Table

Can I use a public server ?

Of course, you just need the proper configuration parameters. However, in most cases you cannot use a publicly available server for authentication unless you happen to have an account with the server. You can definitely use the server to import information. As an example, here are the settings for the Verisign Server

LDAP.useAuth	NO
LDAP.useImport	YES
LDAP.useSSL	NO
LDAP.server	directory.verisign.com
LDAP.port	DEFAULT
LDAP.uid.mapping	mail
LDAP.name.mapping	cn
LDAP.phone.mapping	telephonenumber
LDAP.mail.mapping	mail
LDAP.location.mapping	loc

Comment out the LDAP.dn and LDAP.searchbase parameters

Where can I find out more about what went wrong ?

If the client side dialogs do not provide a clue as to what went wrong, check the server log file.

Why does SSL authentication seem a bit slow ?

SSL authentication employs an encryption, decryption, message code checking and other mechanisms. The speed depends upon the key length used. The longer the key, the higher is the level of security and the slower is the processing speed.

How do I use the UU keyword in scm.properties ?
The UU is a placeholder for the login that you enter. If your LDAP.dn parameter has been set to uid=UU,dc=SpectrumSoftware,dc=net and you enter 'john', the fully resolved distinguished name (DN) that is passed to the LDAP server is uid=john,dc=SpectrumSoftware,dc=net. As another example, if you typically enter the full DN set the LDAP.dn parameter to UU

Can I use wildcards in my searches ?
Yes, you can use LDAP compliant wildcards. The most common is *.

What if multiple entries match the input value while importing an LDAP user ?
The tool will always accept the first value in the set if multiple entries are found.

Chapter

16

16 Graphs and Charts

In this chapter you will become familiar with all the SpectrumSCM graphs and charts - how to create them and the meaning and use of the information in these graphs. You will also learn how to personalize your graphs and save personalized versions as well has how to create a custom graph and install it in the system

The Spectrum system provides comprehensive set of pre-defined graphs . SpectrumSCM provides many graphs and charts necessary for the system to be as useful as possible right out of the box.

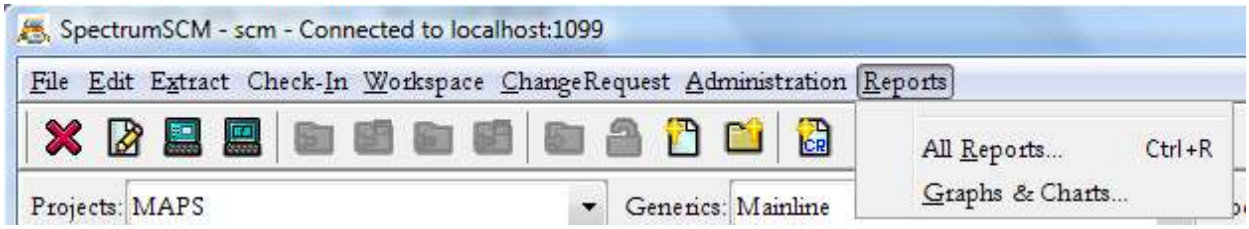
This feature is available as an additional optional module (SpectrumSCM Graphical Dashboard 1.0). The SpectrumSCM Graphical Dashboard 1.0 works in conjunction with the SpectrumSCM 3.0 version. The **Graphs & Charts menu item will be activated** when you purchase this module.

The following list describes the basic graphs that are packaged with the **SpectrumSCM Graphical Dashboard 1.0 Module**.

1. **Graph the CR by Project**: Graphs the CR counts across the selected projects.
2. **Graph the User CRs by State:** Graph all the CRs assigned to the selected user by their state.
3. **Graphs the CRs by State, Attribute:** Graph all the selected CRs by their attribute and optionally any of the defined CR attribute (i.e severity)
4. **Graph the active CRs by attribute**: Graph all active CRs by any of the defined CR attribute (i.e severity).
5. **Graph Change Request assigned to Generic(s)/Release:** Show all CRs that are assigned to the specified generic(s) by each of its releases.
6. **Graph Change Request assigned to Release(s):** Show all CRs that are assigned to the specified release(s) of a generic.
7. **Graph Change Request assigned to a User:** Graph the number of CRs currently assigned to the specified user.
8. **Graph CRs by Period:** Graph all the active CRs by their activity or creation date during a specified period.
9. **Graph CRs by Period/User:** Graph all the active CRs by Period and User
10. **Graph CRs by Period/Attribute:** Graph all the active CRs by Period and any of the defined CR attribute.
11. **Graph CRs by Period/Phase:** Graph all the CRs by Period and phase.

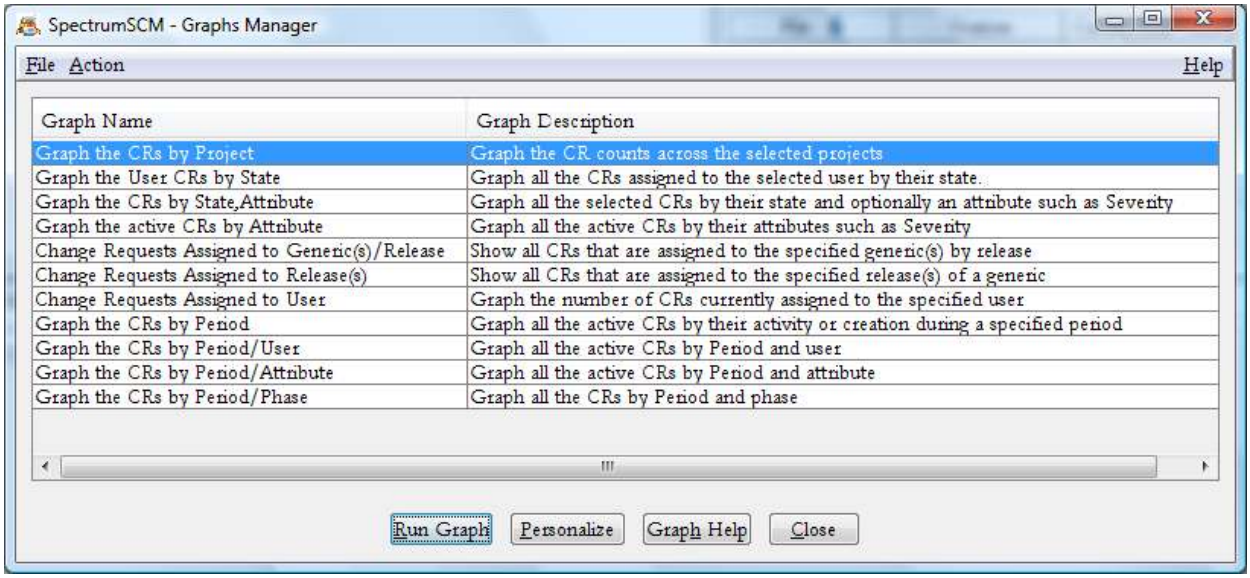
16.1 Running A Graph

Access the reports via the **Graphs & Charts** option on the **Main Screen** menu

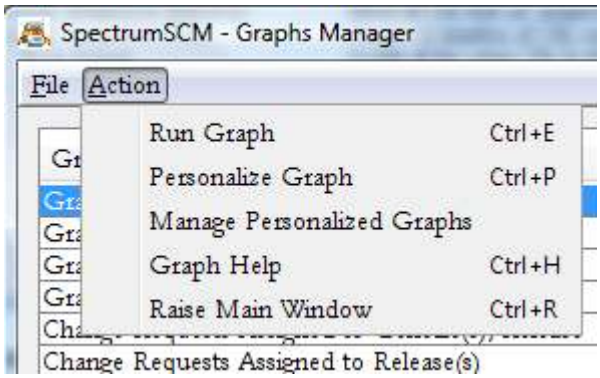


Select a Graph

Select the graph that you desire, a brief description is listed to the right. More graph specific help can be displayed by selecting the **"Graph Help"** button.



The **Action** menu item offers the same options as on the screen plus an additional option to Manage Personalized Reports (see below for details).

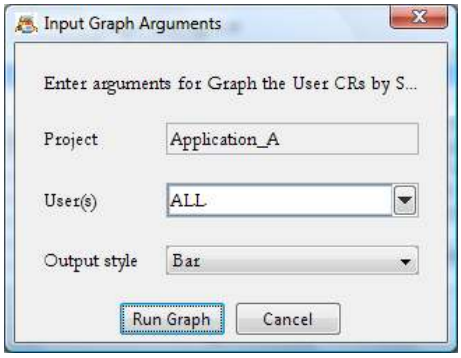


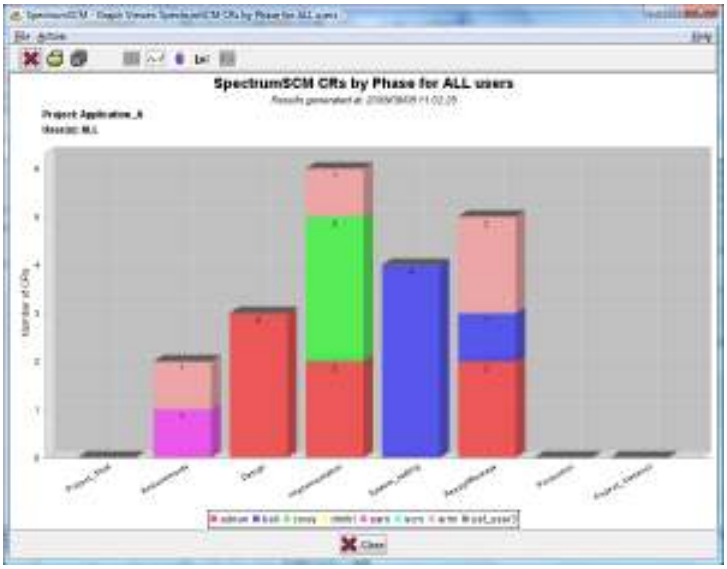
Run a Graph

When the **Run Graph** button is selected a panel will be presented requesting all of the required input parameters. Some of these will be defaulted to values saved or selected on the main screen.

Enter any changes or refinements to the graph parameters and run the graph. The graph will be presented in the standard SpectrumSCM Graph Viewer.

You can print or save these graphs, to publish or include them as part of your status reporting.





16.2 Viewing A Graph

The graph viewer shows the graph by default as bar chart. Line and Pie charts can also be selected as one of the parameters (Output Style) or can be changed dynamically when viewing the graph. Zooming capabilities are also available both from the action menu items and by dragging over an area of interest with your mouse.

More information about the graph display can be found on the SpectrumSCM Graph Viewer Help menu.

The graphs have further drill down capabilities, to launch the specific change request details for further analysis and decision making.

16.3 Printing and Saving A Graph

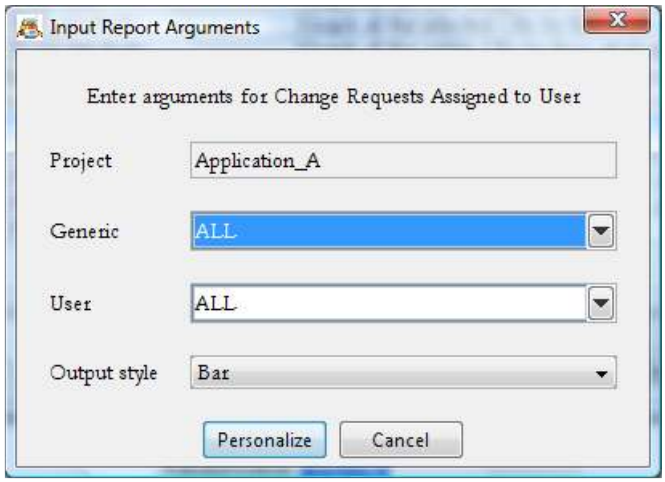
The graph can be **printed or saved** to disk (as JPEG, PNG image or PDF file).

16.4 Personalizing a Graph

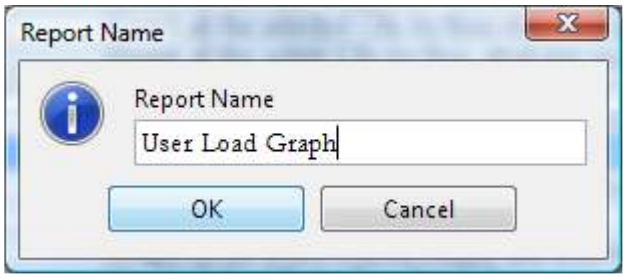
Graphs can be personalized by selecting the **Personalize graph** button. Personalization allows the user to pre-specify popular graphs and assign them up onto his/her main screen reports menu.

The personalize report button presents an argument panel similar to the one for entering the regular graph arguments. If a value is provided to a personalization argument its value is essentially hard-coded to the value for the personalized graph. If a check-box is available and you select it then when the personalized graph is run the value for this field will be obtained from the current main screen selections.

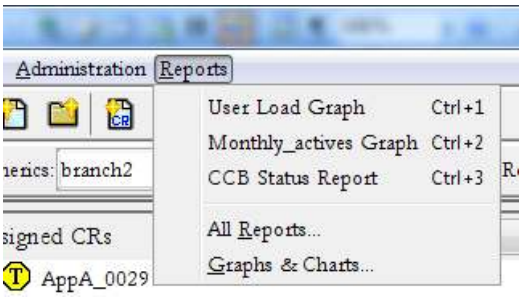
When the **Personalize** button is pressed you can enter the name that you wish to assign to this graph. This graph can now be executed directly from the main screen reports menu. To delete a personalized graph, use the **Manage Personalized Graphs** menu item select the graph(s) you wish to delete and click on the **Delete Graph** button.



Click the **Personalize** button to enter the name that you wish to assign to this report and click OK.



This graph and all other personalized graphs saved by the user can now be executed directly from the main screen reports menu without re-entering parameter values. Quick keys (Control + number 1 thru 9) will execute the first 9 personalized reports and graphs.



By checking the Project and "End-Date" during the personalization of this example report, it means that the report will find data on the current project and to the current date when the report is executed. As opposed to always running to the "Components" project and the specified end-date.

16.5 Custom Graph

Like custom reports, custom graphs can be requested from Spectrum Software Product Support. Straight forward graphs that has simple custom changes is part of your maintenance agreement and most graphs can be created and shipped to the customer within 24 hours of receiving the request. Once the graph has been delivered to the customer, it's simply a matter of starting the graph installer and the custom graph will be automatically installed:

```
$ java –jar GraphInstaller.jar
```

The installer automatically installs the graph in the custom graphs directory and the graph itself is immediately available for use on the graphs & charts screen. There is no need to restart the server after installing the graph. On some operating systems you may be able to simply double click the GraphInstaller.jar file to invoke the installation.

Index

Notes