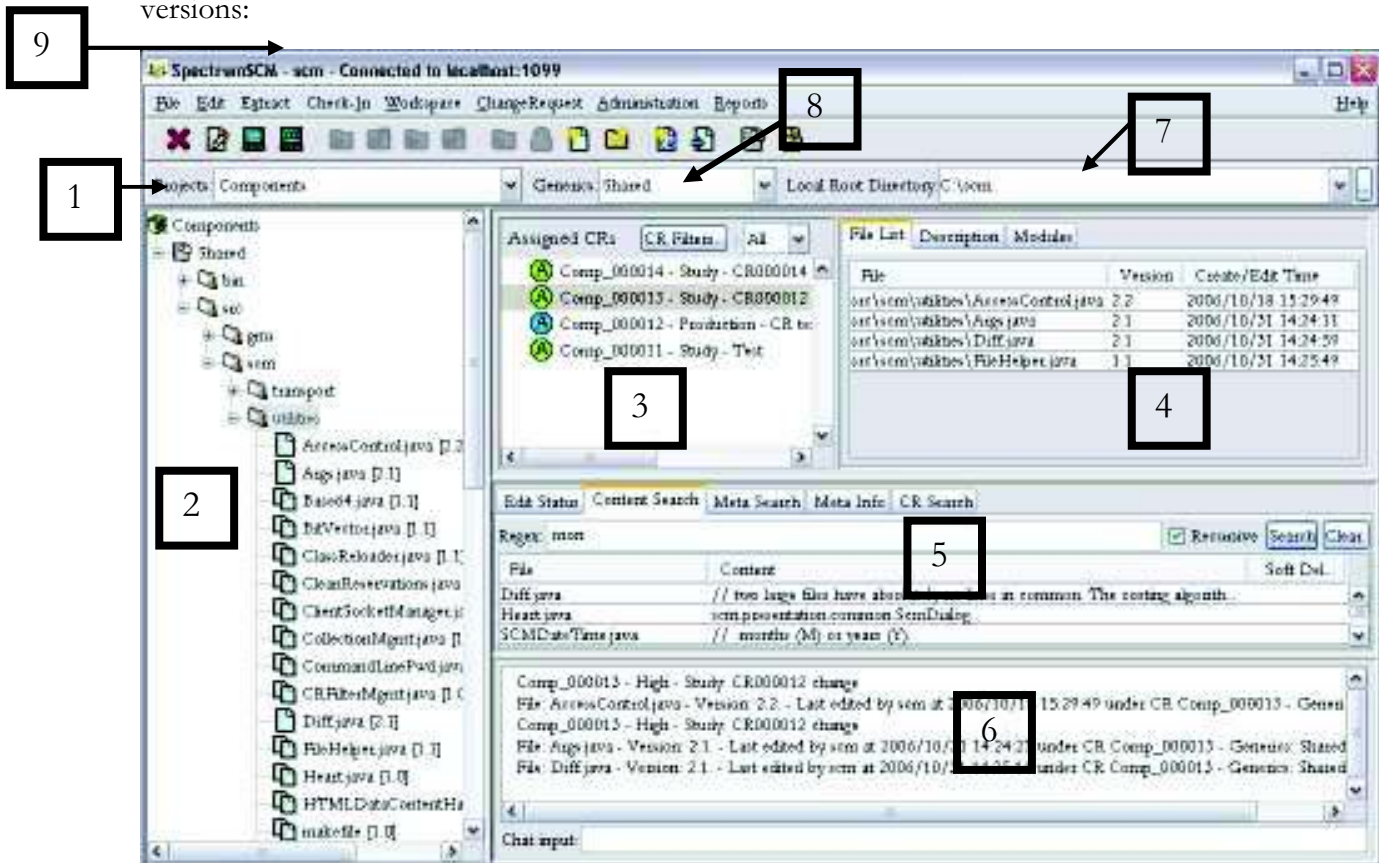


8 Version Control and Source File Management

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.



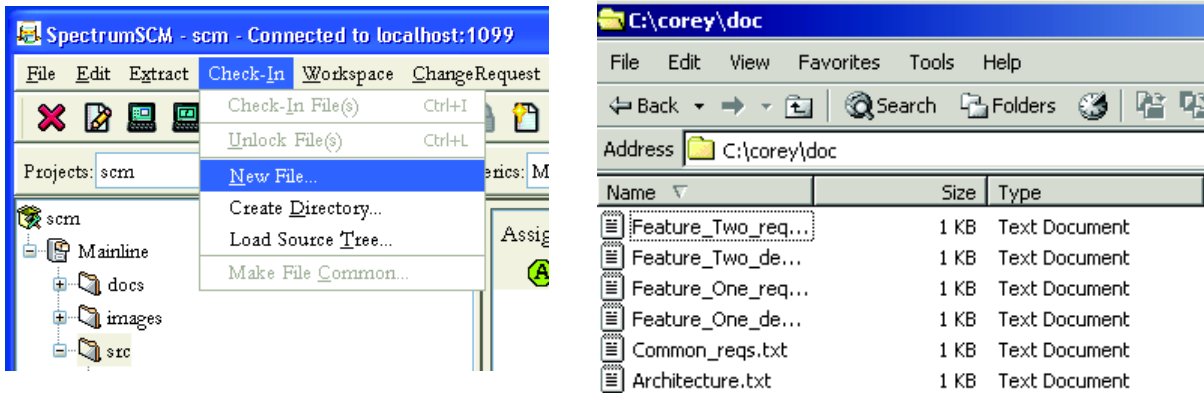
Alternatively you can use the **DragNDrop feature** available in SpectrumSCM to check-in or load a single file or multiple source files by simply **Dragging** the files that are of interest directly from desktop or file system and **Dropping** it into appropriate folder location on the project repository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

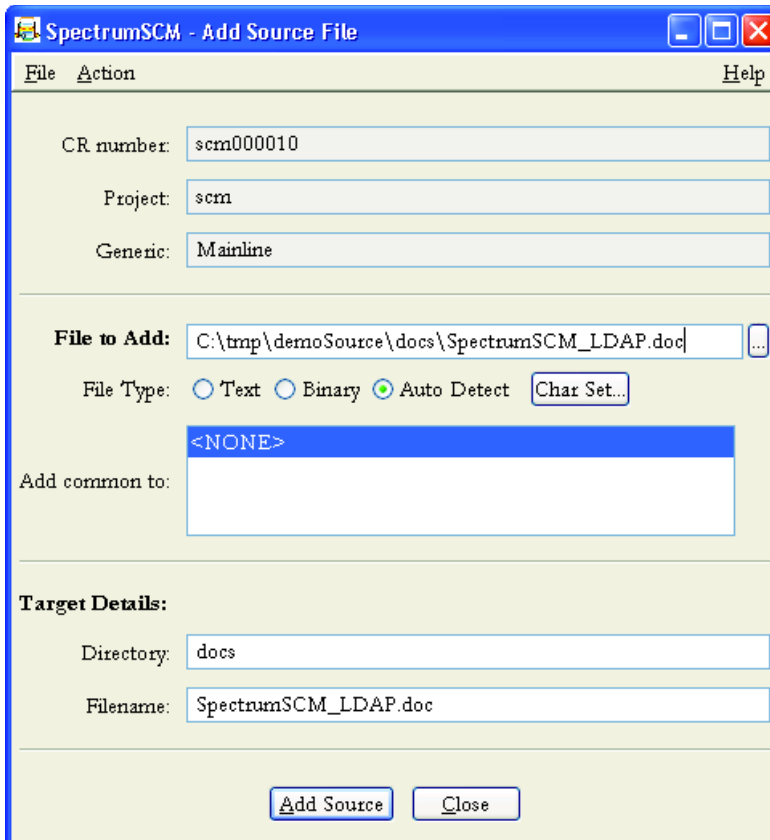
8.1.1 Check-In a new file

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.

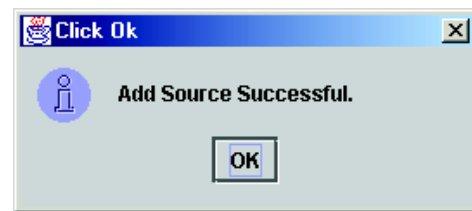


Click the file browser (. . .) and select the file to add. It must be in the specified local directory, under a file structure similar to that in SpectrumSCM. For example, to add the SpectrumSCM_LDAP document to the system. They are in the local directory C:\tmp\demoSource\docs folder. With the local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

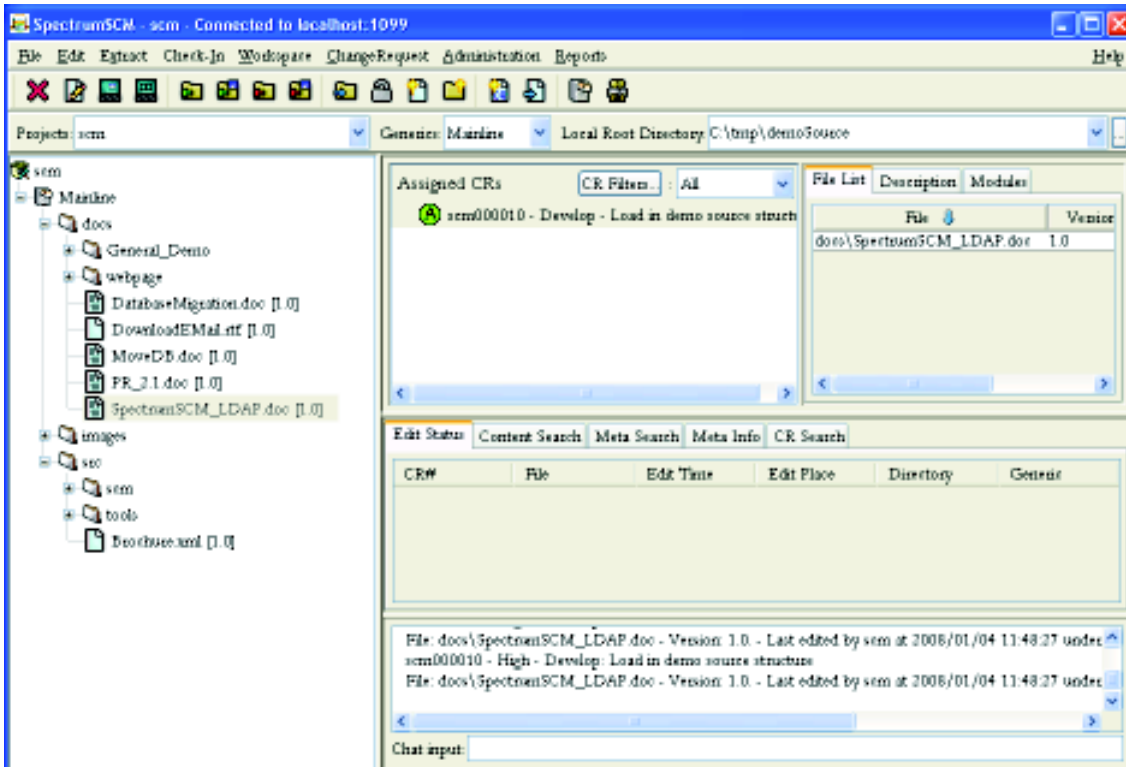
Select <NONE>, or the generic(s) to make this new file common to. If you want to add this file common to a number of generics, you can multi-select them from the list using the *shift* key.

Click Add Source.



A confirmation is displayed

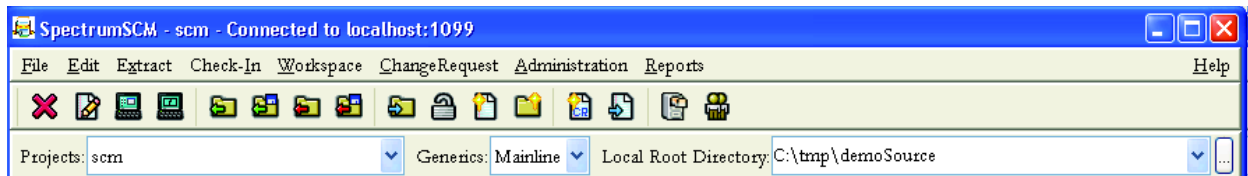
Multiple files associated with the same CR can be added, either one at a time using the **Add Source** button, or by multi-selecting the files through the browse button. To verify that the source file was added correctly and associated with the desired generic and directory, refresh the UI with updated 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.



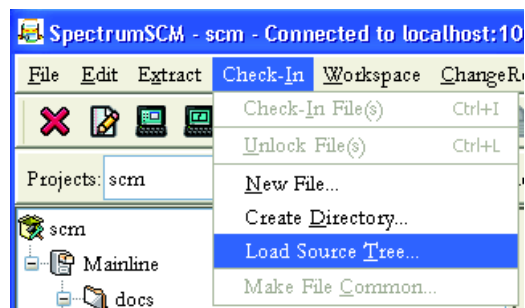
8.1.2 Load a Source Tree or an entire directory into SpectrumSCM

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).

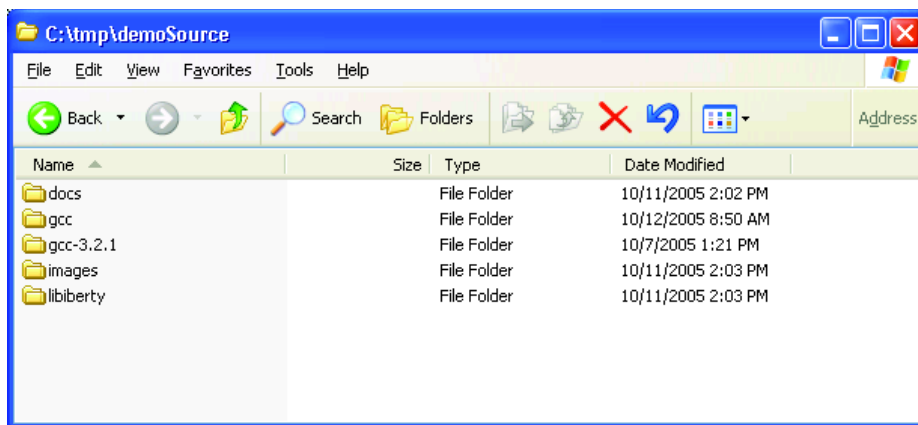


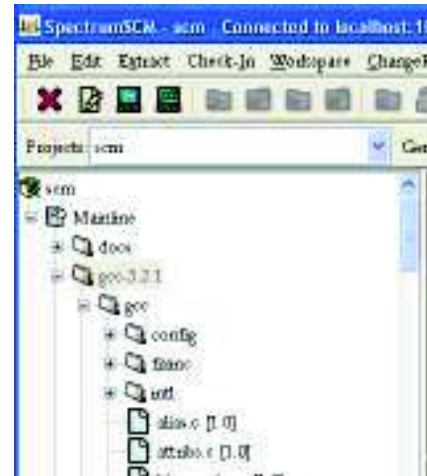
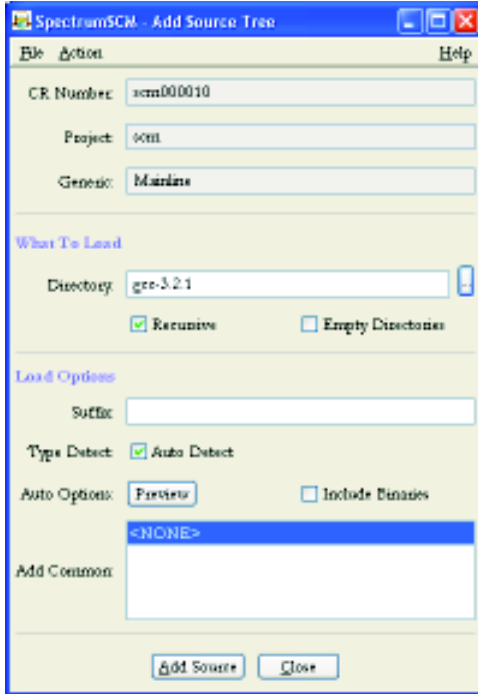
Then access the Add Source Tree screen via the **Check-In / Load Source Tree** menu option.



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.

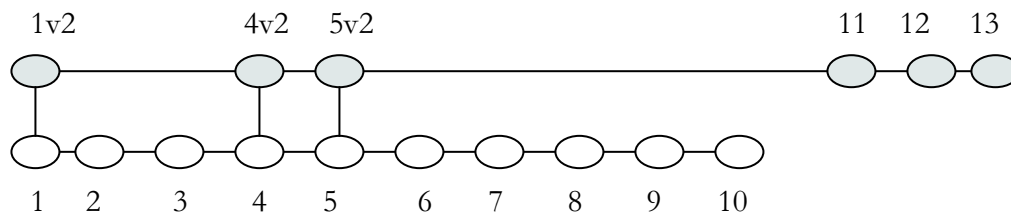
- **Common:** Versioned files that are physically the same across generics
- **Uncommon:** The act of physically separating versioned files from multiple generics

Checking out "common" is a powerful feature since it can be used to apply a single "fix" to multiple generics in one edit.

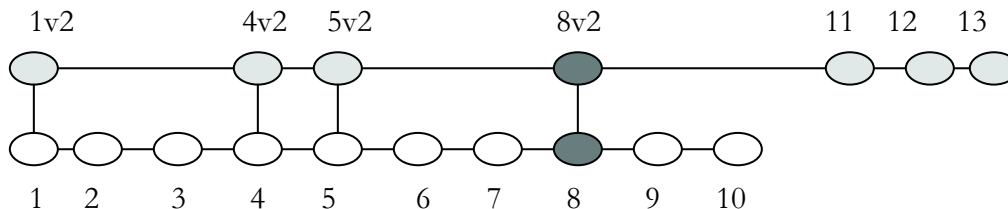
For a new development effort, the default check-out mode (uncommon) is all that is required and the developer need not be concerned about these issues. Checking out "uncommon" will mean any file changes will only be made against that specific generic.

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.

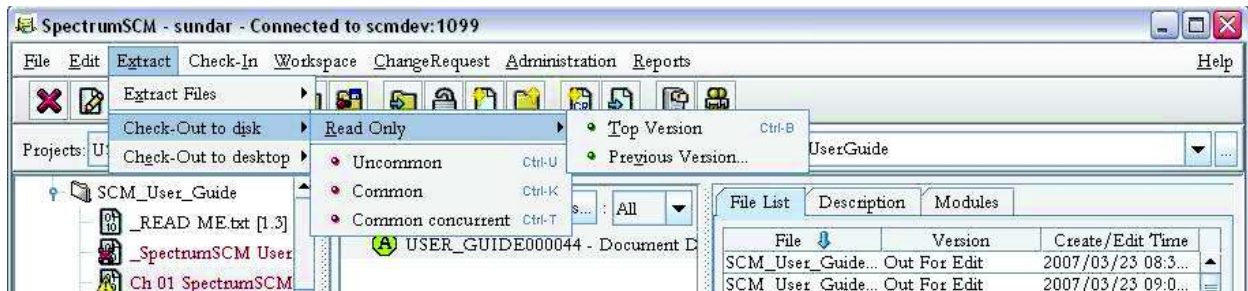
Once the mode of the Generic has been established, if the commonality control is still with the developer (the generic is "unlocked"), then he/she can choose the appropriate check-out mode.

8.2.2 Checking out for read-only to desktop or disk

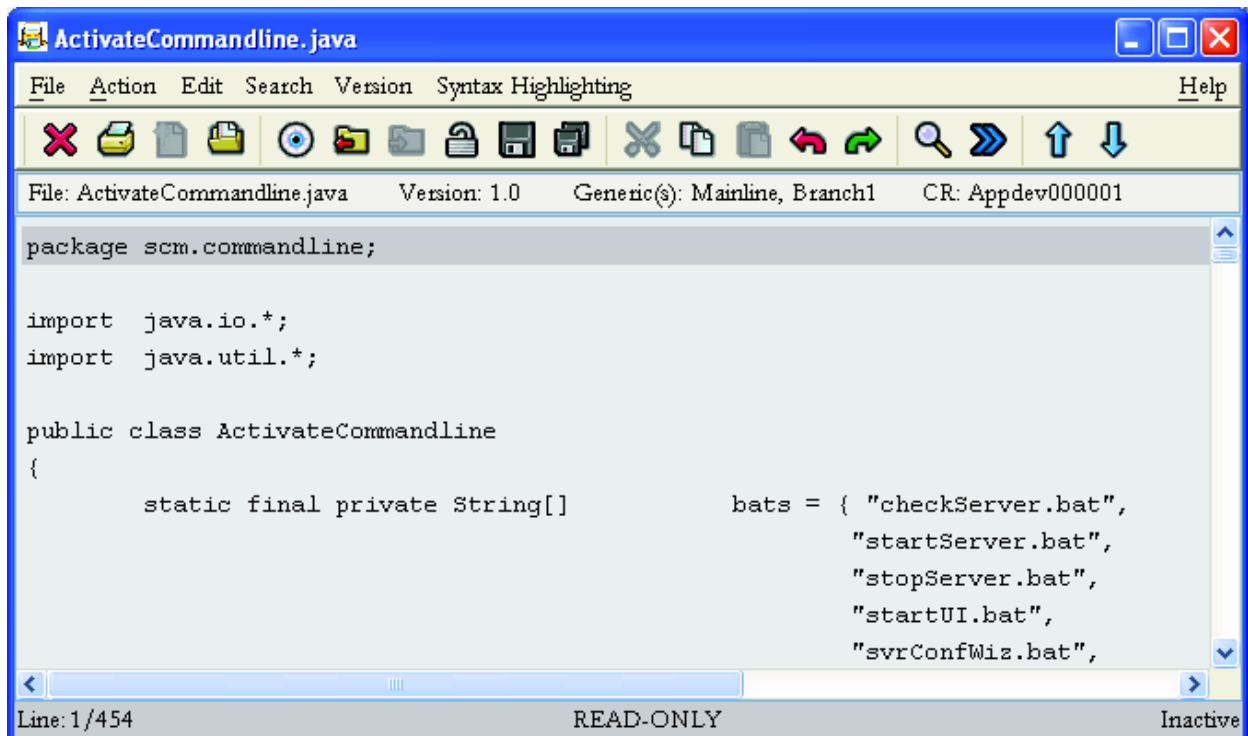
If you select a file without selecting a CR, read-only is your only option.

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.

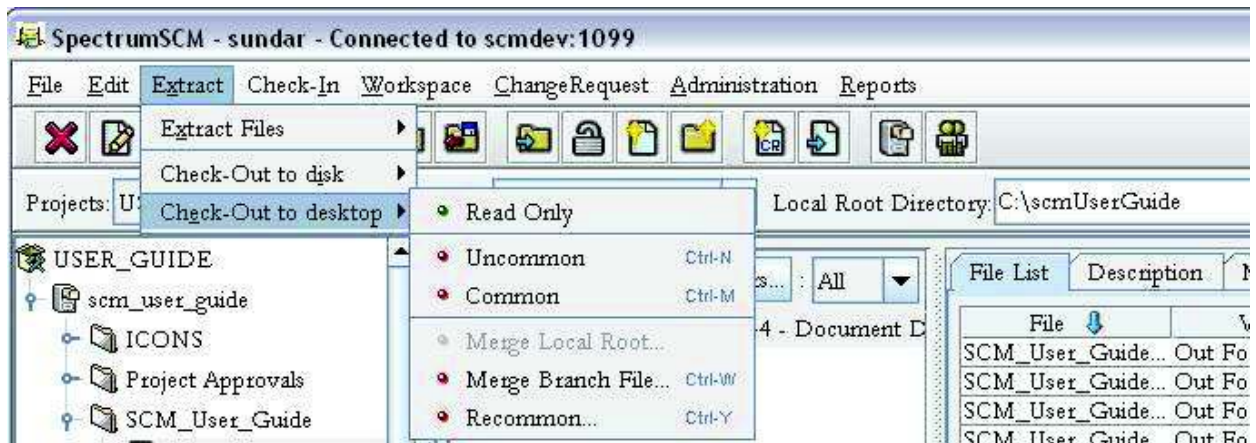


A file checked out read-only to disk is placed into the local directory and is marked as read-only. A file checked out read only to the desktop is brought up in the SpectrumSCM (or custom) editor. You can search the file, copy it, save it to the hard drive, and pull up another version. The file cannot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

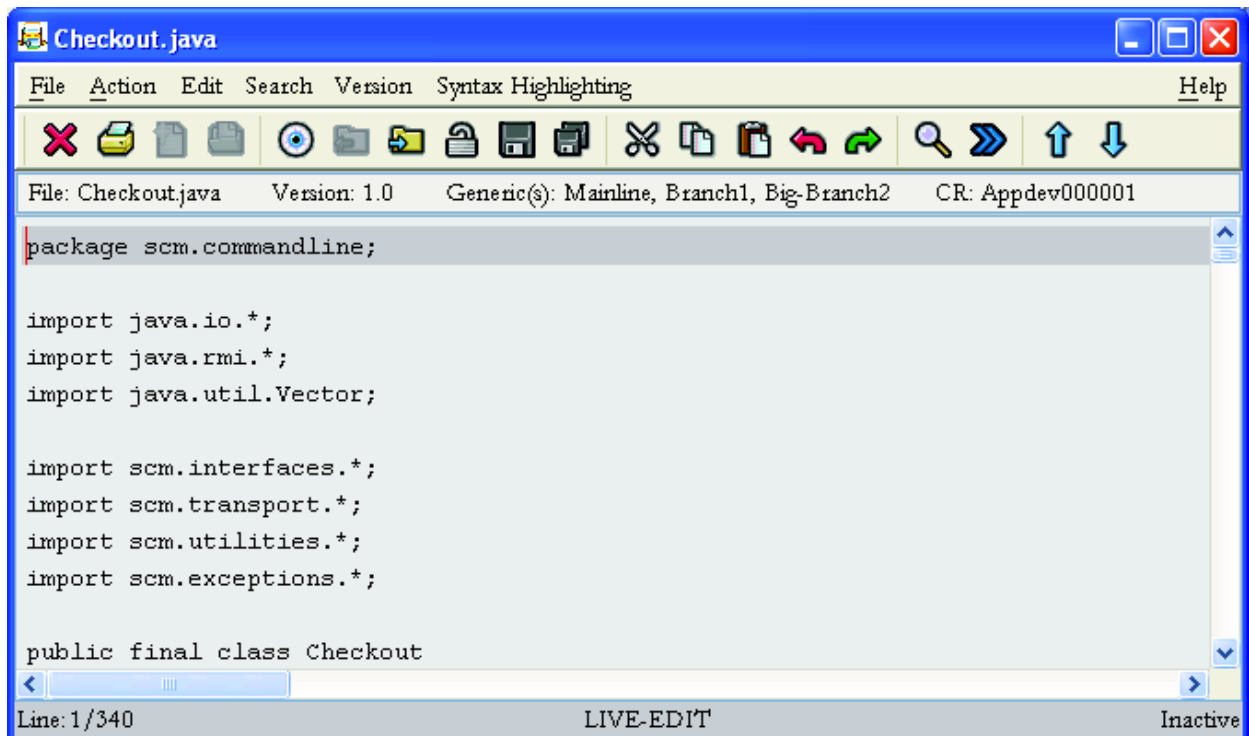


8.2.3 Check out to desktop for edit

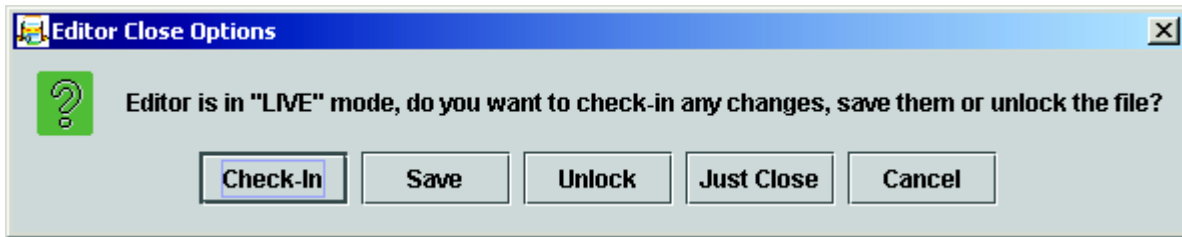
- Select **Checkout to desktop** -> **Common** or **Uncommon** to put the file in a write-able state in the editor. The file to be checked out and a CR must be highlighted on the Main Screen



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



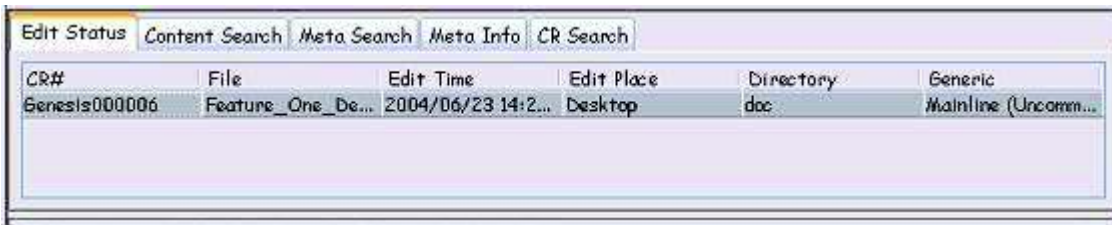
On closing the edit session, the system will present the user with options to check in, save to disk, unlock, or close.



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

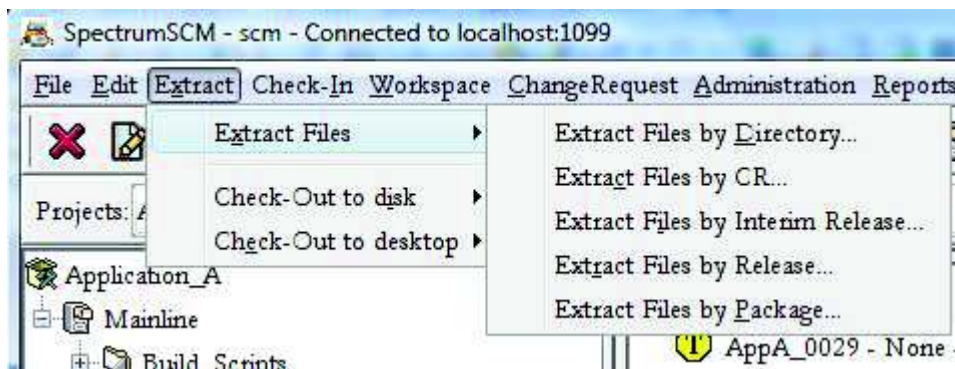
Select **Extract / Check out to disk / Common, Uncommon** or **Common Concurrent** to put the file in a write-able state on the local disk in the specified local root directory.

- 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.

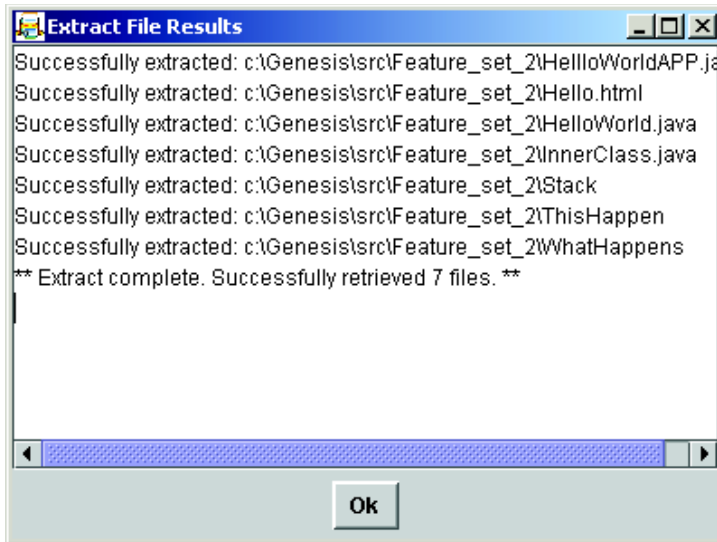
Extract -> Extract Files by Directory.

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.

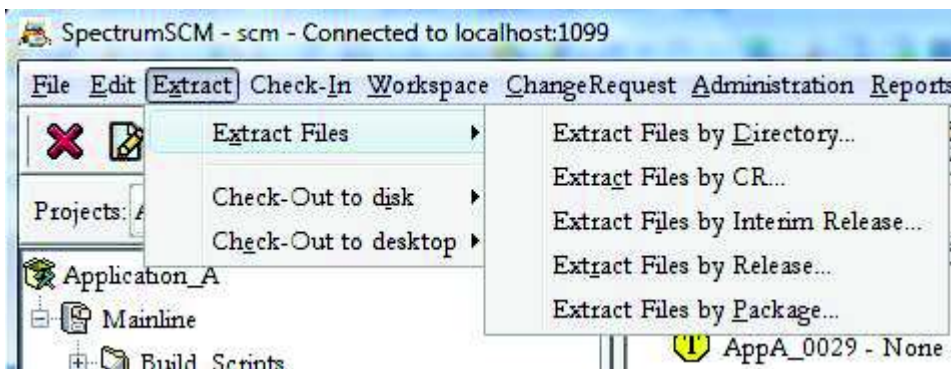


Files are extracted and placed into the user's local root directory.



8.2.8 Extract Files by Release

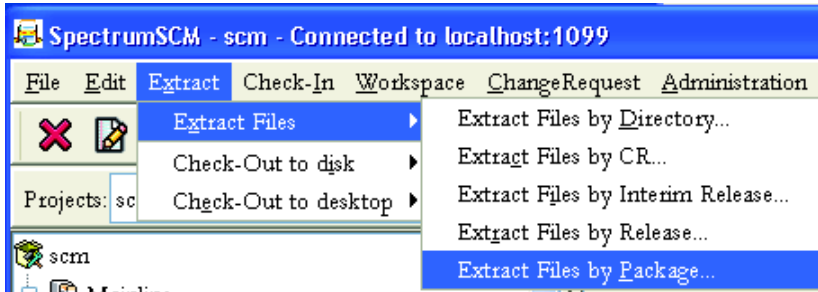
This option can be used if there is a previous release of the project (*See Creating a Release in Chapter 9*). The entire project tree associated with a release is extracted and placed into the local root directory.



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

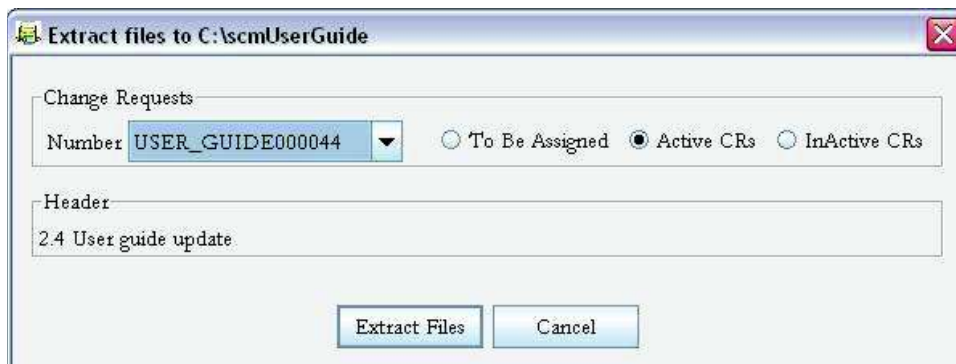
You use **Interim Releases** to extract a set of files based on a phase of your life-cycle. This feature is intend to allow you to form “test” releases where many but not all issues have been resolved, and so a formal release cannot be built yet. For example, if you have an “Integration Test” life-cycle 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.

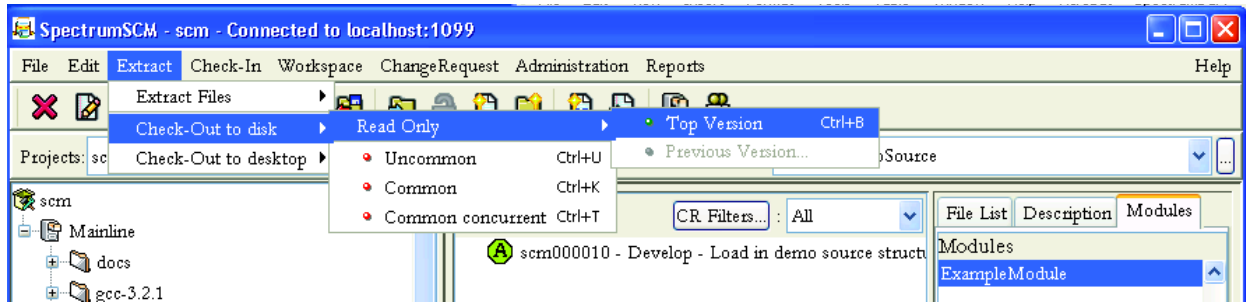
Select a CR with which the work is associated

Select a module from the module list

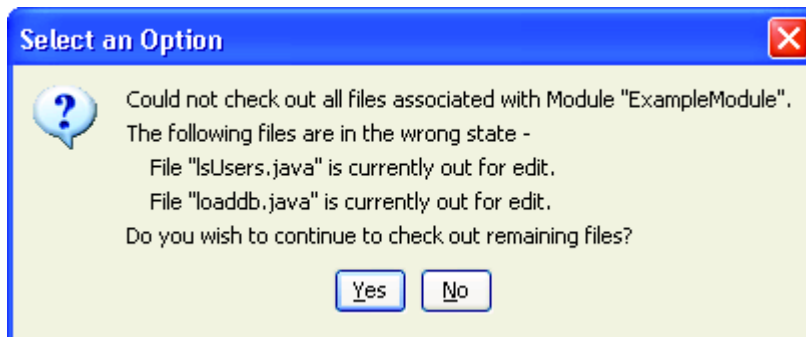
Select a function

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**.

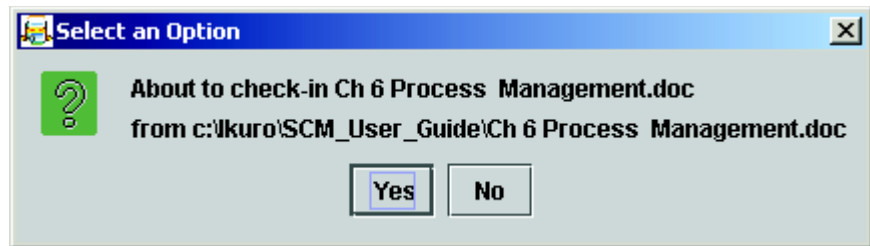
All files selected for check in will be associated with the CRs that were used to check the files out.

CR#	File	Edit Time	Edit Place	Directory	Generic
USER_GUIDE...	_SpectrumSCM ...	2007/03/23 08:...	C:\scmUserGui...	SCM_User_Guide	scm_user_guide ...
USER_GUIDE	Ch 06 Process	2007/03/23 09:...	C:\scmUserGui...	SCM_User_Guide	scm_user_guide ...
USER_GUIDE	Open With ...	/23 09:...	C:\scmUserGui...	SCM_User_Guide	scm_user_guide ...
USER_GUIDE	Check-In	/23 09:...	C:\scmUserGui...	SCM_User_Guide	scm_user_guide ...

File: Ch 01 Sp	Version: 1.12. - Is out for edit by sundar since 2007/03/23 09:03:0
File: _READ	ed by adrian at 2005/04/05 16:13:39 under CR USER_GUIDE00
File: _READ	11/11/2005/04/05 16:13:39 1. CR USER_GUIDE00

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

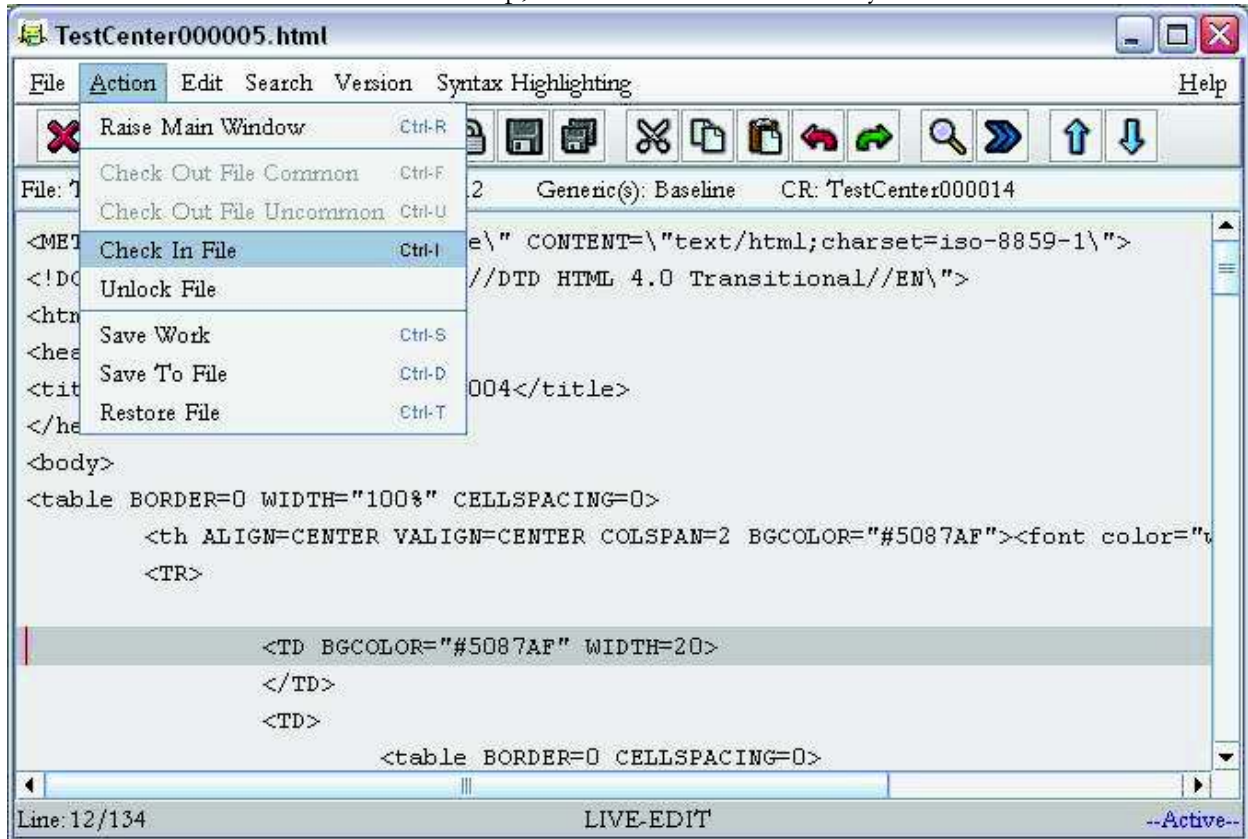
If this is correct, click **Yes**.



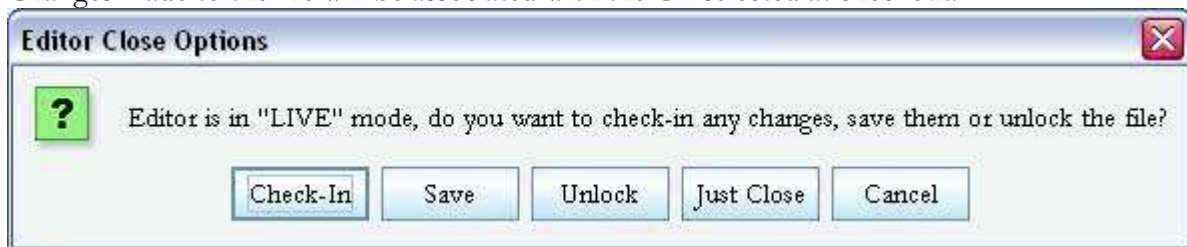
You will receive a confirmation screen when the process is complete. The completion message will show that a new version of the file has been created. If no changes were made, the file will be unlocked and no new version will be created.

8.3.2 Checking in a file from the desktop

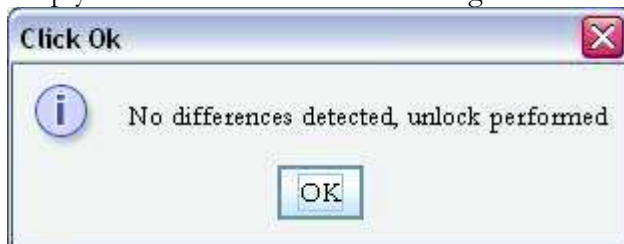
If the file was checked out to the desktop, it can be checked in directly from the editor.



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.



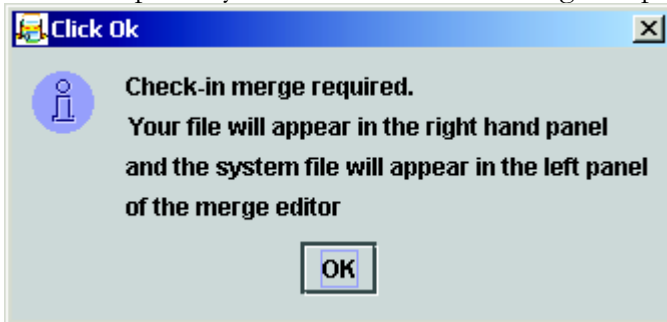
If you attempt to check in a file and there have been no changes made, the system will alert you and simply unlock the file without creating a new version.



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.

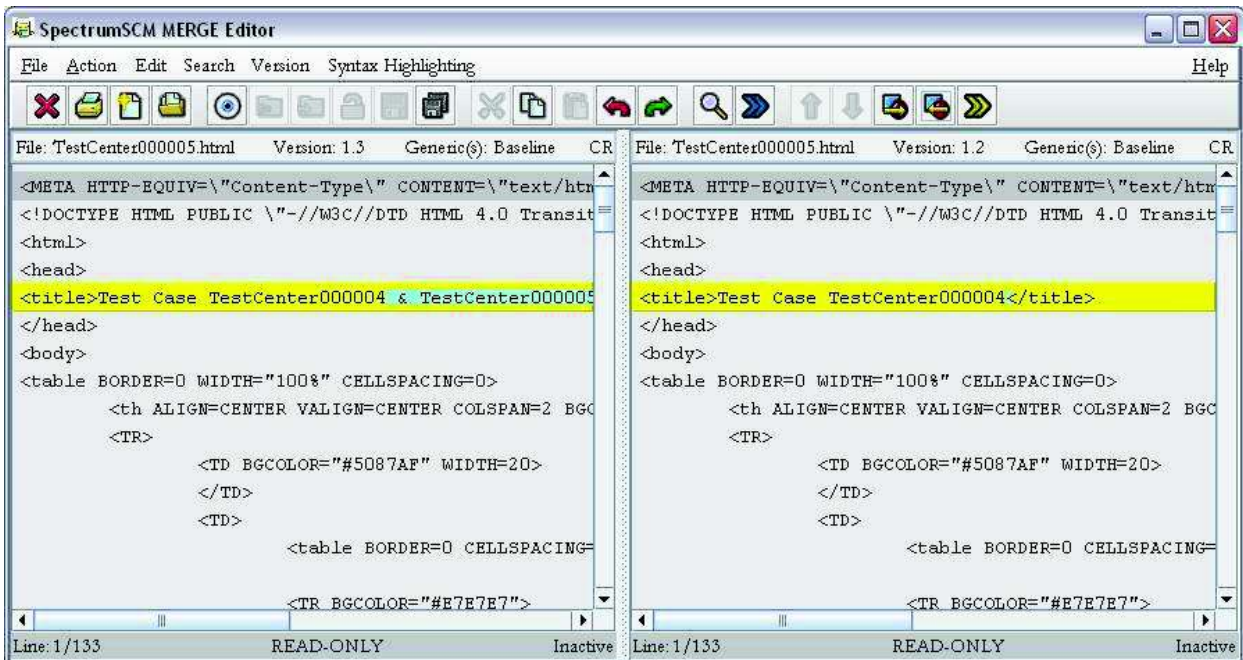
If there have been modifications, then a merge operation is required to merge in your changes with the new repository version. This is done using the split screen merge editor.




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.

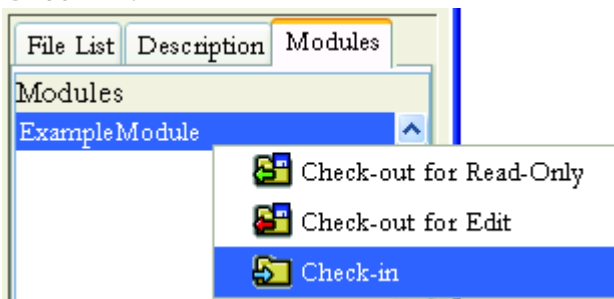


Selecting the right most editor button color codes changes to the files based on what has to happen to the right hand editor to make it look like the left hand editor. Selecting the left most diff button color codes the left hand editor so that applied changes will make it look like the right hand editor. Think of it this way, what do I have to do to this edit session to make it look like the other edit session?

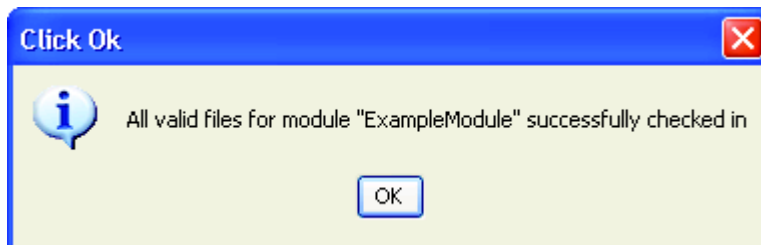
Once merging is complete, check the file back in and a new version of the file will be generated. If there are a number of differences that you are sure all need to be applied, rather than going through and right-clicking each one, you can go to the **Action -> Apply all diff marks** menu option. Make sure that this is really what 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**.

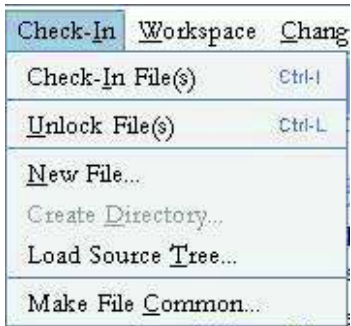


All files in the module that were modified will be checked in and associated with the CR selected when they were checked out. Files that were not changed will simply be unlocked, a new version is not created.



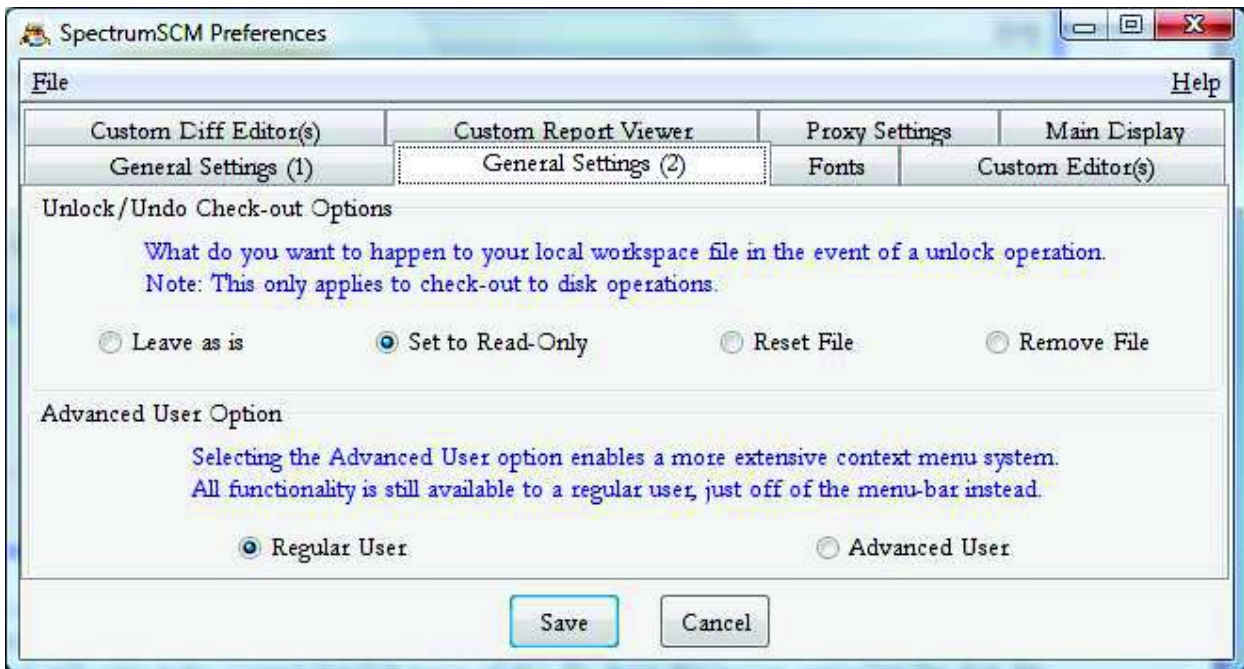
8.3.5 Unlocking a file from Edit

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.

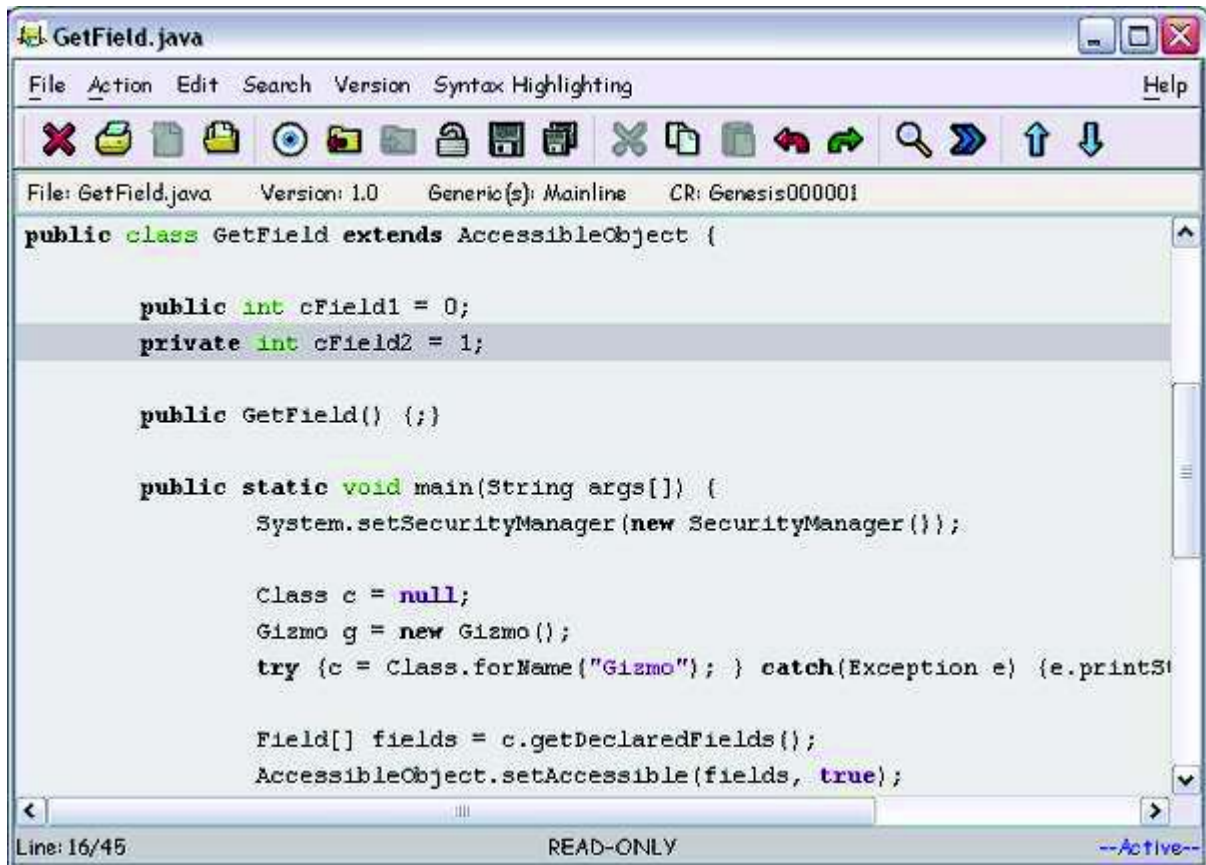
“Reset File” will extract the current head revision of this file from the server, replacing the disk file and its contents.

“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

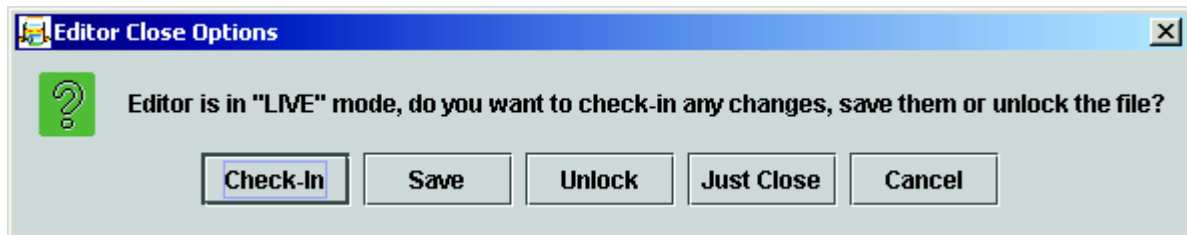
Next – Move to the next most recent version of this file

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.

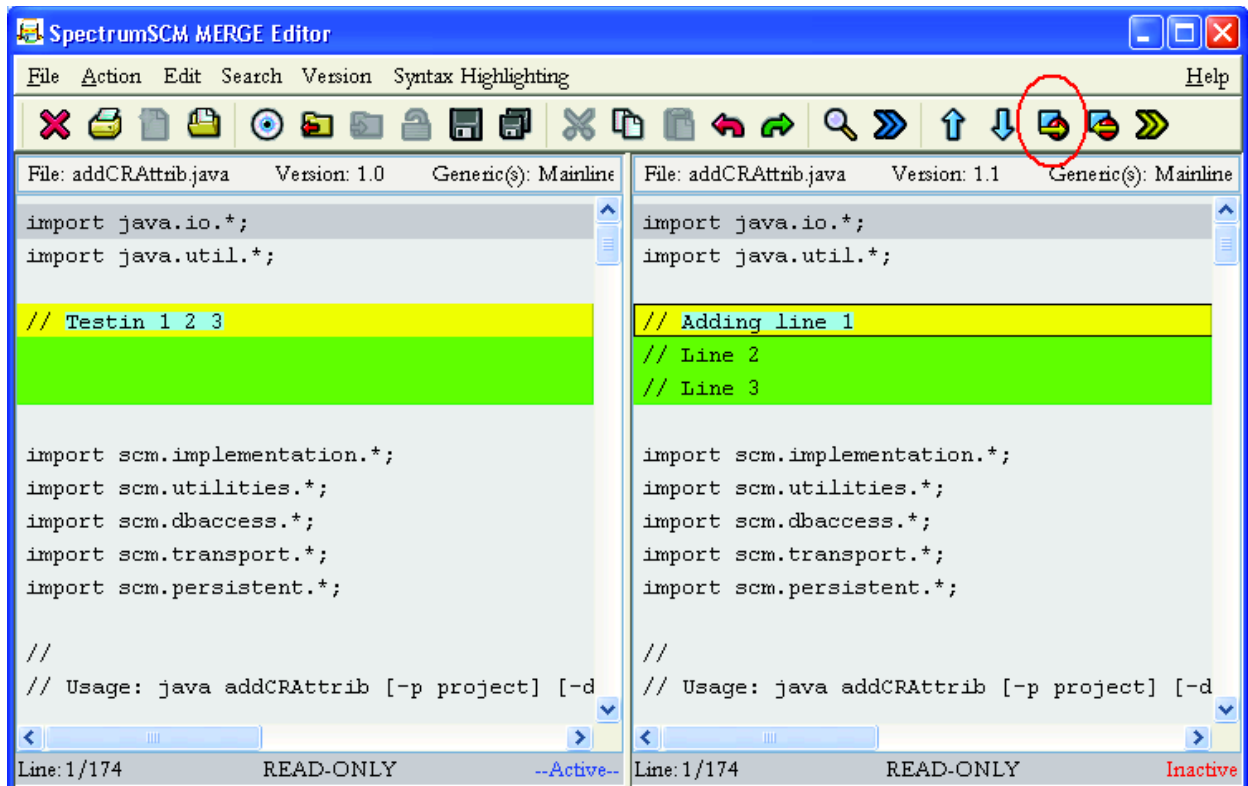
The **Just Close** option throws out the edit changes to the file.

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.

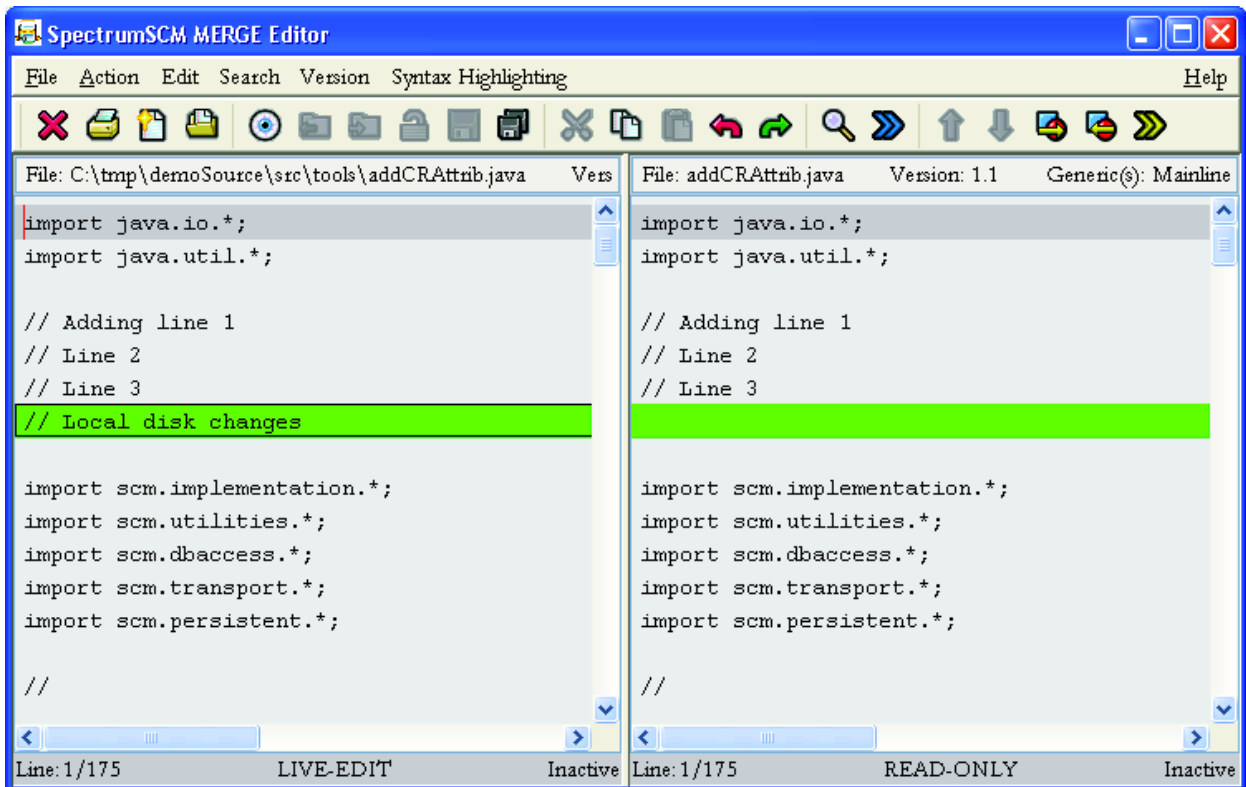
The difference action can also be triggered off of the Action menu, either “Diff left to right” (i.e. what has to be done to the left panel to make it look like the right), or “Diff right to left”.



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.



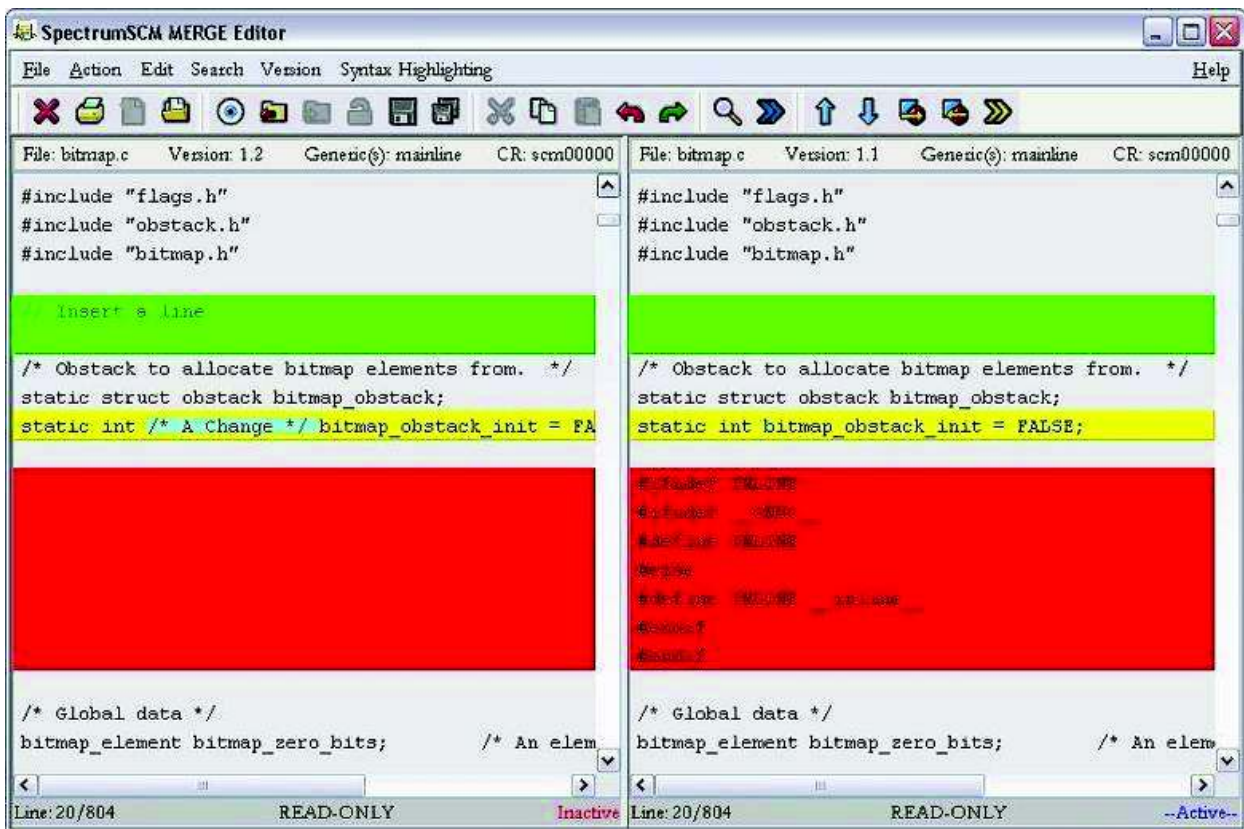
The   icons can be used to display a previous or later version of the file. The   icons can be used to copy selected source to and from the clipboard.


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 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 what 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).

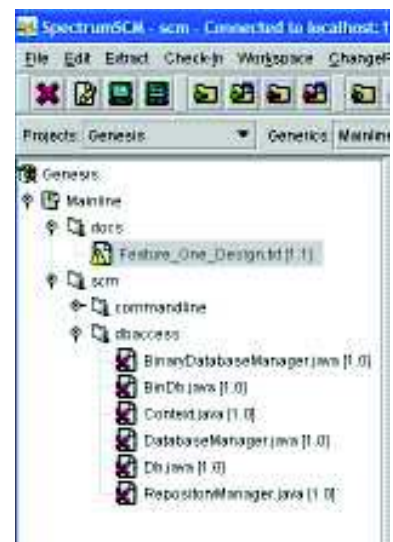
8.6 Workspace Analysis and Synchronization


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

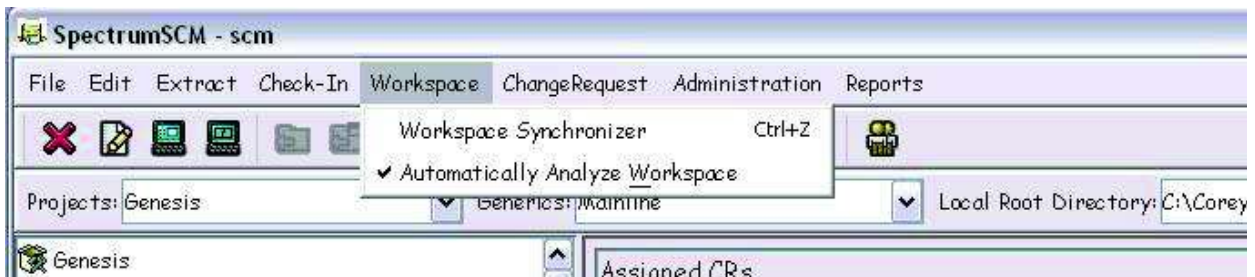


Missing  This icon tells the user that the file is missing from the users local workspace. Users can resolve workspace synchronization issues in several ways. Out of sync or missing files can be manually downloaded simply by right clicking the file and using the context sensitive menu to extract the 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.

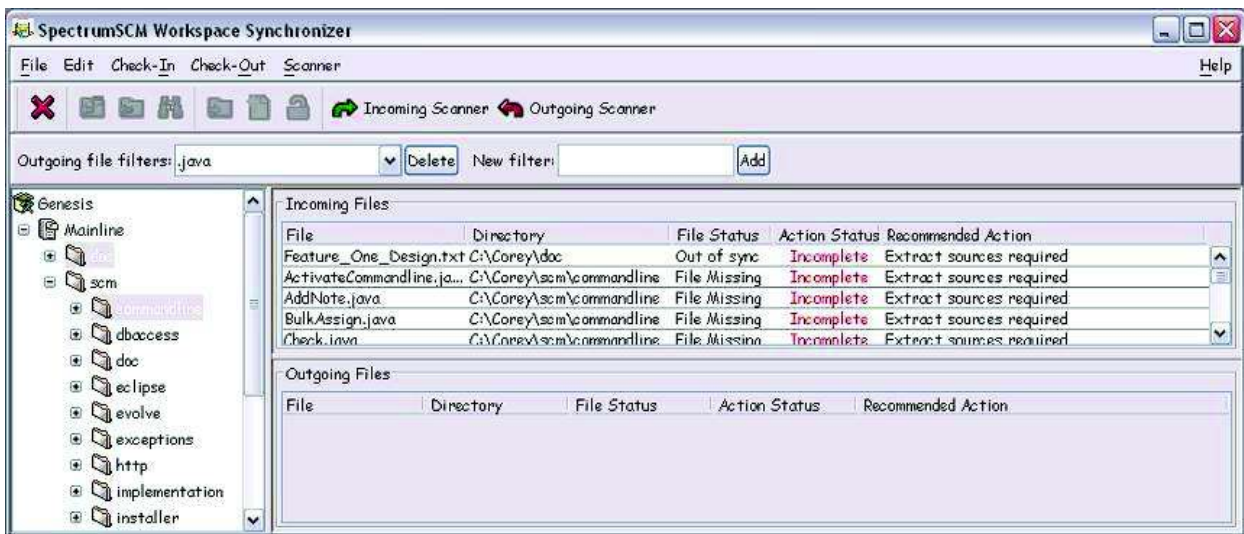
The Workspace Analyzer is responsible for adding the icon decorations outlined above to the regular file icons in the tree view. The Analyzer runs automatically when a user expands the tree view for a directory or when a project level refresh is activated. Project level refreshes can be started either 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

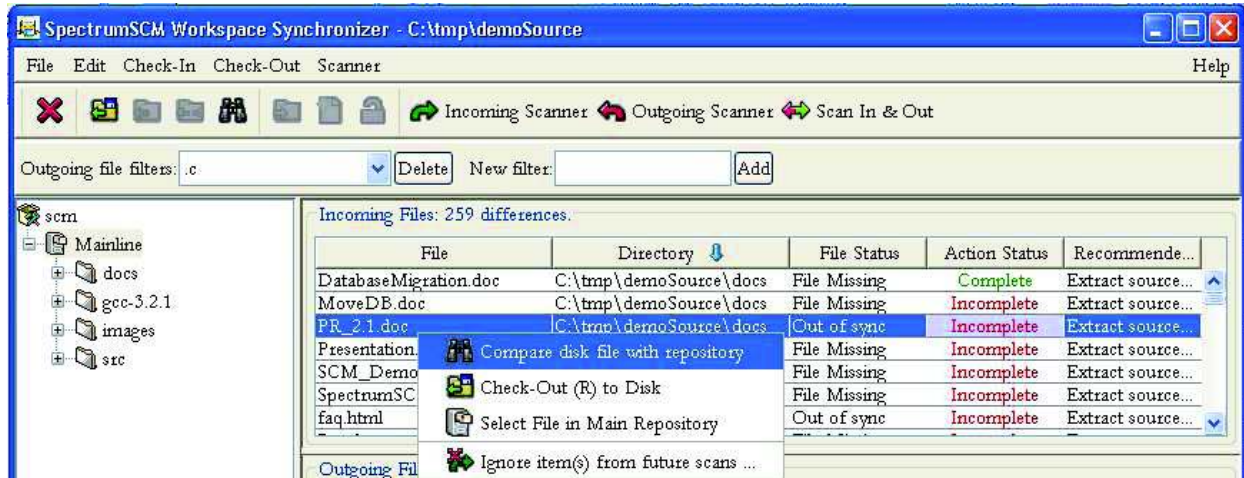
The Workspace Synchronizer is a separate screen that can be used to detect file synchronization issues and can be used to act upon out of sync situations.



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

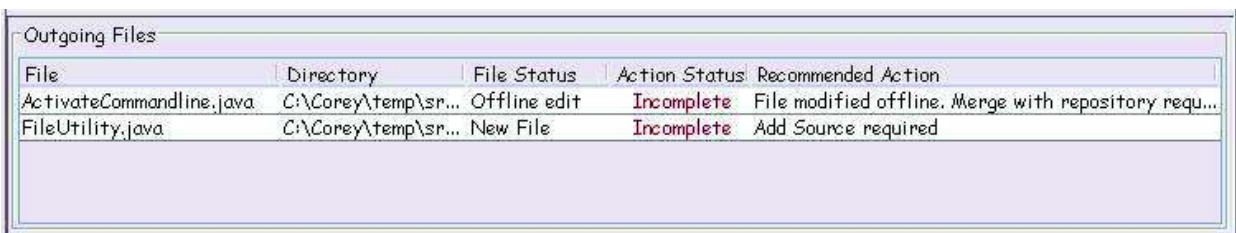


Checking out to disk read-only will overwrite the local file image with the version of the file from the repository.

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 be 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:

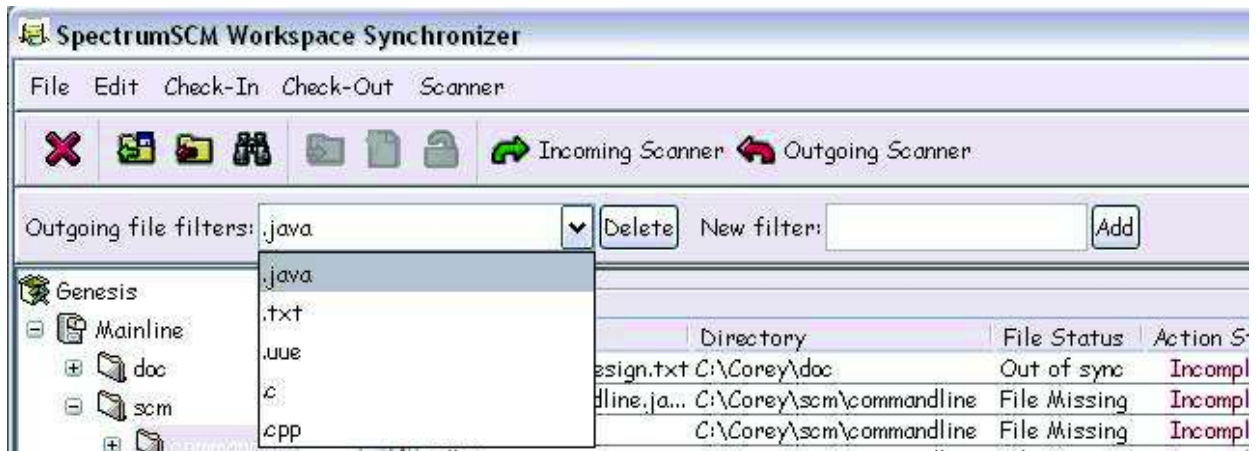
These actions can all be accessed via the context sensitive menu associated with right clicking on the files. Most actions can be applied to multiple files at the same time, but some actions can only be applied to a single file at a time (Compare).



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.



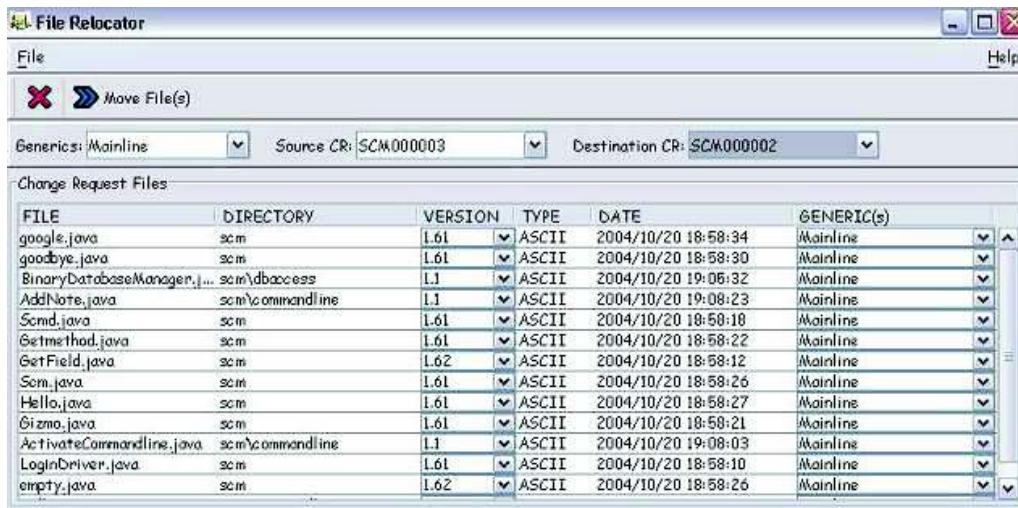
In this case the Outgoing file filter is set to operate on 5 separate file types. New types can be added by simply entering a file type extension into the New Filter section and either hitting Enter or by pressing the “Add” button. File types can be removed just as easily by selecting the type to remove 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.

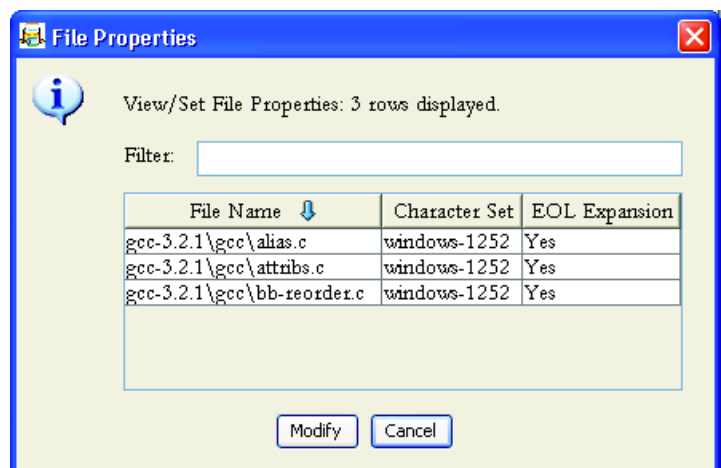
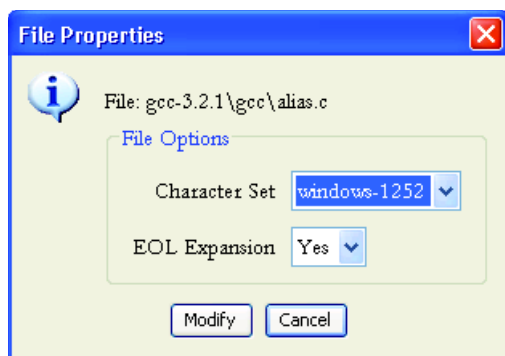
Once the desired file versions have been selected, then you can select all the files that you want to move, and press the “**Move File(s)**” button. Note this is also available on the right-mouse click context menu too.

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 responsible 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 2008/01/07 gcc-3.2.1\gcc\attribs.c: Character Set option
13:23:53 13:23:53 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.

