CA Harvest Software Change Manager

Workbench User Guide

Release 12.5

CA Technologies Product References

This documentation set references the following CA Technologies products:

- CA Harvest Software Change Manager (CA Harvest SCM)
- Unicenter Software Delivery
- CA IT Client Automation (formerly, CA Desktop and Server Management, CA IT Client Manager)
- CA Clarity Agile
- CA Clarity Requirements

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

- Project Dashboard Reports—Added this section to describe the project dashboard reports feature.
- Configure Project Dashboard (see page 251)—Added this section to describe the project dashboard reports feature.
- Package Distribution by State Report (see page 253)—Added this section to describe the project dashboard reports feature.
- Item/Versions Distribution by State Report (see page 253)—Added this section to describe the project dashboard reports feature.
- Version Change Activity Report (see page 257)—Added this section to describe the project dashboard reports feature.
- User Modified Items Report (see page 254)—Added this section to describe the project dashboard reports feature.
- Peer Review Report (see page 255)—Added this section to describe the project dashboard reports feature.
- Custom Reports (see page 258)—Added this section to describe the project dashboard reports feature.
- Delete Version Rules (see page 153)—Updated this section to include new option describe the project dashboard reports feature.
- Execute a User-Defined Process (see page 221)—Updated this section to include the security hole in UDP parameter feature information.
- Find Versions (see page 234)—Updated this section to include the information about the find latest versions based on the package group feature.
- Examples (see page 237)—Added this section to describe the examples for finding the latest versions based on the package group feature.
- Locate Package in the Explorer View (see page 100)—Added this section to describe the project dashboard reports feature.
- Generate Customized Project Dashboard Reports (see page 260)—Added this section to describe the project dashboard reports feature.
- View Package History (see page 79)—Updated this section to include the addition of approval notes in the package history feature information.
- Connect to a Remote Agent (see page 44)—Updated this section to include the enhanced agent checkout option information.
- Check Out Versions (see page 110)—Updated this section to include the enhanced agent checkout option information.

- Workbench Arguments (see page 28)—Added this section to describe the workbench arguments.
- Find Version - Results (see page 237)—Added this section to describe the additional data size column information.
- Locate Package From Form List (see page 244)—Added this section to describe the information about locating a package from form list.
- Filter the Synchronizer View (see page 143)—Updated this section to include the reset synchronizer view information.
- Set BIRT Preferences (see page 56)—Added this section to describe the information about setting BIRT preferences.
- BIRT Report Viewer (see page 261)—Added this section to describe the information about setting BIRT report viewer.
- Run Report (see page 261)—Added this section to describe the information about BIRT reports.
- Run and Export Report (see page 262)—Added this section to describe the information about BIRT reports.
- Broker Dashboard Report (see page 249)—Added this section to describe the broker dashboard reports feature.
- Configure Broker Dashboard (see page 250)—Added this section to describe the broker dashboard reports feature.
-

Contents

Chapter 9: Working in the WorkAreas 123

Chapter 10: Managing Versions, Items, and Paths 149

Chapter 1: Introduction

This section contains the following topics:

What You Need to Know

The Workbench is a graphical user interface that provides you with access to all CA Harvest SCM user functions. To use the Workbench, you need a working knowledge of CA Harvest SCM components and processes. An understanding of basic CA Harvest SCM concepts and of configuration management (managing changes with the help of processes and tested methods to avoid new errors and minimize the impact of changes) is recommended but not required.

The administrative tasks that the Workbench requires include initializing the physical repository, defining the software development lifecycle, setting up user groups, and numerous other procedures. An administrator performs most administrative tasks during the installation and configuration of CA Harvest SCM.

Note: For information about administrative tasks, see the *Web Interface Administrator Help* and the *Administrator Guide*.

Intended Audience [BO Reports]

The following users install, upgrade, and configure CA Harvest SCM Reports:

- System administrators use the information in this guide and their operating system knowledge, to install the product for the first time, upgrade the product from release to release, and configure the product based on your implementation requirements.
- Database administrators use the information in this guide to install the database on which CA Harvest SCM Reporting runs, and configure the database for best performance.

The following persons use CA Harvest SCM Reports:

- CA Harvest SCM Administrators use the reports to measure performance. After you establish a change management process, it is important to monitor and measure the process over time to achieve the goal of continuous process improvement. Indicators that you determine help you measure processes. Over time the indicators can show as trends in reports and managers can determine if a process is becoming better or worse and take appropriate action. For example, the Projects with Activity Summary report displays a list of all projects with activity events during a selected time frame.
- Project Managers use project and package-related reports to monitor their project packages. For example, the Packages Approved report displays a list of packages that have successfully completed the Approval Process defined in the current state for the selected projects, providing basic details of each package.
- Developers use package and source-related reports to help manage their changes. For example, the Items Reserved Summary by User displays a summary of the number of reserved items for the selected user, so developers can quickly determine the items that they have checked out.

Chapter 2: Understanding Basics

This section contains the following topics:

Basic Methodology

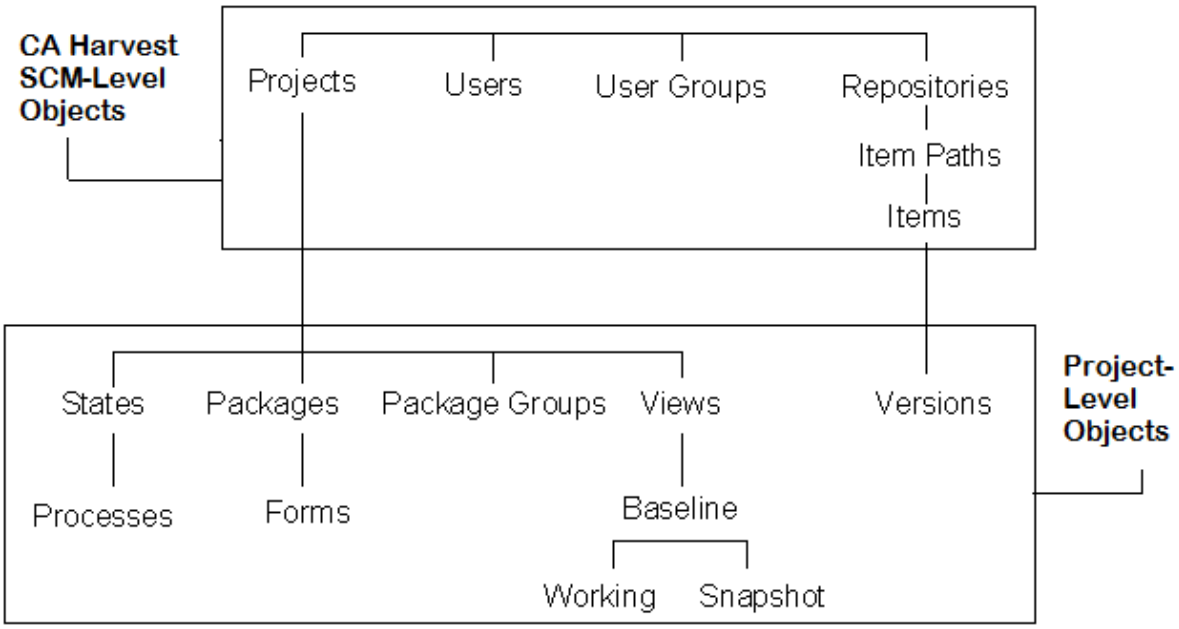
CA Harvest SCM provides you with the tools that you need to maintain a release or multiple releases of the same application. For example, you can work in a project for emergency maintenance release and keep those changes separate from other ongoing changes in another release. The integrated merge facility lets you automate the merging of some or all of those changes into a subsequent development process, thus eliminating labor-intensive manual merges. You can develop both short-term and long-term projects in parallel, knowing that they will not affect each other.

The concurrent development feature of CA Harvest SCM allows more than one developer to work on the same area of code simultaneously without fear of overwriting another's changes. Merge utilities are used to resolve any conflicts between the versions.

Objects

Key objects in CA Harvest SCM include repositories, items, versions, projects, states, views, processes, users, user groups, packages, package groups, and forms. Icons on the Workbench represent particular object types. Each object type represents different data and has specific actions that can be applied to it.

The following diagram shows the highest-level and project-level object types, and their hierarchy, in CA Harvest SCM:



SCM-Level Objects

SCM-level objects are *global*, meaning that they are available to your entire CA Harvest SCM installation.

Note: Forms are global objects; however, forms must always be associated with a package.

The following objects are SCM-level objects:

project

A *project* refers to the entire management framework in CA Harvest SCM that supports a particular development or maintenance activity. You can have many different projects, depending on the applications being controlled and the kind of development activity. A project includes information about how changes progress through the development life cycle, the activities that can occur, what data to access, and user responsibilities.

Note: In previous releases of CA Harvest SCM, the term *environment* was used for *project*; these terms are synonymous.

Because the following objects are highest-level objects, you can use them in one or more projects defined in a CA Harvest SCM installation:

user and user group

A *user* is an individual that is defined in CA Harvest SCM. A *user group* is an association of one or more users. A user can belong to several user groups, and the groups imply no hierarchy. For example, a user in the Development Manager group does not necessarily belong to the Developer group. Two predefined user groups, Public and Administrator, are provided in CA Harvest SCM.

repository

A *repository* is a collection of items and item paths. When CA Harvest SCM is first installed, no repositories exist in the installation. Administrators create repositories and control access to them.

item path

An *item path* is the object that is created when a directory (or Windows folder) is first loaded or checked into a CA Harvest SCM repository. Subsequent changes to the item path (rename, move, or remove) will result in new versions of the item path.

item

An *item* is the object that is created when a file is first loaded or checked in to a CA Harvest SCM repository. Subsequent changes to the item will result in new versions of the item. Repositories are composed of items.

form

CA Harvest SCM *forms* record, organize, and track information. Forms are always associated with packages. This association helps you track problems or issues and changes you made to address them. The information recorded in a form is directly accessible from the associated package.

Project-Level Objects

Project-level objects are specific to a single project. The following objects are specific to a single CA Harvest SCM project:

state

A *state* is a distinct phase in the life cycle in which certain activities can take place as packages move from definition to completion. A life cycle can include several states.

process

A *process* is an action that can be executed on an object in CA Harvest SCM. The processes defined for a state determine what activities can be performed in that state.

package

A *package* is the basic unit of work that moves through a life cycle. It typically represents a problem or a request that needs to be tracked, the changes made in response to the problem or the request, and any other associated information. The package is the user's link to the actual data accessed during the change process. Each package resides in a particular state of the life cycle.

package group

A *package group* is one or more packages, which are usually related. A package can belong to one group, several groups, or no groups. Any operation that can be performed on a package can be performed on all packages in a package group.

baseline

A *baseline* defines the repository items available to a project. A baseline can include items from more than one repository, as in the case of shared code.

view

A *view* determines what items specific users or user groups can access and view in repositories. A state is usually associated with a view. The view determines which item versions are accessible to users operating in a state using that view. Administrators can define a state without a view, if users do not need access to data items. For example, an initial state named Open, which is simply a holding area for newly created change requests would not need a view.

working view

A *working view* defines where users can access specific items. Administrators associate working views with one or more states. Several states can share the same working view, but a state cannot use more than one working view. The working view determines which item versions are accessible to users.

snapshot view

A *snapshot view* is a read-only image of a working view at a specific point in time. Snapshots let you capture a data inventory at significant points in the software's development, such as a release. After you create a snapshot, you can use it to create a baseline.

version

A *version* is an iteration of an item. Each item in a repository consists of versions of the item. Versions are visible to users only through views. The views for a state in the Workbench display versions in the project baseline view and the state's working view.

More information:

Views and Versions (see page 100)

Lifecycles

A lifecycle describes the path that changes take as development progresses in terms of an ordered set of states. In each state, processes define the various activities that can occur. This means that each state can have a uniquely defined scope of work. The demote and promote processes have special significance in a lifecycle because they determine how states are related and how changes can move between them. Changes travel through the lifecycle by means of packages. You can link Notify and UDP processes to the promote and demote processes so that the linked processes execute either before or after the package is moved. Data views (baseline, working views, and snapshot views) determine which item versions are accessible to users, and let administrators capture software inventories.

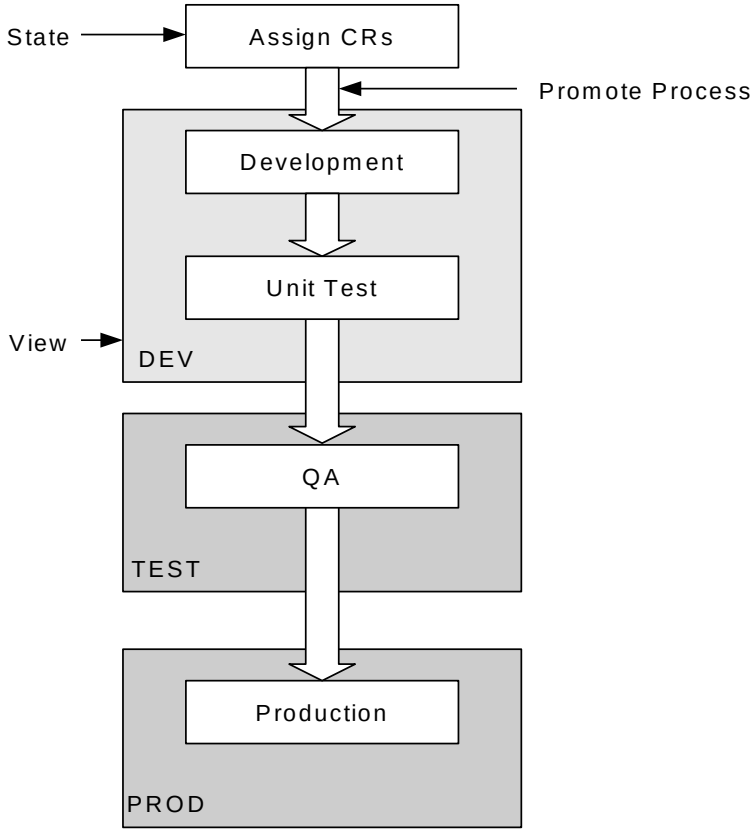
How a Sample Life Cycle Works

The following sample life cycle has these states:

1. Assign CRs (Change Requests)—Administrators create packages to fix change requests.
2. Development—Developers create and modify code.
3. Unit Test—Developers test code.
4. QA—Quality assurance personnel test the new code.
5. Production—Code is held for a final release.

The descending arrows in the sample life cycle indicate promote processes that define how packages move from one state to another. The states and promote processes combine to form a road map through which software changes travel on the way to a software release. The vehicle by which the changes travel is a package. A package typically represents a modification or an enhancement to a software application.

The following diagram shows the sample CA Harvest SCM life cycle:



The sample life cycle works as follows:

1. A team member creates a package in Assign CRs. A view is not associated with the Assign CRs state, because no activities in this state affect actual data. The team member promotes the package to Development.
2. In Development, a developer uses the package to modify versions of items in the DEV view. The developer promotes the package to Unit Test. The developer unit tests the package and promotes it to QA.

   The changes made in the DEV view become visible in the TEST view when the package is promoted to the QA state because it is associated with the TEST view. When the package enters QA, the TEST view is updated with the package changes.
3. In QA, a quality assurance engineer tests the package changes to ensure that the changes meet requirements. The engineer promotes the package to Production if the package passes testing.
4. In Production, packages accumulate and engineers include them in the product builds. When a package enters Production, the PROD view is updated with the package changes.

More information:

Sequential Movement of Packages (see page 205)

Chapter 3: Getting Started in the Workbench

Note: The instances of Linux in this section refer to both the Linux and zLinux operating environments.

This section contains the following topics:

Workbench Arguments

You can set up the following workbench arguments from the workbench.ini file from the CA Harvest SCM home, to use the Workbench effectively.

1. **-data workspacePath**

   Specifies the workspace path for the workbench.

   Example:

   Usage in Windows:

   -data

   C:/Users/workspace

   Usage in Linux/Mac:

   -data

   /home/harvest/workspace

   Note:

   - Ensure to type the *-data* and the workspace path in two lines. The workbench.ini file parses the options and the specified arguments individually.
   - You can edit the workspace prompt option by following the instructions:

     a. Open the org.eclipse.ui.ide.prefs file from the @userprofile\.cascm\configuration\.settings\ location.

     b. Set the SHOW_WORKSPACE_SELECTION_DIALOG=true value to get a workspace prompt.

2. The following parameters are passed to the Java run time and define the amount of memory available to the workbench.

   a. **-vmargs**

   Customizes the Java VM operation to run the workbench.

   b. **-Xms 40m**

   Specifies the minimum memory that the workbench can allocate. This command sets minimum memory size for the pile and heap.

   Default: 40 MB.

   c. **-Xmx 512m**

   Specifies the maximum memory that the workbench can allocate. You can set the memory from the Tools, Preferences, and System page. This command sets maximum memory size for the pile and heap.

   Default: 512 MB.

   Note: Ensure to type the –vmargs, -Xms, and –Xmx arguments on different lines. The workbench.ini file parses the options and the specified arguments individually.

3. **-Dfile.encoding**

   Specifies the text file encoding mode.

   Example:

   -Dfile.encoding=utf-8

4. **-clean**

   Cleans the cached data that is used by the OSGi framework and the cache data that is generated during workbench runtime. This command cleans up the cached data even before starting the Workbench.

   If you encounter any error during the start-up or update tasks, add this command in the workbench.ini file and restart the workbench.

The Development Process

CA Harvest SCM streamlines the development process and lets you do the following:

- **Manage files based on projects not individual files**—Simplifies the process of synchronizing data to the repository. You keep track of only your projects, not the files and folders below them. New, changed, and deleted resources are tracked for you.
- **Integrate with the development environment**—You can perform SCM tasks, such as navigating the CA Harvest SCM package list and working view, and show version history without leaving your local working area.
- **Use the power of the CA Harvest SCM life cycle**—You use CA Harvest SCM packages to manage versions and move them through the CA Harvest SCM life cycle.
- **Use both early and late binding models of SCM**—The CA Harvest SCM administrator has the option of supporting the early binding (pessimistic) model of SCM by forcing all check-outs to be on the trunk; or the the administrator can support the late binding (optimistic) model, by forcing all check-outs to be on the branch and creating concurrent and interactive merge processes. In addition, the administrator can support both models and give individual users a choice.

How to Perform Basic Tasks

Using the Workbench, you can perform the following basic tasks; the order in which you perform some activities in the CA Harvest SCM can vary:

1. Log in to the Workbench. Open the New Connection dialog and define your CA Harvest SCM login credentials. Your broker location persists until you discard it.

2. Navigate the Explorer View in the Workbench, select a folder, and add it to your WorkAreas View.
3. Create, modify, and save files locally.
4. Test your changes.
5. Synchronize your WorkAreas files with the repository (Explorer View).
6. Complete your changes and check in your WorkAreas files to the repository.
7. Merge your versions with the trunk.
8. Use the CA Harvest SCM life cycle to update your form and promote or demote your change package.
9. Log in to a remote computer using an agent and perform functions.

Log In to the Workbench

To use the Workbench, you must log in to a CA Harvest SCM broker. To log in successfully, complete the following prerequisites:

- The database processes must be running.
- The CA Harvest SCM broker must be running.
- At least one CA Harvest SCM server process must be running.
- (Linux only) Set the JAVA_HOME variable export to JAVA_HOME=<dir>.

Follow these steps:

1. Launch the Workbench by doing the following platform-specific step:
   - (Windows) Start the Workbench executable from the program group.
   - (Linux) Run the Workbench script (workbench) in the $CA_SCM_HOME/bin folder.

   The Workbench appears.
2. Click Connection, New Connection.

   The New Connection Wizard appears.
3. Enter the broker name.

4. Enter your CA Harvest SCM user name and password.
5. (Optional) Disable the Use RT Server on same machine as broker option, and enter an RTderver name in the RT Server field.

   Click Finish.

   You are logged in to the Workbench and the broker you specified during login appears in the Explorer View.

   Your broker location persists until you discard it. Exiting CA Harvest SCM does not discard your location or remove any projects.

Note: In one Workbench session, you can log in to multiple brokers served through a single RTserver as long as Enterprise Communicator (PEC) is configured appropriately. Additional broker connections are added as top-level broker nodes in the Explorer View. If the Explorer filter is active, add the new broker as an Explorer filter value to display the broker in the Explorer View. For information about configuring PEC, see the *Implementation Guide*.

Login Authentication Considerations

When CA Harvest SCM is first installed, an initial user is created. This user must add other users who will be accessing CA Harvest SCM. The CA Harvest SCM administrator defines a user's initial password. The administrator can optionally set password policy to define rules for password validation.

If your site uses internal authentication (CA Harvest SCM Authentication), you can log in only if your user name and password are valid login credentials in CA Harvest SCM. If you attempt to log in with an expired password, CA Harvest SCM prompts you for a new password.

If your site uses external authentication, such as Microsoft Active Directory, you can log in if your user name exists in CA Harvest SCM and your user name and password are valid login credentials on the authentication server. If login fails due to password expiration, you are not prompted to change the password; you must change your password by using methods provided by the authentication server. For example, in Microsoft Active Directory, you can modify your password by using Ctrl+Alt+Delete and using the Change Password option.

If your site uses internal authentication and if the password policy defines an expiration warning, CA Harvest SCM issues a warning message indicating how many days remain before password expiration.

If your site uses external authentication, such as Microsoft Active Directory, because the CA Harvest SCM Password Policy is not in effect, you do not get password expiration warning messages.

Note: For information about password policy and external authentication, see the *CA Harvest Software Change Manager Implementation Guide*.

Workbench Development Environment

After you connect to CA Harvest SCM, the Workbench appears. The Workbench depicts CA Harvest SCM objects and their relationships and provides access to functions. The Workbench is comprised of panes called *views*. You can show, hide, move, and resize views to fit your needs.

The CA Harvest SCM Workbench GUI contains the following views:

- Explorer
- Packages
- WorkAreas
- Lists
- Properties
- Editors
- CA Harvest SCM Log

Important! The term *view* is also used in CA Harvest SCM for an object that determines which items specific users or user groups can access and view in repositories.

More information:

How to Explore the Broker (see page 33)
Filter the Packages View (see page 225)
How to Display Objects in the Lists View (see page 36)
View Object Properties (see page 38)
View a Record of CA Harvest SCM Activity (see page 40)

How to Explore the Broker

You can use the Explorer View to explore the CA Harvest SCM brokers and perform CA Harvest SCM activities.

To explore and use the Explorer View, do the following:

1. Log in to the Workbench.

   The Explorer View initially appears in the upper left portion of the Workbench and shows your CA Harvest SCM objects in a hierarchical structure. Icons identify packages, with forms and versions listed beneath the packages. Data views show items and versions.

   All active projects appear as child nodes of brokers. State nodes appear as child nodes of projects. Each state node has its own package and view nodes as its children. The View node displays the entire view tree.

   Note: If the agent -lockdir option has been specified, the tree display is limited to the specified path and its subdirectories. Relative lock directories (for example, ../../user01/CA Harvest SCM) cannot be specified; if CA Harvest SCM detects a relative -lockdir specification, the file agent will not start.

2. Navigate your package lists and repository, perform most of your day-to-day CA Harvest SCM activities, such as creating a package and form, checking items in and out, and moving packages through the life cycle.

3. (Optional) Navigate the Explorer View, select a folder, and add it to your WorkAreas View.

4. (Optional) Access snapshot views in the project in which they were created if the All Snapshot Views option is enabled in the State Properties dialog.

   The state has as its view all snapshots that exist in the project. The versions the snapshot captures are visible in the state and can be checked out and added to the WorkAreas View. Snapshot views are read-only. Items cannot be checked in or modified from this state, and the local projects cannot be shared with the snapshot state.

   Note: For more information about using snapshot views, see the *Administrator Guide*.

Filter the Explorer View

You can customize which objects are visible to you in the Explorer View tree so that you view only the objects pertinent to your work.

Follow these steps:

1. Click Filter on the Explorer View toolbar.

   The Explorer View Filter dialog appears.

2. Select Enable filter.

   The filter selections in the tree are enabled or disabled.

3. Specify the objects you want visible in the Explorer View tree by selecting them and completing the following fields and controls. When you select an item, the ancestor items are marked with a gray checkmark. This checkmark indicates that the item is visible, although not all of its children are visible. A black checkmark indicates that the item and all of its children are visible.

   Select All *and* Deselect All

   Selects or clears all check boxes in the tree.

   Package Filter Pattern

   Specifies a name pattern that limits the set of packages displayed in the Explorer View Filter dialog tree. By entering a name pattern, only those packages that conform to the name pattern are displayed in the dialog tree. The display is not updated until you click OK.

   Filter Packages

   Applies the package name pattern to the displayed tree view in the dialog *only*; it does not affect the Explorer View filtering.

   Use Regular Expression

   Specifies the package name pattern using standard regular expression syntax. The packages permitted by the pattern are visible wherever they appear in the filter dialog tree. You can then designate which packages are visible in the Explorer View by selecting their check boxes.

   Click OK.

   The Explorer View is customized with your filter selections.

Filter Packages in the Explorer View

You can set the Explorer View tree to display packages according to a filtering option. When you do this, the tree automatically reorganizes to display the packages that match the option you selected.

To filter packages in the Explorer View, click View, Filter Packages, and select an option:

All Packages

Shows all the packages for the broker.

My Created Packages

Shows packages that you created.

My Assigned Packages

Shows packages assigned to you.

My Modified Packages

Shows packages that you modified.

The Explorer View package tree shows only the packages that match your option.

Show a Package-Centric Explorer View

You can set the Explorer View tree to display versions from a package perspective. When you do this, the tree automatically reorganizes to display the package's branch versions in folders. Expanding these branch versions reveals other related versions. This feature is particularly useful if you want to view new items or folders created on a branch.

You show a package-centric Explorer View by selecting a package from the Package drop-down list at the top of the Explorer View. When you select a package, the working view root node updates to reflect the package's versions. If you select a package and create a path on a branch using the package, the new path version immediately appears in the Explorer View tree. If you do not select a package, the Explorer View works in the usual trunk-oriented way.

Follow these steps:

1. Select a node in the Explorer tree at the State level or lower.

   The Package field is enabled.

2. Select a package from the Package drop-down list, which lists all packages in your State context.

   The Explorer View tree is reorganized to show the package's branch versions in folders. This includes new items or folders created on a branch.

3. (Optional) Expand any of these branch versions.

   Other related versions are displayed.

Example: Show a Package-Centric Explorer View

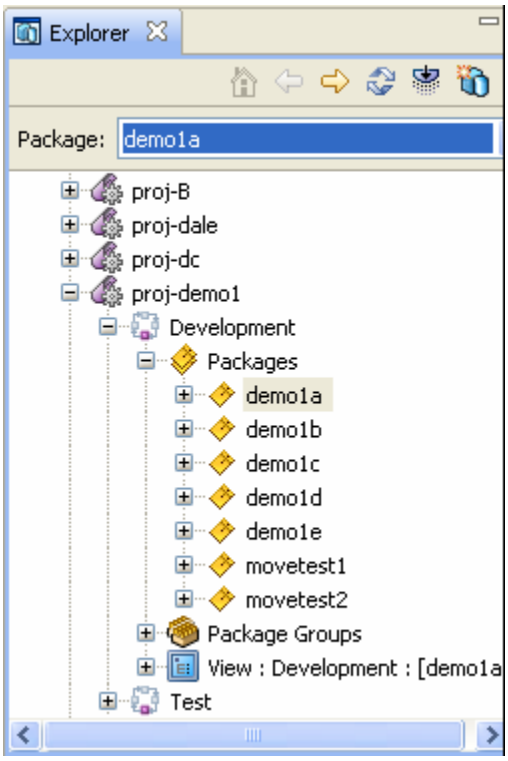
This example shows a package named demo1a in a package-centric view.

1. Set the State context to Development.

   The Package field is enabled.

2. Select demo1a from the Package drop-down list, which lists all packages in Development.

   The Explorer View tree displays the View root as Development : [demo1a] as shown in the following illustration:



How to Display Objects in the Lists View

The Lists View shows CA Harvest SCM objects in a tabular format. You can execute processes specific to items, versions, and packages from this view using the shortcut menu. Any functions that you can execute in the Explorer View are available in the Lists View.

To display objects in the Lists View, select the parent in the Explorer View as follows:

- Click a package folder to list packages.
- Click an item path to list items.
- Click a specific item or a versions folder to list versions and their associated packages.

To rearrange columns in the Lists View, select a column header and drag it to a different column header location.

Note: You can also change column width by dragging the column header boundaries.

Save Lists View Table

You can save or copy the Lists View table contents to a file under a different name, format, or in a different folder.

Follow these steps:

1. Right-click the Lists View contents, and select Save List As from the shortcut menu.

   The Save As dialog appears.

2. Select options to save data. The following fields require explanation:

   CSV

   Specifies Comma Separated Value (CSV) format, which is a type of data format in which a comma separates each piece of data. CSV format can be helpful for transferring data from one application to another, because most database systems are able to import and export comma-delimited data. This format can also be helpful to open the file in spreadsheet applications such as Microsoft Excel and OpenOffice.

   Text

   Specifies ASCII text format, which is a type of data format that is useful for opening the file in a standard text editor.

   Save contents to a file

   Names the output file specifies a location for it.

   Click OK.

   The file is created in the location you specified.

3. Select the following option to copy data.

   Copy to Clipboard Only

   Copies the formatted contents to the operating system's clipboard and exits.

   The table contents are saved or copied.

Properties View

When you select objects, property information for the selected object appears in the Properties view. For example, when you select your WorkArea, the Properties view shows the CA Harvest SCM broker, project, state, and folder associated with the WorkArea, and its location on your computer.

View Object Properties

Each object in CA Harvest SCM has associated properties. Properties let you view the attributes of objects.

To view an object's properties, click an object in the Explorer View, Lists View, Packages View, Versions View, or WorkAreas View.

The object properties appear in the Properties View.

Alternatively, right-click an object, and click Properties from the shortcut menu.

A detailed Properties dialog appears.

Example: View Object Properties

This example of the Properties View shows the CA Harvest SCM broker, project, state, and folder associated with the WorkArea, and its location on your computer.

Click your WorkArea.

The WorkArea properties appear in the Properties View.

View Object Information

Properties includes a description that provides relevant information about the object. The contents of the description can be viewed, but not changed unless you have the proper access.

To view information about an object, right-click an object and select Properties from the shortcut menu.

The object Properties dialog appears. Any comments or instructions that can be helpful to users using the object appear in the description.

Rename an Object

The Rename Object dialog lets you change the name of objects (forms, packages, and package groups) in the Explorer View and List View. Additionally, the Find Form and Filter Packages dialogs display search results in a list of available objects of a certain type, and you can execute the Rename function on your selection. For example, after locating the form you want to rename, you can right-click it and select Rename from the shortcut menu to open the Rename Object dialog.

The following rules apply to the rename feature:

- You can only rename an object to which you have update access.
- You must use the rename item or rename path processes to rename items or rename paths, respectively.

Follow these steps:

1. Navigate the Workbench to the object you want to rename.
2. Right-click the object, and select Rename from the shortcut menu.

   The Rename Object dialog appears and displays the object name you selected.
3. Enter a new name for the object, and click OK.

   The object is renamed.

Edit Files

The Editor View lets you view and edit your files. You can have multiple editors open at once; the active editor is in the foreground. You can switch editors by clicking the tabs.

To view and edit a file's content, double-click a file version.

The file versions contents appear in the Editor View.

Type Ctrl+S to save changes.

Maximize an Editor or View

You can maximize an editor or view so that it is easier to use:

- To maximize a Structure Compare or Version Compare editor, double-click its title bar.

  The editor expands to fill the comparison area.

- To maximize a form editor, double-click its title bar.

  The form editor expands to fill the Workbench.

- To maximize a view, double-click its title bar.

  The view expands to fill the Workbench.

Note: To restore a maximized editor or view to its original size, double-click its title bar again.

Display as a Diagram

The Diagram Overview gives you a structured display of a version or life cycle diagram currently opened in an editor. The outline contents are diagram-specific. For example, a life cycle diagram displays as an outline structure of projects, states, and processes. The Diagram Overview is helpful for viewing large diagrams because it lets you move the diagram.

To display a diagram, click View, Show View, Diagram Overview.

The diagram displays.

View a Record of CA Harvest SCM Activity

You can view a record of your current session activity by clicking the Output Log view tab. Most activities in CA Harvest SCM generate output that is sent to the Output Log view. The log is informational and documents the results of a previous action. The messages are initially sorted chronologically, with the most recent messages appearing at the bottom. The log contents are maintained throughout the session, unless you clear or close the log.

To use the Output Log view, do one of the following:

- Double-click a message row in the log.

  A Details dialog appears and displays the entire text reported by the server.

- Click a column heading to sort the list by a particular category:
  - The exclamation point icon (!) sorts messages by severity.
  - Message sorts messages by message name.
- Right-click the Lists View contents, and select Save As from the shortcut menu.

  The Save As dialog appears.
- Use the Output Log toolbar buttons:

  Clear

  Clears the log.

  Restore Natural Sort Order

  Restores the log to the chronological order of the session output.

  Show details about selected status item

  Shows details about the message.

File and Directory Exclusion

When you perform synchronize operations, you may want to exclude certain files and directories in your project from being committed to the repository. For example, you may not want to commit the following files:

- Temporary files that external editors create
- Class files that Java compilations create
- Binary files that build operations create

These files can be numerous and they may be regenerated whenever a build is performed, resulting in a large number of outgoing changes. Typically, these are not files that you want to remain in the repository or share with other members of a team.

You can specify which files and directories should be excluded from update and commit operations in the following ways:

- Use the Ignored Resources preferences
- Use the .scmignore file

You can use the following wildcards to specify patterns:

asterisk (*)

Represents any sequence of zero or more characters. For example, the pattern of *~ matches any temporary files that end with ~.

question mark (?)

Represents any single character.

Ignored Resources Preferences

Ignored Resources preferences is a global facility that lets you designate files or directories to be ignored by version management. You can set or remove patterns of files and directories to be ignored. Any file or directory that matches any one of the patterns will be ignored during update or save operations. The patterns in the global ignore facility are matched against file and directory names during a synchronize operation. The path to the file name is not included in the matching. For example, for the file /directory/subdirectory/file.txt, only the file.txt string is matched against the patterns. Use this facility for specifying globally applicable patterns but not for specifying fully qualified path names.

.scmignore File

A special file named .scmignore lists files, directories, or patterns to ignore. Any file or subdirectory in the current directory that matches any one of the patterns will be ignored. The .scmignore file differs from the global ignore facility because it applies only to files and directories that exist in the directory where the .scmignore file is located. A project can contain one .scmignore file in each directory.

Change Your Password

The Change Password dialog lets you change your password if your site uses internal authentication.

If your site uses an external authentication server, such as Microsoft Active Directory, the change password function is disabled. Your password must be changed using methods provided by the authentication server. For example, if the server uses Microsoft Active Directory, you can modify your password by entering Ctrl+Alt+Delete and using the Change Password option.

Follow these steps:

1. Right-click the broker name in the Explorer View, and select Properties from the shortcut menu.

   The Broker Connection Properties dialog appears.

2. Click Change Password.

   The Enter a New Password for User *name* dialog appears.

3. Specify and confirm a new password. Click OK.

   Your password is changed.

Note: Click OK to close the Broker Connection Properties dialog.

Clear a Password

You can remove your password from file system storage if you previously saved your password.

Follow these steps:

1. Right-click the broker name in the Explorer View, and select Properties from the shortcut menu.

   The Broker Connection Properties dialog appears.

2. Click Clear Password.

   The password is removed from the file system. You will be prompted for the password if you restart the Workbench.

Note: Click OK to close the Broker Connection Properties dialog.

RT Server Properties

The RT Server name and the RT Server port number details are displayed.

Server Properties

The Server Properties display the CA Harvest SCM Database details, Product Version, and the Build details.

Database details

The database with which the CA Harvest SCM Server is configured and started.

Example: Microsoft SQL Server 2008 R2 -10.50.1600.1(x64)

Product Version

The release level of CA Harvest SCM Server.

Example: r12.5.0.0

Build

The build number of the corresponding CA Harvest SCM Server release level.

User Properties

The User Properties display the details of the presently logged in user in the Workbench session. The details include user name, real name, email, and user group membership details.

Discard Broker

You can discard a broker if you no longer use it and to streamline the Explorer View.

Note: Exiting CA Harvest SCM does not discard your broker or remove any projects.

Follow these steps:

1. Right-click the broker name in the Explorer View, and select Discard Broker from the shortcut menu.

   A confirmation dialog appears.

2. Click OK.

   The broker is discarded.

Connect to a Remote Agent

To access a remote agent, you must first establish a connection to that agent. The agent login function connects to the agent on the specified computer and port number. After connecting to an agent, the remote computer's directories and files appear and are accessible in the Remote Agent node.

Note: When you connect to multiple agents on the same computer and check-out from the Workbench, CA Harvest SCM identifies the agent with a unique combination of agent name, port number, and the initial directory. This selection auto populates the To field with the initial directory of the agent computer.

Follow these steps:

1. Click Connection, New Agent Connection from the main menu.

   The New Agent Connection dialog appears.

2. Complete the dialog fields:

   Machine Name

   Specifies the name of the computer you want to log in to.

   Login Name

   Specifies the login name for the remote agent.

   Password

   Specifies the password for the remote agent.

Initial Folder

Specifies the full path to the initial directory.

- For UNIX, Linux, or z/OS, if the initial directory is not specified you are logged in the user home directory.
- For z/OS remote file systems, the initial directory determines whether the remote z/OS agent displays openMVS Hierarchical File System (HFS) files or MVS Partitioned Data Sets (PDSs). When listing PDSs, the Explorer View lists only non-migrated PDSs with high-level qualifiers equal to the initial directory. When a PDS is selected in the Explorer View, the PDS members appear in the Lists View.

Note: The Universal Naming Convention (UNC) is not supported for specifying client paths. Instead of entering the UNC location *server**share**directory*, you must enter *drive:**directory*.

Save login properties

Specifies that a connection is to be established to the remote file agent the next time the Workbench is started.

Port Number

Specifies the port number for the remote agent.

The agent connection specifications are selected.

3. Click Finish.

   The remote agent connection is established and is added to the Explorer View.

Note: For more information about using agents, see the *Implementation Guide*.

Change Initial Folder

The Change Initial Folder option lets you change the initial folder for a connected agent.

Follow these steps:

1. Right-click a top-level agent node, and select Change Initial Folder from the shortcut menu.

   The Browse for Agent Folder dialog appears.

2. Specify a different folder by entering a folder location in the Browse folders field or by navigating the Folders tree. Click OK.

   Note: MVS filesystems users can use the Browse folders field to enter a data set name (DSN) or high-level qualifier (HLQ).

   The agent node automatically refreshes to reflect the changed initial folder.

Note: If you select a top-level agent node in the dialog, the initial folder will be reset to its default.

Discard an Agent Connection

To discard an agent connection, right-click the agent in the Explorer View, and select Discard Agent from the shortcut menu.

The agent connection is discarded.

View a Life Cycle

The Lifecycle Diagram gives you a graphical view of the development life cycle (CA Harvest SCM project). Project states are depicted as boxes in colors that depict the view types associated with the states. Arrows represent promote and demote processes. Icons in the state boxes represent approval processes.

Follow these steps:

1. Right-click a project, and select Lifecycle Diagram from the shortcut menu.

   A diagram appears in the Workbench editor.

2. (Optional) Use Zoom In and Zoom Out.

   The diagram is magnified or reduced.

3. (Optional) Right-click any node.

   A shortcut menu related to the node appears.

Context

Context is your project, state, package, or snapshot selection in CA Harvest SCM. Your current context is your current location and determines what is displayed and is available according to your access and privileges. For example, selecting a state lets you view packages, package groups, versions, and forms. Each process that is available in the selected state appears on the shortcut menu when you right-click the state and you can invoke any process for which you have access.

Set an Agent Check-In Context

The Agent Checkin Context dialog lets you specify a context to use for checking in files from the Agent Explorer. Setting an agent check-in context, lets you check in files directly from the Agent Explorer tree. Each agent connection has its own unique check-in context. If no check-in context is set for an agent, check-in processes are not available when you right-click a file in the Agent Explorer tree.

Follow these steps:

1. Right-click a top-level agent node, and select Agent Checkin Context from the shortcut menu.

   The Agent Checkin Context dialog appears.

2. Select the broker, project, and state you want to use for checking in files. Click OK.

   The agent check-in context is set.

Note: The settings in the Agent Checkin Context dialog remain the same until you modify them, and are saved if the Workbench is shut down and restarted. After you set a context, you do not need to open the Agent Checkin Context dialog unless you want to change the settings.

Processes and the Objects They Use

Each CA Harvest SCM process works on only certain objects. After you select an object, only appropriate processes for that object show on the Processes menu of the Workbench. For example, after you select a package, you can select a promote process (if defined) from the menu, but a compare view process is not available even if the administrator defined one for your context and access. The process is unavailable because you cannot use a package to compare views. The following table shows each CA Harvest SCM process and the objects it uses.

Process	Object
Approve	Packages

Process	Object
Check-in	Files, versions, packages
Check-out	Versions, packages
Compare View	Items
Concurrent Merge	Packages
Create Package	Context
Cross Project Merge	Packages
Delete Package	Packages
Delete Version	Versions
Demote	Packages
Interactive Merge	Versions
List Version	Versions
Move Item	Items
Move Package	Packages
Move Path	Paths
Notify	Packages
Promote	Packages
Remove Item	Items
Remove Path	Paths
Rename Item	Items
Rename Paths	Paths
Switch Package	Packages
Take Snapshot	Views
User-defined Process	Packages, versions

Execute a Process

The processes defined for a state determine which processes you can execute in that state and which users can execute these processes. When you right-click a state, the menu shows all processes that you have access to in that state.

Follow these steps:

1. Navigate to the state or the object (package, item, or version) that you want to use for the process.
2. Right-click the state or the object, and select *process name* from the shortcut menu.

   The process execution dialog appears.
3. Complete the dialog fields, and click OK or Apply.

   The process executes.

Process Names and Availability

The administrator determines the processes available to Workbench users during the setup of the life cycle. The processes defined for a state determine the processes that can be performed in that state and which users can execute these processes. Selecting a state in the tree of the Workbench and selecting Processes shows all processes that you have access to in that state.

The process name displayed in the Processes menu and in the process execution dialog title bar is specified by the administrator in the Properties dialog for the process. Because process execution dialog names differ depending on your the site, CA Harvest SCM documentation uses lowercase italic text to indicate the process dialog name by the process type.

Note: For information about the Properties dialogs and access for each process, see the *Administrator Guide*.

Linked Processes

Each process in a lifecycle can have one or more processes linked to it. The following methods of process linking are available:

- *Pre-linked* processes execute *before* the start of the parent process. For example, processes pre-linked to a promote process execute before packages are promoted to the next state in the lifecycle.

  Note: The pre-linked method is unavailable for the command-line utility except for the promote and demote processes.
- *Post-linked* processes execute *after* the start of the parent process. For example, post-linked processes execute only after the packages are successfully promoted.

The order of display in the Pre-Linked or Post-Linked tabs determines the order of the linked process execution. Your CA Harvest SCM session is suspended during execution of linked processes. If multiple processes are linked to one parent process, the execution of each depends on the success of the previous one. If a linked process or the parent process fails, none of the remaining processes execute including the parent process.

Note: You must set up linked processes so that they do not require any input from the user upon execution. If the input is insufficient for execution, the process fails.

Access Context-Sensitive Help

When you are using the Workbench, you can access context-sensitive help that provides information about the interface part you are using.

To access context-sensitive help, click on the interface part for which you want information and press F1.

The Help view displays and provides specific information about the view, editor, or dialog you are using, and possibly links to topics for further help.

Note: Alternatively, in dialogs you can achieve the same result by pressing the help (?) button in the left lower portion of the dialog.

Accessibility

Keyboard access lets you use keyboard shortcuts to navigate and use the Workbench. A keyboard shortcut is a key or set of keys that performs a predefined action. If you cannot use a mouse or other pointing device, or if you cannot see the screen, you can use keyboard shortcuts to perform actions in the Workbench. You may find it faster to use keyboard shortcuts rather than a menu or a pointing device.

Underlined letters indicate keyboard mnemonics in the Workbench menus. Click the Alt key to show the keyboard mnemonic. For example, the F in Show File Menu is underlined, which indicates that you can press the Alt key and F to access the Show File Menu.

The following table lists Workbench actions and their corresponding keyboard shortcuts.

Action	Keyboard Shortcut
Copy	Ctrl+C
Cut	Ctrl+X

Action	Keyboard Shortcut
Delete	Delete
Find Form	Ctrl+Alt+F
Find/Replace	Ctrl+F
Find Version	Ctrl+Alt+V
Help	F1
New Agent Connection	Ctrl+Alt+A
New Broker Connection	Ctrl+Alt+B
Paste	Ctrl+V
Print	Ctrl+P
Redo	Ctrl+Y
Save	Ctrl+S
Save All	Ctrl+Shift+S
Select All	Ctrl+A
Show Connection Menu	Alt+C
Show Edit Menu	Alt+E
Show Explorer View	Alt+Shift+Q, E
Show File Menu	Alt+F
Show Help Menu	Alt+H
Show Lists View	Alt+Shift+Q, I
Show Diagram Overview	Alt+Shift+Q, M
Show Output Log View	Alt+Shift+Q, G
Show Packages View	Alt+Shift+Q, U
Show Properties View	Alt+Shift+Q, R
Show Tools Menu	Alt+T
Show Versions View	Alt+Shift+Q, N
Show View Menu	Alt+V
Show Window Menu	Alt+W
Show WorkAreas View	Alt+Shift+Q, A
Undo	Ctrl+Z
Zoom In	Ctrl+=

Action	Keyboard Shortcut
Zoom Out	Ctrl+-

Execute a Keyboard Shortcut

To execute a keyboard shortcut in the Workbench, you press one or more modifier keys and usually one additional key. For shortcuts that consist of multiple keys pressed together, hold down the modifier keys, quickly press and release the regular (non-modifier) key, and release the keys. For any key sequences that include a comma, the final key is a separate keystroke.

The shortcut executes.

Example: Show WorkAreas View

This example shows how to use the Alt+Shift+Q,A shortcut to show the WorkAreas View.

Press the Alt and Shift keys and, while continuing to hold them, press the Q key, release all the keys, and type A.

The Workbench displays the WorkAreas View.

Accessing External Applications from Workbench

CA Harvest SCM lets you launch scripts, custom tools, and external applications such as notepad, Visual Studio, and so on, from the Workbench interface. You can add any number of applications to the Custom menu option. The Custom menu is available under the Tools menu.

Add an External Application to Workbench

To launch an external application from Workbench, you must first add the details of the application in the Workbench interface.

Follow these steps:

1. Click Tools, Custom, Add/Organize in the Workbench interface.

   The Custom Tools Workshop dialog opens.

2. Click the  button and specify the values in the following fields:

   Custom Tool Name

   Defines the name of application that you are adding. This name is displayed in Custom menu.

   Command

   Defines the command to execute the application. For example, notepad.exe.

   Arguments

   Defines the arguments you want to pass when launching the application.

   Initial Directory

   Defines the directory that contains the data on which you want the program to operate.

   Default: CA_SCM_HOME

3. Click Apply.

   The external application is added to the Custom menu.

More information:

Modify or Delete External Applications (see page 54)
Change the Order of Application List in Custom Menu (see page 54)

Modify or Delete External Applications

You can modify the details of the external application after adding them to the Custom menu or delete them when you no longer need them.

Follow these steps:

1. Click Tools, Custom, Add/Organize in the Workbench interface.

   The Custom Tools Workshop dialog opens.

2. Select the external application that you want to modify or delete, and do one of the following:

   - Modify the details of the external application. For more information about the fields, see Add an External Application to Workbench (see page 53).
   - Click  to delete the details.

   Click Apply.

   The details of the selected application are modified or deleted.

Change the Order of Application List in Custom Menu

If you want to organize the applications list in the Custom menu according to your convenience, you change the order of display by moving the applications up or down in the list.

Follow these steps:

1. Click Tools, Custom, Add/Organize in the Workbench interface.

   The Custom Tools Workshop dialog opens.

2. Select the application that you want to move and keep clicking  to move up or  to move down until you position it according to your requirement.

   The selected application is positioned the way you need it.

3. Click OK.

   The application list under the Custom menu reflects the change you have made.

Chapter 4: Setting Preferences

Note: Topics are provided *only* for preferences that are not self-explanatory.

This section contains the following topics:

User Preferences

The Preferences dialog lets you set user preferences. You can search the Preferences dialog pages using the filter function. If you filter by matching the page title, type the name of the page you are seeking; the available pages are presented. The filter also searches on keywords such as appearance and logging. The history controls let you navigate through previously viewed pages. You can step backward or forward several pages at a time by clicking the drop-down arrow to display a list of the most recently viewed preference pages.

Set Workbench Preferences

Setting Workbench preferences lets you set up the Workbench to suit your needs.

Follow these steps:

1. Open the Workbench, and click Tools, Preferences.

   The Preferences dialog appears.

2. Set the preferences you want to use for your Workbench sessions, and click OK to save changes and dismiss the dialog or click Apply to apply changes only.
3. (Optional) Click the Restore Default button to restore dialog settings for the currently displayed preferences to their default values.

   Your Workbench preferences are set.

Set BIRT Preferences

The BIRT Reports preference page lets you set the count of rows in the report viewer to be displayed per chart depending upon your requirement.

For example: 5000

Follow these steps:

1. Click Tools, Preferences, BIRT Reports.

   The BIRT Reports preference page opens.
2. Type the number of rows to display in the text box and click OK.

   The BIRT Report row display count is set.
3. (Optional) Select the Export report without displaying in Report Viewer option to export the reports directly to your local file system.

   The report exports in the selected format.

   Note: The preference of number of rows to be displayed (example, 5000) is applicable only to the report viewer display and not for the direct export of the reports.

Set BusinessObjects Report Preferences

The BusinessObjects page lets you set a URL to use for opening the Log On to BusinessObjects InfoView page or for specifying a specific report URL.

Follow these steps:

1. Click Tools, Preferences, BusinessObjects.

   The BusinessObjects page appears and the BusinessObjects URL is populated with the following URL:

   `http://hostname:portnumber/InfoViewApp/logon.jsp`
2. Change host_name and port_number to specify your environment. Click OK.

   Your BusinessObjects report preference is set.

Set Repository Cache Preferences

The repository cache preferences let you enable cache usage and specify how to access the cache.

You can store and access the cache in the following ways:

- Specify a local (or LAN accessible) directory. Cache files are stored in this directory. Other users that also have LAN access to this directory can use it for a cache.
- Specify a remote agent. A remote agent stores and access the cache files. Users that have permission to log in to this agent can use it for a cache.

Follow these steps:

1. Select Tools, Preferences, Remote Site Cache.

   The Remote site Cache page appears.

2. On this page specify one of the following:

   - A local directory—Enter the directory name or use the browse button to locate the directory. If the directory name does not exist it is created the first time the cache is updated.
   - The location and login parameters for an agent to use for the cache—Enter the agent host name, port number, user name, and password.

3. Enable the Update cache during checkout check box if you want the cache to update as part of check-out operations.
4. Click OK.

   Your repository cache preferences are set.

Set Explorer Tree Preferences

The Explorer Tree preferences page lets you specify how to display files, and set indicators that mark packages to show package conditions.

You can display files in one of the following ways:

- In the Explorer View tree as children of folders. (Enable the Show Files in Tree option.)
- In the Lists View, when you select a folder in the tree; the List View automatically appears in the foreground. (Disable the Show Files in Tree option.)

Follow these steps:

1. Click Tools, Preferences, Explorer Tree.

   The Explorer Tree page appears.

2. Display files by selecting the check box:

   Show Files in Tree

   Shows files in the Explorer View tree as children of folders.

   Check In/Check Out Dialog

   Retrieves cached values and uses them for the check-in and check-out process dialogs.

   Show implicitly refactored versions under Packages

   Displays all the versions under package, that are created implicitly due to refactoring.

   Example:

   When the Folder A is refactored, all the versions that are created for the items in Folder A are also shown in the List view.

   Repository 1
   Folder A
   File1.java

   If the Folder A is renamed, versions created for File1.java are also be shown in the List view.

3. Select the preferences that you want to appear on packages from the Indicator check boxes. Click OK.

   Your Explorer Tree preferences are set.

Set General Preferences

The Workbench contains views and editors that control what appears in certain menus and tool bars. The General page lets you specify preferences for the general appearance and behavior of these views and editors.

Follow these steps:

1. Click Tools, Preferences, General.

   The General page appears.

2. Select the preferences that you want to use for your Workbench sessions:

   Always run in background

   Runs long operations in the background and does not block you from doing other work.

Keep next/previous part dialog open

Keeps editor and view dialogs open when their activation key is released; otherwise, the dialog closes as soon as the key combination is released.

Show Heap Status

Displays information about current Java heap usage.

3. Select one of the following methods for opening resources:

Double click

Selects a resource and a double-click opens the resource in an editor.

Single click (Select on hover)

Selects a resource and a single-click opens the resource in an editor.

Single click (Open when using arrow keys)

Uses the arrow keys to select a resource and open it in an editor.

Note: Depending on which view has focus, selecting and opening a resource may behave differently.

Click OK.

Your general preferences are set.

Set Compare Preferences

The Compare/Patch dialog lets you set preferences for text compare sessions.

Follow these steps:

1. Click Tools, Preferences, General, Compare/Merge.

   The Compare/Patch page appears.

2. Select the General preferences that you want to use for your compare sessions:

Open structure compare automatically

Automatically performs a structure compare whenever a content compare is done.

Show additional compare information in the status line

Shows additional information about a change in the status line.

Ignore white space

Does not show white-space changes in the compare viewer.

Filtered Members

Excludes members from Compare With Each Other.

Note: Names in a list must be separated with a comma.

Show structure compare in Outline view when possible

Displays an outline of a structured file that is currently open in the editor area, and lists structural elements.

Automatically save dirty editors before browsing patches

Controls whether any unsaved changes are automatically saved before a patch is applied. Select this option if you want to save changes automatically.

3. Click the Text Compare tab, and select options:

Synchronize scrolling between panes in compare viewers

Keeps identical and corresponding portions of the code in each pane of the comparison viewer side-by-side.

Initially show ancestor pane

Compares two versions of a resource with the previous version from which they were both derived. This previous version is called their common ancestor, and it appears in its own comparison pane during a three way compare. Turn this option on if you want the ancestor pane to always appear at the start of a comparison.

Show pseudo conflicts

Displays pseudo conflicts, which occur when two developers make the same change (for example, both add or remove the exact same line of code or comment).

Connect ranges with single line

Controls whether differing ranges are visually connected by a single line or a range delimited by two lines.

Highlight individual changes

Highlights differences between each change in the compared files.

Click OK.

The Compare preferences are set.

More information:

View Differences (see page 194)

Set External Compare and Merge Preferences

The external compare and merge preferences let you specify the external compare and merge tool that you want to use from the Workbench and Workarea. You can also specify whether you want to perform a two-way or three-way comparison.

Follow these steps:

1. Click Tools, Preferences.

   The Preferences dialog appears.

2. Navigate to General, Compare/Merge, External Compare/Merge Tools.

   The right pane displays external compare and merge tools with the associated command line options.

3. Select the tool you want to use and select the comparison mode.

4. Click OK.

   The selected external tool is configured for external compare and merge from Workbench and Workarea.

More information:

External Compare or Merge Tools (see page 180)
Using External Compare Tool from WorkArea (see page 139)

Set Network Connections Preferences

The Network Connections Preferences dialog lets you customize the network connection properties used by Workbench features which connect to the Internet.

Follow these steps:

1. Click Tools, Preferences, General, Network Connections.

   The Network Connections page appears.

2. Select the General preferences that you want to use for your network connections:

   Direct connection to the Internet

   Connects remote systems directly without involving a proxy server.

   Manual proxy configuration

   Connects remote systems through a proxy server.

   HTTP Proxy

   Specifies the server and port that is to be used when making HTTP connections. If the port field is empty, the default port of 80 is used. If you select the Use this proxy server for SSL option is selected, the HTTP proxy server is used for SSL connections as well.

   SSL Proxy

   Specifies the server and port that is to be used when making SSL connections. If the port field is empty, the default port of 443 is used.

   SOCKS Proxy

   Specifies the server and port that is to be used when making SOCKS connections. If the port field is empty, the default port of 1080 is used.

   No Proxy for

   Specifies, either by name or pattern, which hosts should not use any proxy but instead should always be connected to directly.

   Enable proxy authentication

   Specifies a user name and password that is to be used when connecting to the proxy server.

   Click OK.

   The Network Connections preferences are set.

Set Help Preferences

The Help preferences page lets you indicate how to display help information.

Note: The options you select on this page can affect how the help view is presented. If the selected browser is not fully compatible with Internet Explorer or Mozilla, or has JavaScript disabled, the help view shown in the browser might be a simplified version.

Follow these steps:

1. Click Tools, Preferences, Help.

   The Help page appears.

2. Specify preferences:

   Use external browsers

   Displays help contents if an embedded web browser is supported on your system. You can select this option to force help to use external browsers. The Web Browser preference page lets you select the browser to use.

   Open window context help

   Specifies whether the window context help will be opened in a dynamic help view or in an infopop.

   Open dialog context help

   Specifies whether the dialog context help is opened in a dynamic help section of help view or in an infopop.

   Click OK.

   The Help preferences are set.

Set History Diagram Preferences

The History Diagram dialog lets you set the preferences for displaying your History diagrams.

Follow these steps:

1. Select Tools, Preferences.

   The Preferences dialog appears.

2. Click History Diagram.

   The History Diagram dialog appears.

3. Select your History Diagram preferences:

   Trunk Orientation

   Specifies the orientation of the trunk versions in the diagram as follows:

   - **Horizontal**—All the trunk versions display in horizontal lines and the branch versions display in vertical lines.
   - **Vertical**—All the trunk versions display in vertical lines and the branch versions display in horizontal lines.

Node Width

Specifies the node width. Changes to node width are automatically reflected in the Preview area.

Node Height

Specifies the node height. Changes to node height are automatically reflected in the Preview area.

Version Details

Specifies pattern-based contents to be displayed within each node. You can edit the list, and right-click the Version Details field to open a shortcut menu that lets you select tokens to add to the list. Your current cursor position establishes the insertion location when you add tokens to the list. The following tokens are supported:

%name

%version

%package

%modifier

%creationTime

%modifiedTime

%clientPath

%description

%creatorName

Changes to version details are automatically reflected in the Preview area. Each token defined in the Version Details pattern will be replaced with its actual value when the diagram is rendered.

4. Click the Color and Fonts tab, select diagram colors and fonts, and click OK.

   Your History Diagram preferences are set.

Set Ignored Resources Preferences

The Ignored Resources dialog lets you designate files or directories to be ignored from version management. You can set or remove patterns of files and directories to be ignored.

Follow these steps:

1. Select Tools, Preferences.

   The Preferences dialog appears.

2. Click Ignored Resources.

   The Ignored Resources dialog appears.

3. Click Add Pattern

   The Enter Ignore Pattern dialog appears.

4. Enter the name of the file or directory or pattern (for example, *.class) to be ignored, and click OK.

   The designated files or directories are ignored for version management.

Follow these steps:

1. Select the file type in the ignore list.
2. Click Remove.

   The selected file type is ignored.

More information:

File and Directory Exclusion (see page 41)

Set Install/Update Preferences

The Install/Update page lets you set preferences for installing and updating the Workbench.

Follow these steps:

1. Click Tools, Preferences, Install/Update.

   The Install/Update page appears.

2. Select preferences:

   Maximum number of History configurations

   Specifies the maximum number of configurations you want maintained in the configuration history. These configurations are maintained to allow you to revert to a previous configuration of installed feature versions.

   Check digital signatures of downloaded archives

   Checks for digital signatures of downloaded archives.

Automatically select mirrors

Automatically selects update site mirrors.

Valid updates

Specifies an update level, assuming that feature versions use the form major.minor.service.

Update Policy

Specifies the update policy URL that controls the redirection of update sites within an organization.

Click OK.

The Install/Update preferences are set.

Update Manager

You can automatically install updates for both the CA Harvest SCM Workbench and CA Harvest SCM Plug-in for Eclipse, using the Update Manager. The Update Manager lets you find and install updates, and configure the Workbench or Plug-in for Eclipse. CA Harvest SCM administrators run the Update Manager from the command line or the interface. Workbench users typically run the Update Manager from the GUI only.

To use Update Manager in SCM Workbench, click Help, Find and Install Updates. To use the Update Manager in Eclipse SDK, click Help, Software Updates, Find and Install.

If your company policy prohibits the use of Update Manager to access the CA Update site, your administrator may set up a local update site where the administrator configures the local site to contain only the update versions approved by your CA Harvest SCM administrator.

Note: For more information about how to run the Update Manager provided by Eclipse from the command line or the interface, as well as setting up a local update site at your company, see http://help.eclipse.org.

Use Update Manager to Update the Workbench

You can automatically install updates to the CA Harvest SCM Workbench using the Update Manager.

Follow these steps:

1. From the main menu, select Help, Find, and Install Updates.

   The installation wizard appears.

2. Select Search for new features to install, and click Next.

3. Click New Remote Site.

   The New Remote Site dialog appears.

4. Complete the dialog fields:

   Name

   Specifies the name of the update site. Enter any name, such as “CA Harvest SCM Workbench Update Site”.

   URL

   Specifies the URL of the CA Harvest SCM Workbench Update Site. You must obtain the URL from your CA Harvest SCM administrator or contact Technical Support at http://ca.com/support.

   Click OK.

   The new CA Harvest SCM Workbench Update Site is added to the update sites list.

5. Select the check box next to the remote site created, and click Next or Finish.

   The Search Results dialog appears.

6. Expand the tree and select the check box for the feature (Workbench update) to be installed if one is found. Click Finish.

7. Accept the license agreement and click Next to continue through the wizard to install the update.

8. When prompted to restart the workbench, click Yes.

   CA Harvest SCM Workbench is updated.

Automatic Updates

You can configure the Workbench to search automatically for updates to the installed features on a periodic basis. The Automatic Updates preference page lets you configure how these updates are scheduled and performed. Update Scheduling options define when searches are performed and Download Options define what happens when updates are found. Each time you start the Workbench, the update scheduler executes in the background when necessary, based on the scheduling options. No messages appear if no updates are found. If updates exist, the behavior depends on the download option you select.

Note: If the application is not active at a scheduled time and the scheduled search is past due, the search immediately starts on the next startup.

If you decide not to install the download features, but later use the Help, Search for Workbench Updates wizard for updating, the downloaded files are reused—unless the server has newer versions or you have restarted the Workbench.

Note: You can automatically install updates by using the Eclipse Update Manager. The Update Manager lets you find and install updates, and configure the Workbench. CA Harvest SCM Administrators run the Eclipse Update Manager from the command line or the interface. Workbench users run the Eclipse Update Manager from the interface only. For more information about how to run the Eclipse Update Manager from the command line or the interface, see http://help.eclipse.org.

Schedule Automatic Updates

The Automatic Updates preferences let you schedule updates for your Workbench installation.

Follow these steps:

1. Click Tools, Preferences, Install/Update, Automatic Updates.

   The Automatic Updates page appears.

2. Click Automatically find new updates and notify me.

3. Select Update Scheduling options and a download option:

   Search for updates and notify me when they are available

   Searches for updates and notifies you when updates are found. Clicking Yes starts an upgrade wizard that downloads the updates.

   Download new updates automatically and notify me when ready to install them

   Downloads updates immediately when updates are found. When all the features have been successfully downloaded, a message dialog appears. Clicking Yes installs the new updates.

   Click OK.

   The Workbench automatic updates are scheduled.

Set Lifecycle Diagram Preferences

The Lifecycle Diagram dialog lets you set the preferences for displaying your life cycle diagrams.

Follow these steps:

1. Select Tools, Preferences.

   The Preferences dialog appears.

2. Click Lifecycle Diagram.

   The Lifecycle Diagram dialog appears.

3. Select Lifecycle Diagram preferences:

   Approval Indicator

   Specifies that an approval icon is to indicate an approve process.

   Horizontal Spacing

   Specifies the horizontal distance between nodes.

   Vertical Spacing

   Specifies the vertical distance between nodes.

   Node Width

   Specifies the node width. Changes to node width are automatically reflected in the Preview area.

   Node Height

   Specifies the node height. Changes to node height are automatically reflected in the Preview area.

   State Details

   Specifies pattern-based contents to be displayed within each node. You can edit the list, and right-click the State Details field to open a shortcut menu that lets you select tokens to add to the list. Your current pointer position establishes the insertion location when you add tokens to the list. The following tokens are supported:

   %name

   %view

   %numPackages

   %modifier

   %modifiedTime

   %creatorName

   %creationTime

   Changes to state details are automatically reflected in the Preview area. Each token defined in the State Details pattern will be replaced with its actual value when the diagram is rendered.

4. Click the Colors and Fonts tab, and select options. Clicking the Color field, opens a color palette that lets you select colors.

   Changes to colors and fonts are automatically reflected in the Preview area.

5. Click the Connections tab, and select options for showing promote and demote processes, their names, and how they should appear as lines. Clicking a Color field opens a color palette that lets you select colors.

   Click OK.

   Your Lifecycle Diagram preferences are set.

Set Logging Preferences

The Logging dialog lets you set the preferences for the output log.

Note: The Logging dialog does not affect the Eclipse metadata log. The Eclipse metadata log, which records application exceptions is always stored on the file system regardless of preference settings.

Follow these steps:

1. Click Tools, Preferences, Logging.

   The Logging dialog appears.

2. Select your logging preferences.

   Save Output Log

   Saves the output log to a file named CASCM.log. The file is created when you log in to CA Harvest SCM, and log information is recorded in it.

   Output Log Path

   Specifies a location for the output log file.

   Log View Length

   Specifies the maximum number of lines displayed in the Log View.

   Click OK.

   Your logging preferences are set.

More information:

View a Record of CA Harvest SCM Activity (see page 40)

Set WorkArea Preferences

The WorkArea page lets you set preferences for your WorkAreas View. These preferences can help keep your WorkAreas View up-to-date automatically.

Follow these steps:

1. Click Tools, Preferences, WorkArea.

   The WorkArea page appears.

2. Select from the following preferences:

   Always prompt for description on check in

   Prompts for a description whenever a content change is committed to the WorkArea.

   Always check out latest version

   Always checks out the latest version in the repository even though an old version of an item exists in the WorkArea.

   Commit edited items on Latest Trunk

   Commits the changes to the latest trunk version in the repository. For more information see, Change Commit Preferences (see page 146).

   Preserve local changes during Undo Check Out

   Preserves the changes in your local copy of the resource when you undo a checkout operation. By default, the undo checkout operation releases the reserved version in the CA Harvest SCM repository and checks out the latest version of the selected resource to the local WorkArea. During this process, the changes made to the resource in the local WorkArea are lost. The changes include edit and modify operations. When you select this option, the undo checkout operation only releases the version in the repository and retains the local changes on the existing version, without checking out the latest version.

3. Click OK.

   Your WorkAreas View preferences are set.

More information:

Define a WorkArea (see page 129)
Change the Commit Preference (see page 146)

Set Peer Review Preferences

The Peer Review page lets you customize the notification format for the review request creation and for the closure of the review request. To customize the notification format, you can use the following predefined variables.

${BROKER.NAME}

${PROJECT.NAME}

${STATE.NAME}

${PACKAGE.NAME}

${REVIEW.NAME}

${REVIEW.NOTES}

${REQUESTER.NAME}

${PRIMARYREVIEWER.NAME}

Follow these steps:

1. Select Tools, Preferences, Peer Review.

   The Peer Review page appears.

2. Enter the message format in the message fields. Use the sample text as a guide.
3. Click OK.

   The notification format for the review request creation and for the closure of the review request is set.

Chapter 5: Using Packages

This section contains the following topics:

Packages

In CA Harvest SCM, all version changes must be made in reference to a package. A package is the basic unit of work that moves through the lifecycle. It typically represents a problem, a task, or an incident that needs to be tracked, the changes made in response to the problem, task, or incident, and any associated information. The create package process lets you create a package in the state defined by the administrator as the initial state.

Packages have the following characteristics:

- **Package Group**—A package group consists of two or more packages. A package can belong to numerous package groups.
- **State**—Promote and demote processes change a package's state.
- **Forms**—A package can have a number of form associations. A form provides the link between CA Harvest SCM change management functions and problem tracking.
- **Versions**—A package can include numerous versions. Double-clicking a package expands the package to show the forms and Versions root node. To see the package's version, expand the Versions root node.

The Package Properties dialog lets you define a package and associate the package to one or more package groups.

Package Status Indicators

Package status indicators are visual cues that give you useful information about packages and the objects they use. After you set preferences to show indicators, package status indicators can appear in the Explorer View.

Indicators appear on a package icon to denote the following conditions:

- A package needs approval before it can be promoted.
- A package contains reserved tag versions of files.
- A package contains unmerged versions of files.

Package Names

When you create a package, the package name you specify must be unique in the project; you cannot create multiple packages with the same name.

Package names can be created in the following ways:

- Enter a package name in the Name field of the create package process dialog.
- Do *not* enter a name in the Name field of the create package process dialog and CA Harvest SCM generates a package name when you click Apply or OK. For example, the administrator sets up the create package process to use the default name template, Package-%N('999'). A counter generates unique package names. When you click Apply, the first package created is Package- 1, the second is Package- 2, and so on.

Create a Package

The create package process lets you create a package and, optionally, automatically create and associate a form to the package. If the form option was selected in the Create Package Properties dialog, a form with the same name as the package is created and the two objects are automatically associated. The create package process streamlines creating and automatically associating a package and form.

Follow these steps:

1. Navigate the Workbench to the state in which you want to create a package.
2. Right-click the state, and select *create package process* from the shortcut menu.

   The *create package* dialog appears.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.

3. Complete the dialog fields:

   Name

   Uniquely names a package, according to how package-naming is defined in the process properties.

   Assign To

   Assigns the package to a user from all defined CA Harvest SCM users in your installation.

4. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.

   Click OK.

   The package is created and appears in the Packages folder in the state in which you created it.

More information:

Add a Package Group to a Package (see page 82)

Set as WorkArea Context Package

You can use the Set as WorkArea context package action to set a package to a project-level context of a workspace project. This action is an alternative to the Team, Edit Context action.

Follow these steps:

1. Navigate the Explorer view to the package for which you want to set project-level context.
2. Right-click the package and select Set As WorkArea Context Package from the shortcut menu.

   The Set project-level context package dialog appears and lists projects that were added to the current CA Harvest SCM state and are available in the WorkArea.

   Note: If no projects are available in the WorkAreafrom the current CA Harvest SCM state, the menu option is disabled.

3. Select the projects to which you want to set the project-level context package. Click OK.

   The package is set to project-level context.

Package and Form Associations

Forms are most useful through association with packages and can be created only in association with one or many packages. Together, packages and forms represent one unified whole—a problem or enhancement request and the changes made in response to it. Each package can have one or more forms associated with it, linking the package to the change tracking features of CA Harvest SCM.

You can associate a form with a package using one of the following methods:

- Define the Create Package process to automatically create a form when executed. The type of form to be associated with the package is defined by the administrator when the process is set up.
- Associate a form with a package using the Forms tab of the Package Properties dialog.
- Add a form to a package.

You can create the following associations between forms and packages. All associations are bidirectional, letting you view an associated form from a package or an associated package from a form.

Note: Deleting a package deletes its associated forms if those forms are not associated with another package. However, deleting a form does not delete any associated packages.

- **One Package to One Form**—This association is the simplest relationship between forms and packages. You typically use this association when one form represents the description of a problem and one package represents the solution to that problem.
- **One Package to Multiple Forms**—When you use one package to resolve several related problems, you can associate multiple forms with the package. This relationship is useful when you need to record various types of information for a package. For example, you can associate Comment, Testing Info, and Problem Report forms to provide comprehensive information about a package.
- **Multiple Packages to One Form**—This association lets you link information about an identical problem in two projects. For example, if a critical problem is reported, you can assign it to a maintenance release of an application (Release 1.1) and at the same time assign it to the current development release (Release 2.0). A package is created in each project and linked to the same Problem Report form. Both packages share the same form rather than separate copies of it. If the form is modified, both development groups can immediately see the changes.

Form and package associations can be changed at any time. For example, several packages with associated forms are created addressing separate issues. Subsequent analysis reveals that these separate issues are all caused by one underlying problem. At this point, all the forms can be associated with one package, and the remaining packages can be deleted.

You can list a package's associated forms by clicking the plus (+) sign next to the package.

Associate a Package and a Form

The Forms list of the package Properties dialog lists the forms associated with the current package and lets you create or modify associations.

Follow these steps:

1. Navigate to the package from which you want to create the association.
2. Right-click the package, and select Properties from the shortcut menu.

   The package Properties dialog appears.
3. Click Forms, and click Add (the plus [+] sign).

   The Find Forms dialog appears and lists all forms in your installation.
4. Select one or more forms that you want to associate with the package, and click Accept Selected.

   The forms are added to the Find Forms Properties dialog.
5. Click OK.

   The associated form is listed below the package.

Remove a Package and Form Association

The Forms list of the package Properties dialog lists the forms associated with the current package and lets you remove associations—but does not delete the selected form.

Follow these steps:

1. Navigate to the package from which you want to remove the association.
2. Right-click the package from which you want to remove the association, and select Properties from the shortcut menu.

   The package Properties dialog appears.
3. Click Forms.

   The forms associated with the package are listed in the Forms list.

4. Select the form you want to disassociate, click Remove (the minus [-] sign), and click OK on the package Properties dialog.

   The associated form is no longer listed below the package.

Delete a Package

The delete package process lets you delete a package—and deletes its associated form if that form is not associated with another package.

Note: You cannot delete a package if it has versions associated with it.

Follow these steps:

1. Navigate the Workbench to the package you want to delete.
2. Right-click the package, and select *delete package process* from the shortcut menu.

   The *delete package process* dialog appears and lists the package you selected.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. (Optional) Do one of the following to add or remove a package from the deletion list:
   - Click Add (the plus [+] sign) to select one or more packages from the current project or state.
   - Click Remove (the minus [-] sign) after selecting a package.

   The selected packages are added or removed.
4. Click OK.

   The package is deleted and no longer appears in the Workbench package lists.

Rename or Reassign a Package

The package Properties dialog lets you rename or reassign a package.

Follow these steps:

1. Navigate the Workbench to the package you want to rename or reassign.
2. Right-click the package, and select Properties from the shortcut menu.

   The package Properties dialog appears.

3. Click Package, and view or modify the dialog fields:

   Name

   Names a package, depending on how package-naming is defined in the process properties.

   Assign To

   Assigns the package to a user. You can select a user name from all CA Harvest SCM users in your installation.

   Click OK.

   The package is renamed or reassigned.

Show Package Approvals

You can view the approval history of the current package to discover who approved a package and when it was approved.

Follow these steps:

1. Navigate to the package for which you want to view the approval history.
2. Right-click the package, and select Properties from the shortcut menu.

   The package Properties dialog appears.
3. Click Approval.

   The Approval list displays the package approval information.

View Package History

You can view the package history to find who has performed the following actions and when these actions occurred:

- Created the package
- Approved or rejected the package, with their comments
- Promoted or demoted the package

Follow these steps:

1. Navigate to the package for which you want to view the history.
2. Right-click the package, and select Properties from the shortcut menu.

   The package Properties dialog appears.

3. Click History.

   The package history displays in the History list.

Package Groups

A *package group* consists of two or more packages. You can use package groups to organize types of packages. For example, you could place all packages that affect the user interface of an application in a group named GUI_Project. If some of these packages also require documentation modifications, you could place them in the Documentation package group because a package can be in multiple groups.

The Package Group Properties dialog and the the New Package Group dialog let you define a package group and associate the package group to one or more packages.

Additional characteristics of package groups include the following:

- Package groups support filtering operations, such as Find Version, and let you set up different handling processes for different kinds of packages. You can perform almost all operations on a package group that you can perform on a package, such as approving and promoting an entire package group. For example, packages belonging to a documentation package group must pass through a special Documentation state before being promoted to the usual quality assurance (QA) and release states. Packages that do not belong to the documentation package group do not progress through the Documentation state.
- Package groups are flexible. You can create and delete package groups and add and remove packages from them at any time. This flexibility lets you group packages dynamically. You can use this type of grouping to define subprojects, which you can manipulate as a unit. For example, assume that all packages in a group named Phase_1 are in the QA state. If a problem is found in one of the packages in that group, you can demote that package to a previous state where more work can be done on it; and then you can promote it to the QA state to join the package group. In this way, you can manage a subproject as a group and manage smaller parts individually.
- You can use package groups to generate reports. You can generate reports on the status of packages in a specified group.

More information:

Create a Package Group (see page 81)

Create a Package Group

You can define a package group and add packages to it. You can use package groups to organize related packages.

Follow these steps:

1. Navigate to the project or state in which you want to create the package group.
2. Right-click the Package Groups node, and select New Package Group from the shortcut menu.

   The New Package Group dialog appears.
3. Click Add (the plus [+] sign) on the New Package Group dialog.

   The Select Packages dialog appears and lists all the packages in your current project or state.
4. Select one or more packages, and click OK.

   The Select Packages dialog closes, and the package is listed in the Packages list of the New Package Group dialog.

   Note: You can remove a package from the list of packages associated with the current package group by selecting the package and clicking Remove (the minus [-] sign). This does not delete the selected package; it simply removes the association.
5. Enter a name for the package group, select Bind Packages if you want to enforce the Bind Packages restrictions, and click OK.

   The package group is created and appears below the Package Groups node.

View and Modify Package Group Properties

The package group Properties dialog lets you view package group properties and modify its attributes.

Follow these steps:

1. Navigate to the package group you want to view and modify.
2. Right-click the package group, and select Properties from the shortcut menu.

   The package group Properties dialog appears.
3. Click Package Group.

   The Package Group page appears.

4. View or modify the dialog field:

   Name

   Names a package, depending on how package-naming is defined in the process properties.

5. Click Packages.

   The Packages page appears.

6. View or modify the dialog field:

   Bind

   Enforces the Bind Packages restrictions.

   Click OK.

   The package group is modified.

Add a Package Group to a Package

Using the package Properties dialog, you can add a package group to a package to organize related packages.

Follow these steps:

1. Navigate the Workbench to the package for which you want to add a package group.
2. Right-click the package, and select Properties from the shortcut menu.

   The package Properties dialog appears.

3. Click Groups.

   The package groups associated with the package are listed.

4. Click Add (the plus [+] sign) to list all the package groups in your current project. Select one or more package groups to add to the list of package groups associated with the current package.

   The package groups are listed in the Groups list.

5. Click OK.

   The package group is added to the package.

More information:

Create a Package (see page 74)

Remove a Package Group From a Package

Using the package Properties dialog, you can remove a package group from a package to organize related packages.

Follow these steps:

1. Navigate to the package for which you want to remove a package group.
2. Right-click the package, and select Properties from the shortcut menu.

   The package Properties dialog appears.
3. Click Groups.

   The package groups associated with the package are listed.
4. Select the package group you want to remove and click Remove (the minus [-] sign). This does not delete the selected package group; it simply removes the association.

   The package group is not listed in the Groups list.
5. Click OK.

   The package group is removed from the package.

Delete a Package Group

A package group consists of two or more packages. Deleting a package group deletes the package association, but does not delete the packages.

Follow these steps:

1. Navigate to the package group you want to delete.
2. Right-click the package group, and select Delete Package Group from the shortcut menu.

   A confirmation dialog appears and lets you add or remove package groups from the group deletion.
3. Click OK.

   The package group is deleted.

Chapter 6: Using Forms

This section contains the following topics:

Forms

CA Harvest SCM forms let you record, organize, and track information in a way that is similar to paper forms. CA Harvest SCM packages use forms to store package and user-entered information about the tasks being used through the package. For example, you can use a form to track support issues or software failures. A form can also help you organize information for a project or about a specific customer. You can create form attachments to provide additional relevant information. The form's modification history is maintained and can be viewed. Every form must have an associated package.

Default Form Types

CA Harvest SCM includes the following default form types to assist users in tracking change information. Administrators can also create customized form types to fit their needs.

Application Change Request

Records information about changes that are associated with SAP/R3 components. This form is associated with the create package process in the Packaged Application Change Management template.

Comment

Records any type of information and associates it with a package if needed.

Defect Tracking

Records information that Help Desk applications use. This form lets CA Harvest SCM users, who are updating their Defect Tracking forms, populate the Help Desk form fields by using the command-line utilities.

ESD Change Request

Records information that pertains to an electronic software distribution project. After the form is created, the ESD coordinator evaluates, prioritizes, and documents the request. The ESD coordinator assigns an urgency level to the package and documents the change in the form with a business justification, risk analysis, and so on.

Modification Request

Records all data regarding a software change request. Modification requests are typically associated with the package used to resolve the problem.

Problem Report

Records the problems that initiate changes to code that CA Harvest SCM controls. Problem Reports are typically associated with the package used to resolve the problem.

Q and A

Creates a database of information in question and answer format. Such a database can be an effective support tool for a help desk or customer support group. Keyword searches can be performed to retrieve answers.

Testing Info

Assists in the package turnover process and is typically associated with a package. Developers can record information of interest to the users who perform tests or QA. In turn, testing and QA personnel can record their responses for management review or to assist developers if the package is rejected.

USD Package Information form and USD Platform Information

Specifies the parameters for deploying software from CA Harvest SCM through Unicenter Software Delivery (previously named Unicenter Software Delivery):

- **USD Package Information** lets you record basic software delivery parameters such as the USD package name prefix and the deploy time.
- **USD Platform Information** lets you record specific details for a software deployment for a particular platform, including the operating system, the computers or computer groups on which to deploy the software, and the actual software components (files) used to install and deploy the software.

The USD forms are associated with the create software deployment package process in the Deploy Release template.

To use the Deploy Release template to deploy software, you must have both CA Harvest SCM and Unicenter Software Delivery installed and configured.

User Contact

Creates a database of customers or other product users. User forms can then be associated with problem forms that the customer submits. The information is organized so that a mailing list can be generated.

Create a Form

The create package process lets you create a form to record information about the package. The form type and the package that the create package process makes is configurable from CA Harvest SCM Administrator GUI.

Follow these steps:

1. Navigate the Workbench to the state in which you want to create the package and its form.
2. Expand the state, right-click Packages, and select *create package process* from the shortcut menu.

   The *create package process* dialog appears.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Complete the dialog fields:

   Name

   Names the package and corresponding form.

   Assign To

   Assigns the package to a user.

   Click OK.

   A package and an associated form are created. The form is listed below the package in the Explorer View.

Add a New Form to a Package

Forms attain their greatest usefulness through association with packages and must be associated with one or many packages. The Add New Form dialog lets you add and associate a form with a package.

Follow these steps:

1. Navigate the Workbench to the package to which you want to associate the form you are creating.
2. Right-click the package, and select Add New Form from the shortcut menu.

   The Add New Form dialog appears.
3. Name the form, select the type of form you want to add, and click OK.

   Note: If you do not enter a form name, a default form name is generated based on a unique ID and the form type name.

   The associated form is listed below the package on the Workbench.

More information:

Package and Form Associations (see page 76)

View or Edit a Form

Forms display in the Form Editor. The fields in a form are specific to the type of form loaded and vary by form type. Forms can be edited concurrently.

To edit or view a form, right-click a form and select Edit Form from the shortcut menu, or double-click a form.

The form displays in the Form Editor. If your form template defines a Harweb URL (for example, harweburl="http://uscomp01:8080/harweb"), the form displays in an embedded web browser.

Type Ctrl-S to save changes in the Form Editor.

Save on Shutdown

If the Workbench is shut down while one or more modified forms is open, the modified forms names are listed in a dialog.

To save the modified forms on shutdown, select them, and click OK.

The forms you selected are saved.

How to Resolve Form Conflicts

Two users can update a form at the same time. When you save the form that you have open, a dialog appears that lets you view and merge the differences between the form that you have open and the form that another user has open. The upper pane of the dialog displays individual form fields that conflict, organized in a tree by form pages.

Follow these steps:

1. Navigate the upper pane to a conflicting form field, and click it.

   Your current form field shows in the left pane and the right pane shows the form values that are currently stored on the server. The panes synchronize so that the two panes display the same fields.

2. Resolve conflicts in the left pane in the following ways:

   - Copy a change from right to left using toolbar actions.
   - Manually edit text in the left pane.
   - Accept changes in the left pane so that they overwrite the server field value.

   Note: The left pane contents *only* will be saved.

   Conflicts are resolved.

3. Click OK.

   The form merge dialog closes and the merged form contents are saved.

Delete a Form

Deleting a form removes its association with the associated package. If the form is not associated with another package, the form is deleted completely after your confirmation.

Follow these steps:

1. Navigate to the form you want to delete.
2. Right-click the form, and select Delete Form from the shortcut menu.

   A confirmation dialog appears and lets you remove forms from the deletion.

3. Click OK.

   The form is deleted.

Print a Form

Forms are listed in the Explorer View and appear in the Form Editor.

To print a form that is displayed in the Form Editor, click the Print icon on the toolbar.

The form prints in a list format.

Form Details Report

The Form Details report lets you obtain current and historical information about a form. The report includes the following information:

- The content of each form field.
- Information about associated packages, including the name, project, and state.
- Form history that shows the user, date, and time when a form has been created or modified. A record is added to this history when a form is created, its field contents are modified, its associations are modified, or its access is modified.

Generate a Form Details Report

The Form Details Report shows the form type and form name at the top of the report. Each tab of the form has its own subsection in the report. Associated Packages and Form History sections appear at the end of the report.

To generate a Form Details report, do *one* of the following:

- Right-click one or more forms in the Explorer View, and select Details Report from the shortcut menu.
- Right-click one or more forms in the Find Form list, and select Details Report from the shortcut menu.
- Select one or more forms on any process or properties dialog where forms are displayed in a table, and click the Form Report button.

The report appears in your default browser window. Multiple reports appear sequentially in the same browser window.

Form Attachments

Each form can have one or more attachments associated with it, linking the form to additional relevant information. You can create form attachments and use them to reference websites, view files, or copy files.

Two types of form attachments are available—file and URL.

- Files are copied to the CA Harvest SCM database. Files can originate on the local computer or a network drive.
- URLs reference web sites. Only the URL name is saved in the database.

More information:

View a Form Attachment (see page 92)
Copy a File Form Attachment (see page 92)
Remove a Form Attachment (see page 93)

Add a Form Attachment

You can add an attachment to a form to provide additional information.

To add an attachment, do one of the following:

- Drag a file from the native platform and drop it on a Form node in any tree view that shows forms.
- Drag a form attachment from one form to another in the Explorer View.
- Right-click a Form node, and select an attachment option from the shortcut menu:
  - Add URL Attachment opens the Add URL dialog that lets you enter a URL address to reference a web site from the form.
  - Add Attachment from Remote Agent opens the Browse for Agent Files dialog that lets you browse for and select files.
  - Add Attachment from Local File System opens a platform file chooser that lets you browse for and select files.

  Click OK.

The attachment node appears in the Explorer View tree below the form node.

Browse for Agent Files

You can browse for agent files to add as attachments to a form to provide additional information.

Follow these steps:

1. Right-click a Form node in the Explorer View, and select Add Attachment from Remote Agent from the shortcut menu.

   The Browse for Agent Files dialog appears.

2. Browse for and select files. Click OK.

   The selected files appear in the Explorer View tree below the form node.

View a Form Attachment

You can view form attachments to learn more about an associated form and package.

Follow these steps:

1. Navigate the Explorer View and expand the form node.

   The form attachments are listed beneath the form.

2. Perform one of the following actions:

   - Drag the file attachment from the tree onto the native file system to make a copy of the file. Double-click the copied file to view it.
   - Double-click the file attachment to open the node in an external editor.
   - Double-click the URL attachment to open the URL in an external web browser.

   Text or document files, graphics, and other files display in their associated applications. URLs display in the Internet Explorer.

Copy a File Form Attachment

You can copy form attachments.

To copy a form attachment to the file system, drag an attachment to the file system, open it in an external editor, and save it.

The form attachment is copied.

Note: You cannot copy URL attachments.

Remove a Form Attachment

You can remove an attachment that you no longer want associated with a form.

Follow these steps:

1. Navigate the Explorer View to the form attachment you want to remove.
2. Right-click the form attachment, and select Remove Attachment from the shortcut menu.

   Note: You can select multiple form attachments to remove.

   A confirmation dialog appears.
3. Click OK.

   The attachment is removed from the form. The selected attachment is not deleted; only the association is removed.

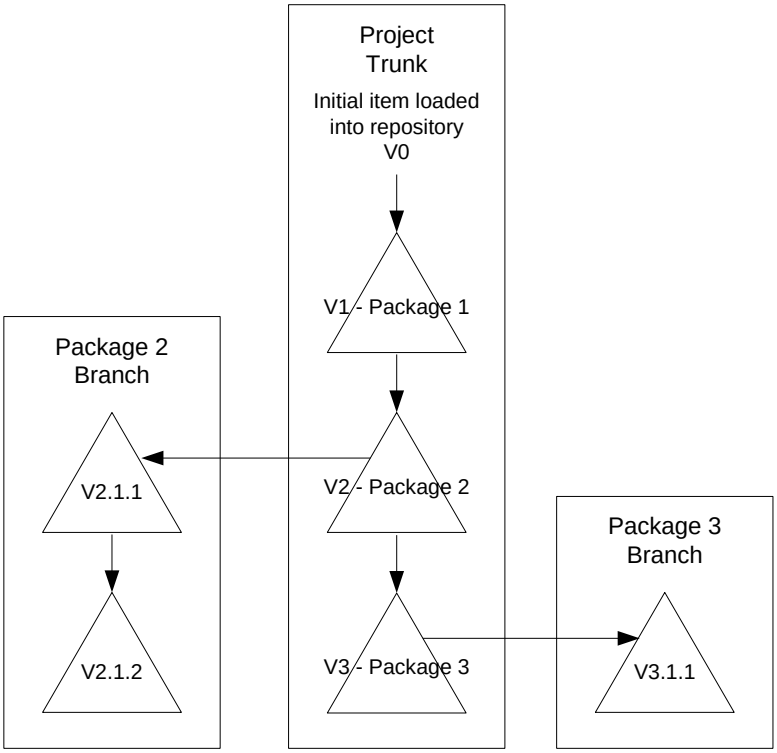
Chapter 7: Understanding Versions

This section contains the following topics:

Versions

Items are stored in the CA Harvest SCM repository. Each item in a repository consists of *versions* of the item. Projects use items to create versions on *trunks* of the delta tree that extend from the repository. These trunks can support versions on multiple *branches*. Branches are independent delta chains that extend from versions on a project trunk.

The following diagram depicts a single item that has been updated in multiple packages. The triangle represents a *delta*, or change, made to the original item since it was added to the repository. Each delta represents a new version.



Version Attributes

When a version is created, CA Harvest SCM maintains extensive information about the change. CA Harvest SCM retains the following version attributes:

- Name of the user making the change
- Date and time the change was made
- Name of the package associated with the change
- A change description entered at the time the update was made
- A tag indicating whether the version is the result of a merge, check-out, or remove item process
- Information about the host file checked in to create the delta, including the file modification time stamp, file size, and user access

Version Tags

Version tags identify the status of items and versions. The different tags denote different versions:

- Normal (N)
- Merged (M)
- Removed, or logically deleted (D)
- Reserved (R)

Branch Versions

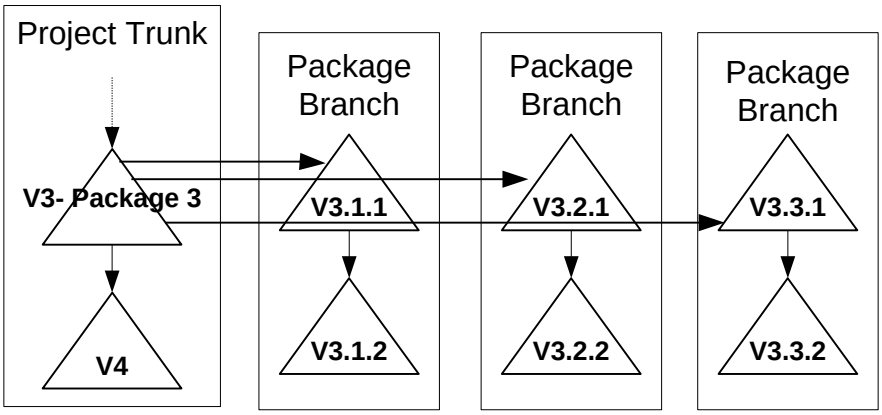
Often the same item needs to be updated by different users at the same time in the same project. CA Harvest SCM supports concurrent development by letting users make changes on branches off the project trunk. *Branches* are independent version chains that extend from trunk versions; a new version on the trunk does not affect branches. The check-out process *concurrent update* mode creates a reserved version on a package branch for each item. Branch versions can be checked out for update or concurrent update; both modes of check-out create a reserved version on the same branch.

Multiple branches can be created off the same trunk version and any trunk version, not only the latest, can be checked out to a branch. Using either the update or concurrent update mode of check-out, branch versions can be checked out and modified. Both modes create a reserved branch version that is replaced when the file is checked in. The package specified for the check-out must be the same as the package that created the branch.

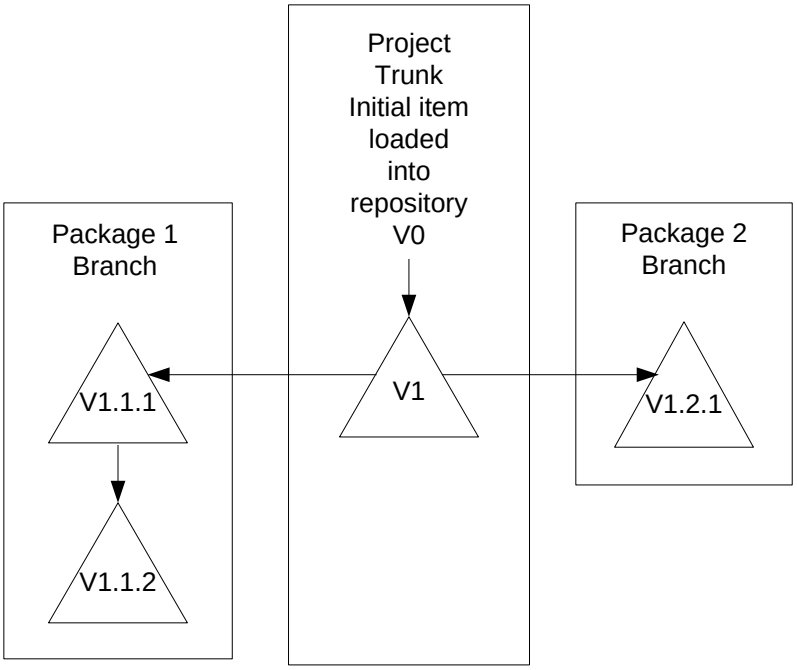
Version Numbers

The first digit of a branch version number refers to the trunk version the branch was created from, the second to the branch number, and the third to the version sequence. For example, version 3.2.1 refers to the first version on the second branch from trunk version 3.

In the following example, version 3 of an item has been checked out for concurrent update three times. When the files are modified and checked in, branch versions 3.1.1, 3.2.1, and 3.3.1 are created. When changes are made to version 3.1.1 on the first branch, and the changes are checked in, the version created on the branch is 3.1.2.



In the following diagram, the triangle represents a *version*, or change, made to the original item (V0) as loaded in the repository. Each triangle represents a new version. Version 1 (V1) of an item is checked out for concurrent update. Using Package 1, the item is modified and checked in to the branch, and a version 1.1.1 is created. Using Package 2, checking out version 1, modifying it, and checking it in to the branch creates a version 1.2.1.

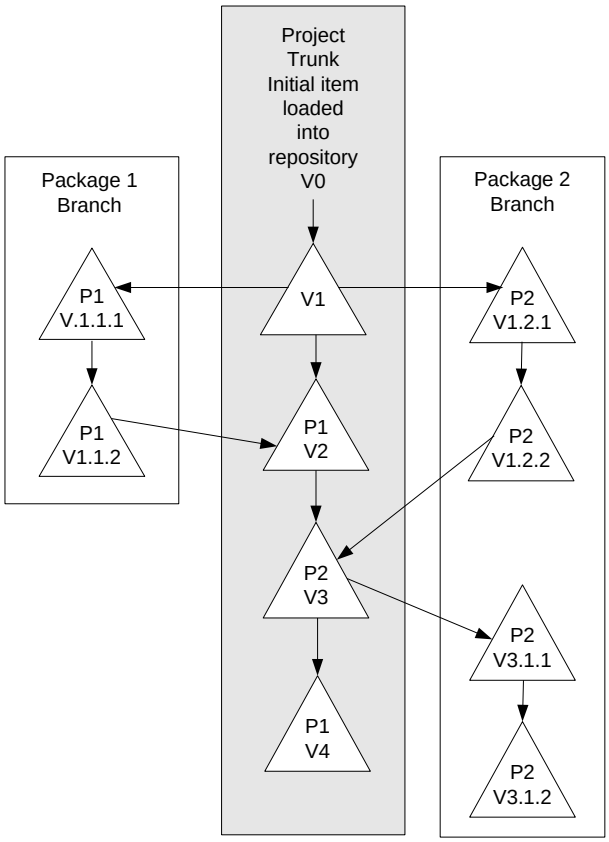


Package and Branch Version Relationships

All version changes must be made in reference to a package. The following rules apply to branch versions and their packages:

- If a package has versions of an item on an unmerged branch, the package cannot be used to create versions of the same item on the trunk or on another branch.
- Each branch is associated with only one package. A branch can have one or more versions, but all versions on a particular branch belong to the same package.
- When a package that contains a branch version is merged, the merge creates another trunk version and, subsequently, a branch version can be created.

The following diagram shows package and branch relationships. The package P1 is merged to the trunk to create a version V2, and the package P2 is merged to the trunk to create V3. Because P2 is merged to the trunk, you can begin a new branch using the same package P2. At the same time, because P1 has been merged to the trunk, you can use P1 to check out for update to create a version V4 on the trunk. Note that until P2 is merged back to the trunk, it can be used only to check out for update from its branch.



Views and Versions

Item and item path versions are visible to users only through views. The View folder for a state in the Workbench displays versions in the project baseline view and versions in the state's working view.

When a user checks in a new version of an item or item path, that new version becomes visible only in the states that use the same view as the state in which the version was checked in. As a package moves through a life cycle, the versions associated with it become visible in a different view only when the package reaches a state in that view. For example, if you check in version 2 of file1.c from a coding state, and a test state shares the same view as the coding state, version 2 of file1.c is visible in both states. However, version 2 is invisible in a release state because that state does not share the same view. For version 2 to show up in the release state, the package that was used to create the version must be promoted to the release state.

Locate Package in the Explorer View

You can locate and highlight the package from which the version was created from the Workbench / Eclipse plug-in. This option is useful when there are multiple branch versions and when the changes are included across several packages for the same version.

Follow these steps:

1. Navigate the Workbench / Eclipse plug-in to the project you want to locate the package in which the version exists.
2. Right-click any version and select Locate Package in Explorer View.

   CA Harvest SCM navigates to the package in the explorer view and selects it.

   Note: This option is disabled for the baseline versions.

Chapter 8: Using Check-In and Check-Out

Note: The instances of Linux in this section refer to both the Linux and zLinux operating environments.

This section contains the following topics:

Client Directories and View Paths

As you move about in the client directories or in the repositories, your path is defined by your current position in the hierarchy. A *view path* is a location in the CA Harvest SCM repository. The project and state you select define which view paths, items, and versions you can see.

Root Directory and Root View Path

CA Harvest SCM uses the concept of a client root directory and a view path root to let you control how your client directory path and view path change. In the Find Version dialog and in the check-in and check-out process execution dialogs, you can set the client root directory or view path root to a certain location. This lets you synchronize the check-in and check-out processes with the client directory and view path structure already established.

Two distinct roots can be defined:

- *Client Root Directory* when viewing files. By selecting your client root directory, you are setting a default client directory that is inserted in the client root directory field during subsequent check-in processes.
- *View Path Root* for repository paths, when viewing items and versions. By selecting your view path root, you are setting a default view path that is inserted in the view path root field during subsequent check-out processes.

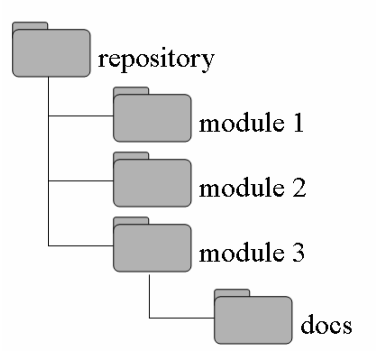
Internal and External Directory Structures

An administrator creates a repository and loads it with a set of directories and files that, typically, make up an application. The administrator duplicates an external application directory structure in the repository and in the baseline of any projects that use it. In the repository structure, the top-level is the repository name and the application directory paths lie beneath it.

Typically, individual developers maintain a client directory structure that is similar to the repository structure, and use it for updating application items and building the application for testing.

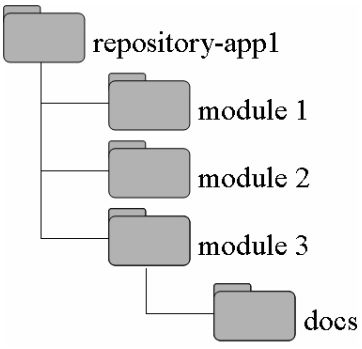
Example: Load a Client Directory Structure into the Repository

This illustration is an example of a client directory structure in an application named app1:



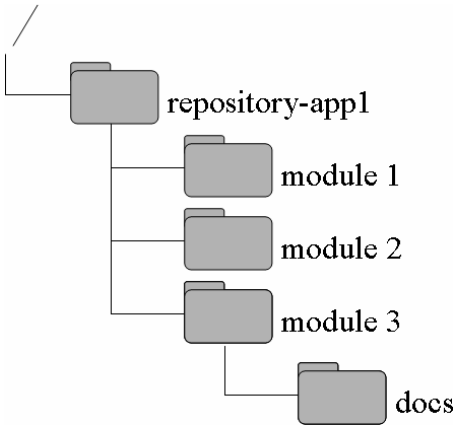
Example: Duplicate an Application Structure in the Repository

This example shows that when this application is loaded into a repository named repository-app1, the application structure is duplicated in the repository. When users access CA Harvest SCM to look at items in a view that includes this repository, the repository view paths look like the following illustration:



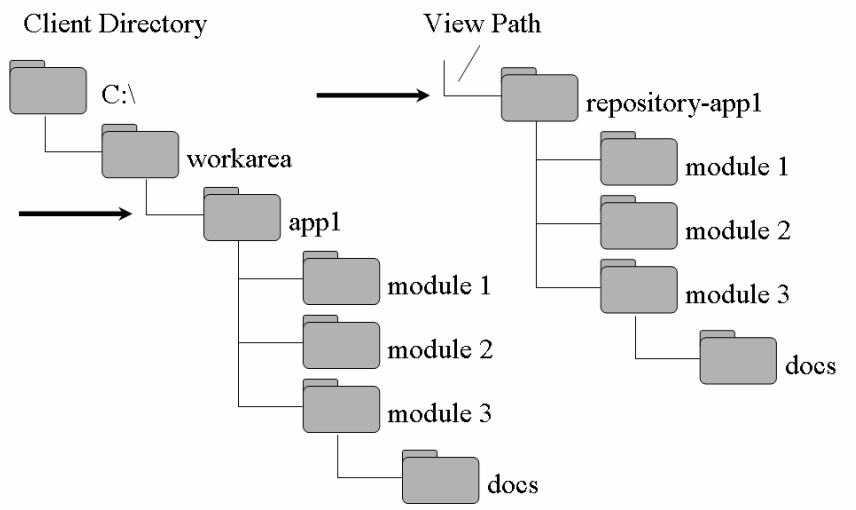
Example: Mirror the Application Structure on a Windows Client

Assume that you have been working on this application, and it has working directories on a Windows client mirroring the application structure. The following illustration shows what your client work area would look like:



Example: Synchronize the Directory Structure Using Roots

This example shows how roots synchronize the internal and external directory structure. In this case, the view path root is set to /repository-app1 and the client directory root is set to c:\workarea\app1. These two roots mark the point of synchronization between items inside the repository and their corresponding files on the client directory.



Browse for Agent Folder

You can browse for a remote agent folder on any connected agent from the check-out or check-in process dialog.

Follow these steps:

1. Click the browse button for the check-out To field or check-in From field on their respective process dialogs.

   You can perform a check-out or check-in from your local file system or from a remote agent. When you select the Browse for agent option, a dialog appears, which lets you locate and select a remote agent.

2. Expand the agent node you want to browse.

   Note: If an agent node is disabled in the tree, the agent is unreachable from the broker and cannot be used to select a remote folder.

3. Expand the agent tree and selected your folder you want. Click OK.

   The Browse For Folder dialog closes and your selection populates the check-out or check-in process dialog field.

File Permissions

CA Harvest SCM assumes a long-term relationship between a set of user working directories and a repository structure. In these working directories, CA Harvest SCM uses file permissions to indicate files not available for update. File permissions are altered during check-out and check-in.

Important! Because permissions are managed according to the state of corresponding items in CA Harvest SCM, we recommend that users do not alter file permissions. Altering file permissions could lead to unexpected results during check-out or check-in.

File permissions are handled differently depending on the client's operating system. For example, UNIX and Linux make greater use of file permissions than Windows operating systems. On Windows operating systems, CA Harvest SCM uses only one level of access permission, represented by the read-only attribute.

- On Windows client systems, checking out a file in browse or synchronize mode creates a file on the client file system marked as read-only. This indicates that the file cannot be modified and checked in because these modes do not create a reserved version in CA Harvest SCM. When checking out for update or concurrent update, the file is given normal (write access) file status.

- On UNIX and Linux, when a file is checked in to CA Harvest SCM, information about file permissions is maintained in the CA Harvest SCM database. When the corresponding item is checked out for update, the same file permissions are set on the client file, with one exception. Any time a user checks out an item for update, write access is granted so that changes can be made to the file.

  For example, consider a UNIX or Linux file with the following permissions:

  ```
  r-xrwxrwx
  ```

  When this file is checked in to CA Harvest SCM, the file permissions are stored. Upon check-out for update, the permissions are:

  ```
  rwxrwxrwx
  ```

  When a file is checked out for browse, CA Harvest SCM does not use the exact information stored with the file. It places the file in the directory without write permission for user, group, and other. The permissions now look like this:

  ```
  r-xr-xr-x
  ```

Display File Type Attributes (Binary/Text)

You can display file type attributes (binary or text) in the following ways:

- Click a file version in the Explorer View.

  The file type displays as an entry in the Stored As column of the Lists View.

- Right-click a file version, and select Properties from the shortcut menu.

  The file type displays in the Stored As field in the Properties View.

File and Item Case-Sensitivity

CA Harvest SCM check-in and check-out behavior differs among UNIX, Linux, and Windows operating systems because of the case-sensitive behavior of file systems on these operating systems. For example, the file File.c is not the same as FILE.C on a UNIX or Linux operating system, but it is the same for a Windows or z/OS operating system.

Case-sensitivity depends on the operating system of the file agent. If you are not using a remote file agent, your client program acts as the file agent.

- If you check in and check out files locally (not with a remote file agent), case-sensitivity depends on the local operating system (where the client program is running).
- If you check in and check out files remotely (with a remote file agent), case-sensitivity depends on the remote operating system (where the file agent is running).

With UNIX or Linux file agents, CA Harvest SCM is case-sensitive for file and item names. If you check out the item file.c using a UNIX or Linux file agent, you must check in the file with this name; if you change the case of the name to FILE.C on the file system prior to check-in, it is checked in to the repository as a new item.

With Windows file agents, CA Harvest SCM is not case-sensitive. If you check out the item File.c using a Windows file agent, you can successfully check in this item as FILE.C, file.c, or as any other case.

Signature Files

Checking in or checking out an item automatically updates information in a signature file maintained by CA Harvest SCM in each directory containing CM-related files. To update or create the signature file, you must have write access to the directory from which you are checking in items.

Note: For more information about signature files, see the *Command Line Reference Guide*.

After checking in a file from a read-only device (for example, a CD-ROM), the CA Harvest SCM output log shows error message E02020053 and informational message I00020100. These messages display that the signature file was not opened but the file was checked in. Error message E02020053 correctly reports that the check-in process was unable to create a signature file on the read-only device. The signature file cannot be created because the read-only device is not writable. Informational message I00020100 correctly reports that the check-in process was successful.

Note: For more information about the messages, see the *Messages Guide*.

Partitioned Data Sets

You can check in and check out the following:

- Partitioned data set (PDS)
- Partitioned data set/extended (PDS/E)
- All other Data Control Block (DCB) for PDSs

  Note: PDSs with an undefined record format are not supported. All other Data Control Block (DCB) for PDSs are supported.

CA Harvest SCM does not list migrated data sets. To check in from migrated data sets or check-out to migrated data sets, you must first recall the data sets using the appropriate commands for your system (for example, HRECALL, RESTORE, DRESTORE, $RA, and so on).

You *cannot* check in and check out the following:

- Sequential data sets

  Note: To check in sequential data sets, copy the sequential data sets to a PDS or PDS/E, and then check in the PDS or PDS/E.

- Load libraries
- Variable Blocked (VB) data sets in binary mode

When checking out a PDS, the check-out dialog dynamically changes after selecting a PDS to populate the To field. The dialog adds an additional option, Remove MVS extension. The Remove MVS extension removes the last-level qualifier from the selected data set. This option works with the To field. If an item has an extension and is checked out to a PDS, the item name is the PDS member name and the extension is added to the data set as the last-level qualifier. If the data sets already exist with the appropriate last-level qualifier, select the Remove MVS extension option prior to check-out. The To field destination name reflects the removal.

The agent applies the item extension to the last-level qualifier at check-out. This option lets you check out multiple items with different extensions to multiple files on a single check-out. If the PDS member does not exist, it is created at check-out. The PDS must exist prior to check-out.

z/OS File Conversion

The z/OS agent translates text files to a unified format at check-in and translates text files to the z/OS format on check-out. Files are read in and written out using stream input/output.

Note: For Hierarchical File System (HFS) files, CA Harvest SCM translates from Extended Binary-Coded Decimal Interchange Code (EBCDIC) to ASCII and performs stream input/output as defined in the *z/OS C/C++ Programming Guide*.

The z/OS agent explicitly performs ASCII to EBCDIC translation. The ASCII code page is set to ASCII code page 437. The EBCDIC code page is set to EBCDIC 037.

To avoid compilation errors on C++ and JAVA text files, the z/OS agent translates the following characters to EBCDIC 1047 code page format: “ [“,”]”, and “ ^”. This means that text files containing these characters are always checked out using the 1047 code page format as shown in the following hexadecimal character table. Files containing EBCDIC hexadecimal characters BA, BB, or B0 are always checked out with EBCDIC hexadecimal characters BD, AD, 5F, respectively.

ASCII HEXADECIMAL	EBCDIC 037 HEXADECIMAL	EBCDIC 1047 HEXADECIMAL

	ASCII HEXADECIMAL	EBCDIC 037 HEXADECIMAL	EBCDIC 1047 HEXADECIMAL
[	5B	BA	BD
]	5D	BB	AD
¬	AA	B0	5F

Check-Out Process

The check-out process lets you copy versions of items under CA Harvest SCM management to external directories where you can update or browse the items. Alternatively, you can reserve the items without copying the data to a client directory. The reserved version placeholder created by the check-out is associated with the package you select.

When you select a tagged version for check-out, CA Harvest SCM automatically eliminates that version from your check-out selection and it does not populate the check-out list. For example, if you select three versions including one that has a reserve tag, and select check-out, only the two untagged versions populate the check-out list.

If you select a folder, it must contain at least one version or the check-out cannot continue. In the case of recursive check-out, the selected folder or its subfolders must contain at least one version or the check-out cannot continue. In both cases, an error message displays.

You can check out an item to a remote location. Access to data files in remote locations requires WRITE permission based on the registered user name and password for the remote computer.

Check-Out Rules

The following rules govern the check-out process:

- **Access**—To check out an item for browse or synchronize, you need at least view access to the item. To check out an item for update or concurrent update, or to reserve an item, you need edit access to the item.
- **Duplicates**—You can check out any version of an item, not only the latest. You can check out only one version of an item at a time to the same directory on the file system.
- **Package**—A package in the current state is required to check out an item for update, concurrent update, or reserve only. A package can only create one reserved version per item, either on a branch or on the trunk.

- **Version**—The only trunk version that you can check out for update is the latest. You can only check out a trunk version for update if it is not reserved by another package. This version is available for updating even if it does not yet appear in the current view. To check out an earlier trunk version for updating, you must use the concurrent update mode.
- **Updating a Branch**—To check out and modify a branch version, you can use the update or concurrent update mode of check-out. Both create a reserved branch version that is replaced when the file is checked in. The package specified for the check-out must be the same as the package that created the branch.
- **Unmerged Branches**—A package can only have one unmerged branch version per item. A second attempt to check out an item for concurrent update fails if an existing unmerged branch is found for that item with that package. The current branch version should be either merged or deleted before attempting to use the concurrent update mode of check-out again for the same item with the package.
- **Tagged Versions**—Versions tagged as merged (M) or removed (D) cannot be checked out. A reserved version (R) can be checked out only for browse mode; its content is the same as the previous version.

Check-Out Paths

The From and To check-out paths work together to help you synchronize the internal (repository) and external (file system) structures. You typically use the From and To paths to establish a point at which paths in the repository mirror working directories on the client.

The view path displayed in this field is determined in one of the following ways:

- If you use the Find Version dialog to select versions, the most common path in the version list populates this field. When your list changes, the field is updated to display the most common path.
- If you select a View folder or an item or version in a View folder, and execute the check-out process, the view path for the selected item is used.
- When you select an item for check-out, CA Harvest SCM automatically selects the item's latest versions without tags to populate the check-out list. If the Versions list contains folders or versions outside the From view path (not a descendant of the view path), the check-out cannot continue. An error message cautions you about unexpected results.

File Date and Time

Although CA Harvest SCM stores the file modification date and time, the date and time associated with the file created when an item is checked out reflects the current date and time. This algorithm supports building an application by ensuring that the source files are more recent than object files.

Replace Read-Only Files Option

The Replace read-only files option controls whether the checked-out files should replace existing read-only files on the file system. This option lets you replace files that you previously checked out as read-only or checked in. If a corresponding file does not exist on the file system, the item is checked out regardless of the value for this option.

Files that are not read-only on the Windows file system, or that have write permission on the UNIX or Linux file system, are never overwritten. CA Harvest SCM assumes that such a file has been checked out for update and not checked in yet. Overwriting such a file can cause you to lose unsaved changes.

On UNIX and Linux, file ownership affects the results of the read-only files option. When used alone, the Replace Read-Only Files option overwrites read-only files only if the user executing the check-out is the owner of the files being replaced.

Check Out Versions

The check-out process dialog lets you copy versions of items under CA Harvest SCM management to external directories.

Follow these steps:

1. Navigate the Workbench to the package you want to use to check out versions.
2. Right-click the package, and select *check-out process* from the shortcut menu.

   The *check-out process* dialog appears.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Select a Mode option:

   Update

   Copies selected versions to the destination client directory and creates a reserved version on each item's trunk, letting you check in the corresponding files. The permission on a read-only file is changed to normal (write access) when you use this mode of check-out.

Browse

Copies the items to the destination directory but does not let you check in the files. The read-only attribute is set on files that are checked out for browse mode.

Synchronize

Identifies the versions of the files in the client file system by using the signature file. Versions are checked out only if the signature file shows:

- Client files that differ from their corresponding CA Harvest SCM versions.
- Client files with timestamps that differ from their corresponding timestamps in the signature file.

Items are checked out in a read-only mode.

Note: The check-out for synchronize mode is useful in the build process. This option compares the version on the file system with the version in the database. If the database version contains a newer timestamp, the file is checked out.

Concurrent Update

Copies selected versions to the destination client directory and creates a reserved version on a branch for each item. All updates accumulate on this branch. The permission on a read-only file is changed to normal (write access) when you use this mode of check-out.

Reserve Only

Creates a reserved version with a reserved tag (R) on each item's trunk. You can check in corresponding files, but cannot move any data to external directories.

4. Select an option to specify the way for checking out items to client paths:

Preserve View Path and Structure

Checks out all selected items to corresponding client directories, if they exist. If the directories do not exist, an error message displays and the items are not checked out.

Note: If you select a single file for check-out, the checkout is successful; in spite of the path not matching with the new view path.

Preserve and Create Directory Structure

Checks out all selected items to corresponding client directories and creates any client directories that do not currently exist.

All Files to the Same Client Directory

Places all selected items directly beneath the specified client directory, ignoring the path structure in the repository.

Note: The options work with the Include subfolders option on the Find Version dialog.

5. Specify a view path in the repository from which items are checked out in the From field.

6. Specify the destination on the file system for the checked-out files in the To field.

   - If either of the preserve structure check-out options is selected, the check-out specifies a path and this specification is appended to the file system directory.
   - If the files being checked out include a directory specification, as in a recursive check-out, this directory specification is appended to the client path specified.

   Agent

   Displays the remote computer name if the client path is on a remote computer.

   Add Remote Agent Connection

   Lets you add the new remote agent connection whose values populate in the Agent name and To fields.

   For more information about adding a remote agent connection, see the Connect to a Remote Agent (see page 44) section.

   Note: When you connect to multiple agents on the same computer and check-out from the Workbench, CA Harvest SCM identifies the agent with a unique combination of agent name, port number, and the initial directory. This selection auto populates the To field with the initial directory of the agent computer.

7. Select versions to check out:

   Snapshot

   Specifies a snapshot from which you can select versions to check out. This field is enabled when a snapshot exists in the view in which you are executing the check-out process.

   Note: You can select versions from the snapshot and can return them to the check-out process execution dialog.

Versions

Lists the versions that you have selected for check-out. If you have selected a version before selecting the check-out process, the version appears in the list. If you have not selected a version before invoking the check-out process, you can add more than one version by clicking the Select Version.

8. (Optional) Select check-out options:

   Replace read-only files

   Controls whether the checked-out files must replace existing read-only files on the file system.

   Recursive

   Selects all versions in the selected folders *and* subfolders in the Versions list. The versions in the subfolders are checked out *only* if you select this option.

   Note: This check box is disabled if no folders are selected.

   Create empty directories

   Creates directories that contain no items during a recursive check-out. This option works with the Include subfolders option in the Find Version dialog.

   Include versions from package

   Selects the latest versions from the trunk and branch that are associated with the package context.

9. (Optional) Click Note to view information about the process.
10. Click OK.

    The versions are checked out.

Repository Cache Use on a Remote Site

If your network access to the CA Harvest SCM server is relatively slow (for example, on a WAN or using VPN), your check-out performance may degrade when performing large recursive check-outs. You can use the optional Repository Cache on Remote Site function in these circumstances to improve your check-out performance. Repository Cache stores version data files locally, accessed using a CA Harvest SCM remote agent or directly in a local directory.

Note: For details about Repository Cache on Remote Site, see the *Administrator Guide*.

Check-In Process

The check-in process lets you copy files from the client directory system to CA Harvest SCM, creating items or updating existing items that have changed. To check in an existing item, it must be reserved by a package. If the item was originally checked out for concurrent update, the check-in process updates the current package's branch. If the item was checked out for update or reserve only, the check-in process updates the trunk for the item.

You can check in files from remote locations. For example, you can check in files on an AIX operating environment to a CA Harvest SCM repository managed by a server process that runs on a Solaris operating environment where the file cannot be accessed through NFS. Access to data files in remote locations requires a user name and password for the remote computer.

Check-In New Item and Item Path Rules

The following rules apply to the check-in of new items and item paths:

- When you check in a new item or item path to the trunk, it is version 0.

  Note: This is the same version designation as items added to a project when you use the configure baseline function.

- When you check in a new item or item path to the branch, it is version x.1.1. When you merge this branch version to the trunk, the version becomes a normal version that is version 0.
- An item can be checked in to a branch using more than one package. In this case, the second version becomes x.2.1, the third x.3.1, and so on. This is the same convention as when you check out the same item in different packages; the only difference is the x. designation.
- When you check in a new item or item path to a view path, it is version 0 or if it is a branch version, x.1.1. If you check in the same item or item path to a different view path in the same project, it is version 0, or if it is a branch version, x.1.1.

- You can check in only one new item path to a branch per project. This means that you cannot create the same item path using a different package, nor can you rename an item path to that new path name. You can check in new items to the new item path in the same package.
- You can check in the same item name with the same item path in more than one package when you check in the item to a branch. You cannot check it in to the same package more than once.
- If you check in and create items and item paths to the trunk using the Preserve and Create Path Structure option, the item paths are immediately visible in all working views of the project. The items are initially visible only in the working view to which they were checked in.
- If you check in and create items and item paths to the branch, neither the items nor the item paths are visible in the data view until they are merged to the trunk. However, you can use a package's versions list or set the package perspective to show the branch versions, for example, by clicking the Versions node of the package.

Check-In Client Path

The check-in client path is determined in one of the following ways:

- If you use the Find File dialog to select files, the most common path in the file list populates this field. When your file list changes, the field is updated to display the most common path.
- If you select a file in the WorkAreas view and execute the check-in process, that context is used in the check-in dialog.

Check In Files

The check-in process lets you copy files from the client directory system to CA Harvest SCM, creating items and item paths or updating existing items that have changed.

Follow these steps:

1. Navigate the Workbench to the state in which you want to check in files.
2. Right-click the state, and select *check-in process* from the shortcut menu.

   The *check-in process* dialog appears.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.

3. Select a mode:

   Update and Release

   Checks in the file to the repository and the reserved tag is removed from the corresponding item. The permission on the client file is automatically set to read-only so that changes cannot be made to it until it is checked out again.

   Update and Keep

   Updates the item in the view, the current package keeps it reserved and the file permissions unchanged so that you can make more updates.

   Release Only

   Does not check in or update the item and the item is no longer marked as reserved for the current package. The permission on the client file is automatically set to read-only so that the file cannot be changed until it is checked out again.

   Release Only does not require the local file system file to exist.

   Note: Selecting Release Only mode implies an item filter of Existing Items Only because this operation requires existing reserved items in the repository.

4. Click the button next to the Package field to open the Select a Package dialog.

   The Select a Package dialog appears.

5. Select a package to associate with updates to an item, and click OK.

   Your package selection appears in the Package field.

6. Establish a point at which paths in the repository mirror working directories on the client:

   From

   Specifies the client file directory. You can click the button next to this field to open the Browse For Folder dialog that lets you browse through available paths, and select a location.

   Machine

   Displays the remote computer name if the client path is on a remote computer.

   To

   Specifies the path location in the destination view to check in files. To change the path, click the button next to this field to open the Repository Path Selection dialog that lets you browse through available paths, and select a location.

   Note: If the files being checked in include a directory specification, as is the case during recursive check-in, this directory specification is appended to the view path specified here when you specify a preserve structure check-in option.

   The internal (repository) and external (file system) structures are synchronized.

7. Specify the way to check in files to view paths:

   Preserve Directory Structure

   Checks in the selected files to repository view paths with names that correspond to their client directory location if these view paths currently exist.

   Preserve and Create Path Structure

   Checks in selected files to paths with names that correspond to their client directory location and creates any view paths that do not currently exist.

   All Files to Same View Path

   Checks in all selected files to the same path in the destination view and ignores the client directory structure.

   The way you check in the files to view paths is set.

8. Specify how to filter the files being checked in:

   New or Existing Items

   Checks in files if they are reserved by the package or did not previously exist in the repository. A corresponding item does not need to exist in the current view; however, if it is in the current view, it must have a reserved version in the current package.

   New Items Only

   Limits the check-in to files that do not have corresponding items in the current view. If the item has been removed from the current view, you can check in the file using this filter.

   Note: If the corresponding item was renamed in the current project, you cannot check in the file. You must rename the file before you can check it in.

   Existing Items Only

   Limits the check-in to files that have corresponding items reserved by the package. Any files without corresponding items are skipped. This filter helps prevent the existence of unwanted files, such as temporary files or templates, in your repository.

   The check-in filter is set.

9. Select a Placement option:

   Trunk

   Creates versions on the trunk.

   Branch

   Creates versions on a branch.

   The location of the checked-in versions is set.

10. Specify files to check in.

    Note: If you select files in the client directory before invoking the process dialog, the files populate the Files field. You can also select files by clicking Add (the plus [+] sign) to open and use the Find File dialog.

11. Select options, and enter and view information about the check-in:

    Delete files after check in

    Deletes from the client file system all files successfully checked in with the mode Update and Release, or Release Only. Deleted files are also removed from the signature file (harvest.sig) on the client.

12. (Optional) Click the tabs to enter and view information:

    Comment

    Specifies comments.

    Note

    Provides notes about the process.

    Note: If the Enforce Change Description option is selected in the check-in Properties dialog, you cannot execute a check-in successfully without entering a description.

    Click OK.

    The files are checked in.

More information:

Find Version Filter Combinations (see page 241)

Drag-and-Drop and the Files List

You can populate the Files list of the check-in process dialog using any of the following methods:

- Selecting one or more files or directories from Windows Explorer and dragging them to the Files list.
- Selecting files from multiple source directories by performing multiple drag-and-drop operations from different source locations.
- Selecting an entire directory and its subdirectories (recursively) to be added to the list of files to be checked in.

When dragging files from multiple directories, the client path is updated each time the files are dropped into the Files list. With each new drag-and-drop, the client path is updated and the Files list is appended. The check-in process uses the path set by the last drag-and-drop operation for the check-in client path.

Check In Reserved Versions

The package check-in lets you check in reserved versions but does not let you specify the view path or client path. The paths you used during check-out are retained and used for the package check-in. For example, you could associate various versions from different view paths with one package and then check out the package to your client directory. If you update the versions, executing a package check-in places the changed versions in their original view path locations.

Follow these steps:

1. Navigate the Workbench to the package that has the reserved versions you want to check in.
2. Right-click the package, and select *check-in process* from the shortcut menu.

   The *check-in process* dialog appears and displays the versions associated with the package.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Select a mode:

   Update and Release

   Checks in the version to the repository and the reserved tag is removed from the item. The permission on the client file is automatically set to read-only so that changes cannot be made to it until it is checked out again.

   Update and Keep

   Updates the item in the view, the current package keeps it reserved and the file permissions unchanged, so that you can make more updates.

Release Only

Does not check in or update the item, and the item is no longer marked as reserved for the current package. The permission on the client file is automatically set to read-only so that the file cannot be changed until it is checked out again.

This mode does not require the local file system file to exist. If the file does not exist on the local file system, right-click the reserved version, and select the check-in process.

Note: Selecting Release Only mode implies an item filter of Existing Items Only because this operation requires existing reserved items in the repository.

4. (Optional) Click the tabs to enter and view information:

Comment

Specifies comments.

Note

Provides notes about the process.

Click OK.

The reserved versions are checked in.

Use -ciignorehostname to Check In Reserved Versions

To perform a version-based check in of all your files from the local computer irrespective of where you checked them out, specify the -ciignorehostname option in HClient.arg. Restart the client for the new settings to take effect.

Important! When you specify the -ciignorehostname option, the version-based check in from the remote agent is disabled. Therefore, use the file-based check in to check in from the remote agent.

Note: For more information about HClient.arg, see the *Implementation Guide*.

How to Fix Errors During Check-In and Check-Out

For the check-in and check-out processes only, you can view errors in the Log View and in a separate Error List dialog. The Error List automatically opens during the check-in and check-out process execution if fatal errors occur.

After viewing any errors on the Error List dialog, do the following:

1. Return to the check-in or check-out process dialog without closing the Error List.
2. Fix the errors.
3. Execute the check-in or check-out process again, without closing the process execution dialog and without checking the Log View.
4. Close the Error List.

Chapter 9: Working in the WorkAreas

This section contains the following topics:

WorkAreas

A WorkArea is a user-defined location on the file system that the Workbench monitors. A WorkArea is associated with a specific CA Harvest SCM repository, project, state, and folder (view path) and shows the objects in a hierarchical structure.

To work on files and folders you must add the corresponding folder (view path) to your WorkArea from the CA Harvest SCM repository. Adding a folder to the WorkArea imports the files and folders from the CA Harvest SCM server to your local file system location. When a WorkArea is initially created, a .project file is created that describes attributes of the WorkArea and it is an internal file. When you commit the WorkArea by checking it in for the first time, the .project file is checked in. Adding the same folder to your WorkArea checks out the .project file, ensuring that the correct metadata is used.

You can edit and delete file resources in your WorkArea. When you save changes to your WorkArea, these changes are saved on your local computer.

The WorkAreas view lets you do the following:

- Work on a local area on the desktop with an anchor set to a point in the data view.
- Perform check-out, undo check-out, and check-in and commit to repository on projects, folders, or file nodes in a WorkArea. These actions are applied recursively to all files below the selected object.
- Delete files and folders. When you delete a folder, the action is applied recursively to all files and folders below the selected folder.
- Update files or folders from the repository to update your WorkArea files or folders with the latest trunk version in the repository.

Synchronizer View

The Synchronizer view is a hierarchical structure of a folder that shows files and indicates if the changes are incoming, outgoing, or conflicting.

The Synchronizer view lets you do the following:

- Work on a local area on the desktop with an anchor set to a point in the data view.
- Perform Commit to Repository on projects, folders, and files in the WorkArea. These actions are applied recursively to all files below the selected object.
- Get files or folders from the repository to update your WorkArea files or folders with the latest trunk version in the repository. All files or folders that you get from the repository must be incoming; that is, they must have left-directional arrows.

- See the status of managed files.
- Evaluate incoming and outgoing changes.
- Manage synchronization between the WorkArea and the CA Harvest SCM repository.
- Compare refactoring changes between local files and folders with their corresponding repository items and item paths, respectively.
- Add new resources folders and files to the WorkArea view and change the existing managed resources which you can later commit to the CA Harvest SCM repository. Outgoing files will have right-directional arrows.

Status Indicators

Indicators (symbols) in the WorkAreas/Synchronizer view identify the status of files or folders as follows:

- Cylinder—Managed by CA Harvest SCM.
- Plus sign (+)—Modified.
- Asterisk (*)—New (not managed by CA Harvest SCM).

  Note: An asterisk to the left of a folder name indicates that the folder is new or a file below it is new.
- Check mark and right angle bracket (>)—Checked out.
- Left arrow (<-)—Incoming mode (changed in the repository, but not on the client computer).
- Right arrow (->)—Outgoing mode (changed on the client computer, but not in the repository).
- Double arrow (<->)—Incoming/outgoing mode (changed on both the client computer and in the repository). Double-headed arrows indicate a conflict between the WorkArea and the repository.
- Plus signs (+) on top of the arrows—New for outgoing changes(not managed by CA Harvest SCM). For example, a right-directional arrow with a plus sign is a local file that has not previously been checked in. For incoming changes it indicates that the file is new and managed by CA Harvest SCM, but just not yet managed in the WorkArea.
- Minus signs (-) on top of the arrows—Removed.

Note: In the Lists view, you can view version tags that identify the status of items.

Examples: Indicate Status

- A check mark and a right angle bracket next to a file indicates that the file is checked out and may need to be checked in to the repository (outgoing change).
- A right angle bracket next to a shared directory file indicates that a file in that directory needs to be checked in to the repository (outgoing change).
- A right angle bracket and plus sign next to a file indicates that the file is checked out and has been modified.
- A right arrow with a plus sign indicates a local file that has not been checked in. Double-headed arrows indicate a conflict between the WorkArea and the repository.
- A plus sign on top of a right arrow indicates a local file that has not previously been checked in and is not managed by CA Harvest SCM.
- A plus sign on top of a left arrow indicates that the file is new and managed by CA Harvest SCM, but just not yet in the WorkArea.
- A minus sign on top of a right arrow indicates that the file was removed from the WorkArea and when the file is committed to the repository, the item will be removed from the repository.
- A minus sign on top of a left arrow indicates that the version was removed in the repository. When you perform a Get from the repository, it will be removed from your WorkArea to be synchronized with the repository.

Refresh Checked-Out Status

The Refresh checked-out status action lets you query the repository to determine if selected objects have reserved (R-tag) versions and then update your WorkArea status indicators accordingly. If you have a file checked out (reserved) and someone else deletes the R-tag version from the repository, the repository does not have the version reserved but the WorkArea still shows its status indicator as reserved (checked out). You can resolve this mismatch in your WorkArea by using the Refresh checked-out status action.

To update the WorkArea status indicators because of changes in the repository, right-click the node for which you want refreshed status indicators, and select Refresh checked-out status from the shortcut menu.

Note: Selected nodes include those explicitly selected and the descendents of the selected nodes.

The WorkArea node is refreshed with the latest status indicators as follows:

- Every selected node in the WorkArea with an R-tag in the repository is marked as checked-out.
- Nodes in the WorkArea that show as checked-out are set to Normal status if an R-tag version is not found for an item.

How to Use the WorkAreas View

A WorkArea represents your local working directories. The project name is the root of a WorkArea, and a project is considered to be one unit of work.

Follow these steps:

1. Add the project folder that corresponds to your WorkArea from the CA Harvest SCM repository.
2. Use the Replace With action to replace files in your WorkArea with the latest repository versions.
3. Modify files in your WorkArea.
4. Save changes to your files for the WorkArea project.

   The changes are saved on your local computer.

Example: Use the WorkAreas View

This example describes a typical scenario for using the WorkAreas view.

1. Navigate the Explorer View and use the Add WorkArea wizard to add folders to a specified location on the local file system.

   The project folder (view paths) appears in the WorkAreas View.
2. Update the files in the WorkArea using tools that are external to the Workbench.
3. Refresh checked-out status (F5) for the WorkArea to synchronize it with the client path and use Synchronize to determine how the WorkArea differs from the Repository.

   The Synchronizer displays which files are modified or new, and displays which files have outgoing, incoming, or conflicting changes.

4. Commit outgoing changes to CA Harvest SCM, get incoming changes from the repository, or merge conflicting changes into your local version.

   The status indicators in the WorkArea dynamically update to reflect the results of these operations.

WorkArea Actions

Actions in the WorkArea help you to use the trees and views. You can perform the following actions in a WorkArea:

Expand All

Expands the entire subtree from the selected node. Expand all shows all files in the filtered view and saves you from individually expanding all the subtrees. Expand all is especially helpful when viewing a filtered view such as Outgoing Only or Show Conflicts Only.

Collapse All

Collapses the subtree of the selected node.

Unshare WorkArea

Disassociates the WorkArea project from the CA Harvest SCM broker, but maintains the project in the WorkArea View. Unsharing removes the CA Harvest SCM project metadata from the project. After you unshare a project, you can do the following:

- Discard the broker (if no other project is associated with CA Harvest SCM using this broker).
- Share the project again with CA Harvest SCM.
- Add the project to your WorkArea.

Disconnect WorkArea

Disassociates the project from the CA Harvest SCM broker but maintains the project metadata and shows the WorkArea in OFFLINE mode in the WorkArea view.

Delete WorkArea

Opens the Delete WorkArea *name* contents? dialog that lets you select an option for how to delete the WorkArea project.

Go Into

Removes the selected node from the filtered view. A hidden item can be restored to the view by a Refresh (F5) or a Synchronize.

Undo Checkout

Performs an undo check-out of files in the subtree of the selected node. You can lose local modifications if the file has been modified, because the file will be replaced with the appropriate version from the repository.

Define a WorkArea

WorkAreas help you see file status and manage your work. Each WorkArea has a single CA Harvest SCM repository context to which it is associated; you can edit the context to modify the package and processes.

Note: You can define a WorkArea as described in the following procedure. In addition to the Add WorkArea action, you can use the Compare to Directory action to create a WorkArea.

Follow these steps:

1. Navigate the view path in the Explorer View.
2. Right-click a view path that you want to include in your WorkArea, and select Add WorkArea from the shortcut menu. You can select only one view path at a time.

   The Add WorkArea wizard appears.
3. Complete the wizard pages and fields. Click OK.

   If the Populate WorkArea option is selected, the WorkArea is recursively populated with the contents of the latest versions of the items in the repository folder. If the Populate WorkArea option is not selected, an empty WorkArea appears at the specified location on the file system.

   Note: The latest trunk version of each file in the folder is used. If a file or folder has a CA Harvest SCM context that differs from the context of the WorkArea, the files or folders appear as new files or folders in the WorkArea.

   The Tree view of folders and files appears in the WorkArea. Each WorkArea has an anchor point on the local file system.

More information:

Set WorkArea Preferences (see page 71)

Add Project Folders to the WorkArea

You can add project folders (view paths) to the WorkArea. The repository settings you specify on this dialog become the initial context settings for the WorkArea project. You can edit these settings subsequently using the Edit Context menu option.

Note: The minimum requirement for this procedure is a check-out for browse process.

To add a project folder to your WorkArea

1. Navigate the Explorer View, and select the folder you want to add to your WorkArea.
2. Right-click the folder, and select Add to WorkArea from the shortcut menu.

   The Add to WorkArea dialog appears.
3. Select the Add as a new project in the WorkArea option.
4. Select a package as the context package. Set the CA Harvest SCM processes and placement options for your context. Click Next.

   The Checkout Options page appears.
5. (Optional) Select check-out options:

   Reserve files

   Checks out files and creates corresponding Reserved-tags (r-tag) in the repository.

   Create empty folders

   Creates directories that contain no items during a get.

   Always overwrite files without asking

   Overwrites files if the files already exist in the WorkArea, without displaying a confirmation dialog.

   Populate WorkArea from Repository

   Populates the WorkArea recursively with the contents of the latest versions of the items in the repository folder. When you do not select this option, an empty WorkArea appears at the specified location on the file system.

   Click Finish.

   The project folder appears in your WorkArea.

Add Multiple Folders to the WorkArea

You can select and add multiple project folders to the WorkArea using the Add to WorkArea wizard. Each project folder that is added has the same initial context that was set during the Add to WorkArea wizard operation on the Set Context page. During the Add to WorkArea operation, when the wizard sets the contexts of the selected folders, an informational message is displayed in the wizard status area. The step may take a few moments to complete after you click Finish.

When you select multiple project folders in the wizard, the option to rename projects is disabled. Each folder is added to the WorkArea with its original name. If any of the selected folders contains a .project file, the first option in the wizard is disabled.

If any of the selected folders does not contain children, the right-click option for Add to WorkArea is disabled. All selected folders must have children for the action to be enabled. The Add to WorkArea option remains disabled when you select a single project without children.

If you select a parent with no .project file the new project wizard option is enabled, even though one or more child folders may contain a .project file.

Because the Add to WorkArea action does not change the .project file, the child subfolders are left unchanged. If one or more of the child subfolders are selected, and any of them contain a .project file, the new project wizard option is disabled.

Edit WorkArea File-Level Context

The Add to WorkArea wizard lets you set the file-level context of items that you add to the CA Harvest SCM Explorer view. You can set the file-level context for multiple WorkArea files at the same time. File-level context and project-level context settings can differ only in a selected package. Both types of context settings share the same processes and placement option information.

Follow these steps:

1. Right-click a WorkArea file in the WorkAreas view, and select SCM WorkArea, Edit Context.

   The WorkArea Context dialog appears.

2. Complete the fields in the dialog. Following are descriptions of fields that are not self-explanatory:

   Inherit package from project-level context

   Sets the existing project-level package context.

   Packages set at the file-resource level override settings at the project level.

   Package

   Specifies a package context if the Inherit package from project-level context option is not selected.

   Note: Packages can be set at either the project level or the file-resource level. Packages set at the file-resource level override settings at the project level.

SCM Process Options

Sets the CA Harvest SCM processes and placement options to your file-level and project-level context.

If you select a CA Harvest SCM process, its corresponding placement option is displayed as a check box button. For a check-out process, the check box represents the check-out mode for all other process placement options.

Click OK.

The file-level context of the WorkArea file is set.

Note: The settings in this dialog remain the same until you modify them. After you set your context, you do not have to open the WorkArea Context dialog unless you want to change the settings.

Edit WorkArea Project-Level Context

The Add to WorkArea wizard lets you set the project-level context of items that you add to the CA Harvest SCM Explorer view. You can set the project-level context for multiple WorkArea projects at the same time. Both projects and folders use the project-level-context. File-level context and project-level context settings can differ only in a selected package. Both types of context settings share the same processes and placement option information.

Follow these steps:

1. Right-click a WorkArea project or folder in the WorkArea view, and select Team, Edit Context.

   The WorkArea Context dialog appears.

2. Complete the fields in the dialog. Following are descriptions of fields that are not self-explanatory:

   Reset all files to inherit this project-level package

   Resets all the files that have file-level context information with project-level information.

   Packages set at the file-resource level override settings at the project level.

   Click OK.

   The project-level context of the WorkArea project or folder is set.

   Note: The settings in this dialog remain the same until you modify them. After you set your context, you do not have to open the WorkArea Context dialog unless you want to change the settings.

Edit Files

The Editor View lets you view and edit your files. You can have multiple editors open at once; the active editor is in the foreground. You can switch editors by clicking the tabs.

To view and edit a file's content, double-click a file version.

The file versions contents appear in the Editor View.

Type Ctrl+S to save changes.

Compare a Repository Folder to a Directory

You can compare a folder in the repository to a directory in a local file system, and perform a Commit to Repository or Get from Repository for files that differ from items in the repository as follows:

- If changes are outgoing, Commit to Repository is enabled.
- If changes are incoming, Get from Repository is enabled.

Note: If changes are either outgoing or incoming, both Commit to Repository and Get from Repository are enabled.

You can create a WorkArea without populating repository items by comparing the repository view path to a directory on a local file system. After you perform this action, the Synchronizer view opens and shows differences between local and remote item changes.

Follow these steps:

1. Navigate the Explorer View tree data view to the folder you want to compare.
2. Right-click the folder, and select Compare to Directory from the shortcut menu.

   The Select Directory dialog appears.
3. In the Select Directory dialog, select the directory that you want to compare. Click OK.

   A comparison tree appears in the WorkAreas view.
4. Click the Edit Context toolbar button and set the package and appropriate context options.
5. Right-click the folder in the comparison tree and select Synchronize from the shortcut menu.

   The Synchronizer view appears and you are ready to perform a Commit or Get.

6. (Optional) Right-click one or more of the nodes in the comparison tree and select an option from the shortcut menu:

   Commit to Repository

   Creates or updates a version of the file in the repository.

   Get from Repository

   Gets changes from the repository that others have made.

   Mark as Merged

   Marks a managed item with conflict changes to indicate that the changes were locally merged from the WorkArea to the repository.

   The changed nodes appear in the Compare Editor.

7. In the Structure Compare view, double-click a file or perform a Compare with Repository operation from the shortcut menu to differentiate the repository version to that of the local file.

   The comparison shows in the Compare Editor.

Synchronizer View WorkArea Actions

Actions in the WorkArea help you to use the trees and views. You can perform the following actions in a Synchronizer view WorkArea:

Expand All

Expands the entire subtree from the selected node. Expand all shows all files in the filtered view and saves you from individually expanding all the subtrees. Expand all is especially helpful when viewing a filtered view such as Outgoing Only or Show Conflicts Only.

Collapse All

Collapses the entire subtree from the selected node. Collapse all hides all files in the filtered view and saves you from individually collapsing all the subtrees.

Hide

Removes the selected node from the filtered view. This action has no effect in the Show All mode of the view. A hidden item can be restored to the view by a Refresh (F5) or a Synchronize.

Show in WorkArea

Shows the selected project, folder resource, or file resource in the WorkArea view.

Revert

Reverts the selected file or folder to the previous version in the CA Harvest SCM repository. At the folder level, the revert option reverts all the files and subfolders under the selected folder. The revert action includes operations such as rename, edit, modify, check out, remove, move, or a combination of these operations at the file or folder level. You are prompted to confirm the revert action for every file or folder that has changed.

Note: File level revert is available from the Synchronizer and Workarea view whereas the folder level revert is available only from the Synchronizer view.

Get Files or Folders From the Repository

Getting files or folders from the repository updates your WorkArea files or folders with the latest trunk version in the repository. All files or folders that you get from the repository must be incoming; that is, they must have left-directional arrows.

To execute a get from repository (update your WorkArea with the latest trunk version in the repository), right-click any file or folder that you want to get in the Incoming view, and select Get from Repository on the shortcut menu.

Your WorkArea is updated.

View or Edit a File in the WorkArea

Typically, you view and edit files using tools that are external to the Workbench; however, you can easily view or edit files in the WorkArea.

Follow these steps:

1. Navigate the WorkAreas View tree to the file you want to view or edit.
2. Right-click the file, and select Edit from the shortcut menu.

   The file contents appear in the Workbench Editor, and you can view or edit the contents.
3. (Optional) Type Ctrl+S to save your changes.

   The changes are saved.

Edit the Repository Context

The Edit Context dialog lets you edit the repository context properties specific to your WorkArea.

Note: The contexts for the package, check-out, check-in, remove item, and remove path processes are stored at the project level. Regardless of where you open the Edit Context dialog to make context changes, the changes to these processes affect all files in the WorkArea.

Follow these steps:

1. Right-click the WorkArea, and select Edit Context from the shortcut menu.

   The Edit Context dialog appears.

2. Complete the dialog fields, and click OK.

   The repository context is changed.

Replace a WorkArea File with a Repository Version

Use the Replace With process to replace files in your WorkAreas with the latest repository versions.

Follow these steps:

1. Right-click the file you want to replace in the WorkArea, and select Replace With, Version in Repository from the shortcut menu.

   A list of versions for the file appears in the Lists view.

2. Right-click the version you want to use for the replacement, and select Get from Repository from the shortcut menu.

   The file is replaced with the version you selected.

Replace a WorkArea Folder with the Latest Trunk Version

Use the Replace With option at the folder level to replace all the files in the selected folder with the latest repository trunk versions. To replace a folder in the WorkArea with the latest trunk version, right-click the folder you want to replace, and select Replace With Latest Trunk Version from the shortcut menu. All the files in the selected folder are replaced with the latest trunk version. If the Trunk version of the folder has additional or new files, they are also added to the folder.

Replace a WorkArea File with the Latest Trunk Version

Use the Replace With process to replace files in your WorkArea with the latest repository trunk versions.

To replace a file in the WorkArea with the latest trunk version, right-click the file you want to replace in the WorkArea, and select Replace With, Latest Trunk Version from the shortcut menu.

The file is replaced with the latest trunk version.

Synchronize a WorkArea Project

After completing your WorkArea project changes, synchronize, and then commit your changes to the repository so that others can have access to the latest versions. While you were working on your files, other developers may have committed changes to the repository.

Synchronizing WorkArea projects rather than individual files is the recommended and most efficient method of using CA Harvest SCM. You keep track of only your WorkArea, not individual files. New, changed, and deleted files are tracked for you. You can synchronize multiple WorkArea files or folders, or any combination of projects, folders, or files, using one synchronize operation. For each project or folder, all files that are descendents of the project or folders are included in the synchronization.

Follow these steps:

1. Navigate the WorkAreas View tree, and select the WorkArea project you want to synchronize.
2. Right-click the project folder, and select Synchronize from the shortcut menu.

   If you are synchronizing a large number of files, an information meter shows the synchronization progress.

   The hierarchical structure of the folder you selected displays in the Structure Compare view. Arrows next to files show whether the change is incoming, outgoing, or conflicting.

3. (Optional) Click a toolbar icon to filter the changes:

   Show Conflicting Changes

   Shows double arrows (<->) that indicate only those items in the repository that have been modified on both the repository and locally.

   Show Incoming Changes

   Shows left arrows (<) that indicate only those items in the repository that need to be brought in to your local WorkArea. This helps ensure that you are current with the current repository state by comparing your local files with the latest items in the repository. This mode also shows items with conflicting changes.

   Show Outgoing Changes

   Shows right arrows (>) that indicate files that have been modified locally and need to be committed to the repository. This mode also shows items with conflicting changes.

   Show Incoming or Outgoing Changes

   Shows right arrows, left arrows, and double arrows—does *not* show files without arrows—that indicate incoming and outgoing changes in a single view from which you can update your WorkArea and commit to the repository, and files with conflicting changes.

   The Structure View shows files that match the filter mode you selected.

   Note: You can right-click a file in the Structure Compare tree to open a shortcut menu with options determined by the direction of the change.

4. (Optional) Click the Synchronize toolbar button to synchronize the Structure Compare view because of changes on the repository or click the Refresh toolbar button to synchronize the Structure Compare view because of changes on the client path.

   Note: Your Mode settings are not changed when you use the Synchronize option.

Using External Compare Tool from WorkArea

You can use an external compare tool to compare the local version of a file with the one on the CA Harvest SCM repository from WorkArea. The external compare tools display the differences in an intuitive way that is easy to view and understand. You can then decide the changes you want to merge. Each compare tool provides different options and benefits; select the tool that best fits your purpose. WorkArea uses the external compare tool that you have configured for Workbench.

Note: The Compare Tool option in the Preferences dialog lets you configure the external compare tool you want to use for performing a 2-way or 3-way compare or merge from the WorkArea. DiffDoc does not support comparison from WorkArea.

For more information about the external compare tools packaged with CA Harvest SCM and the default command line settings, see chapter "Comparing and Merging Versions."

You can invoke the external compare tool by performing the following operations in the WorkArea:

- Compare with Version in Repository
- Compare with Latest Trunk Version
- Double-click on a managed resource in WorkArea Synchronizer View

Note: Managed resource includes outgoing changes, incoming changes, and conflicts.

Configure Two-Way or Three-Way Compare from WorkArea

If you have configured an external compare tool for Workbench, CA Harvest SCM uses the same tool for comparisons at the WorkArea also. The default mode is two-way comparison. If you want to change the comparison mode, you must configure the same.

Follow these steps:

1. Click Tools, Preferences in Workbench.

   The Preferences dialog appears.

2. Navigate to General, Compare/Merge, External Compare/Merge Tools.

   The right pane displays the details of the tool that you have already configured and the associated command line options.

3. Select one of the following options under Compare Tool Options:

   2-way

   Specifies that the compare and merge operation will use two-way comparison.

   3-way

   Specifies that the compare and merge operation will use three-way comparison.

   Note: WinMerge does not support three-way comparison. DiffDoc does not support two-way and three-way comparison. For more information about the parameters in the previously mentioned commands, see Default Command Line Settings (see page 180).

4. Click OK.

   The changes are saved. You can now perform the comparisons in the configured mode from WorkArea.

Check Out and Check In

You can perform Check Out, Undo Check Out, Check In and Keep, and Check In and Commit to Repository on projects, folders or file resource nodes in the WorkAreas view. These actions are applied to all files below the selected object recursively.

Check Out Files to the WorkArea

When you add files to your WorkArea, all files are checked out for browse in read-only mode or checked out for update in read/write mode. When you modify a file, CA Harvest SCM automatically checks out the file according to the check-out process specified in your context. If concurrent update is enabled, the files are checked out to a branch; otherwise, they are checked out to a trunk.

Update Mode

Reserves the version on the trunk, thereby preventing other users from reserving the same resource.

Concurrent Update Mode

Reserves the version on the branch, allowing other users to work on the same resource concurrently.

When checking out files, the exact version is determined by the contents of your context package. If an open branch for that item exists in the context package, a branch version is checked out to continue the existing branch. If there is no open branch, the latest trunk version is checked out.

Important! If a file is renamed or moved in a CA Harvest SCM-managed project, then in a CA Harvest SCM WorkArea, Check Out is disabled for the file. If you want to check out a renamed or moved file, use the Replace With, Version in Repository action, and then check out on the version.

Follow these steps:

1. Right-click the file and select SCM WorkArea, Check Out.

   If a later version of the file exists, the Later Version Exists dialog appears.

2. Click OK to use the latest version.

3. Select whether to use the latest WorkArea or repository version, and click OK.

   Note: If you selected the Always Check Out Latest Version option in the Workbench preferences, the prompts will not appear for subsequent check-out operations.

   The file is checked out using the latest version.

Note: If you clear the Checkout files while refactoring option, refactor operations that cause changes to files will automatically make the affected files writable and mark the files as offline-edited. The affected files will be decorated with the “+” icon, indicating that changes were made. Subsequent synchronize operations will detect these files as offline changes and permit you to check out and release the changes.

Additionally, you can create a package using the Edit shortcut menu option if the Create Package process's initial state is the current check-out state. The Create Package process's initial state is defined in the CA Harvest SCM Administrator.

You can perform a recursive check-out by selecting any project or folder. All files under this selection are checked out.

Undo Check-out

Undoing check-out for a resource deletes the reserve-tagged version in the repository, sets the local file back to read-only mode, and then checks out the file again for browse, effectively undoing the check-out and any changes made to the checked out file.

To undo a check-out, right-click the resource and select SCM WorkArea, Undo Check Out from the shortcut menu.

If the local file has been modified, a warning message appears that the file has been changed and that the changes will not be saved if you perform an undo check-out.

Check In WorkArea Files

There are two options for checking in files:

Check In and Keep

Checks in your file but keeps it reserved so that a copy of a file remains in the repository while leaving you a copy to work on.

Check In and Release

Checks in your file and releases the Reserve tag. To work on the file again, you must check it out again.

To check in a resource

Select SCM WorkArea, Check In and Keep, or Check In and Commit to Repository (Check in and Release).

The recommended alternative to checking in is to use the CA Harvest SCM Synchronizer.

Developer check-in notes for versions and packages display in the Package List view, Package Properties sheet, Version List view, and Version Properties sheet.

Filter the Synchronizer View

You can customize the Synchronizer view so that you view only the files and folders that are pertinent to your work.

To filter the Synchronizer view, click a toolbar icon to use one of the following modes:

Incoming

Shows left arrows (<) that indicate only those items in the repository that must be brought into your local WorkArea. This action helps ensure that your local WorkArea is synchronized with the repository state by comparing your local files with the latest items in the repository.

Outgoing

Shows right arrows (>) that indicate files that have been modified locally and must be committed to the repository.

Incoming/Outgoing

Shows both incoming and outgoing changes in a single view from which you can update your WorkArea and release to the repository.

To filter the Synchronizer view further based on the type of change, click a toolbar icon to use one of the following actions:

Addition

Removes additions from the Incoming or Outgoing or Incoming/Outgoing mode. This means that files appearing in the Synchronizer do not list the newly added items in the WorkArea or newly checked-in items to the repository. Re-selection gets back all the incoming or outgoing additions into the synchronizer view.

Deletion

Removes deletions from the Incoming or Outgoing or Incoming/Outgoing mode. This means that files appearing in the Synchronizer do not list the deleted items in the WorkArea or removed items from the repository. Re-selection gets back all the incoming or outgoing deletions into the synchronizer view.

Normal Change / Edits

Shows normal edited changes with the Incoming or Outgoing or Incoming/Outgoing mode.

Removes changes from the Incoming or Outgoing or Incoming/Outgoing mode. This means that files appearing in the Synchronizer do not list the changed items in the WorkArea or changed items in the repository. Re-selection gets back all the incoming or outgoing changes into the synchronizer view.

The Synchronizer view shows files that match the filter mode you selected.

If you select the addition / deletion / change options and then select the conflict / reset options, the earlier selections are still retained.

Synchronize Multiple Resources

You can synchronize multiple resources with CA Harvest SCM using one synchronize operation. To do so, select any combination of projects, folders, or file resources. For each project or folder, all files that are descendents of the project or folders are included in the synchronization. After synchronization, the results are reported in the Incoming, Outgoing, or Incoming/Outgoing mode view.

By right-clicking a folder or resource in this view, you can get a file from CA Harvest SCM or commit a file to CA Harvest SCM. To recursively get or commit multiple files in one operation, select Get or Commit at the folder level in the Incoming, Outgoing, or Incoming/Outgoing mode view.

Note: We recommend that you synchronize at the folder level or on the set of files for best performance.

Commit Edited Items to the Branch on Latest Trunk

We recommend that you use the Commit Edited Items to the Branch on Latest Trunk preference when using a branch in the context of the WorkArea. This commit preference specifies whether you want to commit branch items to the trunk version of the existing version in the WorkArea or to the latest trunk version available in the repository. Depending on whether you always commit your changes to the WorkArea trunk version or latest trunk version available in the repository, you can set your commit preference accordingly.

Note: By default this option is selected.

This option behaves as follows when branch is set in the context:

- When the option is selected, the commit operation results in a conflict when any later version exists in the repository. A further commit operation results in creating a branch version on the latest trunk version in repository.
- When you clear this option, the commit operation results in an outgoing change even if any later version exists in the repository.

Example: Commit Branch Items on Latest Trunk Version

The WorkArea has version 1 of a file, sample.txt, and the latest trunk version in the repository has version 2 of the same file. The check-in context placement option is on the branch. When you commit the changes, the Synchronization view displays the conflict status of the local and repository versions. Executing a commit operation creates version 2.1.1 of the file as the latest branch version in the repository.

Note: If the check-in placement option is on the trunk, the commit operation warns you that a later version exists. Proceed to reconcile the conflict, mark the status as merged, and commit it. Then, CA Harvest SCM creates version 3 of sample.txt in the repository.

Example: Commit Branch Items on WorkArea Trunk Version

The WorkArea has the version 1 of a file (sample.txt) and the latest trunk version in the repository has version 2 of the same file. When you commit the changes, the Synchronization view displays the outgoing change. If you perform a commit operation on the outgoing change, CA Harvest SCM creates version 1.1.1 of the file as the branch version in the repository.

Note: If the check-in placement option is on the trunk, the synchronize operation results in a conflict. You must reconcile the conflict, mark it as merged, and then perform the commit operation to create version 3 on the latest trunk version.

Change the Commit Preference

You can change the commit preference to specify whether you want to commit the edited items always to the branch version or to the latest trunk version. By default, the commit operation commits the changes to the latest trunk version.

Follow these steps:

1. Click Tools, Preferences in Workbench.

   The Preferences dialog opens.

2. Click WorkArea in the left pane.

   The preferences related to WorkArea appear in the right pane.

3. Do one of the following to the Commit edited items on Latest Trunk option:

   - Select the option to commit the changes always to the latest trunk version.
   - Clear the option to commit the changes always to the branch version.

4. Click OK.

   All the commit operations you perform after this point, will use the preference you have set.

Commit a File or Folder to the Repository

Committing files or folders updates the repository with modified files or folders from your WorkArea.

To execute a commit to repository, in the Structure Compare view, right-click any files or folders that you want to commit to the repository, and click Commit to Repository.

Note: All files or folders must be outgoing; that is, they must have black right-directional arrows. Or, the items can show a red double-headed conflict arrow. A commit on an item with the red double-headed arrow indicates that the local version of the file should replace the conflicting version as the latest version on the repository.

The files or folders are committed to the repository.

Refresh a WorkArea

The Refresh option lets you check for changes on the local file system and update the WorkArea accordingly.

To update a WorkArea because of changes in the file system, right-click the WorkArea, and select Refresh from the shortcut menu.

The WorkArea is refreshed with the latest changes.

Note: Your Mode settings are not changed when you use the Refresh option.

Delete a WorkArea

You can delete a WorkArea to remove it.

Follow these steps:

1. Navigate to the WorkArea that you want to delete.
2. Right-click the WorkArea, and select Delete from the shortcut menu.

   A confirmation dialog appears, and you can decide whether to delete project contents on the disk (this deletion cannot be undone).
3. Click OK.

   The WorkArea is deleted.

Ignore Files

With the CA Harvest SCM ignore facility, you can specify that resources added to a project are not added to version control. A file named .scmignore stores the list of files, directories, or patterns to be ignored. The .scmignore file can be added to any directory of a project.

Note: The .scmignore file is required in each directory where files or subdirectories are to be ignored.

Follow these steps:

1. Select a file or directory currently not being ignored in the WorkAreas view, right-click and select Ignore.

   The Specify Pattern of Files to Ignore dialog appears.

2. Enter the name of the file or directory or pattern to be ignored, and click OK.

   When specifying patterns, use an asterisk (*) and question mark (?) characters as wildcards.

Follow these steps:

1. Select a file or directory that is currently ignored in the WorkAreas view, right-click and select Edit Ignore Files List.

   The editor displays the .scmignore file.

2. Modify the contents of the .scmignore file and save your changes.

   When specifying patterns, use * and ? characters as wildcards.

Chapter 10: Managing Versions, Items, and Paths

Note: The instances of Linux in this section refer to both the Linux and zLinux operating environments.

This section contains the following topics:

How to Move, Rename, and Remove Items or Item Paths

Items and item paths are identified by name, path, and version. Using the Repository tab of the CA Harvest SCM Administrator application, administrators can move, rename, or delete an item or item path only if no project contains a version of the item or item path. To perform any of those actions on an item or item path that has a version in a project, use the Workbench and do the following:

1. Go to the project and state that contains the item or item path you want to move, rename, or remove.

2. Execute the process that corresponds to the action you want to perform on the item or item path:

   - Move item
   - Remove item
   - Rename item
   - Move path
   - Remove path
   - Rename path

   A version is created on a branch or trunk depending on the mode you selected in the process dialog. The Repositories tab of the Administrator interface shows the latest trunk version, regardless of the version's project location.

Example: Move an Item

This example shows an item in \Repository\Directory A named FileX.cpp that has V0, V1, and V2. Versions V0, V1, and V2 all have \Repository\Directory A as their parent path. FileX.cpp is being moved to Directory B.

1. Execute a move item process and specify the Trunk mode to move item FileX.cpp from \Repository\Directory A to \Repository\Directory B.

   A version V3 is created. (If you had selected the Branch mode, a version V2.1.1 would be created.)

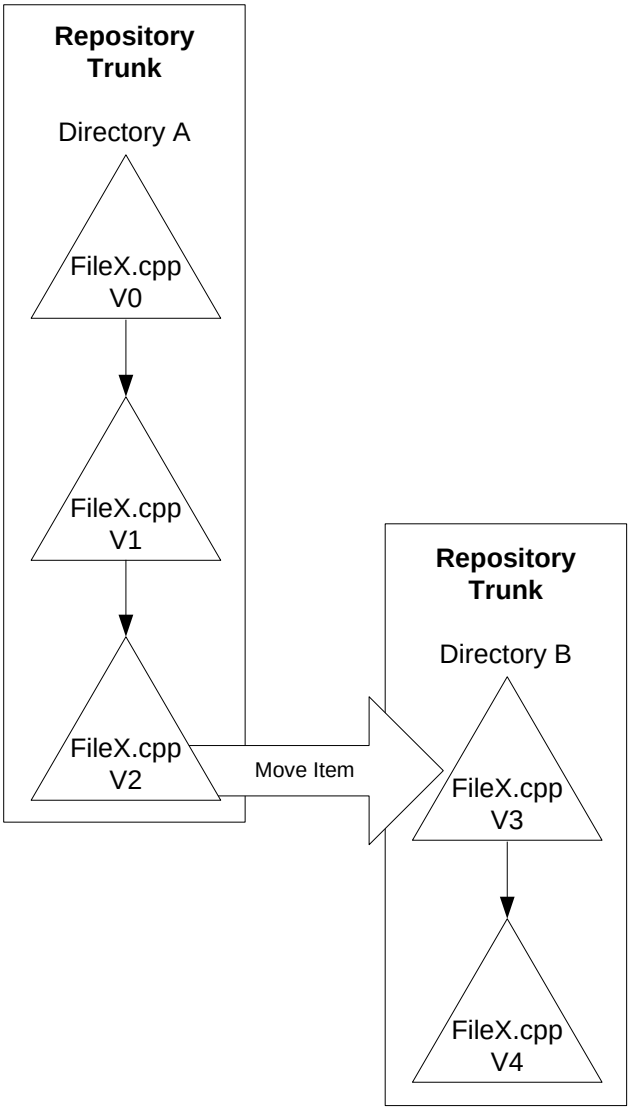
   Version V3 has \Repository\Directory B as its parent path.

   If you expand the package's Versions folder, it lists the following files:

   ```
   FileX.cpp:0 \Repository\Directory A
   FileX.cpp:1 \Repository\Directory A
   FileX.cpp:2 \Repository\Directory A
   FileX.cpp:3 \Repository\Directory B
   ```

2. Continue updating FileX.cpp using \Repository\Directory B to create version V4.

Your current view affects which versions of the item are visible. For example, if Development can see V3, this item is located in \Repository\Directory B. However, if QA can see only V2, this item is located in \Repository\Directory A as shown in the following diagram:



Item Name and Item Path Rules and Considerations

When you modify item names or item paths, for example, by renaming an item path or moving an item, consider the following rules and behaviors:

- You can create versions of existing item paths only by using a rename path, move path, or remove path process, and these versions can be on branches.
- You cannot concurrently update a branch path, that is, another package cannot be used to change the same path on its own branch.
- The interactive merge process lets you resolve item, but not item path, conflicts.

- When using the rename path, remove path, or move path processes, versions are created for all the subitems and paths under the parent item path. If a conflict occurs during the creation of these versions, the entire operation fails. For example, if you attempt to move an item path on the trunk and the item path has a subitem with a merged-tag, the entire move fails.
- You can delete an empty item path if it has no subitems or paths.
- You can delete an original root path version using the delete version process. This also deletes all the subitem and item path dependent versions. If an error occurs during the deletion for any dependent versions, the entire delete version process fails.

Delete Version Rules

The delete version process removes the last change that is made to an item or the initial version of an item. The delete version process has the following rules:

- You can only delete the absolute latest version for an item. You cannot delete intermediate versions.
- You can delete the initial version of an item (version 0) but only if the item was initially checked in as a new item. (An administrator can delete the initial version, created by the load repository function, by using the Repositories tab of the Administrator application.)
- You can delete more than one version of an item simultaneously as long as the versions are sequential and include the latest version.
- You can use the package filter in the Find Version dialog to facilitate your selection. To delete versions from multiple items, all the versions must have been created by the same package.
- You cannot delete a version in the following scenarios:
  - When the "Delete by Creator/Modifier only" option is selected, and if you are not the administator or the creator or modifier of the version.
  - If a version exists in more than one view.

    Note: If you promote a package that created a version, demote it to the previous state before deleting the version.

Delete a Version

The delete version process lets you remove versions of items in the CA Harvest SCM repository.

Follow these steps:

1. Navigate to the version you want to delete.
2. Right-click the version, and select *delete version process* from the shortcut menu.

   The *delete version process* dialog appears, and the version you selected populates the list.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. (Optional) Click Add (the plus [+] sign).

   The Find Version dialog appears; you can select one or more versions.
4. Click Accept Selected.

   The versions are added to the deletion list.
5. (Optional) Remove a version from the list by selecting a version and clicking Remove (the minus [-] sign).

   The selected version is removed from the deletion list.
6. (Optional) Click Note to view information about the process.
7. Click Close.

   A dialog appears in which you can confirm or cancel the delete version process.

Note: You cannot delete a version in the following scenarios:

- When the "Delete by Creator/Modifier only" option is selected, and if you are not an administrator or the creator or modifier of the version.
- If a version exists in more than one view.
- If you promote a package that created a version, demote it to the previous state before deleting the version.

More information:

Item Name and Item Path Rules and Considerations (see page 152)
Restore a Moved or Removed Item (see page 157)
Restore a Moved or Removed Path (see page 162)

Move Item Process

The move item process lets you logically move an item from the current path to another path. The movement is included as a change associated with a package and can progress through the lifecycle as the package is promoted from one view to another. When you move an item, it no longer displays under the original path in the item view. The move item process creates a new version located on the new parent path. This new version has properties like other versions.

Linked processes execute before and after the move item process completes successfully. In this case, “successfully” means that at least one item is moved.

Move an Item

You can logically move an item from the current path to another path.

Follow these steps:

1. Navigate to the item you want to move.
2. Right-click the item, and select *move item process* from the shortcut menu.

   The *move item process* dialog appears and displays the item name you selected.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Select a destination path for the item by clicking the button next to the Target Path field to open the Repository Path Selection dialog. Select a path, and click OK.

   The destination path is selected.
4. Specify a placement option:

   Trunk

   Creates a trunk version for refactoring changes.

   Note: This option is identical to the check-out for update process.

   Branch

   Creates a branch version for refactoring changes.

   Note: This option is identical to the check-out for concurrent update process.
5. Select a package from the Package drop-down list, or click the Package button and use the Select a Package dialog to select a package.

   The package is associated with the moved item.

6. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.

   Click OK.

   The moved item is a new version and appears in the destination path and in the associated package's versions list.

Note: You can also move an item by selecting it in the data view and using drag-and-drop to place it on a destination view path.

More information:

Restore a Moved or Removed Item (see page 157)
Item Name and Item Path Rules and Considerations (see page 152)

Remove an Item

The remove item process lets you logically delete selected items from a view. When you remove an item from a view, the item is not deleted; the remove item process creates a new version tagged as removed (D). This version has attributes like other versions; you can view this version through the Find Version dialog or in the version view, but not in the item view.

Follow these steps:

1. Navigate to the item you want to remove.
2. Right-click the item, and select *remove item process* from the shortcut menu.

   The *remove item process* dialog appears and displays the item name you selected.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Specify a placement option:

   Trunk

   Creates a trunk version for refactoring changes.

   Note: This option is identical to the check-out for update process.

Branch

Creates a branch version for refactoring changes.

Note: This option is identical to the check-out for concurrent update process.

4. (Optional) Remove an item from the list by selecting an item and clicking Remove (the minus [-] sign).
5. Select a package from the Package drop-down list, or click the Package button and use the Select a Package dialog to select a package.

   The package is associated with the removed item.
6. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.

   Click OK.

   The removed item is a new version and appears in the version view and in the associated package's versions list.

More information:

Item Name and Item Path Rules and Considerations (see page 152)

Restore a Moved or Removed Item

To restore a moved or removed item back to its previous location, use the delete version process.

Delete the version created by the move item or remove item process.

The version of the moved or removed item is deleted and the moved or removed item is restored.

More information:

Delete a Version (see page 154)

Rename Item Process Rules

The rename item process logically renames an item. The following rules apply to the rename item process:

- You cannot rename items that have been removed using the remove item process (D-tagged).
- You can only rename an item if it is given a unique name in each project.
- You can rename an item on a branch in a package; however, if you attempt to concurrently merge the package and a duplicate item name exists in the same view path, the name conflict will cause the merge to fail. You must resolve the name conflict by renaming one of the items before you can merge the package successfully.

Rename an Item

The rename item process lets you rename an item.

Follow these steps:

1. Navigate to the item you want to rename.
2. Right-click the item, and select *rename item process* from the shortcut menu.

   The *rename item process* dialog appears and displays the item name you selected.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Enter a new name for the item.
4. Specify a placement option:

   Trunk

   Creates a trunk version for refactoring changes.

   Note: This option is identical to the check-out for update process.

   Branch

   Creates a branch version for refactoring changes.

   Note: This option is identical to the check-out for concurrent update process.
5. Select a package from the Package drop-down list, or click the Package button and use the Select a Package dialog to select a package.

   The package is associated with the renamed item.

6. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.

   Click OK.

   The renamed item is a new version and appears in the version view and in the associated package's versions list.

More information:

Item Name and Item Path Rules and Considerations (see page 152)

Move Path Process

The move path process lets you logically move a path from the current location to another path. The movement is included with the changes associated with a package and can progress through the lifecycle as the package is promoted from one view to another. When a path has been moved, it no longer displays under the original path in the item view. The move path process creates a version located on the new parent path. This version has properties like other versions, which you can view under the package's Versions folder.

When you move a path, the move path process also creates one new version for each item and path under this path. All those versions are combined as one change; individual versions cannot be deleted. Deleting the original moved path version automatically deletes all the sub-item/path versions created by that move path process.

Linked processes execute before and after the move item process completes successfully.

Move a Path

The move path process lets you logically move a path from the current location to another path.

Follow these steps:

1. From the Explorer view, navigate to the path you want to move.
2. Right-click the path, and select *move path process* from the shortcut menu.

   The *move path process* dialog appears and displays the path you selected.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Select a destination path by clicking the button next to the Target Path field to open the Repository Path Selection dialog. Click OK.

   The destination path is selected.
4. Specify a placement:

   Trunk

   Creates a trunk version for refactoring changes.

   Note: This option is identical to the check-out for update process.

   Branch

   Creates a branch version for refactoring changes.

   Note: This option is identical to the check-out for concurrent update process.
5. Select a package from the Package drop-down list, or click the Package button and use the Select a Package dialog to select a package.

   The package is associated with the moved path.
6. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.

   Click OK.

   The moved path is a new version and appears in the destination path and in the associated package's versions list.

Note: You can also move a path by selecting it in the data view and using drag-and-drop to place it on a destination view path.

More information:

Restore a Moved or Removed Path (see page 162)
Item Name and Item Path Rules and Considerations (see page 152)

Remove Path Process

The remove path process lets you logically remove a path. The removal is included with the changes associated with a package and can progress through the lifecycle as the package is promoted from one view to another. When a path is removed from the trunk, it no longer displays in the item view. The remove path process creates a version tagged as removed (D). This version has properties like other versions and can be seen under the package's Versions folder.

When you remove a path, the remove path process also creates one new version (D) for each item and path under this path. All those versions are combined as one change; individual version cannot be deleted. Deleting the original removed path version automatically deletes all of the subitem and path versions created by that remove path process.

Linked processes execute before and after the move path process completes successfully.

Remove a Path

The remove path process lets you logically remove a path.

Follow these steps:

1. From the Explorer view, navigate to the path you want to remove.
2. Right-click the path, and select *remove path process* from the shortcut menu.

   The *remove path process* dialog appears and displays the path name you selected.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Specify a placement:

   Trunk

   Creates a trunk version for refactoring changes.

   Note: This option is identical to the check-out for update process.

Branch

Creates a branch version for refactoring changes.

Note: This option is identical to the check-out for concurrent update process.

4. Select a package from the Package drop-down list, or click the Package button and use the Select a Package dialog to select a package.

   The package is associated with the removed path.

5. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.

   Click OK.

   The removed path is a new version and appears in the versions view and in the associated package's versions list.

More information:

Restore a Moved or Removed Path (see page 162)
Item Name and Item Path Rules and Considerations (see page 152)

Restore a Moved or Removed Path

To restore a moved or removed path to its previous location, use the delete version process.

Delete the version created by the move path or remove path process.

The version of the moved or removed path is deleted and the moved or removed path is restored.

More information:

Delete a Version (see page 154)

Rename Path Process

The rename path process lets users logically rename a path. The renamed path is included with the changes associated with a package and can progress through the lifecycle as the package is promoted from one view to another. When a path has been renamed, it no longer displays the old name in the item view. The rename path process creates a new version with the new name. This version has properties like other versions and you can see it under the package's Versions folder.

When you rename a path, the rename path process also creates one new version for each item and path under this path. All those versions are combined as one change; individual versions cannot be deleted. Deleting the original renamed path version automatically deletes all the sub-item/path versions created by that rename path process.

Linked processes execute before and after the rename path process completes successfully.

Rename a Path

The rename path process lets you logically rename a path.

Follow these steps:

1. From the Explorer view, navigate to the path you want to rename.
2. Right-click the path, and select *rename path process* from the shortcut menu.

   The *rename path process* dialog appears and displays the path name you selected.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Enter a new name for the path.
4. Specify a placement:

   Trunk

   Creates a trunk version for refactoring changes.

   Note: This option is identical to the check-out for update process.

   Branch

   Creates a branch version for refactoring changes.

   Note: This option is identical to the check-out for concurrent update process.
5. Select a package from the Package drop-down list, or click the Package button and use the Select a Package dialog to select a package.

   The package is associated with the renamed path.

6. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.

   Click OK.

   The renamed path is a new version and appears in the versions view and in the associated package's versions list.

More information:

Item Name and Item Path Rules and Considerations (see page 152)

View a Version's Content

You can view a version's content in read-only mode. Merged and deleted versions cannot be viewed.

Follow these steps:

1. Navigate the Workbench to the version you want to view.
2. Double-click the version.

   The editor for that file type displays the file in read-only mode. Reserved versions for a file open in write mode.

View a Version's Content in an External Editor

You can view the content of a version in read-only mode. You cannot view merged and deleted versions.

Follow these steps:

1. Navigate the Workbench to the version you want to view.
2. Right-click the version and select View Version in External Editor from the shortcut menu.

   A window appears with a browse option to select the external editor.
3. Select an external editor.

   The contents appear in the selected editor.

View Item or Version History

The History diagram gives you a graphical view of the version history of an item. Versions are depicted as boxes with the version name, package name, modifier, and last modified date displayed in each box. Trunk and branch versions show their relationships using connecting lines.

You can open any number of diagrams. They display side-by-side with each diagram occupying its own tab in the editor.

Follow these steps:

1. Navigate the Workbench to an item or a version for which you want to view the history.
2. Right-click the item or version, and select History Diagram from the shortcut menu.

   A diagram appears in the Workbench editor.
3. (Optional) Magnify or reduce the diagram view by using the zoom options on the shortcut menu.
4. (Optional) Right-click a diagram node to list diagram options (see page 165) available for the item or version, and execute an option.

   Note: You can set the orientation of the trunk versions, either horizontal or vertical, by using the History Diagram settings (see page 63).

   A menu related to the node appears and includes all actions available in the tree and Lists View.

How to Customize the History Diagram

The History diagram editor lets you do the following:

- Display the version diagram.
- Use various diagram options.

To change the diagram layout, use the following Graph Layouts options:

Default Layout

Displays all the trunk versions vertically and corresponding branch versions horizontally.

Note: The History Diagram (see page 63) preference settings let you change the trunk orientation to vertical or horizontal.

Tree Layout

Displays all the versions vertically.

To view the versions in other projects, use the following Version Options:

Show Changes in Sibling Projects

Displays all the changes associated with the current item in other projects. You can execute this operation by selecting any version node in the diagram and clicking this option. By default, this option returns versions from only active projects. You can specify to exclude or include inactive projects for listing versions in sibling projects by clicking the Include/Exclude Inactive Projects button.

Logical links are shown in the version diagram editor from the selected version. Whenever you select any version and select this option, the diagram editor refreshes with the selected version. If any changes are associated with the selected version in other projects, they are shown as logical links from the selected version. Logical links may not always show the versions derived directly from the selected version.

Note: Only the Tree Layout option is available for this operation, the other options are disabled.

Show Changes in Current Project

Reloads the version diagram.

To decorate the connections of the version diagram, you use the following Connection Options:

Show Connection Labels

Decorates the merged connection with predefined text.

Dashed

Decorates the connections with dashes.

Solid

Decorates the connections with solid lines.

To change the colors of the nodes in the diagram, you use the following Color Options:

Trunk Color

Changes the trunk version color in the displayed version diagram.

Branch Color

Changes the branch version color in the displayed version diagram.

Report on Project Versions

You can report on project versions in the following ways:

- **Show Changes in Sibling Projects**—Lists all the change versions (nonbaseline) in sibling projects. This option is applicable for items and versions only.

  To report on changes in sibling projects, right-click the item or version that you want to report on in the list view or in the explorer view, and select Show Changes in Sibling Projects from the shortcut menu.

  By default, this option returns versions from only active projects. You can specify inactive projects (or switch between active and inactive projects) for listing versions in sibling projects. To switch project types, click the filter option in the upper right corner of the Reports view.

Alter File Type

The following groups can alter file types:

- An administrator logged in to Workbench can perform any type of conversion for any of items in the repository as follows:
  - Item-based conversion
  - Extension-based conversion
- Any user other than administrator, such as any user logged in to Workbench can perform only item-based conversion.

  The item-based conversion is only allowed if the user owns the particular item, that is, if the user owns all the versions of that particular item and no versions of the item exist in other projects. The operation fails when a different user owns any of the versions.

  Extension-based conversion is disabled for the user.
- The following rules apply for both the user types:
  - N-tag and D-tag versions can be converted from binary to text and text to binary.
  - The M-tag version can be converted from binary to text only

1. To convert an item, do one of the following as appropriate:
   - From text to binary, right-click the version you want to convert and select Convert Storage Type, Convert to Binary from the shortcut menu.
   - From binary to text, right-click the version you want to convert and select Convert Storage Type, Convert to Text from the shortcut menu.

   The Change Storage Type dialog appears.

2. Do one of the following:
   - Select "Convert specified item" to convert specific versions of the item, click the item selector button to select versions, and click OK.
   - Select "Convert files or file with extension," select the view path, enter the file name or extension, select the Use Extension check box, and click OK.

List Version Report

The list version process lets you generate reports about the changes made to items in the current project. This process is useful for viewing changes made to an item to create versions on the trunk.

The listing produced by this report is in the same format as that produced by the UNIX diff command (diff is a utility that compares the differences between two files). The differences display as instructions that you can use to change the first version and make it the same as the second. These instructions contain: add a, delete d, or change c commands. A line number follows each command. A less than sign (<) precedes lines from the first version. A greater than symbol (>) precedes lines from the second version. A pair of line numbers separated by a comma represents a range of lines; a single line number represents a single line.

The differencing algorithm converts the first version into the second. The line numbers to the left of each a, c, or d instruction always apply to version 1; numbers to the right of the instructions apply to version 2. To convert version 1 to version 2, you can ignore the numbers on the right.

Generate a List Version Report

The list version process lets you generate reports about the changes made to items in the current project. This process is useful for viewing changes made to an item to create versions on the trunk.

Follow these steps:

1. Navigate to the version you want to report on.
2. Right-click the version, and select *list version process* from the shortcut menu.

   The *list version process* dialog appears, and the version you selected is listed in the dialog.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.

3. Select list version options:

   Show Change Description

   Causes the check-in change description to display.

   Show Actual Change

   Causes the actual line-by-line changes between one version and the next to display.

4. (Optional) Click Add (the plus [+] sign) to open the Find Version dialog.

   The Find Version dialog appears and you can select one or more versions. Click Accept Selected.

   The versions are added to the report.

5. (Optional) Select a version, and click Remove (the minus [-] sign).

   The version is removed from the report.

6. (Optional) Click Note to view information about the process.

   Click OK.

   The List Version report is written to the Output Log, from which you can copy it to the clipboard or save it to a text file.

More information:

List Version Examples (see page 169)

List Version Examples

The following List Version examples are based on versions of a shopping list. You can apply this algorithm to any kind of code. The original example list looks like this:

```
list1
tomatoes
potatoes
corn
```

Example 1: Delete a Line

Remove potatoes from the list and it looks like this:

```
list2
tomatoes
corn
```

The List Version report would display the following output when list1 is compared to list2:

```
2d1
< potatoes
```

This output indicates that you must delete line 2 to make the first list like the second. The line to be deleted displays below the command. The less than sign (<) indicates that this line is from list1.

Example 2: Add a Line

Then instead of leaving out potatoes, add spinach to the list. The second list now looks like this:

```
list2
tomatoes
potatoes
spinach
corn
```

The output of differencing this list against list1 would look like this:

```
2a3
> spinach
```

This output indicates that you need to add a line to list1 after line 2. The line to be added is from the second list, as indicated by the greater than sign (>), and is listed below the instruction.

Example 3: Change a Line

In the third example, you buy peas instead of potatoes. The revised shopping list now looks like this:

```
list2
tomatoes
peas
corn
```

The output of differencing this list against list1 would look like this:

```
2c2
< potatoes
---
> peas
```

This output indicates that line 2 needs to be changed. The line from the first file (< potatoes) needs to be changed to be like the line from the second file (> peas). The three hyphens indicate the end of the text in the first list and the start of the text in the second list that should replace it.

Example 4: Change Several Lines

In the fourth example, you remove the old list and replace it with fruit:

```
list2
apples
pears
bananas
```

The output of differencing this list against list1 would look like this:

```
1,3c1,3
< tomatoes
< potatoes
< corn
---
> apples
> pears
> bananas
```

This output indicates that a range of lines (1-3) in the first list must be changed and replaced with a similar range of lines from the second.

Take Snapshot Process

The take snapshot process lets you create a snapshot view of the current working view. Snapshot views are read-only images of working views at a certain point in time that let you capture a software inventory at significant points in its development. You can use snapshot views to support other application management functions, such as baselining or recreating an application at a certain point in time.

The following rules apply to the take snapshot process:

- If the current working view contains reserved or merged versions, the take snapshot process uses the latest normal trunk versions. To use the reserved or merged versions, the reserved versions must be checked in and the merged versions resolved before executing the process.
- If empty item paths exist in the paths you select, they are included in the snapshot view.
- Branch versions, even if the latest in the current view, are not included in the snapshot. You must perform a concurrent merge to merge the branch versions to the trunk before taking the snapshot.
- Branch versions are included in the snapshot, only if the snapshot is taken using the option “Snapshot view with additional packages.”

Take a Snapshot

The take snapshot process lets you create a snapshot of the current working view.

Follow these steps:

1. Navigate to the state that has the working view you want to capture in a snapshot.
2. Right-click the state, and select *take snapshot process* from the shortcut menu.

   The *take snapshot process* dialog appears.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Name the snapshot, and select options for it:

   Snapshot Name

   Names the snapshot to be created. A default name might have been specified in the process Properties dialog. The name should be used as a template to illustrate a naming convention because multiple snapshots with the same name cannot be created.

   Visible to other Projects

   Makes the snapshot view available to be used in the baseline view of other projects, that is, it will be listed in the Configure Baseline dialog. Typically, only snapshots that represent significant phases of development should be made externally visible.

   Latest versions in this view

   Captures the latest versions in the current working view. If the latest version in the working view has been reserved, the latest one with a normal tag in the trunk is selected.

As of specified date and time

Includes versions in the current working view that were modified before or on a specified date and time. Use the calendar feature to specify a date and time or accept the default of the current date and time.

Important! This behavior deviates from CA Harvest SCM Change Manager r4.x. The date and time is compared with the version modification time. It does not refer to the date and time that versions were present in the working view. For a description of how to simulate CA Harvest SCM Change Manager r4.x behavior, see tech note TEC293284 at http://ca.com/support. CA Harvest SCM Change Manager r4 is no longer supported; however, we provide this information as a courtesy to our CA Harvest SCM Change Manager r4 clients. For information about CA policy for unsupported products, see http://ca.com/support or contact your Account Representative.

4. (Optional) Select packages to include in the snapshot view:

   Snapshot view with additional packages

   Includes the versions contained in the snapshot view specified in the drop-down list, plus the packages listed in Specified packages.

   Note: The drop-down list shows snapshot views in the current project.

   Baseline with additional packages

   Includes the latest versions in the baseline and the latest versions in the Specified packages.

   Specified packages

   Specifies packages from the current state. Click Add (the plus [+] sign) to open the Select Packages dialog. You can remove packages by selecting them and clicking Remove (the minus [-] sign). Initially this list is empty.

5. (Optional) Click the button next to the Path field to open the Repository Path Selection dialog, and select a path to include in the snapshot.

   The selected path populates the Path field.

6. (Optional) Click the button next to the SubPath field to open the Repository Path Selection dialog, which shows the subpaths under the Path selected in the Path field. You can select multiple subpaths. Verify that the subpaths selected belong to the same parent. If not, select a common parent. Click OK.

   All the subpaths that you selected populate the SubPath field.

   Note: This SubPath field is applicable only for the option “Latest versions in this view.”

7. (Optional) Click Note to view information about the process.

   Click OK.

   The versions are captured in the snapshot and the snapshot is listed in the snapshot state.

View Snapshots That Include a Specific Version

You can view the snapshots in your current project that include a specific version. This is helpful, for example, if you have a request to change an item and you need to know if versions of the item exist in different views.

Follow these steps:

1. Right-click a version in the Explorer View, and select View Snapshots from the shortcut menu.

   Note: If the version is not included in a snapshot, a No Snapshots Present dialog appears.

   The View Snapshot Data dialog appears and lists all snapshots in the current project that contain the version.

2. (Optional) Sort columns in ascending or descending order.
3. Click OK.

   The dialog closes.

Rules for Creating Item Paths

You can create item paths in existing data views by using the create item path function.

The following rules apply to the create item path function:

- A check-in process must exist in any state of the project.
- You must have Execute access for the check-in process.
- The check-in process must have either the New, or New or Existing filter option enabled.

If any of these rules are not met, the Create Item Path menu option is disabled.

Create an Item Path

You can create item paths in existing data views.

Follow these steps:

1. Navigate to the data view.
2. Right-click a view path, and select *create item path process* from the shortcut menu.

   The *create item path process* dialog appears, and the Parent field displays the view path you selected.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Enter a name for the new path in the Name field.
4. Specify a placement option:

   Trunk

   Creates a trunk version for refactoring changes.

   Note: This option is identical to the check-out for update process.

   Branch

   Creates a branch version for refactoring changes.

   Note: This option is identical to the check-out for concurrent update process.
5. Use the Package drop-down list, or click the Package button and use the Select a Package dialog, to select a package in the current state to associate with the new path.

   A package association is selected.
6. Click OK.

   The item path is created.

Switch Package Rules

The switch package process lets you move versions from one source package to a target package.

The following rules apply to the switch package process:

- You can switch versions only to a package located in the same state as the source package.
- All or some of the package versions can be switched to another package. For example, Package A has item1 (version 1), item2 (version 2), and you can switch item1 (version1) to a different package but still retain item2 (version2).

- You can switch a branch version to a target package only if the target package does not contain any existing versions of the items being switched.
- If you want to switch a trunk version, the target package cannot have a branch version of the same item.
- Both the branch and the trunk versions must be switched together if the trunk version is a result of a merge from the branch.
- If the target package has branch versions associated with it (even if those branch versions have been merged), the switch package will fail.
- The target package cannot contain the parent path of a version being switched.

Switch Versions from One Package to Another

The switch package process lets you move versions from one source package to a target package.

Note: Any versions that have review comments and attachments are switched with the package. When the target package has a review request, all source package comments and attachments are transferred to that review request. When the target package does not have a review request, the source review request is duplicated before transferring the comments and attachments. The status of the new review is reset to Open or In Progress and the primary reviewer must approve the review again.

Follow these steps:

1. Right-click a package, and select *switch package process* from the shortcut menu.

   The *switch package process* dialog appears, and the Versions list is automatically populated with the source versions.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.

2. Use the Source Package or click the Package button, and use the Select a Package dialog to select a source package.

   The selected package populates the Source Package field.

3. Select the versions in the Versions list that you want to switch to the target package.

   The versions are destined for the target package.

4. Click the Package button, and use the Select a Package dialog to select a target package.

   The selected package populates the Target Package field.

5. (Optional) Click Note to view information about the process.

   Click OK.

   The versions are switched to the target package and are no longer associated with the source package. If a failure occurs, all selected versions remain with the source package.

Chapter 11: Comparing and Merging Versions

This section contains the following topics:

Merging Versions

You resolve branching conflicts in CA Harvest SCM through merge processes. Merge processes let you view changes made to two different versions of an item, combine changes, and discard changes.

You typically accomplish merges in two stages. You can merge versions in the same project or across projects.

- Merging a branch version, created by the Concurrent Update mode of check-out, to the project trunk requires the concurrent merge process.
- Merging across projects requires use of the cross project merge process.
- The interactive merge process is the second stage required for both types of merges. You can also use the interactive merge process as a single step for merging branch versions for a specific item.

External Compare or Merge Tools

The External Compare/Merge Tools Preferences dialog lets you specify an external compare/merge tool to use for file comparison and merging. You must enter command line arguments required to launch the selected external tool. You cannot perform a merge of versions that involves path, name, or status conflicts by using external merge tools. Use the default merge tool to resolve these conflicts.

Note: The selected external compare and merge tool must be installed on your computer before you can use it. Additionally, the path to the selected tool must be listed in the Windows PATH environment variable.

CA Harvest SCM supports the following external difference/merge tools, some of which permit three-way merge:

- Araxis Merge Professional Edition—Two-way compare, two-way merge, three-way merge
- Beyond Compare—Two-way compare, two-way merge, three-way merge
- Diff Doc—Two-way compare
- Guiffy—Two-way compare, two-way merge, three-way merge
- WinMerge—Two-way compare, two-way merge

Note: For operating system support and version information about the listed external tools, see the *Release Notes*.

The default tool is Default, which refers to the CA Harvest SCM interactive merge process. The Command Lines fields are disabled when Default is selected and you do not need to specify command line information to start the interactive merge process.

More information:

Default Command Line Settings (see page 180)

Default Command Line Settings

Default command lines for supported external tools are provided in the list of external tools that follows. All external tools require file names to be specified on the command line.

The following parameters define only file names and not complete file paths:

$(File1), $(File2)

Defines files displayed in the left and right panes of the comparison tool.

$(TrunkFile), $(BranchFile), $(ResultsFile)

Defines files displayed in the left and right panes of the merge tool and the results file for two-way and three-way merge.

$(AncestorFile)

Defines common ancestor file for three-way merge only.

$(Version1), $(Version2)

Defines the path to the files displayed in the left and right panes of the compare tool.

Some external tools require full path names and file names to be specified on the command line. For this case, the following corresponding parameters are defined:

```
$(FilePath1), $(FilePath2), $(TrunkFilePath), $(BranchFilePath),
$(AncestorFilePath), $(ResultsFilePath)
```

Default commands for external tools are defined in the following list; the tools differ with regard to which parameters need to be replaced in command syntax:

Note: For help with additional parameters you can set for a tool, see the help for that tool.

- **Araxis Merge Professional Default Command Lines**

  Use the following compare or merge commands for the Araxis tool.

  The compare command has the following format:

  ```
  Compare.exe /wait /max /2 $(File1) $(File2)
  ```

  The two-way merge command has the following format:

  ```
  Compare.exe /wait /max /2 $(TrunkFile) $(BranchFile) $(ResultsFile)
  ```

  The three-way merge command has the following format:

  ```
  Compare.exe /wait /max /3 /a3 $(TrunkFile) $(BranchFile) $(AncestorFile)
  $(ResultsFile)
  ```

/anumber

Specifies which file in a sequence of files on the command line is the common ancestor file.

The ancestor file displays in the center of the Merge screen.

For example, in the following command /a3 specifies that the third file listed is the ancestor:

```
Compare.exe /wait /3 /a3 $(TrunkFile) $(BranchFile) $(AncestorFile)
$(ResultsFile)
```

For example, in the following command /a1 specifies that the first file listed is the ancestor:

```
Compare.exe /wait /3 /a1 $(AncestorFile) $(TrunkFile) $(BranchFile)
$(ResultsFile)
```

The two-way compare command from WorkArea has the following format:

```
Compare.exe /wait /max /2 $(File1) $(File2) /title1:$(Version1)
/title2:$(Version2)
```

The three-way compare command from WorkArea has the following format:

```
Compare.exe /wait /max /3 /a3 $(File1) $(File2) $(AncestorFile)
/title1:$(Version1) /title2:$(Version2) /title3:$(Version3)
```

- **Beyond Compare Default Command Lines**

  The compare command has the following format:

  ```
  BComp.exe /readonly $(File1) $(File2) /title1=$(Version1) /title2=$(Version2)
  ```

  /readonly

  Specifies that editing the files is not allowed.

  /title1

  Specifies a text version of the first file name without the path; file name(version) displays in the compare pane.

  /title2

  Specifies a text version of the second file name without the path; file name(version) displays in the compare pane.

The two-way merge command has the following format:

```
BComp.exe /rightreadonly $(TrunkFile) $(BranchFile) /mergeoutput=$(ResultsFile)
/title1=$(TrunkFile) /title2=$(BranchFile)
```

The /rightreadonly option enables editing only in the trunk pane.

The three-way merge command has the following format:

```
BComp.exe $(TrunkFile) $(BranchFile) $(AncestorFile) $(ResultsFile)
```

The two-way compare command from WorkArea has the following format:

```
BComp.exe /rightreadonly $(File1) $(File2) /title1=$(Version1)
/title2=$(Version2)
```

The three-way compare command from WorkArea has the following format:

```
BComp.exe /rightreadonly $(File1) $(File2)  $(AncestorFile) $(File1)
/title1=$(Version1) /title2=$(Version2) /title3=$(Version3)
```

- **Diff Doc Default Command Lines**

  Diff Doc can compare Microsoft Word documents (.doc or .rtf) with formatting intact. Merging is not supported. Diff Doc requires full path and file names to be entered in the command line.

  The compare command has the following format:

  ```
  DiffDoc.exe /m$(FilePath1) /s$(FilePath2)
  ```

  The /m option lets you specify the original file. The /s option lets you specify the modified file.

- **Guiffy Default Command Lines**

  The compare command has the following format:

  ```
  Guiffy.exe -gm $(File1) $(File2)
  ```

  –gm

  Suppresses file merge (compare only).

  The two-way merge command has the following format:

  ```
  Guiffy.exe -m $(TrunkFile) $(BranchFile) $(ResultsFile)
  ```

  –m

  Specifies two-way merge.

  The three-way merge command has the following format:

  ```
  Guiffy.exe -s $(TrunkFile) $(BranchFile) $(AncestorFile) $(ResultsFile)
  ```

  –s

  Specifies three-way merge.

  The two-way compare command from WorkArea has the following format:

  ```
  Guiffy.exe -m $(File1) $(File2) $(File1) -h1$(Version1) -h2$(Version2)
  ```

  The three-way compare command from WorkArea has the following format:

  ```
  Guiffy.exe -s $(File1) $(File2)  $(AncestorFile) $(File1) -h1$(Version1)
  -h2$(Version2) -hm(Merge_Result)
  ```

  -h1

  Represents the header for the first file.

  -h2

  Represents the header for the second file.

  -hm

  Represents the header for the merged file.

- **WinMerge Default Command Lines**

  The compare command has the following format:

  ```
  Winmergeu.exe $(File1) $(File2) /dl $(File1) /dr $(File2)
  ```

  /dl

  Specifies a text description of the left-hand file without the path; file name (version) displays in the compare pane.

  /dr

  Specifies a text description of the right-hand file without the path; file name (version) displays in the compare pane.

  The two-way merge command has the following format:

  ```
  Winmergeu.exe $(TrunkFile) $(BranchFile) $(ResultsFile) /dl $(TrunkFile) /dr
  $(BranchFile)
  ```

  WinMerge does not support three-way merge; the three-way Merge command line should be left empty.

  The two-way compare command from WorkArea has the following format:

  ```
  Winmergeu.exe $(File1) $(File2) /dl $(Version1) /dr $(Version2) /wr
  ```

  Note: WinMerge does not support three-way compare for WorkArea.

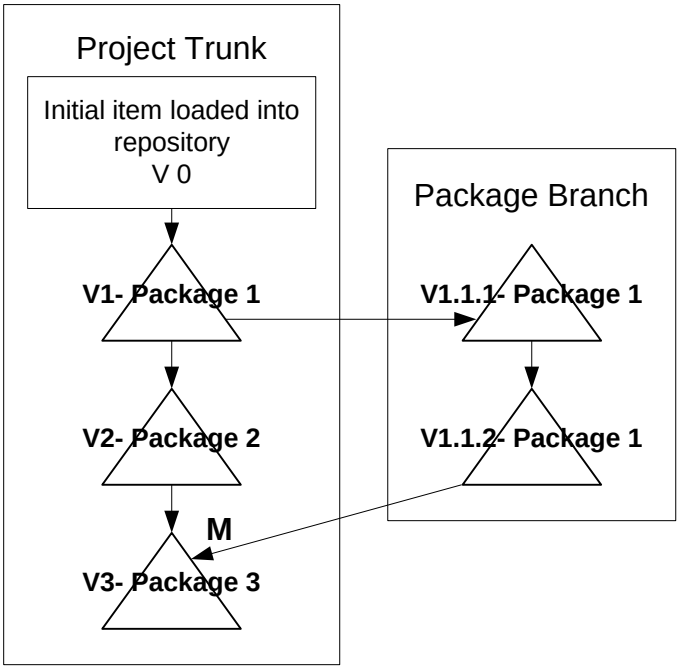
Concurrent Merge Process

A version on a branch becomes part of the project trunk through the concurrent merge process. Until you execute this process, changes made on the branch exist only on the branch. After all items have been checked in to a branch, invoking the concurrent merge process for the package containing the changes causes the versions on the branch to be combined in a single version on the trunk. The new version created on the trunk is the latest in the view.

If more recent versions exist on the trunk than the version from which the branch was created, the version created by the concurrent merge is tagged as merged (M). If the version from which the branch was created is the latest version of the item on the trunk, the version created on the trunk by the concurrent merge has no tag.

Important! CA Harvest SCM does not support concurrent development of item paths; the concurrent merge process cannot create item path versions with the merged tag.

In the following diagram, the branch version 1.1.2 is concurrently merged with the project trunk. The version created by the concurrent merge becomes the latest on the view's trunk, version 3, and is tagged as merged.



In the preceding diagram, if version 2 did not exist on the project trunk, the concurrent merge process would have created a non-tagged version 2. Merged tags appear only when changes have been made on the trunk that could conflict with changes made on the branch. If version 1 is the latest version when the merge executes, no changes exist on the trunk to conflict with the branch versions, so a merged tag is unnecessary.

The merge-tagged version of an item must be resolved through the interactive merge process before another version of the same item can be concurrently merged to the project trunk. The check-out for update and cross project merge processes also cannot be successfully executed on an item for which a merge-tagged version exists; and packages which contain merge-tagged versions cannot be promoted or demoted to a state in a different view.

Merge a Branch Version to the Trunk

The concurrent merge process lets you merge a branch version to the trunk.

Follow these steps:

1. Navigate to the package that has the branch version you want to merge.
2. Right-click the package, and select *concurrent merge process* from the shortcut menu.

   The *concurrent merge process* dialog appears.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Select Merge options:

   Merge Conservatively

   Creates a merge-tagged version, regardless of the contents of the versions. An exception is when the branch version is the latest version in the view; in this case, it is closed and a normal version is created.

   Merge Aggressively

   Creates a merge-tagged version only when conflicts are found. If no conflicts are found, the branch and trunk versions are merged to create a normal version.

   Note: A conflict occurs when a set of lines is modified in both the branch and the trunk; however, insertions and deletions are not conflicts.

   Take Trunk Version

   Selects the trunk (target) to create the final version automatically, without comparing the contents of the versions. This option closes the branch, but does not create any versions on the trunk.

   Take Branch Version

   Selects the branch (source) to create the final version automatically, without comparing the contents of the versions. This option creates a normal version on the trunk and closes the branch.

   The merge behavior is set.

4. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.

   Click OK.

   A version is created on the trunk and is the latest in the view.

Interactive Merge Process

The interactive merge process lets you do the following:

- Combine the changes made on unmerged branches with changes on the trunk.
- Resolve a merge-tagged version after it is created by the concurrent merge or cross project merge process.

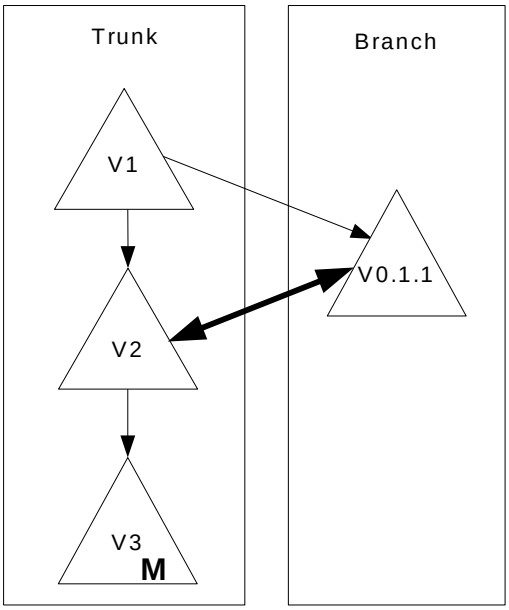
Binary files cannot be used with the interactive merge process. When this process executes, it first checks that the file consists entirely of printable characters (or control characters such as tab, carriage return, and line feed). If any area of the file contains binary data, the merge process does not execute and an error message displays.

The interactive merge process lets you perform a two-way or three-way merge:

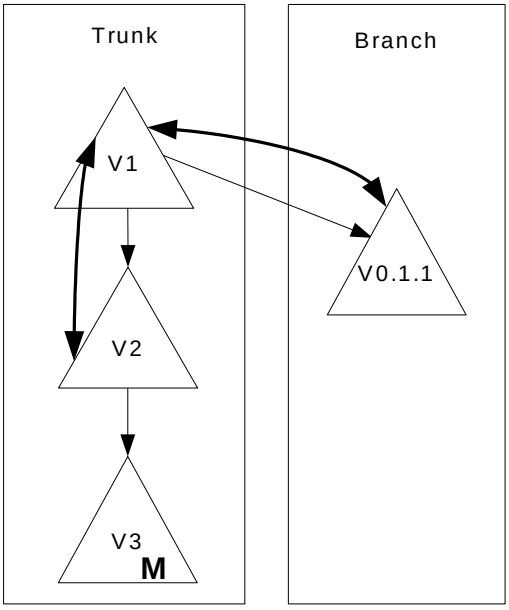
- Two-way merges let you see only the differences between two versions, without considering the common ancestor. You cannot see whether lines were changed, deleted, or added—only that the lines are different.
- Three-way merges let you see the differences between the two versions and the common ancestor. You can see whether lines were changed, deleted, or added.

Note: If you want to check in a branch version and do not need to look at the differences between versions, make sure that it is the correct version, and click OK.

The following diagram shows a two-way merge process. The bold bi-directional arrow shows the comparison between the branch and trunk versions.



The following diagram shows a three-way merge process. The curved arrow shows the comparison between the branch and trunk versions, and the comparison between the common ancestor and the trunk and branch versions.



Merge Versions Interactively

The following procedure describes how to use the internal CA Harvest SCM interactive merge process and assumes that you want to resolve conflicts. If you selected Default for the merge tool preference, invoking interactive merge opens the interactive merge process. If you selected an external comparison tool for the merge tool preference, the external tool appears.

Note: For information about using the external tool to visually compare file versions, see that tool's documentation.

Follow these steps:

1. Navigate to the item or merge-tagged version you want to resolve.
2. Right-click the item or version, and select *interactive merge* from the shortcut menu.

   The *interactive merge* process dialog appears.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Specify values to use for the merge; the versions for the Status, Name, and Path fields can be different.

   The two versions of the item being merged appear in the merge dialog with common blocks (lines identical in both versions) and conflict blocks. Conflict blocks are positioned side-by-side and are outlined.

   Note: A *conflict* occurs when the same line or block of data is modified in both the branch and the trunk. Insertions and deletions are considered *changes*.
4. Resolve conflicts by manually editing the left pane text or by clicking the following toolbar actions:

   - Copy all nonconflicting changes from right to left.
   - Copy current change from right to left.

   The left pane represents the branch version being merged; the right pane represents the latest trunk version.
5. (Optional) Click Note to view information about the process.

   Click OK.

   The merge-tagged version is replaced by a new, Normal-tagged version as the latest in the project's trunk.

Compare Package Versions with Their Trunk Versions

You can compare versions in a package with the versions on the trunk that they were derived from.

Follow these steps:

1. Navigate the Workbench to the package that has the associated versions you want to compare.
2. Right-click the package, and select Compare with Trunk from the shortcut menu.

   The Compare View tab displays the latest version of all the files in the package.
3. Double-click the file that you want to compare.

   The differences in the selected version and the previous trunk version are highlighted in the compare window. Any further comparison uses the same compare window. While comparing the files from a previous trunk, the trunk versions in the present package are not considered.

   When you double-click an R-tagged file, the comparison uses the contents from the local file system to compare with the parent trunk in the CA Harvest SCM repository.

   Note: If you have configured an external tool for compare, a separate instance of the tool opens for different versions of the package.
4. Compare versions by navigating the synchronized lines.
5. (Optional) Click the bookmarks aligned vertically along the right side of the Editor, or click the Select Next Change and Select Previous Change arrows to navigate the lines.

   The lines that differ are located, and you can compare them.

More information:

Configure Two-Way or Three-Way Compare from WorkArea (see page 140)

Compare a Branch Version with Its Parent Trunk Version

You can compare a branch version with its parent trunk version from the Explorer View tree or the Item History diagram.

To compare a branch version with its parent trunk version, right-click a branch version, and select Compare with Trunk from the shortcut menu.

The compare tool appears and displays the differences between the branch version and its parent trunk version. For R-tagged versions, the comparison uses the local content from the file system. This local content displays in the left pane to compare with the parent trunk, which displays in the right pane.

Note: If you configured an external comparison tool in Preferences, the external tool appears.

Compare Two Versions

You can compare any two versions of an item.

Follow these steps:

1. Navigate the Workbench to the item that has versions you want to compare.
2. Select two versions, right-click one of the versions, and select Compare from the shortcut menu.

   The compare tool appears and displays the differences between the two versions. The Compare tool uses the local path and the local file content for an R-tagged file.

   Note: If you configured an external comparison tool in Preferences, the external tool appears.

Compare Versions or Files

You can compare two versions or files.

Note: You can select two versions, right-click, and select Compare from the shortcut menu. This action shows a comparison and bypasses the dialog described in the following procedure.

Follow these steps:

1. On the Workbench toolbar, click Compare.

   The Compare Versions or Files dialog appears.

2. Populate the fields in the Compare Versions or Files dialog by doing one the following:

   - Dragging *versions* from the List View or the Explorer View to the fields. When you drag an R-tagged version, the comparison uses the local path. When you drag an N-tagged version, the comparison uses the repository path.
   - Using the browse buttons to locate and select local *files*.

   The versions or files for the comparison are selected.

3. Click OK.

   The compare tool appears and displays the differences between the two versions. The internal CA Harvest SCM compare files process is read-only; changes cannot be made to the files.

   Note: If you configured an external comparison tool in Preferences, the external tool appears. For some external tools, the compare and merge processes are combined. You can change the files, but CA Harvest SCM does not save these changes.

Compare Views

The compare view process lets you generate a report showing the differences between any two views, either snapshot or working, that exist in any project.

Follow these steps:

1. Navigate the Explorer tree to the state associated with a view you want to compare.
2. Right-click the state, and select Compare View from the shortcut menu.

   The *compare view process* dialog appears and the View fields show the view contexts when you invoked the dialog.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.

3. Click the button next to the Path field to open the Repository Path Selection dialog.

   The dialog appears, and you can select a repository path and a different view. The Show Views with State Nodes option lets you show only views associated with states.

4. Specify the items you want to show in the Compare View list by selecting one or more of the following options and clicking Compare.

   Recursive

   Searches all paths beneath the current path and shows the item versions matching the other filtering criteria. This option works with the Show options. Typically, only items in the path specified in the View and Path fields and specified by the Results criteria are displayed in the dialog list.

Items only in View 1

Specifies that all items in View 1 should be listed.

Items only in View 2

Specifies that all items in View 2 should be listed.

A list of items that match your criteria shows the items unique to one view or which have changes between the views. The dialog changes dynamically to let you change your options and refresh your comparison.

Common Items/Different Contents

Specifies that all items that are common to View 1 and View 2 but have different contents should be listed.

Common Items/Identical Contents

Specifies that all items that are common to View 1 and View 2 and have identical contents should be listed.

5. (Optional) Select an item in the list, and click Difference.

   The Difference dialog appears and shows detailed, line-by-line differences between two text files.

6. (Optional) Click Note to view information about the process.
7. Click Close.

View Differences

The Difference dialog lets you see detailed, line-by-line differences between two text files. The versions being compared for differences are displayed in a read-only mode with common blocks (lines that are the same in both versions) and conflict blocks.

Follow these steps:

1. Select an item in the Compare Views results list, and click Difference.

   The Difference dialog appears, displaying two panes that show the version names and their contexts:

   left pane

   Contains the current project and view context when the Compare Views dialog was invoked.

   right pane

   Contains the comparison view that was selected by the compare views process.

The panes are synchronized to display the same conflict lines. Conflicting blocks are displayed side-by-side, and the portions of the item that are the same for each version are displayed across the width of the dialog. Shaded lines indicate conflicting text.

2. View the conflicts by using the scrollbar or the conflict navigation toolbar buttons. Click Close.

Cross Project Merge Process

The cross project merge process lets you merge the changes made to items in one project with the changes made to the same items in another project. The merge creates versions in the target project that are the latest for each item on the trunk. For example, in release-oriented development, some developers work on a maintenance release such as Release 1.1, while others work on major functional changes for Release 2.0. Several items are updated in both projects and the groups must combine their changes to update the items in both projects.

Cross project merges use packages or snapshots. You select a destination package and one or more source packages or a snapshot for the merge. For the merge to be successful, the source packages cannot contain any reserved or merge-tagged versions. If you set the placement option to trunk only, the destination project cannot contain any reserved or merge-tagged versions of the items being merged; otherwise, a branch version for the incoming items is created.

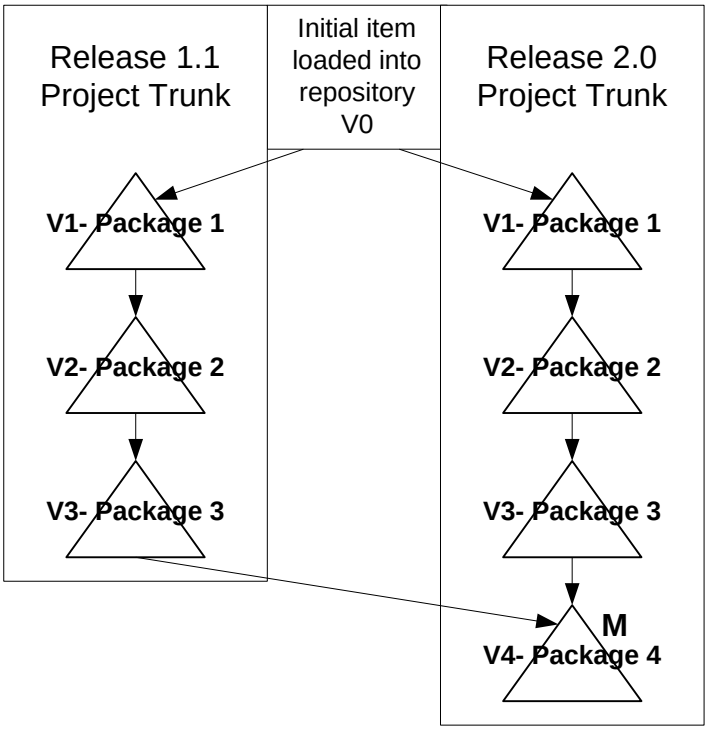
A removed-tagged (D) item can be carried to the destination view if the item has already been removed in the destination; its new versions cannot be carried from the source. Renamed items in the source are renamed in the destination.

All items for the versions that are being merged must exist in one or more repositories shared by the two projects. If two projects do not share a repository, a cross project merge between the projects is not possible.

Branch versions are not eligible for a cross project merge. If the source package of a cross project merge contains branch versions, the merge process ignores those versions and executes a normal merge for trunk versions associated with that package. The presence of branch versions does not generate error messages in the merge process.

The versions created in the target project by the cross project merge are located on the trunk and are the latest on that project's trunk. Each version created as a result of the merge is tagged as merged (M) unless the trunk has not been changed, in which case conflicts do not occur and the source version is simply copied to the trunk of the destination project and not tagged. You can use the interactive merge process to resolve merge-tagged versions.

In the following example, a package in project Release 1.1 that contains version 3 of an item is cross project merged with a package in project Release 2.0, which contains one version of the item.



The new version 4 created in project Release 2.0 by the cross project merge is the latest on the project's trunk, and is tagged as merged (M). The destination package specified in the cross project merge process is associated with the new version, regardless of whether the package contained a previous version of the item.

Cross project merges can sometimes create versions of an item that did not previously exist in the destination project. For this to occur, the following conditions must be met:

- The source and destination projects must share a repository.
- The item for which versions are being merged must be a new item that was loaded or checked in to the shared repository. If the item was loaded into the repository, the destination project must have been associated with the repository before the item was loaded.

- The item must exist in the source project. This rule is true in the following situations:
  - If the new item was checked in from the project.
  - If the project was associated with the repository after the item was loaded.
  - If the item was created in the project by a cross project merge.

When these conditions are met, the cross project merge process creates versions of the item in the destination project. If the source package of the merge contains only version 0 of the item, this version is copied to the destination project as version 0. If the source package contains a non-zero version, the merge creates a version 0, which is the same as the version being merged.

Cross project merges typically produce new, merge-tagged versions in the destination project; however, cross project merges can have different results. The resulting versions depend on the status of the versions in the source package and destination project.

Cross project merges can be performed using a snapshot. The merge rules for a snapshot are similar to that of the merge for a package. You select a destination package and a snapshot from the source project for the merge. By default, the versions that are modified (versions greater than the base versions) in the snapshot view are the only merged versions. When you want to merge the entire view including the base versions, select the Merge from Base Versions check box.

Merge-Tagged Version Restrictions

These restrictions apply to merge-tagged versions of items:

- A merge-tagged version of an item must be resolved through the interactive merge process before another version of the same item can be cross project merged.
- CA Harvest SCM does not support concurrent development of item paths; the cross project merge process never creates item path versions with the merged tag.
- An item that has a merge-tagged version cannot be checked out for Update.
- Packages that contain merge-tagged versions cannot be promoted or demoted to a state in a different view.

Cross Project Merge Options

The cross project merge options work as follows:

Merge Conservatively

Creates a merge-tagged version, regardless of the contents of the versions. The process fails if the target package has an unmerged branched version of an item also in the source package.

Merge Aggressively

Creates a merge-tagged version only when conflicts are found. If no conflicts are found, the branch and trunk versions are merged to create a normal version. Normal tags can only be created when the versions being compared are in the original baseline of both projects. If the versions were checked in after baselining, merge tags are created regardless of whether conflicts exist.

Note: A conflict occurs when a set of lines is modified in both the branch and the trunk; insertions and deletions are not conflicts.

Take Trunk Version

Automatically selects the trunk (target) to create the final version without comparing the contents of the versions. This option creates a normal version on the trunk and closes the branch.

Take Branch Version

Automatically selects the branch (source) to create the final version without comparing the contents of the versions. This option creates a normal version on the trunk and closes the branch.

The cross project merge placement options work as follows:

Branch Only

Creates a version on the target branch. This lets you copy changes from the source project to the target project even if one or more target items are reserved for update in the main trunk. With this option, a branch is created to store the changes. The target package cannot be the same package that contains the items reserved for update on the main trunk.

Trunk Only

Creates a version on the target trunk.

Trunk or Branch

Creates a version on the target trunk or branch. This lets you copy changes from the source project to the target project even if one or more target items are reserved for update in the main trunk. If items are reserved for update on the trunk, a branch is created to store the changes only if the target package is not the same package that contains the items reserved for update. If items are not reserved for update on the trunk, the items are simply copied to the trunk.

Merge Versions Across Projects

The cross project merge process lets you merge the versions made to items in one project with the versions made for the same items in another project. The merge creates versions in the target project that are the latest for each item on the trunk. The merge process affects all items modified by a package, and you can merge multiple items simultaneously.

Follow these steps:

1. Navigate to the package that is the target (destination) for the versions you want to merge.
2. Right-click the package, and select *cross project merge process* from the shortcut menu.

   The *cross project merge process* dialog appears and your selected package is listed in the Target Package field.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Complete the following fields as appropriate to select source versions and a destination package:

   Project

   Specifies a source project.

   Versions from Package

   Specifies versions from a package to use for the merge.

   Versions from Snapshot

   Specifies versions from a snapshot to use for the merge.

   Note: By default, the versions that are modified (versions greater than the base versions) in the snapshot view are the only merged versions. When you want to merge the entire view including the base versions, select the Merge from Base Versions option. Removed items or paths are not removed in the target project, when you use this option.

   State

   Specifies a source state.

   Snapshot

   Specifies the snapshot to use as the version source.

   Merge Base Versions also

   Merges the entire view including the base versions.

   Target Package

   (Optional) Specifies the destination package for the merged versions. You can click the Package button and use the Select a Package dialog to select a different package.

4. Select a merge option and a placement option from the Merge Options and Placement Options drop-down lists, respectively.

   Merge Conservatively

   Creates a merge-tagged version, regardless of the contents of the versions. The process fails if the target package has an unmerged branched version of an item also in the source package.

   Merge Aggressively

   Creates a merge-tagged version only when conflicts are found. If no conflicts are found, the branch and trunk versions are merged to create a normal version. Normal tags can only be created when the versions being compared are in the original baseline of both projects. If the versions were checked in after baselining, merge tags are created regardless of whether conflicts exist.

   Note: A conflict occurs when a set of lines is modified in both the branch and the trunk; insertions and deletions are not conflicts.

   Take Trunk Version

   Automatically selects the trunk (target) to create the final version without comparing the contents of the versions. This option creates a normal version on the trunk and closes the branch.

   Take Branch Version

   Automatically selects the branch (source) to create the final version without comparing the contents of the versions. This option creates a normal version on the trunk and closes the branch.

   Branch Only

   Creates a version on the target branch. This option lets you copy changes from the source project to the target project even if one or more target items are reserved for update in the main trunk. With this option, a branch is created to store the changes. The target package cannot be the same package that contains the items reserved for update on the main trunk.

   Trunk Only

   Creates a version on the target trunk.

   Trunk or Branch

   Creates a version on the target trunk or branch. This option lets you copy changes from the source project to the target project even if one or more target items are reserved for update in the main trunk. Consider the following:

   - When items are reserved for update on the trunk, a branch is created to store the changes only if the target package is not the same package that contains the items reserved for update.
   - When items are not reserved for update on the trunk, the items are simply copied to the trunk.

5. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.

   Click OK.

   The source package versions are merged to the target project.

Merge Versions From a Package

You can merge the changes modified by one or more packages to a target project. This option lets you merge multiple items simultaneously.

Follow these steps:

1. Navigate the Workbench to the package that is the target (destination) for the versions you want to merge.
2. Right-click the package, and select cross project merge process from the shortcut menu.

   The cross project merge process dialog appears and your selected package is listed in the Target Package field.

   Note: Because process execution dialog names differ depending on the site, CA Harvest SCM documentation uses lowercase italic text to indicate the process dialog name by the process type.
3. (Optional) Select a package from the Target Package drop-down list, or click the Package button and use the Select a Package dialog to select a package.

   The target package is selected.

4. Use the Project drop-down list to specify a project from where you want to merge.
5. Select the Versions from Packages option.
6. Select the State from which you want to merge the changes and includes the source package. Click the plus sign (+) and add the packages that you want to merge to the target package. You can also select packages from different views.

   Note: You can select multiple packages from different states. Do not select any state in the merge window and click the package select button; the packages of all states will be listed.
7. Select a merge option and a placement option from the Merge Options and Placement Options drop-down lists, respectively.
8. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.

   Click OK.

   The source package versions are merged to the target project.

More information:

Cross Project Merge Options (see page 197)

Merge Versions From a Snapshot

The cross project merge process lets you merge versions, including branch versions, from a specific snapshot to a target project.

Note: When you create a snapshot, removed paths and items are not included in the snapshot view. Therefore, the changes for the remove operations in the source project are not included in the target project during the cross project merge process when you use the snapshot view option.

Follow these steps:

1. Navigate the Workbench to the package that is the target (destination) for the versions you want to merge.
2. Right-click the package, and select *cross project merge process* from the shortcut menu.

   The *cross project merge process* dialog appears and your selected package is listed in the Target Package field.

   Note: Because process execution dialog names differ depending on the site, CA Harvest SCM documentation uses lowercase italic text to indicate the process dialog name by the process type.
3. Complete the following fields as appropriate, including the following options:

   Project

   Specifies a source project.

   Versions from Snapshot

   Specifies versions from a snapshot to use for the merge.

   Note: By default, the versions that are modified (versions greater than the base versions) in the snapshot view are the only merged versions. When you want to merge the entire view including the base versions, select the Merge from Base Versions option. Removed items or paths are not removed in the target project, when you use this option.

   Snapshot

   Specifies the snapshot to use as the version source.

   Merge Base Versions also

   Merges the entire view including the base versions.
4. Select a merge option and a placement option from the Merge Options and Placement Options drop-down lists, respectively.
5. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.
6. Click OK.

   The versions that are modified (versions greater than the base versions) in the snapshot view are merged to the destination trunk. If you selected the Merge from Base Versions option, the entire view including the base versions is merged.

More information:

Cross Project Merge Options (see page 197)

Chapter 12: Managing Packages in the Life Cycle

Note: The instances of Linux in this section refer to both the Linux and zLinux operating environments.

This section contains the following topics:

Sequential Movement of Packages

Packages typically travel sequentially from one state to the next in a life cycle. Returning packages to a previous state also typically occurs sequentially. The sequential movement of packages is a straightforward way to manage versions; however, you might want to demote a package across multiple states. You can demote packages across multiple states if the states *share* the same view. In the following example, if QA finds a problem, it is simplest to return the package directly to Development for additional work, rather than demoting it to Unit Test and then Development. Skipping the Unit Test state is possible as long as both Unit Test and Development share the same view.

Example: Move Packages Sequentially

Consider a simple life cycle with four states: Development, Unit Test, QA, and Release. The following promote processes move a package sequentially through this life cycle:

1. Development to Unit Test
2. Unit Test to QA
3. QA to Release

Corresponding demote processes return a package sequentially:

1. Release to QA
2. QA to Unit Test
3. Unit Test to Development

How Skipping States Creates Confusion

When packages skip states in a life cycle, a confusing situation can occur in which package versions do not appear in the view associated with the state that the package skipped. To avoid confusion, follow this rule: Always move packages forward or backward sequentially through the life cycle from one state to the next, unless two or more states share the same view.

Example: Avoid Skipping States

Each state is associated with a different view in this example of a life cycle sequence:

1. Development state—Development view
2. Unit Test state—Unit Test view
3. QA state—QA view

The confusion occurs when you do the following:

1. Promote a package from Development to Unit Test and then to QA.

   Each view is updated with the package versions.

2. Demote the package from QA to Development, skipping the Unit Test state.

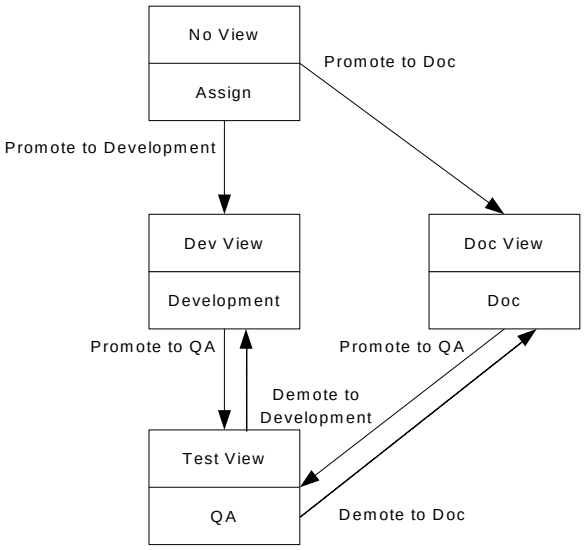
   The demote process removes the versions from the QA view, but the versions still appear in the Unit Test view, although the current state of the package is Development.

A similar situation occurs if you promote a package from Development to QA. The promote process updates the QA view with the package versions, but it does not update the Unit Test view.

Package Movement to Multiple States

Packages can move to more than one next state if multiple promote processes have been set up. You can use this setup to move packages along different paths.

The following diagram shows how users in the Assign state can send a package to either the Development or Doc states, depending on whether the package pertains to software or to documentation. Packages in both Development and Doc are then promoted to QA for quality assurance testing and can be demoted if the packages need more work.

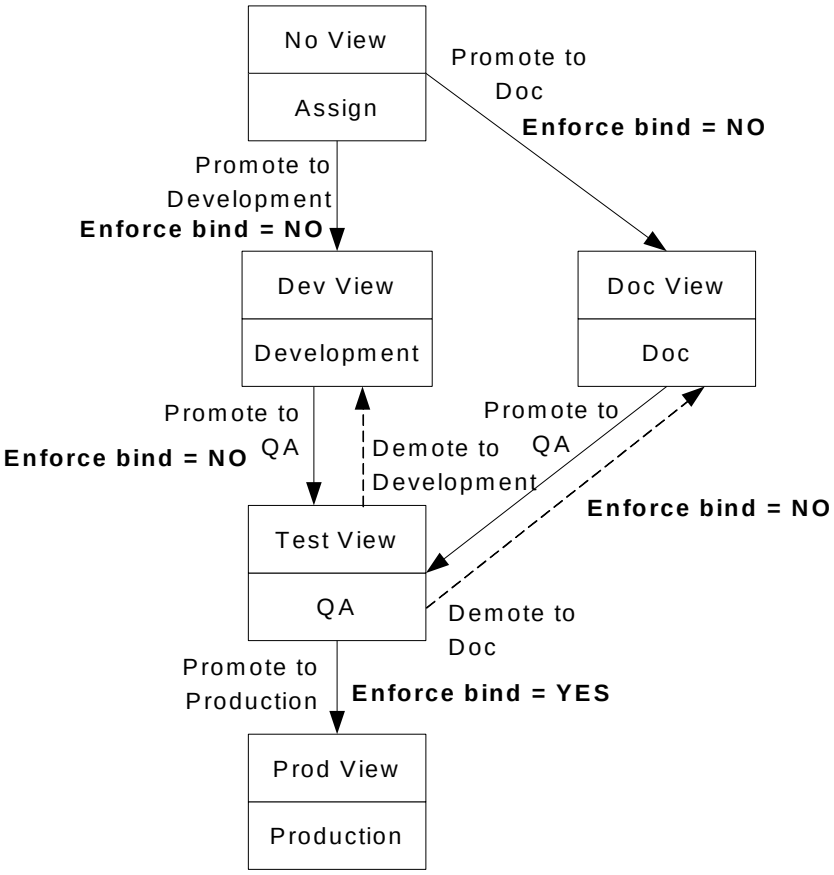


Bind Packages

The bind packages feature enforces the following restrictions on packages belonging to a package group:

- A package can belong to only one bound package group.
- If a package is part of a bound package group and if Enforce Bind is selected on the Demote or Promote Properties dialog, all packages in the group must be demoted or promoted together.
- You cannot promote or demote a single package that belongs to a bound package group while Enforce Package Bind is selected. To promote or demote a single package that is a member of a bound package group, either you must not select Enforce Package Bind or remove the package from the bound package group.

Typically, gathering packages in one state becomes important during the final stages of the life cycle. The following diagram shows how you might not enforce package binding during the early stages of the life cycle to allow flexibility of package movement; later, you might restrict the movement by enforcing package binding. Notice the enforce package bind option is selected only during promotion from QA to Production. This gathers the packages in their bound package group in the final state, ensuring that Production has all necessary packages for the group.



Approve Process

The approve process lets designated users approve or reject a package so that it can be promoted or moved. When more than one approval process is defined in a state, the approval processes can be alternatives to one another, so a package can be promoted or moved as soon as one approval is complete.

A process can require the approval of individual users and user groups. If the approval of a group is required, only one user in the group needs to execute the approval process. If a member of a group rejects a package, that user must approve the package to remove the rejection; the approval of another user in the group does not override it.

The approval of a user can be required in addition to that of a user group to which the user belongs. When the user approves a package, both requirements are met in one execution.

Approval or rejection of a package group results in an approval or rejection of all packages that are included in the package group in the state the approval or rejection is applied.

Linked processes execute before and after the approval process completes successfully, meaning that all packages are approved. If the approval of any selected package fails, linked processes do not execute.

Approve a Package

The approve process lets designated users approve or reject a package so that it can be promoted or moved.

Follow these steps:

1. Navigate the Workbench to the package you want to approve.
2. Right-click the package, and select *approve process* from the shortcut menu.

   The *approve process* dialog appears and lists the package you selected on the Workbench.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. (Optional) Do one of the following to add or remove a package from the approval list:
   - Click Add (the plus [+] sign) to select one or more packages from the current project or state.
   - Click Remove (the minus [-] sign) after selecting a package.

   The selected packages are added or removed from the approval list.

4. Ensure that the Approve option is selected.
5. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.

   Click OK.

   The package is approved.

More information:

Reject a Package (see page 210)

Reject a Package

You can reject a package if it does not meet standards. If a member of a group rejects a package, that particular user must approve the package to remove the rejection; the approval of another user in the group does not override it. If it is necessary to override a rejection, a second approval process can be added.

Follow these steps:

1. Navigate to the package you want to reject.
2. Right-click the package, and select *approve process* from the shortcut menu.

   The *approve process* dialog appears and lists the package you selected.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. (Optional) Do one of the following to add or remove a package from the rejection list:

   - Click Add (the plus [+] sign) to open the Select Packages dialog from which you can select packages in the current state. Click OK.
   - Click Remove (the minus [-] sign) after selecting a package.

   The selected packages are added or removed from the rejection list.
4. Select Reject from the Options group on the dialog.

5. (Optional) Click the tabs to enter and view information:

   Comment

   Specifies comments.

   Note

   Provides notes about the process.

   Click OK.

   The package is rejected.

Promote Process

The promote process lets you send one or more packages in the current state to the next state in the life cycle. If you select multiple packages for promotion, they are promoted as a unit. When you promote a package to a state with another view, all its changes become visible in that view.

These conditions affect whether you can promote a package:

- If a user has rejected a package, you cannot promote the package until that user executes the approval process.
- If one or more approval processes exist in a state, you cannot promote a package until all approvals have been given for at least one approve process.
- If a package is part of a bound package group, all packages in the group must be promoted together. To promote a package that belongs to a bound package group, you must first unbind the package.
- If the promote properties does not allow unmerged packages to be promoted, the latest change for a package must not be on a branch.
- If the promotion includes a change in view, the package being promoted cannot include reserved items.

  Note: For the purpose of this restriction, promoting from a state with a view to a state without a view is considered changing views.

The promote process also performs the following verifications if you promote packages from a state with a view to a state with a different view.

- A version of an item cannot be promoted unless its path version is also being promoted or the path version has already been promoted to the destination view.

- Items in packages being promoted may not have an item name conflict with items in the destination view. An item name conflict occurs when the latest versions of two different items in the same path have the same name.
- If the Verify Package Dependency option is selected, you cannot promote a package with a higher item-version without also promoting the packages with the lower item-versions in the current view, unless the lower item-versions already exist in the destination view. The lower item-version causes a dependency error only when it is on the trunk.

Promote a Package

The promote process lets you move one or more packages in the current state to the next state in the lifecycle. If an approval process exists for a state, approvals are verified before a package can be promoted. When you promote a package to a state with another view, all its changes become visible in that view.

If a package is part of a bound package group, all packages in the group must be promoted together. To promote a single package that belongs to a bound package group, you must first unbind the package.

Follow these steps:

1. Navigate to the package you want to promote.
2. Right-click the package, and select *promote process* from the shortcut menu.

   The *promote process* dialog appears and lists the package you selected.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. (Optional) Click Add (the plus [+] sign) to open the Select Packages dialog from which you can select packages in the current state. Click OK.

   The packages are added to the promotion list.
4. (Optional) Select packages from the list, and click Remove (the minus [-] sign)

   The packages are removed from the promotion list.
5. (Optional) Select options for promoting the package:

   Note: If an option is enabled by default, it is enforced and you cannot override it.

   Enforce Package Bind

   Promotes all the packages belonging to a bound package group together.

Enforce Package Merge

Prohibits promoting packages to the next state if the packages are associated with branch versions. If the promote process is in a state with no view, or if that state's view is the same as the one in the Promote To state, this option is not enforced.

Verify Package Dependency

Prohibits promoting packages that depend upon other packages. The dependency is based on versions in the view. You cannot promote a package with a higher item-version without also promoting the packages with the lower item-versions in the current view unless the lower item-versions already exist in the view of the destination state. If the promote process is in a state that shares the same view as the one in the destination state, this option is not enforced.

Note: The lower item-version causes a dependency error only when it is on the trunk.

Add Dependent Packages

Allows adding dependent packages also to the list of packages being promoted. To avoid conflict, it is recommended to promote all the dependent packages together. This option retrieves details of the dependent packages and lists the details in the Dependent Packages tab. The details include the list of dependent packages, their associated versions which made them dependent, and all other versions.

6. (Optional) Select the Approve before promoting option to approve the package before promoting it.

   Note: If multiple approve processes are defined for the state, a menu appears that lets you select the approve process you want to use.

7. (Optional) Click Note to view notes about the process.
8. Click OK.

   The package is promoted and appears in the destination state.

Demote Process

The demote process lets you send one or more packages in the current state to a previous state in the life cycle. If approvals are required for promotion, the approval status of the package is reset when you demote it. If you select multiple packages for demotion, they are handled as a unit—either all or none are demoted. If a package is part of a bound package group, you must demote all packages in the group together. To demote a single package that belongs to a bound package group, you must first unbind the package.

Demote and Versions in Views

When you demote a package from the current state and view to a prior state in the life cycle with a different view, all versions contained by the package being demoted are removed from the current view.

The possibility of “out of view” versions exists in a life cycle. These are versions that were created in a view, but that currently do not belong to a view. “Out of view” versions can occur when you demote a package, which has been modified in a working view, to a state that has no view. The demote process removes from the view all of the versions associated with the demoted package. The demote process can also remove new versions created in that view. When the package moves to a state with no view, those removed versions are not visible in any working view. To make the versions visible, the package associated with the versions must be promoted or demoted to a state that has a working view.

The presence of a reserved or merge-tagged version in the package selected for demotion causes the demote process to fail if the demotion includes a change in view. If you attempt to demote a package to a state in a different view, and the package has merged or reserved versions associated with it, the demote process fails and an error message is generated.

Demote also performs the following verifications if packages are being demoted from a state with a view to a state with a different view:

- A version of an item path cannot be demoted unless there are no versions in the current view that reference it, or those versions are also being demoted.
- If demoting the selected packages results in item name conflicts in the current view, the demote will fail.
- The Verify Package Dependency option prevents demotion of packages upon which other packages depend. Dependency is based on versions in the current view. In the current state, a package with a lower item-version cannot be demoted without also demoting the packages with the higher item-versions in the current view. (If the destination state shares the view, this option is not enforced).

Demote a Package

The demote process lets you move one or more packages in the current state to a previous state in the lifecycle.

Follow these steps:

1. Navigate to the package you want to demote.
2. Right-click the package, and select *demote process* from the shortcut menu.

   The *demote process* dialog appears and lists the package you selected. If a package is part of a bound package group, all packages in the group must be demoted together. To demote a package that belongs to a bound package group, you must first unbind the package.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. (Optional) Click Add (the plus [+] sign) to open the Select Packages dialog from which you can select packages in the current state. Click OK.

   The packages are added to the demotion list.
4. (Optional) Select packages from the list, and click Remove (the minus [-] sign).

   The packages are removed from the demotion list.
5. (Optional) Select demotion options:

   Note: If an option is enabled by default, it is enforced and you cannot override it.

   Enforce Package Bind

   Enforces that you demote all the packages belonging to a bound package group together; otherwise, an error occurs.

Verify Package Dependency

Prohibits you from demoting packages that depend upon other packages. The dependency is based on versions in the view. You cannot demote a package with a lower item-version without also demoting the packages with the higher item-versions in the current view unless the higher item-versions already exist in the view of the destination state. If the demote process is in a state that shares the same view as the one in the destination state, this option is not enforced.

Note: The higher item-version causes a dependency error when it is on the trunk or on the branch.

Add Dependent Packages

Allows adding dependent packages also to the list of packages being demoted. To avoid conflict, it is recommended to demote all the dependent packages together. This option retrieves details of the dependent packages and lists the details in the Dependent Packages tab. The details include the list of dependent packages, their associated versions which made them dependent, and all other versions.

Demotion options are specified.

6. (Optional) Click Note to view information about the process.
7. Click OK.

   The package is demoted and appears in the destination state.

Package Dependency Report

The Package Dependency Report lets you view the dependent packages of packages and provides the ability to add all the dependent packages, while executing the Promote or Demote process. Using this feature, you can ensure that you promote or demote all the dependent packages together. Thus, you can also avoid any inconsistency in the code.

To maintain consistency, promote and demote the dependent packages together because there could be dependent package for any package.

You can view the package dependency report while performing the following actions:

- Promote packages
- Demote Packages

While performing the Promote or Demote process, you can view the dependent packages details in the Dependent Package tab of the Promote or Demote dialog. Optionally, you can add all the dependent packages to the promote or demote list and promote or demote all the dependent packages together.

View, Add, and Promote Dependent Packages

Follow these steps:

1. Open SCM Explorer.
2. Connect to the broker.
3. Navigate to a state node under a project and expand the state node.
4. Expand the Packages node under the state node.
5. Right click the required package and select the appropriate promote process.
   The Promote Dialog opens.
6. Click the Dependent Packages tab to view the package dependency report.
7. Select the Show Version Details check box
8. Select a package in the Dependent Packages list to view the versions causing dependency and all others versions available in that package.
   The version details appear in the Versions Causing Dependency list and the All Other Versions list.
9. Click the Add All To Promote List button.
   All the dependent packages are added to the Promote List and the Option tab is displayed.

   Note: If the dependent packages are present in other states, appropriate message is displayed.
10. Click OK.
    All the dependent packages are promoted together with the selected package.

View, Add, and Demote Dependent Packages

Follow these steps:

1. Open SCM Explorer.
2. Connect to the broker.
3. Navigate to a state node under a project and expand the state node.
4. Expand the Packages node under the state node.
5. Right click the required package and select the appropriate demote process.
   The Demote Dialog opens.
6. Click the Dependent Packages tab to view the package dependency report.
7. Select the Show Version Details check box.
8. Select a package in the Dependent Packages list to view the versions causing dependency and all others versions available in that package.

   The version details appear in the Versions Causing Dependency list and the All Other Versions list.
9. Click the Add All To Demote List button.
   All the dependent packages are added to the Demote List and the Option tab is displayed.

   Note: If the dependent packages are present in other states, appropriate message is displayed.
10. Click OK
    All the dependent packages are demoted together with the selected package.

Add and Promote or Demote Dependent Packages Without Viewing Package Dependency Report

Follow these steps:

1. Open SCM Explorer.
2. Connect to the broker.
3. Navigate to a state node under a project and expand the state node.
4. Expand the Packages node under the state node.
5. Right click the required package and select the appropriate promote or demote process.
   The Promote or Demote Dialog opens.
6. Select the Add Dependent Packages check box.
   All the dependent packages are added to the Promote or Demote list.
7. Click OK.
   All the dependent packages and the selected package are promoted or demoted.

Move Package Process

The life span of a package is usually limited to one project; however, you can create packages in one project and then move them to another project to begin a new life cycle. The move package process moves one or more packages from the current project and state to a state in another project. This process moves only the package definition and history, not any changes made in the first project. You cannot move changes associated with packages with this process.

Moving packages is typically used to link a problem-tracking project to a change management project. For example, a customer support group records problems using CA Harvest SCM forms. Packages are created in the support project and associated with various problem forms. These packages can move through different states in the support project until they are ready to be assigned. Rather than creating a package in the development project, the original package with its history intact and any form associations can be moved from the support project to the development project. If multiple packages are selected, they are all moved as a unit.

For the package process to succeed, CA Harvest SCM verifies that all packages selected meet the following criteria:

- No versions associated with the package can exist in the source project.
- A package with the same name cannot exist in the target project.
- If an approval process is in place for a state, you cannot move a package until it has been approved.
- The user executing the process must have access to create packages in the destination project.

The following list indicates what happens to various package attributes in the destination project:

- The package note is retained.
- All form associations are maintained.
- The package groups are not carried across to the destination project because package groups are not global objects. You must edit the package in the destination project to add it to any groups.
- The package history is moved with the package if the appropriate option is selected, and an additional record is added to the history indicating when it was moved.

When you use the move package with the Keep Source Package option, a duplicate of the source package is created in the source project and the same form is associated with both packages. Because the package relationship is maintained by the associated form, a new form property page is also added. Both the packages share the same history and after the move both are independent packages linked only through forms.

Move a Package to a Different Project

The move package process lets you move one or more packages from the current project and state to a state in another project. This process moves only the package definition and history, not any changes made in the first project. Changes associated with packages cannot be moved with this process.

Follow these steps:

1. Navigate to the package you want to move.
2. Right-click the package, and select *move package process* from the shortcut menu.

   The *move package process* dialog appears and lists the package you selected.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.

3. (Optional) Click Add (the plus [+] sign) to open the Select Packages dialog from which you can select packages in the current state. Click OK.

   The packages are added to the move package list.

4. (Optional) Select packages from the list, and click Remove (the minus [-] sign).

   The packages are removed from the move package list.

5. Select a target project and state by using the drop-down lists.

   The target project and state are specified.

6. (Optional) Select the Include Package History option.

   All the package history records that have been created in former projects are included in the moved package.

   Note: If you no longer require the history, excluding it can reduce processing time.

7. (Optional) Select the Keep Source Package option.

   Duplicates the source package and the same form is associated with both packages. Because the package relationship is maintained by the associated form, a form property page is also added. Both the packages share the same history and after the move both are independent packages linked only through forms.

8. (Optional) Click Note to view information about the process.

   Click OK.

   The package appears in the destination state of the destination project.

Execute a User-Defined Process

The user-defined process (UDP) lets you invoke an external program to run as a process in your lifecycle. The Administrator defines the program to execute, any command line parameters, and the output options. You cannot modify them.

For programs that read from the default input device, default input parameters are specified. If the administrator defines this field as editable, you can override or modify them at execution time. You can also supply additional command line parameters.

Note: Processes invoked from CA Harvest SCM are executed in synchronous mode. After you execute a user-defined process, you cannot initiate any other action from the keyboard until it completes.

Follow these steps:

1. Navigate to the package or version you want to use to execute a UDP.
2. Right-click the package or versions, and select the UDP process from the shortcut menu.

   The *user-defined process* dialog appears and the [package] or [version] system variables are expanded to a list.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. (Optional) Enter any additional command line parameters to the program executed by this UDP in the Additional Command Line Parameters field.

   The additional parameters tailor the command invocation.

   Note: This option is available only if the Administrator allows any additional parameters for this process.
4. (Optional) Edit the default input parameters in the Default Input field.

   Note: This field applies only to programs that read from the standard input device. If you defined the default input as editable in the process Properties dialog, this field becomes modifiable.
5. (Optional) Click Note to view information about the process.

   Click OK.

   The user-defined process executes.

Notify Process

The notify process lets you send a mail message to individual users or all members of one or more user groups. When a notify process is set up, a default message can be specified and default users and groups to notify. These defaults determine how the execution dialog appears when it is first accessed. You can override the defaults.

Note: If two or more users share the same email account only one mail message is generated. Duplicate mail messages are not generated when the same user is a member of multiple user groups.

The mail program to use and the destination of any program output are also specified during setup, but these aspects of the notify process cannot be modified at execution time.

- On UNIX and Linux, the mail program executes on the client.
- On Windows, the mail program executes from the server. The message header generated by the mail program records that the user who started the server process sent the message. A line following the header displays the name of the client user who initiated the notification process.

Notify Users

The notify process lets you send a mail message to individual users or all members of one or more user groups.

Follow these steps:

1. Navigate the Workbench to the state that has the notify process you want to execute.
2. Right-click the state, and select *notify process* from the shortcut menu.

   The *notify process* dialog appears.

   Note: The lowercase *italic* text indicates the process dialog name by the process type, since the process execution dialog names differ to each site.
3. Complete the dialog fields:

   Mail Utility

   Names a mail program and any arguments you want to supply when the program is invoked. Specify a full path to the program.

   - On UNIX and Linux clients, CA Harvest SCM searches for a path in the PATH variable of the user executing the process.
   - On Windows clients, the mail program is executed on the server and must be in the path of the user who started the server.

   Subject

   Specifies a default subject. The subject appears on the subject line of the email message when it is sent. You can modify the content of this field. For AIX and Linux platforms, the Subject field cannot contain double quotes.

   Mail Message

   Specifies a message to be sent by this notify process.

   For example, if this process is going to be used to notify a manager that a package has been promoted, you can specify the appropriate message in this field. If the notify process is being linked to another process, you can use system variables in the messages to represent various parameters.

   When you use the package or version system variable in this field, you can right-click packages on the Workbench and select a notify process from the shortcut menu; the package or version names are listed in the notify message.

   Display

   Specifies a default action to be taken with any general or error outputs generated by the notify process by selecting an option in the Output and Errors menus. The choices are Display or Discard. All output is automatically appended to the output log if the Display choice is selected.

Users to Notify

Specifies the users who receive the notification. You can add a user to the list by clicking Add, selecting the user, and clicking OK. You can remove a user from the list by selecting the user and clicking Remove.

User Groups to Notify

Specifies the user groups who are notified. To be notified, users in the group must have Use Project access to the project unless they are listed in the Users to Notify list. You can add a user group to the list by clicking Add, selecting the user group, and clicking OK. You can remove a user group from the list by selecting the user group and clicking Remove.

4. (Optional) Click Note to view information about the process. Click OK.

   The mail message is sent.

Chapter 13: Finding Objects and Reporting

This section contains the following topics:

Find Objects

Use the Find and Filter dialogs to locate objects by specifying filtering criteria. The dialogs display search results in a list of available objects of a certain type, and you can execute available functions on your selection. For example, after locating all packages awaiting your approval, you can execute the approve process by right-clicking each package and selecting the approve process from the shortcut menu to open the approve process execution dialog.

Follow these steps:

1. Open a Find or Filter dialog.
2. Enter or select criteria, and click the execution option.

   Results are listed.
3. (Optional) Right-click one or more objects in the list to open a shortcut menu with functions you can use on the object.

Filter the Packages View

The Packages View lets you filter packages to narrow your package selection. You can define filter settings to use, and save and remove them, or use the default filters:

All Packages

Shows all the packages for the broker.

My Assigned Packages

Shows packages assigned to you.

Follow these steps:

1. Click the Packages View.

   The Filter appears.

2. Select a filter from the Filter drop-down list, and click Find.

   The results appear in the results list.

Follow these steps:

1. Click the Packages View.

   The Filter appears.

2. Specify search criteria, and click Save Filter.

   The Save Filter dialog appears.

3. Enter a name for the filter, and click OK.

   Note: If the filter name already exists, you are prompted about whether to overwrite the existing filter.

   The filter is saved and is available in the Filter drop-down list.

Follow these steps:

1. Click the Packages View.

   The Filter appears.

2. Select a filter setting from the Filter drop-down list, and click Remove Filter.

   A confirmation dialog appears.

3. Click Yes.

   After the next Filter operation is executed, the filter setting no longer appears in the Filter drop-down list.

Show All Packages

The All Packages filter shows all packages for the broker. Use All Packages to locate all packages. You can use the Advanced Options to narrow your selection.

Follow these steps:

1. Click the Packages View.
2. Select All Packages from the Filter drop-down list, and click Find.

   The packages for the broker are listed in the Packages list, and you can perform CA Harvest SCM actions on them.

Show My Assigned Packages

The My Assigned Packages filter shows packages assigned to you, regardless of project or state context. Use My Assigned Packages to locate all of your packages. You can use the Advanced Options to narrow your selection.

Follow these steps:

1. Click the Packages View.
2. Select My Assigned Packages from the Filter drop-down list, and click Find.

   The packages assigned to you are listed in the Packages list, and you can perform CA Harvest SCM actions on them.

Filter Packages by Package Name

The Filter Packages feature lets you filter packages by package name. The Name field is not case-sensitive, for example, entering N* locates packages whose names begin with N or n.

Follow these steps:

1. Click the Packages View.
2. Enter a package name in the Name field.

   Note: You can use wildcards. You can specify multiple package names at the same time by separating the names with a semicolon. You can use the question mark (?) for single character matching.
3. Verify that the Use Advanced Options check box is not selected.

   Click Find.

   The packages matching the filtering criteria are displayed in the results list.

Filter Packages by Project Status

The Project Status field has two options--Active and All. When you select the Active option, only Active projects are considered for package search. If you select the All option, both active and inactive projects are considered for package search.

Note: By default, the Active option is selected. When you select the All option and select the Use Advanced Options check-box, *only* active projects are considered for the search.

Follow these steps:

1. Click the Packages View.
2. Enter a package name in the Name field.

   Note: You can use wildcards. You can specify multiple package names at the same time by separating the names with a semicolon. You can use the question mark (?) for single character matching.
3. Select the project status option.
4. Click Find.

   The packages matching the filtering criteria are displayed in the results list.

Filter Packages Using Advanced Options

The Filter Packages feature lets you execute complex filtering operations to locate packages with common attributes.

Follow these steps:

1. Click the Packages View.
2. Select All Packages from the Filter drop-down list.
3. Select Use Advanced Options.

   The Advanced Options are enabled.
4. Complete the fields that you want to use for filtering:

   Context

   Specifies a context to filter for packages.

   Attributes

   Locates packages that were created, assigned to, or modified by a specific user.

   Note

   Locates packages according to a text substring value that you enter.

   Important! You can do compound searches by separating text substring values with a blank space.

Description

Locates packages according to a text substring value that you enter.

Important! You can do compound searches by separating text substring values with a blank space.

Dates

Specifies a date to locate packages.

Approval Status

Locates packages that are based on their status.

Click Find.

The packages matching the filtering criteria display in the results list.

Filtering Packages with and without Changes

The following new options are available in the Packages view in Workbench:

- Empty Packages Only
- Non-Empty Packages Only
- Both

The third option is the default option.

Empty Packages Only

This option selects the packages which are empty in conjunction with the other filter criteria specified. This helps to identify the packages which no longer have any versions and can be deleted or to carry forward any package history changes to another project using Move package process.

Non-Empty Packages Only

This option chooses the packages which are non-empty in conjunction with the other filter criteria specified.

Both

This option is the default option and selects both packages with versions and without versions in conjunction with the other filter criteria specified.

Use Package Filter Results

Package filter results lets you execute actions without exiting the results list.

To perform actions on package filter results, right-click a package in the results list and select an option from the shortcut menu:

Compare with Trunk

Opens the Compare tool, which shows the differences between the package's branch version and its parent trunk version.

Rename

Specifies a new name for the package.

Add New Form

Opens the Add New Form dialog, which lets you add and associate a form to the package.

New Package Group

Opens the New Package Group dialog, which lets you create or modify package group associations.

Locate Package in Explorer

Expands the Explorer View tree (if necessary) and highlights the package in the tree.

Save List As

Opens a dialog that lets you save your current search results to a specified name and location.

Set as Workspace Context Package

Uses the package to set the workspace package context.

View Forms

Opens a dialog that lists the package's associated forms. You can select one or more forms, and click Open to view or modify the forms in the form editor.

Properties

Opens the package Properties dialog, which lets you view and modify the package attributes.

Processes

Executes package processes.

Note: The selected package must exist in the workspace.

The action is performed according to your selection.

Find Files

The Find File dialog lets you locate files and select files from a list. It also supports multiple filtering operations to let you search precisely. In addition, you can use the Find File dialog simply to obtain file information, which can be output to the Output Log.

You can open the Find File dialog from a number of process execution dialogs by clicking the browser button on the dialog. For example, to open the Find File dialog from the check-in process dialog click the plus sign.

Follow these steps:

1. Click the browser button next to a files list on a process execution dialog.

   The Find File dialog appears.

2. Complete the dialog fields:

   Name

   Specifies a name or naming pattern; this enables Find.

   Note: You can use any number of wild cards (*) in any position for multiple character matching. For single character matching you can use the question mark (?). You can specify multiple file types at the same time by separating them using the OR symbol (|). For example, you can enter: *.cpp|*.h|*.txt to locate cpp, h, and txt files.

   Look in

   Specifies a directory to search for a file. Clicking the button next to this field opens a dialog, which lists directories in the current client directory. You can navigate through the file system by double-clicking a directory or by selecting a directory and clicking OK.

   Include subfolders

   Shows files in every directory below the current one that match the file name filtering criteria.

   When you select Include subfolders and click Find, CA Harvest SCM expands each directory so that the file names displayed in the list include relative paths from your current position.

   Date

   Specifies a date range for the search. By default, the All files option is selected, indicating date checking will not be performed. Selecting the Modified between option lets you select different date searches that can be performed. You can search between two dates, search the previous *x* number of months, or search the previous *x* number of days.

Use Regular Expression

Specifies the file name pattern using standard regular expression syntax.

Click Find.

The files matching the filtering criteria appear in the list and are selected.

3. Click OK.

   All selected files are returned to the calling dialog.

Filter the Versions View

The Versions View lets you filter versions to narrow your version selection. You can define filter settings to use, and save and remove them, or use the default filters:

All Versions

Shows all the versions for the broker, project, and state context.

My Unmerged Versions

Shows all your versions that are unmerged.

My Reserved Versions

Shows all your versions that are reserved.

Note: You must set a broker, project, and state context to enable the Find button.

Follow these steps:

1. Click the Versions View.

   The Filter appears.

2. Select a filter from the Filter drop-down list, and click Find.

   The results appear in the results list.

Follow these steps:

1. Click the Versions View.

   The Filter appears.

2. Specify search criteria, and click Save Filter.

   The Save Filter dialog appears.

3. Enter a name for the filter, and click OK.

   Note: If the filter name already exists, you are prompted about whether to overwrite the existing filter.

   The filter is saved and is available in the Filter drop-down list.

Follow these steps:

1. Click the Versions View.

   The Filter appears.

2. Select a filter setting from the Filter drop-down list, and click Remove Filter.

   A confirmation dialog appears.

3. Click Yes.

   After the next Filter operation is executed, the filter setting no longer appears in the Filter drop-down list.

Show My Unmerged Versions

The My Unmerged Versions filter shows all your versions that are unmerged in a state. Use My Unmerged Versions to locate all your unmerged versions in a state. The Versions View filter options are the same as in the Find Versions dialog and you can use them to narrow your selection.

Follow these steps:

1. Click the Versions View.
2. Select My Unmerged Versions from the Filter drop-down list.
3. Select search options, and select a project and state from the Project and State drop-down lists, respectively.

   The context is set.

4. Click Find.

   Your versions that are unmerged in the project and state context are listed in the Versions list, and you can perform CA Harvest SCM actions on them.

Show My Reserved Versions

The My Reserved Versions filter shows all versions that you reserved in a state. Use My Reserved Versions to locate all the reserved versions you reserved in a state. The Versions View filter options are the same as in the Find Versions dialog and you can use them to narrow your selection.

Follow these steps:

1. Click the Versions View.
2. Select My Reserved Versions from the Filter drop-down list.

3. Select search options, and select a project and state from the Project and State drop-down lists, respectively.

   The context is set.

4. Click Find.

   The versions you reserved in the project and state context are listed in the Versions list, and you can perform CA Harvest SCM actions on them.

Find Versions

The Find Version dialog lets you select item versions from a list. The various options on this dialog work with one another to let you select versions according to multiple criteria. In addition, you can use the Find Version dialog to obtain version information.

Follow these steps:

1. Click Tools, Find Version.

   The Find Version dialog appears.

2. Specify a context for the search.

3. Complete the fields in the dialog:

   Name

   Filters versions according to a naming pattern. You can use any number of wild cards (*) in any position for multiple character matching. You can use the question mark (?) for single character matching. You can specify multiple version names at the same time by separating the names with a semicolon. The Name field is not case-sensitive, for example, entering N* locates versions with names that begin with N or n.

   Package

   Filters versions that are based on the name of the package that created them. Clicking the button next to the Package field opens the Select a Package dialog listing all packages in the current context. This filter is not available when viewing versions in snapshot views.

   Package Group

   Filters versions that are based on the packages from the package group in the selected state. Clicking the button next to the Package Group field opens a Package Group dialog listing all package groups in the current context. This filter is not available when viewing versions in snapshot views.

   Providing the Package or Package Group name is mutually exclusive.

View Path

Changes the current location to any position in the current path. You can change the path by clicking the button next to this field. The Repository Path Selection dialog appears and lets you browse through available paths and select a location.

Include subfolders

Specifies versions that match the other filtering criteria in every path below the current one. You can then select versions from multiple paths to include in this operation. By selecting this option and clicking Find displays expanded paths displays the item names and their relative paths from your current position.

Word/Phrase in File

Specifies text to search for in text files as the Storage Type buttons indicate.

Version/View

Filters the versions in one of the following ways.

If you select the *package* option, the following attributes are available:	**If you select the *package* group option, the following attributes are available:**
Latest in View	Latest in View
Latest in Package	Latest in Package Group
Latest	Latest
All in View	All in View
All in Package	All in Package Group
All	All

Based on your selection for the package or package group, the following attributes in the version/view are available.

Latest in View

Displays the latest versions for each item in the current view that match the other filtering criteria. Only one version per item displays. This option is incompatible with viewing branch versions because they are not in a view. When selected, the only Branch option available is Trunk Only.

Latest in Package

Displays the latest versions that are associated with the package specified in the Package field. This option is available only when the Package field is populated.

Latest

Displays the latest version for each item that matches the other filtering criteria. Versions that do not exist in the view but are associated with the current state also displays. Only one version per item displays. This option is not available when viewing versions in snapshot views.

All in View

Displays all versions matching the other filtering criteria that exist in the view that is associated with the current state. Only promoted versions display in another view. This option is incompatible with viewing branch versions because they are not in a view.

All in Package

Displays all versions that are associated with the package specified in the Package field. This option is available only when the Package field is populated.

All

Displays all versions. Versions include the branch and trunk even if that do not exist in the view but are associated with current state. This option is not available when viewing versions in snapshot views.

Latest in Package Group

Displays the latest versions that are associated with the packages within the package group specified in the Package group field.

All in Package Group

Displays all versions that are associated with the packages within the package group specified in the Package group field.

Description

Specifies text to search for in the description of the version. You can select the case-sensitive check box based on your search requirements.

For example: If you search for *No* keyword with the case-sensitive option selected, the search results display all the possible combinations like NO, no, No, or nO.

4. Click Find.

   The versions matching the filtering criteria are listed.

5. (For selecting versions only) Click OK to return all files to the calling dialog.

More information:

Version Attributes (see page 96)
Version Tags (see page 96)
Views and Versions (see page 100)
Find Version Latest Filter (see page 240)
Find Version Filter Combinations (see page 241)

Find Version - Results

The versions that are displayed in the result pane of the Find version has more attributes information included for the following values:

- Name
- Path
- Version
- Stored As
- Status
- Package/Packagegroup
- Creator
- Created on
- Modifier
- Modified On
- Data size
- Description

Examples

The following examples provide more insight into the filters usage that is based on your selection for the package or package group.

Example 1:

This example depicts the use case when you select the package option.

Follow these steps:

1. Maintain an SCM project with the Development, Test, and Release views.
2. Create pack1, pack2, and pack3 as three packages in the Development view of the project.

3. Maintain few versions of a sample.java file in the Development view as follows:
   - sample.java (1) N in pack1
   - sample.java (1.1.1) N in pack2
   - sample.java (2) N in pack2
   - sample.java (3) N in pack3
4. Promote pack3 to the Test state.
5. Create another trunk version in the Test view as sample.java(4) in pack3.
6. Promote pack3 to the Release state.
7. Create another trunk version in the Release view as sample.java(5) in pack3.
8. Select pack2 package in the Package drop-down list.

   The following table displays list of packages that display in combination of Version/View filters usage:

Filters	Packages
Latest in view	sample.java(3)-N
Latest in package	sample.java(2)-N
Latest	sample.java(5)-N
All in view	■ sample.java (0)-N ■ sample.java (1)-N ■ sample.java (1.1.1)-N ■ sample.java (2)-N ■ sample.java (3)-N
All in package	sample.java (2) -N
All	■ sample.java (0)-N ■ sample.java (1)-N ■ sample.java (1.1.1)-N ■ sample.java (2)-N ■ sample.java (3)-N ■ sample.java (4)-N ■ sample.java (5)-N

Example 2:

This example depicts the use case when you select the package group option.

Follow these steps:

1. Maintain an SCM project with the Development, Test, and Release views.
2. Create pack1, pack2, and pack3 as three packages in the Development view of the project.
3. Include pack1 and pack2 in one package group pg12.
4. Maintain few versions of a sample.java file in the Development view as follows:
   - sample.java (1) N in pack1
   - sample.java (1.1.1) N in pack2
   - sample.java (2) N in pack2
   - sample.java (3) N in pack3
5. Promote pack3 to the Test state.
6. Create another trunk version in the Test view as sample.java(4) in pack3.
7. Promote pack3 to the Release state.
8. Create another trunk version in the Release view as sample.java (5) N in pack3.
9. Select pg12 package group in the Package Group drop-down list.

   The following table displays list of packages that display in combination of Version/View filters usage:

Filters	Use Case-1	Use Case-2
	The versions displayed when you select the pg12 package group.	**The versions displayed when you select the pg12 package group and when you promote the pack2 package from Development to the Test state.**
Latest in view	sample.java(3)-N	sample.java(3)-N
Latest in package group	sample.java(2)-N	sample.java(1)-N **Note:** This selection filters the versions from the existing packages within the view. Some of the packages do not exist in this state, though they are a part of the package group.

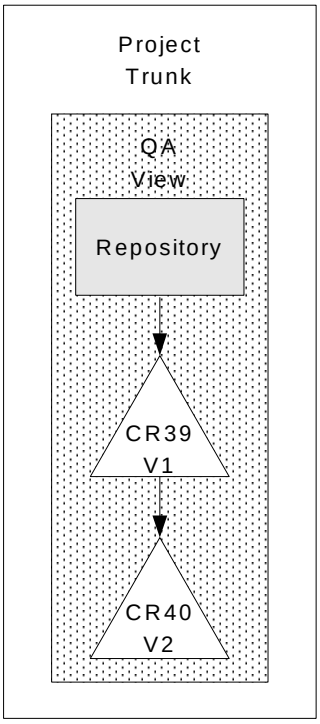
Filters	Use Case-1	Use Case-2
Latest	sample.java(5)-N	sample.java(5)-N
All in view	■ sample.java (0)-N ■ sample.java (1)-N ■ sample.java (1.1.1)-N ■ sample.java (2)-N ■ sample.java (3)-N	■ sample.java (0)-N ■ sample.java (1)-N ■ sample.java (1.1.1)-N ■ sample.java (2)-N ■ sample.java (3)-N
All in package group	■ sample.java (1) -N ■ sample.java (1.1.1)N ■ sample.java (2) -N	sample.java (1) -N
All	■ sample.java (0)-N ■ sample.java (1)-N ■ sample.java (1.1.1)-N ■ sample.java (2)-N ■ sample.java (3)-N ■ sample.java (4)-N ■ sample.java (5)-N	■ sample.java (0)-N ■ sample.java (1)-N ■ sample.java (1.1.1)-N ■ sample.java (2)-N ■ sample.java (3)-N ■ sample.java (4)-N ■ sample.java (5)-N

Find Version Latest Filter

The *latest* filter options work with the Package or Created by filters in the following ways:

- When you use the Package filter with the Latest in View filter, the items selected are all those that have changes associated with the package, but the version displayed for each item is the actual latest in the view, even if this version was not created by the package. This prevents overwriting changes when performing package check-out.
- When you use the Package filter with the Latest filter, the search returns the latest versions associated with the package, regardless of view.
- When you use Created by filter with the Latest in View Version filter, the search returns the latest version in that view only if the specified user created it; if the user did not create this version, no version is listed.
- When you use Created by filter with the Latest filter, the search returns the latest version created by the user.

Consider two packages named CR39 and CR40 that have been promoted to the QA state in the QA view. The following diagram shows the version tree for one item that was changed by each package:



In QA, a user checks out the item using the Latest in View option with CR39 as the associated package. The Package filter selects this item because it has been changed by CR39; but the Version filter selects version 2 of the item because the version filter selects the latest in view regardless of the package. As the latest version in the QA view, version 2 of the item, which includes the changes made for CR40 and CR39, is checked out.

Find Version Filter Combinations

To use the Find Version dialog effectively, you need to understand how the various filtering criteria interact. The following table shows the results of using the Find Version dialog with various filtering criteria typically used for the check-in and check-out processes. This table is a partial listing of the many possible combinations.

Item	Version/View	Package	Results
Modified and Not Modified	Latest in View		Latest version of all items in view.
Modified and Not Modified	Latest in View	Package	Latest version of all items in view where at least one version is associated with the selected package.

Item	Version/View	Package	Results
Modified and Not Modified	All in View		All version of all items in view.
Modified and Not Modified	All in View	Package	All versions of all items in view where at least one version of each item is associated with the selected package.
Modified and Not Modified	All		All versions of all items in the selected project.
Modified and Not Modified	All	Package	All versions of all items in the selected project where at least one version of each item is associated with the selected package.
Modified and Not Modified	Latest		Latest versions of all items in the selected project.
Modified and Not Modified	Latest	Package	Latest versions of all items in the selected project where at least one version of each item is associated with the selected package.
Modified	Latest in View		Latest versions of modified items in view.
Modified	Latest in View	Package	Latest versions of modified items in view where at least one version of each item is associated with the selected package.
Modified	All in View		All versions of modified items in view.
Modified	All in View	Package	All versions of modified items in view where at least one version of each item is associated with the selected package.
Modified	All		All versions of modified items in the selected project.
Modified	All	Package	All versions of modified items in the selected project where at least one version of each item is associated with the selected package.
Modified	Latest		Latest versions of modified items in the selected project.
Modified	Latest	Package	Latest versions of modified items in the selected project where at least one version of each item is associated with the selected package.

More information:

Check In Files (see page 115)
Check Out Versions (see page 110)

Find Forms

Forms exist at the highest level and are available in all projects defined in a CA Harvest SCM installation; you can access them without any context. Packages exist in a state and project. In these contexts, you can use the associated packages to determine which forms to display:

- If your context is at the project level (no state specified), only forms associated with packages in this project display.
- If your context is at the state level, only forms associated with packages in this project currently located in this state display.

The Find Form dialog lets you execute complex filtering operations to locate forms with common attributes. You can also use the Find Form dialog to select forms according to their association with packages.

Follow these steps:

1. Click Tools, Find Form.

   The Find Form dialog appears.

2. Complete the fields in the dialog. Following are descriptions of fields that are not self-explanatory:

   Note: Form fields in the Find Form dialog are not case-sensitive; however, information is stored exactly as it is entered.

   Name

   Filters forms according to a naming pattern. If you leave the default wild card (*), forms display regardless of name. You can use any number of wild cards (*) in any position for multiple character matching. The use of the question mark (?) is also supported for single character matching. The Name field is not case-sensitive, for example, entering N* shows forms whose names begin with N or n. You can specify multiple form names at the same time by separating the names with a semicolon.

Form Type

Specifies the type of form to display to filter by its fields. When the specified form displays, you can query on the various fields of that form.

You do not need to know the precise value for a field to use it as a filter. You can use asterisks (*) in any field for wild card matching. For example, you might want to search for all Problem Reports that mention performance anywhere in the problem description. To do this, enter *performance* in the Problem Description field. CA Harvest SCM returns all matching Problem Reports to the Form dialog's list.

In general, you can use trailing wild cards in short text fields such as Category and Hardware on the Problem Report form. For longer fields such as Problem Description, you can use any number of wild cards (*) in any position for multiple character matching. The question mark (?) is also supported for single character matching.

Click Find.

The forms matching the filtering criteria are listed.

Note: Until you click Find the results are not refreshed to your specified settings.

Locate Package from Form List

You can select the list of associated packages to be displayed from the Find Form results search based on your criteria.

From the displayed list of associated packages, right-click on any package and select Locate Package in the explorer. This action locates the package in the explorer view and highlights the package.

Use Find Form Results

You can execute actions from the Find Form results list.

Note: Verify that the Explorer View is open before you perform actions on the displayed forms.

To perform actions on find form results, right-click a form in the results list and select an option from the shortcut menu:

Delete Form

Opens a confirmation dialog that lets you delete or cancel the deletion.

Rename

Specifies a new name for the form.

Save List As

Opens a dialog that lets you save your current search results to a specified name and location.

View Packages

Lists packages that are associated with the form.

Properties

Displays the form properties and the packages that are associated with the form.

Details Report

Shows the Form Details report in your default browser window.

Edit Form

Opens the form in the form editor, letting you view or modify the form.

Note: You can select and edit a single form or any number of forms side-by-side.

Add URL Attachment

Opens the URL Form Attachment dialog that lets you enter a URL address to reference a website from the form.

Add Attachment from Remote Agent

Opens a platform file chooser that lets you browse for and select files from a remote computer.

Add Attachment from Local File System

Opens a platform file chooser that lets you browse for and select files from your computer.

The action is performed according to your selection.

To perform actions on associated packages results, right-click a package in the results list and select an option from the shortcut menu:

Compare with Trunk

Opens the Compare tool, which shows the differences between the package's branch version and its parent trunk version.

Rename

Specifies a new name for the package.

Add New Form

Opens the Add New Form dialog, which lets you add and associate a form to the package.

New Package Group

Opens the New Package Group dialog, which lets you create or modify package group associations.

Save List As

Opens a dialog that lets you save your current search results to a specified name and location.

The action is performed according to your selection.

Filter in Find Form Dialog

You can create new filters and update existing filters using the Find Form dialog. You can open the dialog from the Tools menu or by right- clicking a Broker in the SCM Explorer view.

Save Filter or Create New Filter

After you have provided some search criteria in the Find Form dialog, click the Save Filter button to save the filter data. This action opens a dialog box and allows you to specify a name to the filter. After you specify a name, click OK to save the filter by that name.

If a filter of the same name exists, a prompt allows you to specify whether to overwrite the previously saved filter or not. If you select Yes, it overwrites the previously saved filter else, it does not overwrite and cancels the operation.

Selection of Saved Filters

A combo box displays the list of available filters for the Find Form dialog.

After you create a new filter and save it, the filter is added in the list of all saved filters and is immediately populated in the combo box.

You can select any of the filters from the combo box. The corresponding data is populated in the respective fields in the Find Form dialog.

Remove Filter

This option removes an existing filter.

After you select a filter in the filter combo box, this button is enabled. Click the Remove Filter to remove the selected filter from the combo box.

Update or Overwrite an Existing Filter

Use the Save Filter button to update or overwrite a filter.

When you select a filter from the filter combo box, the Save Filter button remains disabled. After you modify some of the search criteria, the Save Filter button is enabled.

When you click the Save Filter button, you can either update or overwrite the existing filter by saving the filter with the same name, or create a new filter by specifying a new name.

Report on Objects

BusinessObjects reports let you access your data, format it, and deliver it as reports.

Follow these steps:

1. Verify that your BusinessObjects report preference is set.
2. Click Tools, Reports from the main menu.

   The Log On to BusinessObjects InfoView page appears.
3. Log in to BusinessObjects InfoView.

   After you log in to BusinessObjects InfoView, you can view CA Harvest SCM database information organized in report formats.
4. Click the folder that corresponds to the type of report you want to view:

   Package Change Activity

   Provides information about packages and package activities, such as promotions, approvals, assignments, distribution, and so on.

   Project/Lifecycle Change Activity

   Provides information about projects and lifecycle change activities, such as lifecycle definitions, change history, approvers, and so on.

   Security

   Provides information about authentication and security aspects of the projects, packages, and user accounts, such as password policies, failed and successful logins, and so on.

   Source Change Activity

   Provides information about items and other source change activities, such as snapshots, version changes and deletions, and so on.

   Reports that correspond to the type you selected are listed.
5. Click the report name for the type of information you want to see.

   The report appears in BusinessObjects InfoView.

More information:

Set BusinessObjects Report Preferences (see page 56)

Reports

Broker Dashboard Reports

Important! You can generate the broker dashboard reports, only if you have the administrator or CM administrator access.

You can configure and generate the broker dashboard reports for the followings items:

- Active and inactive projects
- Active and disabled users.
- Versions, items, and item paths count
- User count per user group
- Failed login audit summary

Configure Broker Dashboard

Configure the Broker Dashboard to generate various reports.

Follow these steps:

1. Open the Workbench Client and navigate to the broker to configure and generate reports.
2. Right-click the broker node and then select the Reports, Broker Dashboard.

   The broker level report displays a high-level information about the selected broker.
3. Select the following options to configure the report:

   Show Active/Inactive projects

   Displays the count of active and inactive projects from the total projects count in the selected broker in a pie diagrammatic representation.

   Show Active/Inactive users

   Displays the count of active and disabled users from the total users in the selected broker in a pie diagrammatic representation.

   Show Versions and Items/ItemPaths count

   Displays the logical version counts, physical version count, and size of compressed versions in a bar chart.

   Show User count per UserGroup

   Displays the count of users across each user group from the selected broker in a tabular format.

   Show Failed Login Audit Summary

   Displays the failed login attempts summary for a specified date and duration in a tabular format.
4. (Optional) Select one of the options:

   Select All

   Selects all the options from the Show the reports to be shown in dashboard section.

   Deselect All

   Clears all the selected options from the Show the reports to be shown in dashboard section.
5. Click OK.

   You have now configured the broker dashboard to generate reports.

Project Dashboard Reports

You can access the project dashboard report by right clicking any selected project from the Explorer view and by selecting the Reports, Project Dashboard. You can generate a set of project dashboard reports:

Configured Dashboard Reports

- Package Distribution By State Report
- Item/Versions Distribution by State Report
- User Modified Items Report
- Peer Review Report
- Origination Repositories Information Report
- Versions from Other Projects Report
- Version Change Activity Report

Configure Project Dashboard

Configure the project dashboard to generate various reports.

Follow these steps:

1. Open the Workbench Client and navigate to the project to configure and generate reports.
2. Right-click the project, select Reports, and then Project Dashboard.
3. Select the following options to configure the report:

   Show package distribution chart

   Displays the count of package distribution across the states in the selected project.

   Show item/Version distribution chart

   Displays the details about the total items, modified items, total versions, and modified versions in the selected project.

   Show user modified items chart

   Displays the total modified items and the total assigned packages for all the users in the selected project.

   Show peer review chart

   Displays the code review requests summary for the chosen project.

Show Origination Repositories Information

Provides the origination information about the baseline repository.

For Example: A sample report shows the following project origination information:

a. ProjectA configured RepA, RepB as the baseline.

b. ProjectA created snapshot view SS1 after ProjectA had more changes.

c. ProjectB configured RepA from the repository as the baseline, but RepB from SS1.

d. ProjectB created snapshot view SS2 after ProjectB had more changes.

e. ProjectC configured RepA and RepB from SS2 as the baseline.

Now if you report on ProjectC, the origination information about RepA and RepB appears.

In this example, the report shows ProjectC's RepA is derived from ProjectB's SS2 and RepA is originated from ProjectA's SS1.

Show Versions from Other Projects

Displays all versions which are merged from other projects to the current selected project.

Show Version change activity on graph

Displays the frequency of the changes for a specified state and duration.

Highlight and show logged-in user information

Displays information about the logged in users in the current session.

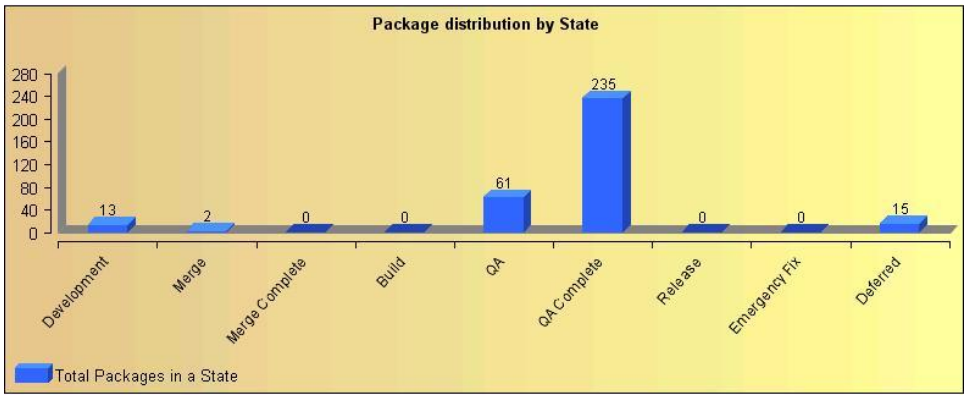
4. Click OK.

The configurations are saved to generate the report.

Package Distribution by State Report

The package distribution by state report shows the number of packages that exist in each state at the current time. This report helps you to estimate the packages that are required to be completed before being promoted to the next state. You can generate a bar chart for this report with the State names on X-axis and the Package count on the Y-axis.

The following graph illustrates a sample package distribution report. This report displays the total packages that are created in each state by the users that are logged in at a given time.

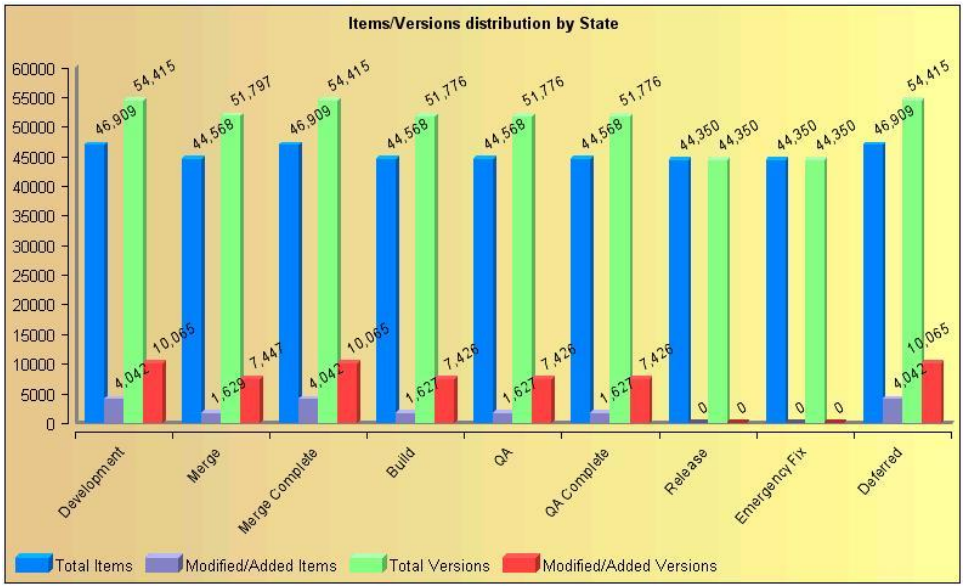


Item/Versions Distribution by State Report

The item/versions distribution by state chart displays the following information for each state in the selected project:

- A total number of items
- The modified items count
- A total number of versions
- The modified version count

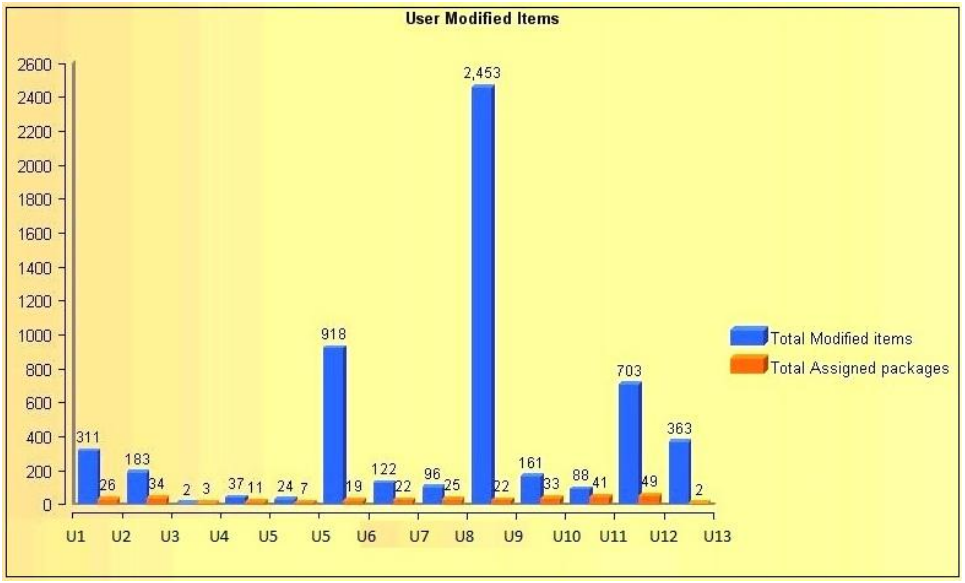
The following graph illustrates a sample item/versions distribution by state report displaying information in a selected project.



User Modified Items Report

The user modified items report displays the total number of modified items by each user in the project and the number of working packages.

The following graph illustrates a sample user modified items report that displays total modified items against total assigned packages.

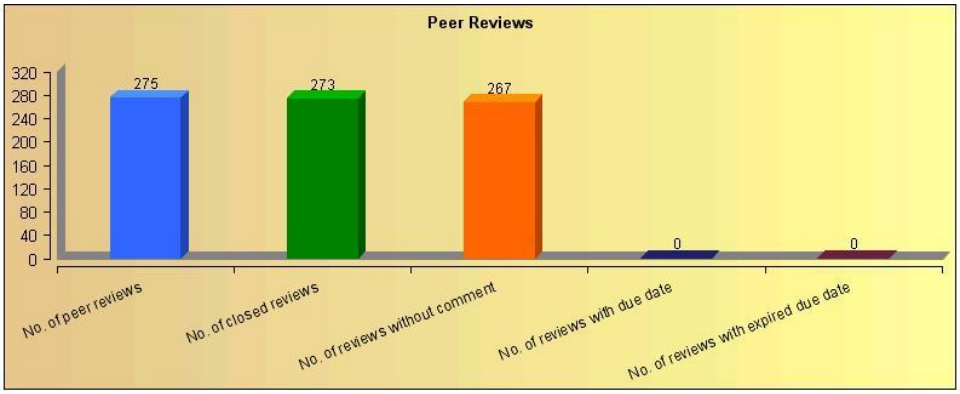


Peer Review Report

The peer review report displays the following peer reviews information in the form of a bar chart:

- A total number of the peer reviews
- A total number of closed reviews
- A total number of peer reviews without any comment
- A total number of peer reviews which are assigned with the due date
- A total number of peer reviews with expired due date

The following graph illustrates a sample peer review report.



Origination Repositories Report

The origination repositories report provides origination information about the baseline repository in a tabular form.

Example:

A sample report shows the following project origination information:

a. ProjectA configured RepA, RepB as the baseline.

b. ProjectA created snapshot view SS1 after ProjectA had more changes.

c. ProjectB configured RepA from the repository as the baseline, but RepB from SS1.

d. ProjectB created snapshot view SS2 after ProjectB had more changes.

e. ProjectC configured RepA and RepB from SS2 as the baseline.

Now if you report on ProjectC, the origination information about RepA and RepB appears.

In this example, the report shows ProjectC's RepA is derived from ProjectB's SS2 and RepA is originated from ProjectA's SS1.

The following table displays a sample origination repositories information.

Origin repository information

Repository	View Type	Parent Project	Originated Project
BOInstall	Snapshot	CA SCM Development Archive <Snapshot view : Harvest r12.NET SP 021209>	CA SCM Development Archive <Snapshot view : r12.NET SP 021209>
Build Automation	Snapshot	CA SCM Development Archive <Snapshot view : Harvest r12.NET SP 021209>	CA SCM Development Archive <Snapshot view : r12.NET SP 021209>
CACRYPT	Snapshot	CA SCM Development Archive <Snapshot view : Harvest r12.NET SP 021209>	CA Crypt Maintenance <Snapshot view : Harvest
CHSDK	Snapshot	CA SCM Development Archive <Snapshot view : Harvest r12.NET SP 021209>	CA SCM Development Archive <Snapshot view : r12.NET SP 021209>
CMDLN_TEST	Snapshot	CA SCM Development Archive <Snapshot view : Harvest r12.NET SP 021209>	CCC/HARVEST RINCON <Snapshot view : Harvest5_1_B16_PatchC>

Versions from Other Projects Report

The Versions from Other Projects report displays all versions which are merged from other projects to the current selected project.

The following table displays a sample version from other project report.

Versions from other project

Item	Target Version	Status	Target Package	Taget User	Source Project	Source Version	Source Package
\Eclipse Clients\plugins\MultiProduct\plugin.properties BIRT Report Viewer	1.2.1	N	Changes porting from FP2	goosr01	CA SCM r12 Fix Pack 2	1	Defect TBD -1- r12 FP2 - Enhar file synch
\Eclipse Clients\plugins\MultiProduct\plugin.properties	0.1.1	N	INTERNAL - r12.1 - Get chnages from archive project	sebge01	CA SCM Development Archive	32	Internal - r12 SP1 - adding View property pages
\Eclipse Clients\plugins\MultiProduct\srcIdePlugin\com\ca\harvest\core\SCMProjectSetCapability.java	0.1.1	N	INTERNAL - r12.1 - Get chnages from archive project	sebge01	CA SCM Development Archive	7	6907 - r12 SP1 - add support fo state in a project context
\Eclipse Clients\plugins\MultiProduct\srcIdePlugin\com\ca\harvest\core\SCMProjectSetCapability.java	1.1.1	N	Changes porting from FP2	goosr01	CA SCM r12 Fix Pack 2	2	Defect 1295 - r12 FP2 -Sharing similar name deleted earlier fror workspace-results in endless a
\Eclipse Clients\plugins\MultiProduct\srcIdePlugin\com\ca\harvest\core\SCMProjectSetSerializer.java	1.1.1	N	Changes porting from FP2	goosr01	CA SCM r12 Fix Pack 2	3	Defect 1295 - r12 FP2 -Sharing similar name deleted earlier fror workspace-results in endless a

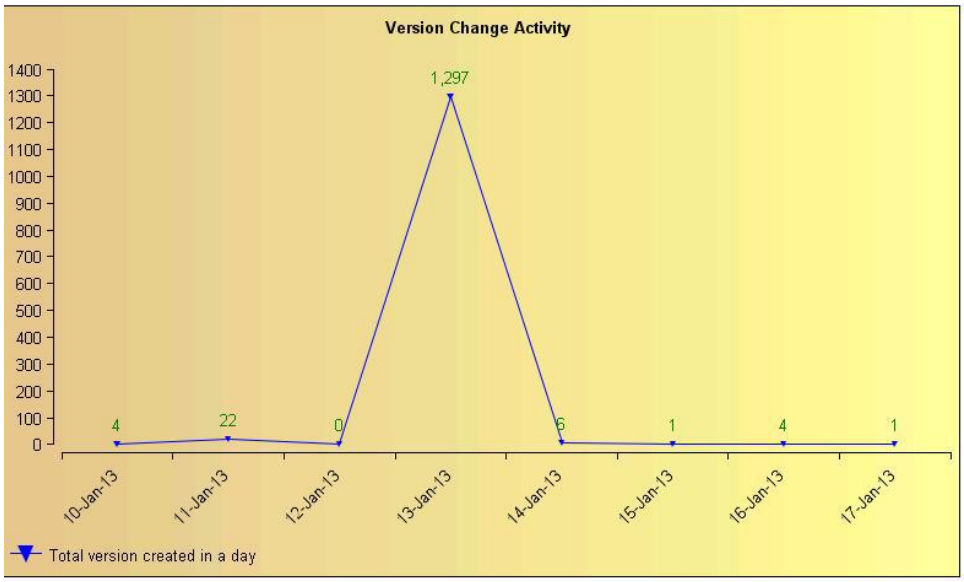
Version Change Activity Report

Version change activity report displays the graph on the frequency of the changes that are made to the SCM Repository over a period. This graph indicates time on X-axis and count of versions on the Y-axis.

Note: Depending on your access permissions, you can customize and generate the report that is based on the frequency of changes.

As a developer, you can generate the report that is based on the frequency of changes that you made. As a manager, you can generate the report that is based on the frequency of changes that the development team made.

The following graph illustrates a sample version change activity report that displays the total versions that are created at a given time.



Custom Reports

Custom reports are useful to display data for a sub group from your overall data.

Important!

- If you have the *administrator or CM administrator* access, you can peform the following tasks:
  - Create (see page 258) the custom dashboard reports for broker, project, state, and package levels.
  - Generate (see page 260) reports at all levels (broker, project, state and package).
- If you have the *use* access permissions at the project level, you can *only* generate *(see page 260)* the custom dashboard reports for the project, state, and package levels.

If you have the *administrator or CM administrator* access, you can create the customized SQL statements and can generate reports using the BIRT reports option. You can formulate the hsql queries for versions, packages, user modified versions, auditable events, or any other relevant data using the custom reports. You can create a custom report, edit, or delete the existing reports.

If you have the *use* access at the project level, you can *only* generate the custom dashboard reports (created by a user with the *administrator or CM administrator* access) for the following levels:

- Project
- State
- Package

Note: You can only run a previously created custom report but cannot edit or delete the existing reports. For more information about administering data, see the *CA Harvest SCM Administrator Guide.*

Create Custom Reports

You can create custom reports *only* if you have with the *administrator or CM administrator* access.

Follow these steps:

1. Open the Workbench Client and select the SCM broker.
2. Select the BIRT Reports toolbar button.

3. Select Custom Reports on the left pane in the Custom Dashboard Reports dialog.
4. Click Add.

   The New Custom Report dialog box opens.
5. Select a scope (Broker, Project, State, or Package objects) for which you want to create a custom report from the drop-down list.
6. Provide a title for your custom report in the Title box.
7. Select a Broker in which you want to save the custom report from the Save To drop-down list.
8. Select one of the following output formats for your custom report.

   Table

   Displays the query results in a tabular form.

   Bar Chart

   Displays the query results as a bar chart, where X-axis indicates the first column, and Y-axis indicates the number in the SQL query.

   Pie Chart

   Displays the query results as a pie chart, X-axis indicates the category, and Y-axis indicates the number that is used in the SQL query.
9. You can add custom SQL in two ways. You can either import custom SQL from an external file or type the custom SQL in the text box.
10. Use predefined keywords that are based on the selected Scope, while writing SQL.

    For example, in the Package scope, use ${PROJECT_ID}, ${STATE_ID}, ${WORKING_VIEW_ID}, ${PACKAGE_ID}, ${USER_ID}.

    These predefined keywords are available based on the selection of your scope.

    Broker

    ${USER_ID}

    Project

    ${PROJECT_ID}, ${USER_ID}

    State

    ${PROJECT_ID}, ${STATE_ID},${WORKING_VIEW_ID},${USER_ID}

    Package

    ${PROJECT_ID}, ${STATE_ID},${WORKING_VIEW_ID}, ${PACKAGE_ID}, ${USER_ID}

    Note: You can test the custom SQL before saving it as a custom report.

11. (Optional) Select the Test SQL.
12. Provide SCM Context information like the Broker name, Project name, state name, package name that is based on your selected Scope and click OK.

    A confirmation message displays the test result.

Note: If you anticipate huge records as a result of custom query execution, we recommend you to choose the tabular format.

Generate Custom Reports

If you have the *administrator or CM administrator* access, you can generate the custom reports for the following levels:

- Broker
- Project
- State
- Package

Follow these steps:

1. Open the Workbench and navigate to the broker, project, state, or package to generate reports.
2. Right-click the selected SCM item, select Reports, and then select one of the custom reports that are created from the menu.

   The custom report is generated for your analysis in the output format that you have chosen while creating that custom report.

If you have the *use* access at the project level, you can generate the custom reports for the following levels:

- Project
- State
- Package

Note: You cannot access the broker level custom dashboard reports.

Follow these steps:

1. Open the Workbench and navigate to the project, state, or package to generate reports.
2. Right-click the selected SCM item, select Reports, and then select one of the custom reports that are created from the menu.

   The custom report is generated for your analysis in the output format that the *administrator user* has chosen while creating that custom report.

BIRT Report Viewer

BIRT Report Viewer provides the Run Report and Export Report as toolbar options. You can navigate to any of the displayed pages. By default, the records displayed per page are 5000. For more information about the setting preferences, see the Set BIRT Preferences (see page 56) section.

You can export the Dashboard reports and Custom reports generated from any Harvest SCM level to the local file system. The export option supports the following output formats:

- PDF
- MS Word
- MS PowerPoint

Follow these steps:

1. Click the Export Report icon  , on a generated report from the BIRT report viewer toolbar.

   The Export Report Dialog opens.
2. Select one of the available export formats--PDF, MS Word, or MS PowerPoint from the Export Format drop-down list.
   - You can choose the range of pages to export the report.

     For Example, All pages, current page, or specific pages.
   - You can also choose the display settings depending on your requirements.
3. Click OK in the Export Report Dialog box.

   A dialog opens providing you options to open or save the report.
4. (Optional) Click Open to open the file in the selected format.
5. Click Save to save the file in the specified format.
6. Choose the destination from the local file system where you want to save the reports.

   Export process completes and you can view the saved report.

Run Report

Clicking the Run report icon re-executes the SQL query and refreshes the displayed report with latest modifications.

Run and Export Report

Clicking the Run and Export icon re-executes the SQL query to ensure that the latest data is fetched from database. The updated report gets exported to the selected format. However, the report does not refresh in the viewer.

This export fetches all the available records by overriding the No of rows to display in Report viewer preference under the BIRT Report preferences.

The export option supports the PDF, MS Word, and MS PowerPoint formats.

Follow these steps:

1. Click the Run and Export Report icon  , on a generated report on the upper right side of the report viewer.

   The Run and Export Report Dialog opens.
2. Select one of the available export formats--PDF, MS Word, or MS PowerPoint from the Export Format drop-down list.
3. Choose the destination from local file system where you want to save the reports.
4. Provide a name for the report and click Save.
5. Click OK.

   The export process is complete now and you can view the saved report.

Chapter 14: Team Collaboration for Code Review

This section contains the following topics:

Team Collaboration for Code Review

Team Collaboration for Code Review supports out-of-the-box peer reviews of code changes during a project development cycle. A developer who finishes code changes and unit tests can create a review request for other team members to review the code review.

The review request facilitates the collaboration between the requester and the reviewers in the team. The review request also facilitates the execution of the code review as follows:

- The creation of a review request for code changes is based on a change package.
- An email notification option for both reviewer and requester informs team members of relevant information during the code review cycle.

  Note: For more information about configuring the email notification option, see the *Administrator Guide*.

- Reviewers can record in-line comments or add attachments on versions in the change package.
- An assigned primary reviewer gives a final vote on the approval or rejection of code changes, and can close a review request.
- A package approval can be invoked when the primary reviewer closed the review request.
- A pre-linked UDP can be used to enforce the completion of code review request before the change package is promoted to the next state.

  Note: For more information about this integration option, see the *Administrator Guide*.

With this support, a code review is no longer an isolated individual manual operation. Instead, it organizes group efforts with direct and indirect communication among team members in a project to accomplish the review of code changes. Furthermore, all activities are recorded during the code review cycle.

How the Review Request Works

In general, you create one review request for a given package. In some cases, each individual who has a part of the code changes for the given package creates a review request, resulting in multiple review requests.

The CA Harvest SCM server maintains the status (Open or In Progress) of the Peer Review. When you create a Peer Review, its status is Open. During the life of the Peer Review, when reviewers make any Votes, Comments, or Attachments, the server changes the Peer Review status to In Progress. If all Votes, Comments, and Attachments are deleted from the Peer Review, its status is changed back to Open.

The steps for processing a review request are as follows:

1. As the review requester, you create a review request for the contents of a package. You invite one or more users in a project to participate the code review. You designate one user as the primary reviewer.

   Note: The primary reviewer determines the final result of the code review. The vote decided by other reviewers is for reference only.

   You can set an option to send email notifications to all the reviewers and requester. The emails inform participants of their review request assignments and of any updates to the review.

2. A reviewer can double-click the version associated with a review request to open the Review Comments Editor to view the changes in the selected version and enter comments. Reviewers use this editor to do the following:

   - View the changes in the new version and compare the changes to the parent version or another selected version.

- Add comments to any line of the version for review.
- Insert multiple comments at the same line by one or more reviewers.

  While these comments can express support and suggest improvement, typically the comments identify problems, issues, or concerns. The comments can support or identify why the reviewer did not approve the package changes.
- Add attachments to any line of a version for review.

All reviewers can vote Yes or No for a review request. By default, the status is shown as pending for each reviewer in the review request. The vote indicates that you approve or disapprove the changes for the entire review request. When an approve package process exists in the current working state, the primary reviewer can also approve the package during the closure of review request.

Note: The Approve Package option is enabled when the Vote is Yes from the primary reviewer. In addition, before you can use this option, an approve process must be defined. The primary reviewer must belong to the approval group with execute access granted to the approval process. For more information about defining the approve process, see the *Administrator Guide*.

More information:

Reviewers (see page 265)
Create a Review Request (see page 266)
Review and Add Comments (see page 271)
Vote on a Review Request (see page 274)

Reviewers

When you create a review request, you assign users to review the changes that belong to the package associated with the review. Each review must include one reviewer who is designated as the *primary reviewer*. You can assign other team members in a project to participate in the code review cycle. The primary reviewer is the participant who makes the final decision on the approval of code changes. The comments and voting from other reviewers provides reference information about the review request.

Note: You can change the reviewer assignments at any time.

More information:

Select Reviewers (see page 267)

Display the Peer Review View

From the Peer Review view on the Workbench, you can perform the following code review actions:

- List the review requests in a tree format.
- View and delete comments.
- View, download, and delete attachments.
- Mark version status.
- Delete review requests.
- View version status and view path

To display the Peer Review view, click Peer Review on the Workbench toolbar.

Note: If the Peer Review view does not appear, select View, Show View, Other. Expand the CA Harvest SCM folder, select Peer Review, and click OK.

The Peer Review view appears with the following tabs:

Requested Reviews

Lists all Open or In Progress review requests requested by you (the logged-in Workbench user).

Assigned Reviews

Lists all Open or In Progress review requests that you are assigned to you as a reviewer (including the primary reviewer).

Search Reviews

Lists review requests as determined by the search criteria.

The Requested Reviews and Assigned Reviews tabs list the projects in the broker and displays the same level of information for the review requests listed in each project.

More information:

How to Use the Peer Review View (see page 268)

Create a Review Request

You create a review request to initiate the code review process. When you create a request, you assign reviewers, and optionally change the primary reviewer, specify a due date, add notes, and so on.

Follow these steps:

1. Open the Explorer View and navigate to the package you want to be reviewed.
2. Right-click the package and select the Request for Peer Review option from the shortcut menu.

   The New Review Request dialog appears. The selected package name appears in the Package field.
3. Complete the dialog fields. The following fields require explanation:

   Assigned To

   Assigns users for the package review. You can add or remove reviewers, or change the automatically assigned primary reviewer.

   Include Due Date

   Enables the Due date drop-down list.

   Due date

   (Optional) Specifies a date for which the reviewers must complete the review. When you click the drop-down list arrow, a calendar feature opens that lets you specify a date and time. You save your changes by selecting the checkmark.

   General Notes

   (Optional) Provides information about the package changes that are being reviewed.

   Send an Email to Reviewer

   (Optional) Sends an email to each reviewer notifying them that they are assigned to a review.

   Note: The administrator must configure email for this option to execute. For information about configuring this option, see the Peer Review information in the *Administrator Guide*.
4. Click OK.

   CA Harvest SCM creates a review request and lists it in the Peer Review view.

Select Reviewers

You can add or remove any number of secondary reviewers to a review request, and assign or change the primary reviewer.

Follow these steps:

1. On the New Review Request dialog or the Requested Reviewer Editor, locate the Assigned To field.

2. Click the plus sign.

   The Reviewer Selection dialog appears.

3. Select one or more check boxes next to the user names of the reviewers you want to assign. Click OK.

   The reviewers are assigned to the review request. The Assigned To field lists their names. A primary reviewer is automatically designated.

4. Perform optional actions:

   - Change the primary reviewer by selecting a name in the list and clicking the star symbol.

     The star symbol appears on the name in the list to indicate the new primary reviewer.

   - Remove a reviewer by selecting a name in the list and clicking the minus sign.

     The user name no longer appears in the list.

   Click OK.

   Reviewers are selected for the review request.

How to Use the Peer Review View

You display (see page 266) the Peer Review view and use Requested Reviews or Assigned Reviews tabs to perform all your review request activities at various levels in the tree:

- You can perform the following actions at the review request object level:
  - List versions for the review request by expanding a review request node.
  - Update the review request properties by double-clicking a review request.

    The requested Review Editor appears.

    As a requester, you can update the review request properties such as Assigned To, Due Date, and General Notes.
  - Locate the review request package by right-clicking a review request and selecting Locate in Explorer View from the shortcut menu.

    The Explorer view appears and the corresponding review request package is highlighted.
  - Delete a review request by right-clicking a review request on the Requested Reviews tab and selecting Delete from the shortcut menu.

    Note: You must be the requester to delete a review request and the review request must have an Open status.

    The review request is deleted.

- You can perform the following actions at the version level:
  - List the comments and attachments by all reviewers and the requester of the version by expanding a version node.

    All comments and attachments are listed. Status icons indicate review progress.
  - Comment on the version or add an attachment by double-clicking a version.

    The Review Comments Editor appears, and you can comment on the version or add attachments.
  - Track your review progress by right-clicking one or more of your assigned versions on the Assigned Reviews tab, and select Mark As, *status _option* from the shortcut menu.

    A progress indicator appears on the version.
- You can perform the following actions at the comment level:
  - View commented text or attached file name by moving the mouseover the comment or attachment.

    The text of the comment or the attached file name appears.
  - Download or delete the attachment available under the version node.
  - Add comment or attachment by double-clicking the version or by double-clicking the comment or attachment made to the version.

    The comment or attachment appears in the Review Comments Editor. You can add comments and attachments to the version and also download the attachment made to the version.
  - Delete a comment or attachment by right-clicking a comment or attachment and selecting Delete from the shortcut menu.

    The comment or attachment is deleted.

Refresh the Peer Review View

You can refresh the Requested Reviews or Assigned Reviews information in the Peer Review view to view the latest review request changes.

Follow these steps:

1. In the Peer Review view, select the Requested Review or Assigned Reviews tab.
2. Select the broker that includes the review request you want to refresh.
3. Click the Refresh toolbar button.

   The Peer Review view is refreshed with the latest changes.

List and Update Review Requests

You can list and update the requests that are assigned to you for review.

Follow these steps:

1. On the Workbench, select the Peer Review view, and then the Assigned Reviews tab. Expand the broker node.

   A list of projects that are associated with the broker appears.

2. Expand the project node of interest.

   A list of your assigned review requests for the project appears.

3. Double-click the review request that you want to update.

   The Assigned Review Editor appears.

4. Perform any of the following actions:

   - View General Notes.
   - View reviewers including the primary reviewer.
   - View the requester.
   - View the Created Date and Due Date.
   - View the review request Status:

     Open

     Indicates that no comments or attachments has been made to at least one version of the review request and that no reviewer has voted either Yes or No for the review request.

     In Progress

     Indicates that the comment or attachment has been made to the version or that a reviewer has voted either Yes or No for the review request.

     Closed

     (Primary reviewer only) Indicates that the review request is closed and unavailable for commenting and for file uploading.

   - Vote (see page 274)on the review request changes.

- View the Review Progress pane to see Vote results and reviewer comments. By clicking a reviewer name, a list of version comments and attachments added by the respective reviewer appears in the right pane. You can mouseover a comment or attachment to view the comment text or attached file name. You can also double-click the comment or attachment to open the Review Comments Editor.
- Approve the corresponding package by selecting the Approve Package option.

  Note: The Approve Package option is enabled when the Vote is Yes from the primary reviewer. In addition, before you can use this option, an approve process must be defined. The primary reviewer must belong to the approval group with execute access granted to the approval process. For more information about defining the approve process, see the *Administrator Guide*.

5. Select File, Save.

   The review request is updated.

Review Comment Rules and Considerations

When you use the Review Comments Editor to review and add comments for versions in a review request, consider the following rules and behaviors:

- You can compare normal (N) or removed (D) versions.

  When you use a D-tagged version in the comparison, the version is empty, however, you can enter a comment at line 1. For example, you can comment that a team member should not remove the version.
- You cannot compare merged (M) or reserved (R) versions.
- The parent version of the version you select for review always appears in the Parent Version pane. Therefore, when another reviewer comments on a version that has the same parent version as your review version, you do not see their comments. You can view all comments for a version by expanding the version on the Assigned Review tab and double-clicking the comment you want to view.
- Comments or attachments made to a version are specific to that review request. They are not specific to the same package to which the review request was created.

Review and Add Comments

As a reviewer, you review and add comments on all the versions that your assigned review request includes. After you review the versions, you can vote on the review request.

Note: Version comparisons in the editor for adding in-line comments use the Eclipse-based built-in diff tool.

Follow these steps:

1. Click the Assigned Reviews tab of the Peer Review view.

   A list of your assigned review requests appears under the project node.

2. Expand the Review Request node and double-click the version you want to review.

   The Review Comments Editor appears with a comparison of the version under review and the parent version. Highlighted text indicates changed lines for you to review.

3. Add a comment for a changed line:

   a. Go to the changed line and double-click the leftmost ruler.

      The Comments at Line Number dialog appears.

   b. Add the comment text in the lower text area, and click the plus sign.

   c. (Optional) Add an attachment for a changed line by clicking the paper clip symbol.

      A browser opens that lets you locate and select an attachment.

4. Repeat Steps 2 and 3 for each changed line.

5. Close the Comments at Line Number dialog after you add all your comments, and click OK when prompted.

   All comments are listed under the version object.

6. (Optional) Mark your progress (see page 273).

7. (Optional) Reopen the review session by double-clicking a comment or attachment in the list.

   The Review Comments Editor appears.

More information:

Mark Version Status (see page 273)
Vote on a Review Request (see page 274)
Review Comment Rules and Considerations (see page 271)

Mark Version Status

You can optionally track your review progress. For example, you can mark whether you have started or completed reviewing a particular version in a review request. This bookkeeping feature helps you track your progress as you perform your reviews of the versions associated with a review request.

Follow these steps:

1. On the Workbench, select the Peer Review view, and then the Assigned Reviews tab.

   All Open or In Progress review requests that are assigned to you as a reviewer are listed.

2. Select and expand a review request object.

   The versions in the review request package are listed.

3. Right-click one or more of the versions that you want to mark, select Mark As from the shortcut menu, and select one of the following options:

   Pending

   Indicates that you have not worked on this version.

   In Progress

   Indicates that you have opened the Review Comments Editor at least once to review this version.

   Completed

   Indicates that you completed your review of this version.

   The review status icons appear on the versions you marked. When you mouseover a version, text that describes the status appears.

Vote on a Review Request

After you review all the versions associated with a review request, you can vote on the review. Your vote indicates whether you approve or disapprove the changes for the entire review request. All reviewers can vote Yes, No, or Pending on the review request. The vote of the primary reviewer for the review request indicates the approval or disapproval of the review request. Votes by other reviewers provide input for the primary reviewer. When you are a primary reviewer, you can perform additional review request actions after you vote.

Follow these steps:

1. On the Peer Review view, browse the Assigned Reviews tab to the review request that you want to vote on.
2. Double-click the review request.

   The Assigned Reviews Editor appears.
3. In the Your Vote field, click the option that corresponds to the vote you want:

   Yes

   Approves the review request.

   No

   Disapproves the review request.

   Pending

   Specifies that the review request vote is undecided.

   Select File, Save.

   Your vote is recorded for the review request and appears in the Review Progress table.
4. (Optional for primary reviewers *only*) Perform the following actions:
   - Close the review at any time by selecting Close and then Save.
   - Approve the corresponding package by selecting Approve Package after the review request is closed.

     Note: The Approve Package option is enabled when the Vote is Yes from the primary reviewer. In addition, before you can use this option, an approve process must be defined. The primary reviewer must belong to the approval group with execute access granted to the approval process. For more information about defining the approve process, see the *Administrator Guide*.

Close a Review Request

The primary reviewer can close a review request at any time to make the request unavailable for commenting or file uploading.

Follow these steps:

1. Open the Assigned Review Editor for the review you want to close. For example, you can navigate the Peer Review view and double-click the review request from the tree.

   The Assigned Review Editor appears.

2. Select the Closed option in the Status field.
3. (Optional) Select the Approve Package option.

   This option approves the package that corresponds to the review request.

   Note: The Approve Package option is enabled when the Vote is Yes from the primary reviewer. In addition, before you can use this option, an approve process must be defined. The primary reviewer must belong to the approval group with execute access granted to the approval process. For more information about defining the approve process, see the *Administrator Guide*.

4. Click Save.

   The review request is closed and, optionally, its corresponding package is approved.

Delete a Review Request Comment

The Peer Review view lets you delete review request comments and attachments. When multiple comments and attachments exist for a single line of the source code, these comments are considered a “comment thread” (similar to an email thread). You are allowed to delete any comment or attachment that you made.

Follow these steps:

1. Select the Peer Review view and then the Assigned Reviews tab on the Workbench.
2. Navigate the review requests to the node of the version which contains the comment or attachment that you want to delete.
3. Expand the node, right-click the comment or attachment that you want to delete, and select Delete from the shortcut menu.

   A confirmation dialog appears.

4. Click OK.

   The comment or attachment is deleted.

Find Review Requests

Search Reviews lets you browse the review requests of interest. After you locate requests, you can perform actions (see page 277) on them, such as reopening a closed request.

Search Reviews is the only Peer Review function through which any user can view the review requests of any status. Review requests with a Closed status are not visible in Requested Reviews or Assigned Reviews but are visible in Search Reviews.

Search Criteria

Specifies search criteria for review requests such as Broker, Project Name, Package Name, Created By, Assigned to, Review Status (Open, In Progress, Closed) filters. The Advanced options let you search review requests using criteria such as Include Due Date Between, Include Creation Date Between, Comment, and Notes.

The following wildcards let you specify patterns to use in your search. You can specify multiple values at the same time by separating the values with a semicolon (;).

asterisk (*) or percent sign (%)

Represents any sequence of zero or more characters. For example, the pattern of *~ or %~ matches any temporary files that end with ~. You can also use these wildcard characters as the first or last characters in a character string.

question mark (?)

Represents any single character.

Follow these steps:

1. On the Peer Review view, select the Search Reviews tab.
2. Select a broker from the Broker drop-down list.

   The search fields are enabled.

   The following fields require explanation:

   Review Status

   Tracks the actions performed on the review request and the comments, attachments, or votes added by the reviewers as follows:

   Open

   Indicates that no reviewers have started the review.

   In Progress

   Indicates that either the reviewers have started commenting on versions or attaching files to the versions, or have voted on the review request.

   Closed

   Indicates that the primary reviewer closed the review request.

3. Enter or select criteria, and click Find.

   Note: The search fields are not case-sensitive.

   Results are listed.

Use Search Reviews Results

Search Reviews results let you execute actions without exiting the results list.

To perform actions on the review requests in the results, do any of the following:

- List versions for the review request by expanding a review request node. Expanding a version node lists the comments and attachments made by the reviewers and the requester.
- View or download the comment or attachment made under the version node.
- Delete the comments and attachments that you made.
- Review a version by expanding a review request in the list and double-clicking the version you want to review.

  The Review Comments Editor appears. As a requester or reviewer, you can add comments and attachments to the version.
- View or download the comments or attachments made to the version and its parent version.
- View or update a review request by double-clicking the review request you want to view or update.

  The review editor appears with review details.
- Reopen a closed review request by right-clicking the review request and selecting Reopen from the shortcut menu.

  Note: Only the primary reviewer or the requester of the review request can reopen a review request.
- Locate the package associated with the review request by right-clicking the review request and selecting Locate in Explorer View from the shortcut menu.

  The view changes to the Explorer View and the review request package is highlighted in the tree.

Generate Peer Review Reports

The CA Harvest SCM Peer Review supports reports that provide information about pending reviews.

Follow these steps:

1. Select the Peer Review view, and select one of the tabs: Requested Reviews, Assigned Reviews, or Search Reviews.
2. Expand the broker node that includes the project you want.
3. Right-click the project for which you want to generate the report, and select one of the following reports from the shortcut menu:

   Get All pending reviews

   Lists all the review requests that have not been closed and are pending (by specified date and time).

   Get All pending review counts by user

   Shows the number of nonclosed review objects based on the primary reviewer and are pending (by specified date and time).

   The Date Selection Dialog appears.
4. Select a date/time and click OK.

   The report appears.

You can export the report date to either a text file or CSV file.

Follow these steps:

1. Select the “Get All Pending Reviews” or “Get All pending review counts by user” view.
2. Select the rows that you want to export.
3. Right-click and select the Save List As option from the shortcut menu.
4. Provide the output file path, select the format CSV or text, and click OK.

   The report is exported to an output file.

Chapter 15: CA Vision Integration

The CA Vision integration provides traceability of requirements, user stories, tasks, and issues from CA Agile Vision™ or CA Product Vision to CA Harvest SCM Package. This integration also facilitates logging work hours and generating reports based on objects, to trace changes at a granular level.

This section contains the following topics:

Synch Server

The Synch server is a unique invocation of an SCM HServer process that is driven on an interval basis by the SCM Broker. The broker starts the synch server, which performs both inbound and outbound CA Vision object synchronization. Inbound processing retrieves records from the CA Vision server that are relevent to the SCM Projects that are associated with CA Vision Product Releases. Outbound processing includes posting SCM Package activity and posting task worklog hours. After the synchronization is complete, the HServer process exits. When the next Synch server interval time occurs, the SCM Broker starts a new synch server process. In this way, resources are not wasted by a long-waiting process that would be idle between Synch server cycles.

Synch Server Behavior

The SCM Broker controls the Synch server at the configured interval. The first time the Synch server runs and finds there are no CA Vision objects loaded into the SCM Repository, it queries the CA Vision server for Product, Release, and User database data. The synch server also stores a flag in the SCM database indicating that the CA Vision integration is enabled. The SCM Workbench uses this flag to determine whether to make CA Vision integration operations available.

On subsequent Synch server intervals, the Synch server performs two main operations:

- Obtains new or updated records from the CA Vision server
- Applies any changes requested or automatically generated in the SCM Server, such as posting worklog hours or updating task status

The Synch server obtains only the CA Vision objects, other than the Product, Release and User records, for CA Vision Releases that are associated with SCM Projects. When a release is associated with an SCM Project, a special synchronization operation is performed. All Requirement, Sprint, User Story, and Task records belonging to that release are loaded from the CA Vision server into the SCM Repository. This process is referred to as an initial product load.

After releases are associated with SCM Projects, new or changed objects associated with those releases are queried during every Synch interval, and updated in the SCM Repository as required.

Updates to the CA Vision server are performed before the retrieval of new and changed objects. In this way, updates are reflected in the retrieved objects. For example, posting worklog hours causes the associated User Story record to be changed. This changed user story is retrieved during the same Synch cycle.

Package Associations

After the SCM Project is associated with a CA Vision Release, the CA Vision integration involves associating SCM Packages with CA Vision objects such as Requirements, User Stories, and/or Tasks. After a Package is associated with a CA Vision object, the following activities and operations are enabled:

1. Package Activity - When a package is associated with a requirement, user story, or task, the package activity similar to the SCM package history is posted to the CA Vision object. This activity is displayed in the CA Harvest SCM package display segment of the requirement or user Story. All package operations such as Created, Approved, Promoted, and Demoted are logged to the CA Vision object.
2. (Optional) Code Changes - When a package is associated with a requirement, user story, or task, the trunk item versions of that package are posted to that user story. When the package is promoted to the configured Development Completion state, the code changes are posted to the user story. If the package is demoted to a state previous to the Development Completion state, the code change records are removed from the user story. Similarly, when a switch package is executed, the code changes are removed from the user story.
3. Posting Worklog Hours - If a package is associated with a task or a user story, the task worklog hours are posted using SCM Workbench. If the package is associated with a user story, then the user selects the task from the Tasks list. This list consists of the tasks that belong to the selected user story.
4. (Optional) Automatic Task Status Update - If you enable this option for the SCM Project, an Implementation, Design, or Doc Task associated with the package is automatically updated to the Completed status when the package is promoted to the Development Completion state. If the package is later demoted out of this state, its status is returned to In Progress. Similarly, QA associated with an SCM Package is updated to the Completed status when the package is promoted to the QA Completion state.

Posting Hours to a Task

Before you post worklog hours to a task, verify that the package is associated with the task or the user story that the task belongs to.

To post worklog hours, right-click the package and select the Post Worklog Hours option from the menu list. The Post Worklog Hours dialog is displayed.

You can select only one task from the task list. Select the task you want to post the hours to and specify the work date and hours worked. Click OK to post the hours. The hours are not posted to the CA Vision server immediately but are queued in an SCM database table. The Synch server posts the hours to the CA Vision server at the next Synch interval.

This operation works only when the SCM User is mapped to the corresponding CA Vision User. See the Administration Guide for details on mapping SCM Users to CA Vision Users.

Special Package Activity and Code Change Operations

For packages that are associated with requirements or user stories, the package activity is stored in the CA Vision server in the respective objects.

Rename Package

Renaming a package is reflected in the CA Vision object. A renamed activity record is written to the CA Harvest SCM Package segment of the User Story or Requirement and all subsequent activities reference the new package name. If code changes are also stored in the User Stories, then SCM updates all the code change records for the associated Package to specify the new package name.

Switch Package

If code changes are stored in user stories and a Switch Package operation is applied to a package associated with a user story, the user story is updated to reflect the moved items. If both the source and target packages are associated with user stories, then both code change segments of the user stories are updated.

Approve Package

Approve package updates the package history in the CA Vision server. The concept of a frozen package created by approving a package does not apply to CA Vision object associations. SCM does not prevent a CA Vision object from being associated or disassociated from a package even when it has been approved.

CA Vision View

You can use the CA Vision View dialog to view CA Vision Objects and how they relate to SCM Objects. The dialog contains three tabs:

1. Requirements
2. Sprints
3. User Stories

To navigate to a tree, click the corresponding tab from the list at the bottom of the view.

The tool bar at the top right corner of the view contains four buttons:

1. Refresh
2. CA Integration Preferences
3. Find CA Vision Items
4. Reports

Requirements Tree

The Requirements tree displays the CA Vision Items associated with the SCM Project from a requirement perspective. The requirements of all CA Vision Releases are displayed under the Project node. You can expand each requirement node to view the child requirements, user stories, tasks, and the associated SCM Packages at any of the selected levels.

Sprints Tree

The Sprints tree provides an overview of the status of the selected sprint from an SCM package perspective. Configure the SCM Project Lifecycle so that it can determine whether a package can be considered in the Implementation, QA, or Completed category. Every package associated with a user story or task of a given sprint is grouped into one of these three categories. In this way, you can determine the status of the sprint from an SCM perspective.

User Stories Tree

The User Stories tree displays the CA Vision Items associated with the SCM Project from a user story perspective. The user stories of all CA Vision Releases are displayed under the Project Node. You can expand each requirement node to view the child requirements, user stories, tasks, and the associated SCM packages at any of the selected levels.

CA Vision Item Editor Display

The CA Vision Item editor display lets you view the Requirements, User Stories, and Tasks object types in detail. To display the details of an object, double-click the required object in any of the CA Vision View tree displays.

Filter Options

To control whether objects are displayed from all sprints or only the ongoing sprints, use the CA Vision Preferences Page. You can access this page from the task tool bar in the CA Vision View.

- To show objects from the current sprints, select the Ongoing Sprints button.
- To show objects from all sprints, select the All Sprints button.
- To show objects from all non-closed sprints, select the Ongoing and Future Sprints button.

These options apply to both the Sprints and User Stories tabs of the CA Vision View.

Find CA Vision Objects

To find a CA Vision item, select the Find CA Vision Items button at the top of the CA Vision View. You can also open the Find CA Vision Items dialog by selecting the Associate with CA Vision Items option from a Package node in the Explorer view.

Follow these steps:

This dialog provides a wide range of variables that you can use to find a CA Vision Item. These variables include Name or Title, Product, Release, Sprint, Users and Dates. For the text fields, you can enter wildcard values by using an “*”.

1. Enter the values of the variables you want to use for the search. For the text fields, enter wildcard values by using an “*”.

   Note: Some fields are disabled if they are not valid for the current set of parameters selected.

2. Click Find to see the list of items found.

You can also use the Show Associated Packages option to search the associated packages of a specific CA Vision item.

Reports

SCM provides two types of reports associated with CA Vision. The reports provide both graphical and list data associated with SCM Package and CA Vision Objects. The charts provide CA Vision Item distribution and SCM Package Lifecycle distribution. The list provides all CA Vision Items associated with SCM Packages for the selected SCM Project or CA Vision release.

To generate a report, click the Report icon from the CA Vision tool bar.The available report types are displayed.

- Select an available report type and click Generate.
- The project-based report includes all releases associated with the SCM Project.
- With the Release based report, you can select any release that is currently associated with one or more SCM Projects.
- The Mapped Users report displays the list of all SCM users mapped to the corresponding CA Vision user.
- SCM Life cycle settings display a list of all SCM Projects with their life cycle settings.
- The SCM Project Association report displays the mapping information of all SCM Projects with CA Vision products and releases.

Chapter 16: Using Other Interfaces

This section contains the following topics:

User Interfaces

In addition to the Workbench, user clients for CA Harvest SCM consist of the following interfaces:

- **Command-line utilities**—Lets you perform CA Harvest SCM processes from a command line.

  Note: For information about the command-line utilities, see the *Command Line Reference Guide*.

- **Web Interface**—Lets you perform CA Harvest SCM processes from a web browser.

  Note: For information about Web Interface, see the Web Interface help.

- **Windows Shell Extension**—Lets you perform CA Harvest SCM check-in and check-out functions by right-clicking files in Windows Explorer.

Windows Shell Extension

When you install the client on Windows, you can optionally install the CA Harvest SCM Windows Shell Extension. Windows Shell Extension lets you access the product's version control system using Windows Explorer menus. You can check in and check out files from Windows Explorer and execute these common tasks without CA Harvest SCM running on your client computer.

Use Windows Shell Extension

CA Harvest SCM Windows Shell Extension lets you check in and check out files from Windows Explorer.

Follow these steps:

1. Right-click any folder in your Windows Explorer, and select CA Harvest SCM from the shortcut menu.

   A shortcut menu appears and lists all of the CA Harvest SCM actions that you can perform.

   Note: Initially some of the menu options are disabled until you set the Set Default Context options.

2. Select Login.

   The CA Harvest SCM Windows Extension-Login dialog appears.

3. Enter your CA Harvest SCM credentials and the name of the computer where the CA Harvest SCM broker is running, and click OK.

   You are logged in to Windows Shell Extension and remain logged in until you execute the logout function. Closing the Windows Explorer does not log you out of Windows Shell Extension.

   The Message Log appears after you successfully log in. All Windows Shell Extension activity is recorded in the Message Log. The Message Log remains open during your Windows Shell Extension session and operates in the same way as the Log View.

   After you have successfully logged in and before you can perform any check-in or check-out functions, you need to set your default context.

4. Go to Windows Explorer and right-click the directory you want to use as your destination directory for checking out, and select CA Harvest SCM, Set Context.

   The Set Default Context appears.

5. Specify context settings in the dialog, and click OK.

   The context is set and remains the same until you modify it; you do not need to open the Set Default Context dialog again unless you want to change the settings.

6. You can check in and check out files in the following ways:

   - Right-click the files or directory in Windows Explorer that you want to check in or check out, and select CA Harvest SCM, check in or check out from the shortcut menu.
   - Use the Windows Explorer Search feature to locate files, right-click the files you want to check in or out from the results list, and select CA Harvest SCM, check in or check out from the shortcut menu.

   The check-in or check-out dialog appears and you can set options and execute the process.

7. To log out of Windows Shell Extension, right-click any file or directory in Windows Explorer, and select CA Harvest SCM, Logout from the shortcut menu.

   You are logged out of Windows Shell Extension.

Glossary

access control

Access control regulates which user groups are permitted to perform tasks (methods) with CA Harvest SCM objects. Each securable object in CA Harvest SCM has a list of methods associated with it; each user group can be assigned access to any of the method-object combinations.

administrator

An *administrator* is a CA Harvest SCM user belonging to the Administrator group. Administrators can perform all actions in CA Harvest SCM without access checks. An administrator typically initializes, activates, set ups, and assigns users to CA Harvest SCM. The administrator also defines life cycles, sets up access control, and performs repository recovery operations when necessary.

baselining

Baselining is the beginning of a life cycle from a point in an existing life cycle. Baselining lets groups begin development for a new release from the end of the previous release, or from a significant point in a continuing development cycle.

change

A *change* is a modification to an application that requires traceability and review.

concurrent update

Concurrent update is a mode of the check-out process that creates reserved versions of branches for the specified package. Branching can be used to support concurrent development, letting more than one package check out an item for update at the same time or more than one view to update an item.

context

A *context* is a collection of CA Harvest SCM settings, including, state, package, check-in process, check-out process, and view path. A context is used by CA Harvest SCM to determine necessary properties such as view path, and which change package to check in to.

get

A *get from repository* is a procedure used when performing an incoming synchronization. Performing a get from repository on a resource creates it in the local workspace if it does not exist, updates it if has changed, or deletes it if it was removed from the server. If the resource is a folder, the get from repository is performed on every resource under the folder as well.

item

An *item* is a component of a repository containing the actual information that makes up a software application. Items are the units of information modified by developers, analogous to operating system files. A versioned item represents many versions of a file. It has an initial version and it can have many versions on the trunk or on branches.

item path

An *item path* is a conventional way of referring to the equivalent of directories in the CA Harvest SCM repository, to distinguish them from directories in the external file system.

life cycle

A *life cycle* is the set of all the stages a change or release goes through from identification to completion. A life cycle is made up of states and processes.

package

A *package* is a CA Harvest SCM object that groups a set of changes together. A package is the fundamental unit of work in the CA Harvest SCM life cycle.

process

A *process* is a unit of work in CA Harvest SCM. Processes can be added to states or linked to other processes. Linked processes execute automatically; processes in states are optional. CA Harvest SCM provides a number of processes like concurrent merge, promote, and demote.

project

A *project* refers to the entire management framework in CA Harvest SCM that supports a particular development or maintenance activity. You can have many different projects, depending on the applications being controlled and the kind of development activity. A project includes information about how changes progress through the development life cycle, the activities that can occur, what data to access, and user responsibilities.

recursive

Recursive describes a function that starts at one level and continues down until no more levels exist. CA Harvest SCM provides a recursive option for check-in and check-out. When this option is selected, all files and directories below a specified directory can be included in a check-in operation, or all items and paths below a specified view path can be included in a check-out operation.

repository

A *repository* is a hierarchical collection of versioned items. It could include all the code for an application or only a module.

state

A *state* is part of the life cycle where a series of activities take place to correct a problem.

tag

A *tag* is an informative attribute associated with a version and displays in Repositories view and Lists view. Reserved, removed, and merged versions are tagged.

version

A *version* is a variation or edition of a CA Harvest SCM item. Versions are created when changes are made to an item, primarily through the check-out and check-in processes. A version represents the initial item and all changes associated with previous versions in its line of descent.

view path

A *view path* is a conventional way of referring to the equivalent of directories in the CA Harvest SCM repository, to distinguish them from directories in the external file system. When a view path is set, only resources below that path are visible. For this reason, a set view path is known as a view path anchor. All folders directly under the view path anchor are considered to be projects.

Index

A

B

C

D

E

F

Q

R

S

T

U

V

W

Z