ENTERPRISE ARCHITECT

User Guide Series

Change Management

Author: Sparx Systems

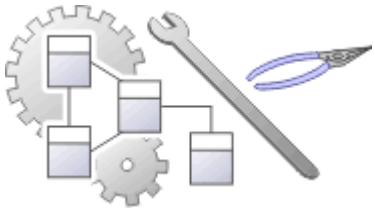
Date: 2021-09-02

Version: 15.2

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

Change Management



As users contribute new content and change existing content, a repository will become a valuable data store of organizational information assets. It is imperative that this asset is protected and changes are managed and controlled including being able to revert models back to previous versions or states. Enterprise Architect has a number of sophisticated tools to ensure the information is protected, including full integration with all the leading **Version Control** Systems, **Baselines** that are snapshots of your model that can be taken at important milestones, and **Auditing** that can track the finest changes to a model. A **Project Transfer** function helps you to make backups of models without involving information technology personnel. There are also **Model Validation** and **Project Integrity Checkers** to ensure the repository is maintained with a clean bill of health, and a powerful role based **Security System** to ensure users can collaborate easily and sections of the model can be locked to users or groups.

Facilities

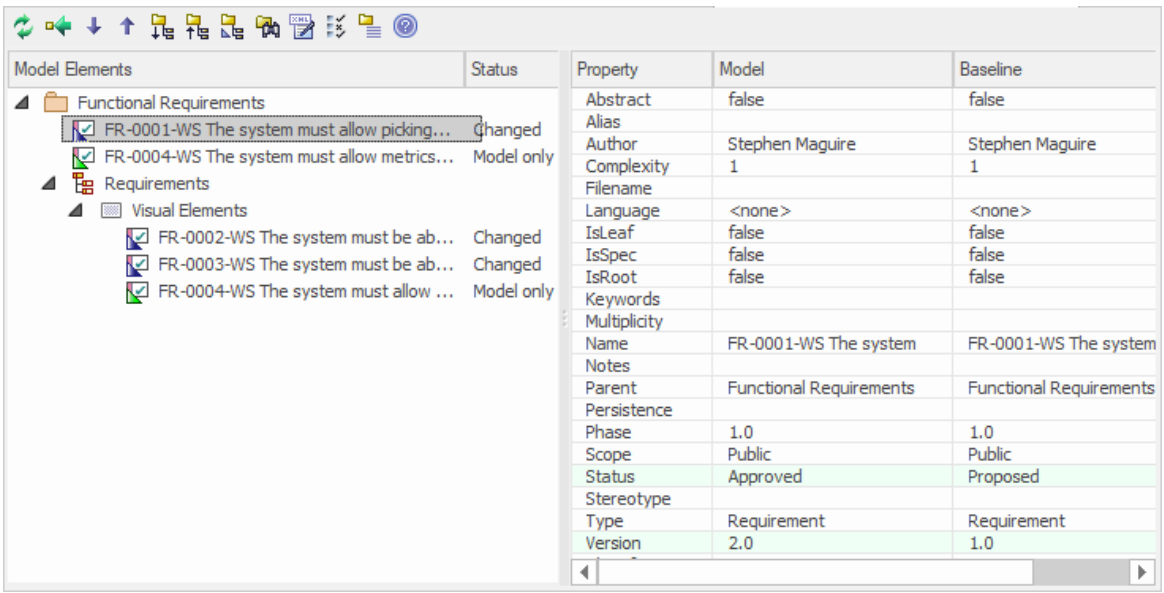
Facility	Detail
Baselines	Baseline create snapshots of model content, a changed model can then be compared to the snapshot and changes can be reverted to the baseline content if required. • **Baselines** are created at the package level providing snapshots of an entire package including elements, features and diagrams and optionally also the packages it contains. • As a model is altered the changed package can be compared with a baseline, including visual diagram changes and if required the current model can be reverted to the content stored in the baseline. • Baselines require no setup and provide an easy way for business and technical users to restore models to previous states at any level of granularity.
Version Control	**Version Control** allows model content to be versioned allowing check-out and check-in of packages to facilitate keeping track of different versions of parts of the model and allowing rollback to prior versions. • Coordinate sharing of Packages between users, with either read-only access or update access, ensuring that work on different areas of the model is coordinated and synchronous rather than conflicting • Allow check-out and check-in of packages to and from working storage including users working offline. • Save and retrieve a history of changes to Packages To use Version Control in Enterprise Architect, you require a third-party source-code control application such as: • Subversion • CVS • MS Team Foundation Server (TFS), or

	• Any other Version Control product that complies with the Microsoft Common Source Code Control standard
Auditing	**Auditing** records changes to the content in a repository and provides a powerful visualization view that can be used to drill down into changes from the package level down to changes to element features. • Auditing is typically used by librarians and administrators to investigate who made a change to the model and the date and time it was changed. • It can be used to locate problems in model development or for contractual reasons where third parties have been given access to manage model content and the source of a change needs to be identified. • Auditing can be enabled and disabled at any time and the audit logs can be saved and reload as needed.
Project Data Transfer	Enterprise Architect enables you to transfer project data between project data repositories either for: • Sections of the project (XMI and CSV) or • The whole project, row by row, table by table (in the Corporate, Unified and Ultimate Editions of Enterprise Architect)

Baselines

Snapshot parts of the repository at a given point in time, for later comparison and restoration if required

A Baseline creates a snapshot of a model Package and its contents at a point in time and, as the model changes, allows you to compare the current state and the snapshot; if required you can revert to the previous (baselined) state. **Baselines** are created at the Package level and can include child Packages; any number of Packages can be baselined, and a Package can be baselined any number of times, typically at important project milestones. Baselines are by default conveniently stored within the repository making them available to any model user who has the security privileges to work with them. Alternatively, a Baseline can be stored in a Cloud-based **Reusable Asset Service**. These options will be explored in a later topic.



This diagram illustrates the Baseline comparison tool, showing model and Baseline properties; Status and Version have changed.

Baselines are a type of insurance policy that ensures you can always revert to a previously captured state. In this topic you will learn how to create Baselines, and some of the common business and technical purposes of the technique. You will also learn to use the comparison too,l which allows you to identify what has changed in the model since a Baseline (snapshot) was created. You can reverse changes back to the Baseline state at any level of granularity. This will be indispensable when working with Project Teams using conventional or agile techniques, and where there are important governance or contractual requirements to manage change. You will learn how to manage Baselines, including exploring the options and benefits of storing them in the repository or alternatively in a Reusable Asset Service (RAS).

Brief Introduction

Baselines are easy to set up and require only a small amount of meta-data, such as the Version Number and some Notes that are pre-filled with the time and date the Baseline was created. You can add your own comments such as 'After Requirements Sign-Off' to indicate the significance of a milestone or reason for the Baseline.

The basic steps in working with Baselines are:

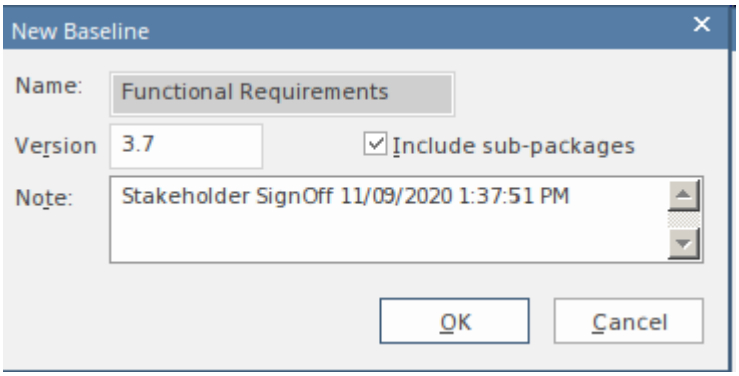
1. Create a Baseline - this will be for a selected Package and optionally its child Packages.
2. Compare the model to the selected Baseline - there can be multiple Baselines of the same Package.
3. Revert the model to one or more Baseline items - or choose to ignore the differences.

It is important to understand a fundamental difference between Baselines and **Version Control**. Baselines are user-initiated, so if a user has not created a Baseline for a given Package at a previous time it is not possible to compare the model to an earlier 'version'. However, once Version Control is set up for a Package, any changes that are made to the Package will be automatically versioned and will be available at a later time. These two facilities can be used in conjunction and it would be the responsibility of a Librarian or Administrator to determine how best to use them together.

Baselines are stored in the same XML format as is used for Version Control, but are stored in compressed format either within the project or in a **Reusable Asset Service** Registry. By default, Baselines are stored within the model; you can save a Baseline to an external XML file for storage or archive, or for distributing to other users working on models derived from a master project.

Creating Baseline

With a package selected in the Browser Window choose the Create Baseline option to create a new Baseline of the selected package. You will be required to complete some details for the baseline including the following:



New Baseline window showing details entered when creating a Baseline including the version number.

- Version - a user defined version number used to identify the baseline when running comparisons
- Include sub-packages - choose whether to baseline subordinate packages as well as the main package
- Notes - used to describe the baseline e.g. 'Requirements Sign-Off' pre-filled with the baseline time and date.

Comparing to the Model

The management of baselines makes use of the comparison tool and presents a tabular presentation of the differences. The comparison tool shows the differences between the elements, their properties and features, relationships or visual changes in diagrams at the current time and the time of the baseline or model export. A tree view presents the

comparison items and can be used to navigate through the changes which are presented on the panel on the right hand side.

Reverting to Baselines

Once the comparison (differencing) is complete, you can revert changes by merging information from the Baseline into the current project; it is not possible to go the other way.

You can:

- Merge information manually, change by change
- Merge information automatically by electing to merge in all changes in one batch procedure
- Revert completely to the original Baseline by importing the stored XMI directly
- Merge information and elements from a Baseline in a different project, making it possible to keep multiple versions of a single model in synch

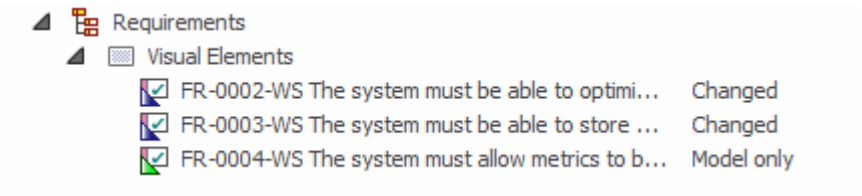
The merge options are available through the toolbar, context menus and the keyboard on the 'Compare Utility' view, which shows the results of a comparison.

Visual Differencing

Changes to a model might include:

- Adding or removing or changing elements and connectors on a diagram, or
- Changing the position of elements or the overall layout of a diagram

You might believe that a diagram has changed, and select to compare it with a baseline using a ribbon option. Alternatively, you might perform a baseline comparison on a Package or a model and select from the comparison output any diagrams that are flagged as changed.



Visual (diagram) differences for a number of existing and new requirement elements.

Creating Baselines

This topic details the basics of creating new **Baselines** of model Packages. You can create Baselines in the model (using the 'Baselines' dialog) or in a **Reusable Asset Service** Registry (using the 'Reusable Asset Service - Baseline' view). The appropriate screen displays by default according to whether the 'Store Package Baselines in a Reusable Asset Service Registry' checkbox is selected on the 'Baselines' page of the 'Manage Project Options' dialog.

Access

Select a Package in the **Browser window**, then open the appropriate screen for the storage system using one of the methods outlined here.

Ribbon	Design > Model > Manage > Manage **Baselines** : 'Baselines' dialog : New Baseline (or) Design > Model > Manage > Manage Baselines : '**Reusable Asset Service** - Baseline' view : New
Keyboard Shortcuts	**Ctrl+Alt+B** : '**Baselines**' dialog > New Baseline (or) Ctrl+Alt+B : '**Reusable Asset Service** - Baseline' view : New

Create a new baseline

Field	Action
Name	Display the Package name of the currently selected model branch.
Version	Type a unique version reference for this Baseline, which can consist of any alphanumeric characters. The 'Package Baselines' dialog sorts the **Baselines** according to the value of this field.
Include sub-Packages	Include the entire sub-Package hierarchy of this branch in the Baseline; this option defaults to selected. If you deselect the checkbox, only the immediate contents (XMI stubs) of the Package are included in the Baseline.
Note	Edit the default current time and date to any other value. The field is a single-line entry, for display on the 'Package Baselines' dialog (a one-line-per-entry list).
OK	Click to create a new Baseline and return to the 'Package Baselines' dialog.

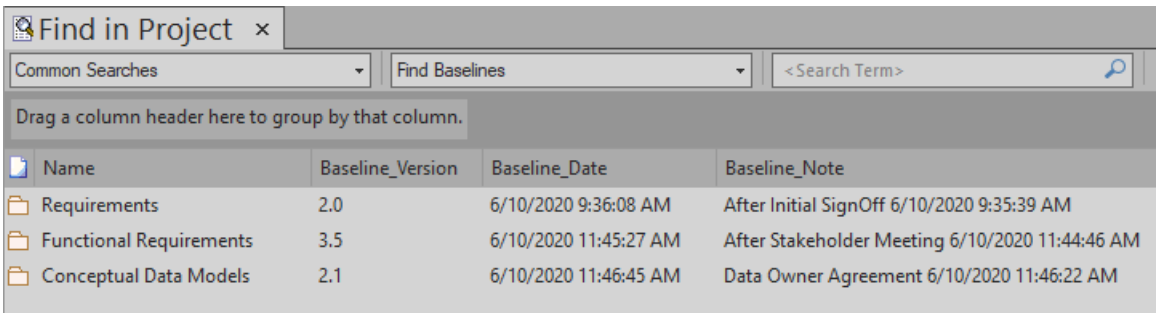
Compare a Model to Baselines

A baseline is created with an intention that it may be required as a backup or point of reference at some time in the future so that a model that has changed can be compared with the snapshot and differences determined. This may be needed just to identify that differences exist or it may be needed so that the entire package or individual changes can be rolled back to the state contained in the baseline (snapshot).

The first step in this process is to locate the required baseline and then to identify the differences. The baseline comparison tool can be used to visualize the changes that have occurred between the time of the baseline and the current time. Individual changes are not recorded as they are in the auditing facility but rather the accumulative results of the changes are presented.

Find Baselines

You can locate a model baselines by selecting a package in the Browser and then opening the Manage **Baselines** window. In the event that you have forgotten or are unsure of the location of one or more baselines you can use the **Model Search** utility to locate all packages that have baselines. The window conveniently displays the name of the Package and the Baseline: Version, Date and Notes. From this window you can locate the package in the **Browser window** and then launch the Manage Baselines window.



The Compare Utility (Diff)

Enterprise Architect has a comprehensive and powerful built-in Compare (diff) utility, which enables you to:

- Explore what has changed within a model over time
- Explore how previous versions of a model branch differ from what is currently in the model
- Perform a full model comparison by exporting all of Model A to XMI, then using 'Compare Model to File' from within the current model (Model B)

Comparing and checking model development at various points in the process is an important aspect of managing change and development, monitoring what is being modified and ensuring the development and design process is on track.

Using the Compare Utility you can Compare a model branch in Enterprise Architect with:

- A Baseline created using the Baseline functionality (Corporate, Unified and Ultimate Editions)
- A Baseline stored in a different model
- An XML 1.1 file created previously using the Enterprise Architect XML export facility (user selects file)
- The current version-controlled XMI 1.1 file as created when using **Version Control** in Enterprise Architect (file automatically selected)

Access

Select a Package in the **Browser window**, then open the '**Baselines**' dialog or '**Reusable Asset Service** - Baseline' view (depending on whether Baselines are stored in model or in a Registry) using one of the methods outlined here.

Ribbon	Publish > Model Exchange > **Package Control** > Compare Package to XMI Design > Model > Manage > Manage **Baselines** : Show Differences ('Baselines' dialog) Design > Model > Manage > Manage Baselines : Compare ('**Reusable Asset Service** - Baseline' view)
Keyboard Shortcuts	**Ctrl+Alt+B** : Show Differences ('**Baselines**' dialog) Ctrl+Alt+B : Compare ('**Reusable Asset Service** - Baseline' view)

Differencing With Baselines

As a Baseline contains all information about elements and connections for a Package at a point in time, it can be used within Enterprise Architect to track changes to model elements over time.

The Differencing engine first builds a representation of the current Package in memory, based on what is currently in the model.

It then compares this with the stored Baseline, highlighting changes, new elements, missing elements and elements that have been moved to other Packages.

It is possible to filter the resultant output to display only one particular kind of change: for example, additions to the model.

If a Baseline has been created to ignore child Package content, a comparison between that Baseline and the model does not include any child Package content in the model.

See the example provided in the *Example Comparison* Help topic.

Notes

- This utility is available in the Professional, Corporate, Unified and Ultimate Editions of Enterprise Architect
- You cannot compare the current model with an XMI 2.1 file; the utility can only compare with an XMI 1.1 file

Compare Options

You use the 'Baseline Compare Options' dialog to refine the output of the Compare utility when it compares the current model with a Baseline.

To display the dialog, either:

- Click on the **Options button** on the 'Package Baselines' dialog, or
- Click on the 'Compare Options' icon in the 'Compare Utility' view toolbar

If the 'Compare Utility' view shows the results of a Baseline comparison, when you click on the **OK button** the display refreshes to refine the information according to the options you have selected.

Options

Option	Action
Always Expand to Differences	Always display the list of elements fully expanded to show changes. If you deselect the checkbox, when the 'Compare Utility' view is first opened it lists the Package contents to element level, and you expand each element as required to show the changed items. For large branches of the model, it is better to leave the checkbox unselected.
Show Elements that are	List elements that: • Have been changed since the Baseline was created • Are in the Baseline only (that is, have been deleted from the model since the Baseline was created) • Are in the model only (that is, have been created since the Baseline was created) • Have not changed since the Baseline was created (you might generally leave this checkbox unselected)
Suppress these Changes	Exclude: • Changes to diagrams • Changes to the 'Date Modified' field for an item • Changes to the 'Date Created' field for an item • Child items of a deleted item • Changes to advanced properties (defaults to selected)
Baseline Diagram Compare Option	Select the checkbox to always open the first parent Package for which there is a Baseline, when you select the diagram for comparison from the **Browser window**.

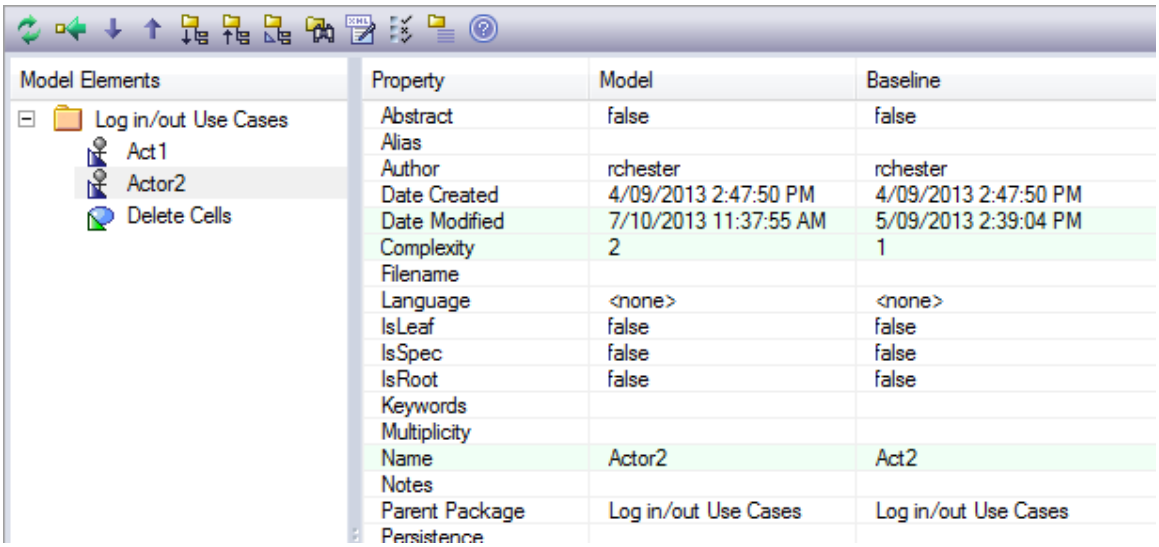
Notes

- Package Baseline facilities are available in the Corporate, Unified and Ultimate Editions of Enterprise Architect

Example Comparison

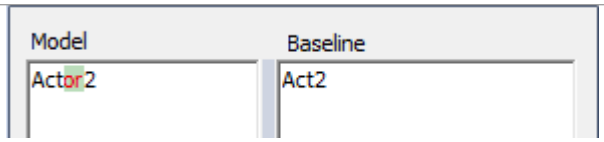
This diagram shows the result of a comparison between a Package (Log in/out Use Cases) in the current project and that Package in a Baseline captured at an earlier date.

The results of the comparison are displayed on the 'Comparison Utility' view.



Review Changes

Aspect	Description
Interpretation	A hierarchy of model elements is displayed in the left-hand pane. From the triangle-based icons and the highlighted lines on the report, it is clearly visible which items in the hierarchy have, since the Baseline was captured, been: • Changed • Deleted from the model (in the Baseline only) • Added to the model (in the Model only) or • Switched to different Packages (changes in the Parent Package property) If you click on an item in the left hand pane, the right-hand pane displays a table of properties showing the values of those properties in the current model and in the Baseline. For each property where there is a difference between the model and the Baseline, the row is highlighted. The 'Compare Utility' view enables you to perform operations (such as merging or rolling back changes) on the reported information, using the toolbar, context menu and keyboard.
Increase Level of Detail	The right panel of the 'Compare Utility' view might, for some fields, display only part of the value. It might also not be immediately obvious what a change is. In either case, you can double-click on the property to display full details and to highlight the exact differences; this example shows the highlighted changes to the 'Name' property.

Notes

- The Compare utility is available in the Professional, Corporate, Unified and Ultimate Editions of Enterprise Architect

Compare Utility View Options

The 'Comparison Utility' view enables you to perform operations on the reported information, using the toolbar, context menu, 'Merge' dialog and certain keyboard keys.

- The toolbar is at the top of the left-hand panel; the icons operate either on the comparison as a whole or on the currently-selected item in the left hand panel of the 'Comparison Utility' view
- Each item in the hierarchy has a context menu, which you display by right-clicking on the item; the options displayed depend on the level of the item in the hierarchy
- The 'Merge' dialog enables you to specify which changes to roll back in the model from the baseline
- You can use a selection of keyboard keys to move up and down the hierarchy, or to roll back changes

Toolbar Options

Option	Action
Refresh	Re-run the comparison to refresh the current display.
Merge To Model	Merge the values of the currently-selected item in the Baseline back into the model.
Next Change	Highlight the next changed item (this skips Moved items).
Previous Change	Highlight the previously-changed item.
Expand All	Fully expand the selected item.
Collapse All	Collapse the changed items in the selected item.
Expand To Changed Items	Expand the selected item to show changed items only (in the event that you have selected to also show unchanged items in the comparison).
Find in Project Browser	Highlight the item in the **Browser window**.
Log To XML	Log the changes to an XML file. A browser displays, on which you specify the file name and location.
Compare Options	Display the 'Baseline Compare Options' dialog.
Manage Package Baselines	Re-display the 'Package Baselines' dialog or '**Reusable Asset Service** - Baseline' view, as appropriate.
Help	Display the Help topic *Package Baselines*.

Context Menu Options

Option	Action
Merge from Baseline Add from Baseline	Restore the item in the model to the Baseline state, or restore a deleted item from the Baseline.
Delete from Model	Remove a recently-created item from the model.
Merge From Baseline (with Options)	(For the root node of the hierarchy on the 'Compare Utility' view.) Display the 'Merge' dialog, which you can use to specify options for rolling back the whole model branch to the Baseline state.
Refresh	(Object-level items.) Re-run the comparison to refresh the current display.
Find in Project Browser	Locate and highlight the item in the **Browser window**.
Open Baseline Diagram Compare	(For a diagram listed in the comparison.) Display the **Baseline Diagram Compare window**, showing differences in diagram content and layout.
Expand All	Fully expand the selected item.
Expand To Changed Items	Expand the selected item to show changed items only.
Collapse All	Collapse the changed items in the selected item.
Log To XML	Log the changes to an XML file. A browser displays, on which you specify the file name and location.
Baseline Compare Options	Display the 'Baseline Compare Options' dialog.

Merge Dialog Options

Option	Action
Changed	Restore all changed items in the model branch to the Baseline state.
In Baseline Only	Restore all deleted items to the model branch from the Baseline.
In Model Only	Remove all recently-created items from the model branch.
Moved	Restore all moved items to their original locations, as identified in the Baseline.
Full Restore from XMI	Completely restore the model branch to the version held in the Baseline XMI 1.1 file, (using the 'XMI Import' function). (This option automatically selects all the other options)

Keyboard Keys

- Ctrl+ ↓ - expand and highlight the next changed item
- Ctrl+ ↑ - expand and highlight the previous changed item
- Ctrl+ ← - undo the changes for a selected item (roll back to the Baseline values)

Visual Diagram Changes

The **Baseline Diagram Compare** feature is a quick and easy way to visually compare a current diagram with an earlier version from a saved Baseline, and highlight any elements in the diagram that have been added, deleted, resized or moved.

You can then review these changes and optionally roll back individual changes if needed to their previous state from the Baseline.

The changes are identified on the 'Baseline Diagram Compare' dialog and on the diagram itself.

Access

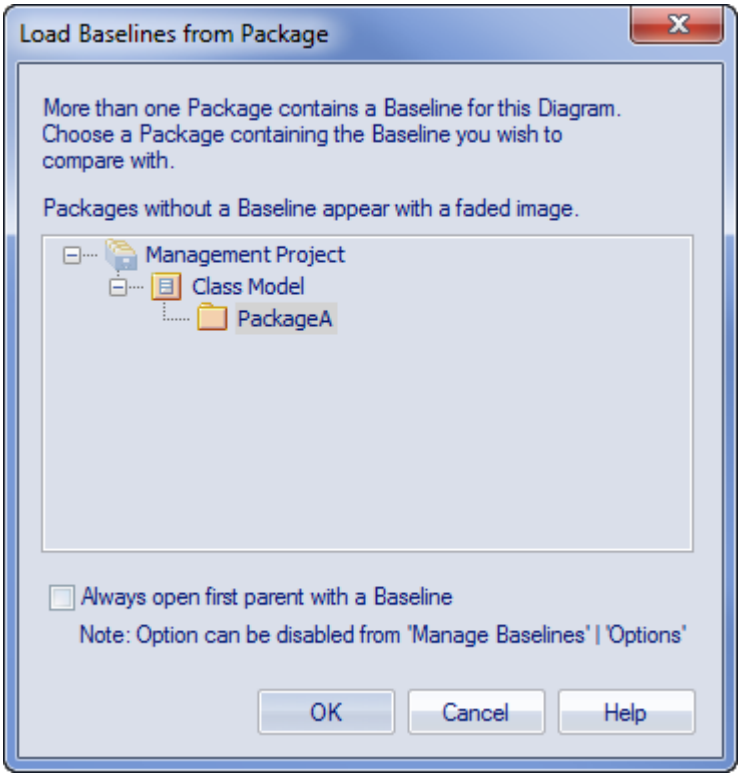
Select the diagram to be checked, then apply one of these access methods:

Ribbon	Layout > Diagram > Options > Compare to Baseline (on an open diagram) Design > Diagram > Manage > Compare to Baseline (on an open diagram)
Other	1. Perform a comparison of a Package and Baseline (see *The Compare Utility (Diff)* Help topic). 2. Locate the diagram in the results on the 'Baseline Comparison' view. 3. Right-click on the diagram name and select the 'Compare to Baseline' option to display the '**Baseline Diagram Compare**' dialog and to open the diagram (refer to the *Results* section).

Multiple Owner Packages

When you create a Baseline, it can be for a Package that contains one or more levels of child Package, and you might create **Baselines** for the Package(s) at each level. If the diagram you are checking is at a lower level in the hierarchy, there might therefore be a number of Baselines that contain information on the diagram, perhaps taken at different times and capturing different changes to the diagram.

When you open the diagram before performing the check, and select one of the direct ribbon access paths, if the diagram is referenced in multiple Baselines you might be required to select the Package from which to use a Baseline, on the 'Load Baselines from Package' dialog.



The 'Load Baselines from Package' dialog provides the facility to compare the diagram with one of a broader range of Baselines than just those from the diagram's immediate parent.

This dialog displays if you have NOT selected the 'Always open first parent with a Baseline' checkbox on either:

- The 'Load Baselines from Package' dialog itself or
- The 'Baseline Compare Options' dialog

Selecting or deselecting the option in one location also selects or deselects it in the other.

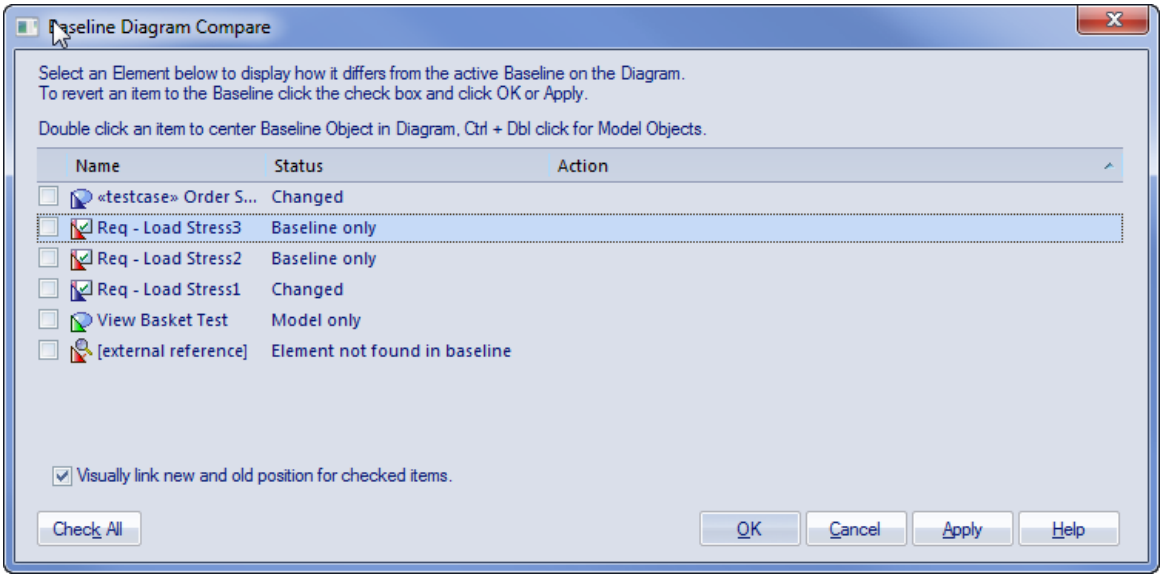
Click on the required Package, and click on the **OK button**. In this case, **or** if the dialog did not display at all (the checkboxes were selected), the 'Baselines dialog displays.

Processing

Click on the required Baseline and on the **Show Differences button**. The '**Baseline Diagram Compare**' dialog displays. Refer to the *Results* section and the *Options* table.

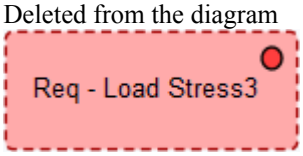
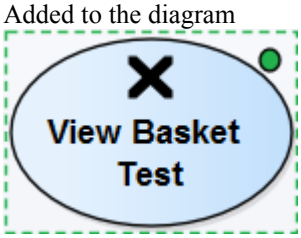
Results

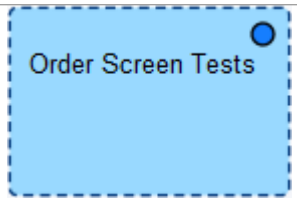
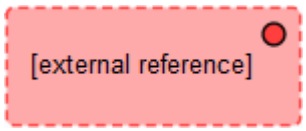
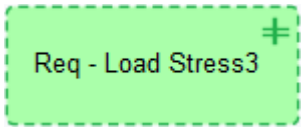
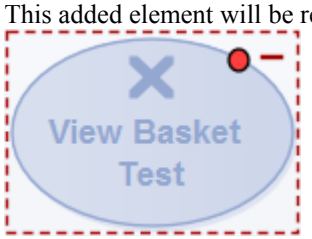
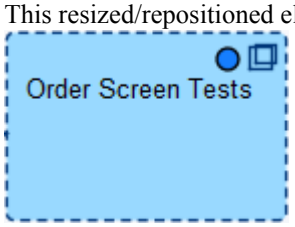
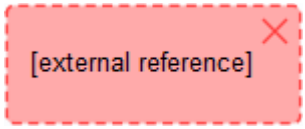
The '**Baseline Diagram Compare**' dialog shows the elements that have been changed on the diagram, and what kind of change was made (examine the 'Status' field).



As you select elements on the dialog, images are shown on the diagram itself to indicate where the changed element was and what kind of change it underwent.

Options

Option	Detail
Select (click on) name of element	The 'Status' column indicates whether the element has been: • Moved or re-sized (Changed) • Deleted from the diagram (Baseline only) • Added to the diagram since the Baseline was captured (Model only), or • Deleted from its parent external Package, and there is no record in the current Baseline (because the Baseline is only for the current diagram's parent Package) This diagram-linked element has been deleted from the model. The element might be found in a different baseline either in a parent Package baseline or a different Package baseline outside of the current Package. If the external referenced element is restored to the model, the visual comparison will be able to resolve the missing diagram object in the current baseline. When an item is selected, the corresponding element on the diagram is highlighted as shown: • Deleted from the diagram  • Added to the diagram  • Resized or moved to a new position

	 • A deleted external element on the diagram  The highlighted element on the diagram is marked with a colored dot, as shown, to indicate that it is in focus.
Position the diagram to show the selected element	To scroll the diagram so that you can see the original (Baseline) position of an element, double-click on the item in the list. To scroll the diagram so that you can see the current (model) position of the element, press and hold **Ctrl** while you double-click on the item.
Leave the changes in the item as they are	Ensure that the checkbox against the item is not selected. Click on the **OK button**.
Roll the changes back to the Baseline position	Click on the checkbox against each required item (or click on the **Check All button** to select every item). The 'Action' column displays the action required to roll each element's relationship to the diagram back to the Baseline relationship, and on the diagram the selected elements are represented as shown: • This deleted element will be restored  • This added element will be removed  • This resized/repositioned element will be put back in its original position  • This element from another Package, deleted from the diagram, cannot be restored from this Baseline  The comparison automatically shows a blue direction arrow for each reposition or resize that has been checked. For a heavily edited diagram this might be confusing.

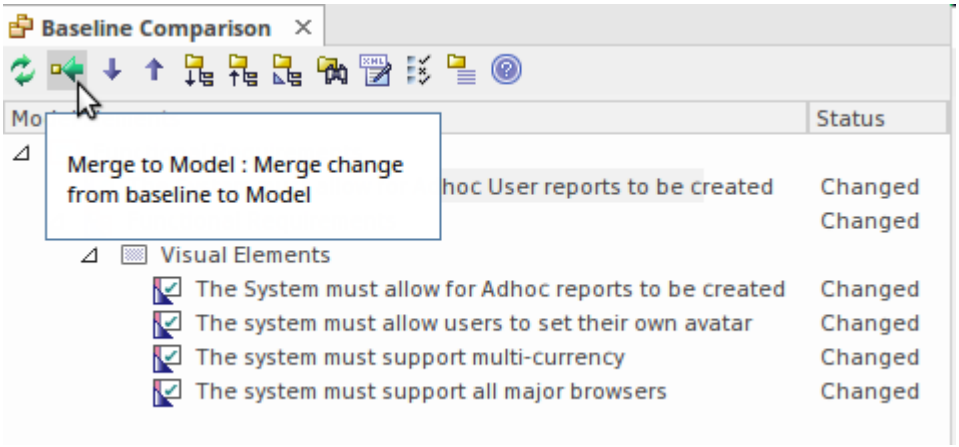
	However, you can hide the arrow for all elements except the one currently in focus; to do this: • Deselect the 'Visually link new and old position for checked items' checkbox To roll back the changes for all items for which a checkbox is selected: • Click on the **Apply button**

Notes

- Diagram Baseline facilities are available in the Corporate, Unified and Ultimate Editions of Enterprise Architect

Revert Model to a Baseline

You can return any part of a model to the content stored in the Baseline. This is a simple process of selecting the item in the Baseline comparison tool and using the 'Merge to Model' toolbar icon.



The 'Merge to Model' option for a selected Requirement element.

The 'Merge to Model' option is useful when you or your team identify unwanted changes that have been made to the model since the Baseline was created. There could be items such as elements, Features, Tagged Values, connectors and diagram objects that have been:

- Deleted
- Modified, or
- Created

You will be warned that this option is not reversible, and if changes are merged from the Baseline to the model the changes to the model will be lost. Note that it is not possible to merge model changes to a Baseline, as the Baseline is a snapshot and cannot be altered.

Manage Baselines

Baselines are snapshots of a repository package taken at a point in time. These snapshots are stored in a format called xmi which is an xml format and these files must be stored so that can be recalled and used for comparisons at a future time. There are two primary places to store baselines:

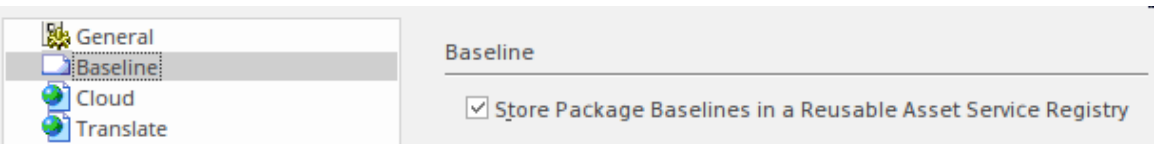
- In the current repository - this is the default location
- In a **Reusable Asset Service** - a reusable asset service needs to be nominated as the location

Baselines are by default stored in the repository, so when for example you create a Baseline of a Requirements Package at a given milestone, the 'snapshot' would by default be stored in the same model as the Requirements. This is expedient and sufficient for small, isolated project teams, but for larger teams and endeavors Enterprise Architect provides some powerful mechanisms to store the Baselines as reusable and universally accessible snapshots available to any project or organizational team, anywhere in the world. This is particularly useful in this age of innovation and agility, where geographically independent teams need to share information and reuse existing assets at Internet speed. The facility uses the Reusable Asset Server (RAS), which is Cloud-based (including on the home premises) and provides sophisticated mechanisms to manage these valuable corporate assets.

A librarian or administrator would typically make the decision about which of these options is most appropriated for a given repository and team. Factors such as whether the baselines need to be made available to a distributed audience and whether dependency analysis is required would be considerations in this decision. The management of the baselines differs depending on the option selected.

Specify Reusable Assert Server for Baselines

To



Access

Keyboard Shortcuts	
Ribbon	Configure > Model > Options > Baseline

Manage Baselines in Model

When you manage baselines in the model all of the Baseline features are available through the Manage **Baselines** window. As discussed in a prior topic, this is the default way to work with baselines and the files themselves are stored in compressed format inside the repository. This ensures these baseline snapshots are available to all model users who have the requisite permissions to manage the baselines where security has been enabled.

Storing the baselines in the repository is much more immediate and convenient and is suitable for small teams and situations where baselines are used in a cohesive team environment. If these baselines are important assets that need to be shared or managed by wider communities of users using the Reusable Asset Server may be used to provide access to distributed users.

Access

Ribbon	Design > Model > Manage > Manage **Baselines**
Keyboard Shortcuts	**Ctrl+Alt+B**

Baseline Management

Create, select and process **Baselines** using the 'Package Baselines' dialog.

Option	Action
Current Baselines For Package: <Name>	Review the **Baselines** for the current model branch, listed by version reference with the highest alphabetical/numerical value at the top. If an entry is longer than the display area, a horizontal scroll bar displays at the bottom of the panel; use this to scroll to the text that is not shown.
Show Differences	Run the Compare utility on the selected Baseline and the current model branch or diagram, to display the differences between the two.
Restore to Baseline	Completely restore the model branch from the selected Baseline.
New Baseline	Create a new Baseline.
Delete Selected	Delete the selected Baseline.
Load Other Baselines	Display a drop-down menu that enables you to load **Baselines** from another model, in either a project file or a DBMS repository. • For project files, a browser displays; locate the required project file • For DBMS repositories, the Windows 'Data Link Properties' dialog displays; select the data provider and click on the **OK button** to display the 'Select Data Source' dialog, from which you select the required project In either case, the *Connected To:* message at the bottom of the 'Package Baselines' dialog changes to the name of the alternative model. To return the dialog to the original project, select the third option on the drop down list: 'Load From Selected Package'.

Import File	Import an XML 1.1 file from the file system as a new Baseline for this current model branch.
Export File	Export the selected Baseline to an XML file.
Compare Model to File	Compare the selected model branch with an XML 1.1 file in the file management system; a browser displays, which you use to locate the file.
Options	Set filters to make the comparison more specific.

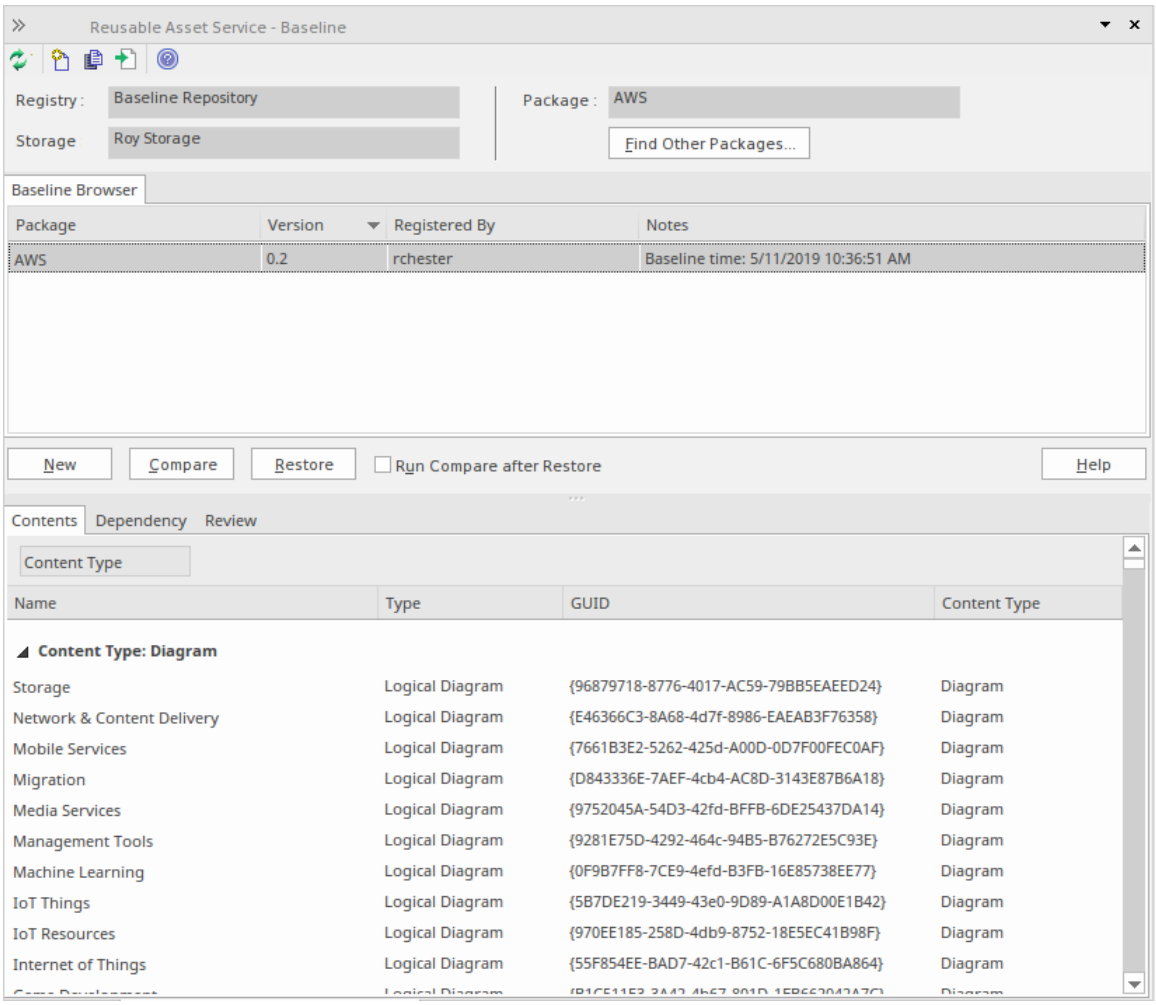
Notes

- Package Baseline facilities are available in the Corporate, Unified and Ultimate Editions of Enterprise Architect

Manage Baselines in Reusable Asset Service

When you manage baselines through the Reusable Asset Server (RAS) all of the Baseline features are available but in addition there are a range of other features only available when using RAS baselines. As discussed in a prior topic the baselines are available to a much wider audience, including other teams and departments, customers, standards authorities, partners, consultants, testers and more, all protected by robust industry standard security. **Baselines** stored in the RAS are more transparent and searchable, making it easier to understand and work with the content they contain.

Enterprise Architect provides a range of facilities for working with and managing Baselines stored in a **Reusable Asset Service** Registry, using the 'Reusable Asset Service - Baseline' view. When you open this view, the 'Baseline Browser' tab immediately displays a list of the Baselines available for the Package that is currently selected in the **Browser window**. You can review and use these Baselines using the context menu and buttons on the tab.



You can review the contents of the selected Baseline using the tabs in the lower half of the 'Reusable Asset Service - Baseline' view, underneath the 'Baseline Browser' tab.

If you have left the 'Baseline Browser' open for a while, there is a possibility that the contents of the Storage in Registry might have changed. Click on the  icon in the 'Reusable Asset Service - Baseline' toolbar to refresh the Browser with the latest contents of the Storage.

Note: To store Baselines in a Reusable Asset Service Registry, select the 'Baseline' page in the 'Manage Project Options' dialog and configure the Cloud connection to the Registry, then select or create the Storage in which to store all the Baselines for the Packages in the model.

Access

Ribbon	Design > Model > Manage > Manage **Baselines**
Keyboard Shortcuts	**Ctrl+Alt+B**

Baseline Management

Create, select and process **Baselines** using the '**Reusable Asset Service** - Baseline' view.

Option	Action
Registry	Displays the Registry name, as specified on the 'Baseline' page of the 'Manage Project Options' dialog.
Storage	Displays the Storage name, as defined on the 'Baseline' page of the 'Manage Project Options' dialog.
Package	Displays the name of the Package currently selected in the **Browser window**.
Find Other Packages	Click on this button to open the 'Reusable Asset Services' view, displaying the Storage and its full contents; that is, all Packages with **Baselines**, and any other Packages held in the Storage.
Baseline Browser	Lists the **Baselines**, if any, for the current Package. The Baselines are listed by version - with the highest alphabetical or numerical value at the top. When you select a Baseline in the Browser, details of the Baseline are displayed in the tabs beneath the 'Baseline Browser' tab. **Note**: You can right-click on the column headings and select the 'Toggle **Filter** Bar' option to show or hide the Filter Bar on the display. If you show the Filter Bar, type in the appropriate characters to list only those Baselines that have that string of characters in the value in the corresponding column.
New	Click on this button to create a new Baseline. **Note**: You can also click on the  icon in the '**Reusable Asset Service** - Baseline' toolbar to create a new Baseline.
Compare	Click on this button to run the Compare utility on the selected Baseline and the current Package, to display any differences between the two. The results of the comparison are displayed on the 'Comparison Utility' view. **Note**: You can also run the Compare utility by: • Clicking on the  icon in the '**Reusable Asset Service** - Baseline' toolbar or • Right-clicking on the Baseline name and selecting the 'Compare' context menu option
Restore	Click on this button to completely restore the Package in the model from the selected Baseline. **Note**:

	You can also restore the Package from the selected Baseline by: • Clicking on the  icon in the '**Reusable Asset Service** - Baseline' toolbar • Right-clicking on the Baseline name and selecting the 'Restore' context menu option
Run Compare after Restore	Check this option to automatically run the Compare utility once the Package is completely restored from the selected Baseline.

Notes

- The Package Baseline facilities are available in the Corporate, Unified and Ultimate Editions of Enterprise Architect
- **Reusable Asset Service** Registry details must be configured in the 'Baseline' page of the 'Manage Project Options' dialog
- If a 'Read-Only Access' password is entered for the Storage in the 'Baseline' page of the 'Manage Project Options' dialog, you will not be able to create new **Baselines** in the 'Reusable Asset Service - Baseline' view as the 'New' button will be disabled
- If security is enabled, you must have 'Baselines - Restore model' permission to restore a Package from the selected Baseline

Baseline Contents

In the **'Reusable Asset Service** - Baseline' view, when you select a Baseline in the 'Baseline Browser' tab, the three tabs in the lower half of the view are updated with information on the Baseline. The 'Contents' tab lists the diagrams and elements (including any child Packages) held in the selected Baseline, listing the two types of object separately. You can organize the information within a column into alphabetical or reverse-alphabetical order for ease of reference, and use the **Filter bar** to filter the display to show only items with values containing specific characters or digits.

Access

Open the '**Reusable Asset Service** - Baseline' view using one of the methods outlined here.

Select a Baseline in the 'Baseline Browser' tab, then click on the 'Contents' tab to display the contents of that Baseline.

Ribbon	Design > Model > Manage > Manage **Baselines**
Keyboard Shortcuts	**Ctrl+Alt+B**

Review Baseline Contents

Option	Detail
Content Type	Click on this button to toggle between listing the Baseline diagrams first and listing the Baseline elements first.
Toggle Filter Bar	Right-click on the column headings and select this option to show or hide the **Filter** Bar on the display.
(filter bar fields)	Type in the appropriate characters to list only elements and diagrams that have that string of characters in the values in the corresponding column.
Content Type: Diagram Content Type: Element	These are the headers for the two types of object listed in this tab. Click on the appropriate white or black arrow head to hide or show the list of diagrams or elements under the heading.
Name	Displays the name of the element or diagram available in the Baseline.
Type	Displays the type of the element or diagram, such as Use Case or Use Case diagram.
GUID	Displays the Global Unique Identifier of the element or diagram.
Content Type	Displays the object type of the item - Element or Diagram.
Find in Project Browser	Right-click on an element or diagram line and select this option to see if the element or diagram also exists in your model and, if it does, to highlight it in the **Browser window**.
	Right-click on a diagram name and select this option to display the diagram within

View Diagram	a labeled frame, as an image. Alternatively, double-click on the diagram name.

Baseline Dependencies

When you select a Baseline on the 'Baseline Browser' tab, the three tabs in the lower half of the '**Reusable Asset Service** - Baseline' view are updated with information from the Baseline. A Package might contain elements and diagrams that have relationships with objects in other Packages. When you generate a Baseline for that Package, the name and Global Unique Identifier (GUID) of each Package containing these 'external' objects will be stored along with the Baseline and displayed in the 'Dependency' tab.

Note that Package A depends on Package B if any of these constructs (or their Tagged Values) in Package A reference elements in Package B:

- Elements
- Attributes
- Operations
- Operation Parameters
- Diagrams
- Connectors

Access

Open the '**Reusable Asset Service** - Baseline' view using one of the methods outlined here.

Select a Baseline on the 'Baseline Browser' tab, then click on the 'Dependency' tab to display a list of dependencies for that Baseline.

Ribbon	Design > Model > Manage > Manage **Baselines**
Keyboard Shortcuts	**Ctrl+Alt+B**

Check Baseline Dependencies

Option	Detail
Package	Displays the name of the related Package.
GUID	Displays the Global Unique Identifier of the related Package.
Find in Registry	Right-click on the 'Package' line and select this option to see if the Package exists in the Registry. Selecting this option will : • Open the '**Reusable Asset Service**' view • Connect to the Registry and load the Storage that is currently selected in the 'Reusable Asset Service - Baseline' view • Open the 'Search Registry' dialog • Search for the selected Package using its GUID and display the results of the search, if any
Find in Project Browser	Right-click on the 'Package' line and select this option to see if the Package also

	exists in your model and, if it does, to highlight it in the **Browser window**.
Copy Package Name to Clipboard	Right-click on the 'Package' line and select this option to copy the Package name to the clipboard.
Copy Package GUID to Clipboard	Right-click on the 'Package' line and select this option to copy the Package GUID to the clipboard.

Notes

- The Name and Global Unique Identifier (GUID) of a Package containing 'external' objects will not be stored with the **Baselines** if the option 'Check Package dependency when creating Baseline' is disabled on the 'Baseline' page of the 'Manage Project Options' dialog

Add Review Comments

When you select a Baseline on the 'Baseline Browser' tab, the three tabs in the lower half of the '**Reusable Asset Service** - Baseline' view are updated with information from the Baseline. For any Baseline, you can add individual review comments on any aspect of the Baseline. These comments provide a permanent record on the Baseline - once they have been saved they cannot be edited or deleted. Each comment is attributed to the ID of the user who entered it, and is date stamped.

Access

Open the '**Reusable Asset Service** - Baseline' view using one of the methods outlined here.

Select a Baseline on the 'Baseline Browser' tab, then click on the 'Review' tab to display the review comments of that Baseline.

Ribbon	Design > Model > Manage > Manage **Baselines**
Keyboard Shortcuts	**Ctrl+Alt+B**

Actions

Action	Detail
Review Existing Comments	If there appears to be a longer comment than can be shown in the 'Review' tab, click on it. The full text displays in the 'Comments' field in the lower panel.
Create Comments	Click on the **New button** and start typing your comment on the Package in the 'Comments' field. You can delete and edit text as you type. When you have finished writing your comment, click on the **Save button**. Your comment, preceded by your user ID and the current date, is displayed in the 'Review' panel. Once you have saved your comment, you cannot edit or delete it.

More Information

Edition Information

- Package Baseline facilities are available in the Corporate, Unified and Ultimate Editions of Enterprise Architect
- The Compare utility (used to compare an exported Package with a model Package) is available in the Professional Edition of Enterprise Architect, as well as in the Corporate, Unified and Ultimate Editions
- The Enterprise Architect Corporate, Unified and Ultimate Editions provide another facility, **Auditing**, which you can switch on to perform continuous monitoring of changes across the project; you can dovetail your use of each facility to meet the range of your change management requirements

Considerations

Baselines are based on the Global Unique Identifier (GUID) or unique ID of a particular Package:

- Enterprise Architect checks for that ID as the root element within the XML document being used as a Baseline
- When you export a Package to XML, the Package you export is the root element; likewise when you create a Baseline, the current Package is the root Package of the XML Baseline
- When you save information in a **Version Control** system, the current Version Controlled Package is again the root Package of the document
- It is not useful to create a Baseline by importing an XMI Package file created by Version Controlling a Package that itself contains Version Controlled child Packages; that type of XMI Package file contains stubs for the child Packages, not full information on the child Packages and elements
- If a Package under Version Control forms part of a Baseline, and that Package is checked in to the model, you cannot merge the original data from the Baseline into that Package

XML files must be in the same format used by the Baseline engine - currently the UML 1.3 XMI 1.1 format (plus Enterprise Architect extensions), which contains all the information necessary to reconstruct a UML model, even a UML 2.x model.

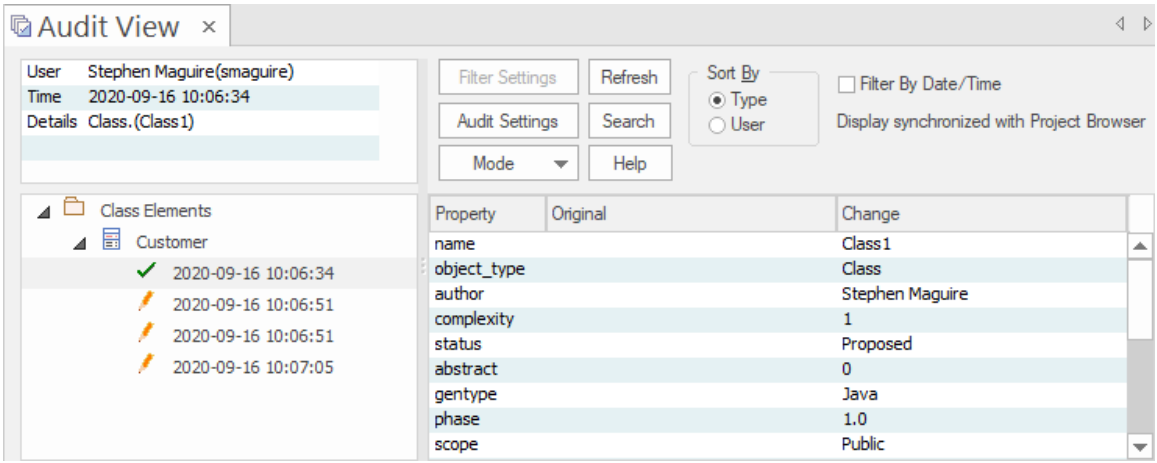
Notes

- If a Package under **Version Control** forms part of a Baseline, and that Package is checked in to the model, you cannot merge the original data from the Baseline into that Package
- You can also obtain a snapshot of selected items in the model, using the **Model Views** facility; this facility enables you to automatically generate the snapshot at intervals and, if there are changes in the items collected by the defined search, to trigger a notification to you of such changes, which enables you to monitor workflow and other events of concern to you
- If security is enabled you must have '**Baselines** - Manage' permission to create, import and delete Baselines, and 'Baselines - Restore' permission to merge data from a Baseline; security permissions are not required to select an existing Baseline and perform a comparison with the model

Auditing

Enable a Transcript of Model Changes Over Time

Auditing records model changes including when the change occurred and who made it. It records changes to Packages, Elements, Features, Connectors and Diagrams, detailing the creation, modification and deletion of these items. A repository administrator or librarian typically enables auditing during selected project phases or continuously during the lifetime of the repository. An Enterprise Architect repository contains important organization information and ensuring that changes to the models are consistent with an organization's intention and governance principles is critical to the success of this important information asset.

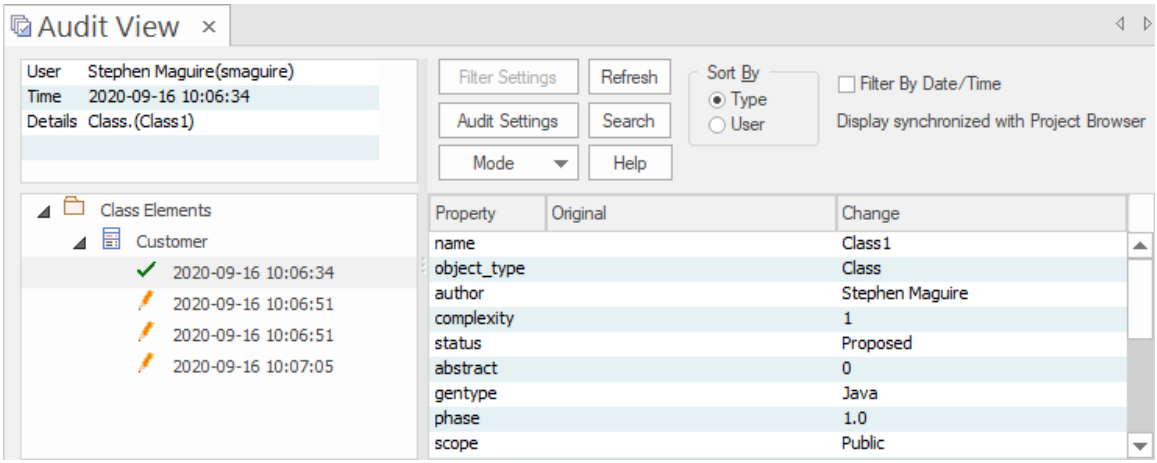


Audit View window showing the creation and change of a Customer Class and its properties.

Audits are a powerful model management mechanism ensuring that changes to models are well understood and governed. In this topic You will learn how to enable auditing, and using the Settings window define the level of detail to record in the audit, the elements to be audited and how batch imports are handled. You can simply enable auditing and it will silently record the changes the users make to the model. You will also learn how to manage audit logs to ensure there are minimal impacts to performance as logs are stored. Auditing can be used for a variety purposes beyond governance for example arresting sub-standard modeling practices by identifying the user and providing them with training and guidance of how to model in a given context.

Brief Introduction

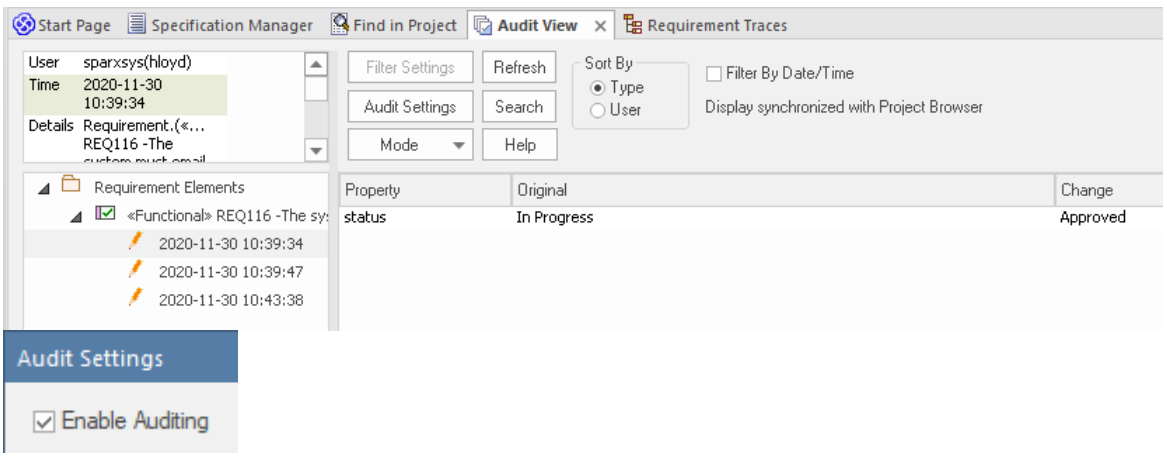
The **Auditing** facility in Enterprise Architect allows changes to the repository to be recorded to an Audit log. This powerful governance tool can be profoundly useful in finding out who changed a model and when it was changed. It is also useful to see the complete list of changes made by a given user or the list of changes made to a given element. The **Audit View** is a dashboard that allows you to display the before and after values of item properties and to drill down from Package level through elements, features and connectors to diagram objects.



Audit View showing the creation of, and three subsequent changes to, the Customer Class.

The basic steps to record Audit logs of the changes to a repository are:

1. Enable Auditing - this initiates the process of changes being written to the Audit log.
2. Configure Audit Settings - set the Audit Level and the Audit Options that define which elements will be audited, and set options for importing and reverse engineering.
3. Allow models to be changed - users perform their normal modeling, creating, modifying, and deleting elements, features and diagrams.
4. View the Audit log - to visualize the changes that have been made since the last log clearance.
5. Save and Clear logs - logs can be cleared to increase performance that degrades as the logs fill up.
6. Disable Auditing - optionally Auditing can be disabled if no longer required, and re-enabled when required.



Audit View showing changes made to a Requirement's Status property, from **Validate** to Implemented.

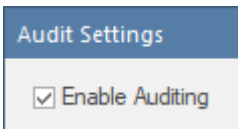
It is important to point out a fundamental difference between Auditing and **Baselines** and **Version Control**. While all

three of these facilities are concerned with changes to a repository over time, with Auditing it is not possible to revert the model to a previous state, which is possible with Version Control and Baselines. In contrast with a Baseline Compare, which only displays the original value and the current values of an item property, the Auditing View can display all the incremental changes over time, including when they were made and who made them.

Setting up Auditing is a simple process, and once set up it will begin recording information about what is changed in the repository based on the settings that you have specified. The next few sections provide the basic steps so you and your teams can get started with Auditing and controlling changes to your repository.

Enabling Auditing

You can, as a Librarian or Administrator, enable **Auditing**, which initiates the process of model changes being written to the Audit log. You might choose to only enable Auditing for a period of time such as when a new modeler joins the team, or when contractors have access to the model, or in the final stages of a sprint or iteration.

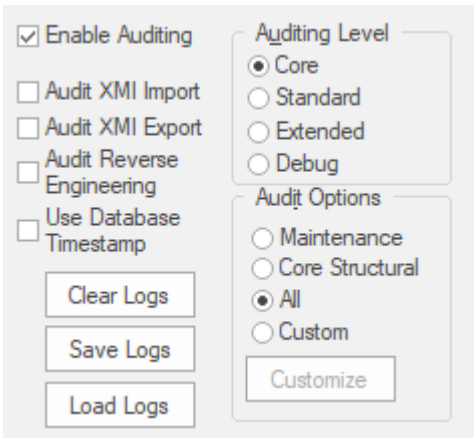


Audit Settings window, allowing Auditing to be enabled or disabled.

Auditing can be disabled at any time and enabled again at a later time. The logs, if not cleared, will simply be added to once Auditing is re-enabled.

Configuring Settings

The configuration of the Audit facility is an important step in setting up the tool, to ensure that the changes that you want to be able to visualize will be recorded in the Audit log and will be available when the time comes to view them. There are three main parts to the configuration:

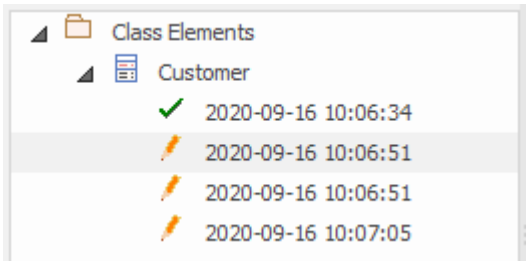


Audit Settings window, showing Audit Level, Option, Batch import and Log settings.

- Import and Reverse Engineering options - allow you to specify whether these bulk items should be recorded in the Audit log
- **Auditing** Level - You can specify the extent of the information recorded
- Auditing Options - You can specify which elements will be audited

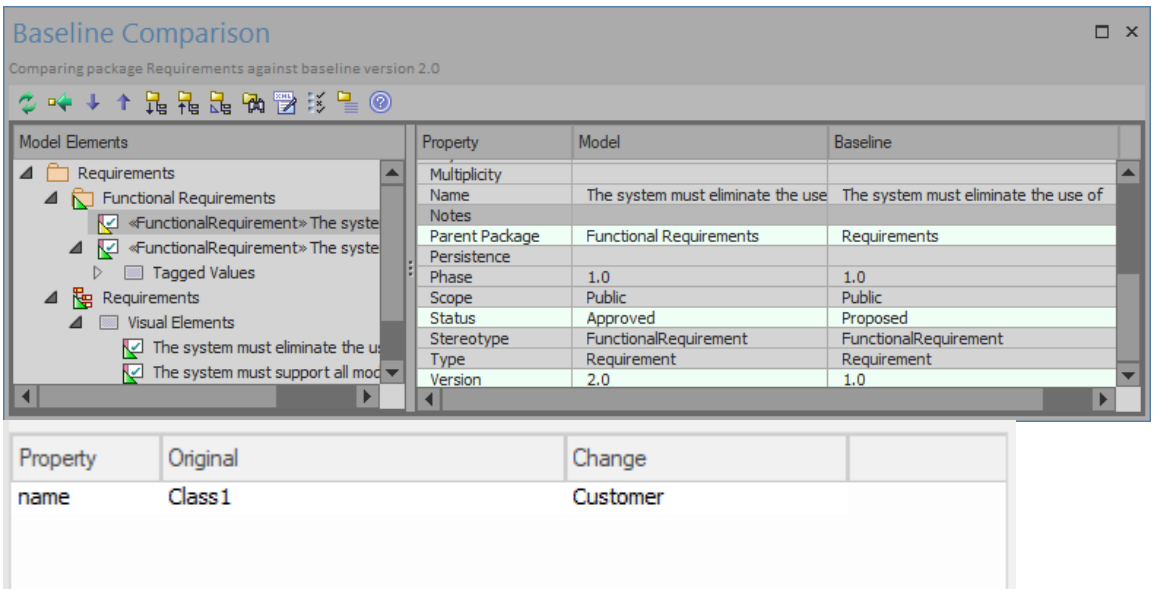
Viewing Audits

As a Librarian or Administrator, you can view the content of the Audit either through the View Audit window or in the **System Output** window. The **Audit View** provides a number of options for changing the display and provides a convenient Tree View control for navigating through the changes.



Audit View window showing the tree of items and their changes.

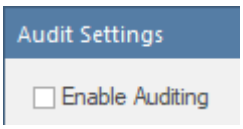
Changes are listed and ordered in time order with an icon that indicates if the change was a creation, modification or deletion. A panel on the right displays the change itself, while the header section at the top describes who made the change, when it was made and the nature of the changes.



Audit View window showing changes to a Requirement's properties, parent Package, Status and Version

Disabling Auditing

As a librarian or administrator your can disable auditing, which terminates the process of model changes being written to the audit log. You can choose to only enable

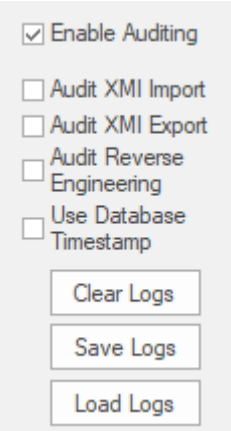


***Audit Settings window** allowing **Auditing** to be enabled or disabled.*

Auditing can be enabled again at any point in time when required. The logs, if not cleared, will simply be added to once auditing is re-enabled.

Managing Logs

When **Auditing** has been enabled changes to the elements specified in the audit options are recorded in an audit log. The management of logs is critical to the performance of the Auditing facility. When the logs fill up the time it takes to write a change will increase, subtly degrading the performance experienced by modelers as they work in the repository. The management of logs can ensure that users don't notice any perceivable change in the their save times. As the administrator or librarian you can save the logs to a network file and then clear the logs. It is recommended that this process is managed to ensure an optimal user experience. The logs can be managed from the left hand panel of the **Audit Settings window**.



Audit Settings options to Clear, Save and Load Logs.

Audit Settings

The **Audit Settings window** allows you to enable or disable auditing but in addition there are a number of options that you can use to manage what is recorded by the **Auditing** facility. These can be altered during different sessions of auditing, for example you may only want to record limited information about requirements during a phase of the project. Subsequently you may required detailed information about all element types and you could make adjustment to the settings to specify this. The important options are summarized below:

- Define whether bulk operations should be included in the audit
- Administer your audit records such as saving and clearing logs
- Specify the detail of the recorded audit by specifying the audit level
- Specify the type of objects you wish to be included in the audit

Access

Ribbon	Configure > Model > Audit : Audit Settings

Configure Settings

Option	Action
Enable Auditing	Select this checkbox to turn the **Auditing** facility on.
Audit XMI Import	Select this checkbox to record changes arising from XMI importing in the audit. As **Version Control** uses XMI, you must select this option to record changes from checking out Packages.
Audit XMI Export	Select this checkbox to record changes arising from XMI exporting in the audit. As **Version Control** uses XMI, you must select this option to record changes from checking out Packages.
Audit Reverse Engineering	Select this checkbox to record changes arising from reverse engineering in the audit.
Use Database Timestamp	Select this checkbox to use the database server's timestamp instead of each user's local timestamp; this improves security. This option is not available for project files.
Clear Logs	Click on this button to permanently delete all log data from the current project; use the **Save Logs** button first, to save the audit records outside the project. The system prompts you to specify whether to clear items logged over a specific period of time. • Click on the **No button** to clear all log items currently held in the database • Click on the **Yes button** to display the 'Time **Filter**' dialog, on which you select a standard time period or define your own This function can be accessed through the **Automation Interface**.

Save Logs	Click on this button to save a copy of the logged audit items currently held in the project. These items remain in the project; you can use the **Clear Logs button** to remove them. The system prompts you to specify whether to save items logged over a specific period of time. • Click on the **No button** to save all log items currently held in the database • Click on the **Yes button** to display the 'Time **Filter**' dialog, on which you select a standard time period or define your own This function can be accessed through the **Automation Interface**.
Load Logs	Click on this button to load a previously saved set of logs back into the project. A browser displays through which you select the log file to reload. If the same record exists in both project and log file, that record is not reloaded.
Core	Select this radio button to record changes to elements (including attributes and operations), Packages, connectors and some model-level information.
Standard	Select this radio button to record the same changes as for the Core option, plus changes to some of the 'housekeeping' data for diagrams - Name, Author, Version, Stereotype, Notes and Date Modified. You can check for changes to diagram content and structure using the Baseline facility for reviewing visual changes to diagrams.
Extended	Select this radio button to record the same changes as for the 'Standard' option, plus changes to project security.
Debug	Select this radio button to record the same changes as for the 'Extended' option plus changes to any database tables. Some of the auditing information will not be visible unless the **Audit View** mode is set to 'Raw'.
Maintenance	Select this radio button to audit changes to maintenance elements only; that is: • Package (element) • Requirement • Feature • Use Case • Actor • Note • Issue and • Change
Core Structural	Select this radio button to audit changes to maintenance elements plus certain structural elements; that is: • Package (structure) • Class • Interface • Signal • Node • Component • Artifact

	• Part • Port and • Device
All	Select this radio button to audit changes to all types of element.
Custom	Select this radio button to audit changes to element types that you specify. The **Customize button** is made available; click on this button to display a list of element types, and select the checkbox against each element type to include in the audit (or click on the **Select All** button to select every element type). Click on the **OK button** to save the selection.

Notes

- As the number of records increases, the performance of the **Audit View** reduces; it is recommended that audit records that are not regularly required are saved to file, then cleared from the project, to help ensure high performance
- Connectors are audited when they are connected to an element that is included in the Audit Options
- If security is enabled, you must have Audit View permission to turn **Auditing** on, and Audit Settings permission to change the audit settings

View the Audits

As an Administrator or Librarian you can view the changes that have been made to the items in the repository. The changes that are recorded depend on three main settings:

- Import and Reverse Engineering options - allow you to specify whether these bulk items should be recorded in the audit log
- **Auditing** Level - You can specify the extent of the information recorded
- Auditing Options - You can specify which elements will be audited

There are two different views of the changes that are available to you as follows:

- The **Audit View** - a fully featured window that allows you to navigate through the changes and change the display
- The **System Output** window - a textual display that changes as the element context changes.

Audit View

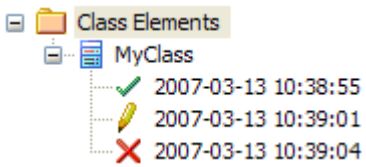
The **Audit View** provides a convenient dashboard of information allowing an administrator or librarian to view and manage the changes to the information that has been recorded by auditing. A tree view control provides a quick way to navigate through a branch of the repository showing all the items that have been changed. Colored Icons provide a quick visual queue as to whether a particular item has been created, changed or deleted. When a change is selected in the tree view the panel to the right displays the properties that have been created, modified or deleted and the top panel shows the responsible user, the time and date of the change and a summary.

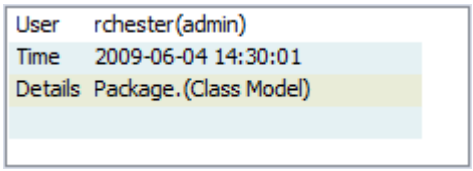
Access

Ribbon	Configure > Model > Audit

Sections of the Audit View

The **Audit View** is divided into three main areas:

Section	Description
View controls	The view controls provide a variety of settings for controlling auditing and the display of audit records.
Audit tree	The audit tree displays the log items that have been recorded by auditing. What is displayed in the tree is affected by the view controls, such as: • Sorting • **Filter** (by time) • Mode • **Auditing** settings (what was actually recorded) If synchronizing with the **Browser window**, it is also affected by the Package, diagram or element you have selected.  In the audit tree: • The green tick indicates a creation • The yellow pencil indicates an edit • The red cross indicates a deletion Right-clicking an element in the audit tree (such as MyClass) displays a context menu that enables you to locate the selected element in: • The Browser window • Any diagrams in which it exists
Record display	The record display is in two parts: the identity of the selected change, and the actual change made.

	The data in the record display is determined by the view controls and mode and, if synchronizing with the **Browser window**, by the Package, diagram or element you have selected. **Identity**  The identity of a change consists of: • The Windows username of the user that made the change; if security is enabled, the name is of the format WindowsUsername(SecurityUser) • When the change was made • The path of the change; for example: *Class* Class1 - *Attribute* Att1 - *Attribute Constraint* Constraint **Changes** Changes are displayed in a table format, showing the: • Property (or data item) name • Its original value before the change and • Its value after the change If you double-click on an item in the list of changes (or right-click and select the 'Show Differences' option) the **Difference window** displays:  This shows the specific changes that have been made, highlighting the edited, created, deleted or formatted characters. Changes to the format of text in the element, made through the element 'Properties' dialog, are not apparent in the initial table; they are visible in the Difference window, identified by HTML formatting tags.

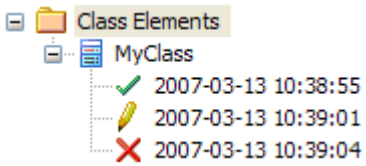
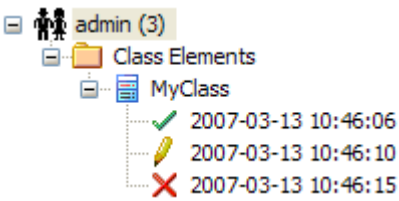
Notes

- If security is enabled, you must have **Audit View** permission to display data in the Audit View

Audit View Controls

The **Audit View** controls provide a variety of settings for controlling auditing and the display of audit records.

Options

Option	Action
Load or Refresh	Click to reload the Audit Tree, updated with any new audit results.
Search	Click to search through log items for a particular area - you can search by Name, Type or GUID. The items are loaded and filtered with the current Sort By, Time **Filter** and Mode settings. If you refresh the **Audit View**, you must run the search again.
Audit Settings	Click to open the 'Audit Settings' dialog.
Sort-by	Select the appropriate radio button for the required display setting: • Type - changes are grouped under element type (such as Class or Requirement), and then grouped under the changed element  • User - changes are grouped under user name, each with the number of changes for that user Under each user name, changes are grouped as for the Type sort 
Filter By Date/Time	Select to enable the **Filter Settings button**, to filter the audit results by time period. Changes that occur outside the selected filter period are not shown in the **Audit View**.
Filter Settings	Click to display the 'Time **Filter**' dialog, to set the filter time period: • Today - to display changes occurring today • Previous Hour - to display changes occurring in the last 60 minutes • Previous 24 Hours - to display changes occurring in the last 24 hours • Previous Week - to display changes occurring in the last 7 days • Previous 30 Days - to display changes occurring in the last 30 days • Previous Year - to display changes occurring in the last 365 days

	• Custom - to define your own time period, using the From and To fields The six pre-configured time periods automatically update when you click on the **Refresh button**; custom periods are static and do not automatically update. If you have set a filter period, and you deselect the 'Filter By Date/Time' checkbox, the period remains set; the custom time period, too, is retained so that you can re-use it or modify it later if required.
Status Text	Read to see which: • Mode has been selected and • Time filter is being applied to the data
Mode	Click to display a menu of options to change the mode of the **Audit View**. Select: • 'Standard' - to automatically synchronize with the **Browser window**; where changes have been made, the Audit View reflects your selection from the Browser window - if you click on: - An element, the Audit View displays the history for that element - A Package, the Audit View displays the history for that Package and its immediate children (but not the contents of nested Packages) - A diagram, the Audit View displays the history for that diagram and its contents (which could be drawn from a wide area of the Project Browser) • 'Advanced' - to load large sets of log items independent of the Browser window; in this mode, a special Audit Settings group can be displayed in the Audit Tree, which records: - When **Auditing** has been enabled and disabled - Who made the change - The date and time of the change - Changes to the Audit Settings - When Audit Operations are executed - Security changes (which can be browsed in the same way as other changes) • 'Deleted' - to display only deleted records, but otherwise data is shown as in 'Advanced' mode; records can be sorted by element type or by user as required • 'Raw' - to display all audit records in chronological order without sorting, enabling you to see a progression of changes; this can be especially useful in determining date-time inconsistencies Any search and filtering you define still apply, enabling you to view all of today's changes in order, or all changes for a particular element in order, or both Additional database information is displayed; this additional information might be insignificant or only in machine-readable format

Audit History Tab

When **Auditing** is turned on, an 'Audit History' tab is enabled and can be viewed in the **System Output** window. The tab shows a history of changes to whichever element or connector you have selected in the:

- Current diagram
- **Browser window**
- **Audit View**, or
- **Package Browser**

System Audit History AuditLog

User	Timestamp	Property	Original	Change
Stephen Maguire(smaguire)	2020-09-16 10:07:05	version	1.0	2.0
Stephen Maguire(smaguire)	2020-09-16 10:06:51	status	Proposed	Approved
Stephen Maguire(smaguire)	2020-09-16 10:06:51	name	Class1	Customer
Stephen Maguire(smaguire)	2020-09-16 10:06:34	name		Class1

Output window Audit History Tab showing changes to the elements in the repository.

As you select different elements or connectors, the 'Audit History' tab automatically updates to reflect your current selection. For each change made to the element or connector, the tab shows:

- Who made the change
- When the change was made
- Where the change was made
- The value of the characteristic before the change
- The value of the characteristic after the change

Access

Ribbon	Start > Desktop > Design > **System Output** > Audit History

Notes

- To see this tab, you must have the **Audit View** open; the tab continues to display if you subsequently close the Audit View
- If security is enabled, you must have Audit View permission to display data on the 'Audit History' tab

Performance Considerations

Auditing is a powerful tool for determining who made changes to the model and when the changes were made. When auditing has been enabled every time a change is made to the model the Enterprise Architect auditing engine records the change. These changes will naturally, like any transaction system, take some to be written to the audit log. When the logs are small this write time will be imperceivable but when the logs grow in size the time taken to write to the log will increase and could delay the saving of changes. The tool provides a number of ways to reduce theses delays in the saving of changes. The delays can be experienced in two different parts of the auditing process.

- **When making changes to the repository.** As described above the saving of model changes may be delayed effecting you or any user who is updating an aspect of the model that is being audited. For example a change to an element's name, changing the position of an object on a diagram or if configured the creation of tables or code classes resulting from reverse engineering a database or code package will all result in a write to the audit logs.
- **When accessing the Audit View.** The audit view that is typically accessed by a librarian or administrator can become slow dues to the volume of audit records that need to be sorted and displayed.

The performance of both these aspects of the auditing process will be covered in the following topics

Auditing Performance

Enabling auditing on a project increases the time taken for most actions such as insertions, changes and deletions of objects and changes to properties of elements, connectors and diagrams including moving objects in diagrams. When auditing is initially enabled the time required to write to the audit logs will be almost unperceivable but as the logs grow in size the log write time will become longer and may cause some frustration to you or you colleagues making whose man focus is creating useful models. Fortunately there are some simple housekeeping and administration procedures that can reduce this time typically performed by a librarian or administrator..

Operation Delays

Operation	Detail
Large Deletions	Deleting large Packages or Package structures, or large numbers of elements, takes significantly longer with auditing on. You might disable auditing before performing such a deletion.
XMI Imports	Importing XMI takes longer with auditing enabled. A project-level option is provided for disabling auditing of XMI Imports.
Reverse Engineering	Reverse engineering code takes longer with auditing enabled. A project-level option is provided for disabling auditing of reverse engineering.
Replication	You cannot remove replication from a model with **Auditing** enabled. If you have to remove replication, disable Auditing and - if prompted to do so - allow Enterprise Architect to roll back the database version; you can then remove replication.

Audit View Performance

As the logs become larger the **Audit View** can exhibit slow performance, simply because of the volume of records it needs to process to create the visualizations that are used by the librarian or administrator. Fortunately there are ways to reduce the delays to the presentation of changed items and these only require a small amount of house keeping.

Considerations and Responses

Consideration	Detail
Navigating the Browser window Within Auditing is Slow	Try setting the time filter to a period in the immediate past, such as 'Today', 'Previous 24 Hours' or 'Previous Week'; this time period updates each time you open or refresh the **Audit View**. Save log items outside the project with the **Save Logs** button; if you then clear the logs you have just saved, the load time of the Audit View is reduced. You can reload logs into the project at any time, using the **Load Logs button**.
The Audit View is Slow in Loading and Changing Modes	Try setting the time filter to a period in the immediate past, such as 'Today', 'Previous 24 Hours' or 'Previous Week'; this time period updates each time you open or refresh the **Audit View**. Save log items outside the project with the **Save Logs** button; if you then clear the logs you have just saved, the load time of the Audit View is reduced. You can reload logs into the project at any time, using the **Load Logs button**.
Navigating the Audit Tree is Slow	Close the 'Audit History' tab in the **System Output** window

More Information

Edition Information

- The **Auditing** facility is available in the Corporate, Unified and Ultimate Editions
- Warning - If your site runs separate Editions of Enterprise Architect, when Auditing is turned on in a project any Professional Edition users are locked out of the project; to restore access, turn Auditing off in the project from the Corporate, Unified or Ultimate Edition instance of Enterprise Architect

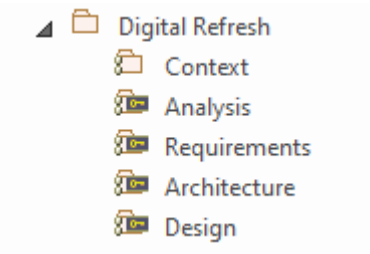
Related Tools

- Version Control
- Baselines
- The Compare Utility (Diff)

Version Control

Stores Versions of parts of a Repository for Comparison and Rollback

Version Control provides a mechanism to store previous versions of any section of a model and later compared to a current state. Version Control is a powerful team feature that allows models to be developed in a coordinated way within a team environment. Sections of the model can be placed under Version Control and you can manage changes to and revisions of your model content by placing individual model Packages, view nodes or root nodes under Version Control. The information stored within the repository are valuable corporate assets and need to be managed as such.



Browser window showing one Checked-Out Package (Context) and four Checked-In Packages under Version Control.

Version Control provides a safeguard against losing previous model changes. In this topic you as the Librarian will learn how to configure a Version Control product for use within Enterprise Architect and to apply the configuration to control Packages, individually or in groups. You will also learn how to use Version Control to Check-Out and Check-In Packages, how to view previous versions, and compare a current version to a previous version.

Brief Introduction

The **Version Control** feature allows you to check-out a Package, which then **locks** it so that no other user can modify it until you have checked it back in again. You can then modify the Package, adding new elements and diagrams, changing others and deleting still others. When you have completed your changes and are ready for other modelers to see your work, you can check-in the Package. Other modelers would then need to do a Get Latest to pull the new changes down from the Version Control Server.

Enterprise Architect does not version the Packages internally; instead it integrates with industry standard Version Control servers, which an administrator must install and configure for use by Enterprise Architect. The files that Enterprise Architect uses are industry standard XMI files, which are best seen as 'binary' from the point of view of Version Control. This means that it is not possible for two modelers to be working on the same file, as it would not be possible to merge the files as is possible with source code or text files.

The basic steps to use Version Control are:

1. Select and set up a Version Control Server and Client side Software.
2. Place the required Packages under Version Control.
3. Check Out a Package.
4. Make modifications to the contents of the Checked Out Package.
5. Check In the Package, entering comments that describe the changes.
6. Compare the Repository Package with the Version Controlled Package.
7. Retrieve (rollback to) a previous version.

Enterprise Architect's Version Control integration provides several key facilities, including:

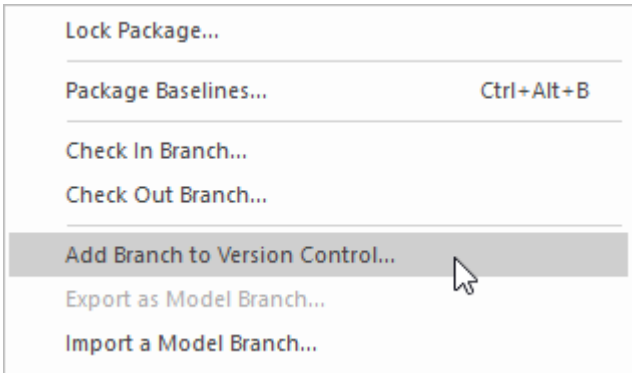
- Saving a history of changes made to your model's Packages
- Retrieving previous revisions of Packages
- Propagating model updates between team members
- Coordinating the sharing of Packages between team members

Commissioning a Version Control System

You apply **Version Control** through a third-party source-code control application that manages access to and stores revisions of the controlled Packages. Once the Version Control software has been installed and configured, you or a librarian must define a Version Control Configuration within your project. You can then use Version Control to manage changes to the Packages of your model.

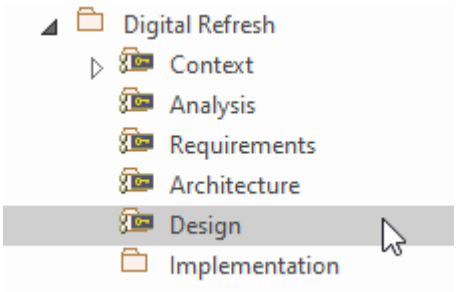
Configuring Packages

Before you can Check Out a package and begin making changes you or the librarian must first configure the package for **Version Control**. You simply select the Add Branch to Version Control option and choose an existing configuration. This to adds the package to the Version Control System and places the package in a Checked-In state.



Context menu showing the Add Branch to Version Control option.

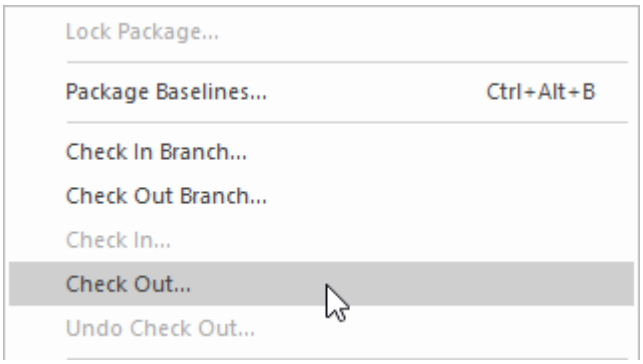
Packages under Version Control are identified in the **Browser window** by icons that indicate the current status of the Package. When a package is initially added to the Version Control System it will have a locked icon as shown in the diagram below.



Browser Window showing the Design Package with a Lock icon after Check-In.

Checking Out Packages

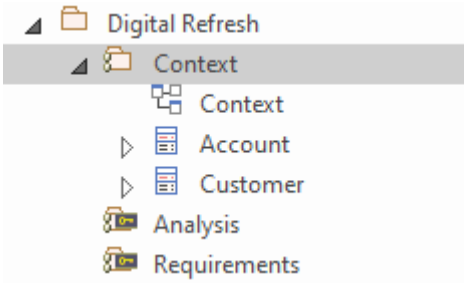
Once **Version Control** has been set up and configured you can start checking out Packages. When you want to work on a Package that is Version Controlled, you need first to Check Out that Package. This will change the Package icon in the **Browser window** from locked to unlocked. While you have the Package checked-out no other user will be able to work on the Package and they will in turn see a locked Package icon.



Context menu showing the Check Out option.

Make Modifications to Package Content

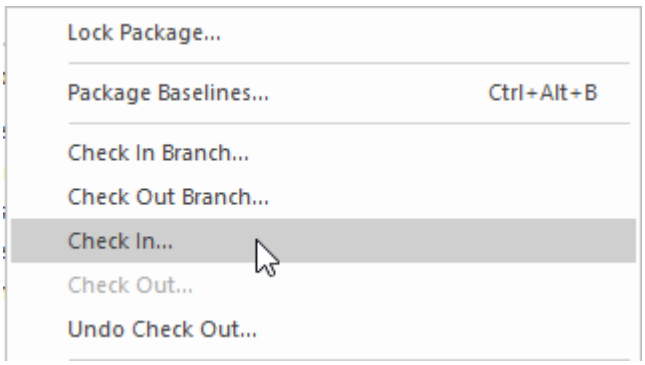
Once a Package has been checked out you are free to make modification to the content, just as you would in a repository that doesn't have any **Version Control** implemented. You will be the only one able to edit the content, as other users that attempt this will receive a message saying the Package is checked out.



Browser window showing the Context Package that has been checked out, showing the Checked-Out Package icon

Checking In Packages

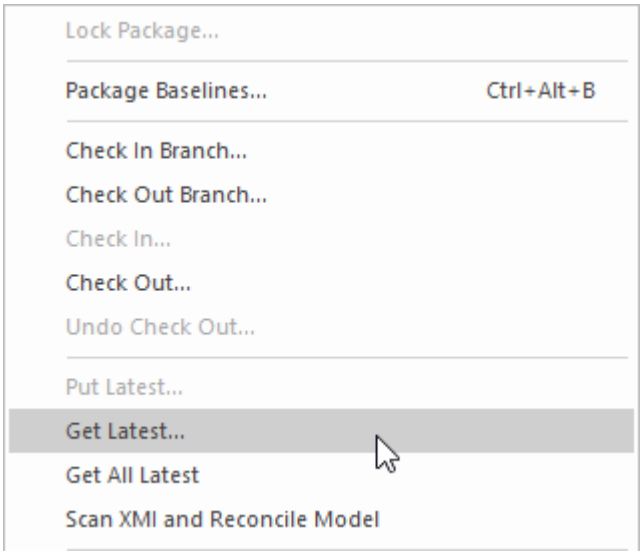
Once you have finished making changes to the contents of a Checked-Out Package, you can Check In that Package. This will change the Package icon in the **Browser window** from unlocked to locked. As soon as you Check-In the Package it will display a locked Package icon and the **Version Control** system will be updated. You can also enter a comment that describes the changes that you made while the Package was under your control, which will be useful when looking back at version changes. Once the Check-In is complete, other users will be able to Check-Out the Package.



Context menu showing the Check In option.

Getting Latest Changes

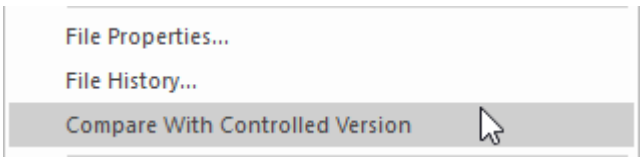
If you are working in a single shared model any changes that your colleagues have made will be available by simply reloading the package. If however you are working in private models then you will need to periodically get the latest changes that your colleagues have check in to the **Version Control** system. You can Get Latest for a single package or for the entire model depending on whether you are interest are in just one or all of the changes.



Context menu showing the Get Latest Changes option.

Comparing to Previous Version

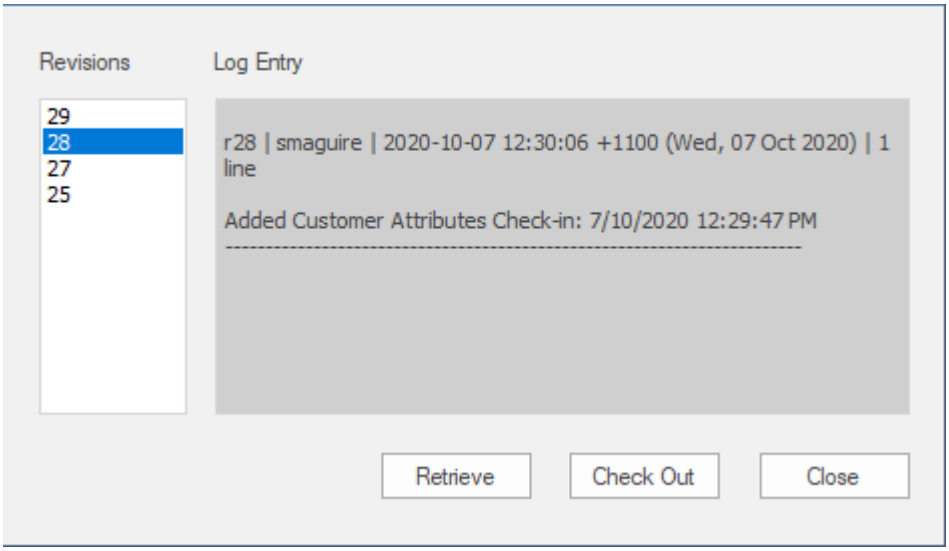
At any stage of a model's evolution you can compare the current version of a Package to the version stored in the **Version Control** system. This option launches the Comparison window that displays the differences between the model's current state and the latest revision held in the Version Control system.



Context menu showing the 'Compare with Controlled Version' option.

Reverting to a Prior Version

As your model evolves you might want to revert to a previous version of a Package. By using the 'File History' dialog you can select any prior version. By viewing the Check-In comments and the Date and Time of the Check-In the required version can be selected. The Retrieve option will restore the current model to the selected version without the need to check-out the Package.



*File Version **History window** showing a number of versions (revisions) with Comments.*

Setup Options

There are a number of options that can be chosen when setting up **Version Control** for one or more teams in your organization. There are also some fundamentals about the way that Enterprise Architect integrates with the Version Control software with respect to locking that you should review before setting up Version Control.

Version Control Locking Overview

Enterprise Architect implements **Version Control** of your model by exporting Package data from the project database to XMI Package files, which are placed under Version Control in the source-code control application. The XMI file format cannot be merged in the same way as ordinary text files can be merged, which is why Enterprise Architect must enforce serialized editing of Version Controlled Packages, as discussed here.

The Lock-Modify-Unlock Solution

Many **Version Control** systems use a lock-modify-unlock model to address the problem of different authors in a shared source overwriting each other's work. In this model, the Version Control repository allows only one person to change a file at a time, and access is managed using **locks**.

Harry must lock a file before he can begin making changes to it. If Harry has locked a file, Sally cannot also lock it, and therefore cannot make any changes to that file. All she can do is read the file, and wait for Harry to finish his changes and release the lock. After Harry unlocks the file, Sally can take her turn in locking and editing the file.

The Copy-Modify-Merge Solution

Subversion, CVS and a number of other **Version Control** systems use a copy-modify-merge model as an alternative to locking. In this model, each user's client contacts the project repository and creates a personal working copy - a local reflection of the repository's files and directories. Users then work simultaneously and independently, modifying their private copies. In due course, the private copies are merged together into a new, final version. The Version Control system often assists with the merging, but ultimately a person is responsible for making it happen correctly.

When Locking is Necessary

While the lock-modify-unlock model is generally considered a hindrance to collaboration, there are still times when locking is necessary.

The copy-modify-merge model is based on the assumption that files are contextually merge-able: that is, that the files in the repository are line-based text files (such as program source code). However, for files with binary formats, such as artwork or sound, it is often impossible to merge conflicting changes. In these situations, it really is necessary for users to take strict turns in changing the file. Without serialized access, some users end up wasting time on changes that are ultimately overwritten.

Repository Options

The **Version Control** facility can be used in many different ways, although these roughly fall into one of four types of use as discussed in this table.

Usage

Type	Details
Single Shared model	Users share a model stored in a central project file or DBMS repository. In this configuration you can view changes to other users' Packages without explicitly having to check them out, by simply refreshing your view of the model. **Version Control** is used to: • Archive successive versions of your work to date • Maintain Package revision history • Recover from unwanted changes or accidental deletions, through an 'Undo' facility • Regulate access to Packages
Multiple Private models	One model is created by a single user who configures it for **Version Control**. The model file is then distributed to other users, with each user storing their own private copy of the model. Version Control is used to: • Propagate changes to the model, throughout the team • Archive successive versions of your work to date • Maintain Package revision history • Recover from unwanted changes or accidental deletions, through an 'undo' facility • Regulate access to Packages
Shared Packages	Individual users create separate models but share one or more Packages: • Users share Packages through **Version Control**
Standard Packages	A company might have a standard set of read-only Packages that are broadly shared: • Individual users retrieve Packages with the Get Package menu option

Factors to consider

Factor	Detail
System Requirements and Configuration	You apply **Version Control** through a third-party source-code control application that manages access to and stores revisions of the controlled Packages. Typically the configuration consists of: • A server component that manages a Version Control repository, and

	• Client components on the workstations, that manage local working copies of controlled files Enterprise Architect uses the client component to communicate with the server. A Version Control client must be installed on every machine where you run Enterprise Architect and want to access your Version Control system.
Version Control Usage	There are two main ways in which projects can be deployed: • Centralized Shared Model • Distributed Private Models **Version Control** is employed in the same way for both scenarios; however, when using Private Model deployment you have the additional facility of propagating model updates throughout the team. Version Control can also be used to share standard Packages between different projects.
Team Deployment	Consider the process of setting up a **Version Control** environment and applying Version Control to a project to be accessed by a number of users.
Version Control Basics	Enterprise Architect enforces serialized editing of Version Controlled Packages, using the lock-modify-unlock mode of operation.
Applying Version Control to Models	Using **Version Control** consists of placing individual model Packages under Version Control, rather than Version Controlling the project as a whole.
Version Control and Project Reference Data	To share changes in reference data between users in a version-controlled project deployed as multiple private models, you periodically export the reference data from the model where the changes were made, and import it into the other models maintained by the team.
Offline Version Control	You can prevent the system from attempting to make any **Version Control** connections by choosing to Work Offline before loading a model. If Enterprise Architect is unable to connect a Version Control Configuration for any reason, it displays warning messages to notify you and provides 'offline' Version Control functionality for all Packages associated with the failed connection.

Version Control of Model Data

When applying **Version Control** in Enterprise Architect, you place individual model Packages under Version Control, and not the project as a whole.

All Enterprise Architect models are stored in databases - even the .eap file is an MS Jet database. In simple terms, the project file is a single entity of binary data. Being binary data, the project file would require the use of the lock-modify-unlock model of Version Control, which would mean that only a single user at a time could work on any given (Version Controlled) model. Therefore, it is not practical to apply Version Control to the database (.eap file) as a whole; this can also create problems for you in that:

- Most Version Control systems mark their controlled files as read only, unless they are specifically checked-out to you
- The .eap file is an MS Jet database, and Enterprise Architect must be able to open this file for read/write access when you load your model; if the model is read-only, the system displays an error message and fails to load the model

Version Controlling Packages in your Model

To overcome these limitations, Enterprise Architect exports discrete units of the model - the Packages - as XMI Package files, and it is these XMI files, not the project file, that are placed under **Version Control**. The XMI file format used by Enterprise Architect dictates that they too be treated as binary files - therefore it is not possible to merge the XMI files either; however, by splitting the model into much smaller parts, many users can work on separate parts of the model simultaneously.

Version Controlled Nested Packages

Version Controlled nested Packages result in much smaller XMI files being exported, as the parent Packages' XMI files do not contain any content for the Version Controlled child Packages.

Version Control of nested Packages, together with a model structure of small individual Packages, provides greater scope for multiple users to work concurrently, as individual users are locking much smaller parts of the model.

Version Control Nested Packages

When you save a Package to the **Version Control** system, only stub information is exported for any nested Packages. This protects information in a nested Package from being inadvertently over-written by a top level Package.

Operations on Nested Packages

Operation	Detail
Checking Out	When you check out a Package, you do not modify or delete nested Packages; only the top level Package is modified. As a consequence of this behavior, if you check out or get a Version Controlled Package with nested Packages that are not already in your model, you see stubs in the model for the nested Packages only.
Get All Latest	If you select the 'Get All Latest' option from the **Version Control** menu, any new stubs are populated from the Version Control system.
Importing Models	You can populate a large and complex model, by 'getting' only the root Packages, then using 'Get All Latest' to recursively iterate through the attached and nested Packages. This is a powerful and efficient means of managing your project and simplifies handling very large models, even in a distributed environment. The 'Import a Model Branch' option combines the procedure steps into a single operation.

Notes

- It is recommended that, when sharing a Version Controlled model, you do not mix versions of Enterprise Architect later than version 4.5 with earlier versions; if this is necessary it is best to go to the '**Version Control** Settings' dialog and deselect the 'Save nested version controlled packages to stubs only' checkbox, setting Enterprise Architect to the pre-version 4.5 behavior (for the current model only)

Version Control and Reference Data

Reference data is data that is used across a model or project; it is not Package-specific. **Version Control** operates at Package level, and therefore does not capture changes in reference data. Where Version Control is used in a multiple private model set up, changes in reference data are not brought into the model when Packages are updated from Version Control.

In a Shared Model environment, all users are accessing the same project reference data. Changes in reference data can be shared between users in a Version Controlled project deployed as multiple private models, by periodically exporting the reference data from the model where the changes were made, and importing it into the other models maintained by the team.

Reference data is exported and imported as an XMI file, which contains whatever types of reference data you want to transfer. You can place your project reference data under Version Control by exporting the data as an XMI file and apply Version Control to that file using your Version Control software external to Enterprise Architect.

Version Control and Teams

This is a summary of the process of setting up a **Version Control** environment and applying Version Control to a project to be accessed by a number of users.

Version Control - Process Overview

Step	Action
1	Install your **Version Control** product.
2	Create a **Version Control** repository.
3	Create a **Version Control** project to be used with your Enterprise Architect project.
4	Check-out a working copy of the **Version Control** project (a module, project or folder within the Version Control system) into a local folder. You must do this for every team member that is accessing the Version Controlled Packages, whether you are using a single shared model or each team member stores his own private copy of the model.
5	Before attempting to access the new **Version Control** environment from within Enterprise Architect, you should first verify that it functions correctly when used outside of Enterprise Architect.
6	Within Enterprise Architect, define a **Version Control** configuration to provide access to the working copy files. The name of the Version Control configuration must be the same across all machines throughout a team. That is, all Version Control access to a given Package must be through Version Control configurations with the same name, across all models and all users. The easiest way to perform this step (throughout the team) is to have one user set up Version Control on the model and then share that model with the rest of the team. • In Shared Model deployment, all users connect to a single instance of the model database, so the model is shared automatically • In Private Model deployment, it is easiest to distribute copies of the original model (after Version Control has been set up) to all other members of the team Whenever you open a model (Private or Shared) that uses a Version Control configuration that is not yet defined on your workstation, a prompt displays to complete the definition for that configuration. This typically means specifying the local working copy directory and perhaps choosing the Version Control project associated with this Enterprise Architect project. Once this has been done, the Version Controlled Packages that already exist in the model are ready for use.
7	Configure Packages within the Enterprise Architect model for **Version Control**. That is, apply Version Control to individual Packages.
8	Check-out and check-in Packages as required.

Notes

- It is possible to use multiple **Version Control** Configurations within the same model; different Packages can still use different Version Control Configurations within the model, as long as any given Package is always accessed via the same Version Control Configuration

Offline Version Control

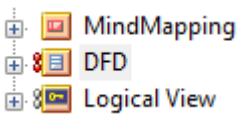
When loading a model that uses **Version Control**, Enterprise Architect normally initializes a connection to the Version Control system for each Version Control configuration defined in the model. If Enterprise Architect is unable to connect a Version Control configuration for any reason, it displays warning messages to notify you and provides offline Version Control functionality for all Packages associated with the failed connection.

You can prevent Enterprise Architect from starting to make any Version Control connections, by selecting to work offline before loading your model.

Access

Ribbon	Configure > **Version Control** > Work Offline

Working Offline

Concept	Discussion
Choosing to Work Offline	Selecting to work offline is useful if you know beforehand that Enterprise Architect cannot connect to your **Version Control** system. For example: If you are working on a laptop computer that is disconnected from your network, on an Enterprise Architect model that uses a large number of Version Control configurations, you can select to work offline before you load the model to avoid all the error messages that the system would normally display as each Version Control connection attempt failed. You can switch between working offline and working online at any time, either before or after a model is loaded, by toggling the 'Work Offline' menu option. Enterprise Architect disconnects or reconnects Version Control (depending on the connection availability) according to your selection.
Using Version Control Whilst Disconnected From the Version Control Server	Enterprise Architect 'remembers' the status of a model's Version Controlled Packages. Packages that were checked out to you prior to disconnecting from the server are still shown as checked out to you, even though you are no longer connected to the server. You can still edit these Packages as you normally would. Packages that were not checked out to you prior to disconnecting from the server are shown as Version Controlled and locked. You cannot edit these Packages until you check them out.
Offline Check Out	You can 'check-out' and edit a Version Controlled Package even when your machine is disconnected from the **Version Control** server. In this example, the colored 'figure 8' icon for DFD indicates that you have checked it out whilst off line.  (The gray 'figure 8' icon shown against Logical View indicates that you have

	checked out a Version Controlled Package on line.) You should be aware that the Version Control system, and therefore other users, have no way of knowing that you have 'checked-out' a Package whilst offline. It is not possible to merge changes to an XMI file that result from two users editing the same Package at the same time. If an offline checkout leads to two people editing the same Package at the same time, when the changes are brought back online the first-saved set of changes is lost.
Checking In a Package That Was Checked Out Offline	Once you reconnect your system to the **Version Control** server, if the Package you checked out offline is not currently checked out by another user, you can check in that Package. However, before Enterprise Architect checks in the Package, it compares the local working copy of the Package file with the latest revision in the repository. (These Package files remain unchanged in your work area until Enterprise Architect exports the Package again before checking in.) If the repository version remains unchanged from when you last updated your local copy, Enterprise Architect exports and checks in your Package without further prompting. On the other hand, if the repository now contains a file that has changed since you last updated your local copy, checking in your Package would overwrite those changes. Enterprise Architect displays a message warning you of the pending data loss and giving you the opportunity to abort the check in. At this point, you must decide whether to discard your own changes, using the 'Undo Check Out' command, or continue with your check in and overwrite the changes that have been committed to the repository since you last updated your local copy from the repository. You can use the 'File Properties' command to determine who checked in the last changes to this Package. This might help you to discover what changes have been uploaded and decide whose changes take precedence.
Update Before You Disconnect	Whenever you are connected to the **Version Control** server, you are always working with the latest version of a Package. This is because you cannot modify a Package until you check it out from Version Control, and checking it out loads the latest revision from the repository into your model. This cannot happen when you are disconnected from the Version Control server. You are working on whatever versions you have on your machine, dating back to the last time you updated your local copy of each Version Controlled Package. So, if you are planning to work on a model whilst disconnected from Version Control, it is a very good idea to make sure that you have the latest versions of all Packages before you disconnect. The 'Get All Latest' option makes this a simple task.

Version Control Branching

Currently, Enterprise Architect does not support **Version Control** Branching.

Work-arounds to achieve similar results might be possible for certain version-control products; contact Sparx Support for advice.

Contacts

User Type	Contact via
Registered users	https://sparxsystems.com/registered/reg_support.html
Trial users	support@sparxsystems.com

Version Control Product Setup

To control and maintain the different revisions of your project Packages, Enterprise Architect uses third-party **Version Control** products. Once your Version Control product is installed and a suitable environment has been created, Enterprise Architect can use that environment to control your project's Packages.

Typically, Version Control products consist of:

- A server component
- A client component

Enterprise Architect integrates with the Version Control client components for Subversion, CVS and MS Team Foundation Server command line clients, as well as for products having API clients that comply with the MS SCCI specification.

Version Control System Components

Component	Detail
Version Control Server	The server component maintains the controlled files in their many revisions, in a central repository. The server component is usually located on a server machine that is accessible to all team members who are using **Version Control**.
Server Configuration	The steps for configuring a **Version Control** server are, broadly: • Install the software • Create a repository • Create Version Control projects (or modules or folders for use with specific projects) • Configure user IDs and passwords For details on configuring any particular Version Control server, consult the appropriate documentation provided with the Version Control product.
Version Control Client	The client component manages the working copies of the controlled files, submitting files to or retrieving files from the server as required. A **Version Control** client must be installed on every machine on which you run Enterprise Architect and want to access your Version Control system.
Client Configuration	The steps for configuring a **Version Control** client are, broadly: • Install the software • Create a new directory for use as a local working copy folder • Log in to the Version Control server • Associate the working copy folder with its corresponding server repository folder For details on setting up a product-specific Version Control environment for use with Enterprise Architect, click on the appropriate link in the next column.

System Requirements

Enterprise Architect is a Windows-based application and requires a Windows-based **Version Control** client for integration. It is independent of the Version Control server component and the platform on which that runs.

Version Control Product Requirements

Product	Detail
Subversion	Subversion is free, open source software. Subversion server components are available to run on a wide range of different hardware and operating systems. Enterprise Architect is not affected by your choice of server components, but requires Subversion's Windows-based command line client for integration. There are many graphical user interface clients available for use with Subversion, such as TortoiseSVN; this type of client cannot be used directly for integration with Enterprise Architect, but can be very useful in preparing a working Subversion environment for use by Enterprise Architect. Binary Packages are available for download from: • http://subversion.apache.org/packages.html Subversion documentation is available from: • http://svnbook.red-bean.com/nightly/en/index.html
Concurrent Versions System (CVS)	CVS is free, open source software. CVS server components are available to run on a wide range of different hardware and operating systems. Enterprise Architect is not affected by your choice of server components, but requires CVS's Windows-based command line client for integration. There are many graphical user interface clients available for use with CVS, such as TortoiseCVS; this type of client cannot be used directly for integration with Enterprise Architect, but can be very useful in preparing a working CVS environment for use by Enterprise Architect. The software is available for download from: • http://www.nongnu.org/cvs/ CVS documentation is available from: • http://cvsbook.red-bean.com/cvsbook.html
Microsoft Team Foundation Server	Enterprise Architect is able to use either the: • Command line client for TFS, or • MS TFS-SCC client Your choice of client affects how you specify your **Version Control** Configuration. MS TFS-SCC clients are available for download from Microsoft's web pages. (Run an Internet search for 'TFS MSSCCI'.)
Common Source Code Control (SCC)-compatible products	Any **Version Control** product that provides a client that complies with the Microsoft Common Source Code Control standard, version 1.1 or higher, can be integrated with Enterprise Architect. These products are SCC-compatible and are known to successfully integrate with

	Enterprise Architect: • Accurev • ClearCase • MS Visual Source Safe • MS TFS-SCC • MKS Source Integrity • Perforce • Source Offsite • Snapshot CM Products that do not appear in this list should still integrate successfully with Enterprise Architect, if there is a client available for that product that complies with the MS SCC API specification.

Create a Subversion Environment

You can use Subversion as a **Version Control** provider for Enterprise Architect. The first step in doing this is for a Subversion administrator to install and configure the appropriate software. A number of basic tasks are performed in creating an operational Subversion environment, and useful tools are available for performing some of these tasks.

Tasks in Creating a Subversion Environment

Task	Detail
Install server components	Executable files for Subversion can be obtained from the Apache Software Foundation. Subversion server components are available to run on a wide range of different hardware and operating systems; Enterprise Architect is not affected by your choice of server components. VisualSVN is a package that can greatly simplify the installation, configuration and management of your Subversion server.
Create a repository	Please consult the official Subversion documentation.
Create Subversion users	Please consult the official Subversion documentation.
Create a new repository sub-tree	It is good practice to create a new repository sub-tree in Subversion for each new Enterprise Architect model being added to **Version Control** with Subversion. Users should create a new local working copy from the sub-tree to be used with that model. TortoiseSVN can greatly simplify the process of creating new repository sub-trees.
Install client components	Executable files for Subversion can be obtained from the Apache Software Foundation.
Create a working copy folder	A working copy folder must exist on each users' machine, for Enterprise Architect to use when exporting and importing the Version Controlled Package files. It is this folder that is specified as the Local Project Path, when defining your **Version Control** Configurations. The working copy folder is the 'sandbox' where you modify the controlled files. The working copy folder is usually associated with a folder that exists within the Version Control repository. In Subversion, to create a local working copy you perform an initial check-out of a folder from the Subversion repository; this downloads a copy of the folder and its contents, to create your local working copy. TortoiseSVN can greatly simplify the initial check out of a working copy folder.
Setting up Subversion under Wine/CrossOver	The process of setting up and using Subversion with Enterprise Architect under Wine is almost identical to the process when running natively under Windows, apart from minor differences in installing the Subversion client and performing the initial check out of the working copy folder.

Notes

- Enterprise Architect relies on exclusive file locking when applying **Version Control** to its Packages; file locking was not introduced into Subversion until version 1.2, therefore Enterprise Architect does not work with Subversion releases earlier than Subversion 1.2
- Enterprise Architect can only communicate with the Subversion server using the Subversion command line client svn.exe

Create a new Repository Sub-tree

When you set up Subversion as your **Version Control** tool, it is good practice to create a new repository sub-tree in Subversion for each new Enterprise Architect model. The sub-tree can be used to control the Package files for your model.

Create a new sub-tree in the Subversion Repository

Step	Action
1	Use Windows Explorer to create a temporary directory on your PC file system, to be imported into the Subversion repository as a new repository sub-tree. The directory would have this structure:
2	Open a Windows command prompt, navigate to *tempDir* and issue the Subversion command *import*. For example: C:\Documents and Settings\user> cd \tempDir C:\tempDir> svn import . https://host.example.com:8443/repos/ --message "Repository Initialization"
3	On your PC, delete the temporary directory structure (tempDir) and all its contents.

Notes

- After the import is finished, the original tree is not converted into a Subversion working copy; you should delete the temporary structure and check out a fresh working copy of the tree
- The process can also be performed using TortoiseSVN's Repository Browser, which provides commands for simply creating new folders directly in the repository

Create a Local Working Copy

In order to use Subversion to provide **Version Control** of the Packages in a model, you need to prepare a functional SVN working copy folder that can be accessed through an Enterprise Architect Version Control configuration within that model.

Create a Subversion working copy folder

Step	Action
1	Choose or create a suitable directory on your system in which to create your Subversion working copy.
2	Open a command prompt window and navigate to the directory you have selected. For example: C:\> mkdir mySVNWorkSpace C:\> mkdir mySVNWorkSpace/myEAModelName C:\> cd mySVNWorkSpace/myEAModelName
3	Perform the initial check out from the Subversion repository, specifying the repository folder and local working copy folder, as well as your user name and password. For example: C:\> svn checkout "https://myserver:8443/svn/repos_folder "C:\mySVNWorkSpace/myEAModelName" --username myUserName --password myPassword (By specifying your Subversion username and password, you ensure that they are correctly cached by Subversion and available for use by Enterprise Architect.) If you specify the HTTPS protocol when performing the initial Subversion check out, a prompt displays to accept a security certificate; in this instance, press the **P key** to permanently accept the certificate. The nominated local folder is configured as a Subversion working copy folder. Any files already existing in the repository folder are downloaded to the working copy folder as working copy files.
4	Verify that your Subversion environment functions correctly.

Notes

- It is important that Subversion caches your username and password, so that it never has to prompt for user input; a check-out operation might not request authentication, and if it does not you should perform an action that does request authentication, such as adding and committing a dummy test file
- The process can also be performed using TortoiseSVN's Checkout command (which provides options to browse when specifying both your repository folder and your local folder); when prompted for authentication details by TortoiseSVN, make sure you place a check against the 'Save Authentication Data' option

Verify the SVN Workspace

After creating the Subversion local working copy to hold the working copies of your Package files, you can verify that it functions correctly before attempting to use it with Enterprise Architect. You need to be able to add files to Subversion, lock the files and commit changes to those files.

Verify correct operation of a Subversion working copy folder

Step	Action
1	Use Windows to open a command prompt window.
2	Select the directory you specified as the working copy, in the Subversion checkout command when preparing the SVN workspace. For example: C:\> cd mySVNWorkSpace
3	Create a test file, such as *Test.txt*, containing the text *Subversion Test*. For example: C:\> echo Subversion Test > Test.txt
4	Execute each of these Subversion commands: • svn add Test.txt • svn commit -m"Commit comment" Test.txt • svn update Test.txt • svn lock Test.txt • Use your preferred editor to modify the file and save the changes • svn commit -m"Second commit comment" Test.txt The commands should execute without any errors and without prompting the user for any extra input.

Notes

- Your environment must be set up such that you can perform these operations without ever being prompted for user ID or password; for further information, please see the Caching Client Credentials topic in the official Subversion documentation

Subversion Under Wine-Crossover

If you want to set up and use Subversion with Enterprise Architect under Wine, the process is almost identical to the process for setting up and using the systems under Windows. When running Enterprise Architect under Wine or CrossOver, you still use a Windows-based Subversion command line client.

There are some minor differences in the way you prepare the Subversion working environment, specifically in the way you install your Subversion client and the way you check out a working copy folder from the Subversion repository.

System Requirements

Sparx Systems has tested the use of Enterprise Architect with Subversion under Wine 1.2, on macOS 10.4 and 10.6.2, and on Ubuntu 10.04. All tests were passed.

When using Wine 1.2 on the Ubuntu 9.10 platform, Sparx Systems was able to use the svn: and file: protocols to communicate with the SVN server; but not the https: protocol.

Installing a Subversion Client

Wine is able to install applications from either a Windows .exe file, or a .msi installer file.

Place the installer for your Windows Subversion client in a convenient location on the native file system, then open a Wine console window from within Enterprise Architect and run the installer from within the Wine console. Your Subversion installation can then access the same C: drive and folders that Enterprise Architect is accessing.

Preparing a Subversion Environment Under Wine

Under Wine, you can install Subversion from either a Windows .exe file, or a .msi file. By performing your Subversion installation and initial check out from a Wine console window opened from within Enterprise Architect, you have access to the same C: drive and folders that Enterprise Architect is accessing.

Set up Subversion for use with Enterprise Architect, running under Wine

Step	Action
1	Start Enterprise Architect. You do not have to load a project at this point.
2	From the 'Start' ribbon, select 'Desktop > Preferences > Toolbars... The 'Customize' dialog is opened. Select the 'Tools' tab, then click on the  icon. A new, blank entry is opened on the 'Tools' tab of the 'Customize' dialog.
3	Define the new menu item entry: • In the newly-opened 'Menu contents' field, type the name 'Wine Console' • In the 'Command' field, type 'wineconsole' • In the 'Arguments' field, type 'cmd' • Leave the 'Initial directory' field blank
4	Click on the **Close button**. The 'Customize' dialog closes.
5	Select the ribbon option: 'Configure > User Tools > Wine Console'. A Wine console window opens.
6	Type 'C:' and press the **Enter key**. The Wine console switches to the C: drive.
7	Issue the command to install your Subversion client. For example: C:\>/Installers/Subversion-client-1.9.5.win32.exe To install from a .msi file, use Wine's msiexec utility. For example: C:\>msiexec /i "Installers\Slik-Subversion-1.9.7-win32.msi" Installation of the Subversion command line client begins.
8	Create a folder to serve as the working copy folder to be used by Enterprise Architect. For example: C:\>mkdir C:\VC_workspaces\SVN_workcopy
9	Issue the command to perform the initial checkout from the Subversion repository, specifying the repository folder, working copy folder, username and password. For example: C:\>svn checkout "https://myServer:8443/svn/repos_folder" "C:\VC_workspaces\SVN_workcopy "

	--username myUserName --password myPassword (After specifying your Subversion username and password, they are correctly cached by Subversion and are available for use by Enterprise Architect.) If the HTTPS protocol is specified when performing the initial Subversion check out, you are prompted to accept a security certificate; in this instance, press the **P key** to permanently accept the certificate. The nominated local folder is configured as a Subversion working copy folder. Any files already existing in the repository folder are downloaded to the working copy folder as working copy files.
10	Type 'Exit' and press the **Enter key**. The Wine console window closes. You are now ready to load a project in Enterprise Architect and apply **Version Control** to it, using the normal Windows-based procedures.

Notes

- You should copy the installer for your Windows Subversion client to a convenient location on the native file system, so that you can easily refer to it from within the Wine console window in step 7

TortoiseSVN

TortoiseSVN is a Windows shell extension for Subversion; it provides icon overlays in Windows Explorer that are useful as a tool for observing the status of your Subversion controlled files. You can also use it to create your repository sub-trees and check out local working copies from within Windows Explorer, using simple menu commands.

You can download TortoiseSVN from http://tortoisesvn.net/downloads.html.

Notes

- Enterprise Architect can only communicate with the Subversion server using the Subversion command line client svn.exe

Create a TFS Environment

You can use Microsoft Team Foundation Server (TFS) as a **Version Control** provider for Enterprise Architect. The first step in doing this is for a TFS administrator to install and configure the TFS server and client applications. A number of basic tasks are performed in creating an operational TFS environment.

Tasks in Creating a TFS Environment

Task	Detail
Obtain and install TFS	Enterprise Architect uses the TFS command line client to integrate TFS **Version Control**. The TFS command line client is normally available as part of your Visual Studio installation.
Choose a TFS project	It is good practice to create a new TFS project, or least a new Source Control Folder within a project, for each Enterprise Architect project being added to **Version Control** with TFS. If you have a single Enterprise Architect project that contains many different models (for example, a DBMS hosted project with multiple model root nodes), you might choose to create a new TFS project for each separate model. For further information, please consult your TFS product documentation.
Create a TFS workspace	A working copy folder must exist on each users' machine, for Enterprise Architect to use when exporting and importing the Version Controlled Package files. It is this folder that is specified as the Local Project Path, when defining your **Version Control** Configurations. The working copy folder is the 'sandbox' where you modify the controlled files. The working copy folder is usually associated with a folder that exists within the Version Control repository. In TFS, the TFS workspace is used to map a local working folder on your PC to a Source Control Folder within a TFS project. TFS 2012 and VS 2012 (and later versions) feature a new type of workspace called 'local' workspaces. Do not attempt to use TFS 'local' workspaces with Enterprise Architect. You must use only 'server' workspaces for Enterprise Architect Version Control, as 'local' workspaces do not support the application of checkout **locks** to files. Enterprise Architect relies on the presence of checkout locks to ensure that Packages can only be checked out exclusively and that a given Package is not already checked-out in some other project (for instance, in a Private Model deployment). This is necessary because it is not practical to merge the XMI Package files that Enterprise Architect uses for Version Control. A single TFS workspace can map many different local folders, each one to a separate Source Control Folder. In this case, TFS can take a long time to work through and update the files in all of those folders, and the system might appear to 'freeze' whilst it waits for TFS to hand back program control. You can avoid this if you keep your Version Controlled Package files in a folder that is separate from other Artifacts, such as source code files, creating a separate work space to use just for your Package files, or creating and mapping a separate folder for Package files within an existing workspace.
Configure exclusive check-outs	The XMI format files used for the **Version Control** of Enterprise Architect's Packages can not be merged as if they were ordinary text files. Therefore, Enterprise Architect must enforce serialized editing of its Version Controlled

	Packages. As a consequence, it is important that TFS is configured to use 'exclusive checkouts' for XML files.
Verify the TFS Workspace	Enterprise Architect uses the TFS command line client to checkin and checkout files from the TFS repository. After creating the TFS workspace, it is important to verify that the command line client can be used to add, check-in and check-out files residing in the working copy folder that is mapped through this workspace.

Notes

- TFS can also be used with an SCC client; the MS TFS-SCC client is available for download from Microsoft's web site
- MDG Integration for Visual Studio 2005 or 2008 enhances TFS support by providing access to, for example, work items and bugs within both Enterprise Architect and the MDG Integration product

TFS Workspaces

In order to use TFS to provide **Version Control** of the Packages in a model, you prepare a functional TFS workspace that can be accessed through an Enterprise Architect Version Control configuration within that model. You use the TFS workspace to map a local working folder on your PC to a Source Control Folder within a TFS project repository.

It is assumed that have TFS 2013 open and a TFS Team Project already exists for you to use in Version Control of the Packages of your Enterprise Architect project. You can create the TFS workspace through MS Visual Studio.

TFS 2013 and later versions support the use of Local workspaces. However, Local workspaces do not support the application of checkout **locks**. For this reason, Sparx Systems strongly advises against the use of Local workspaces when Version Controlling your Enterprise Architect Package files. Using Local workspaces carries a high risk of merge conflicts when checking in, which would almost certainly result in loss of data or a corrupted model database.

Map a local folder to a TFS Source Control Folder

Step	Action
1	Connect to your TFS server. From the TFS main menu, choose 'View | Team Explorer'. In the Team Explorer toolbar, click on the **Connect to Team Projects button** (fourth from left, with a power plug icon). The 'Team Explorer Connect' page displays.
2	Click on the link 'Select Team Projects...' The 'Connect to Team Foundation Server' dialog displays.
3	Click on the appropriate Team Foundation Server, Team Project Collection and Team Project, then click on the Connect button.
4	Click on the Home page of the Team Explorer, and then on the **Source Control Explorer** button. The Source Control Explorer view displays.
5	In the **Source Control Explorer**, click on the **drop-down arrow** in the 'Workspace' field in the toolbar, then click on 'Workspaces' at the bottom of the list. The 'Manage Workspaces' dialog displays.
6	Click on the **Add button**. The 'Add Workspace' dialog displays.
7	Click on the **Advanced button**. A number of new fields display on the dialog.
8	Type in an appropriate name for the new workspace and, if required, type in a comment.
9	Set the 'Location' field to 'Server'. This setting is important. This setting is not available in TFS 2010 and earlier releases, where all workspaces are Server based.
10	Click in the Source Control Folder column, then click on the **Browse button** to select a Source Control Folder. Select the Source Control folder (in the Team Project) to use for controlling Enterprise Architect

	Packages.
11	Click on the **Browse button** in the 'Local Folder' column and create a new local folder. This is the working copy folder into which Enterprise Architect exports the Package files.
12	Click on the **OK button**. The new workspace is created and saved. The 'Add Workspace' dialog closes.
13	Click on the **OK button**. The 'Manage Workspaces' dialog closes.

Notes

- The local folder referenced in step 11 is the Working Copy Path that should be specified when defining an Enterprise Architect **Version Control** Configuration to use this TFS workspace

TFS Exclusive Check Outs

The XMI format files used for the **Version Control** of Enterprise Architect's Packages can not be merged as ordinary text files can. Therefore Enterprise Architect must enforce serialized editing of its Version Controlled Packages, and it is important that Team Foundation Server is configured to use 'exclusive checkouts' for XML files. Otherwise, TFS can return file statuses that make it look as if the Package file is not checked-out by another user when indeed it is.

You set exclusive checkouts in the TFS workspace through MS Visual Studio.

Configure TFS to enforce exclusive check outs for XML files

Step	Action
1	Using Visual Studio, from the main menu select 'View | Team Explorer'.
2	In the 'Team Explorer' pane, right-click on the TFS Server name that is controlling the Enterprise Architect Package files, then from the context menu select 'Team Foundation Server Settings | Source Control File Types'.
3	Select the entry for XML files (or create an entry if necessary) then click on the **Edit button**.
4	Deselect the 'Enable file merging and multiple check out' checkbox.

Verify the TFS Workspace

Enterprise Architect will be using the TFS command line client to commit file updates from the local working copy folder into the TFS repository. After creating the TFS workspace that maps your local working copy folder to a TFS repository folder, it is good practice to verify that the TFS workspace functions correctly before attempting to use it with Enterprise Architect. You need to be able to add files to TFS, place a check-out lock on the files and check-in changes to those files.

Verify correct operation of a TFS Workspace

Step	Action
1	Use Windows to open a command prompt window.
2	Select the directory you specified as the local working copy folder when creating the TFS workspace. For example: C:\> cd myTFSWorkingCopyFolder
3	Create a test file, such as *Test.xml*, containing the text '*my TFS Test*'. For example: C:\> echo my TFS Test > Test.xml (The filename extension '.xml' is used because the TFS server should have been configured to allow exclusive checkouts for XML files.)
4	Using the command line client, execute each of these TFS commands: • tf.exe add Test.xml • tf.exe checkin Test.xml /override:"Manual check in." /comment:"a message" • tf.exe checkout /lock:checkout Test.xml Use your preferred editor to modify the file and save the changes. • tf.exe checkin Test.xml /override:"Manual check in." /comment:"a message" • tf.exe get . /recursive

Notes

- All of the commands in step 4 should execute without any errors; in particular, you should be able to apply a checkout lock when checking out the file
- If the checkout command reports any errors, you might have to reconfigure the TFS workspace as a server based workspace, or you might have to configure the TFS Server to use 'exclusive checkouts' for XML files

Create a CVS Environment

You can use Concurrent Versions System (CVS) as a **Version Control** provider for Enterprise Architect. The first step in doing this is for a CVS administrator to install and configure the appropriate software. A number of basic tasks are performed in creating an operational CVS environment, and useful tools are available for performing some of these tasks.

Tasks in Creating a CVS Environment

Task	Detail
Install server components	Executable files for CVS can be obtained from March Hare Software. CVS server components are available to run on a wide range of different hardware and operating systems; Enterprise Architect is not affected by your choice of server components.
Create a repository	Please consult the official CVS documentation.
Create CVS users	Please consult the official CVS documentation.
Create a new repository module	It is recommended good practice to create a new repository module in CVS for each new Enterprise Architect model being added to **Version Control** with CVS. Users should create a new local working copy folder from the module to be used with that model. A repository module represents a project, or a set of related files in the repository. TortoiseCVS can greatly simplify the process of creating new repository sub-trees.
Install client components	Executable files for CVS can be obtained from March Hare Software. Enterprise Architect is a Windows based application - it requires a Windows based CVS command line client for integration.
Create a working copy folder	A working copy folder must exist on each users' machine, for Enterprise Architect to use when exporting and importing the Version Controlled Package files. It is this folder that is specified as the Local Project Path, when defining your **Version Control** Configurations. The working copy folder is the 'sandbox' where you modify the controlled files. The working copy folder is usually associated with a folder that exists within the Version Control repository. In CVS, to create a local working copy you perform an initial check-out of a folder from the CVS repository; this downloads a copy of the folder and its contents, to create your local working copy. TortoiseCVS can greatly simplify the initial check out of a working copy folder.
Setting up CVS under Wine/CrossOver	The process of setting up and using CVS with Enterprise Architect under Wine is almost identical to the process when running natively under Windows, apart from minor differences in installing the CVS client and performing the initial checkout of the working copy folder.

Notes

- If you do not already use CVS for **Version Control**, you should consider using Subversion instead; Subversion's client-server protocols provide a broader range of possibilities for connecting to remote servers, with easier set up of secure connections
- TortoiseCVS is a Windows shell extension; Enterprise Architect cannot use TortoiseCVS as its client, it must use the CVS command line client

Prepare a CVS Local Workspace

In order to use CVS to provide **Version Control** of the Packages in a model, you need to prepare a functional CVS working copy folder that can be accessed through an Enterprise Architect Version Control configuration within that model.

Prepare a CVS Working Copy Folder

Step	Action
1	Ask your System Administrator to install CVS and create a remote repository, with a module that you can use to control your Enterprise Architect Package files. Your administrator must create a username and password for you before you can make a connection.
2	Select or create a suitable directory to use as your CVS working copy directory.
3	Open a command prompt window and navigate to your CVS working copy directory. For example: C:\> mkdir myCVSWorkSpace C:\> cd myCVSWorkSpace
4	Log in to the remote CVS repository. For example: C:\myCVSWorkSpace> cvs -d:pserver:myUserID@ServerName:/reposPath login Replace *myUserID* with your CVS username, replace *ServerName* with the name of your CVS server and replace *reposPath* with the path to the repository on the server. A prompt for a password displays.
5	Enter your password. You are logged in to the CVS server.
6	Perform the initial checkout of the CVS repository module, into the local working copy directory. For example: C:\myCVSWorkSpace> cvs -d:pserver:myUserID@ServerName:/reposPath checkout moduleName (Replace *moduleName* with the name of the repository module that you want to check out.) A subdirectory is created in your current working directory, with the same name as the module being checked out. Any files already existing in the repository module are downloaded to the working copy folder as working copy files.
7	Verify that your CVS environment functions correctly.

Notes

- Much of the process can also be performed (more simply) using the TortoiseCVS command 'Make New Module'

Verify the CVS Workspace

After creating the CVS local working copy to hold the working copies of your Package files, you can verify that it functions correctly before attempting to use it with Enterprise Architect. You need to be able to add files to CVS, and commit changes to those files. You also need to be able to register as an editor of the file and retrieve the list of currently registered editors.

Verify correct operation of a CVS working copy folder

Step	Action
1	Use Windows to open a command prompt window.
2	Select the directory you specified as the working copy in the cvs checkout command, when preparing the CVS workspace. For example: C:\> cd myCVSWorkSpace
3	Create a test file, such as *Test.txt*, containing the text *CVS Test*. For example: C:\> echo CVS Test > Test.txt
4	Execute these CVS commands: • cvs add Test.txt • cvs commit -m"Commit comment" Test.txt • cvs update Test.txt • cvs edit Test.txt • cvs editors Test.txt The commands should execute without any errors and without prompting the user for any extra input. The editors command should produce output that resembles this: Test1.txt myUserID Tue Aug 9 10:08:43 2009 GMT myComputer C:\myCVSWorkSpace\moduleName
5	Take note of the userID that follows the filename. Enterprise Architect must find and use this user ID when you create your **Version Control** configuration.

Notes

- Your environment must be set up such that you can perform these operations without ever being prompted for input, such as user ID or password

TortoiseCVS

TortoiseCVS is a Windows shell extension for CVS; it provides icon overlays in Windows Explorer that are useful as a tool for observing the status of your CVS controlled files. You can also use it to create your repository modules and check out local working copies from within Windows Explorer using simple menu commands.

You can download TortoiseCVS from: http://www.tortoisecvs.org.

Notes

- Enterprise Architect must use the CVS command line client to communicate with the CVS server; it cannot use TortoiseCVS

Create an SCC Environment

You can use a Microsoft Common Source Code Control (SCC) compatible product as a **Version Control** provider for Enterprise Architect. The first step in doing this is for an administrator to install and configure the server and client applications. A number of basic tasks are performed in creating an operational SCC-based environment.

Tasks in Creating an SCC Environment

Task	Detail
Install and configure your chosen Version Control product	A **Version Control** server component is typically installed on a dedicated server machine. All Enterprise Architect users who require access to Version Control must be able to connect to the server machine. After installing the Version Control software, the administrator should also create Version Control user IDs for all users who require access to Version Control. For further information, consult the product documentation for your particular Version Control product.
Create a new SCC project	It is good practice to create a new SCC **Version Control** project, or least a new folder within a project, for each Enterprise Architect project being added to Version Control with SCC. If you have a single Enterprise Architect project that contains many different models (for example, a DBMS hosted project with multiple model root nodes), you might choose to create a new SCC Version Control project for each separate model. For further information, consult the product documentation for your particular Version Control product.
Configure your SCC project to support exclusive check-outs for .XML files	The XMI-format files used for the **Version Control** of Enterprise Architect Packages can not be merged in the same way as ordinary text files can. Therefore, Enterprise Architect must enforce serialized editing of its Version Controlled Packages. As a consequence, it is important that your Version Control application is configured to use 'exclusive checkouts' for XML files.
Create a local working copy folder	A working copy folder must exist on each users' machine, for Enterprise Architect to use when exporting and importing the Version Controlled Package files. It is this folder that is specified as the Local Project Path, when defining your **Version Control** Configurations. The working copy folder is the 'sandbox' where you modify the controlled files. The working copy folder is usually associated with a folder that exists within the Version Control repository. Your Version Control product provides some means by which you associate a working copy folder with a repository folder. For further information, consult the documentation for your particular Version Control product.

Notes

- When installing the client component software on users' PCs, check that the SCC client is also installed, as it might not be a part of the default installation

Upgrade at Enterprise Architect Version 4.5, Under SCC Version Control

If you are working in Enterprise Architect release 4.5 or later and you open an SCC Version Controlled project that was created under a release earlier than 4.5, you must identify the SCC connection with a new unique ID. You can assign a name to the existing SCC configuration or associate the project with another configuration that has previously been assigned a unique ID.

By having a unique ID for each **Version Control** configuration, you can assign a configuration quickly and efficiently using configurations that have been created previously for other Version Controlled repositories. This enables you to configure the many Packages to use an existing Version Control repository; this can apply to Packages created for more than just one model enabling a great deal of flexibility.

Upgrade an existing SCC Version Controlled project created before release 4.5, in Enterprise Architect release 4.5 or later

Step	Action
1	Open the project that has an SCC **Version Control** configuration created in Enterprise Architect earlier than version 4.5. The 'Select or Create Unique ID for Version Control' dialog automatically displays.
2	On the dialog, either create an ID for an existing configuration or choose a previously created one from the 'Unique ID' drop-down list.
3	The existing SCC configuration is the initial value, represented by SCC-XXXXX; this number is not especially meaningful, therefore it is recommended that the configuration be given a meaningful name.
4	You can associate the Version Controlled Package with a previously-defined configuration by selecting an existing configuration from the 'Unique ID' drop-down list (if one exists).
5	After you have assigned the unique ID, click on the **OK button** to load the model.

Version Control Configuration

Once you or an Administrator have installed and configured the **Version Control** product software, to start using Version Control you must first define a Version Control configuration within your project in Enterprise Architect, to be used to control the Packages in the project. You can define any number of Version Control configurations in a single model, but any given Package is associated with only one configuration.

Access

Ribbon	Configure > **Version Control** > Project-VC
Context Menu	Right-click on Package > **Package Control** > **Version Control** Settings

Define Version Control Configuration

Step	Action
1	On the '**Version Control** Settings' dialog, click on the **New button**.
2	In the 'Unique ID' field, type a suitable configuration name.
3	Select the Type of **Version Control** product you are connecting to, by clicking on the corresponding radio button.
4	At this point, the middle section of the dialog changes to display a collection of fields specific to the type of **Version Control** configuration you are defining. Enter details relating to the Version Control workspace that this configuration is to use.
5	Click on the **Save button**. The new configuration is added to the Defined Configurations list.
6	If you want to create another **Version Control** configuration, return to step 1. When you have finished defining your Version Control configurations, click on the **Close button**.

Notes

- **Version Control** Configuration details are stored in the user's Windows Registry settings, but each project stores a list of the configurations it uses, so that Version Control connections can be initialized as the project is being loaded
- If you are using the Corporate or extended editions of Enterprise Architect with security enabled, the administrator must also set up access permissions to configure and use Version Control

Version Control Settings

Defining **Version Control** settings within Enterprise Architect should only be performed after you have successfully created the Version Control environment that you intend to use to control your Package files. If you have not yet done so, please refer to the Help topic *Applying Version Control in a Team Environment* for a summary of what is required.

To set up a Version Control configuration on your model, or update an existing Version Control configuration, you define a number of settings that control how the status of your model is communicated to your Version Control system. You define these settings using the 'Version Control Settings' dialog.

Access

Ribbon	Configure > **Version Control** > Project-VC
Context Menu	Right-click on Package > **Package Control** > **Version Control** Settings

Configuration Options

Option	Action
This model is private	Select to specify that this model database is to be accessed by just a single user (Private Model). Leave unselected (the default) or deselect to specify that the database is to be accessed by multiple concurrent users (Shared Model). If in doubt, use the default setting.
Save nested Version Controlled Packages to stubs only	Select to specify that the exported XMI file for a Version Controlled Package will contain Package stubs (place holders) for nested Version Controlled child Packages (recommended). Deselect to specify that the exported XMI file will contain the full content of nested Version Controlled child Packages.
For all Packages, create placeholders for external references	Select to force all XMI 1.1 imports across the model to exclude incoming relationships and instead create external references. If the 'Create placeholders for missing External References during XMI 1.1/2.1 Import' checkbox is not selected in the XML Specifications options for a user, this field overrides that setting. The *XML Specifications* Help topic provides an example of the benefits of selecting this option.
Unique ID	Specify a name that uniquely identifies the configuration. Either: • Type a name to identify a new configuration, or • Click on the **drop-down arrow** and select the name of a configuration previously defined in a different project (if any exist)
Type	Click on the appropriate radio button for the type of **Version Control** system you are associating with this configuration.

	The middle section of the dialog changes to display a collection of fields relating to the type of Version Control configuration you are defining. Set the type to SCC for: • MS Visual Source Safe • Rational Clear Case • Perforce • AccuRev • Any other SCC-compatible clients For any other product that you are using, select the type that matches the product - CVS, Subversion or TFS.
New	Click on this button to clear the fields and create a new **Version Control** configuration.
Save	Click on this button to save the details of a new or updated configuration.
Delete	Click on an entry in the Defined Configurations list and click this button to remove the definition of the selected configuration from this model.
Defined Configurations	Review a list of configurations that are in use in the current model.
In future, do not prompt for incomplete configurations	Select to specify that the user is not prompted to complete the definition of configurations that are not fully specified (the default). Deselect to prompt the user to complete configurations that are not fully defined.
Close	Close the '**Version Control** Settings' dialog.
Help	Display this Help topic.

Notes

- It is important that, for any given Version Controlled Package file, any user accessing that file from any model uses **Version Control** configurations having the same Unique ID
- When you first open a model that was created by another user and that uses Version Control, the Version Control configuration(s) used by that model do not yet exist in your Windows registry settings; you have to complete the definitions of those configurations before you can use Version Control in that project
- If User Security is enabled, you must have 'Configure Version Control' permission to set up Version Control options for the current model
- It is possible to use multiple Version Control configurations in the same model

SCC Settings

When you are setting up your **Version Control** configurations on the 'Version Control Settings' dialog, and you set the configuration type to 'SCC', the dialog presents a set of fields specific to SCC-based configurations. You can then define details such as:

- The working copy folder to be used with the configuration
- The details necessary to connect to the SCC Version Control system

You set the Version Control configuration type to SCC for version Control providers such as:

- MS Visual Source Safe
- Rational Clear Case
- Perforce
- AccuRev
- Any other SCC-compatible clients

Access

Ribbon	Configure > **Version Control** > Project-VC : Type: SCC
Context Menu	Right-click on Package > **Package Control** > **Version Control** Settings: Type: SCC

Settings

Option	Action
Local Project Path	Displays the full path of the folder that contains the local (working) copies of the XMI Package files. This field is read-only; its value can only be set through use of the **Select Path button**.
Select Path	Click on this button to choose the Local Project Path, by opening a file browser dialog and navigating through the file system to the appropriate folder. • After you choose the appropriate folder path, the 'Select SCC Provider' dialog displays, listing all SCC providers that are installed on the current workstation; choose the SCC provider to use and click on the **OK button** • At this point, the SCC client opens its own dialog to prompt you for information; SCC products implement this functionality in varied ways, but typically you are prompted to log in to the **Version Control** system, then prompted to choose the SCC project to use and possibly a (server) folder contained within that project • At the conclusion of this process, all of the SCC details should be filled in; you can then save the definition by clicking on the **Save button** on the 'Version Control Settings' dialog
Current User	Displays the user name used to log on to the **Version Control** system that is

	accessed through this configuration. This field is read-only; the value it displays is retrieved from the SCC client.
SCC Provider	Displays the name of the SCC provider. This field is read-only; the value it displays is retrieved from the SCC client.
SCC Project	Displays the name of the SCC Project that this configuration attaches to. This field is read-only; the value it displays is retrieved from the SCC client.

Notes

- You define the SCC-specific details as part of the broader process of setting up a **Version Control** Configuration on the 'Version Control Settings' dialog

CVS Settings

When you are setting up your **Version Control** configurations on the 'Version Control Settings' dialog, and you set the configuration type to CVS, the dialog presents a set of fields specific to CVS-based configurations. You can then define details such as:

- The working copy folder to be used with the configuration
- The path to the CVS command line client

Access

Ribbon	Configure > **Version Control** > Project-VC : Type: CVS
Context Menu	Right-click on Package > **Package Control** > **Version Control** Settings: Type: CVS

Settings

Option	Action
Working Copy Path	Displays the full path of the folder that contains the local (working) copies of the XMI Package files. This field is read-only; its value can only be set through use of the **Select Path button**.
Select Path	Click on this button to choose the working copy path, by opening a file browser dialog and navigating through the file system to the appropriate folder.
Current User	This field is read-only; its value is retrieved from a file named CVS\Root, located in the folder selected using the **Select Path button**.
CVS Exe Path	Displays the full path of the CVS command line client executable file. This field is read-only; its value can only be set through use of the **Select Path button**.
Select Path	Click on this button to specify the path to the CVS command line client, by opening a file browser dialog and navigating through the file system to locate the appropriate file.
VC Time-Out Value	Specify the amount of time that Enterprise Architect waits for a CVS command to complete; if the command does not complete within the allowed time, the system displays a Time-out error message, detailing the command that failed to complete. This single time-out value is applied to all **Version Control** Configurations (of types SVN, TFS and CVS) that the user accesses from this workstation.

Notes

- When connecting to a remote CVS repository, the 'Current User' field should display the user name used to log into that repository; if this does not happen, it indicates that Enterprise Architect cannot extract the user name from the file ...\WorkingCopyPath\CVS\Root and the configuration does not work correctly
- You define the CVS-specific details as part of the broader process of setting up a **Version Control** configuration on the 'Version Control Settings' dialog

SVN Settings

When you are setting up your **Version Control** configurations on the 'Version Control Settings' dialog, and you set the configuration type to 'Subversion', the dialog presents a set of fields specific to Subversion-based configurations. You can then define details such as:

- The working copy folder to be used with the configuration
- The path to the Subversion command line client

Access

Ribbon	Configure > **Version Control** > Project-VC : Type: Subversion
Context Menu	Right-click on Package > **Package Control** > **Version Control** Settings: Type: Subversion

Settings

Option	Action
Working Copy Path	Displays the full path of the folder that contains the local (working) copies of the XMI Package files. This field is read-only; its value can only be set through use of the **Select Path button**.
Select Path	Click on this button to choose the Working Copy Path, by opening a file browser dialog and navigating through the file system to the appropriate folder.
Subversion Exe Path	Displays the full path of the Subversion command line client executable file. This field is read-only; its value can only be set through use of the associated **Select Path button**.
Select Path	Click on this button to specify the path to the Subversion command line client, by opening a file browser dialog and navigating through the file system to locate the appropriate file.
VC Time-Out Value	Specify the amount of time that Enterprise Architect should wait for a Subversion command to complete; if the command does not complete within the allowed time, the system displays a Time-out error message, detailing the command that failed to complete. This single time-out value is applied to all **Version Control** Configurations (of types SVN, TFS and CVS) that the user accesses from this workstation.

Notes

- You define the Subversion-specific details as part of the broader process of setting up a **Version Control** configuration on the 'Version Control Settings' dialog

TFS Settings

When you are setting up your **Version Control** configurations on the 'Version Control Settings' dialog, and you set the configuration type to 'TFS', the dialog presents a set of fields specific to TFS-based configurations. You can then define details such as:

- The working copy folder to be used with the configuration
- The user name and password to log in to the TFS server
- The path to the TFS command line client

Access

Ribbon	Configure > **Version Control** > Project-VC: Type: TFS
Context Menu	Right-click on Package > **Package Control** > **Version Control** Settings: Type: TFS

Settings

Option	Action
Working Copy Path	Displays the full path of the folder that contains the local (working) copies of the XMI Package files. This field is read-only; its value can only be set through use of the associated **Select Path button**.
Select Path	Click on this button to choose the Working Copy Path. The file browser dialog opens, through which you navigate through the file system to the appropriate folder.
Server Name	Displays the name of the TFS Server that is associated with the working copy folder specified in the 'Working Copy Path' field. This field is read-only; Enterprise Architect retrieves the value it displays by querying the TFS client.
Workspace Name	Displays the name of the TFS Workspace that is associated with the working copy folder specified in the 'Working Copy Path' field. This field is read-only; Enterprise Architect retrieves the value it displays by querying the TFS client.
User Name	Type in the user name with which to log into the TFS Server.
Password	Type in the password with which to log into the TFS Server.
Display Name	TFS 2013 and later releases use a Display Name to report on who owns a checkout lock on a file. TFS 2010 uses AccountID when reporting lock owners and so does not require a Display Name. If using TFS 2013, type in your TFS Display Name as shown in the User column of

	the Visual Studio **Source Control Explorer** when you checkout a file.
Checkout Locks Required	This checkbox defaults to selected. TFS 2013 supports the use of Local workspaces, but these do not support checkout **locks**. If you want to use TFS Local workspaces (not recommended), you must deselect this checkbox. Otherwise, leave the checkbox selected. It is recommended that all workspaces used for Version Controlling Enterprise Architect Package files are set as 'Server' workspaces.
TFS Exe Path	Displays the full path of the TFS command line client executable file. This field is read-only; its value can only be set through use of the associated **Select Path button**.
Select Path	Click on this button to specify the path to the TFS command line client. The file browser dialog opens, through which you navigate through the file system to the appropriate folder.
VC Time-Out Value	Type in the number of seconds that Enterprise Architect should wait for a TFS command to complete; if a command does not complete within this allowed time, the system displays a Time-out error message, detailing the command that failed to complete. This single time-out value is applied to all **Version Control** configurations (of types SVN, TFS and CVS) that the user accesses from this workstation.

Notes

- If you automatically log in to TFS through a path external to Enterprise Architect (for example, through MS Integrated Security), you can leave the 'User Name' and 'Password' fields blank
- If the 'Password' field is blank, Enterprise Architect retrieves your Windows username and uses that value when determining whether a Package file is checked out to you or to another user
- TFS **Version Control** can also be accessed using the TFS MSSCCI client; to make use of the TFS MSSCCI client, please define an SCC based Version Control configuration
- You define the TFS-specific details as part of the broader process of setting up a Version Control configuration on the 'Version Control Settings' dialog

Re-use an Existing Configuration

Once a **Version Control** configuration has been defined for use in one project, it is possible to re-use that configuration in other projects to provide access to:

- An already existing Version Control environment (a working copy directory and its associated repository that is already in use)
- Version Controlled Packages that were created (and Version Controlled) in another project

Access

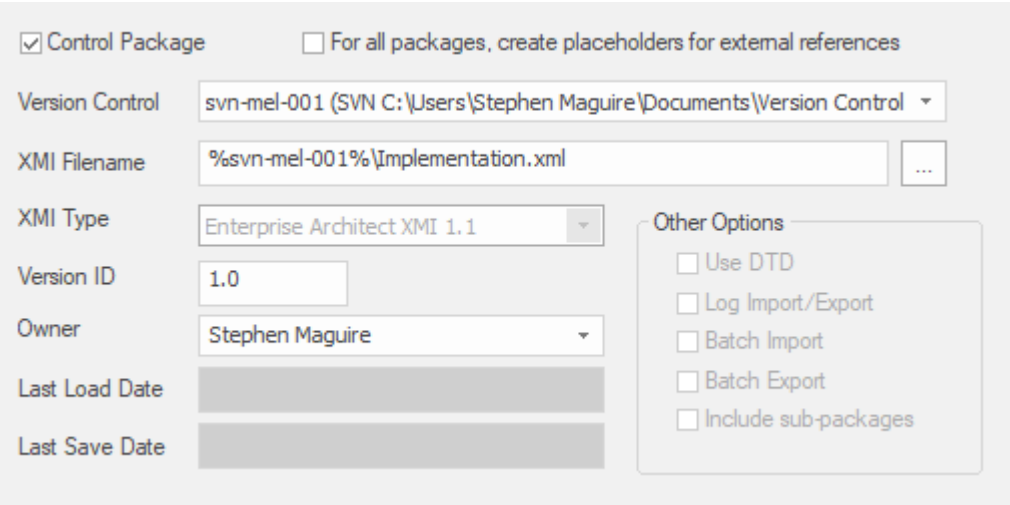
Ribbon	Configure > **Version Control** > Project-VC
Context Menu	**Browser window** | right-click on Package | **Package Control** | **Version Control** Settings

Select existing configuration

Step	Action
1	On the '**Version Control** Settings' dialog, click on the **New button**. The various fields on the dialog are cleared, ready for data entry.
2	In the 'Unique ID' field, click on the **drop-down arrow** and select one of the previously-defined **Version Control** configurations. The details of the selected configuration are displayed in the dialog.
3	Click on the **Save button**. The configuration details are saved and are ready for use in the current project.

Applying to Packages

Version Control in Enterprise Architect operates at a Package level. Any Package in the repository can be put under Version Control and assigned to any Version Control Configuration. This means that different Packages can be controlled by different Version Control systems.



The **Package Control** Options window showing the setting for the selected Package.

Configure Controlled Package

Once your **Version Control** application is set up and you have Version Control configurations in place, you can place the individual Packages in your model under Version Control. To put a Package under Version Control, you:

- Flag the Package as a controlled Package
- Specify the Version Control configuration to control it and
- Associate an XMI file with the Package

You can then export and import the Package data to and from the file and issue commands to the Version Control system.

Access

Ribbon	Configure > **Version Control** > Package-VC
Keyboard Shortcuts	**Ctrl+Alt+P**

Apply Version Control to a single Package

Step	Action
1	Click on the 'Control Package' checkbox to select it, indicating that this Package is to be controlled.
2	Click on the '**Version Control' drop-down arrow** and select the Version Control configuration to be used to control this Package.
3	The 'XMI Filename' field displays a default filename for the Package export file, based on the Package name. Optionally, modify the filename. Either: • Type a new name, or • Click on the  button to open a file selection dialog The target file must be located within the working copy folder of the selected **Version Control** configuration, or one of its sub-folders.
4	The 'Version ID' field defaults to '1.0'. Optionally, change this to a different version number.
5	The 'Owner' field defaults to your user name. Optionally, type or select the name of the user who actually owns the Package.
6	Click on the **OK button**. The 'Add Package to **Version Control**' dialog displays.
7	Optionally, clear the 'Keep checked out' checkbox.

	After applying **Version Control**, the Package either remains checked-out for editing, or is checked-in and locked against editing, depending on this setting.
8	Click on the **OK button**. The 'Add Comment' dialog displays.
9	Optionally, add any further comments to the default comment. Enterprise Architect provides a default comment that includes the current date & time.
10	Click on the **OK button**. The current Package is exported to the nominated XMI file, which is then committed to **Version Control**. The Package icon in the **Browser window** is updated to reflect the Package's Version Control status.

Notes

- If you are using the Corporate or extended editions of Enterprise Architect with security enabled, these features are only available to users who have been granted permission to configure and use **Version Control**

Browser Window Indicators

Packages under **Version Control** are identified in the **Browser window** by icons that indicate the current status of the Package.

Indicators

Icon	Indicates that
	This Package is Version Controlled and not checked out to you. You cannot edit the Package (until you check out the Package yourself).
	This Package is Version Controlled and checked out to you. You can edit the Package.
	This Package is Version Controlled, but you checked it out whilst not connected to the **Version Control** server. You can edit the Package but there could be version conflicts when you check the Package in again.
	This Package is controlled and is represented by an XMI file in the file management system, but it is not under **Version Control**. You can edit this Package.

Apply Version Control To Branches

It is possible to apply **Version Control** to all Packages within a selected model branch, in a single operation. In this context, a model branch is a Package that is currently selected in the **Browser window**, and all of the Packages contained within it.

Access

Context Menu	**Browser window** | right-click on Package | **Package Control** | Add Branch to **Version Control**

Apply Version Control to all Packages within a selected model branch

Step	Action
1	On the 'Apply VC to Branch' dialog, click on the **drop-down arrow** in the '**Version Control** Configuration' field and select the configuration to use.
2	Optionally, tick the checkbox 'Export as Model Branch'. Once the **Version Control** operation is complete a Model Branch file (.EAB file) is created for this branch.
3	Click on the **OK button**. The system creates a number of sub-folders within the **Version Control** working copy folder, then exports all of the Packages within the selected model branch. The system generates filenames for the XMI files, based on the Package GUIDs.

Notes

- The **Version Control** configuration to be used in this operation must be defined within the model before selecting this command
- When invoked on the model root node, this command applies Version Control to every Package within the model

Fundamental Usage

Once your **Version Control** product is installed and you have created a suitable environment, Enterprise Architect can make use of that environment to control the Packages in your project. The fundamental Version Control functions are outlined in this table.

Fundamental Functions

Facility	Detail
Define Version Control Settings	**Version Control** Configurations are used by Enterprise Architect to communicate with your Version Control system. You define one or more Version Control configurations in your project and then use those configurations to control the Packages in your project.
Configure a Package	To put a Package under **Version Control** you mark the Package as a controlled Package, specify the Version Control configuration to control it, and associate an XMI file with the Package.
Check Out a Package	Checks out the Version Controlled Package currently selected in the **Browser window**, so that you can update modeling objects within it.
Undo Check Out of a Package	Reverses the check-out of a Package, discarding any modifications that have been made by restoring the Package content to the latest revision held in **Version Control**.
Check In a Package	Checks in the Package currently selected in the **Browser window**. Checking-in updates the reference version of a Package or group of Packages in the model.
Check In a Model Branch	Checks in all Packages involved in a particular unit of work, as a single operation. Checking-in updates the reference version of a Package or group of Packages in the model.
Check Out a Model Branch	Checks out all Packages within a selected model branch as a single operation, so that you can update modeling objects within them.
Import a Package From Version Control	Retrieves Packages from **Version Control** that have been created by other users, or by you in another model, and imports them into your current model.
Apply Version Control to a Model Branch	Applies **Version Control** to all Packages within the selected model branch, in a single operation. In this context, a model branch is a Package in the **Browser window**, and all of the Packages contained within it.
View Package Revision History	Displays the change history of Version Controlled Packages. You can also check out a prior revision of the Package for editing, effectively rolling-back to a prior revision of the Package.

Package Version Control Options

When you have set up a Package for **Version Control**, you gain access to a range of Version Control operations that can be performed on that Package, such as:

- Open the dialog for working with baselines of the Package
- Check-in and check-out single Packages or a selected hierarchy of Packages
- Update Packages to the latest revision from the Version Control repository
- Inspect the revision history or properties of the XMI file associated with a Package
- Revert a Package to a previous revision
- Compare the current model content of a Package, against the latest revision of the Package in Version Control
- Import and export hierarchies of Packages (model branches) to and from the model, through the Version Control system
- Synchronize the status of a Package, with the Version Control system

Access

Context Menu	Right-click on Version Controlled Package > **Package Control**

Options

Option	Action
Check In Branch	Check-in Packages contained in the currently selected model branch (that is, the selected Package and all of its child Packages). The 'Select Packages to Check In' dialog lists all Version Controlled Packages within that branch that are checked out to you; you can then select Packages in the displayed list, to be submitted for check-in. You can also choose to keep the Packages checked-out after committing a new revision to **Version Control**.
Check Out Branch	Recursively check out all Packages contained within the currently selected model branch (that is, the selected Package and all of its child Packages) that are Version Controlled and checked-in.
Check In	Commit a new revision of the currently selected Package to the **Version Control** repository and lock the Package against further editing. Only available for Packages that you have checked-out yourself.
Check Out	Synchronize the currently selected Package with the latest revision from the **Version Control** repository and unlock the Package to allow editing. Only available for Packages that are not already checked-out (and whose associated Package file is not checked-out).
Undo Check Out	Restore the selected Package to the latest revision in the **Version Control**

	repository and lock the Package against further editing.
Put Latest	Commit a new revision of the currently selected Package to the **Version Control** system, while keeping the Package checked-out. This is equivalent to checking a Package in and immediately checking it back out again. Only available for Packages that you have checked-out yourself.
Get Latest	Synchronize the currently selected Package with the latest revision from the **Version Control** repository. Available only for Packages that are checked in.
Get All Latest	Updates all of the Version Controlled Packages in the project, to the latest revision retrieved from **Version Control**. Only updates Packages that are currently checked in. Once the latest revisions are retrieved, the system scans all the controlled Packages and fixes any missing cross-references by comparing the Package with its XMI 1.1 file. If the cross-reference information in the XMI does not match the model, the system updates the model with the information from the XMI and records this update in the **System Output** window. You can roll back such updates by selecting the entry in the System Output window and using the context menu option 'Rollback Update' (or 'Rollback Selected Updates' if multiple entries are selected). • Closing the model clears the entries in the System Output window • An entry in the System Output window is also cleared as and when you roll-back the update for it
Scan XMI and Reconcile Model	Scan the Package XMI files associated with each of the project's controlled Packages and restore any diagram objects or cross-references that are detected as missing from the project. This function is useful in team environments where each user maintains their own private copy of the model database (that is, multiple private project files) and model updates are propagated through the use of controlled Packages. It provides no benefit when the model is hosted in a single shared database that is accessed by all team members. Each controlled Package is compared with its associated XMI file and, if the cross-reference information in the model does not match the XMI, the system updates the model with the information from the XMI and records the update in the **System Output** window. You can roll back such updates by right-clicking on the entry in the System Output window and selecting the 'Rollback Update' option (or 'Rollback Selected Updates' if multiple entries are selected). Closing the model clears the entries in the System Output window; an entry in the window is also cleared as and when you roll-back the update for it. This functionality is invoked automatically as part of the 'Get All Latest' operation. When working in an environment that uses a Private Model deployment and your model contains a significant number of cross-Package references, it is recommended that you invoke 'Scan XMI and Reconcile Model' from time to time, following the re-importation of controlled Packages - for example, after using 'Get Latest' to update a number of Packages - or after performing a number of Package check-outs. As a general rule, avoid running 'Scan XMI and Reconcile Model' while you have

	uncommitted changes in your model; generally, you: • Check-out a number of Packages • Invoke 'Scan XMI and Reconcile Model' • Make your modifications • Commit any outstanding changes before you check-out more Packages and run 'Scan XMI and Reconcile Model' again
File Properties	Display **Version Control** properties pertaining to the XMI export file associated with the currently selected Package; this also identifies who has checked out the Package.
File History	Display change history information for the currently selected Package. Revert to or check-out a prior revision of the Package.
Compare with Controlled Version	Compare the currently selected Package with the latest revision of its associated XMI file retrieved from **Version Control**.
Add Branch to Version Control	Apply **Version Control** to all Packages within a selected model branch, in a single operation. In this context, a model branch is a Package that is currently selected in the **Browser window**, and all of the Packages contained within it.
Export as Model Branch	Export a newly created model branch from your own private copy of a model.
Import a Model Branch	Retrieve a model branch and import it into either the source model or another model.
Get Package	Access Packages in the **Version Control** repository that are not currently available in your model.
Re-synch Status With VC Provider	Update the **Version Control** status value recorded for the selected Package in the project to match the value reported by the Version Control provider, without performing an XMI import or export. Use this function when the Package's Version Control status recorded in your project is out of synchrony with the Version Control status reported by your Version Control provider.
Version Control Settings	Display the '**Version Control** Settings' dialog.

Notes

- You set up **Version Control** using options from the project 'Version Control' submenu
- If the selected Package is not under Version Control, a different set of options is available
- If a Version Control configuration has not been defined for the model, no options for using Version Control are available, only the options for configuring Version Control

Check Out a Package

When you need to work on a Version Controlled Package, you check it out. The local XMI file associated with the Package is then checked-out from **Version Control**. No other user can check out the Package to make changes to it until it has been checked in again.

Access

Context Menu	Right-click on Package > **Package Control** > Check Out
Keyboard Shortcuts	**Ctrl+Alt+L**

Check out a single Package

The Package file is imported into your model, and the 'Package' icon is updated to reflect the change in the Package's **Version Control** status.

When working in a Private Model, if the system detects that the Package content in the model is already up to date with the latest revision of the Package file retrieved from Version Control, then the 'Import Package' dialog displays first. This dialog is not displayed for a Shared Model.

These options are available:

- Force Reload From XMI - reload the Package from XMI regardless of whether it is up to date or not
- Accept current Package - select to skip the process of re-importing the Package from XMI
- Refresh model view - select to refresh the **Browser window** and diagrams, by reloading the Package content from the project database
- Always use these settings - when selected, if you subsequently check out a Package that is found to be up to date, the same settings are applied again without displaying the dialog

Notes

- If you check out a Version Controlled Package whilst offline, the 'Package' icon has a red figure 8 in front of it
- If you have selected the 'Always use these settings' checkbox and you want to reconfigure the 'Import Package' dialog, press the **Ctrl key** whilst you select the '**Package Control** | Check Out' context menu option; the dialog displays and you can change the settings

Undo Check Out of a Package

If you check out a Package and then decide not to proceed, you can undo the check-out and discard any modifications that have been made, by restoring the Package content to the latest revision held in **Version Control**. The Package returns to a checked-in state and subsequently can be checked out by any user, including yourself if, for example, you need to reverse incorrect changes before checking the Package out and starting again.

Access

Context Menu	Right-click on Package > **Package Control** > Undo Check Out

Undo check out of a selected Package

A confirmation dialog displays; click on the **OK button**.

The latest revision of the Package is retrieved from **Version Control** and re-imported into your model. The icon against the Package in the **Browser window** is updated to reflect the change in the Package's Version Control status.

Check In a Package

When you have finished working on the contents of a Package under **Version Control**, and you want to return it to the model for other users to see, you check it in.

Access

Context Menu	Right-click on Package > **Package Control** > Check In
Keyboard Shortcuts	**Ctrl+Alt+S**

Check in a single Package

Step	Action
1	The selected Package is exported and the 'Add Comment' dialog displays. A default comment is provided that contains the current date and time. Optionally, modify the default check-in comment
2	Click OK. The Package file is checked-in to **Version Control** and the Package icon is updated to reflect the change in Version Control status.

Check Out a Model Branch

If you need to check out a number of related Packages involved in a particular unit of work, to update the contents, you can do so in a single operation by checking out the whole model branch that contains them.

Access

Context Menu	Right-click on Package > **Package Control** > Check Out Branch

Check out a sub-tree of model Packages

Step	Action
1	The selected root-node Package and all of its contained sub-Packages are recursively checked out. Any Packages that cannot be checked-out are listed in a message box, with a brief description of the problem; for example: *The Package is already checked out by user 'Fred'*.
2	When Project Security is enabled in Lock to Edit mode, Enterprise Architect prompts you to apply a User Lock throughout the selected model branch before proceeding.

Check In a Model Branch

If you need to check in a number of related Packages involved in a particular unit of work, and that you have updated, you can do so in a single operation by checking in the whole model branch that contains them. You can also commit new revisions of the affected Packages as you complete milestones, whilst keeping the Packages checked-out for further editing.

Access

Context Menu	Right-click on Package > **Package Control** > Check In Branch

Check in Packages from within a model branch

Step	Action
1	The 'Select Packages to Check-in' dialog lists all Version Controlled and checked-out Packages within the selected model branch. By default, the entire list is selected. Optionally: • Click an individual Package to select just that Package • **Ctrl+click** on an individual Package to add or remove it from the selection • **Shift+click** on a range of Packages to select them • Click on the **All button** to select all listed Packages • Click on the **None button** to clear the selection
2	Optionally, you can commit into **Version Control** a new revision of all selected Packages, while keeping those Packages checked out for further editing. To do this, select the 'Keep packages checked-out after committing new revision' checkbox.
3	Click on the **OK button**. The 'Add Comment' dialog displays. A default comment is provided that contains the current date and time. This comment is applied to all Packages that are checked in.
4	Optionally, modify the default check-in comment.
5	Click on the **OK button**. The selected Packages are exported and checked-in. The Package icons are updated to reflect any change in **Version Control** status. If you opted to keep Packages checked out, there is no change in status.

Update to the Latest Revision of Selected Package

When you are part of a team working in a Distributed Model environment, you will want to periodically update your model with the changes that other team members have committed into **Version Control**. You can transfer the other users' updates from Version Control into the selected Package in the **Browser window**.

Access

Context Menu	Right-click on Package > **Package Control** > Get Latest

Update Package to latest revision

The local XMI file associated with the Package is updated to the latest revision from **Version Control**. The XMI file is imported into your model database, updating the Package in your model.

When working in a Private Model, if the system detects that the Package content in the model is already up to date with the latest revision of the Package file retrieved from Version Control, then the 'Import Package' dialog displays first. This dialog is not displayed for a Shared Model.

These options are available:

- 'Force Reload From XMI' - reload the Package from XMI regardless of whether it is up to date or not
- 'Accept current Package' - select to skip the process of re-importing the Package from XMI
- 'Refresh model view' - select to refresh the **Browser window** and diagrams, by reloading the Package content from the project database
- 'Always use these settings' - when selected, if you subsequently check out a Package that is found to be up to date, the same settings are applied again without displaying the dialog

Notes

- The 'Get Latest' command is disabled for any Package that is checked-out (to anybody) in the currently loaded project
- When using a Shared Model environment, where all users are connected to a single model database, you should reload the Package from the database, rather than using the 'Get Latest' command
- If you have selected the 'Always use these settings' checkbox and you want to reconfigure the 'Import Package' dialog, press the **Ctrl key** whilst you select the '**Package Control** | Get Latest' context menu option; the dialog displays and you can change the settings

Update to the Latest Revision of All Packages

When you are part of a team working in a Distributed Model environment, you will want to periodically update your model with the changes that other team members have committed into **Version Control**. You can transfer the other users' updates to all Version Controlled Packages into the currently loaded project.

Access

Context Menu	Right-click on Package > **Package Control** > Get All Latest

Update all Packages in project to latest revision retrieved from Version Control

All the local XMI files for all the **Version Control** configurations used in the project are updated to the latest revision from Version Control. The system then scans the Packages in the model, to determine which ones are up to date and which are not, compared to the latest revisions of the associated Package files.

A prompt displays, providing these import options for Packages that are up to date:

- Import changed files only
- Always import
- Prompt for each file

Click on the **OK button**. The Version Controlled Packages in your project are updated according to the option you selected; if you chose the 'Prompt for each file' option, a prompt displays to confirm import of each file.

Notes

- There is no need to re-import Packages that are already up to date - re-importing Packages first deletes them from the project and then re-imports them from the XMI file, which is time consuming as well as unnecessary; we strongly recommend using the default option 'Import changed files only'
- The 'Get All Latest' command does not update any Package that is checked-out (to anybody) in the currently loaded project; otherwise, any changes not yet committed to **Version Control** would be discarded
- When using a Shared Model environment, where all users are connected to a single model database, the information in the model database is always the same as, or ahead of, what is committed into Version Control; in this situation, the Get All Latest command will simply refresh your view of the model database, by reloading diagrams or reloading Package content in the **Browser window**

Review Package History

It is possible to review the change history of Version Controlled Packages by examining previous revisions. If necessary, you can check out one of these earlier revisions of a Package for editing, effectively rolling-back to that prior revision of the Package.

Access

Context Menu	Right-click on Package > **Package Control** > File History

Review change history of a version-controlled Package

Step	Action
1	For **Version Control** environments using Subversion, CVS or TFS command line clients, the 'File Version History' dialog displays. Click on a revision number in the 'Revisions' field, to select that revision and view its log entry. For Version Control environments using SCC based clients, your particular product opens its own 'File Version History' dialog.
2	On the 'File Version History' dialog, you can optionally click on either: • Check Out - the selected revision of the Package file is retrieved from **Version Control** and imported into your model as a Package that is checked out for editing; you can subsequently check in this revision as a new HEAD revision, effectively allowing you to revert the Package to a prior revision or • Retrieve - the selected revision of the Package file is retrieved from Version Control and imported into your model, but the Package remains flagged as checked-in and cannot be modified; subsequently checking out the Package updates it to the latest revision before it is unlocked for editing

Notes

- If the selected Package was already checked out in the current model, the Retrieve and Check Out buttons are disabled
- If the selected Package contains any sub-Package that is already checked-out in the current model, a warning will be displayed and the retrieval or check-out will not go ahead
- If you check out a prior revision of a Package, but do not want to commit it as a new revision, right-click on the Package and select **Package Control** | Undo Check Out

Review Package History - SCC Client

It is possible to review the change history of Version Controlled Packages by examining previous revisions. If necessary, you can check out one of these earlier revisions of a Package for editing, effectively rolling-back to that prior revision of the Package. The process for reviewing the change history of Packages configured for **Version Control** with an SCC client (including products such as Visual Source Safe, TFS-SCC, ClearCase, Perforce, AccuRev and MKS Source Integrity) differs from that for Subversion, CVS or TFS command line clients.

Access

Context Menu	Right-click on Package > **Package Control** > File History

Review change history of a version-controlled Package (SCC client)

Step	Action
1	The change history mechanism offered by the third party SCC provider displays. To import a prior revision of the Package into your model, use the 'SCC History' dialog to retrieve the revision, then close the dialog.
2	The SCC client notifies Enterprise Architect that a different revision has been retrieved. A prompt then displays, asking whether you want to check-out the prior revision.
3	Optionally, click on either: • Yes, to check out the prior revision - the selected revision of the Package file is retrieved from **Version Control** and imported into your model as a Package that is checked out for editing; you can subsequently check in this revision as a new HEAD revision, effectively reverting the Package to the prior revision OR • No, to import the prior revision as read-only - the selected revision of the Package file is retrieved from Version Control and imported into your model, but the Package remains flagged as checked-in and cannot be modified; subsequently checking out the Package updates it to the latest revision before it is unlocked for editing

Notes

- If the selected Package was already checked out in the current model, the system does not proceed with retrieving a prior revision
- If you check out a prior revision of the Package, but do not want to commit it as a new revision, right-click on the Package and select '**Package Control** | Undo Check Out'

Retrieve Prior Revision - SCC Client

Depending on your **Version Control** product, retrieving a prior revision of a controlled Package can involve a number of prompts regarding overwriting the current local copy.

This example details retrieval of a prior revision from a TFS-SCC Version Control configuration.

Access

Context Menu	Right-click on Package > **Package Control** > File History

Example Procedure - retrieve prior revision, TFS-SCC client

Step	Action
1	Display the 'TFS File History' dialog.
2	Click on the **Get button**. The TFS-SCC client displays the 'Resolve Conflicts' dialog. This dialog offers the 'Automerge All XML Package files' option. DO NOT select this option. It is important to prevent any merging of Enterprise Architect's XML Package files.
3	Click on the **Resolve button**. The TFS-SCC client displays the 'Resolve writeable file conflict' dialog.
4	Select the 'Overwrite local file/folder' option. The existing working copy of the Package file is overwritten by the prior revision retrieved from **Version Control**.
5	Click the **OK button**. The TFS-SCC client redisplays the 'Resolve writeable file conflict' dialog; it should now show no conflicts.
6	Click on the **Close button**. The TFS-SCC client redisplays the 'File History' dialog.
7	Click on the **Close button**. Enterprise Architect displays a prompt, asking whether to check-out the prior revision.
8	Click on the: • **Yes button** to check-out the prior revision • **No button** to retrieve a read-only version of the Package, that is NOT checked-out and is NOT editable

Advanced Usage

Once you get familiar with using the fundamental functions of **Version Control** there are a number of more advanced functions that you might want to use in working with your models. The advanced Version Control functions are outlined in this table.

Advanced Functions

Export a Version Controlled Model Branch	Exports **Version Control** information about the root Package of a model Branch, that is used to simplify the process of exporting and importing a hierarchy of Packages from one model to another.
Include Other Users' Packages	Other users might be developing Packages in their own models that you could use in your model, or you might have other models containing Packages that you want to use in the current model.
Import a Model Branch From Version Control	Uses Enterprise Architect's Model Branch files, of which there are few, to retrieve information about the root Package file and to import the model branch. Model branch files can simplify the process of exporting and importing a hierarchy of Packages from one model to another.
Add Connectors To Locked Elements	Generally, when working in a diagram containing locked elements, you cannot add a connector to a locked element. There are scenarios in which a connector can be created on a locked element depending on the locking status of the parent Packages.
Validate Package Configurations	You can test the validity of the **Version Control** settings associated with each Version Controlled Package within your current model.
Resynchronize the Status of Version Controlled Packages	Re-synchronizes the **Version Control** status of Packages within your project with the status reported by your Version Control provider.

Include Other Users' Packages

Other users might be developing Packages in their own models that you could use in your model, or you might have other models containing Packages that you want to use in the current model. Unless you are sharing an SQL database or project file, those Packages are not automatically available to you. However, if the Packages have been placed into **Version Control**, you can import them into your model as children of one of your model's Packages.

Access

Context Menu	Right-click on Package > **Package Control** > Get Package

Import Packages from Version Control into current model

Step	Action
1	On the 'Get Shared File' dialog, click on the **drop-down arrow** of the '**Version Control** Configuration' field and select the Version Control configuration associated with the Package to retrieve. The file list is populated with the names of files available through that configuration, for retrieval and import into your model.
2	Click on the Package file to import into your model and click on the **OK button**. The Package file is imported as a new child Package, under the parent Package you selected.

Notes

- You must have access to the Package files through the **Version Control** system and you must define a Version Control configuration through which to access those files
- The Version Control configuration must use the same unique ID that was originally used to add the Package to Version Control
- XMI Package files associated with Packages that are already part of your project, are NOT included in the list of files available for import

Export Controlled Model Branch

Applying **Version Control** to a model can result in many XMI files being placed under Version Control. It might then be hard to locate and import the file corresponding to the root of a particular model branch. Using Model Branch Files (.eab files) overcomes this problem by making it easier to export and import Package hierarchies from one model to another.

You could export a newly created model branch from your own private copy of a model so that, for example:

- Another user can import that branch into their own private copy of the same model
- It can be imported for inclusion as a common branch in a number of different models

Access

Context Menu	Right-click on Package > **Package Control** > Export as Model Branch

Create a Model Branch File to represent a Package hierarchy stored in Version Control

Step	Action
1	On the 'Export as Model Branch' dialog, in the 'EAB Filename' field, type a name for your Model Branch File. Alternatively, click on the  button and browse for the file location. Note that the Package name is supplied as a default.
2	Click on the **OK button**. A branch file is created to represent the selected Package. The branch file is committed to **Version Control** using the same Version Control configuration that controls the Package you selected.

Notes

- You can specify any file name, including sub-folder names, as long as the file is contained in or below the working folder of your **Version Control** configuration
- The facility is only enabled for Packages that are already under Version Control

Import Controlled Model Branch

Applying **Version Control** to a model can result in many XMI files being placed under Version Control. It could then be hard to locate and import the file corresponding to the root of a particular model branch, if you want to:

- Retrieve a model branch created by another user in a private copy of a model, to import it into your own private copy of the same model
- Retrieve a model branch that is common in many models, for inclusion in a new model

Model Branch Files overcome this problem by simplifying the retrieval of Package hierarchies for use in other models. You use Enterprise Architect's Model Branch Files, of which there are few, to retrieve information about the root Package file - such as the name and type of the Version Control configuration for the selected Package, and the relative filename of the Version Controlled XMI file associated with the Package. The system then uses this information to import the branch into your model.

Prerequisites

Before you begin, you must have:

- An operational **Version Control** environment that can be accessed by Enterprise Architect, and
- All of the Version Controlled Package files and the model branch file associated with the model branch to import, in a valid and accessible working copy folder

Access

Context Menu	Right-click on target Package for import > **Package Control** > Import a Model Branch

Import a Model Branch

Step	Action
1	On the 'Import VC Model Branch' dialog, either: • Use the lower portion of the dialog to select a model branch file (this is the simpler option if the associated **Version Control** configuration has already been saved in the current model; continue to step 2) OR • Click on the **Find a Model Branch (.EAB) file button** (this option is useful when you have not yet defined the Version Control configuration that is associated with the model branch to be imported; see the *Manually Locating Model Branch Files* topic)
2	Click on the **drop-down arrow** in the 'Select a **Version Control** Configuration' field and select a configuration. A list of .eab files controlled by that configuration is displayed in the 'Select a Model Branch (.EAB) file' list.
3	Select the Model Branch File you need, then click on the **OK button**.

	The system imports the root Package specified in the Model Branch File and recursively imports and populates all the sub-Packages contained in the root Package.

Notes

- The Import a Model Branch command is only enabled for Packages that you (the current user) are able to edit, as the imported model branch is inserted into the model under your selected Package

Manually Locating Model Branch Files

When importing a Model Branch File from **Version Control**, you might not have the associated Version Control configuration saved in the model that is receiving the import. In this situation, it is simpler to manually browse the file system to locate the Model Branch File (.eab) and let Enterprise Architect derive the details of the configuration from the branch file you select.

Prerequisites

Before you begin, you must have:

- An operational **Version Control** environment that can be accessed by Enterprise Architect, and
- All of the Version Controlled Package files and the model branch file associated with the model branch to import, in a valid and accessible working copy folder

Access

Context Menu	Right-click on target Package for import > **Package Control** > Import a Model Branch : Find a Model Branch (.EAB) file

Locate the Model Branch File

Step	Action
1	Access the 'Import VC Model Branch' dialog, then browse for and select the Model Branch File that represents the model branch to import.
2	Click on the **Open button**. If the **Version Control** configuration referenced by the file is fully defined within the current model, the import commences at this point. Otherwise, Enterprise Architect displays a dialog prompting you to complete the required configuration.
3	Click 'Yes' to proceed with completing the definition of the **Version Control** configuration. The 'Version Control Settings' dialog is displayed.
4	Complete the definition of the configuration. (Typically this involves simply specifying the working copy folder.)
5	Click on the **Save button**. The configuration details are saved. The import of the model branch proceeds.

Notes

- The Import a Model Branch command is only enabled for Packages that you (the current user) are able to edit, as the imported model branch is inserted into the model under your selected Package

Add Connectors To Locked Elements

Generally, when working in a diagram containing locked elements, you cannot add a connector to a locked element. However, this depends on the lock status of the source and target elements (or more precisely, the lock status of the parent Packages of the source and target elements, when the source and target element are held in different Packages). There are scenarios in which a connector can be created on a locked element.

Lock Scenarios

Element Status	Add Connectors
Source unlocked, target unlocked	Yes, any kind of connector can be added
Source unlocked, target locked	Yes, except for Composition connectors
Source locked, target unlocked	No, except for Composition connectors
Source locked, target locked	No, prohibited for all connectors

Notes

- A connector can be added if its source is unlocked - you are modifying what the source can see
- The exception is Composition connectors, where the target (the parent) must be unlocked - you are modifying the parent by adding children

Validate Package Configurations

Having defined the **Version Control** settings for your current model, you can test the validity of those settings associated with each Version Controlled Package within the model.

Access

Ribbon	Configure > **Version Control** > Check Configuration

Validate Version Control settings

Step	Action
1	The validation process scans the model database and verifies that the **Version Control** configuration associated with each Version Controlled Package is fully specified in the current model. It also queries the corresponding Version Control provider to find the status of the Package file associated with each Version Controlled Package. The results of the validation process are sent to the **System Output** window.
2	Open the '**Version Control** Settings' dialog to complete the definition of any invalid or missing Version Control configurations.
3	Click on an error message in the **System Output** window to highlight the corresponding Package in the **Browser window**.
4	Right-click a Package node and select '**Package Control** | Configure Package' to open the 'Package Control Options' dialog. Correct any problems with the **Version Control** details for the Package. Correct any problems with the Package's associated XMI file.

Resynchronize the Status of Version Controlled Packages

It is possible to update the **Version Control** status of Packages within your project to re-synchronize with the status reported by your Version Control provider. This can be useful if you are creating copies of your project, where checking in a Package from one copy of the model leaves the Package in the second copy of the model with an out-of-date Version Control status.

For a given Package, the re-synchronization process queries the corresponding Version Control provider to find the status of the Package file associated with the Version Controlled Package. If necessary, the process then updates the Package flags within the model database, to synchronize the Package status recorded in the model with the value reported by the Version Control provider.

Access

Ribbon	Configure > **Version Control** > Re-Synch Status (applies to all Packages in model)
Context Menu	Right-click on Package > **Package Control** > Re-synch Status With VC Provider (applies to single Package only)

Resynchronize Version Control status

Step	Action
1	The results of the re-synchronization process are sent to the **System Output** window.
2	Double-click on any result message to select, in the **Browser window**, the corresponding Package.

Notes

- This process does not cause any Package data to be either exported from your model to the associated Package file, or imported from a Package file into your model's Package data
- If a Package has been checked-out and modified with Enterprise Architect, but your **Version Control** provider reports the Package file as checked-in, running this process marks the Package within Enterprise Architect as being checked-in, without exporting and committing the pending changes; subsequently checking-out the Package imports the latest revision of the Package file from Version Control, effectively discarding the uncommitted modifications from the model
- Similarly, if a Package file is checked-out to you in your local working copy folder, but not in the Enterprise Architect model, running this process marks the Package within the model as checked-out, but it does not import the associated Package file from the Version Control system; consequently, it is possible to check-in a Package from Enterprise Architect that is potentially out of date, compared to the latest revision of the Package file within the Version Control system

More Information

Edition Information

- **Version Control** facilities are available in the Professional, Corporate, Unified and Ultimate Editions of Enterprise Architect.
- A team needs to provide their own Version Control Server and Client that are compatible versions.

Notes

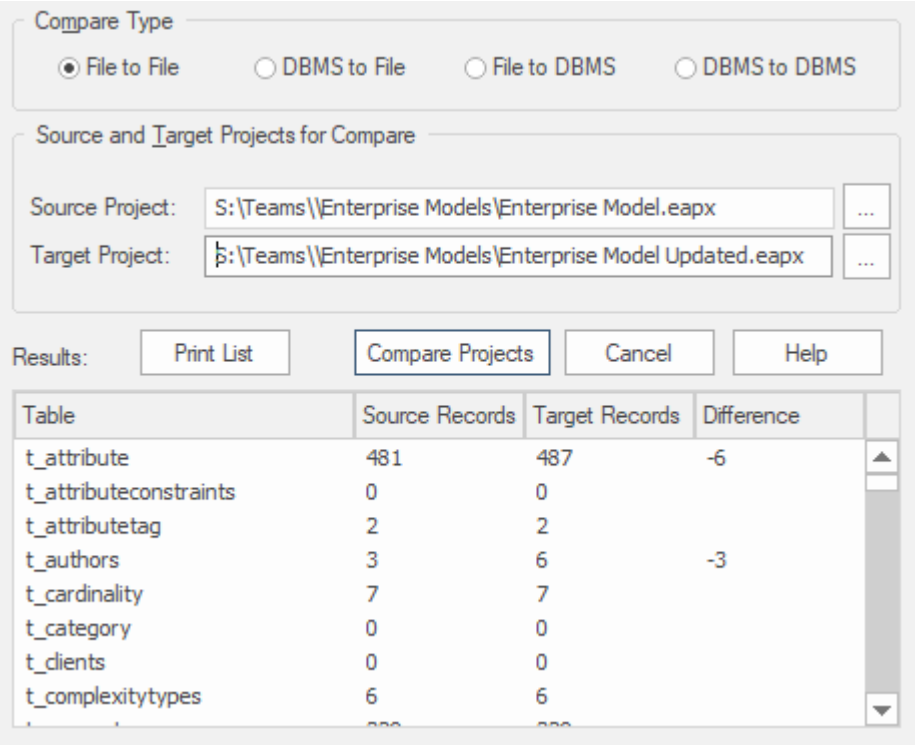
- Sparx Systems strongly urge you not to manipulate Version Controlled Package files outside of Enterprise Architect; it is possible to leave the Package files in a state that Enterprise Architect cannot recognize
- Database replication should not be combined with Version Controlled Packages
- If the Packages under **Version Control** contain any alternative images and those images are subject to frequent change, you can set the 'Export alternate images' option on the 'Preferences' dialog to export the images to the Version Control repository when you check in the Packages; if the images are not subject to frequent change, do not select this option and instead use 'Export/Import Reference Data' to manage alternative images

Compare Projects

Compares a Source and Target Repository for Differences

The Compare Project facility provides a summary of the changes to a repository by providing a comparison of the number of rows in each system table in the source and the target repositories. The modeling information contained in an Enterprise Architect repository is stored in a set of tables in a relational database. These are system tables that allow you to compare the number of rows in each of the tables, which will provide valuable insights into the differences between two repositories. The tables (with rare exceptions) have intuitive names as indicated by these examples:

- t_attribute - stores element attributes
- t_diagram - stores diagrams
- t_object - stores elements
- t_package - stores Packages (folders)



Apart from the deliberate changes that you and other modelers make when working on your repository, a number of operations can also make changes to your project that you either want to monitor carefully or that you do not want to retain. Such events include:

- Recovering from a database problem
- Restoring a backup
- Performing a Project Data Transfer
- Importing from XMI, and
- Deleting model elements

You might have made a copy of the original project, or the purpose of the operation is to generate a copy, in which case you can compare the size and row counts of the 'before' and 'after' copies as a convenient 'sanity check' that the two repositories are equivalent. The repositories can be on different platforms and the comparison can be made between files and server based repositories as described here:.

- Compare a project file to another project file (e.g. a *eapx file and a *.feap)
- Compare a project file to a DBMS-based repository (e.g. a *eapx file and an Oracle db)

- Compare two DBMS repositories (e.g. a MySQL db and an Oracle db)

The comparison examines the number of rows in each database table, producing a report indicating the total records in each and the difference in record count between the two. If discrepancies are found, you would need to investigate further manually. The comparison does not examine the actual data in the tables but rather provides the summary in the form of table row counts

Access

Ribbon	Configure > Model > Integrity > **Project Compare**

Compare two projects

Step	Action
1	On the '**Project Compare**' dialog, select the radio button for the database types of the two projects you want to compare: • File to File • DBMS to File • File to DBMS • DBMS to DBMS
2	In the 'Source Project' and 'Target Project' fields, type the name or connection string for the source and target projects to compare.
3	Click on the **Compare Projects button**. The results of the comparison display in the panel at the bottom of the dialog.
4	If you want to print the results of the comparison, click on the **Print List button**.