

Experion PKS
Qualification and Version Control User's Guide

EPDOC-XX90-en-431A
February 2015

Release 431

Document	Release	Issue	Date
EPDOC-XX90-en-431A	431	0	February 2015

Disclaimer

This document contains Honeywell proprietary information. Information contained herein is to be used solely for the purpose submitted, and no part of this document or its contents shall be reproduced, published, or disclosed to a third party without the express permission of Honeywell International Sàrl.

While this information is presented in good faith and believed to be accurate, Honeywell disclaims the implied warranties of merchantability and fitness for a purpose and makes no express warranties except as may be stated in its written agreement with and for its customer.

In no event is Honeywell liable to anyone for any direct, special, or consequential damages. The information and specifications in this document are subject to change without notice.

Copyright 2015 - Honeywell International Sàrl

Contents

1 About This Document	7
2 Qualification and Version Control System Overview	9
2.1 Comparison of Possible QVCS Operations	10
2.2 Version Number Display in Tree Views	13
2.3 About Custom Block Type (CBT) support	15
2.4 QVCS Administration and Access Privileges	16
2.5 QVCS Operations versus Access Privileges	17
2.6 QVCS Related Terms	18
2.7 QVCS State Chart	19
3 QVCS User Interface	21
3.1 QVCS Manager	22
3.2 QVCS Manager Menus	23
3.2.1 QVCS File menu	23
3.2.2 QVCS Edit Menu	24
3.2.3 QVCS View Menu	24
3.2.4 QVCS Tools Menu	25
3.2.5 QVCS Window menu	32
3.2.6 QVCS Help menu	32
3.3 QVCS Manager Toolbar	33
3.4 QVCS Manager Main Window	34
3.5 QVCS Manager Query Window	35
3.6 QVCS Manager Versions Window	37
3.7 QVCS Manager History Window	39
3.8 QVCS Reference View Window	42
3.9 QVCS Manager QVCS Admin Logs Window	43
3.10 Description of the QVCS Admin action	46
3.11 QVCS Difference Tool (Diff Tool) Menus	48
3.11.1 Diff Tool File Menu	48
3.11.2 Diff Tool Edit Menu	48
3.11.3 Diff Tool View Menu	48
3.11.4 Diff Tool Navigate Menu	49
3.11.5 Diff Tool Help Menu	49
3.12 QVCS Diff Tool Toolbar	51
3.13 QVCS Difference Tool (Diff Tool) Window	52
4 QVCS Support for CBT and UDT Overview	53
4.1 Basic rules for UDT support in QVCS	54
4.2 Differences between CBT and UDT	55
4.3 Specific rules for CBT support in QVCS	56
5 QVCS General Operations	59
5.1 Checking in an object to QVCS	60
5.1.1 Multiple object check in considerations	60
5.1.2 CBT and UDT check in considerations	60
5.1.3 Checking in an object	62
5.1.4 Example check in scenarios	64

5.2	Checking out an object from QVCS	66
5.2.1	Multiple object check out considerations	66
5.2.2	CBT and UDT check out considerations	66
5.2.3	Checking out an object	74
5.2.4	Example check out scenarios	75
5.3	Undoing Check out from QVCS	77
5.3.1	Multiple object undo check out considerations	77
5.3.2	Objects with connections undo check out considerations	77
5.3.3	CBT and UDT undo check out considerations	77
5.3.4	Undoing a check out	78
5.3.5	Example undo check out scenarios	79
5.4	Graphical Example of Layered Recipe Check In and Check Out Scenario	81
5.4.1	Example of objects that can be versioned	81
5.4.2	Example of versioned objects and rules for check in and check out	82
5.4.3	Example of rules for check in and check out with UDT	83
5.5	Instantiating CBT	84
5.5.1	Rules when a CBT Instance can be under QVCS	84
5.6	Viewing and Changing Object Properties	87
5.7	Making changes to an object	89
5.7.1	Changing a connection	89
5.7.2	Changing a user defined template (UDT)	89
5.7.3	Changing a custom block type (CBT)	89
5.8	Making multiple object qualification state transitions	90
5.9	Renaming an object	92
5.9.1	Renaming a basic block when a parameter connection exists	92
5.10	Deleting a versionable object	93
5.10.1	Deleting multiple objects	93
5.10.2	Deleting a user defined template (UDT)	93
5.10.3	Deleting a custom block type (CBT)	93
5.10.4	Example CBT delete scenarios	95
5.11	Restoring Deleted Object	97
5.12	Editing CBT in QVCS	99
5.12.1	Rules for CBT under QVCS	99
5.12.2	Example CBT edit scenarios	100
5.13	Using Change Parent with CBT	102
5.13.1	Rules for using Change Parent with CBT instance	102
5.13.2	Initiating bulk Change Parent operation	102
5.13.3	Example CBT Change Parent scenarios	102
5.14	Viewing Information for a Version of an Object	104
5.15	Viewing History details of an Object	106
5.16	Viewing Reference data for Objects	107
5.17	Example Reference View Scenario	110
5.17.1	Sample configuration	110
5.17.2	Example Reference Views for sample configuration	110
5.18	Retrieving Version of an Object	112
5.19	Comparing Versions of an Object Using Diff Tool	116
5.20	Loading QVCS Objects to a Controller	117
5.20.1	Typical version mismatch load scenario	117
5.20.2	Graphical example of CBT load scenario	117
5.20.3	Example CBT load scenarios	118
5.21	Using upload or update to project commands	120
5.21.1	Considerations for upload command	120
5.21.2	Considerations for update to project command	120
5.21.3	Example CBT Update to Project scenarios	120

5.22 Importing Objects	122
5.22.1 Import Rules for QVCS	122
5.22.2 Version number rules	122
5.22.3 General import considerations	123
5.22.4 CBT and UDT import considerations	123
5.22.5 Example CBT import scenarios	124
5.23 Exporting Objects	127
6 Relaxed Loading Support for QVCS	129
6.1 Relaxed Load Function	130
6.1.1 Typical relaxed load functional scenario	130
6.1.2 Initiating queries for No-QVCS objects	130
7 QVCS Revert Operations	133
7.1 About QVCS Revert function	134
7.2 Maintaining QVCS Revert Labels	135
7.2.1 Calling up the Revert Label Maintenance dialog box	135
7.2.2 Adding, deleting, modifying or viewing assignments for revert labels	135
7.3 Applying or Removing QVCS Revert Labels	138
7.3.1 Calling up the Apply/Remove type dialog box	138
7.3.2 Applying or removing revert labels	139
7.4 Reverting to a Version or a Label	141
7.4.1 Graphical example of CBT revert considerations	141
7.4.2 Graphical example of CBT instance revert considerations	142
7.4.3 Graphical example of CBT version consistent set revert considerations	143
7.4.4 Graphical example of CBT with UDT revert considerations	143
7.4.5 Reverting to a version	144
7.4.6 Reverting to a label	145
7.5 Example revert operation scenarios	146
7.5.1 Error Message related to revert operations	147
8 QVCS Administration Operations	149
8.1 About QVCS Qualification State	150
8.2 Configuring Qualification States	151
8.2.1 Qualification life cycle configuration tips	151
8.2.2 Calling up the Qualification State Configuration dialog box	151
8.2.3 Adding, deleting, or modifying qualification states	152
8.2.4 Defining state transition requirements	154
8.2.5 Graphic Example of Qualification States Configuration and State Chart	155
8.3 Qualification States and Import Considerations	158
8.4 Qualification State and Template Defining Parameters Consideration	159
8.5 QVCS Electronic Signature Purpose	160
8.6 Configuring QVCS Electronic Signature	161
8.7 Viewing QVCS Logs	162
9 QVCS Migration and Interoperability	163
9.1 Migration Considerations	164
9.2 Interoperability Notes	165
10 QVCS Maintenance	167
10.1 Using DBADMIN for QVCS database maintenance	168
11 QVCS Troubleshooting	169
11.1 Isolating Problems	170
11.2 Initial Checks	171
11.2.1 Checking Control Builder error code reference	171
11.2.2 Viewing flash log	171

- 11.2.3 Viewing release information log 171
 - 11.2.4 Viewing trace log 172
 - 11.2.5 Checking version and revision log 172
 - 11.2.6 Checking server point build log 172
 - 11.2.7 Checking server point build error log 172
 - 11.2.8 Checking error log 172
 - 11.3 Getting Further Assistance 173
 - 11.3.1 Other troubleshooting sources 173
 - 11.3.2 Guidelines for requesting support 174
- 12 Notices 175**
 - 12.1 Documentation feedback 176
 - 12.2 How to report a security vulnerability 177
 - 12.3 Support 178
 - 12.4 Training classes 179

1 About This Document

This document describes how to use the Qualification and Version Control System to track changes made to the Experion control strategy.

Revision history

Revision	Date	Description
A	February 2015	Initial release of the document.

2 Qualification and Version Control System Overview

The Qualification and Version Control System (QVCS) provides version control for Experion control system objects as well as a flexible qualification system framework for managing these objects in a regulated environment. You must purchase a license to use the **Full** version of QVCS that provides a framework for you to implement a qualification system that supports multiple levels of qualification for checked in objects with potential electronic signature support. The **Basic** version of QVCS only provides automatic version numbering when a change is made.

The QVCS components are installed as part of the Experion software installation. The QVCS database must be installed on the same computer as the Engineering Repository Database (ERDB), but it will run on any node configuration that supports running Control Builder.

Related topics

- “Comparison of Possible QVCS Operations” on page 10
- “Version Number Display in Tree Views” on page 13
- “About Custom Block Type (CBT) support” on page 15
- “QVCS Administration and Access Privileges” on page 16
- “QVCS Operations versus Access Privileges” on page 17
- “QVCS Related Terms” on page 18
- “QVCS State Chart” on page 19

2.1 Comparison of Possible QVCS Operations

The Experion system provides up to three versioning type operations for tracking information about the history of changes made to objects in Control Builder. If you do not purchase a QVCS license, the installed Experion system is considered to be a No-QVCS system. You can change a No-QVCS system to a Basic-QVCS system through a configuration setting on the System Preferences dialog in Control Builder. The Basic-QVCS option does not appear on the dialog, if a QVCS license is installed. The following table summarizes the differences among the three versioning type operations.

Parameter or Operation	No-QVCS System	Basic-QVCS System	Full QVCS System
<i>Enabling mechanism</i>	None	Set through System Preferences dialog in CB	Purchase license
<i>Display of version number in titles</i>	None	VERSIONNUM is used as source of displayed number	VERSIONNUM is used as source of displayed number
<i>Version menu and toolbar</i>	No	No	Yes
<i>VERSION^Q</i>	User entered text	Formatted string of VERSIONNUM	Formatted string of VERSIONNUM
<i>VERSIONNUM^Q</i>	Not used	Auto-incremented by save changes	Incremented when checked in to the VCS, major/minor set by user
<i>CREATEDBY^{Q,I}</i>	Set by first save changes	Set by first save changes	Set when first checked in
<i>MODIFIEDBY^{Q,I}</i>	Set by save changes	Set by save changes	Set when checked in
<i>DATECREATED^{Q,I}</i>	Set by first save changes	Set by first save changes	Set when first checked in
<i>VERSIONDATE^{Q,I}</i>	Set by save changes	Set by save changes	Set when checked in
<i>QUALSTATE^Q</i>	Not used	Not used	Set by user (only when already checked in)
<i>BLCKCOMMENT1... BLCKCOMMENT4^I</i>	User entered	User entered	User entered
^Q Appears on QVCS tab of configuration form for the tabbed object.			
^{Q,I} Appears on both QVCS and Identification tabs of configuration form for the tabbed object.			
^I Appears on Identification tab of configuration form for the tagged object.			

QVCS

Server History

Server Displays

Version Properties

Name:

C300_133

Version:

1.00

Status:

Checked in

Comment:

Add to QVCS

Created by:

Process User

Created on:

2/22/2007 8:03:23 AM

Last modified by:

Process User

Last modified on:

2/22/2007 8:03:23 AM

Qualification State Configuration

Current state:

Default state

Change...

Revert Label Configuration

#	Applied revert labels
---	-----------------------

Apply/Remove Labels...

Figure 1: Typical information shown on QVCS tab for tagged object in Full QVCS system

Soft Failures		QVCS		Server History		Server Displays		Control Confirmation		Identification	
Name:		FIM4_142									
Description:		Fieldbus Interface (4 FF Links)									
Block Comment 1:											
Block Comment 2:											
Block Comment 3:											
Block Comment 4:											
Library:											
System Template:		SYSTEM:FIM4									
Base Template:		SYSTEM:FIM4									
Created By:		Process User									
Date Created:		2/22/2007 8:22:59 AM									
Last Modified By:		Process User									
Date Last Modified:		2/22/2007 8:22:59 AM									

Figure 2: Typical information shown on Identification tab for tagged object

2.2 Version Number Display in Tree Views

The following table summarizes the display of version number (VERSIONNUM) parameter in Control Builder tree views for given QVCS operation. The following figure illustrates how version numbers are typically displayed in the **Project** and **Library** tree views. Custom Block Types (CBT) are QVCS versionable and are displayed like other versioned objects, as shown in the following figure. Note that rules governing CBTs are slightly different than those for other versionable objects and differences are clearly indicated in the various consideration sections in this document. See the “Specific rules for CBT support in QVCS” on page 56 section for more information.

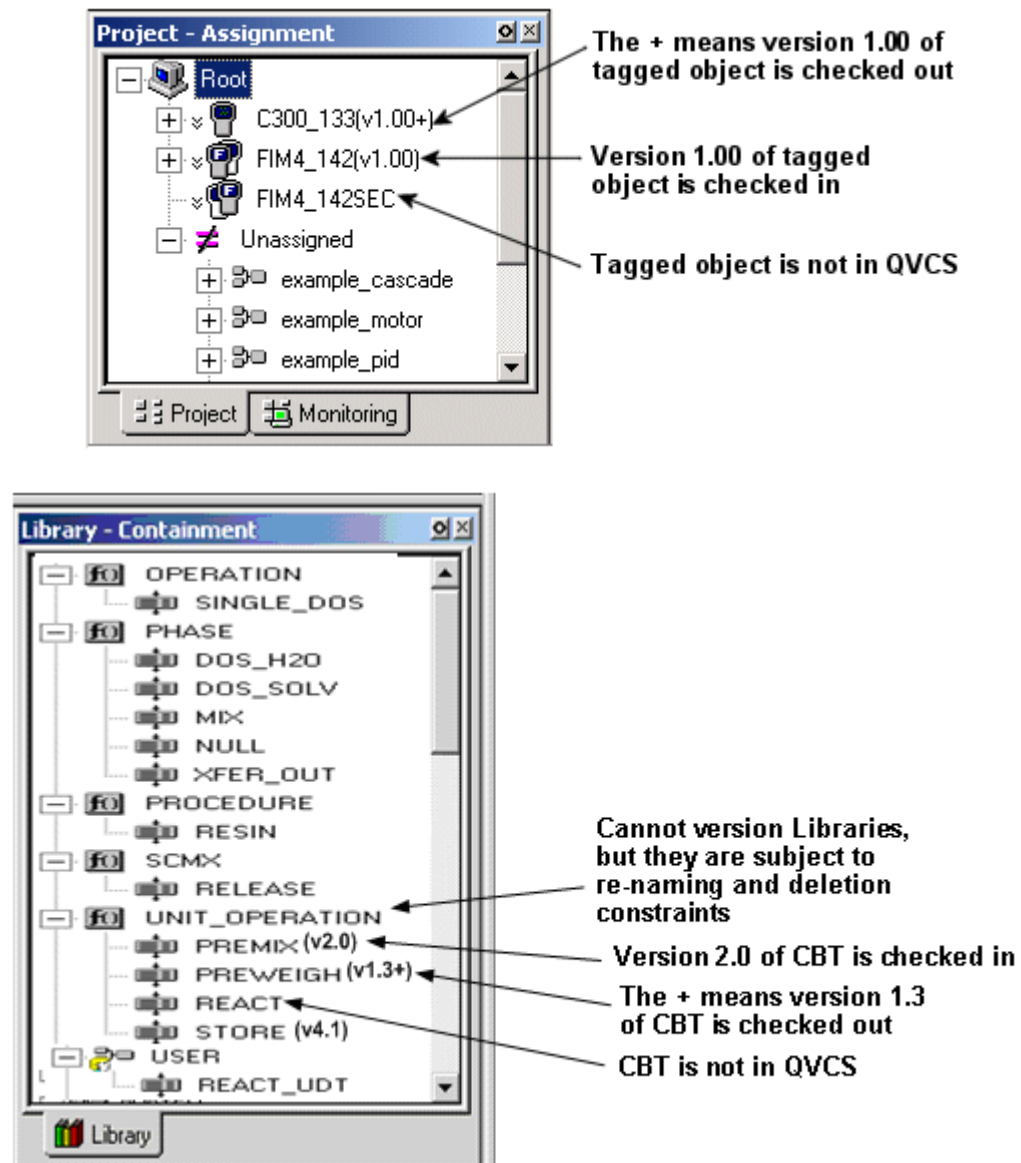


Figure 3: Typical version number display in CB Project and Library tree views.

If System is . . .		
<i>No-QVCS, Then:</i>	<i>Basic-QVCS, Then:</i>	<i>Full-QVCS, Then:</i>
<p>The version number (VERSIONNUM) is neither displayed nor incremented for tagged block objects in CB tree view.</p> <p>If you change a No-QVCS to a Basic-QVCS or upgrade to Full-QVCS, any user entered VERSION text will be replaced with the system maintained VERSIONNUM formatted text.</p>	<p>The VERSIONNUM is automatically incremented each time a changed tagged block object is saved. The increment will either be a minor increment (.01), if no structural changes have been made; or a major increment (to the next whole number), if structural changes have been made. The VERSION parameter contains the formatted string of VERSIONNUM.</p>	<p>The VERSIONNUM parameter will only be incremented when an object is checked in. The VERSION parameter contains the formatted string of VERSIONNUM.</p>

2.3 About Custom Block Type (CBT) support

The term Custom Block Type (CBT) is used as a generic reference for a user defined Custom Algorithm Block (CAB), Custom Data Block (CDB), or PHASE block. Once a CBT is defined it resides in a user named library in the **Library** tree in Control Builder.

All CBT block types are defined by the user, as opposed to the standard system types (such as CM, SCM, RCM, I/O, and so on). Once a CBT is defined, it exists in a Custom Block Library, and is available for creating an instance in a container object (CM, SCM, and RCM) or as a User Defined Template (UDT), as any other block type in the system. The CBT Instance can then be versioned as any other versionable objects. The CBT objects in the **Library** tree can be versioned as any other objects in the **Project/Library** tree, and changes to the CBT objects can be kept track of in the QVCS database.

2.4 QVCS Administration and Access Privileges

The licensed QVCS application installation includes one user with QVCS administrator privileges and one user with QVCS access based on the Experion server security configuration of Manager (MNGR) and Engineer (ENGR) or Supervisor (SUPV), respectively. This means that any user who is a member of the ENGR or SUPV group has rights to access the QVCS and any user who is a member of the MNGR group has QVCS administration rights.

The following table shows how the Control Builder security level maps to the QVCS access privileges. This is a fixed relationship and cannot be changed by the user. You can change the security level required to launch QVCS through the **Operation Permissions** selection under the **Tools** menu in Control Builder.

Table 1: Control Builder Security Level versus QVCS Access Privilege

If Control Builder Security Level is . . .	Then, QVCS Access Privilege is . . .
View Only (ViewOnly)	NONE
Acknowledge Only (AckOnly)	NONE
Operator (OPER)	READ
Supervisor (SUPV)	FULL
Engineer (ENGR)	FULL
Manager (MNGR)	ADMIN

2.5 QVCS Operations versus Access Privileges

The following table shows the access privilege a user needs to perform a given QVCS operation. The operations covered in this document assume that users have sufficient access privileges to perform them.

Can Perform This QVCS Operation . . .	If QVCS Access Privilege is . . .		
	<i>READ</i>	<i>FULL</i>	<i>ADMIN</i>
Check In	NO	YES	YES
Check Out	NO	YES	YES
Undo Check Out	NO	YES	YES
Revert	NO	YES	YES
Compare Versions of an Object Using Diff Tool	YES	YES	YES
Reference View	YES	YES	YES
Retrieve	YES	YES	YES
View Object Properties	YES	YES	YES
View QVCS History	YES	YES	YES
Apply Revert Labels	NO	YES	YES
Revert Label Maintenance	NO	YES	YES
Change Qualification State	NO	YES	YES
Configure Electronic Signature	NO	NO	YES
Configure Qualification States	NO	NO	YES
View QVCS Logs	NO	NO	YES
Perform QVCS Manger Query	YES	YES	YES
View Object Versions	YES	YES	YES
Preview and Print various QVCS Reports	YES	YES	YES

2.6 QVCS Related Terms

The following table list definitions for some QVCS related terms.

Term	Definition
Basic QVCS	A license for QVCS has not been purchased. The created and modified dates and user ID's are tracked, and an automatically incremented version number is displayed for tagged objects. This state is set through a System Preference selection in Control Builder.
Check in	The action that places the current image of an object that is in the Project or Library tab of Control Builder into the Qualification and Version Control System. If a user performs an import when an object is checked in, it results in a new checked in version.
Check out	The action that marks a specific version of an object that is in the Qualification and Version Control System as editable. If a user checks out a non-current version of an object, it replaces the current version in the Project or Library tab of Control Builder.
Fallback state	The qualification state that an object is placed in after it is checked in to the QVCS.
Full QVCS	A license has been purchased for QVCS. All the features described in this document (check in, check out, revert, history, etc.) are operational.
No QVCS	A license for QVCS has not been purchased. Only the created and modified dates and user ID's are tracked. This is the default state of Experion system.
Qualification state	The definition of the stages, or states, that a control strategy passes through to become a qualified control. It is defined by the user and is the implementation of their standard operating procedures for meeting the stringent record keeping requirements of various regulatory agencies. Some regulatory guidelines refer to a "Qualification Life Cycle State", which is identical to the qualification state concept.
Restricted signing state	The restricted signing state is a qualification state that can be referenced by another qualification state. If a qualification state has a restricted signing state defined, and both states require a signature, the signatures must be from different users.
Retrieve	The action of obtaining a copy of a specific version of a specific object from the QVCS. It does not affect the contents in the Project or Library tab of Control Builder.
Version repository	The QVCS maintains a repository with all versions of each configured object.

2.7 QVCS State Chart

The following state chart provides a basic description of the QVCS control states and transitions.

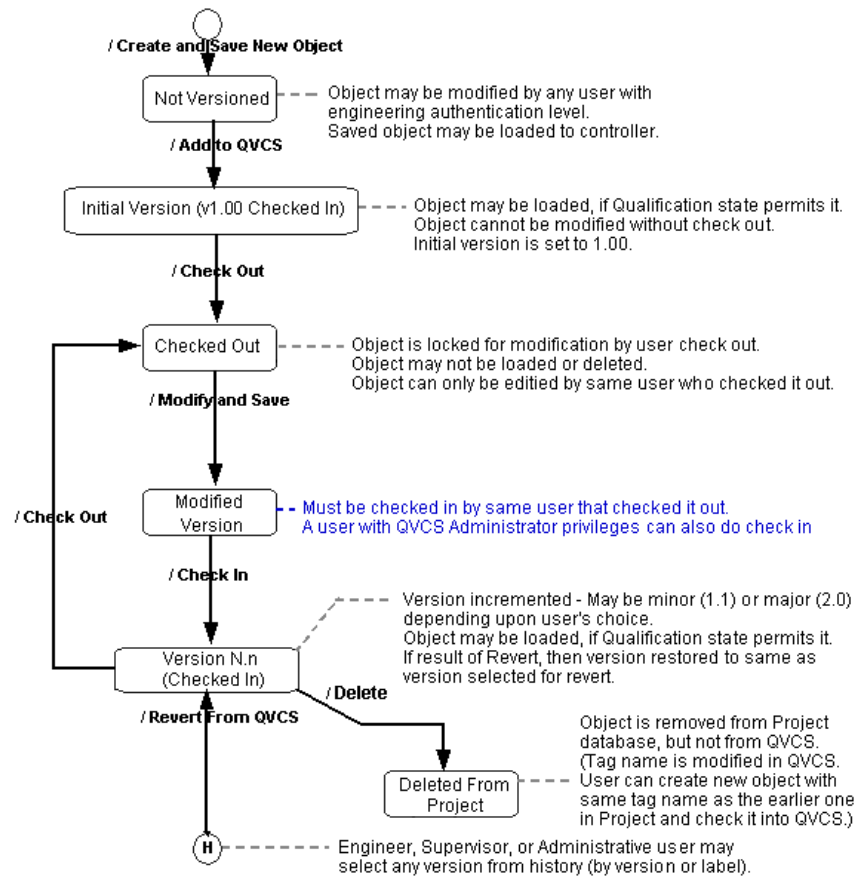


Figure 4: Full-QVCS version control state chart

3 QVCS User Interface

The QVCS Manager window is the main user interface for the Full-QVCS system. You can only launch the QVCS Manager through Control Builder and it is tightly coupled to the instance of Control Builder from which it was launched.

Related topics

- “QVCS Manager” on page 22
- “QVCS Manager Menus” on page 23
- “QVCS Manager Toolbar” on page 33
- “QVCS Manager Main Window” on page 34
- “QVCS Manager Query Window” on page 35
- “QVCS Manager Versions Window” on page 37
- “QVCS Manager History Window” on page 39
- “QVCS Reference View Window” on page 42
- “QVCS Manager QVCS Admin Logs Window” on page 43
- “Description of the QVCS Admin action” on page 46
- “QVCS Difference Tool (Diff Tool) Menus” on page 48
- “QVCS Diff Tool Toolbar” on page 51
- “QVCS Difference Tool (Diff Tool) Window” on page 52

3.1 QVCS Manager

The QVCS Manager Window host menus and toolbars along with multiple document interface (MDI) type child windows. The following MDI child windows make up the QVCS Manager.

- Main Window
- Query Window
- Versions Window
- History Window
- Versions Control System Log Window

The following illustration shows the typical default view of a QVCS Manager upon its initial launch from Control Builder with tagged object selected.

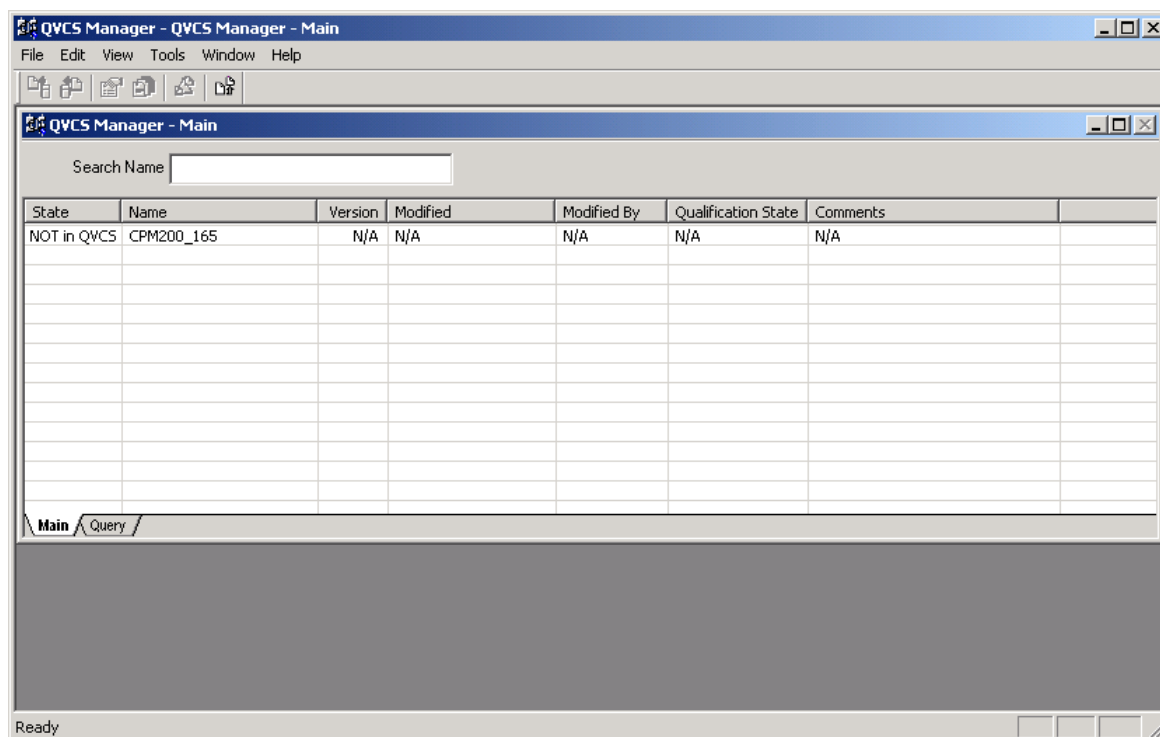
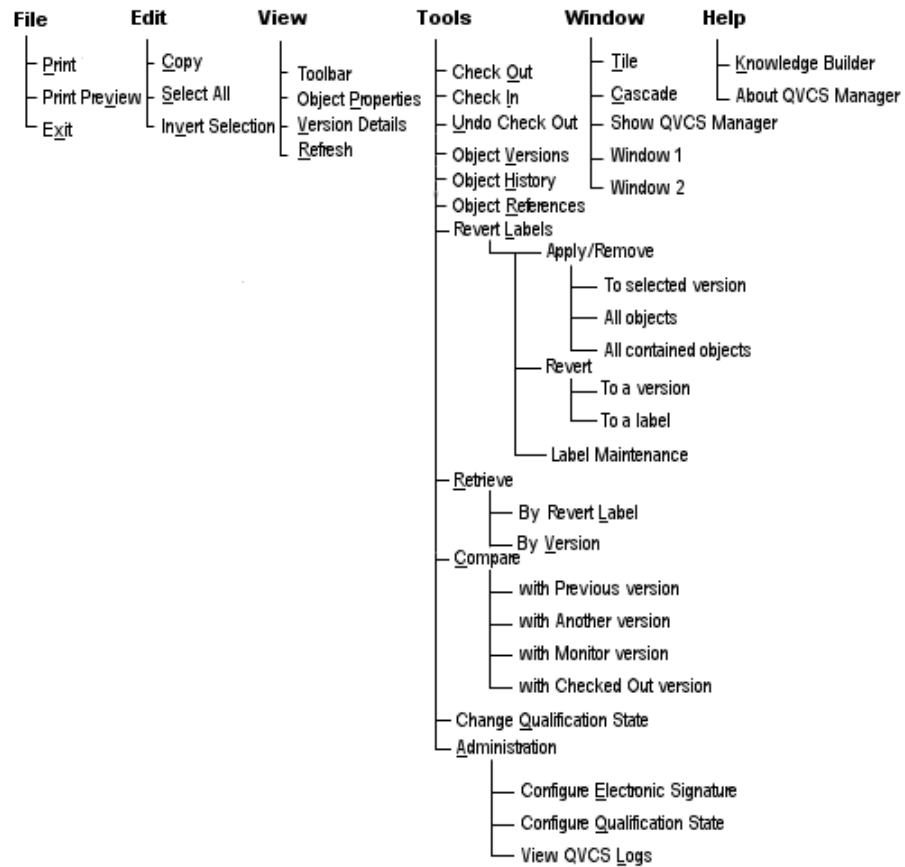


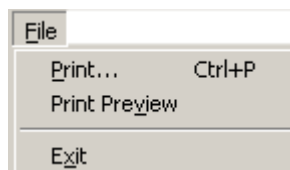
Figure 5: Typical QVCS Manager Main tab view with object selected in Control Builder

3.2 QVCS Manager Menus

The following illustration shows the menu tree for the QVCS Manager.



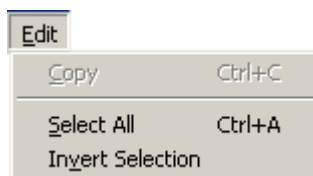
3.2.1 QVCS File menu



Menu Selection	Description	Available
<i>Print</i>	Prints the current active window contents. Brings up the Print dialog box to choose the printer and the contents is printed. Option is available, if printer is connected.	This selection is available in the <ul style="list-style-type: none"> • Query Window • Versions Window • History Window • Reference View Window • QVCS System Log Window

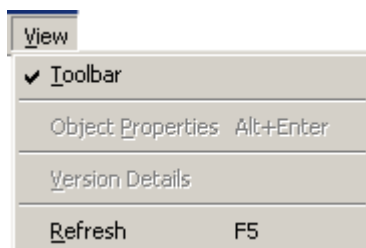
Menu Selection	Description	Available
<i>Print Preview</i>	Shows the preview of the contents of the active window that will be sent to the printer	This selection is available in the: <ul style="list-style-type: none"> • Query Window • Versions Window • History Window • Reference View Window • QVCS System Log Window
<i>Exit</i>	Closes the QVCS Manager	Always available

3.2.2 QVCS Edit Menu



Menu Selection	Description	Available
<i>Copy</i>	Copy the contents of the Version Information in text window to clipboard.	This selection is available in the: Viewing object version is text format and any text is selected in the view
<i>Select All</i>	Select all the items in the active window.	This selection is available in the: <ul style="list-style-type: none"> • Main Window • Query Window
<i>Invert Selection</i>	Invert the selected items in the active window. All the selected items in the list are unselected and the rest are selected.	This selection is available in the: <ul style="list-style-type: none"> • Main Window • Query Window

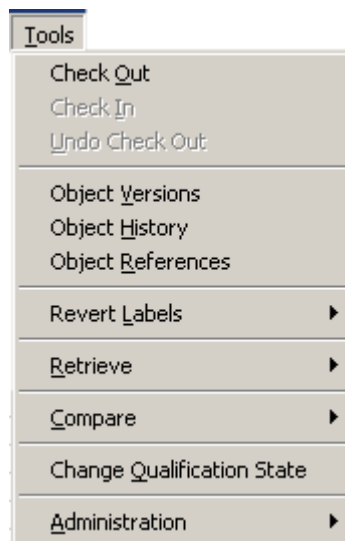
3.2.3 QVCS View Menu



Menu Selection	Description	Available
<i>Toolbar</i>	Show/Hide the QVCS toolbar. A checkmark towards left of the text "Toolbar" indicates the toolbar is shown otherwise toolbar will be hidden. Menu item works like a check box.	Always available

Menu Selection	Description	Available
<i>Object Properties</i>	Shows the Properties Dialog of the selected object.	This selection is available in the: <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window Also a single object should be selected for this menu to be available.
<i>Version Details</i>	Load the details of the version in text form in the Version Details view of the Versions Window.	This selection is available in the: <ul style="list-style-type: none"> • Versions Window
<i>Refresh</i>	Refresh the active window.	All windows

3.2.4 QVCS Tools Menu

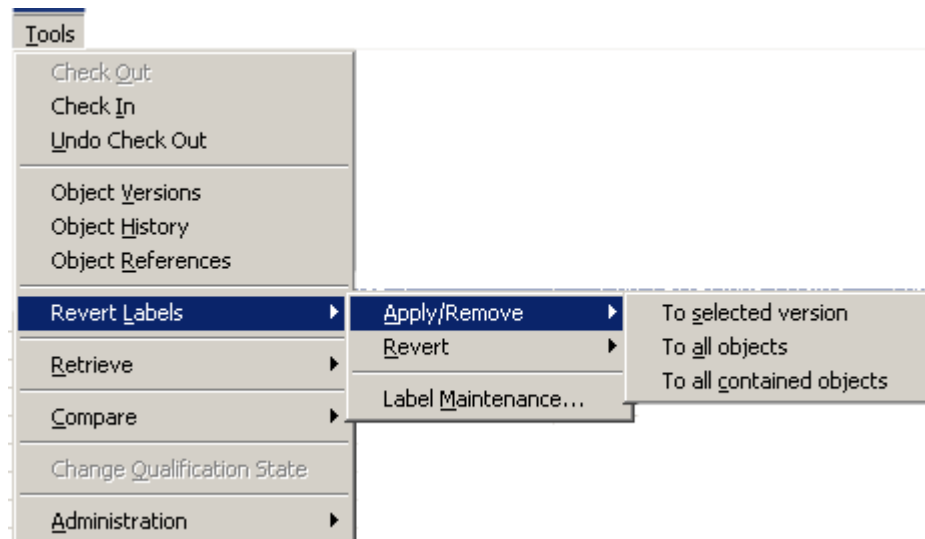


Menu Selection	Description	Available
<i>Check Out</i>	Call up Check Out dialog. The Check Out is performed for all the objects selected in the list.	This selection is available in the: <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window When multiple objects are selected, if all the selected objects have the Checked In state, then the menu is available. The menu item will be available only for users with QVCS Engineer or Manager access.

Menu Selection	Description	Available
<i>Check In</i>	Calls up Check In dialog. The Check In is performed for all the objects selected in the list.	<p>This selection is available in the:</p> <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window <p>When multiple objects are selected, if all the selected objects have the Checked Out state or The object is not added to VCS, then the menu will be active</p> <p>The menu item will be available only for users with QVCS Engineer or Manager access.</p>
<i>Undo Check Out</i>	Calls up Undo Check Out dialog. The Undo Check Out is performed for all the objects selected in the list.	<p>This selection is available in the:</p> <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window <p>When multiple objects are selected, if all the selected objects have the Checked In state, then the menu is available.</p> <p>The menu item will be available only for users with QVCS Engineer or Manager access.</p>
<i>Object Versions</i>	Calls up the Versions Window for the selected object. A new window will be opened for each object.	<p>This selection is available in the:</p> <ul style="list-style-type: none"> • Main Window • Query Window <p>Also, a single object should be selected for this menu to be available.</p>
<i>Object History</i>	Calls up the History Window for the selected object. A new window will be opened for each object.	<p>This selection is available in the:</p> <ul style="list-style-type: none"> • Main Window • Query Window <p>Also, a single object should be selected for this menu to be available.</p>
<i>Object References</i>	Calls up the Reference View Window for the selected object or objects	<p>This selection is available in the:</p> <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window <p>Also, at least one object should be selected for this menu to be available.</p>
<i>Revert Labels</i>	See the following Revert Label submenu topic.	N/A
<i>Retrieve</i>	See the following Retrieve submenu topic.	N/A
<i>Compare</i>	See the following Compare submenu topic.	N/A

Menu Selection	Description	Available
<i>Change Qualification State</i>	Calls up the Change Qualification State dialog for the selected object.	<p>This selection is available in the:</p> <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window <p>Also, a single object should be selected for this menu to be available.</p> <p>This menu option is available even if multiple objects are selected. All objects Qualification state is changed. For multiple objects, change qualification state can happen provided they are all in the same state.</p>
<i>Administration</i>	See the following Administration submenu topic.	N/A

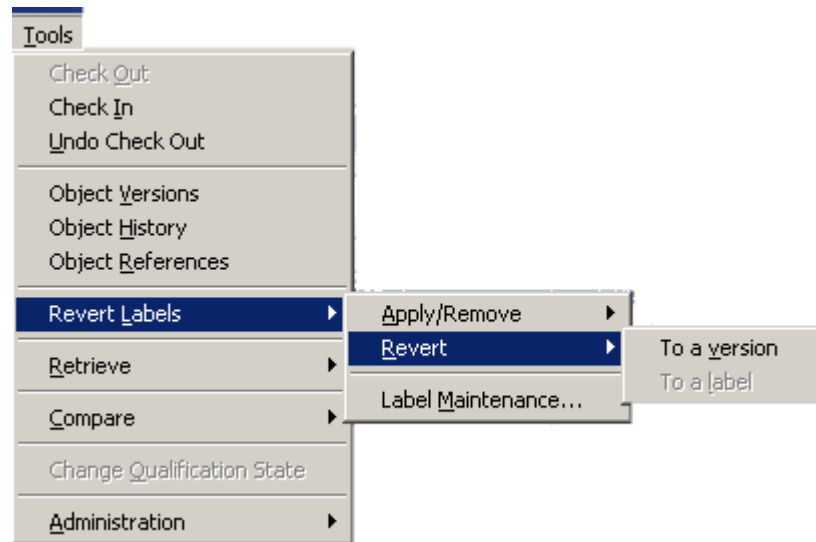
Revert Labels - Apply/Remove Submenu



Menu Selection	Description	Available
<i>To selected version</i>	Apply/Remove the labels to only the selected version of the object. This will show the Apply/Remove Labels to Version dialog	<p>This selection is available in the:</p> <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window <p>Also a single object should be selected for this menu to be available.</p> <p>The menu item is enabled only for users with QVCS Engineer or Manager access.</p>

Menu Selection	Description	Available
<i>To all objects</i>	Apply/Remove the labels to the current version of all the objects in the Project and Library tree in Control Builder. This will show the Apply/Remove Labels to All dialog	This selection is available in the: <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window • Reference View Window The menu item is enabled only for users with QVCS Engineer or Manager access.
<i>To all contained objects</i>	Apply/Remove the labels to all the contained objects. This will show the Apply/Remove Labels to Contained dialog	This selection is available in the: <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window The menu item is enabled only for users with QVCS Engineer or Manager access.

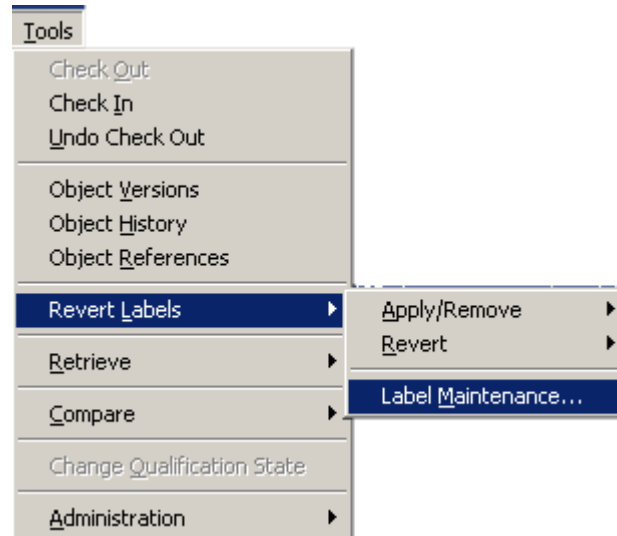
Revert Labels - Revert Submenu



Menu Selection	Description	Available
<i>To a version</i>	Calls up the Revert to Version dialog. The Revert to version will be applied to a single object.	This selection is available in the: <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window Also a single object should be selected for this menu to be available. <p>The menu item is enabled only for users with QVCS access.</p>

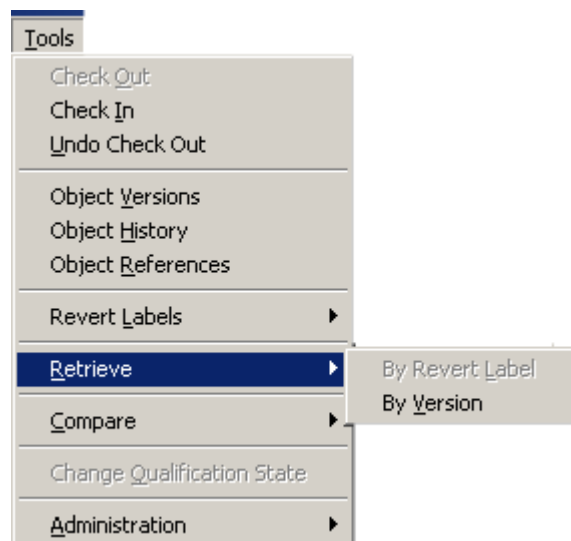
Menu Selection	Description	Available
<i>To a label</i>	Calls up the Revert to Label dialog	<p>This selection is only available if no object is selected in the:</p> <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window <p>The menu item is enabled only for users with QVCS access.</p>

Revert Labels - Label Maintenance Submenu



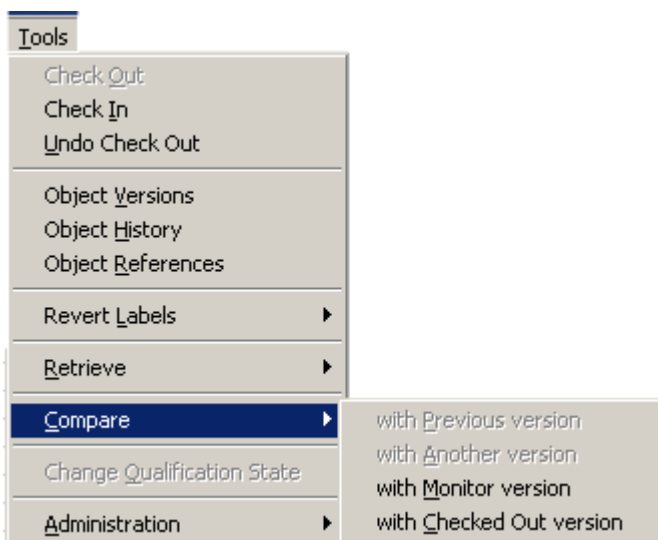
Menu Selection	Description	Available
<i>Label Maintenance</i>	Calls up the Revert Label Maintenance dialog	Always available.

Retrieve Submenu



Menu Selection	Description	Available
<i>By Revert Label</i>	Calls up the Retrieve By Revert Label dialog	This selection is only available if no object is selected in the: <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window The menu item is enabled only for users with QVCS access.
<i>By Version</i>	Calls up the Retrieve By Version dialog. The Revert to version will be applied to a single object.	This selection is available in the: <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window Also a single object should be selected for this menu to be available. The menu item is enabled only for users with QVCS access.

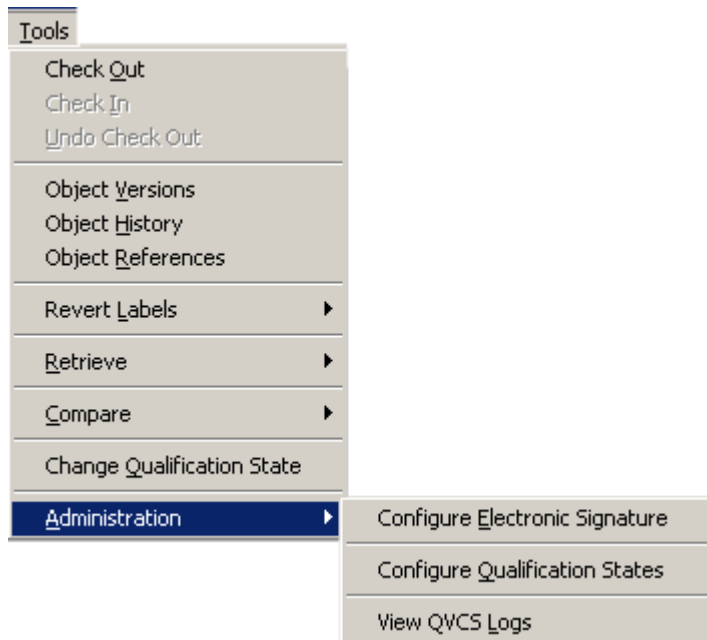
Compare Submenu



Menu Selection	Description	Available
<i>With Previous Version</i>	Compare the Selected object version with the previous object version.	This selection is available in the: <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window Also a single object should be selected for this menu to be available.
<i>With Another Version</i>	Compare the Selected object version with another object version.	This selection is available in the: <ul style="list-style-type: none"> • Versions Window Also a single object should be selected for this menu to be available.

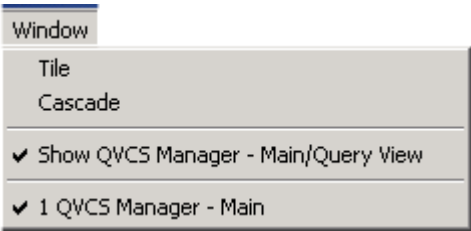
Menu Selection	Description	Available
<i>With Monitor version</i>	Compare the selected object version with the version in the Monitor tree of the Control Builder	This selection is available in the: <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window Also a single object should be selected for this menu to be available.
<i>With Checked Out version</i>	Compare the selected object version with the currently checked out version of the object in the Project/Library tree of the Control Builder.	This selection is available in the: <ul style="list-style-type: none"> • Main Window • Query Window • Versions Window Also a single object should be selected for this menu to be available.

Administration Submenu



Menu Selection	Description	Available
<i>Configure Electronic Signature</i>	Calls up the Configure Electronic Signature dialog.	Always available for the user with Administrative privileges.
<i>Configure Qualification States</i>	Calls up the Qualification State Configuration dialog.	Always available for the user with Administrative privileges.
<i>View QVCS Logs</i>	Calls up the QVCS Log window.	Always available for the user with Administrative privileges.

3.2.5 QVCS Window menu



Menu Selection	Description	Available
<i>Tile</i>	Tile all the opened windows	Always available
<i>Cascade</i>	Cascade all the opened windows	Always available
<i>Show QVCS Manager</i>	Bring the QVCS Manager window on top of all the windows	Always available
<i>Window1#8230;n</i>	List all the MDI windows opened.	Always available

3.2.6 QVCS Help menu

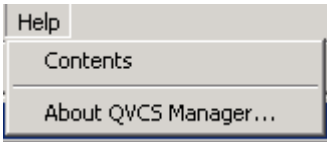
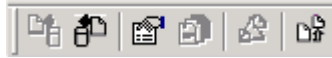








Figure 6: QVCS Help menu

Menu Selection	Description	Available
Contents	Launches the Control Builder Help and makes the main page of QVCS Manager window available. Also, F1 help is supported for the QVCS user interface.	Always available
About QVCS Manager	Shows the about dialog	Always available

3.3 QVCS Manager Toolbar

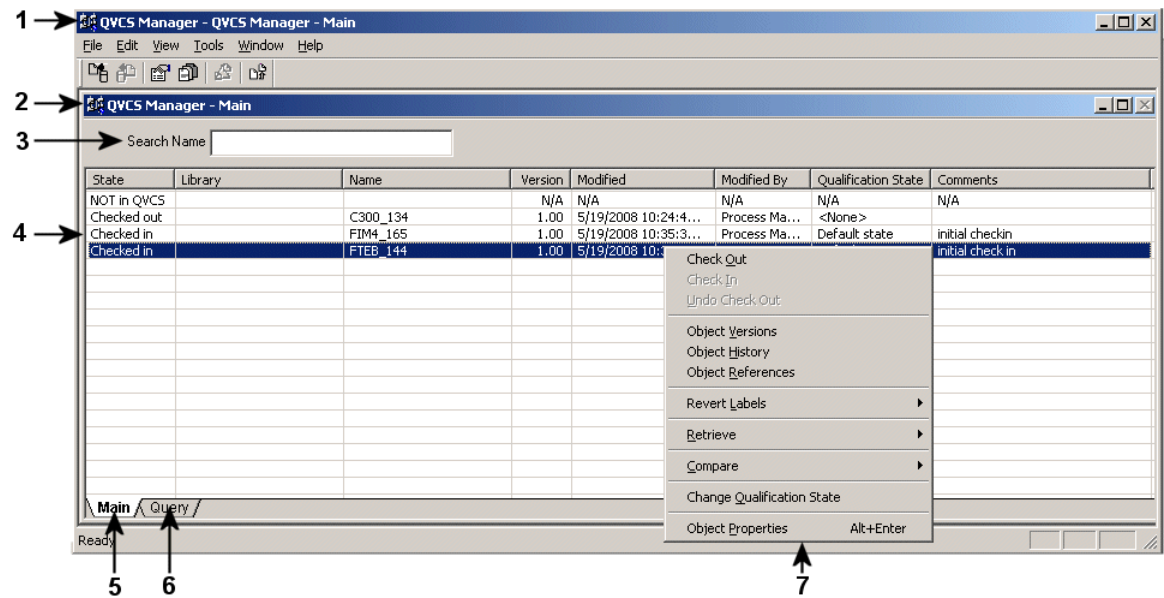
The toolbar provides convenient access to some of the frequently used QVCS Manager functions. You just click the button icon shown in the following illustration to invoke the associated function listed in the following table. The availability of the toolbar icon is directly related to the menu function it represents. For example, if a menu function is not available for a particular operation/user, then the corresponding toolbar button is also not available.



Toolbar Button Icon	Function Invoked	Related QVCS Manager Menu Function
	Check Out	Tools->Check Out
	Check In	Tools->Check In
	Object Properties	View->Object Properties
	Object History	Tools->Object History
	Compare Previous	Tools->Compare->with Previous version
	Revert	Tools->Revert Labels->Revert->To label

3.4 QVCS Manager Main Window

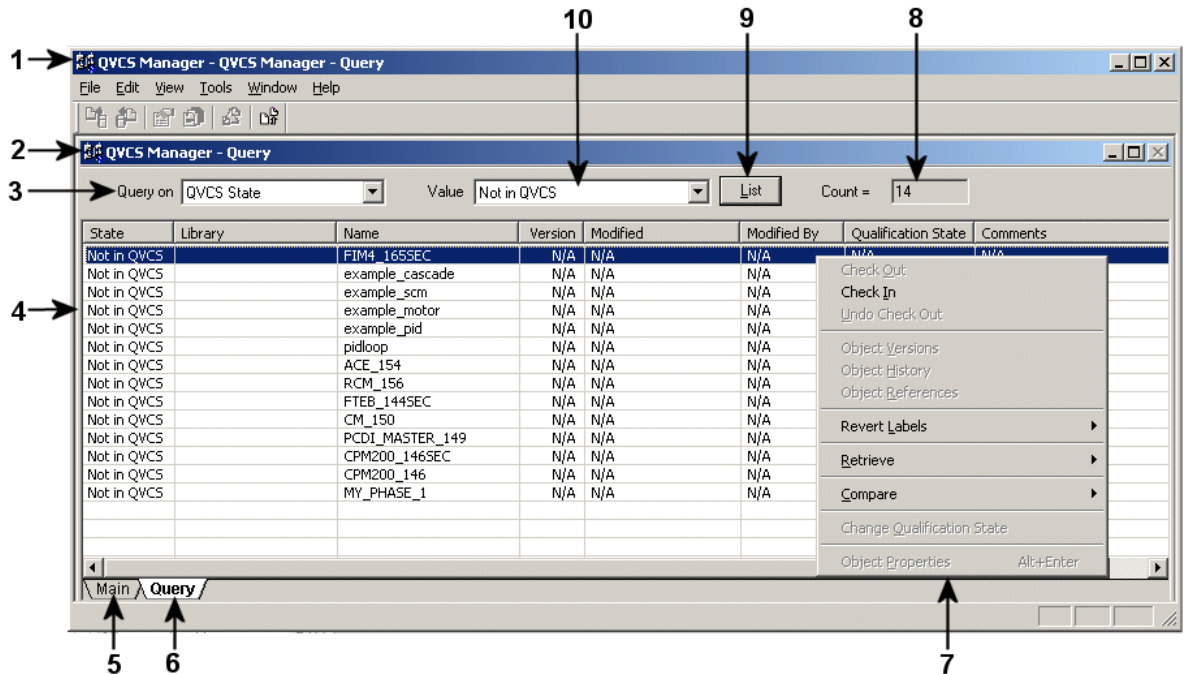
The Main window shows all the objects that you selected in Control Builder before launching the QVCS Manager as shown in the illustration example below. You initiate all Full-QVCS operations on a selected object from this view using either a menu selection, toolbar button, or context menu selection.



Callout	Item	Description
1	QVCS Manager	The main user interface for Full-QVCS system.
2	Main Window	You perform all QVCS operations on selected objects through this window. It displays all the objects that were selected in Control Builder tree before launching the QVCS Manager along with their version attributes in the list box.
3	Search Name	Lets you search for an object by name. Type in the name of the object in the field and the nearest matching object is moved to the top row in the list box. This is only available when list is sorted by name - default selection. Otherwise, click the name column heading to make it available.
4	List Box	<ul style="list-style-type: none">Lists all the objects selected in Control Builder in alphabetical order by their name.Click on a column heading to toggle the sort order between ascending and descending based on the selected column.The list does not automatically refresh when new objects are selected in the Control Builder tree. To update the list, return to Control Builder, select the new objects in Project tab to be shown, and launch the QVCS Manager again.You can use common shortcut keystrokes to select multiple objects in the list such as Click, Shift+Click and Ctrl+Click.
5	Main Tab	Click to call up the Main Window.
6	Query Tab	Click to call up the Query Window.
7	Context Menu	Right-click on an object for shortcut to QVCS Manager menu selections. See the previous <i>QVCS Manager Menus</i> section for more information.

3.5 QVCS Manager Query Window

The Query window shows all the objects in Control Builder that match the defined query selections as shown in the illustration example below. You initiate all Full-QVCS operations on a selected object from this view using either a menu selection, toolbar button, or context menu selection.



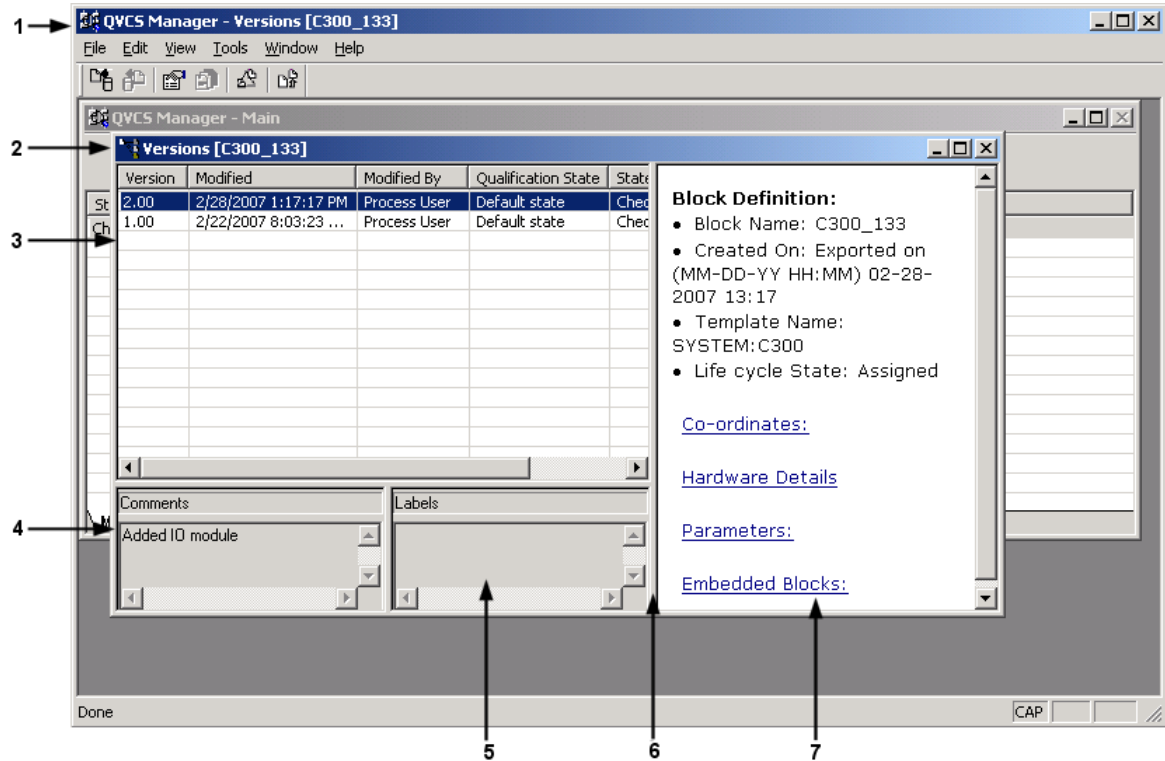
Callout	Item	Description
1	QVCS Manager	The main user interface for Full-QVCS system.
2	Query Window	You perform all QVCS operations on selected objects through this window. It displays all the objects that match the defined query selection found in Control Builder along with their version attributes in the list box.
3	Query on	Click the down arrow in the field to select one of the following options to query on: <div data-bbox="974 1344 1299 1512"> <p>Select query option</p> <ul style="list-style-type: none"> QVCS State Qualification State Qualification State, latest version Revert Label Reference </div>
4	List Box	<ul style="list-style-type: none"> Lists all the objects in Control Builder that match the defined query selection along with their version attributes. Click on a column heading to toggle the sort order between ascending and descending based on the selected column. You can use common shortcut keystrokes to select multiple objects in the list such as Click, Shift+Click and Ctrl+Click.
5	Main Tab	Click to call up the Main Window.
6	Query Tab	Click to call up the Query Window.
7	Context Menu	Right-click on an object for shortcut to QVCS Manager menu selections. See the previous <i>QVCS Manager Menus</i> section for more information.

Callout	Item	Description
8	Count	Shows number of objects detected in given query.
9	List Button	Click to initiate the query defined through the Query on and Value box selections.
10	Value	<p>Define the value for the Query on selection as follows:</p> <ul style="list-style-type: none"> For <i>QVCS state</i>, click the down arrow button to select one of the following values: <div data-bbox="971 436 1344 573" data-label="Image"> </div> For <i>Qualification State</i>, click the down arrow button to select from the user defined qualification state or the default state. Only the Default state appears, if the user has not configured any qualification states. For <i>Qualification State, latest version</i>, click the down arrow button to select from the user defined qualification state or the default state. Only the Default state appears, if the user has not configured any qualification states. For <i>Revert Label</i>, click the down arrow button to select from the user defined revert label or None. Only None appears, if the user has not configured any revert labels. For <i>Reference</i>, user must key in the object name to find references.

3.6 QVCS Manager Versions Window

The Versions window lets you view all the versions of the selected object in the Full-QVCS system. You can open a separate Versions window for each object selected in the Main or Query Window. You cannot open two version windows for the same object.

You initiate all Full-QVCS operations on a selected object from this window using either a menu selection, toolbar button, or context menu selection.



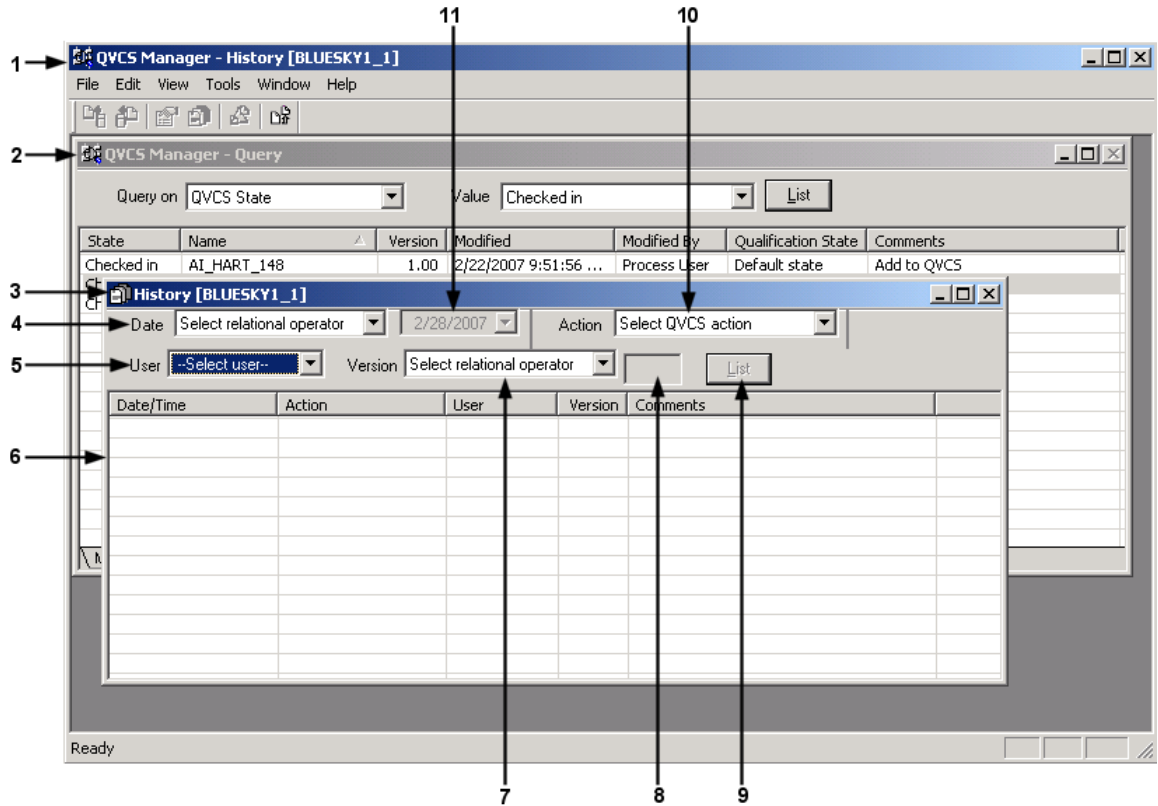
Callout	Item	Description
1	QVCS Manager	The main user interface for Full-QVCS system.
2	Versions Window	Displays all versions of the selected object.
3	List Box	<ul style="list-style-type: none"> Arranges all versions of the object in reverse chronological order or latest version first. Click on a column heading to toggle the sort order between ascending and descending based on the selected column. You can only select one version at a time in the list.
4	Comments Scroll Box	Shows the comments associated with the selected version.
5	Labels Scroll Box	Shows the labels associated with the selected version.
6	Splitter Bar	Click to move bar right or left to resize list box.
7	Details box	Shows the version details for the selected object in a text format.
<i>Items not shown in the illustration</i>		

Callout	Item	Description
	Context Menu for List Box	<p>Right-click on an object in the List Box to call up the following context menu.</p> <div><div>Check In</div><div>Check Out</div><div>Undo Check Out</div><div>Object PropertiesAlt+Enter</div><div>View Version Details</div><div>Revert Labels▶</div><div>Retrieve▶</div><div>Compare▶</div><div>Change Qualification State</div></div> <p>See the previous <i>QVCS Manager Menus</i> section for more information.</p>

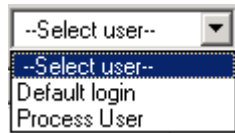
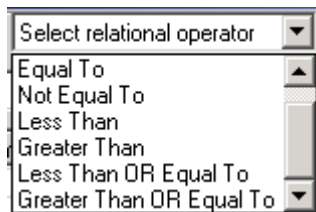
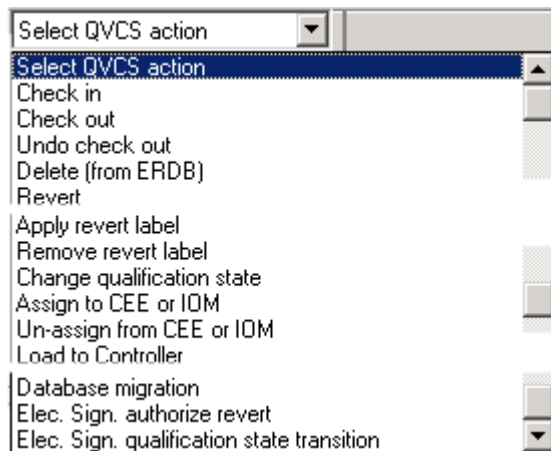
3.7 QVCS Manager History Window

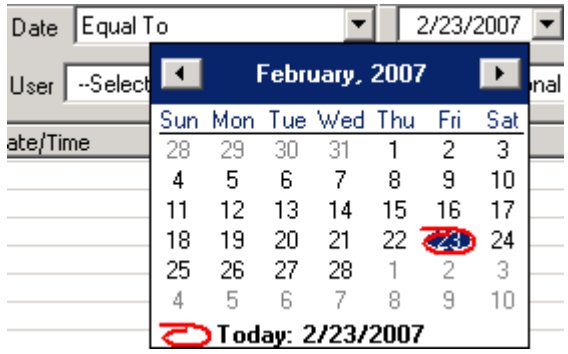
The History window lets you display a set of logs associated with the selected object in the Full-QVCS system based on user defined filter criteria. You can open a separate History window for each object selected in the Main or Query Window. You cannot open two History windows for the same object.

You initiate all Full-QVCS operations on a selected object from this window using either a menu selection, toolbar button, or context menu selection.



Callout	Item	Description
1	QVCS Manager	The main user interface for Full-QVCS system.
2	Query Window	See the previous “QVCS Manager Query Window” on page 35 section for details about the window.
3	History Window	Displays history data for selected object based on filter criteria you define through configuration entries on this window.
4	Date Box	Click the down arrow button to select one of the following relational operators to filter the selected date: <div data-bbox="974 1587 1286 1793" data-label="Image"> </div> <p>This filter is inactive when <i>Select relational operator</i> is configured.</p>

Callout	Item	Description
5	User Box	<p>Click the down arrow button to select desired user to be used as filter criteria for the selected object's history data as shown in the following example selections.</p>  <p>This filter is inactive when <i>Select user</i> is configured.</p>
6	List Box	<ul style="list-style-type: none"> Displays all the history logs for selected object based on the configured filter criteria. Arranges all logs in reverse chronological order or latest log first. Click on a column heading to toggle the sort order between ascending and descending based on the selected column. You can only select one log at a time in the list.
7	Version Box	<p>Click the down arrow button to select one of the following relational operators to filter the specified version:</p>  <p>This filter is inactive when <i>Select relational operator</i> is configured.</p>
8	Version Number Box	<p>Key in desired version number to be used as the reference for the relational operator selected in the Version box.</p> <p>This box is only available when a relational operator such as Equal To is selected in the Version box.</p>
9	List Button	<p>Click to initiate history log based on the selected filter criteria.</p> <p>This button is only available when one of the filter criteria boxes on this window is configured with a filter value.</p>
10	Action Box	<p>Click the down arrow button to select one of the following QVCS actions as filter criteria for the selected object's history data:</p>  <p>This filter is inactive when <i>Select QVCS action</i> is configured.</p>

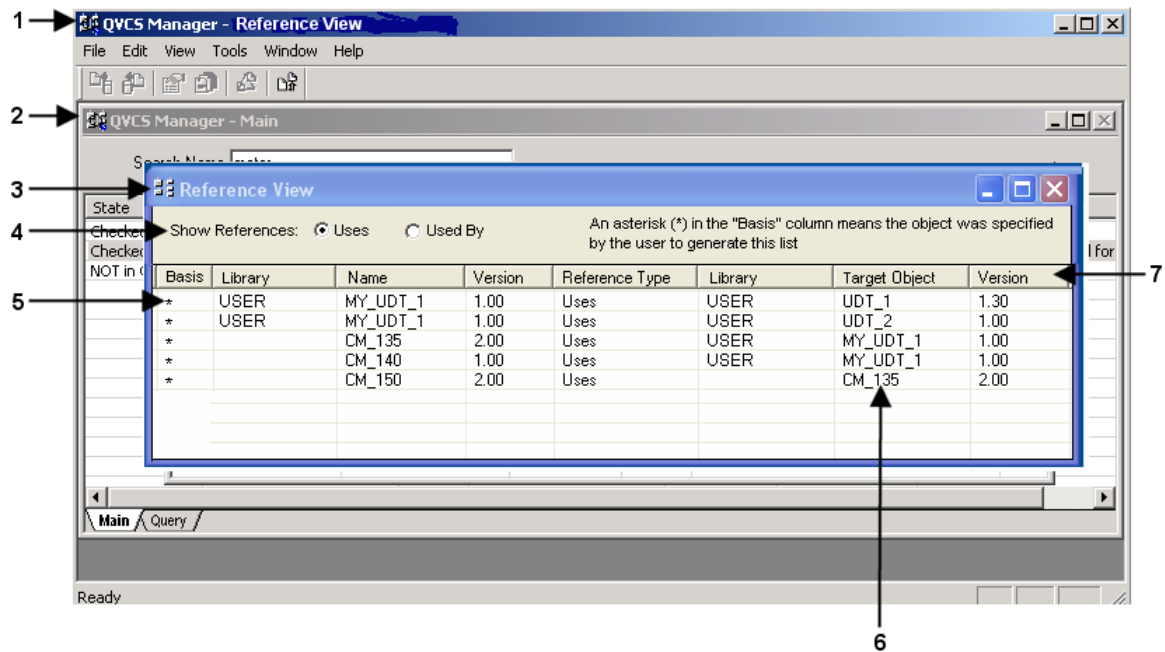
Callout	Item	Description
11	Date Calendar Box	<p>Click the down arrow button to select calendar data to be used with the selected Date relational operator as filter criteria for the selected object's history data:</p>  <p>This box is only available when a relational operator such as Equal To is selected in the Date box.</p>

3.8 QVCS Reference View Window

The References View window lets you display information about the version dependencies that exist to assist you in making decisions about reverting and checking out objects. You can select multiple objects as the basis for the reference view.

You can open the Reference View window from anywhere in QVCS where the right-click menu is supported, such as the Main window, Query window, and Versions window.

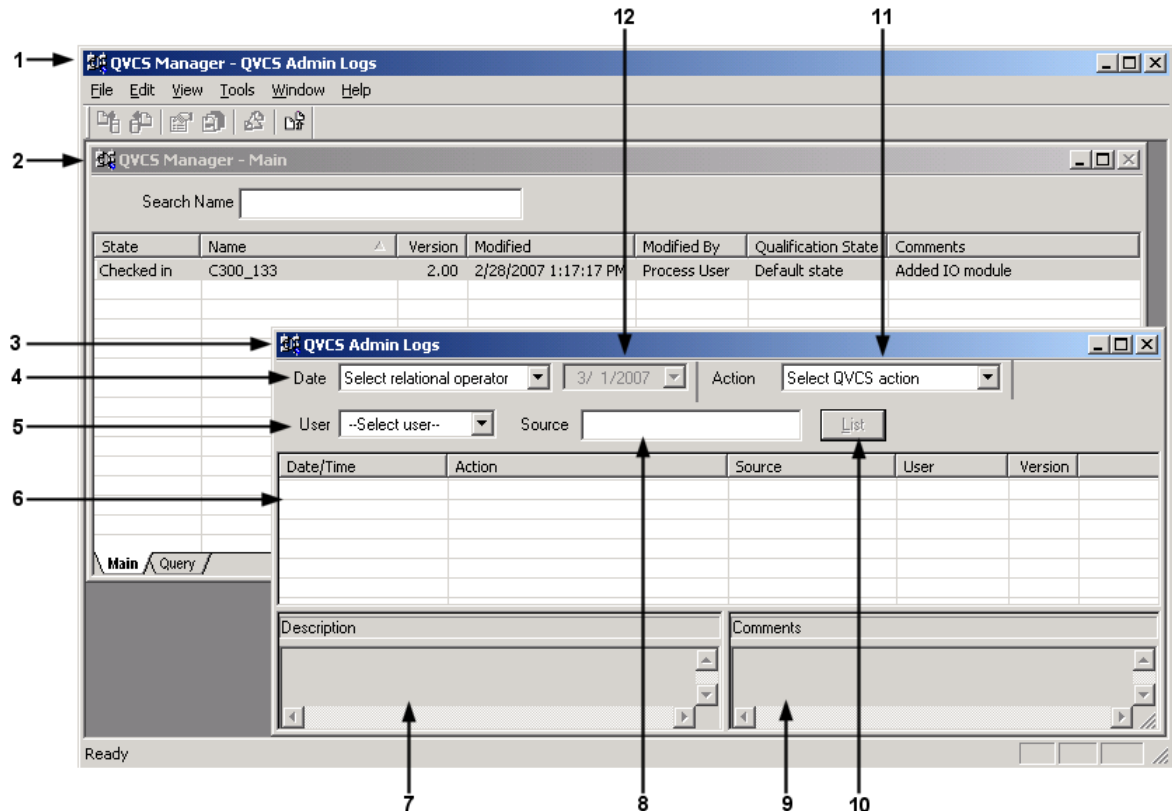
The following Reference View illustration is for example purposes only. The Reference View shows the entire *tree* of references from the starting object. In this illustration, the user defined template MY_UDT_1 uses two templates UDT_1 and UDT_2. The MY_UDT_1 is used by, which means there are instances of it, CM_135 and CM_140. The CM_135 is used by CM_150. The CM_150 is the root here.



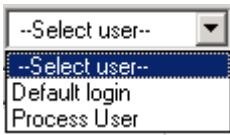
Callout	Item	Description
1	QVCS Manager	The main user interface for Full-QVCS system.
2	Main Window	See the previous “QVCS Manager Main Window” on page 34section for details about the window.
3	Reference View Window	Provides information about the version dependencies that exist.
4	Show References	User can select the type of reference to display. Type of reference indicates if the object uses another object, or is used by another object. A user defined template UDT can be both used by and use another object.
5	Basis (part of reference grid)	User can select multiple objects as the basis for the view.
6	Target Object (part of reference grid)	Multiple target objects can be selected and pasted into the Main window, where they can be selected for QVCS actions.
7	Version (part of reference grid)	Shows name and version of reference object.

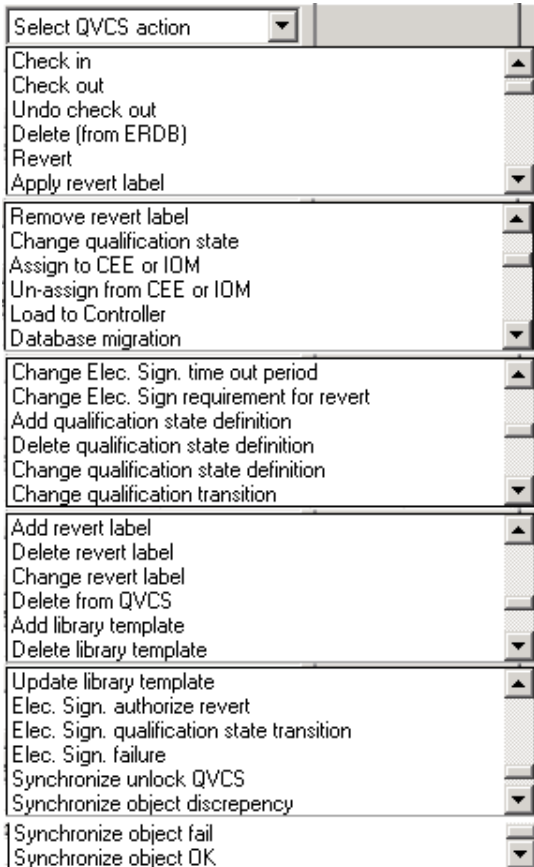
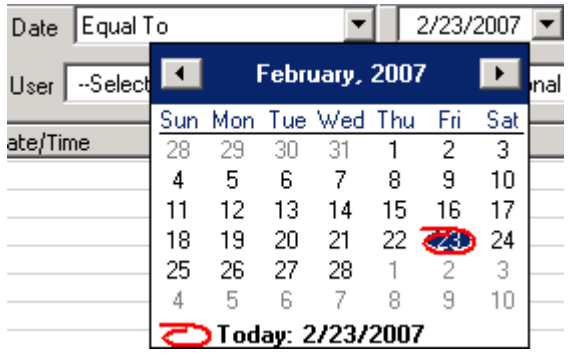
3.9 QVCS Manager QVCS Admin Logs Window

The QVCS Admin Logs window lets a user with **administrative** privileges display any logs that have been generated by the Full-QVCS system. The logs can include administrative logs, operational logs, and object version specific logs. You can filter the logs displayed by configuring given query options on the window. You can open only one instance of this window at a time.



Callout	Item	Description
1	QVCS Manager	The main user interface for Full-QVCS system.
2	Main Window	See the previous “QVCS Manager Main Window” on page 34 section for details about the window.
3	QVCS Admin Log Window	Displays QVCS logs based on filter criteria you define through configuration entries on this window. Only users with administrative privileges can access this window.
4	Date Box	Click the down arrow button to select one of the following relational operators to filter the selected date: <div data-bbox="974 1591 1286 1797" data-label="Image"> </div> This filter is inactive when <i>Select relational operator</i> is configured.

Callout	Item	Description
5	User Box	<p>Click the down arrow button to select desired user to be used as filter criteria for the selected object's history data as shown in the following example selections.</p>  <p>This filter is inactive when <i>Select user</i> is configured.</p>
6	List Box	<ul style="list-style-type: none"> • Displays all the logs generated based on the configured filter criteria. • Arranges all logs in reverse chronological order or latest log first. • Click on a column heading to toggle the sort order between ascending and descending based on the selected column. • You can only select one log at a time in the list.
7	Description Scroll Box	Displays any descriptive text associated with the selected log in the list box.
8	Source Box	Key in an object name or workstation name to be used as the source filter for the log query.
9	Comment Scroll Box	Displays any comments associated with the selected log in the list box.
10	List Button	<p>Click to initiate log query based on the selected filter criteria.</p> <p>This button is only available when one of the filter criteria boxes on this window is configured with a filter value.</p>

Callout	Item	Description
11	Action Box	<p>Click the down arrow button to select one of the following QVCS actions as filter criteria for the log query:</p>  <p>This filter is inactive when <i>Select QVCS action</i> is configured.</p> <p>See “Description of the QVCS Admin action” on page 46 for more information on the QVCS actions.</p>
12	Date Calendar Box	<p>Click the down arrow button to select calendar data to be used with the selected Date relational operator as filter criteria for the selected object's history data:</p>  <p>This box is only available when a relational operator such as Equal To is selected in the Date box.</p>

3.10 Description of the QVCS Admin action

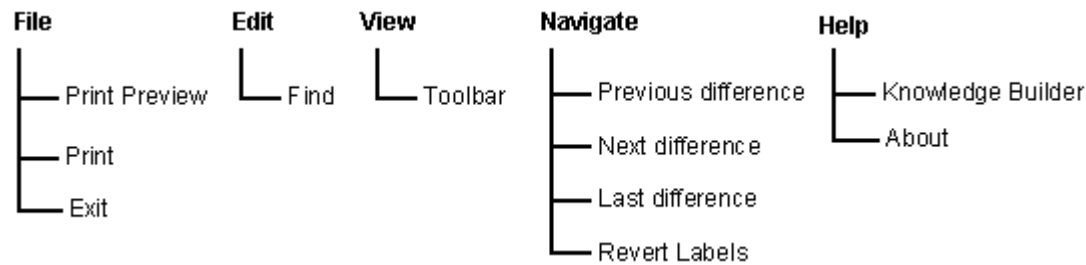
The following table lists the QVCS actions and a brief description of the possible actions that causes the log entry. This table also lists the messages that you can enter while performing any of the QVCS actions.

QVCS Action	Messages	Possible action that causes the log entry
Check in	Check in for:	When you check in an object/strategy into QVCS for version control.
Check out	Check out for:	When you check out an object for editing from QVCS.
Undo check out	Undo check out for:	When you cancel the check out action.
Delete (from ERDB)	Deleted:	When you delete the object from the (CB Tree) database.
Undo Delete	Delete undone for:	When you cancel the delete operation.
Revert	Reverted:	When you perform a revert to any individual version of an object or a set of objects that is grouped together with a user defined label.
Apply revert label	Applied revert label:	When you apply revert label for objects checked into QVCS.
Remove revert label	Removed revert label:	When you delete revert labels for objects checked into the QVCS.
Change qualification state	State transition:	When you change or modify the qualification state of an object in the QVCS.
Assign to CEE or IOM	Assigned:	When you assign a QVCS object/strategy to a CEE or IOM.
Un-assign from CEE or IOM	Un-assigned:	When you unassign a QVCS object/strategy from a CEE or IOM.
Load to Controller	Loaded:	When you load a QVCS object to a controller.
Database migration	Database migrated:	When a database migration happens.
Change Elec. Sign. time out period	Time out period (minutes) changed to:	When you change electronic signature time out period during configuring electronic signature.
Change Elec. Sign. retries	Number of retries changed to:	When you change the number of retry limit of electronic signature.
Change Elec. Sign requirement for revert	Signature requirement for revert changed to:	When you change the electronic signature requirement for revert.
Add qualification state definition	Qualification state added:	When you define a new qualification state.
Delete qualification state definition	Qualification state deleted:	When you delete the qualification state.
Change qualification state definition	Qualification state changed	When you change transition rules for the selected qualification state.
Change qualification transition	Transition rules changed for qualification state:	When you change qualification state for a QVCS object.
Add revert label	Added revert label:	When you add a new revert label in QVCS system. The new revert label is added to the revert label list box.
Delete revert label	Deleted revert label:	When you select an existing revert label to delete from the revert label list box.
Change revert label	Changed revert label.	User changes the configuration of a revert label.
Delete from QVCS	Deleted from the QVCS:	When you delete object from QVCS query window.
Add library template	Added:	When you create a new custom block type template in CB Library tree. The library name and the template name are added to QVCS logs.

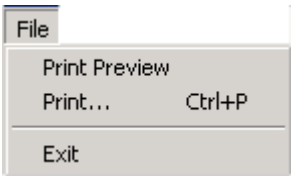
QVCS Action	Messages	Possible action that causes the log entry
Delete library template	Deleted:	When you delete CBT library from CB Library tree.
Update library template	Updated:	When you update an existing CB Library tree.
Elec. Sign. authorize revert	Revert authorized by:	User name that electronically authorized revert operation.
Elec. Sign. qualification state transition	Qualification state transition authorized by:	User name that electronically authorized change qualification state for a QVCS object.
Elec. Sign. failure	Time out, retries exceeded, invalid user/ password, or no privilege:	When electronic signature fails or cancels.
Synchronize unlock QVCS	Synchronize unlocked QVCS:	When you synchronize the QVCS database with the Engineering Repository database and clear any QVCS locks through Configuration Studio. If the QVCS db is locked, QVCS adds a log 'Synchronize unlock QVCS' which says system is unlocking QVCS DB.
Synchronize object discrepancy	Synchronize found discrepancy between DBs for:	When discrepancies found between databases during synchronization.
Synchronize object fail	Synchronize could not correct discrepancy for:	When synchronization fails for QVCS object.
Synchronize object OK	Synchronize corrected discrepancy for:	When synchronization passes for object.

3.11 QVCS Difference Tool (Diff Tool) Menus

The following illustration shows the menu tree for the QVCS Diff Tool.

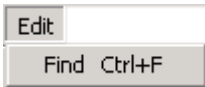



3.11.1 Diff Tool File Menu



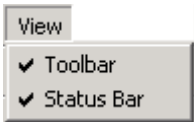
Menu Selection	Description	Available
<i>Print Preview</i>	Shows the preview of the contents of the active window that will be sent to the printer.	Always available
<i>Print</i>	Prints the current active window contents. Brings up the Print dialog box to choose the printer and the contents is printed. Assume that printer is connected to the server.	Always available
<i>Exit</i>	Closes the Diff Tool.	Always available

3.11.2 Diff Tool Edit Menu



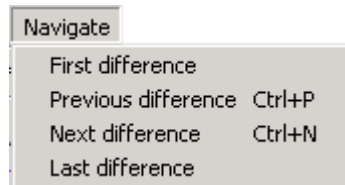
Menu Selection	Description	Available
<i>Find</i>	Calls up the standard Windows Find text dialog shown below.	Always available
		

3.11.3 Diff Tool View Menu



Menu Selection	Description	Available
<i>Toolbar</i>	Show/Hide the Diff Tool toolbar. A check mark indicates that the toolbar is shown. Menu item works like a check box.	Always available
<i>Status Bar</i>	Show/Hide the Diff Tool status bar. A check mark indicates that the toolbar is shown. Menu item works like a check box.	Always available

3.11.4 Diff Tool Navigate Menu



Menu Selection	Description	Available
<i>First difference</i>	Moves the current window selection to the first difference in the list of differences found by the Diff Tool	<ul style="list-style-type: none"> Available, if there are any differences between the versions. Not available, if no differences are found between versions.
<i>Previous difference</i>	Moves the current window selection to the previous difference from the current difference position.	<ul style="list-style-type: none"> Available, if there are any differences between the versions. Not available, if no differences are found between versions. Not available, if the current selection is the first difference.
<i>Next difference</i>	Moves the current window selection to the next difference from the current difference position.	<ul style="list-style-type: none"> Available if there are any differences between the versions. Not available, if no differences are found between versions. Not available, if the current selection is the last difference.
<i>Last difference</i>	Moves the current window selection to the last difference in the list of differences found by the Diff Tool	<ul style="list-style-type: none"> Available, if there are any differences between the versions. Not available, if no differences are found between versions.

3.11.5 Diff Tool Help Menu

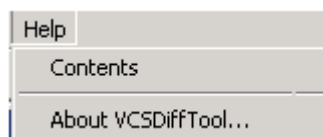








Figure 7: Diff Tool Help menu

Menu Selection	Description	Available
Contents	Launches the Control Builder Help and makes the main page of QVCS Manager window available. Also, F1 help is supported for the QVCS user interface.	Always available
About VCSDiffTool	Shows the about dialog	Always available

3.12 QVCS Diff Tool Toolbar

The toolbar provides convenient access to some of the frequently used Diff Tool functions. You just click the button icon shown in the following illustration to invoke the associated function listed in the following table. The availability of the toolbar icon is directly related to the menu function it represents. For example, if a menu function is not available for a particular operation/user, then the corresponding toolbar button is also not available.

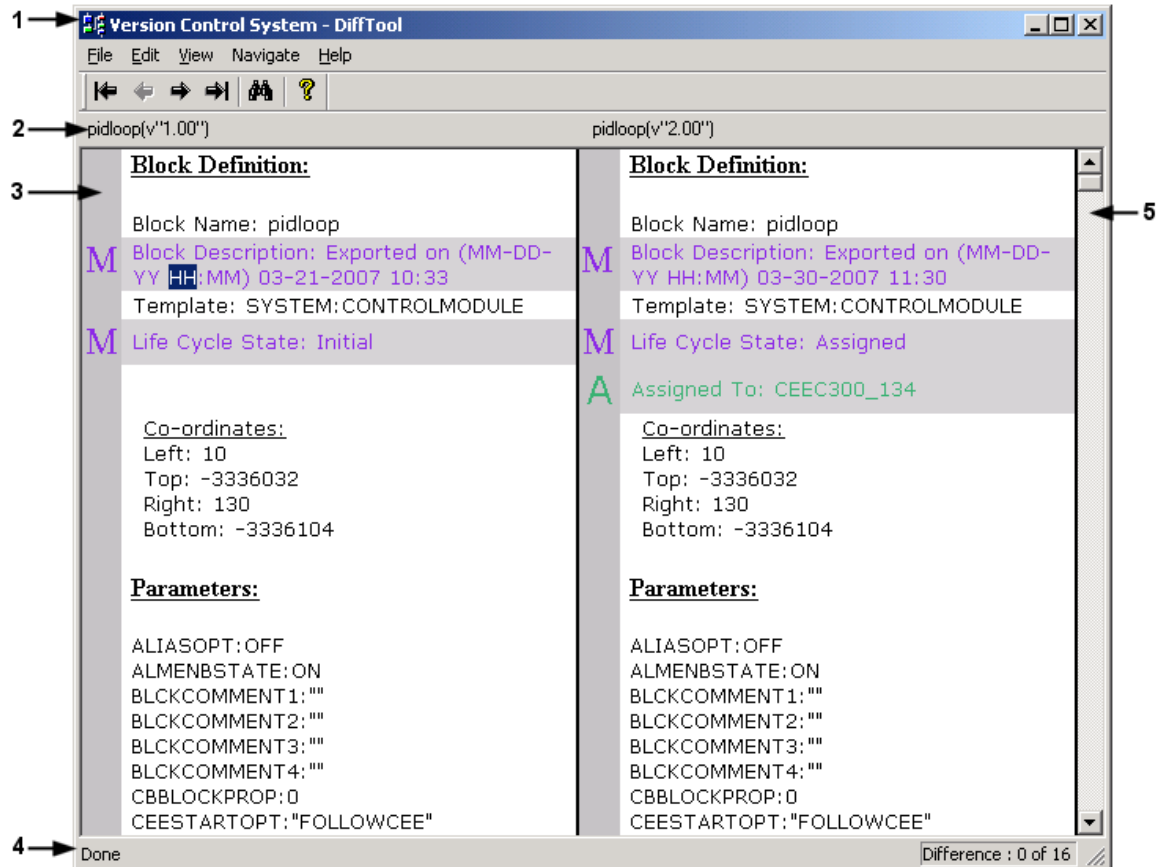


Toolbar Icon	Tool Tip	Menu Item
	First difference	Navigate->First difference
	Previous difference	Navigate->Previous difference
	Next difference	Navigate->Next difference
	Last difference	Navigate->Last difference
	Find	Edit->Find
	Help	Help->Contents

3.13 QVCS Difference Tool (Diff Tool) Window

You can use the QVCS Version Difference Tool (Diff Tool) to view and compare differences between versions of a QVCS object. The Diff Tool functionality is included when a QVCS license is purchased.

Typically, a compare operation is performed on two versions of a single object. The Diff Tool shows the differences between the two versions, highlighting additions to the strategy in green, modifications in blue, and deletions in red.



Callout	Item	Description
1	Diff Tool Window	You can call up the Diff Tool window through the QVCS Manager Tools menu selection Compare->With . . . , as applicable.
2	Version Number	Displays the version number of the object whose data is being displayed in the list box below.
3	Version List Box	Displays the data associated with the given version number. <ul style="list-style-type: none"> The letter A and green text identifies data that has been added between the selected versions. The letter D and red text identifies data that has been deleted between the selected versions. The letter M and blue text identifies data that has been modified between the selected versions.
4	Status Bar	Shows the status of the current action.
5	Scroll Bar	Lets you scroll data in both list boxes, which are synchronized with respect to scroll bar position.

4 QVCS Support for CBT and UDT Overview

This section provides a summary of the rules that govern the behavior of CBTs and UDTs beginning in R400.

In Experion releases prior to R400, QVCS did not support the versioning of CBTs. It only kept track of instances of CBTs that exist in objects that are placed in QVCS, such as CM, SCM, or RCM. Once an object containing a CBT was checked into QVCS, the CBT was *frozen* and the following rules were enforced by QVCS.

- The CBT cannot be edited and saved with the same name
- The CBT cannot be renamed
- The library in which the CBT is located cannot be renamed
- The CBT cannot be deleted (or, as a consequence, the library where it is located)

4.1 Basic rules for UDT support in QVCS

The basic rules for UDT and QVCS are as follows and they are very similar to the rules to be enforced for CBT and QVCS starting in R400.

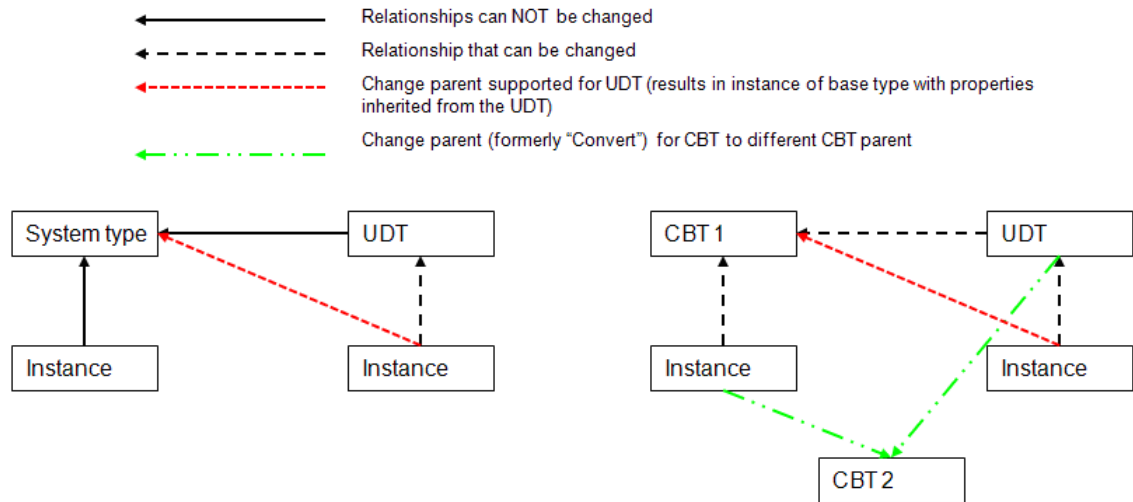
- An object containing an instance of a UDT or an object that is an instance of the UDT cannot be checked in unless the UDT is in a state of checked in (or is included in the same check in operation as the instance). An UDT can be based on a container, such as a CM, SCM, and so on., and thus can be a versionable object; or a UDT can be derived from another UDT and can thus be a versionable object.
- When a UDT is checked out, all of the objects containing instances of the UDT are automatically checked out with it.
 - A consequence of this is that all of the instances must be in a state of checked in prior to the UDT check out, or be included in the same check out operation
- When an object containing an instance of an UDT or an object that is an instance of an UDT is checked out or reverted, QVCS checks to determine if the version of the UDT that the instance uses exists in the **Library**. If it does not, then the user is asked if they want to: “keep” or “break” the relationship with the UDT.

These rules are in addition to the standard rules that QVCS enforces for all objects, such as:

- Once an object has been placed into QVCS it cannot be renamed,
- An object must be in the checked in state to be deleted,
- An object must be checked out to be edited,
- And so on.

4.2 Differences between CBT and UDT

Although there are many similarities between CBT and UDT, there are two significant differences. The first is that unlike an instance (in this case, an UDT is the instance) of a system type where the instance cannot have its relationship with the type changed, it is possible to change the “type” for instances of CBTs, as illustrated in the following diagram.



The second difference between a CBT and a UDT is that unlike a UDT, where the parameters are determined by the type (either system or CBT) on which it is based, it is possible to change the parameter definitions in a CBT. Because QVCS uses the import/export function to transfer object definitions between the ERDB and the QVCS database, changes to object definitions must adhere to migration rules. The migration rules do not allow a “major” change to parameters, where a major change is any one or more of the following:

- Parameter deletion
- Parameter rename
- Parameter data type change
- Parameter dimension change

4.3 Specific rules for CBT support in QVCS

The QVCS must enforce rules on CBT that are more restrictive than the rules for UDT to insure that the migration rules will not be violated when performing check out and revert operations. Since QVCS does not track the specific changes to an object, it is not possible for QVCS to know whether a change to a CBT is a “major” change or not. And, since QVCS will not invoke the Change Parent (Convert) function in the R400 release, the following rules will be enforced by QVCS for CBT and CBT instances:

- It is not possible to check out an earlier version of a CBT, even when no instances of the CBT exist anywhere. This is because even though the check out might not directly violate any version dependency rules, it can lead to a state where a later version of a CBT has fewer parameters (and/or different parameters) than an earlier version. See “CBT and UDT check out considerations” on page 66 for more information.
- It is not possible for QVCS to revert an earlier version of a CBT if there are instances of the CBT in the ERDB (since it is not known if this earlier version has fewer parameters, and so on.). For Example, consider the case where Version 1.0 of CBT contains 1 parameter, and Version 2 contains 2 parameters. And, in the ERDB, there is an instance of Version 2 of the CBT. Reverting to older version of CBT (to Version 1.0) would lead to fewer parameters than what the instance has. This violates the migration rules that types cannot have parameters deleted, and so would be disallowed.
 - If the CBT has been deleted from the **Library**, a revert operation can restore it. This is possible since no instances exist if the CBT has been deleted
- It is not possible for QVCS to check out or revert an instance of a CBT to the ERDB if its version dependency on the CBT is greater than the version of the CBT in the **Library** (since it is not known if the “older” version of the CBT in the **Library** would have fewer parameters than what the instance requires, and so on.)

These rules will be clarified in subsequent sections of this document when describing various scenarios, but a brief example follows. A term that can be used to describe the effect of the rules is that following any QVCS operation such as check in, check out, revert, and undo check out, a CBT and its instances in the **Project** and **Library** trees (**Library** is included since a UDT can be created from a CBT) must be a “*version consistent*” set. To maintain the version consistency of the CBT and its instances, QVCS maintains a version dependency table that it uses when performing these types of operations. An example version dependency is shown in the following table:

Table 2: Example Version Dependency

Version of RESIN containing PREMIX1. (PREMIX1 is an instance of CBT PREMIX)	Version of PREMIX used by the RESIN RCM
1.0	1.0
2.0	1.0
3.0 *	2.0 *
4.0	3.0
5.0	3.0
6.0	4.0

The versions with an asterisk are currently in the **Library** and **Project** trees for the CBT and its instance. Note that it is possible for different versions of an object to use the same version of a CBT. In this example, version 1.0 and 2.0 of RESIN both use version 1.0 of PREMIX. It is not possible for multiple versions of a CBT to be used by a single version of an object. This restriction applies since only a single version of a CBT may exist in the **Library**. This rule also implies that all instances in the **Project** and **Library** (that is, UDT based on CBT) must also reference the same CBT version (same rule as for UDTs and their instances).

On subsequent check out, revert, and undo check out actions, QVCS compares the version numbers of the instance and the CBT that are in the trees and that are having the QVCS action performed to determine what is allowed and not allowed. In this example, given that RESIN v3.0 is in the **Project** tree, and PREMIX v2.0 is in the **Library**, if the user wishes to place PREMIX v4.0 in the **Library** (through check out or revert), this will

not be allowed since it violates the rule that an instance cannot have a dependency on a CBT version that is greater than what is in the **Library**. The details of how the check out, undo check out, and revert actions handle these conditions are detailed in subsequent sections of this document. The following diagram illustrates the version dependency rules.

This diagram is based on the version dependency information in the preceding table, and where the current contents of the **Project** and **Library** trees are as shown. Thus, if version 2.0 of PREMIX is in the **Library**, you cannot bring version 4.0, 5.0, or 6.0 of RESIN back to the **Project** tree since that would violate the version dependency rules.

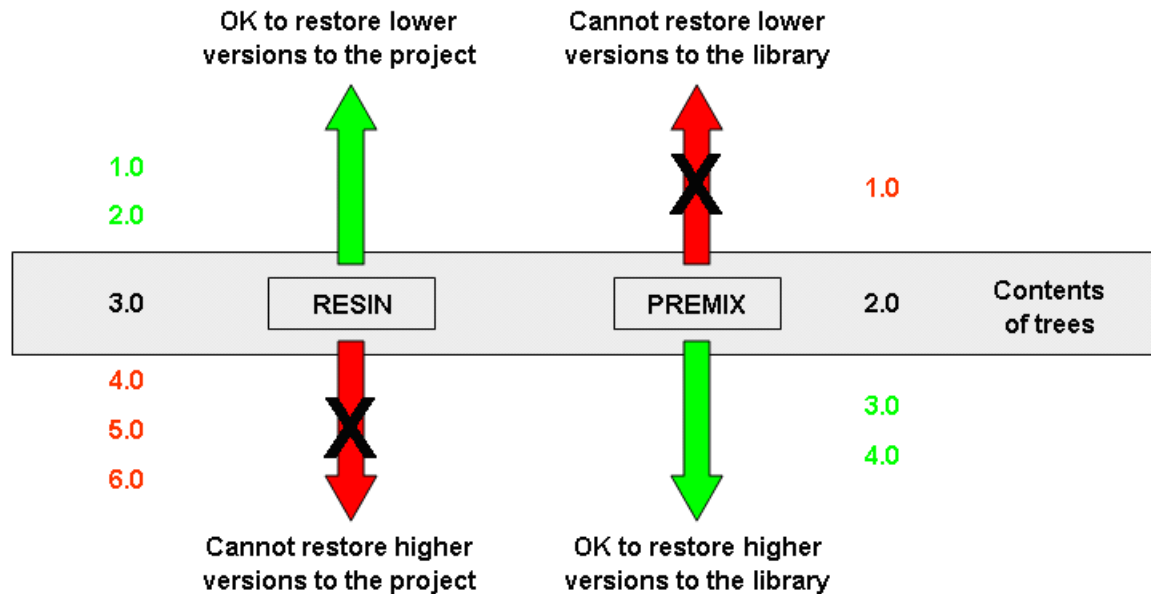


Figure 8: Example of Version Consistent Sets

5 QVCS General Operations

This section provides procedural references for doing general Full-QVCS operations such as checking in an object, checking out an object and viewing object properties.

Related topics

- “Checking in an object to QVCS” on page 60
- “Checking out an object from QVCS” on page 66
- “Undoing Check out from QVCS” on page 77
- “Graphical Example of Layered Recipe Check In and Check Out Scenario” on page 81
- “Instantiating CBT” on page 84
- “Viewing and Changing Object Properties” on page 87
- “Making changes to an object” on page 89
- “Making multiple object qualification state transitions” on page 90
- “Renaming an object” on page 92
- “Deleting a versionable object” on page 93
- “Restoring Deleted Object” on page 97
- “Editing CBT in QVCS” on page 99
- “Using Change Parent with CBT” on page 102
- “Viewing Information for a Version of an Object” on page 104
- “Viewing History details of an Object” on page 106
- “Viewing Reference data for Objects” on page 107
- “Example Reference View Scenario” on page 110
- “Retrieving Version of an Object” on page 112
- “Comparing Versions of an Object Using Diff Tool” on page 116
- “Loading QVCS Objects to a Controller” on page 117
- “Using upload or update to project commands” on page 120
- “Importing Objects” on page 122
- “Exporting Objects” on page 127

5.1 Checking in an object to QVCS

- When an object/control strategy is first created in Control Builder, QVCS has no knowledge of its existence until it is checked in to QVCS. Once an object is checked in, it is under version control and cannot be un-versioned. The object is still considered to be under version control even when it is checked out.
- Users are asked to supply the following information when checking in an object.
 - Must enter a comment describing the reason for the given action. The comment cannot be changed after the check in is completed.
 - Must classify any revisions made as either major or minor. A connector change or block deletion would be an example of a major change.
 - If applicable, select a revert label. You can change the revert label at any time for an object that is checked in through the View->Object Properties menu selection in the QVCS Manager.
- The first time an object is checked in to QVCS its version number is changed from 0.0 to 1.0. On subsequent check ins, the version number is incremented by one hundredth (.01) for a **minor** change or rounded up to the next integer for a **major** change. For example, version number 1.2 is rounded up to 2.0. While manual override of the version number is not permitted, it is possible to assign a revert label to a given version.
- QVCS does not keep track of parameter connections, module assignments to Controller/CEE's or I/O channel associations to I/O modules. However, the last two actions are logged in the QVCS audit trail, but not versioned. A particular version of a Controller/CEE does not contain information about how many or which versions of objects are assigned to it. If you revert to an earlier version of a Controller/CEE, the current assignments remain intact - Unless the current version is unassigned, and then the assignment is obtained from the version being reverted. In the same manner, I/O Module versions do not contain any data about I/O Channel associations. If knowing the state of assignments and associations at a particular point in time is important to you, you must view the QVCS log for the particular Controller/CEE or I/O Module. Note that assigning or associating requires the object being assigned or associated to be checked out, and thus will be assigned a new version number when they are checked in.

5.1.1 Multiple object check in considerations

You can select multiple objects for check in at the same time. The multiple objects are treated as a single group and the check in information, such as comment and minor or major revision, applies to the group as whole. Before you check in multiple objects, be sure you review the following considerations.

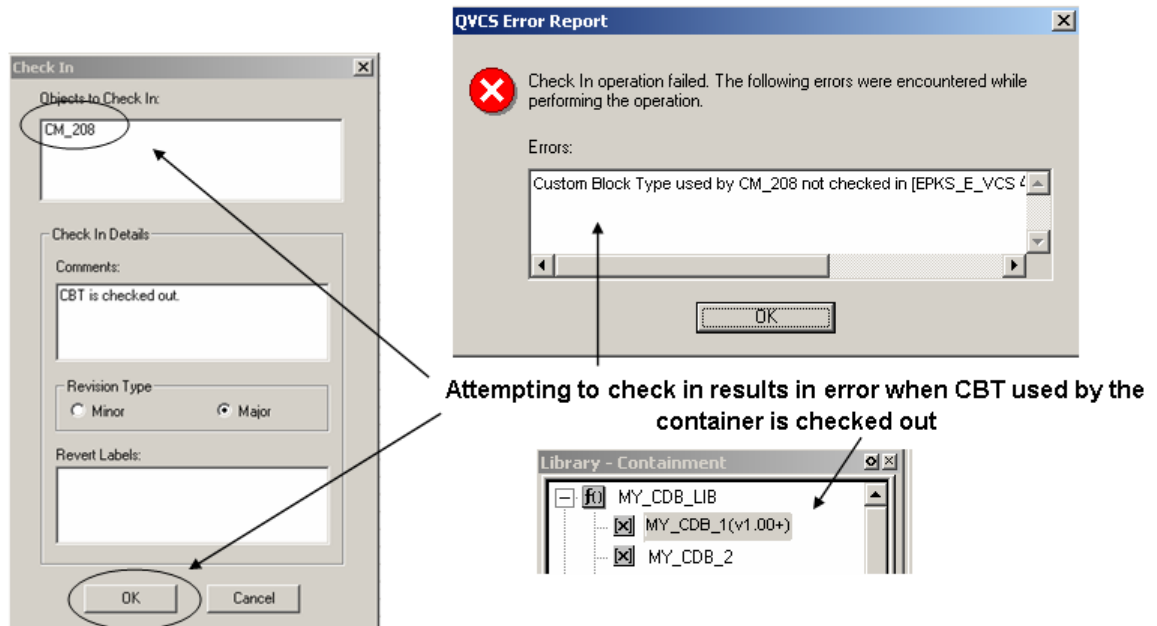
- The major/minor revision selection applies to the group as a whole. If the group contains objects that had structural changes as well as objects without structural changes, you must decide whether a major or a minor revision increment is to be applied to the entire group. Each object in the group retains its own version number that is consistent with the type of revision selected. You can apply a revert label to group the versions. If you specify a revert label on a multiple object check in, that label is applied to the new versions of all the objects being checked in.
- If any of the objects that are selected for multiple check in are not checked out, are locked by another user, or were not checked out by the current user, QVCS generates a notice that includes the reason for the error and the user name for each object that cannot be checked in as applicable. None of the selected objects will be checked in. An example notice could be "Cannot check in CM_01, it is checked out by operator01". All notices will then be displayed in a single error. The user is responsible for taking the appropriate corrective action such as changing the selection to only include the objects that can be checked in.

5.1.2 CBT and UDT check in considerations

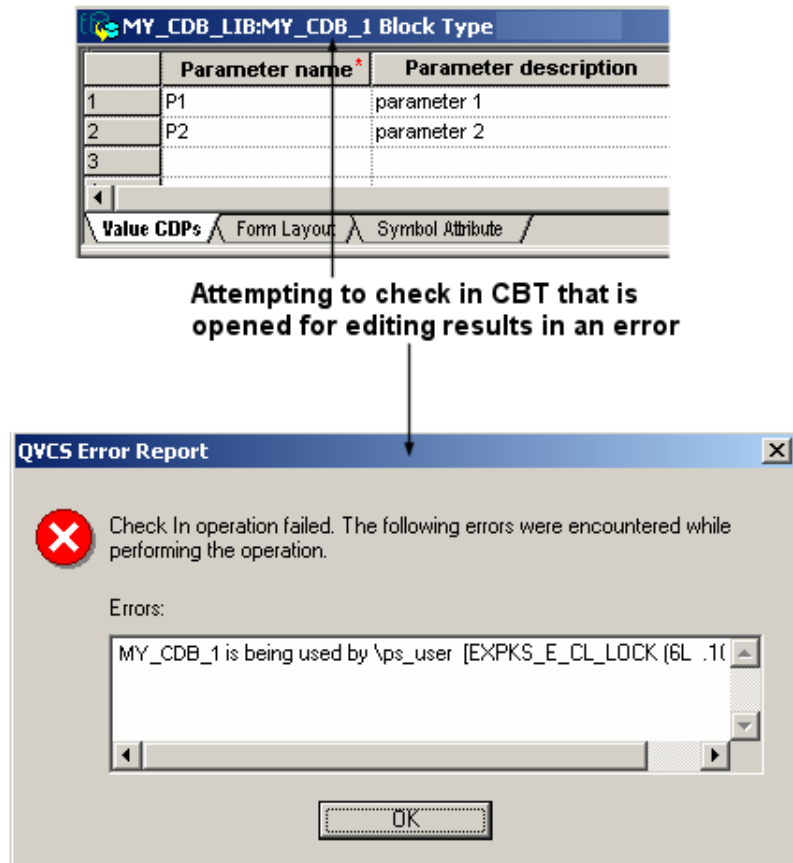
QVCS saves the version number of a user defined template (UDT) or custom block type (CBT) with any derived templates/types or instantiations of the template/type to keep track of UDT and CBT dependencies. Be sure to review the following considerations for more information.

- You can check in CBT or UDT independently of instances.

- You cannot check in instances of CBT or UDT unless the CBT or UDT is checked in either previously or as part of the current check in as illustrated the following graphic example.

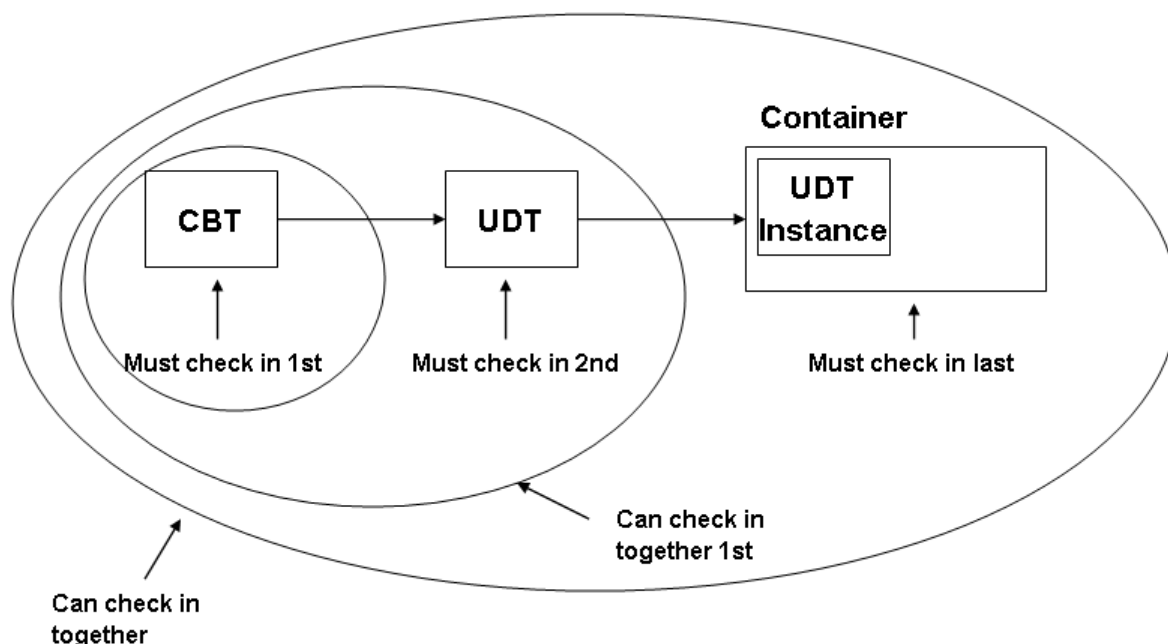


- You cannot check in a CBT, if it is opened for editing as illustrated in the following graphic example.



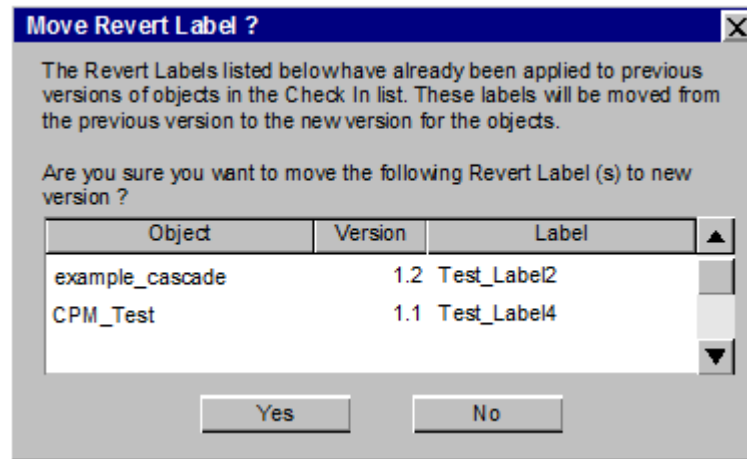
- When an object containing an instance of a CBT is checked in, QVCS records the CBT version "Dependencies" of that object.

- The dependencies with CBT and UDT can become quite complex. The rule is that you must check in the *parent* before or at the same time as the *child*, as illustrated in the following graphic.

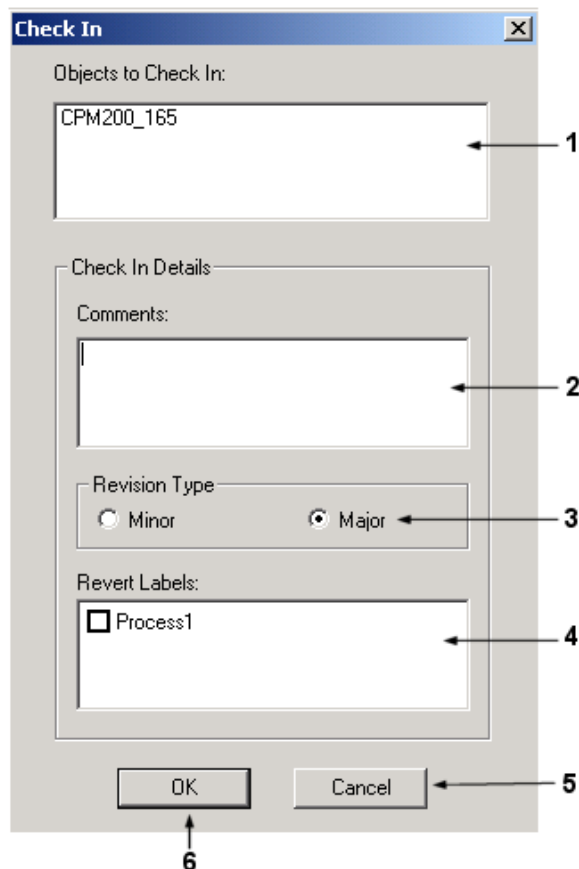


5.1.3 Checking in an object

- 1 Select one or more new or checked out objects in the **Project** or **Library** tab of Control Builder to be added or checked in to QVCS.
- 2 In the **Tools** menu, click **QVCS Manager** to call up the QVCS Manager Main window.
You can alternately right-click selected object and select **QVCS Manager** from the list or click the QVCS button in the toolbar to call up the QVCS Manager Main Window.
- 3 Select one or more *Not in QVCS* or *Checked Out* objects in the list box of the Main window.
- 4 In the **Tools** menu of the QVCS Manager, click **Check In** to call up the Check In dialog as shown in the following Figure.
You can alternately right-click selected object and select **Check In** from the list or click the Check In button in the toolbar to call up the Check In dialog.
- 5 Confirm that the objects to be checked in appear in the list box at the top of the dialog.
- 6 You must key in information about this check in in the Comments list box. This is mandatory, since check in will not continue if a comment is not entered.
- 7 Select the applicable revision type as either **Minor** or **Major**.
- 8 If applicable, select one or more revert labels to be applied to the checked in versions of the objects.
See the *Revert Label Operations* section in this document for more information about adding, deleting, and modifying revert labels.
- 9 If any of the selected labels have been applied on another version of any of the selected objects for check in, a Move Revert Label dialog appears asking for confirmation of the move as shown in the example illustration below. Otherwise, go to the next Step.



- Click the **Yes** button to okay the move and initiate the check in.
 - Click the **No** button to return to the Check In dialog and modify revert label selections as required.
- 10 Click the **OK** button to initiate the check in operation and close the Check In dialog. Or, click the **Cancel** button to abort the check in function.
- 11 This completes the procedure.



Callout	Item	Description
1	Objects to Check In	Shows list of objects that have been selected for check in.

Callout	Item	Description
2	Comments	You must key in a comment about this check in. This is mandatory for the check in function to continue.
3	Revision Type	Lets you classify the type of revisions made as either Minor or Major.
4	Revert Labels	If applicable, lets you assign existing revert labels to the objects.
5	Cancel Button	Click to abort the check in function.
6	OK Button	Click to initiate the check in function.

5.1.4 Example check in scenarios

The following table outlines various check in scenarios based on the “Graphical Example of Layered Recipe Check In and Check Out Scenario” on page 81 provided in another section of this document. Refer to the Control Builder Error Codes Reference document for more information about listed Error Codes.

Table 3: Layered Recipe Scenarios for Check In

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (To Be Checked In)	CBT Instance Only (To Be Checked In)	CBT + Instance (To Be Checked In)
1	PREMIX (Not in QVCS)	RESIN (Not in QVCS)	Allowed	Not allowed (Error Code 14709)	Allowed
2	2.0 (Checked In)	3.0 (Checked In)	Cannot Check in CBT (Error Code 14627)	Cannot Check in Instance (Error Code 14627)	Cannot Check in CBT and Instance. (Error Code 14627)
3	2.0 (Checked In)	3.0 (Checked Out)	Cannot Check in CBT (Error Code 14627)	Allowed	Cannot Check in CBT and Instance (Error Code 14627)
4	2.0 (Checked Out)	3.0 (checked In)	Not Possible - Instance cannot be Checked In if CBT is Checked Out	Not Possible - Instance cannot be Checked In if CBT is Checked Out	Not Possible - Instance cannot be Checked In if CBT is Checked Out
5	2.0 (Checked Out)	3.0 (Checked Out)	Allowed	Not allowed (Error Code 14710)	Allowed
6	2.0 (Checked In)	Instance 3.0 is loaded to Controller and is Checked In.	Cannot Check In Type (Error Code 14627)	Cannot Check In Instance (Error Code 14627)	Cannot Check In CBT and Instance (Error Code 14627)
7	2.0 (Checked Out)	Instance 3.0 is loaded to Controller and is Checked In	Not Possible since Instance cannot be Checked in if CBT is Checked Out	Not Possible since Instance cannot be Checked In if CBT is Checked Out	Not Possible since instance cannot be Checked In if CBT is Checked Out
8	2.0 (Checked In)	3.0 (Checked Out - After it is loaded to the controller)	Cannot Check in CBT (Error Code 14627)	Allowed	Cannot Check In CBT and Instance (Error Code 14627)
9	2.0 (Checked Out)	3.0 (Checked Out - After it is loaded to controller)	Allowed	Not Allowed (Error Code 14626)	Allowed
10	CBT is Deleted from Library	Deleted from Project	Not Allowed (Error Code 14200)	Not Allowed (Error Code 14200)	Not Allowed (Error Code 14200)

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (To Be Checked In)	CBT Instance Only (To Be Checked In)	CBT + Instance (To Be Checked In)
11	2.0 (Checked In)	Deleted from Project	Cannot Check in CBT (Error Code 14200)	Not Allowed (Error Code 14200)	Cannot Check in CBT and Instance (Error Code 14200)
12	2.0 (Checked Out)	Deleted from Project	Allowed	Not Allowed (Error Code 14200)	Not Allowed (Error Code 14200)

5.2 Checking out an object from QVCS

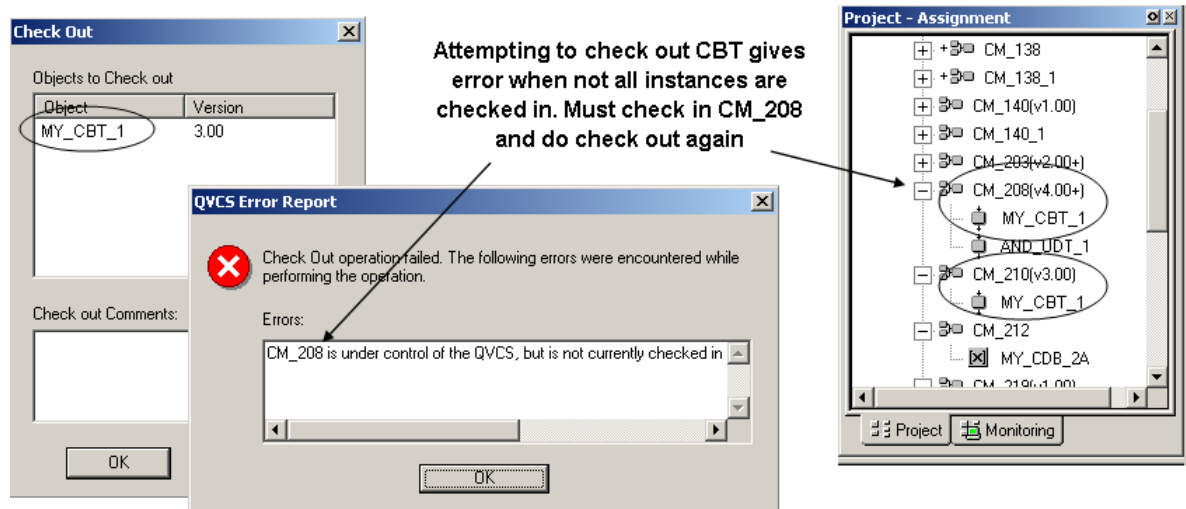
- To check out an object, you must have checked in the object in QVCS.
- While doing a check out operation, the object in question is locked exclusively by QVCS.
- If a user tries to modify/delete/open the object that is being checked out, an error message notifies the user that the object is locked and hence the operation has failed. Note that for CBT, the status of the “EDITLOCK” is also checked to insure that no changes can occur during the QVCS operation. Thus, if a CBT is being edited, QVCS will not be able to perform the check out operation.
- The user who checks out the object is the only one allowed to edit it.
- Once a user checks out an object, no other users may check out that object or any other versions of that object nor make any changes to the checked out object.
- Except for properties associated with the QVCS, you cannot make any changes to an object that is under QVCS control and is not checked out. Any object configuration form will only open in its Read Only state.
- You cannot check out an object that is already checked out.
- Check-out marks a specific version of an object in the Qualification and Version Control System as editable. Check-out takes the contents of a particular version from the QVCS to the ERDB and subsequently to the **Project** and the **Library** trees of the Control Builder. The **Monitoring** side of the Control Builder is not affected and will always show the version of the object that is loaded to the controller.

5.2.1 Multiple object check out considerations

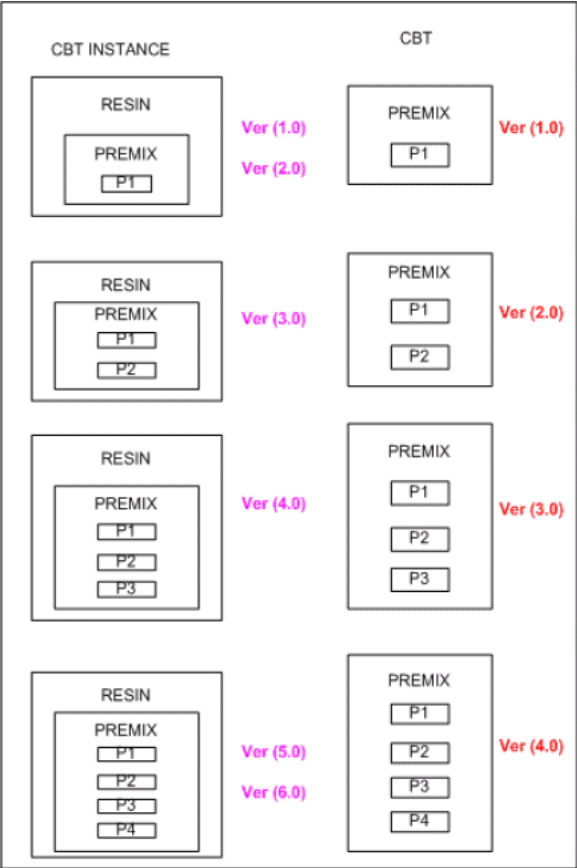
- You can select multiple objects for check out at the same time. Any check out information that is entered is applied to all the objects selected for check out.
- If any of the objects that are selected for multiple check out are not checked in, QVCS generates a notice that includes the reason for the error and the user name for each object that cannot be checked out as applicable. None of the selected objects will be checked out. All notices will then be displayed in a single error. The user is responsible for taking the appropriate corrective action such as changing the selection to only include the objects that are checked in.

5.2.2 CBT and UDT check out considerations

- When a CBT or UDT is checked out, all the instances get checked out automatically. If any instance is already checked out, an error is generated as shown in the following example illustration.



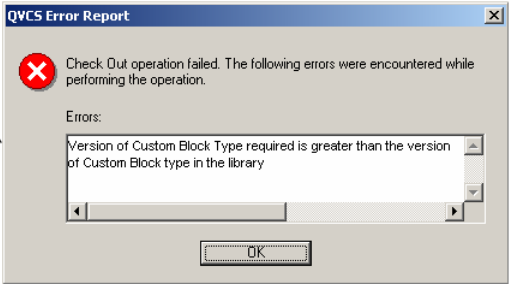
- Check out of non-current version of CBT less than current version is not allowed. This restriction enforces the migration rules requiring that a type never have fewer parameters than the current type. The following example illustration shows how permitting the check out of an earlier version of a CBT could lead to a violation of this rule. In this example, there are four versions of the PREMIX CBT, each version having one more parameter than the previous version. Now, if the user were permitted to check out version 2.0, it could be modified so that it has a new parameter. When this edited version is checked in, it would become the highest version number (5.0). This would violate the migration rules, since P3 and P4 found in version 4.0 would no longer exist, and migrating an instance that uses version 3.0 or 4.0 would not be possible. Users can use the Retrieve function ("Retrieving Version of an Object" on page 112) to make changes to an earlier version of a CBT and perform an import to create a new type.



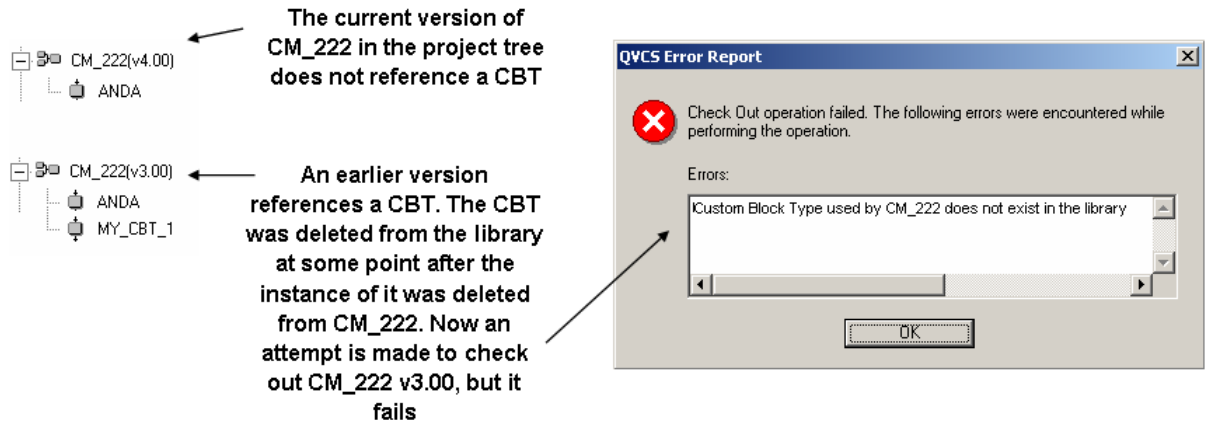
- Since only a single version of CBT can exist, all instances must be of that version.
- You cannot check out objects with instances of CBT whose version is not in the library.
 - QVCS cannot restore an instance of a CBT to the ERDB if its version dependency on the CBT is greater than the version of the CBT in the library.
 - As shown in the following example illustration, the current version of CM_1 is 3.0 and the current version of CBT_1 is 2.0. This condition can be reached by deleting CBT_1 and CM_1 and then reverting to CBT_1 v2.0 and CM_1 v3.0.

CM_1 Version	Uses CBT_1 Version	CM_1 Check Out Allowed
1.0	1.0	Yes
2.0	1.0	Yes
3.0	2.0	Yes
4.0	3.0	No
5.0	3.0	No

The checked out version of CM_1 will use CBT_1 version 2.0. When CM_1 is checked in and becomes version 6.0, its version dependency on CBT_1 will be version 2.0



- You cannot check out an object that contains instance of CBT or UDT that does not exist in the library as shown in the following example illustration. You must do a revert operation of CBT or UDT to get it back into the library to do a check out.



- You can specify a version consistent set of CBTs and instances of CBTs for check out as a group without restrictions on going forward or backward with versions unless there are instances that are loaded to controllers.
 - When instances of CBTs are loaded, the version rules are enforced. As indicated by an error message, you must first unload the loaded objects before you can re-initiate a revert operation.
 - As shown in the following example, assume that version 4.0 of CM_222 and version 3.0 of MY_CBT_1 are in the ERDB, and that both are selected for check out, which is allowed for a version consistent set as noted in the example.

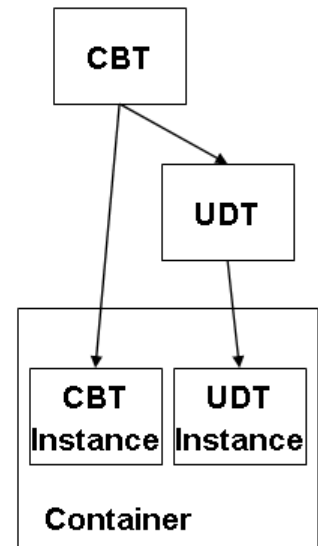
CM_222 Version	Uses MY_CBT_1 Version	Check out of CM_222 and MY_CBT_1 allowed?
1.0	1.0	Yes
2.0	1.0	Yes
3.0	2.0	Yes
4.0	3.0	Yes
5.0	3.0	Yes

Each row constitutes a version consistent set, so check out is allowed for any row

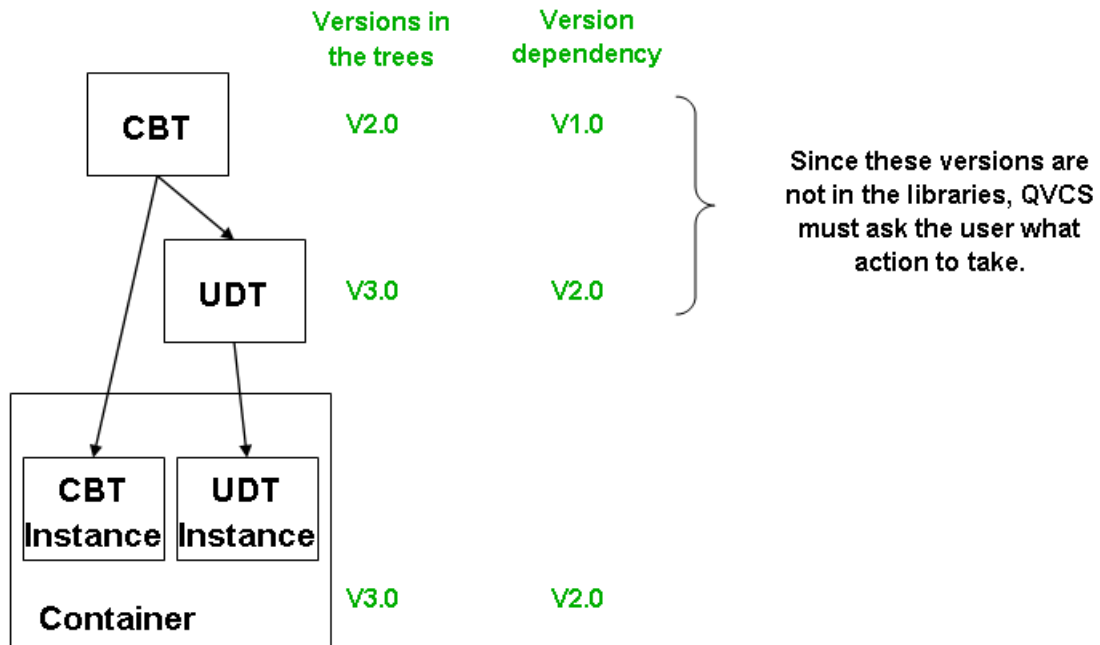
- When a check out includes both CBT and UDT, validation for CBT occurs first before UDT validation as shown in the following example illustrations.

Version Dependency table for the objects represented on the right

Object	Version	Uses Object	Version
Container	1.0	UDT	1.0
Container	1.0	CBT	1.0
Container	2.0	UDT	2.0
Container	2.0	CBT	1.0
Container	3.0	UDT	3.0
Container	3.0	CBT	2.0
UDT	1.0	CBT	1.0
UDT	2.0	CBT	1.0
UDT	3.0	CBT	2.0



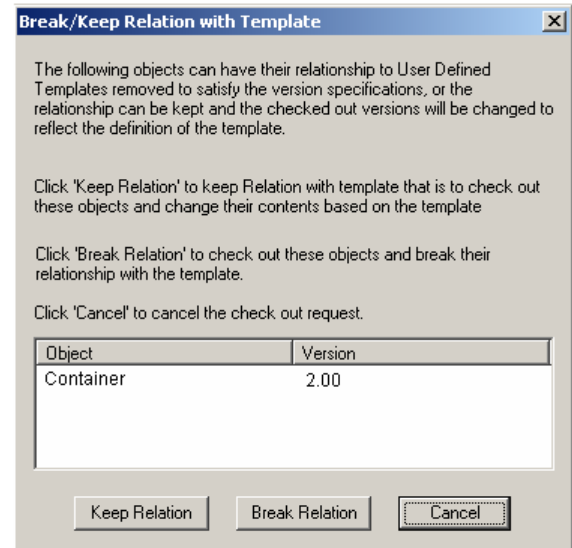
- Based on the version table in the illustration above and active current versions shown in the following illustration, user wants to check out version 2.0 of the container.



- User must respond to the dialog shown in the following illustration to successfully complete the check out of version 2.0 of the container.

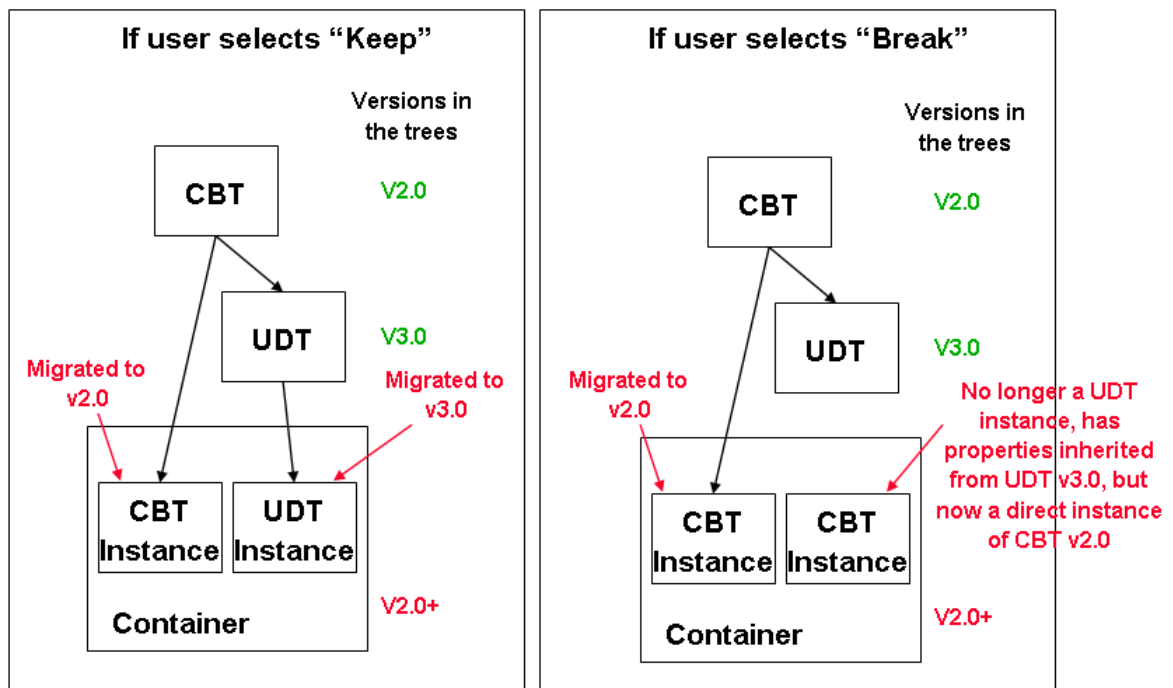
QVCS first determines that the version of the CBT that is required by version 2.0 of the container is version 1.0. Since version 2.0 of the CBT is in the library this is an acceptable check out (the instance in the container will be migrated to version 2.0 of the container).

Then, QVCS checks the dependency on the UDT and this results in dialog to the right.

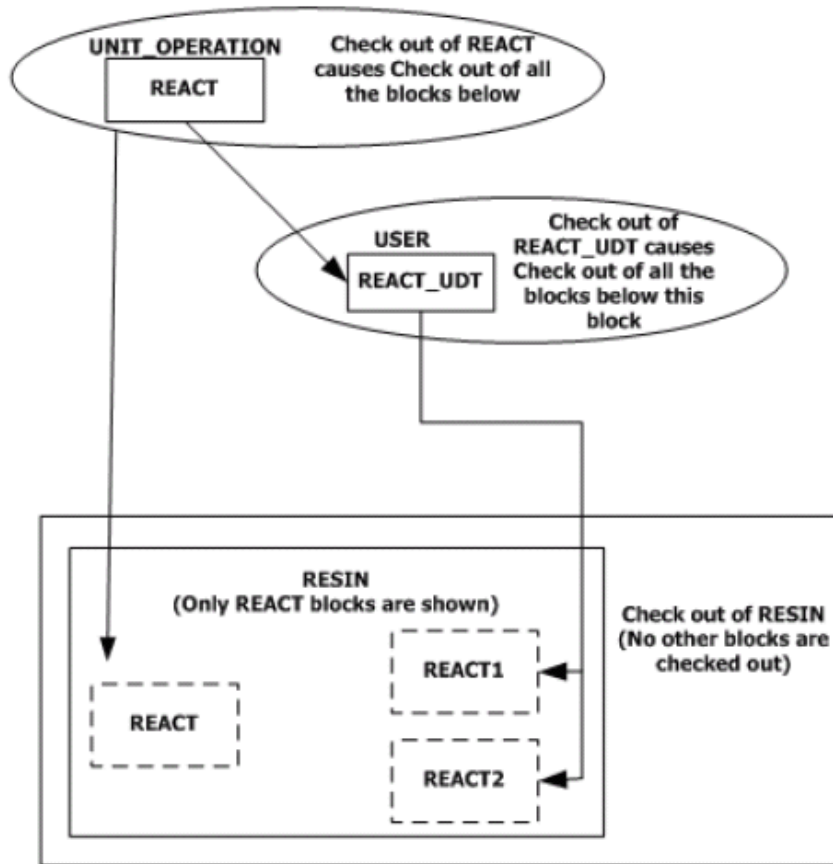


Dialog asks how to handle version discrepancy with UDT. User must select "Keep" or "Break" to continue

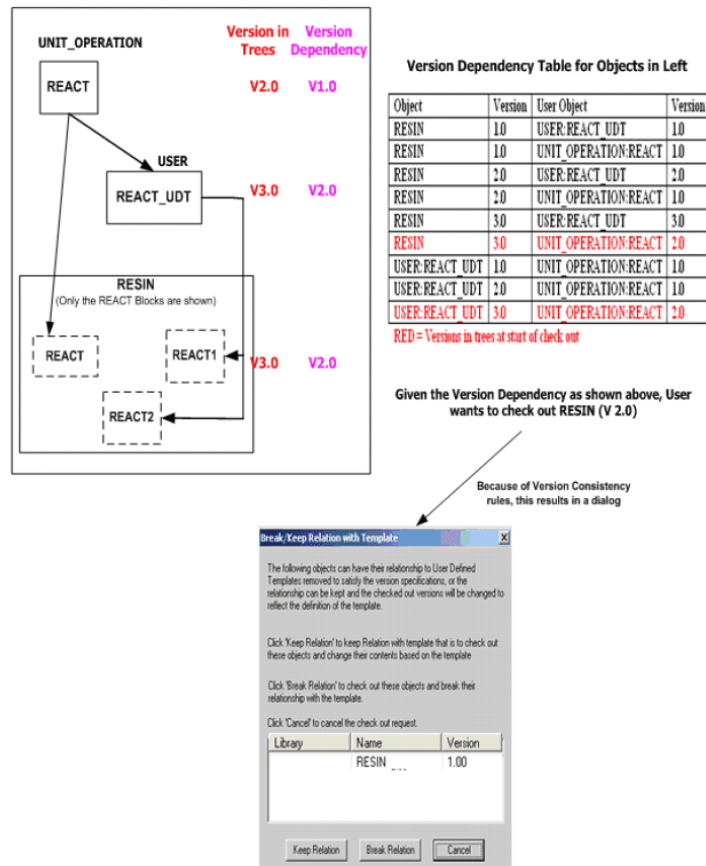
- The following illustrations summarize the results of choosing either Keep Relation or Break Relation in the dialog shown above.



- Since checking out the parent will check out all of the children, the dependencies with CBT and UDT can become quite complex as shown in the following illustration.



- Another example of the complexities involving both CBT and UDT is shown in the following illustration. In this example, QVCS first determines which version of the CBT (REACT) is required by the container (RESIN). If the version consistency rule is satisfied, then it will be an acceptable check out. Then, QVCS checks the dependency on the UDT (REACT_UDT). In case there is a mismatch, the user is given an option of either breaking the relationship or keeping the relationship with the UDT.



- When a UDT is checked out, all of its children are checked out automatically. If any children were previously checked out, an error occurs and no check outs are made.
- You can check out a strategy or UDT without its parent being checked out provided the following conditions are met.
 - If an object being checked out has a parent in the library with a different version and it is not in the check out list, a dialog appears announcing a mismatch in versions and giving the user three choices: *keep relation and check out*, *break relation and check out*, or *cancel*. If the user selects *keep relation and check out*, the check out continues and the object being checked out will be morphed to the current version of the parent. If the user selects *break relation and check out* the parent is changed to the system template and the reference to the UDT is removed. If the user selects *cancel*, the check out is canceled. Note that if there are multiple objects in the response list, the same choice will apply to all objects. The user cannot select *keep relation* for some objects in the list and *break relation* for others.
 - If an object being checked out has a version loaded to a controller that is not based on the UDT such that the relation was broken, a dialog appears stating that there is a mismatch in versions and if the check out proceeds the relation will be broken. The dialog gives the user two choices, *break relation and check out*, or *cancel*. If the user does not want to break the relation, they must select *cancel*, manually delete the strategy from the controller, and then reinitiate the check out process. This check takes precedence over the previous condition. If any objects meet this condition, then this dialog will be presented for all objects meeting it or the previous condition.
- If a derived child of a template is not under version control, its derivation parent cannot be changed unless the parent is checked in. Also, if a user defined template is not under version control, none of its derived children can be under version control either.

5.2.3 Checking out an object

- 1 Select one or more checked in objects in the **Project** or **Library** tab of Control Builder to be checked out from QVCS.
- 2 In the **Tools** menu, click **QVCS Manager** to call up the QVCS Manager Main window.
You can alternately right-click selected object and select **QVCS Manager** from the list or click the QVCS button in the toolbar to call up the QVCS Manager Main Window.
- 3 Select one or more *Checked In* objects in the list box of the Main window.
You can alternately select *Checked In* objects in the Query window or Versions window
- 4 In the **Tools** menu of the QVCS Manager, click **Check Out** to call up the Check Out dialog as shown in the following Figure.
You can alternately right-click selected object and select **Check Out** from the list or click the Check Out button in the toolbar to call up the Check Out dialog.
- 5 Confirm that the objects to be checked out appear in the list box at the top of the dialog.
- 6 You can key in information about this check out in the Comments list box. This is **not** mandatory, since check out will continue if a comment is not entered.
- 7 Click the **OK** button to initiate the check out operation and close the Check Out dialog. Or, click the **Cancel** button to abort the check out function.
- 8 This completes the procedure.

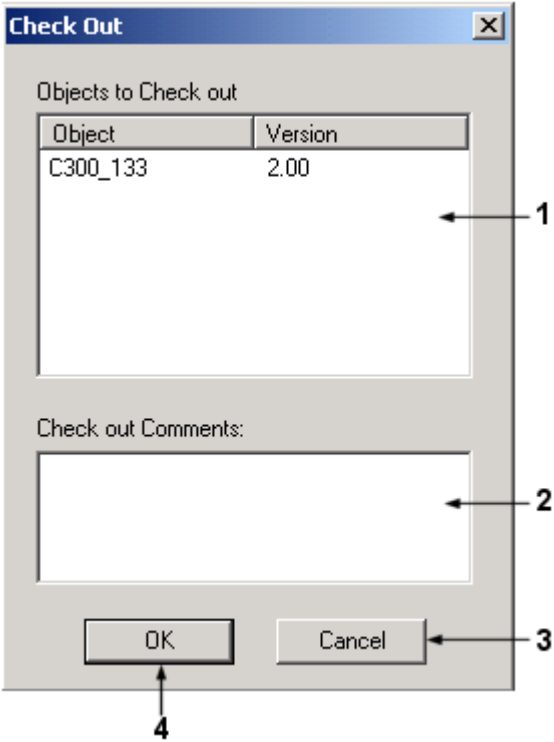


Table 4: Check Out Dialog Description

Callout	Item	Description
1	Objects to Check out	Shows list of objects that have been selected for check out.
2	Check out Comments	You can optionally key in a comment about this check out. This is not mandatory for the check out function to continue.

Callout	Item	Description
3	Cancel Button	Click to abort the check out function.
4	OK Button	Click to initiate the check out function.

5.2.4 Example check out scenarios

The following table outlines various check out scenarios based on the “Graphical Example of Layered Recipe Check In and Check Out Scenario” on page 81 provided in another section of this document. Refer to the Control Builder Error Codes Reference document for more information about listed Error Codes.

Table 5: Layered Recipe Scenarios for Check Out

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (To Be Checked Out)	CBT Instance Only (To Be Checked Out)	CBT + Instance (To Be Checked Out)
1	2.0 (Checked In)	3.0 (Checked In)	Can only check out version 4.0 (or latest version). Otherwise, see Error Code 14711	Can check out version 1.0 to 3.0. (Higher versions check out are not allowed - See Error Code 14713)	Only version 4.0 of CBT and any version of the instance can be checked out. (See Error Code 14711)
2	2.0 (Checked In)	3.0 (Checked Out)	Not allowed as its instance is checked out. (See Error Code 14712)	Not Allowed. (See Error Code 14712)	Not Allowed. (See Error Code 14712)
3	2.0 (Checked Out)	3.0 (Checked In)	Not possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out	Not possible since instance cannot be checked in if CBT is checked out
4	2.0 (Checked Out)	3.0 (Checked Out)	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14626)
5	2.0 (Checked In)	Instance 3.0 is loaded to Controller and is Checked In.	Can only check out version 4.0 (or latest version). See Error Code 14711)	Can check out version 1.0 to 3.0. Higher versions check out are not allowed. (See Error Codes 14713)	Only version 4.0 of CBT and any version of the instance can be checked out. (See Error Code 14711)
6	2.0 (Checked Out)	Instance 3.0 is loaded to Controller and is Checked In	Not possible since instance cannot be check in if CBT is checked out	Not Possible since instance cannot be check in if CBT is checked out	Not possible since instance cannot be check in if CBT is checked out
7	2.0 (Checked In)	Instance 3.0 Checked Out. (After it is loaded to the controller)	Not allowed as its instance is checked out. (See Error Code 14712)	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14626)
8	2.0 (Checked Out)	3.0 Checked Out (After it is loaded to controller)	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14626)
9	CBT is Deleted from Library	Deleted from Project	Not Allowed. (See Error Code 14200)	Not Allowed. (See Error Code 14200)	Not Allowed. (See Error Code 14200)
10.	2.0 (Checked In)	Deleted from Project	Can only check out version 4.0 (or Latest Version). (See Error Code 14711)	Not Allowed. (See Error Code 14200)	Cannot Check out Type and Instance. (See Error Code 14200)

ScenarioNumber	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (To Be Checked Out)	CBT Instance Only (To Be Checked Out)	CBT + Instance (To Be Checked Out)
11.	2.0 (Checked Out)	Deleted from Project	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14200)	Not Allowed. (See Error Code 14200)

5.3 Undoing Check out from QVCS

- For undoing a check out, you must have checked out objects from QVCS.
- While doing an undo check out operation, the object in question is locked exclusively by QVCS.
- If a user tries to modify/delete/open the object that is being checked in, an error message notifies the user that the object is locked and hence the operation has failed. Note that for CBT, the status of the “EDITLOCK” is also checked to insure that no changes can occur during the QVCS operation. Thus, if a CBT is being edited, QVCS will not be able to perform the undo check out operation.

5.3.1 Multiple object undo check out considerations

- You can select multiple objects for undo check out at the same time. When multiple objects are selected and undo check out is requested, the QVCS displays the standard undo check out confirmation dialog.
- If any of the objects that are selected for multiple object undo check out are not checked out or are locked by another user, QVCS generates a line of text for each object that cannot have its check out undone that includes the reason, and if applicable, a user name, and no undo check outs are performed for any of the objects in the selection. All error lines will then be displayed in a single error. The user is responsible for taking the appropriate corrective action such as changing the selection to only include the objects that are checked out by the user.

5.3.2 Objects with connections undo check out considerations

- When undoing a check out for an object that has connections, you must consider the type of connections that exist for the object. Certain types of connections have “hidden” connections, such as Regulatory Control block Back Initialization, that are not explicitly made by the user. Regardless of the type of connection, the rules governing an undo check out are the same as exist for importing an object in a No-QVCS system. For the import to be error free, the necessary locking and block existence criteria must be satisfied. Depending on the existence of errors or warnings, the undo check out may not succeed, or the object that is restored may not be the complete image that was initially versioned. For example, for a standard parameter connection, if a parameter connection was removed in an object that was checked out, and the block that it was connecting to was also removed, then if an undo check out were performed on the object that originally had the connection, but an undo check out was not performed on the object that contained the block, the result would be no connection in the restored object. Based on messages generated by the undo check out operation, the user is responsible for making the necessary corrections.

5.3.3 CBT and UDT undo check out considerations

- If you undo check out of CBT or UDT, all instances of the CBT or UDT will automatically be included with the CBT or UDT.
- If the instance of the CBT has had a “Change Parent” operation performed, then it will not be included as part of the undo check out operation (same as for UDT's). A message will be displayed asking the user if it is okay to continue the undo operation. Subsequent to the undo of the CBT, the user can choose to undo the check out of the instances that were not included.
- If you undo check out of objects containing instances of CBT or UDT, you cannot undo check out if the CBT or UDT is checked out.
 - This guarantees that the CBT version will support the CBT version dependency for the restored instance.
- If all the criteria for both CBT and UDT is not met, the undo check out is canceled and the appropriate error message is displayed.
- All parents of a template or strategy must be checked in previously or be included with the children at the time of the undo check out. If this is not the case, then an error is generated and no undo check outs are performed.

- An undo checkout of a template will undo the checkout of all of its children. If any children have had their relationship with the parent broken during the checkout, a warning message is generated and users will need to perform the undo checkout manually on those objects.
- If an object having its checkout undone has a parent in the library with a different version and it is not in the undo checkout list, a dialog appears stating that there is a mismatch in versions and gives the user two choices, *break relation and revert*, or *cancel*. If the user selects *break relation and revert*, the parent is changed to the system template and the reference to the UDT is removed. If the user selects *Cancel*, the undo checkout is canceled.
- The following table summarizes conditions and options for undo checkout of instance.

State of current instance	UDT version matches instance version being restored	UDT version does not match instance version
Not loaded and UDT derived	Normal undo checkout action	Not Possible
Not loaded and UDT relation broken	Normal undo checkout action	User choice: <ul style="list-style-type: none"> • Break relation and undo checkout • Cancel
Loaded, and UDT derived	Normal undo checkout action	Not Possible
Loaded and UDT relation broken	An error message is displayed stating that the instance must be deleted from the controller before the undo can be performed. If the user deletes the instance from the controller and then reinitiates the undo checkout, the relationship will be restored	User choice: <ul style="list-style-type: none"> • Break relation and undo checkout • Cancel

- The following table summarizes conditions and options for undo checkout of UDT.

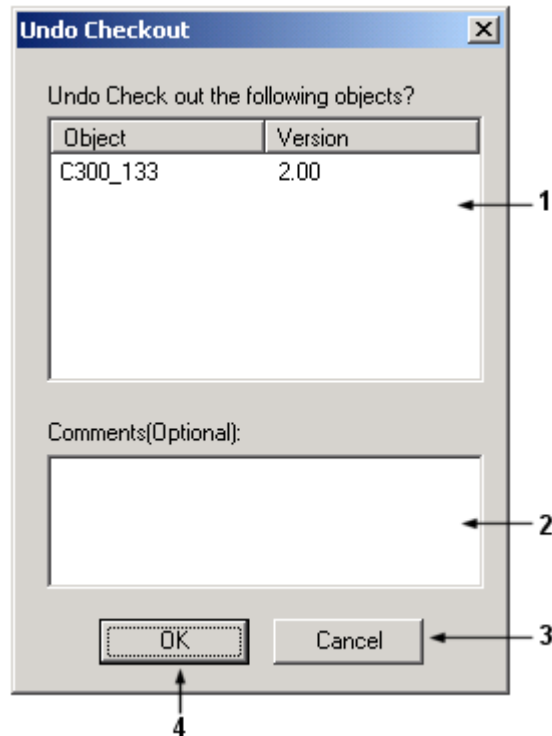
State of current instance	Instance version matches UDT version being restored	Instance version does not match UDT version
Not loaded and UDT derived	Normal undo checkout action	Not possible
Not loaded and UDT relation broken	Will not be included in undo checkout. User will be given a warning	Will not be included in undo checkout. User will be given a warning
Loaded, and UDT derived	Normal undo checkout action	Not Possible
Loaded and UDT relation broken	Will not be included in undo checkout. User will be given a warning	Will not be included in undo checkout. User will be given a warning

5.3.4 Undoing a check out

- 1 Select one or more checked out objects in the **Project** or **Library** tab of Control Builder to be undone.
- 2 In the **Tools** menu, click **QVCS Manager** to call up the QVCS Manager Main window.
You can alternately right-click selected object and select **QVCS Manager** from the list or click the QVCS button in the toolbar to call up the QVCS Manager Main Window.
- 3 Select one or more *Checked Out* objects in the list box of the Main window.
You can alternately select *Checked Out* objects in the Query window or Versions window
- 4 In the **Tools** menu of the QVCS Manager, click **Undo Check Out** to call up the Undo Check Out dialog as shown in the following Figure.
You can alternately right-click selected object and select **Undo Check Out** from the list.
- 5 Confirm that the objects to be checked out appear in the list box at the top of the dialog.

- 6 You can key in information about this undo check out in the Comments list box. This is optional, since undo check out will continue if a comment is not entered.
- 7 Click the **OK** button to initiate the undo check out operation and close the Check Out dialog. Or, click the **Cancel** button to abort the undo check out function.
- 8 This completes the procedure.

Table 6: Undo Checkout Dialog Description



Callout	Item	Description
1	Undo Check out the following objects?	Shows list of objects that have been selected for undo check out.
2	Check out Comments	You can optionally key in a comment about this undo check out. This is not mandatory for the undo check out function to continue.
3	Cancel Button	Click to abort the undo check out function.
4	OK Button	Click to initiate the undo check out function.

5.3.5 Example undo check out scenarios

The following table outlines various undo check out scenarios based on the “Graphical Example of Layered Recipe Check In and Check Out Scenario” on page 81 provided in another section of this document. Refer to the Control Builder Error Codes Reference document for more information about listed Error Codes.

Table 7: Layered Recipe Scenarios for Undo Check Out

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform undo Checked Out)	CBT Instance Only (Perform undo Checked Out)	CBT + Instance (Perform undo Checked Out)
1	2.0 (Checked In)	3.0 (Checked In)	Cannot do an Undo Check Out of the CBT. (see Error Code 14627)	Cannot do an Undo Check Out of the Instance. (See Error Code 14627)	Cannot do an Undo Check Out. (See Error Code 14627)
2	2.0 (Checked In)	3.0 (Checked Out)	Cannot do an Undo Check Out of the CBT. (see Error Code 14627)	Allowed	Cannot do an Undo Check Out. (See Error Code 14627)
3	2.0 (Checked Out)	3.0 (Checked In)	Not Possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out
4	2.0 (Checked Out)	3.0 (Checked Out)	Allowed for the CBT. The instances will all get Undo Check Out.	Not Allowed. (See Error Code 14710)	Allowed
5	2.0 (Checked In)	Instance 3.0 is loaded to Controller and is Checked In.	Cannot do an Undo Check Out of the CBT. (see Error Code 14627)	Cannot do an Undo Check Out of the Instance. (See Error Code 14627)	Cannot do an Undo Check Out. (See Error Code 14627)
6	2.0 (Checked Out)	Instance 3.0 is loaded to Controller and is checked In	Not possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out	Not possible since instance cannot be checked in if CBT is checked out
7	2.0 (Checked In)	Instance 3.0 Checked Out. (After it is loaded to the controller)	Cannot do an Undo Check Out of the CBT. (see Error Code 14627)	Allowed	Cannot do an Undo Check Out. (See Error Code 14627)
8	2.0 (Checked Out)	3.0 Checked Out (After it is loaded to controller)	Allowed for the CBT. The Instances will all get Undo Check Out.	Not Allowed. (See Error Code 14626)	Allowed
9	CBT is Deleted from Library	Deleted from Project	Cannot do an Undo Check Out of the CBT. (See Error Code 14200)	Not Allowed. (See Error Code 14200)	Cannot do an Undo Check Out. (See Error Code 14200)
10.	2.0 (Checked In)	Deleted from Project	Cannot do an Undo Check Out of the CBT. (See Error Code 14627)	Not Allowed. (See Error Code 14200)	Cannot do an Undo Check Out. (See Error Code 14200)
11.	2.0 (Checked Out)	Deleted from Project	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14200)	Not Allowed. (See Error Code 14200)

5.4 Graphical Example of Layered Recipe Check In and Check Out Scenario

The following illustrations provide a graphical representation of what occurs when layered recipe objects such as Recipe Control Modules (RCM), Sequential Control Modules (SCM), and Unit Control Modules (UCM) are checked in or checked out of QVCS.

5.4.1 Example of objects that can be versioned

The objects that can be versioned, such as RCM, SCM and UCM, are *independent* of each other in terms of being able to check in, check out, revert without consideration of the status of the other objects, as shown in the following illustration. You should use revert labels, if you want to keep a *version consistent* set.

- Red objects can be versioned
- Green objects are contained in objects and are not independently versioned.

Pre-mixer	PM_01	UCM
Pre-mixer	PM_01A	UCM
Pre-weigh	PW_01	UCM
Reactor	R_01	UCM
Resin	RESIN_01	UCM
Storage	ST_01	UCM

Recipes

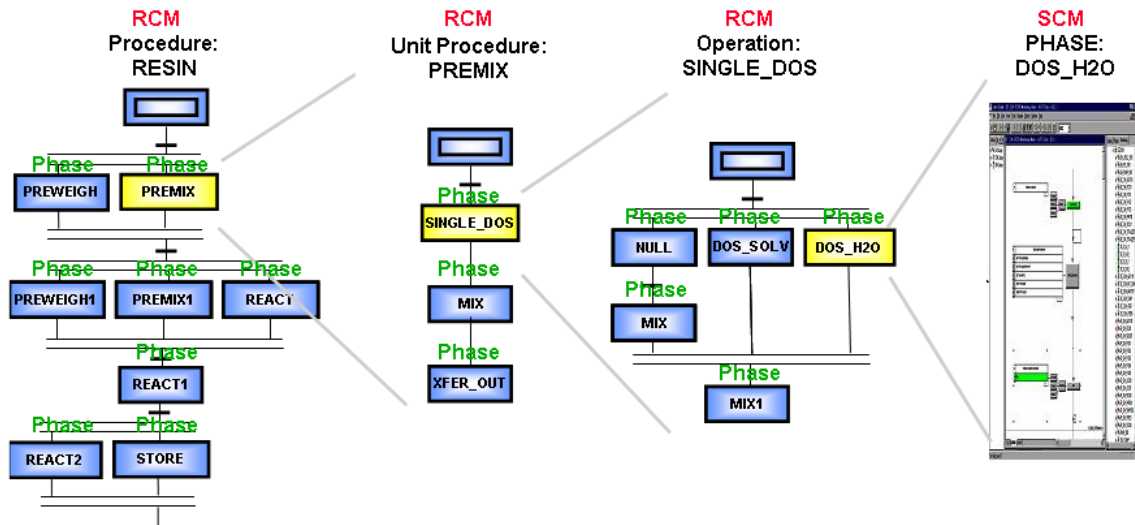


Figure 9: Layered Recipe Graphical Example Scenario

5.4.2 Example of versioned objects and rules for check in and check out

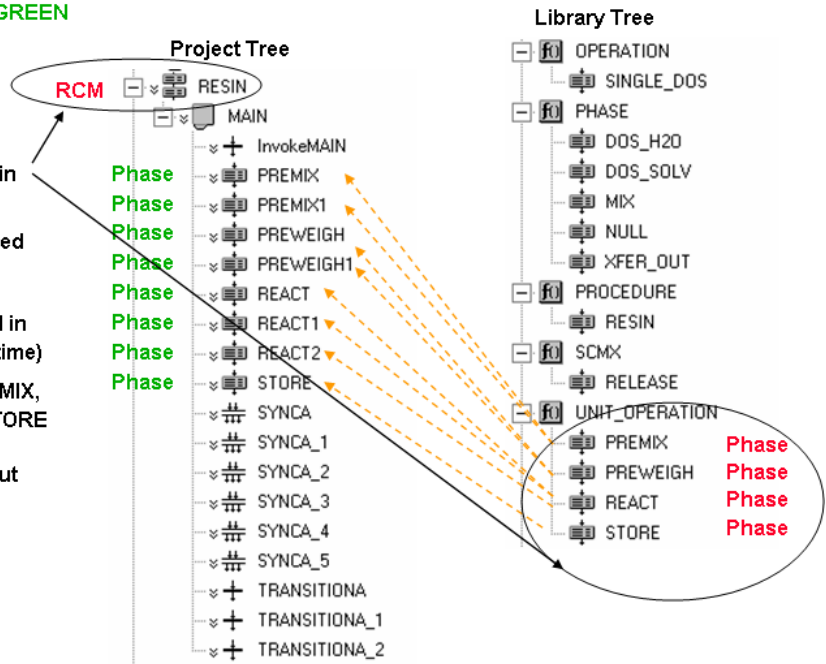
Objects that can be versioned in **RED**

← - - - Indicates instantiation of CBT

Objects that are contained in objects, and that are not independently versioned are in **GREEN**

Based on QVCS rules for check in and check out:

1. Before RESIN can be checked in, the CBTs PREMIX, PREWEIGHT, REACT, and STORE must all be checked in (or checked in at the same time)
2. When any of the CBTs PREMIX, PREWEIGHT, REACT, or STORE are checked out, RESIN will automatically be checked out



5.4.3 Example of rules for check in and check out with UDT

Objects that can be versioned in RED

Objects that are contained in objects, and that are not independently versioned are in **GREEN**

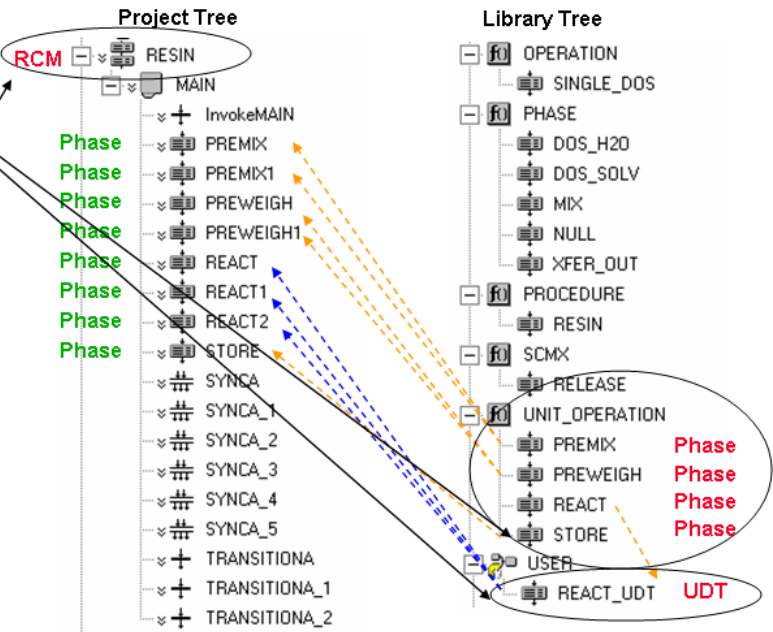
← - - - Indicates instantiation of CBT

←--- Indicates instantiation of UDT

In this scenario, a UDT has been created based on REACT, and it is used in RESIN

Based on QVCS rules for check in
and check out:

1. Before RESIN can be checked in, REACT_UDT must be checked in (or at the same time) and the CBT's PREMIX, PREWEIGHT, REACT, and STORE must be checked in (or at the same time)
2. Before REACT_UDT can be checked in, the CBT REACT, must be checked in (or at the same time)
3. When REACT_UDT is checked out, RESIN will automatically be checked out
4. When any of the CBTs PREMIX, PREWEIGHT, REACT, or STORE are checked out, RESIN will automatically be checked out



5.5 Instantiating CBT

You can instantiate a CBT in a container objects such as a Control Module (CM), Recipe Control Module (RCM), Sequential Control Module (SCM), and so on. As shown in the following illustration, the CBT must also be under QVCS to place the CBT instances under version control.

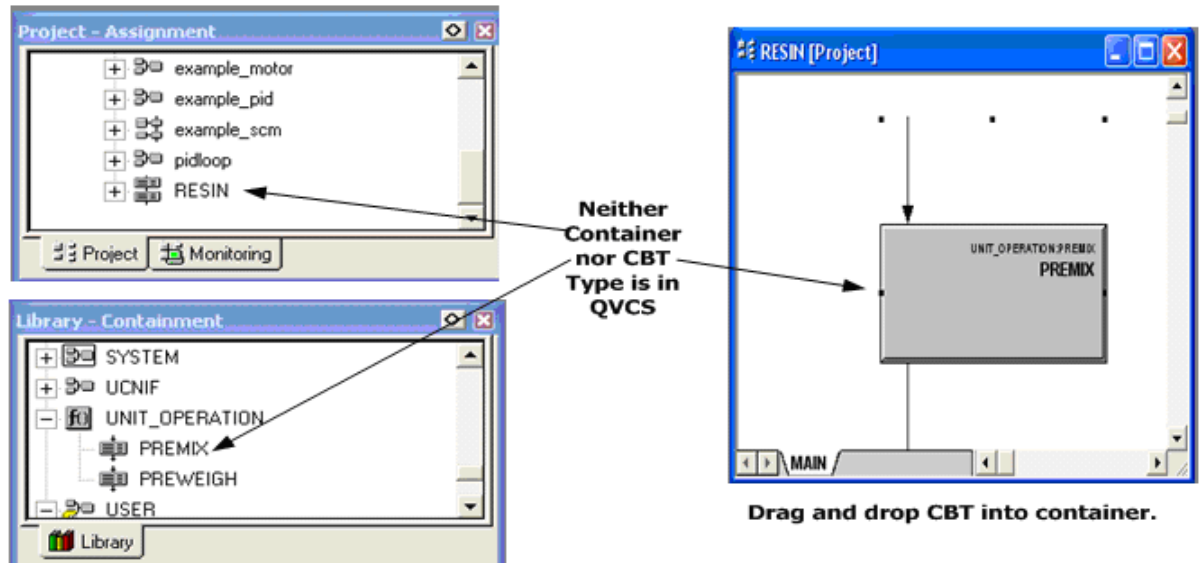


Figure 10: CBT and its instance not in QVCS

5.5.1 Rules when a CBT Instance can be under QVCS

Instantiate a CBT in a Container such as CM, RCM, SCM, and so on.

- CBT not in QVCS and the Container is Checked Out before containing the CBT instance. The Container cannot be Checked In until the CBT is also Checked In. Or, both the Container and the CBT can be Checked In together, as shown in the following illustration See “Checking in an object to QVCS” on page 60 for more details.

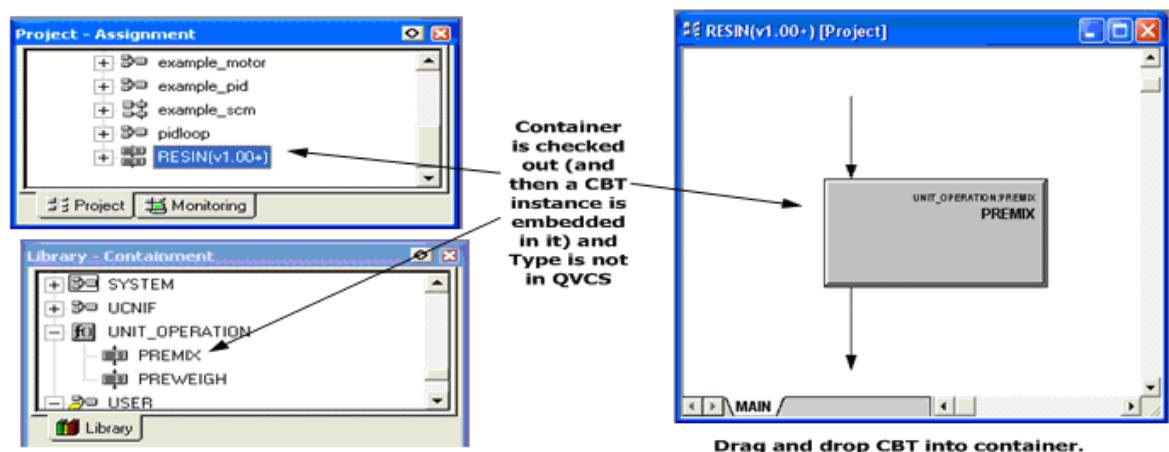


Figure 11: CBT not in QVCS and Container is checked out.

- If the CBT is in QVCS (Checked In or Checked Out) and the Container is not in QVCS, there are no restrictions on the container, as shown in the following illustration. Any changes made to the CBT when it is Checked Out are propagated to the instances, as occurs normally.

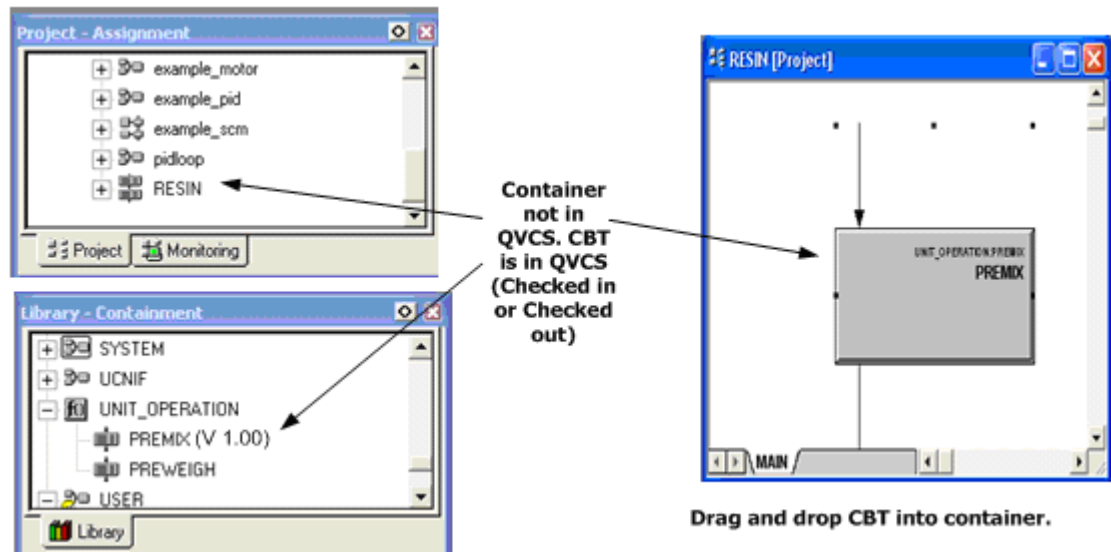


Figure 12: Container is not in QVCS and CBT is in QVCS (checked in or checked out)

- The CBT is in QVCS (Checked Out) and the Container is Checked Out. Only restriction is that the Container cannot be Checked In until the CBT is Checked In or both are Checked In at the same time.

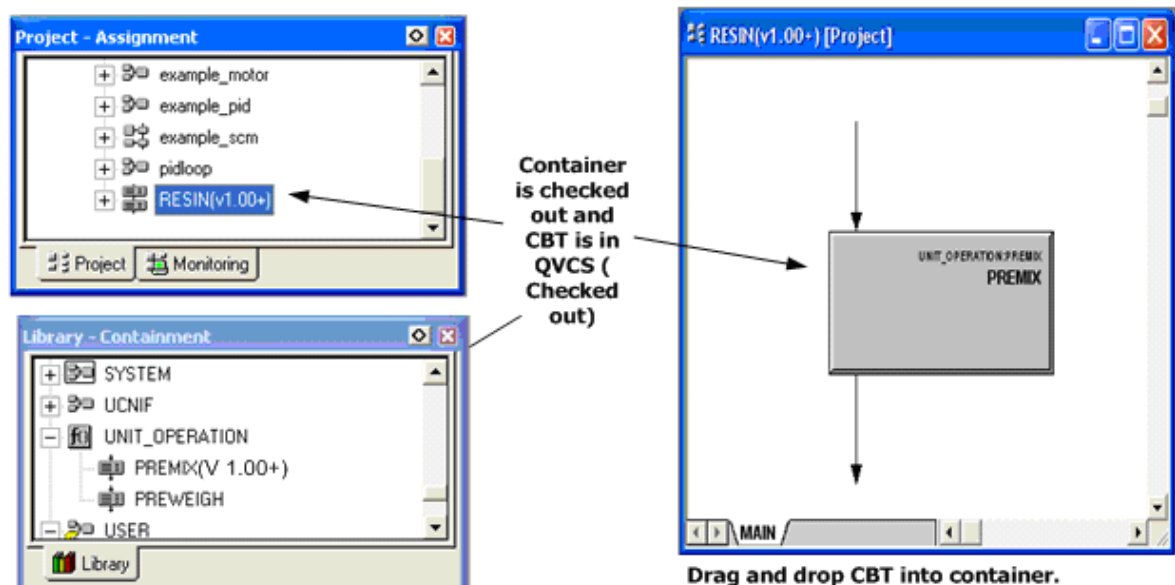


Figure 13: Container and CBT are in QVCS

- The CBT is not in QVCS or it is Checked In to QVCS and the Container is not Checked Out. Not allowed since container cannot be modified unless Checked Out. If the CBT is Checked Out, the container will also be Checked Out since an object cannot be Checked In if it has an instance of a CBT that is based on a CBT that is Checked Out. This is shown in the following figure.

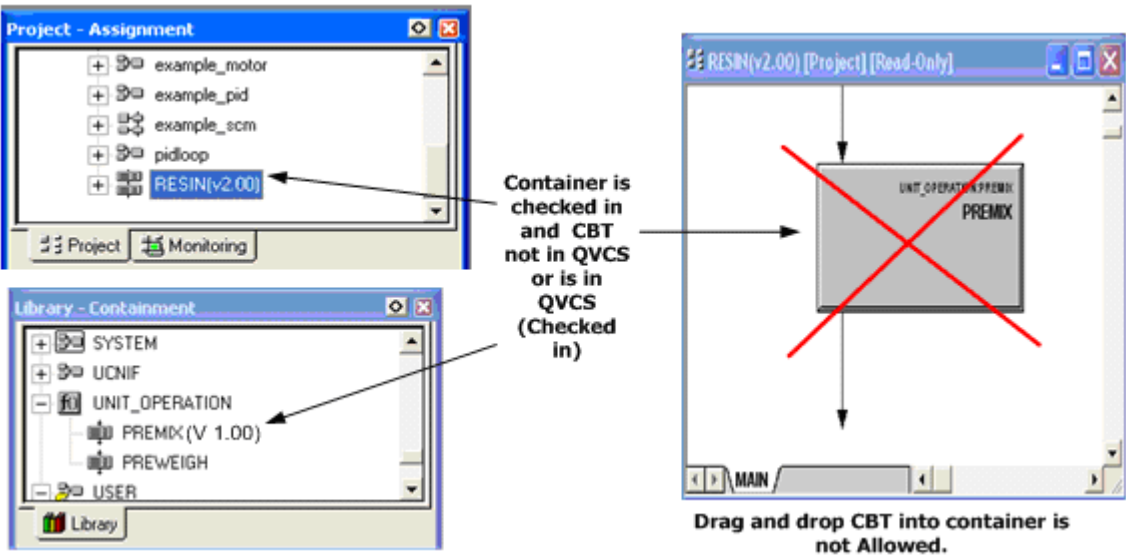


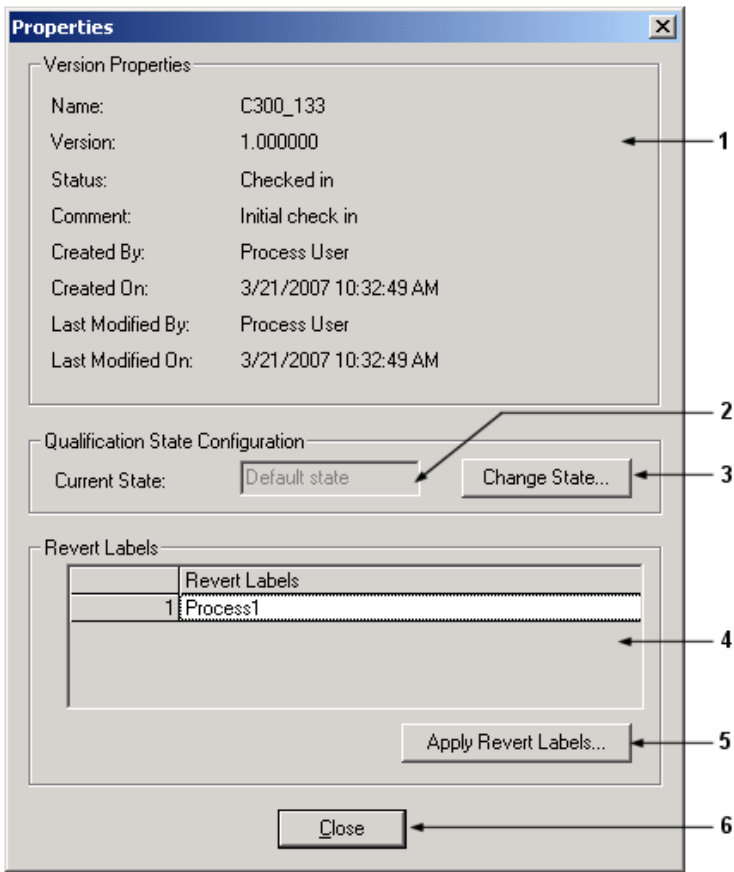
Figure 14: Container is Checked In and CBT can be or may not be in QVCS

5.6 Viewing and Changing Object Properties

- For viewing and changing object properties, you must have checked in objects to QVCS.
- The Properties dialog permits changes to the qualification state after check in and the revert label at or after check in for the currently selected version of the object. All other information in the dialog is read-only except of the comment, which must be entered at check in.
- You can also view properties for tagged blocks through the QVCS tab on the block's configuration parameters form.

To view object properties

- 1 Select one or more checked in and/or checked out objects in the **Project** or **Library** tab of Control Builder to be viewed.
- 2 In the **Tools** menu, click **QVCS Manager** to call up the QVCS Manager Main window.
You can alternately right-click selected object and select **QVCS Manager** from the list or click the QVCS button in the toolbar to call up the QVCS Manager Main Window.
- 3 Select object you want to view properties of in the list box of the Main window.
You can alternately select object in the Query window or Versions window
- 4 In the **View** menu of the QVCS Manager, click **Object Properties** to call up the **Properties** dialog as shown in the following Figure.
You can alternately press **Alt** and **Enter** keys together to call up the **Properties** dialog for selected object.
- 5 View read-only information in the Version Properties portion of the dialog
- 6 If applicable, you can change the current qualification state by clicking the **Change State** button to call up the **Change Qualification State** dialog. .
- 7 If applicable, you can apply revert labels by clicking the **Apply Revert Labels** button to call up the **Apply/Remove Revert Labels to Version** dialog.
- 8 Click the Close button to close the dialog.
- 9 Repeat Steps 3 to 7 to view properties for other objects as applicable.
- 10 This completes the procedure.



Callout	Item	Description
1	Version Properties	Provides read only list of selected objects QVCS version properties information.
2	Qualification State Configuration	Shows current qualification state
3	Change State Button	Click to call up the Change Qualification State dialog to change the current state.
4	Revert Labels	Shows list of revert labels applied to the selected object.
5	Apply Revert Labels Button	Click to call up the Apply/Remove Revert Labels to Version dialog to change revert labels for current version.
6	Close Button	Click to close the dialog.

5.7 Making changes to an object

It is possible to save the object with intermediate changes without checking it into the QVCS. In this case, a user cannot restore to an intermediate stored object, since they can only go back to the last version stored in the QVCS. As in the current Control Builder, if an object is opened on one station, it will be opened with read only access on other stations. And, only the user who checked out an object is allowed to make changes to it.

5.7.1 Changing a connection

Normally, when defining a connection to another object, only the object that contains the definition of the connection needs to be checked out, since it is the only object that includes the information about the connection. However, this is not the case when *hidden* connections are involved. A hidden connection is made automatically by Control Builder rather than manually by a user. Examples of these hidden connections are Back Initialization for Regulatory Control type blocks and Foundation Fieldbus connections.

In strategies involving hidden connections, you must check out all of the objects that are part of the connection definition. For example, for PID block Back Initialization connections both the primary and secondary blocks contain definitions of the connections and the secondary block contains a hidden connection. If the user attempts to make a change to a PID block Back Initialization connection without both objects checked out from QVCS, an error message is generated identifying the object that could not be modified. The user is responsible for checking out the secondary object so that the connection can be defined. A similar situation exists for fieldbus connections except there are three objects that are included in the definition of the connection: one *main*, one *hidden*, and one *hook*.

5.7.2 Changing a user defined template (UDT)

In a No-QVCS system, when a user defined template is modified, all derived templates and instantiated children will automatically be modified. The same is true in a Full-QVCS system for any derived children, other templates and strategies, of user defined templates that have not been placed under version control, since by definition these children are also not under control of the QVCS. As previously described for UDT check out considerations, only those objects that have had their derivation parent changed will be affected by the change. All of the objects with the derivation parent that is being modified must be checked out of the QVCS for any changes to be made.

5.7.3 Changing a custom block type (CBT)

A CBT that is checked into QVCS will be opened as **read only** in the Parameter Definition Editor (PDE) form. You can use the PDE **File** menu selection **Save As** to create a new block type from the checked in CBT by selecting a library and entering a new block type name.

You can make minor edits to a checked out CBT that is opened in the PDE and save the changes so that the type is updated. If you make a major change to the CBT, you must do a **Save As** operation to create an entirely new type that is not under QVCS control. The source CBT is unchanged and it remains checked out. You can check the new CBT into QVCS where it is treated as a completely distinct QVCS object with its own history and so on. For instances to use the new CBT, you must do a manual *Change Parent* operation.

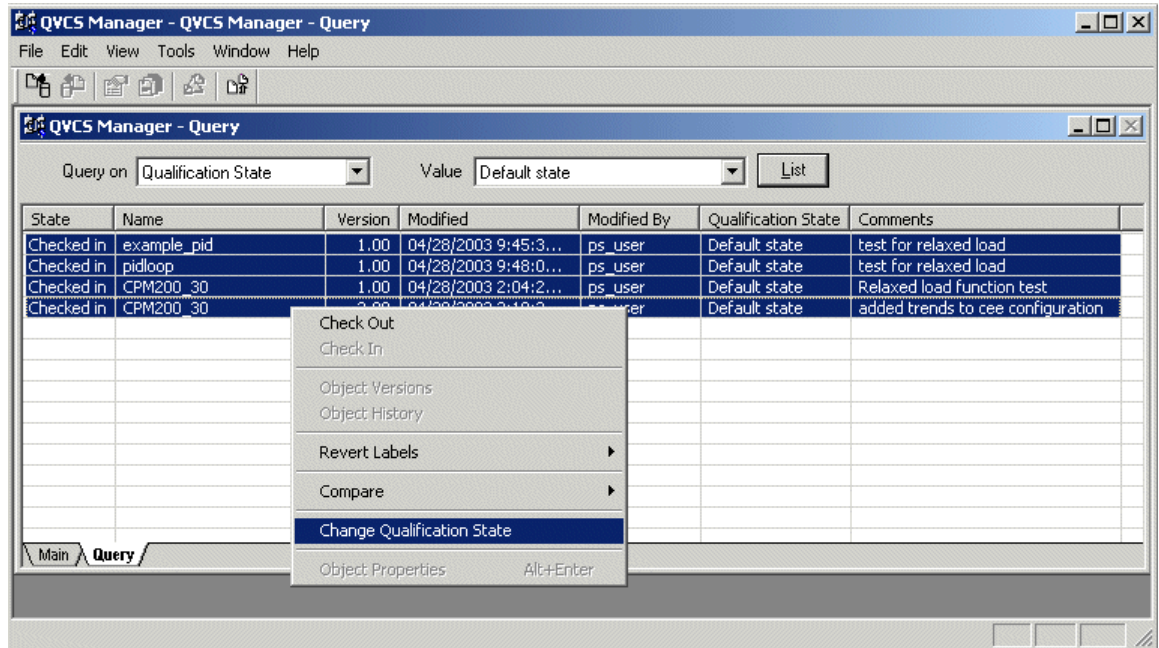
Remember that checking out a CBT from QVCS also checks out all objects with instances of the CBT. And, the instances cannot be checked in until the CBT is checked in. Since the instance was checked out when the CBT was modified, it must be checked in and re-qualified regardless of the actual change that was made to the CBT.

- For example, adding a parameter does not change the behavior of the instance, but still requires re-qualifying the object.
- For example, changing the default value for a parameter will change the behavior of the instance, if the instance is using the default value. A non-default value will not be replaced.

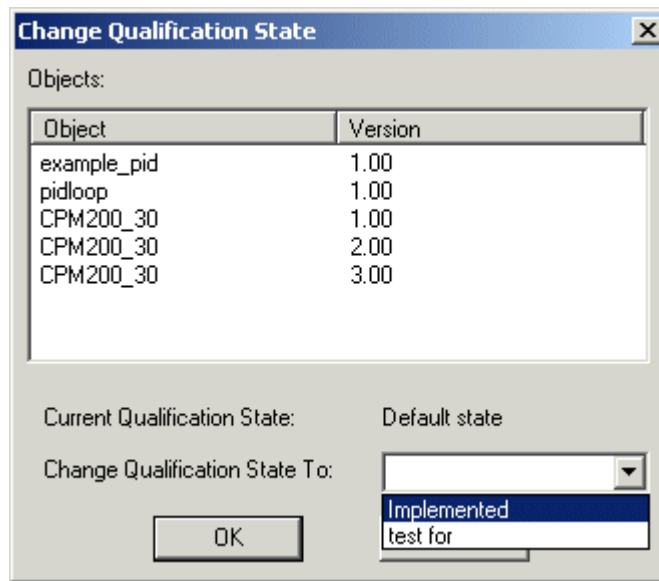
5.8 Making multiple object qualification state transitions

The following table outlines the typical steps for making qualification state transitions for multiple objects. You can easily adapt the procedure to apply to changing the qualification state for a single object. The illustrations used are for example purposes only.

- 1 Click **Tools** -> **QVCS Manager** to launch the Manager.
- 2 Click the Query tab.
- 3 Click the down-arrow button to the right of **Query on** field and select **Qualification State** from the shortcut menu.
(This query guarantees that the objects are all in the same state, but you can initiate any other query; and then, as long as the selected objects are all in the same qualification state, it is possible to change the state of all objects.)
- 4 Click the down-arrow button to the right of **Value** field and select **Default State** or other desired state from the shortcut menu.
- 5 Click the first object in the dialog list and Shift + Click the last object in the list to select all the objects.
- 6 Click **Tools** > **Change Qualification State** or right click and select Change Qualification State from the shortcut menu.



- 7 Click the down-arrow button to right of **Change Qualification State To** field and select the desired state from the shortcut menu.



- 8 Click the **OK** button to initiate qualification state change validation.



Attention

- If any validation fails, none of the qualification states for any of the object versions will be updated. Check displayed error message(s) to identify the source of the validation failure(s).
- If an electronic signature is configured for the requested transition, a single signature dialog is displayed and the same signature is applied to all of the transitions. A log entry, including applicable associated electronic signature log entry, is generated for each object in the list.

- 9 The qualification states for the selected objects are changed.
- 10 This completes the procedure.

5.9 Renaming an object

If an object has never been under version control, it may be renamed just as in a No-QVCS system. You **cannot** rename an object that is under QVCS control. You can rename a basic block that is contained in an object that is under QVCS control. The name of the basic block is not changed in earlier versions of the object.

You **cannot** rename a Library once a CBT that exists in it is placed under QVCS control. Libraries are not placed in QVCS, but their names are part of the CBT identification.

5.9.1 Renaming a basic block when a parameter connection exists

There are no restrictions placed on renaming a basic block that is referenced by another object. Information is recorded that permits Control Builder to update the references to renamed blocks, if an older version of an object making the reference is restored.

5.10 Deleting a versionable object

- If an object that is capable of being versioned, such as a tagged block, CBT, or UDT, has never been under QVCS control, delete it just as you would in a No-QVCS system. If an object is under QVCS control, you must check in the object before you can delete it. You invoke the delete process in the same way for both cases.
- The delete removes the object from the Project or Library tab and leaves all of its information saved in the QVCS. It is marked as *Deleted from the tree* in QVCS Query window. For an object that had been checked in to QVCS, it is still possible to view its history and revert the object in QVCS.

Note: When an object which is under control of QVCS is deleted from CB (Project or Library Tree), it continues to exist in QVCS. In QVCS a GUID is appended to the name of the deleted object and the QVCS marks it as “deleted from tree”. This deleted object is listed under QVCS Query “deleted from tree”. If needed, user can bring this object back into ERDB by using revert options available.

- Users are responsible for complying with the operating procedures at their site by performing the necessary backup/archiving actions before deleting any object.

5.10.1 Deleting multiple objects

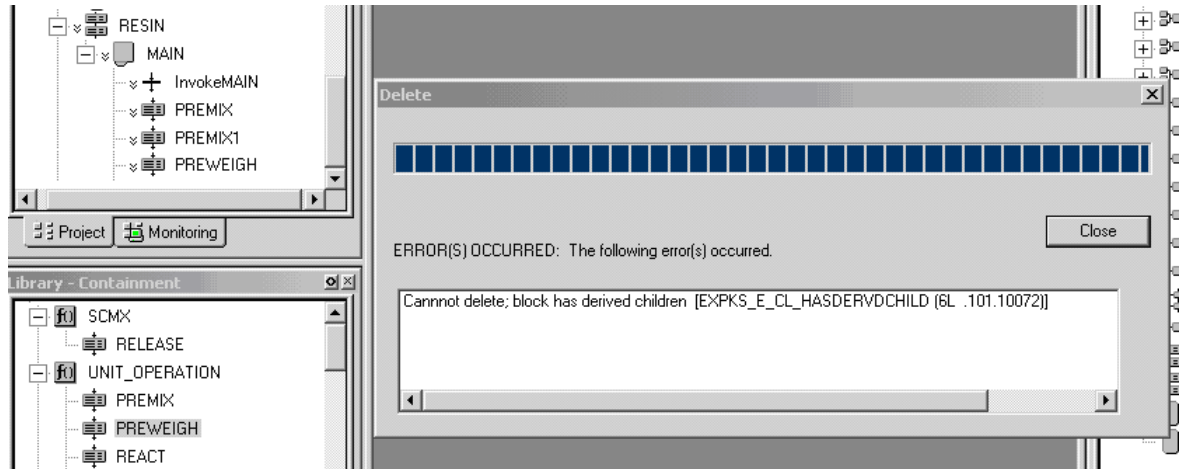
You can select multiple objects for deletion in a single transaction. A confirmation dialog listing the objects selected for deletion will ask for confirmation of the action. If any of the versioned objects selected for deletion is not checked in, an error message will be displayed, delete action will be halted, and none of the selected objects will be deleted. The user is responsible for taking the appropriate corrective action, such as only selecting versioned objects that are checked in.

5.10.2 Deleting a user defined template (UDT)

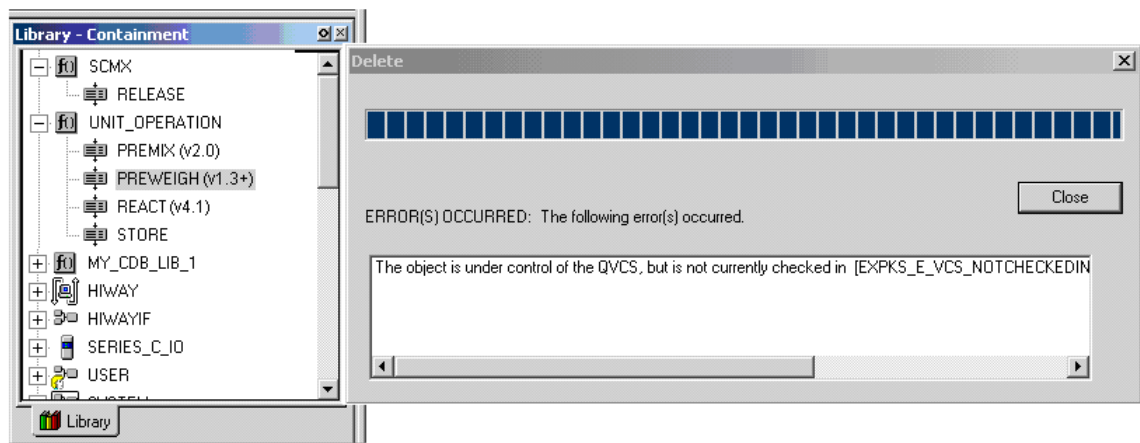
- You cannot delete a UDT with derived children. You must first delete all of the derived children or change their derivation parent before you can delete the parent. The difference between a Full-QVCS and No-QVCS system is that a UDT can have multiple versions in the library. This means that a given version of a template cannot have any dependent children for it to be deleted. Other versions of that template with dependent children may still exist in the library.
- There is a difference in the delete behavior for UDT and strategies. When you delete a strategy from Project, it is also marked as Deleted from the tree in the QVCS Query window. When UDT is deleted from the library, it is not marked as Deleted from the tree in the QVCS Query window unless it is the only version that is currently in the library. The existence of multiple versions is not allowed.

5.10.3 Deleting a custom block type (CBT)

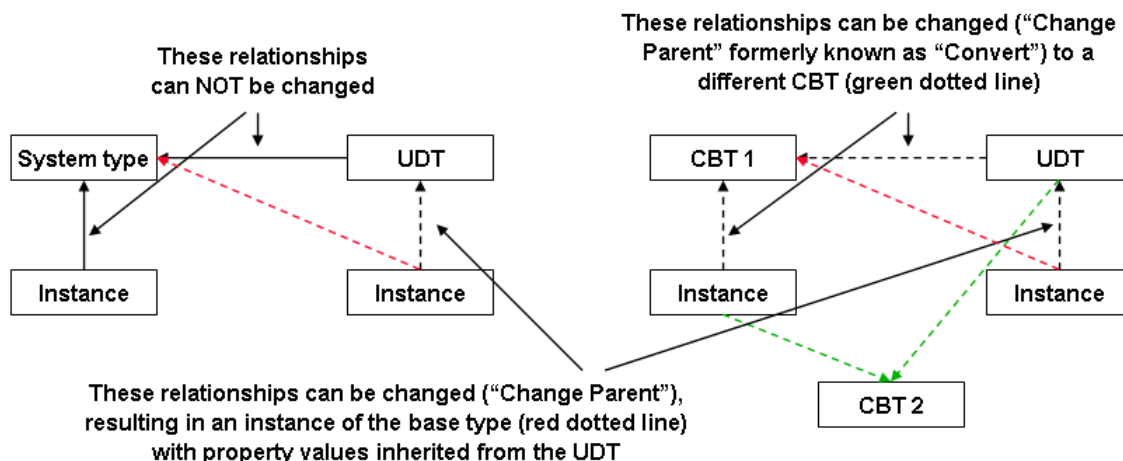
- You cannot delete a CBT from the Library, if any instance of it exists in the Project/Monitoring tab. This is true whether the CBT is in QVCS or not. You can use the Search function to find instances of CBT. For example, the following illustration shows the error generated when a user tries to delete PREWEIGH when RESIN contains an instance of it.



- An object with an instance of a CBT cannot be deleted if it is loaded to the Monitoring tab. This is true whether the object is in QVCS or not, and is the same rule that is currently enforced in R301.
- You can delete a CBT that is under QVCS control as long as it is checked in and no objects are in the Library or Project tabs that reference it. For example, the following illustration shows the error generated when a user tries to delete PREWEIGH while it is checked out. This example assumes that there are no instances of PREWEIGH since the validation for instances are done before the QVCS validation.



- An object with an instance of a CBT can be deleted when it is under control of QVCS, if it is checked in.
- Validation for an instance is done prior to doing the validation for the Type.
- A library can be deleted if it is empty. A library itself is not a versioned object.
- You can reuse the name of a deleted CBT on a new CBT. To revert to the deleted CBT or any deleted object, the name must not be currently used by any object in the ERDB. For a CBT this name also includes the library. Thus, the CBT with the same name will have to be either deleted, if checked into QVCS, or renamed, if not checked into QVCS, before the revert action.
- The rules governing the behavior of CBTs and UDTs for QVCS operations differ slightly. While a system type instance cannot have its relationship with the type changed, it is possible to change the type for instances of CBTs as shown in following example illustration.



5.10.4 Example CBT delete scenarios

The following table outlines various delete scenarios based on the “Graphical Example of Layered Recipe Check In and Check Out Scenario” on page 81 provided in another section of this document. These scenarios assume that the delete operation is being performed on the objects in the **Library** and **Project** tabs. The objects in the **Monitoring** tab do not follow these rules. The QVCS ensures that any instances of CBT that exist in the **Monitoring** tab will be supported by the CBT in the **Library** tab.

Table 8: Layered Recipe Scenarios for Delete Operation

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform Delete)	CBT Instance Only (Perform Delete)	CBT + Instance (Perform Delete)
1	2.0 (Checked In)	3.0 (Checked In)	Cannot do a delete since an instance exists.	Allowed	Cannot select both CBT and an instance of a CBT at the same time in CB.
2	2.0 (Checked In)	3.0 (Checked Out)	Cannot do a delete since an instance exists	Cannot do a delete since instance is checked out	N/A
3	2.0 (Checked Out)	3.0 (Checked In)	Not Possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out
4	2.0 (Checked Out)	3.0 (Checked Out)	Cannot do a delete since an instance exists.	Cannot do a delete since instance is checked out	N/A
5	2.0 (Checked In)	Instance 3.0 is loaded to Controller and is Checked In.	Cannot do a delete since an instance exists.	Cannot do a delete since instance is loaded	N/A
6	2.0 (Checked Out)	Instance 3.0 is loaded to Controller and is checked In	Not possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out	Not possible since instance cannot be checked in if CBT is checked out

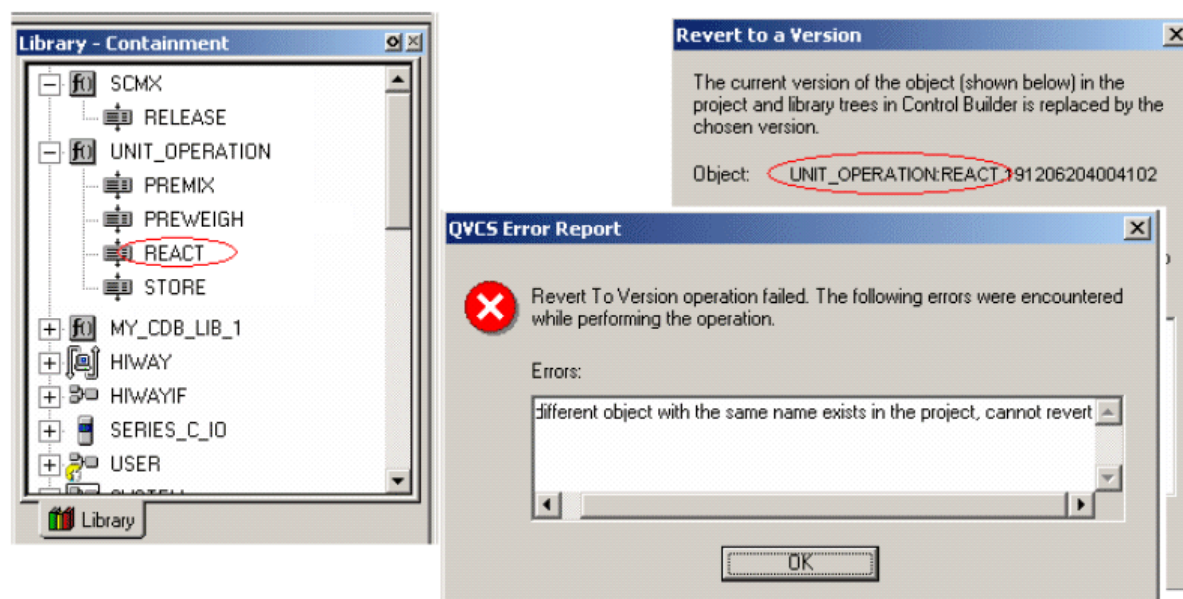
Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform Delete)	CBT Instance Only (Perform Delete)	CBT + Instance (Perform Delete)
7	2.0 (Checked In)	Instance 3.0 Checked Out. (After it is loaded to the controller)	Cannot do a delete since an instance exists.	Cannot do a delete since instance is loaded	N/A
8	2.0 (Checked Out)	3.0 Checked Out (After it is loaded to controller)	Cannot do a delete since an instance exists.	Cannot do a delete since instance is loaded	N/A
9	CBT is Deleted from Library	Deleted from Project	N/A	N/A	N/A
10	2.0 (Checked In)	Deleted from Project	Allowed	N/A	N/A
11	2.0 (Checked Out)	Deleted from Project	Cannot do a delete since CBT is checked out	N/A	N/A

5.11 Restoring Deleted Object

You can restore an object that is in QVCS after it has been deleted from the Project or Library tab in Control Builder by using the Revert function. You can initiate a revert to version with label action to restore the version of the object back to the Project or Library tab; but it may not be the most recent version of the deleted object, depending on the version/label selection.

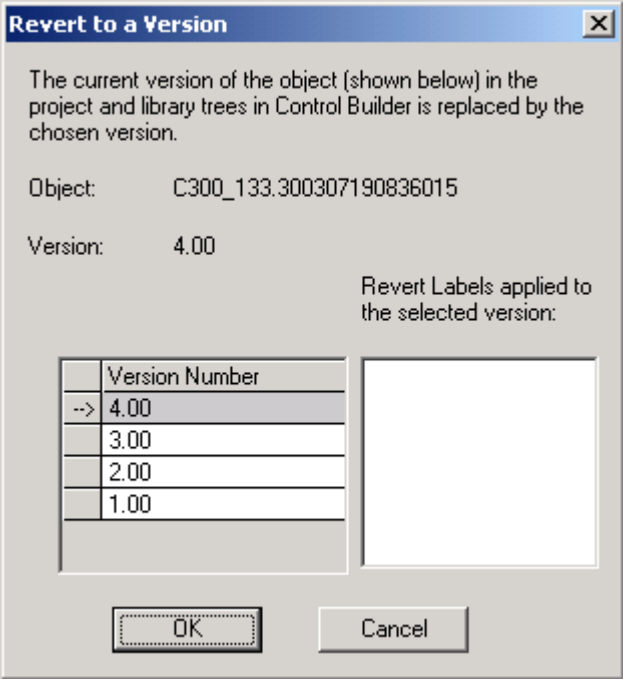
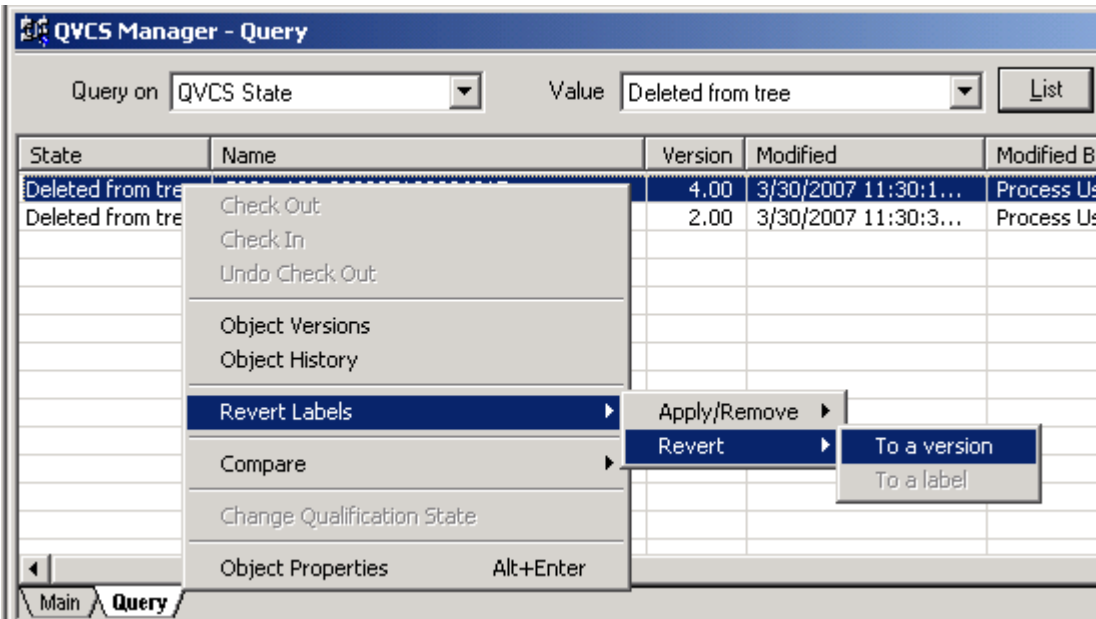
See the “QVCS Revert Operations” on page 133 section for more information about using revert operations.

- In QVCS, deleted objects have a status of “Deleted from tree”, and can be located using the standard QVCS query for the same. Deleted objects have a unique name identifier appended to their name so that the original name can be reused. For example, if UNIT_OPERATION:REACT is deleted from the library, then its name in QVCS will be UNIT_OPERATION:REACT.nnnnnnnnnnnnnnn, where the “ns” represent a 15 character numeric sequence. This behavior is the same for CBT as for any object in QVCS.
- If you also delete the library that was the location of a CBT, when you restore the CBT the library name is also restored. Since libraries are not versioned objects, other CBT's that may have been located in the library (but had been deleted) will not be restored. If the entire library is to be reconstructed, each CBT must be manually restored using the Revert function
- If you plan on re-using the name of a deleted object, the deleted object cannot be restored if its name conflicts with an existing object in the ERDB. As shown in the following example, REACT had been deleted from the UNIT_OPERATION library. Then, a new REACT had been created, and then an attempt was made to restore the original REACT. This is not allowed since even though the REACT new object is the same type of object, it is in a fact a completely different object and so it cannot be overwritten. You can use the QVCS “Retrieve” function described in “Retrieving Version of an Object” on page 112 in this document to generate export files without having to restore the deleted object to the library.



To restore object using revert operation

- 1 In the **Tools** menu, click **QVCS Manager** to call up the QVCS Manager Main window.
- 2 Click the **Query** tab to view it.
- 3 Click the down arrow button in the **Query on** box and select **QVCS State**.
- 4 Click the down arrow button in the **Value** box and select **Deleted from tree**.
- 5 Click the **List** button.
- 6 Right-click the deleted object you want to revert and select **Revert Labels > Revert > To a version** to call up the **Revert to a Version** dialog.



- 7 Select the desired version in the **Version Number** list box with applied label, as applicable.
- 8 Click the **OK** button.
- 9 Wait for the **Revert to a Version** dialog to close.
Check that selected version of the object is restored to the **Project** or **Library** tab, as applicable.
- 10 This completes the procedure.

5.12 Editing CBT in QVCS

If you make one of the following major changes to a CBT, you must do a *Save As* operation, if an instance of the CBT exists.

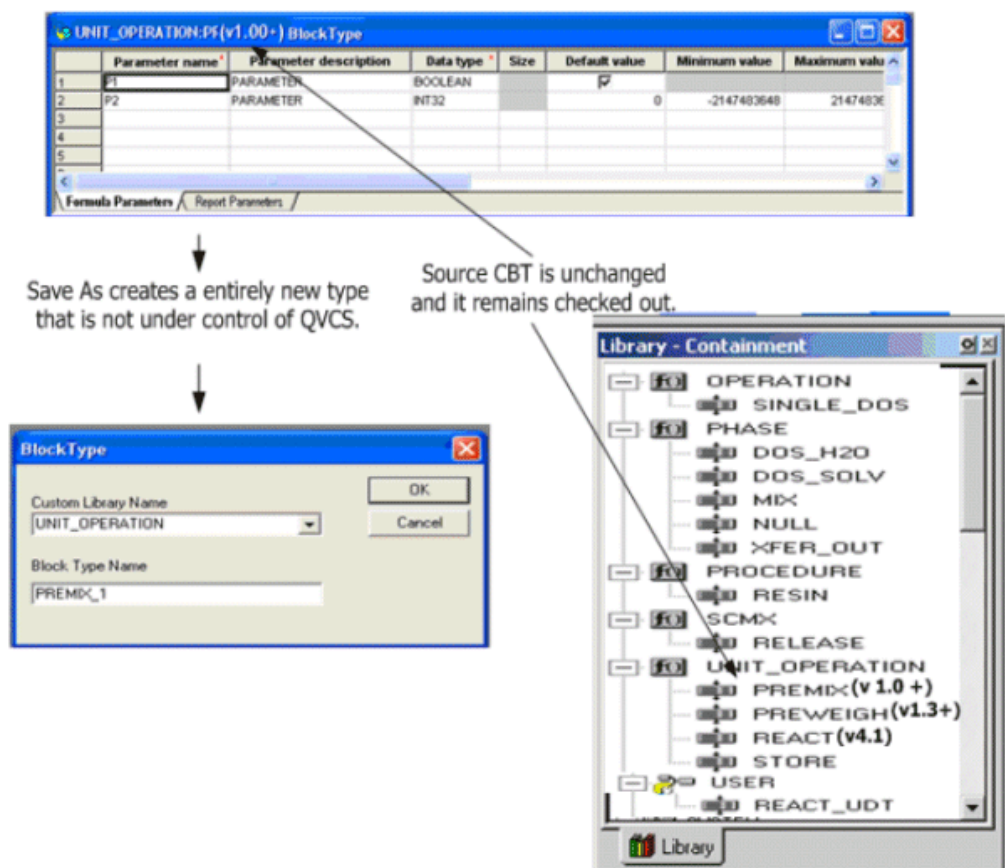
- Re-name a parameter
- Delete a parameter
- Change a parameter data type
- Change a parameter dimension

Any other change is considered a minor change, if you can do a *Save* operation. If you try to do a *Save* operation after a major change, you will get an error message notification. See error code 12505 in the Control Builder Error Codes Reference for more information.

If the CBT is not under control of QVCS, there cannot be any instances of the CBT in QVCS. If any instances are in QVCS, the QVCS rules require the CBT to be under control of QVCS.

5.12.1 Rules for CBT under QVCS

- Can only open CBT that is Checked In as Read Only.
 - The CBT Block Properties Form and View Type Display (PDE) are always read only.
 - You can do a *Save As* operation from an edit type display to create a new type from a checked in CBT.
 - For CAB types that are checked in, the *Save to ERDB*, *Save PDE Data*, and *Save-Renew PDE Data* menu selections are disabled. For PHASE blocks and CDB types that are checked in, the **Save** menu item is disabled.
 - When you do a *Save As*, the source of the edit is unchanged.
- You must check out a CBT that is under QVCS to edit it.
 - If no major change is made, you can *Save* the CBT so that the type is updated
 - If a major change is made, you must always do a *Save As* operation on the CBT. This creates a new type that can be checked in to QVCS. It is handled as a completely distinct QVCS object, with its own QVCS history, and so on. The Source CBT is unchanged and it remains checked out, as shown in the following illustration.



- Remember that checking out a CBT also checks out all objects with instances of the CBT, and that the instances cannot be checked in until the CBT is checked in. See the “Checking out an object from QVCS” on page 66 for more information.
- The following apply when updating instances of edited CBT.
 - Checking out a CBT also checks out all objects with instances of the CBT, and the instances cannot be checked in until the CBT is checked in.
 - The behavior of the instance can change, however, since the definition of access lock, default value, and configuration load may have been modified.
 - Since the instance was checked out when the CBT was modified, it must be checked in and re-qualified, regardless of the actual change that was made to the CBT (QVCS does not track the type of change).
 - For example, adding a parameter does not change the behavior of the instance, but still requires re-qualifying the object.
 - For example, changing the default value for a parameter will change the behavior of the instance if the instance is using the default value (a non-default value will not be replaced).

5.12.2 Example CBT edit scenarios

The following table outlines various edit scenarios based on the “Graphical Example of Layered Recipe Check In and Check Out Scenario” on page 81 provided in another section of this document. Since you cannot edit type and instance together, the CBT plus Instance column does not apply (N/A).

Table 9: Layered Recipe Scenarios for Edit Operation

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform Load)	CBT Instance Only (Perform Load)	CBT + Instance(Perform Load)
1	2.0 (Checked In)	3.0 (Checked In)	Cannot Edit CBT. (Opens in Read Only mode)	Cannot edit the Instance. (Opens in Read Only mode)	N/A
2	2.0 (Checked In)	3.0 (Checked Out)	Cannot Edit CBT. (Opens in Read Only mode)	Allowed	N/A
3	2.0 (Checked Out)	3.0 (Checked In)	Not possible since instance cannot be check in if CBT is checked out	Not possible since instance cannot be check in if CBT is checked out	N/A
4	2.0 (Checked Out)	3.0 (Checked Out)	Allowed	Allowed	N/A
5	2.0 (Checked In)	Instance 3.0 is loaded to Controller and is Checked In.	Cannot Edit CBT. (Opens in Read Only mode)	Cannot edit the Instance. (Opens in Read Only mode)	N/A
6	2.0 (Checked Out)	Instance 3.0 is loaded to Controller and is checked In	Not possible since instance cannot be check in if CBT is checked out	Not possible since instance cannot be check in if CBT is checked out	N/A
7	2.0 (Checked In)	Instance 3.0 Checked Out. (After it is loaded to the controller)	Cannot Edit CBT. (Opens in Read Only mode)	Allowed	N/A
8	2.0 (Checked Out)	3.0 Checked Out (After it is loaded to controller)	Allowed	Allowed	N/A
9	CBT is Deleted from Library	Deleted from Project	Not Allowed. Since object does not exist.	Not Allowed. Since object does not exist.	N/A
10	2.0 (Checked In)	Deleted from Project	Cannot Edit CBT. (Opens in Read Only mode)N/A	Not Allowed. Since object does not exist.	N/A
11	2.0 (Checked Out)	Deleted from Project	Allowed	Not Allowed. Since object does not exist. N/A	N/A

5.13 Using Change Parent with CBT

You can use the Change Parent (previously known as *Convert*) function to reassign the base CBT of an instance to another CBT. Since this requires a modification of the instance, QVCS enforces the following rules on the Change Parent function.

5.13.1 Rules for using Change Parent with CBT instance

- If the container of the CBT instance is not in QVCS, Change Parent occurs normally.
- If the container of the CBT instance is in QVCS, it must be checked out since Change Parent is equivalent to an edit.
- The QVCS status of CBT target or source (original parent) does not matter. Either CBT can be checked in, checked out, or not in QVCS since the Change Parent function does not impact the CBT configuration, only the instances.

5.13.2 Initiating bulk Change Parent operation

Beginning in R400, you can select multiple objects at one time for a Change Parent operation. The preceding rules for using Change Parent with a single instance will be applied to each object that is selected for the bulk Change Parent operation.

The Change Parent user interface reports the success or failure of each selected object. QVCS generates the appropriate error message for each object, if it is not checked out. The user will then need to check out the object and redo the Change Parent for it. One way to “guarantee” success for the bulk operation is to check out the current parent of the objects. While you can perform this operation without checking out the CBT, doing so results in the check out of all its instances, placing them in the correct state.

You can use the “Viewing History details of an Object” on page 106 to locate object dependencies.

5.13.3 Example CBT Change Parent scenarios

The following table outlines various Change Parent scenarios based on the “Graphical Example of Layered Recipe Check In and Check Out Scenario” on page 81 provided in another section of this document.

Table 10: Layered Recipe Scenarios for Change Parent Operation

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform Change Parent)	CBT Instance Only (Perform Change Parent)	CBT + Instance (Perform Change Parent)
1	2.0 (Checked In)	3.0 (Checked In)	Does not apply since Change Parent is only supported for instances.	Cannot do a change parent since instance is checked in	Does not apply since Change Parent is only supported for instances.
2	2.0 (Checked In)	3.0 (Checked Out)	N/A	Allowed	N/A
3	2.0 (Checked Out)	3.0 (Checked In)	Not Possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out
4	2.0 (Checked Out)	3.0 (Checked Out)	N/A.	Allowed	N/A
5	2.0 (Checked In)	Instance 3.0 is loaded to Controller and is Checked In.	N/A.	Cannot do a Change Parent since instance is checked in	N/A

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform Change Parent)	CBT Instance Only (Perform Change Parent)	CBT + Instance (Perform Change Parent)
6	2.0 (Checked Out)	Instance 3.0 is loaded to Controller and is checked In	Not possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out	Not possible since instance cannot be checked in if CBT is checked out
7	2.0 (Checked In)	Instance 3.0 Checked Out. (After it is loaded to the controller)	N/A	Allowed	N/A
8	2.0 (Checked Out)	3.0 Checked Out (After it is loaded to controller)	N/A.	Allowed	N/A
9	CBT is Deleted from Library	Deleted from Project	N/A	N/A	N/A
10	2.0 (Checked In)	Deleted from Project	N/A	N/A	N/A
11	2.0 (Checked Out)	Deleted from Project	N/A	N/A	N/A

5.14 Viewing Information for a Version of an Object

You can view configuration details for a given version of an object through the Version Window.

- For viewing configuration details for a given version of an object, you must have checked in objects to QVCS.
- The procedure is the same for all versioned objects including CBT objects.
- You must select an object in the given **QVCS Manager** window for the **Object Versions** menu selection to be available.
- A CDB type can be used to hold data that is shared by multiple block instances within a CBT container or across multiple CBT containers. There are no CDB fixed data parameters (FDPs) whose default values are configurable by the user when the CDB is defined in Control Builder so version view does not show FDPs for the CDB.

To View Object Version Information

- 1 In the Control Builder **Project** tab or **Library** tab for CBT, select the object that you want to review.
- 2 Right-click the object and select **QVCS Manager** to call up the application.
- 3 In the **Main** window, right-click the listed object and select **Object Versions** to call up the **Versions** window. See the previous “QVCS Manager Versions Window” on page 37 for more information about the window.
- 4 View the Details box for information about the selected object. Click on a heading in the box to expand or collapse the details, as shown in the following illustration.
Note that details shown are tailored to the selected object and differ for CBT objects. Most of the information is self-explanatory and reflects what is configured on the object's Module Properties form.
- 5 Repeat the Steps above to view details for other objects.
- 6 This completes the procedure.

You can alternately select the function from the QVCS **Tools** menu.

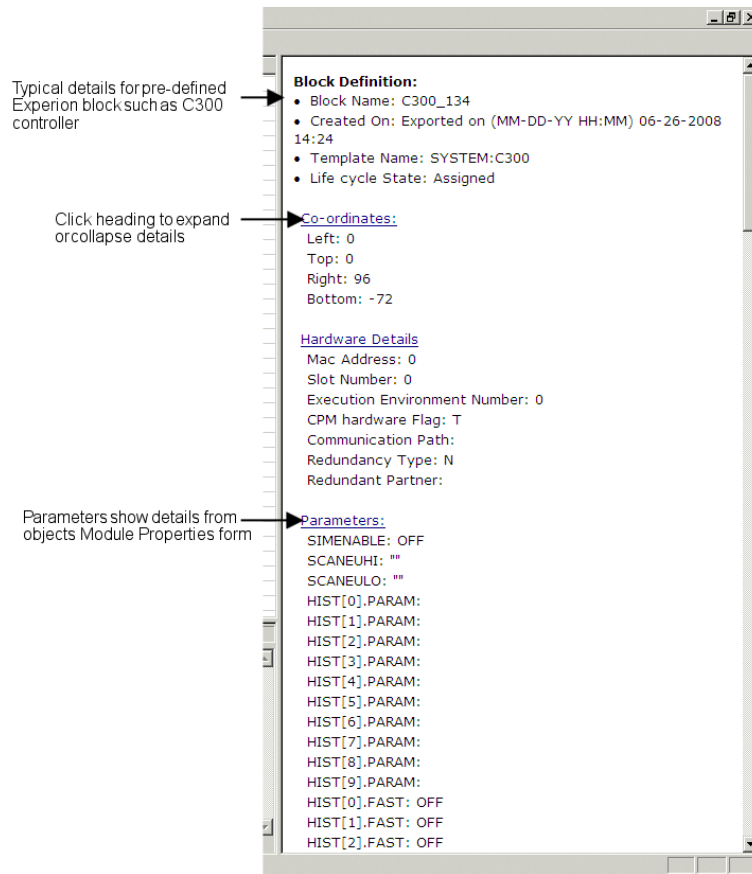


Figure 15: Typical details shown for Object Versions

5.15 Viewing History details of an Object

You can view historical details for a given version of an object through the History Window.

- For viewing history details of an object, you must have checked in objects to QVCS.
- The procedure is the same for all versioned objects including CBT objects.
- You must select an object in the given **QVCS Manager** window for the **Object History** menu selection to be available.

To View Object History Details

- 1 In the Control Builder **Project**tab or **Library** tab for CBT, select the object that you want to review.
- 2 Right-click the object and select **QVCS Manager** to call up the application.
- 3 In the **Main** window, right-click the listed object and select **Object History** to call up the **History** window. See the previous “QVCS Manager History Window” on page 39 for more information about the window.
- 4 Define the desired filter criteria for the history data through the configuration entries on this window.
- 5 Once you define the filter criteria, the history details for the selected object are displayed in the window, as shown in the following illustration.
- 6 Repeat the Steps above to view details for other objects as applicable.
- 7 This completes the procedure.

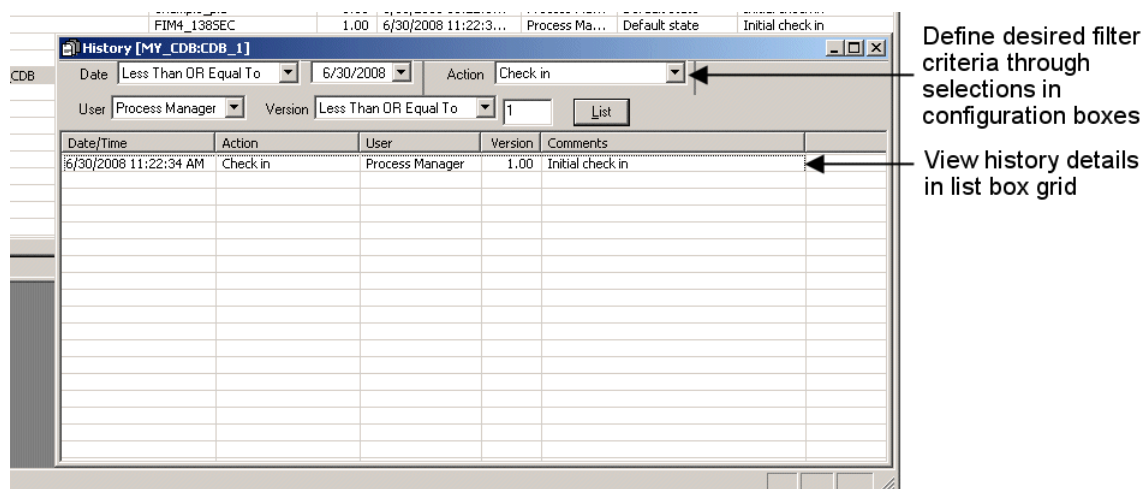


Figure 16: Typical view of History details for an object.

You can alternately select the function from the QVCS **Tools** menu.

5.16 Viewing Reference data for Objects

You can view reference data about the version dependencies that exist for an object to assist you in making decisions about reverting and checking out objects. You can select multiple objects as the basis for the reference view.

- For viewing reference data for objects, you must have checked in objects to QVCS.
- The procedure is the same for all versioned objects including CBT objects.
- You must select at least one object in the given **QVCS Manager** window for the **Object References** menu selection to be available.
- The Show References selections of **Uses** and **Used By** only refer to object derivations and instances and their version dependencies. It does not include connections, assignment relationships, and so on.
- You can click on a column heading in the grid to sort the view.
- Objects that were selected when you selected **Object References** from the menu are included as the “Basis” objects for the view. These objects are indicated by an asterisk (*) in the column labeled **Basis**.
- The reference view shows the entire “tree” of references from the starting object to show the entire set of relationships.
- You can copy Target Objects listed in the view and paste them into other applications or to the Main window of QVCS Manager, where they will be available for additional operations as any objects in that window. You can also select Target Objects to open another reference view, such as Object Versions, Object History, and so on, in QVCS Manager that support multiple windows.
- The copy operation copies the target library name, the target object name, and the target version and removes any duplicate entries. The data is copied as text with tab delimiters that can be pasted into any clipboard text enabled application, such as Microsoft Excel. A Paste selection is added to the Edit menu that is available when the QVCS Main window is active. If the clipboard contents contain the properly formatted text, the paste operation inserts the library name, object name, and version number into the Main window. If there are any rows selected in the Main window, the paste replaces them with the clipboard contents. Once the paste is complete, the data in the remaining columns will be retrieved from the QVCS data repository.
- You can use the **File** menu **Print Preview** and **Print** selections to generate a printed report that shows the contents of the active **Reference View** window.

To View Object References Data

- 1 In the Control Builder **Project** tab or **Library** tab for CBT, select the object or objects that you want to review.
- 2 Right-click the object or multiple selected objects and select **QVCS Manager** to call up the application.
- 3 In the **Main** window, right-click the listed object or multiple selected objects and select **Object References** to call up the **Reference View** window. See the previous “QVCS Reference View Window” on page 42 for more information about the window.
- 4 In **Reference View** window, select desired radio button to view references by **Uses** or **Used By** criteria.
- 5 View the list box grid for information about the selected object(s), as shown in the following illustration.
- 6 Repeat the Steps above to view data for other objects as applicable.
- 7 This completes the procedure.

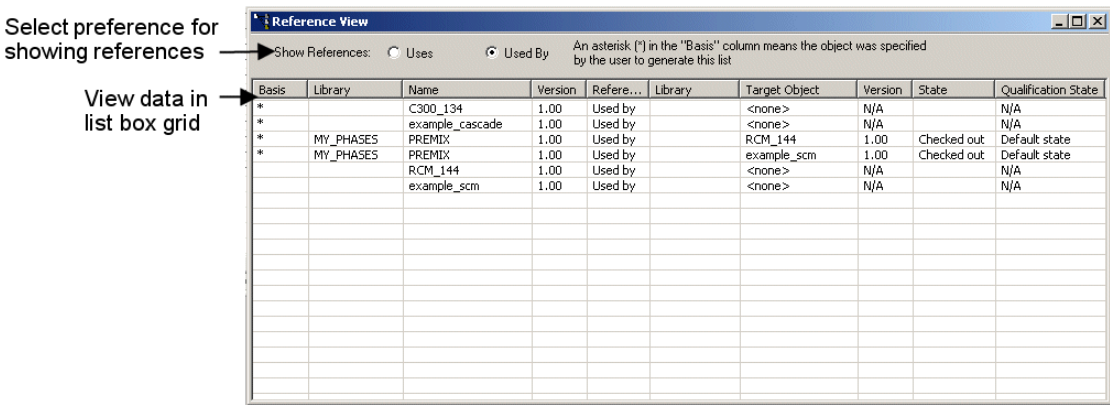
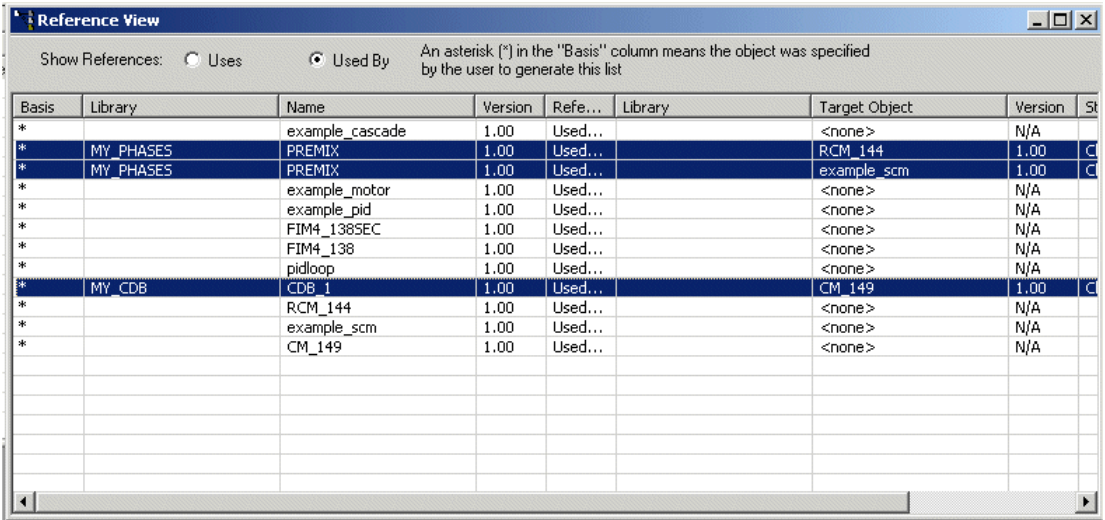


Figure 17: Typical view of Reference data for objects

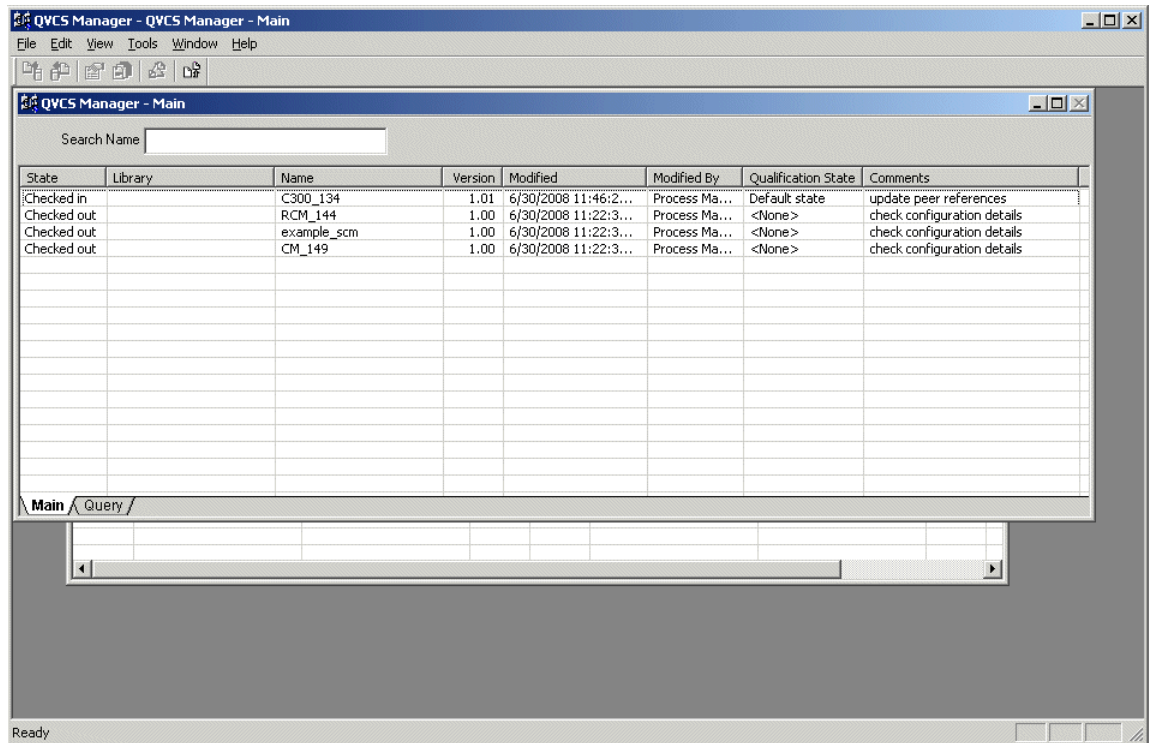
You can alternately select the function from the QVCS **Tools** menu.

To copy data from Reference View

- 1 Open **Reference View** as outlined in the previous procedure ().
- 2 In the **Reference View**, select the object or objects you want to copy. Be sure the objects selected have a **Target Object** specified other than <none>;, as shown in the following illustration. Otherwise, the copy function will **not** work.



- 3 Press **ctrl+C** or right-click and select **Copy** from the list.
- 4 Click the **Main** tab to bring the window on top. click a row in the grid, and press **ctrl+V** or right-click and select **Paste** from the list to copy Target Object(s) to the **Main** window, as shown in the following illustration.



- 5 Open another applicable application and repeat Step 4 to paste the text to any clipboard text enabled application such as Microsoft Excel.
- 6 Click the **Reference View** window to bring it on top. Click the close button in the upper right-hand corner to close the window.
- 7 This completes the procedure.

5.17 Example Reference View Scenario

Related topics

“Sample configuration” on page 110

“Example Reference Views for sample configuration” on page 110

5.17.1 Sample configuration

The following illustration depicts the object dependencies for the sample configuration used for this scenario. As shown, the configuration includes:

- Two CBTs named CBT_1 and CBT_2,
- Three UDTs named UDT_1, UDT_2, and CM_UDT.

In this sample, the UDT_1 is a template created from CBT_1, and UDT_2 is derived from UDT_1. The CM_UDT contains an “instance” of UDT_1, UDT_2, and CBT_2. The CM_UDT_INST was instantiated from CM_UDT, and then an instance of CBT_2 was added to it.

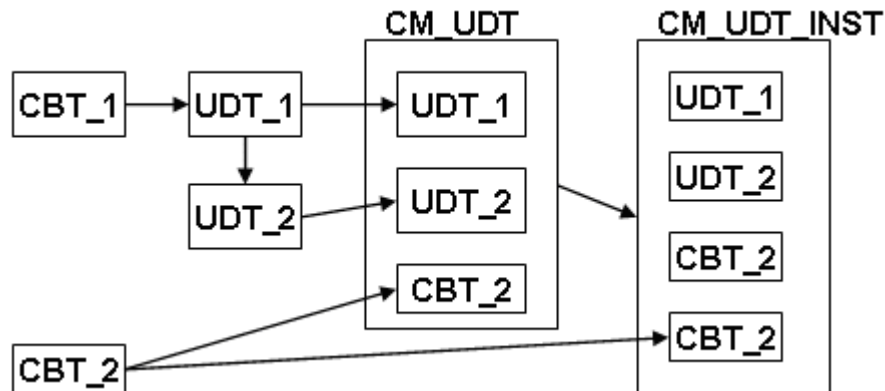
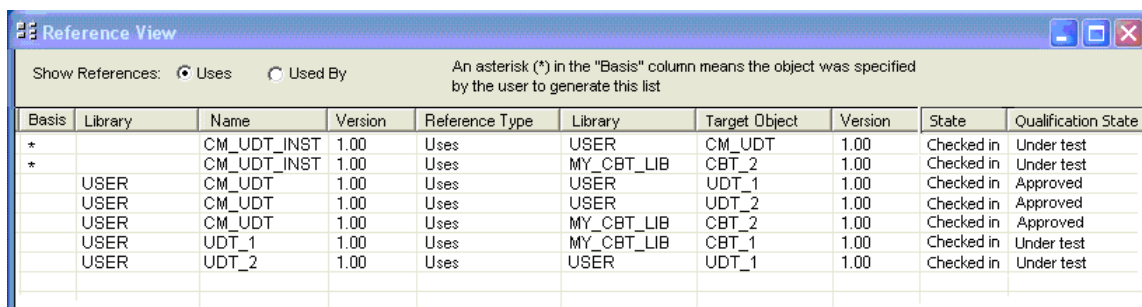


Figure 18: Sample Configuration Object Dependencies

5.17.2 Example Reference Views for sample configuration

- With **CM_UDT_INST** object selected in QVCS **Main** window, selecting **Object References** from the **Tools** menu results in the following **Reference View** with **Uses** selected. The **Uses** view option is helpful for determining what objects and versions must exist for a check out or revert operation to succeed.



Reference View

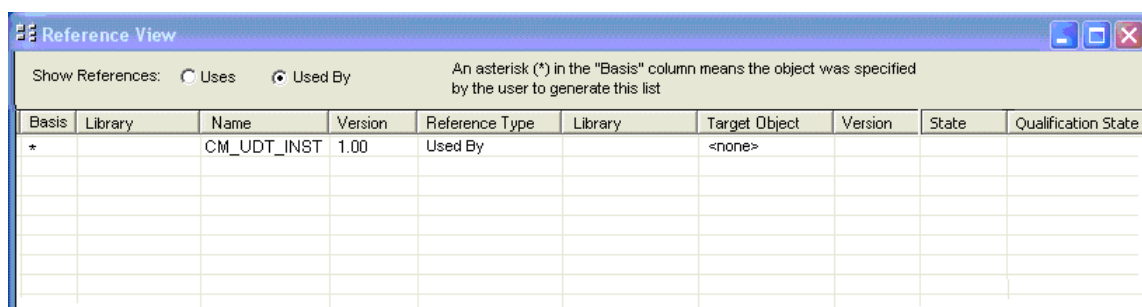
Show References: ☒ Uses ☐ Used By

An asterisk (*) in the "Basis" column means the object was specified by the user to generate this list

Basis	Library	Name	Version	Reference Type	Library	Target Object	Version	State	Qualification State
*		CM_UDT_INST	1.00	Uses	USER	CM_UDT	1.00	Checked in	Under test
*		CM_UDT_INST	1.00	Uses	MY_CBT_LIB	CBT_2	1.00	Checked in	Under test
	USER	CM_UDT	1.00	Uses	USER	UDT_1	1.00	Checked in	Approved
	USER	CM_UDT	1.00	Uses	USER	UDT_2	1.00	Checked in	Approved
	USER	CM_UDT	1.00	Uses	MY_CBT_LIB	CBT_2	1.00	Checked in	Approved
	USER	UDT_1	1.00	Uses	MY_CBT_LIB	CBT_1	1.00	Checked in	Under test
	USER	UDT_2	1.00	Uses	USER	UDT_1	1.00	Checked in	Under test

Figure 19: Example Reference View for CM_UDT_INST with Uses Selection

- Switching to the **Used By** view option, results in the following **Reference View** since CM_UDT_INST is the only object selected. The following view shows that CM_UDT_INST is not used by any other object. The **Used By** view option is helpful for determining what objects will be automatically checked out when a CBT or UDT is checked out.



Reference View

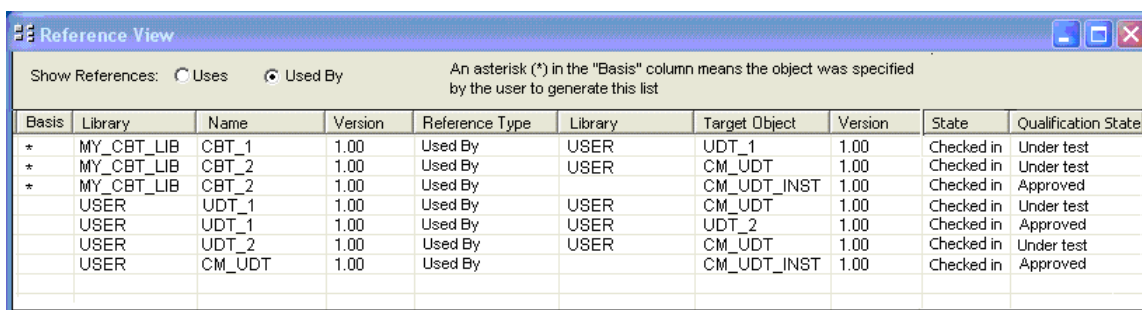
Show References: ☐ Uses ☒ Used By

An asterisk (*) in the "Basis" column means the object was specified by the user to generate this list

Basis	Library	Name	Version	Reference Type	Library	Target Object	Version	State	Qualification State
*		CM_UDT_INST	1.00	Used By		<none>			

Figure 20: Example Reference View for CM_UDT_INST with Used By Selected

- With CBT_1 and CBT_2 objects selected in QVCS **Main** window, selecting **Object References** from the **Tools** menu results in the following **Reference View** with **Used By** selected. With the **Uses** view option selected, the **Reference View** displays only CBT_1 and CBT_2 with both having a **Target Object** of <none>, which similar to the view in the previous figure. .



Reference View

Show References: ☐ Uses ☒ Used By

An asterisk (*) in the "Basis" column means the object was specified by the user to generate this list

Basis	Library	Name	Version	Reference Type	Library	Target Object	Version	State	Qualification State
*	MY_CBT_LIB	CBT_1	1.00	Used By	USER	UDT_1	1.00	Checked in	Under test
*	MY_CBT_LIB	CBT_2	1.00	Used By	USER	CM_UDT	1.00	Checked in	Under test
*	MY_CBT_LIB	CBT_2	1.00	Used By		CM_UDT_INST	1.00	Checked in	Approved
	USER	UDT_1	1.00	Used By	USER	CM_UDT	1.00	Checked in	Under test
	USER	UDT_1	1.00	Used By	USER	UDT_2	1.00	Checked in	Approved
	USER	UDT_2	1.00	Used By	USER	CM_UDT	1.00	Checked in	Under test
	USER	CM_UDT	1.00	Used By		CM_UDT_INST	1.00	Checked in	Approved

Figure 21: Example Reference View for CBT_1 and CBT_2 with Used By Selected

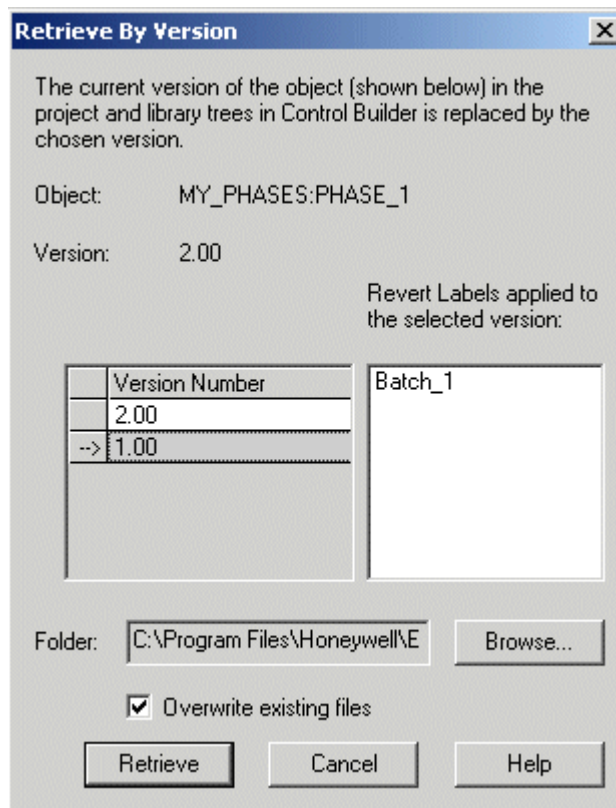
5.18 Retrieving Version of an Object

You can retrieve data for any version of an object for storage to a user selected location.

- For retrieving version of an object, you must have checked in object to QVCS.
- You use the retrieved files to import an object to the ERDB.
 - If imported to the same ERDB, the QVCS rules regarding import will be enforced.
 - If imported to a different ERDB, new objects will be created and the user will have to decide if they are to be placed in QVCS on that ERDB.
- Typically, you would not use the retrieve and import functions to the same ERDB for non-CBT blocks since the revert function accomplishes the same action.
- The retrieve function lets users access a CBT from QVCS that would not be accessible through check out or revert due to version rule restrictions.
- You can retrieve by revert label or object version. A retrieve by revert label will export the version of all objects that have the specified revert label. Note that a revert label cannot be applied to multiple versions of the same object.

To retrieve version of an object by version

- 1 In the Control Builder **Tools** menu, click **QVCS Manager** to call up the application.
- 2 Click the **Query** tab to view it.
- 3 Click the down arrow button in the **Query on** box and select desired query function such as **QVCS State**.
- 4 Click the down arrow button in the **Value** box and select desired value such as **Checked in**.
- 5 Click the **List** button.
- 6 Right-click the desired object you want to retrieve and select **Retrieve->By Version** to call up the **Retrieve by Version** dialog.



- 7 Select the desired version in the **Version Number** list box with applied label. The -> symbol identifies the selection.
- 8 Click the **Browse** button to navigate to the desired directory location.
- 9 Leave the **Overwrite existing file** check box checked to overwrite any identically named pre-existing files in the directory. Or, click the box to leave it blank and disable the overwrite function.
- 10 Click the **Retrieve** button.
- 11 Wait for the **Retrieve** dialog to close.
Check that the file for the selected version of the object is stored in the designated directory location.
- 12 This completes the procedure.
You can alternately select the Retrieve function from the QVCS **Tools** menu, **Main** window, **Versions** window, and so on. It is similar to Check In operations.

To retrieve version of an object by revert label

- 1 In the Control Builder **Tools** menu, click **QVCS Manager** to call up the application.
- 2 Click the **Query** tab to view it.
- 3 Click the down arrow button in the **Query on** box and select desired query function such as **QVCS State**.
- 4 Click the down arrow button in the **Value** box and select desired value such as **Checked in**.
- 5 Click the **List** button.
- 6 With no object selected in the list box, select **Retrieve->By Revert Label** to call up the **Retrieve By Revert Label** dialog.

Retrieve By Revert Label [X]

Retrieve objects with revert label: []

Retrieve object files to: [C:\Program Files\Honeywell\Experion PKS\Engineering Tools\Ixpport\] [Browse...]

☒ Overwrite existing files

Total number of objects with label: [0] Total number of objects selected: [0]

List Object with Revert Label:

Select	Library	Name	Version	Status

[Select All] [Clear All]

Retrieve progress: []

[Retrieve] [Close] [Help]

- 7 Click the down-arrow button in the **Retrieve objects with revert label** field and select the desired revert label from the list.
- 8 Leave the **Overwrite existing file** check box checked to overwrite any identically named pre-existing files in the directory. Or, click the box to leave it blank and disable the overwrite function.
- 9 The **Total number of objects with label** and **Total number of objects selected** are read only fields for reference.

- 10 Check the check box for each object you want to retrieve in the list box. You can use the **Select All** button to select all objects at once or the **Clear All** button to clear all check boxes at once.

11

Retrieve objects with revert label: Batch_1

Retrieve object files to: C:\Program Files\Honeywell\Experion PKS\Engineering Tools\Export\ Browse...

☒ Overwrite existing files

Total number of objects with label: 4 Total number of objects selected: 1

List Object with Revert Label:

Select	Library	Name	Version	Status
<input type="checkbox"/>		C300_134	1.00	
<input checked="" type="checkbox"/>	MY_CDB	CDB_1	1.00	
<input type="checkbox"/>	MY_PHASES	PHASE_1	1.00	
<input type="checkbox"/>	MY_PHASES	PREMIX_1	1.00	

Select All Retrieve progress: Retrieve Close Help

- 12 Confirm the selections made in the previous Steps. Click the **Retrieve** button to initiate the action. The button is only available after making the selections.
- 13 Monitor the **Retrieve progress** bargraph to track the status of the action. Note that the **Retrieve** button changes to **Cancel** while the retrieve function is in progress. You can click the **Cancel** button to abort the operation, if necessary. You will be prompted to confirm the cancel request. The same is true if you click the **Close** button during a retrieve operation. Any objects that were retrieved prior to the Cancel request will be saved and not deleted.

Retrieve By Revert Label

Retrieve objects with revert label:

Retrieve object files to:

☒ Overwrite existing files

Total number of objects with label: Total number of objects selected:

List Object with Revert Label:

Select	Library	Name	Version	Status
<input type="checkbox"/>		C300_134	1.00	
<input checked="" type="checkbox"/>	MY_CDB	CDB_1	1.00	Retrieved
<input type="checkbox"/>	MY_PHASES	PHASE_1	1.00	
<input type="checkbox"/>	MY_PHASES	PREMIX_1	1.00	

Retrieve progress:


- 14 When progress is completed and the Status field shows retrieved, click the **Close** button to exit the function or repeat the previous Steps to select another label or object, as desired.
- 15 This completes the procedure.
You can alternately select the Retrieve function from the QVCS **Tools** menu, **Main** window, **Versions** window, and so on. It is similar to Check In operations.

5.19 Comparing Versions of an Object Using Diff Tool

You can use the Diff Tool to help track and compare changes that are made to control strategies between versions by viewing and comparing differences between versions of a QVCS object.

- For comparing versions of an object, you must have checked in object to QVCS.
- You can compare versions for objects residing in the monitor database as long as you first choose the desired object in the **Monitoring** tab.
- You can compare a checked out version that has been modified and saved, but not checked in, to an existing version of the same object that resides in the QVCS database.

To compare versions in Diff Tool

- 1 In Control Builder, select object with version data you want to compare.
 - 2 In the **Tools** menu, click **QVCS Manager** to call up the QVCS Manager **Main** window.
You can alternately right-click selected object and select **QVCS Manager** from the list or click the QVCS button in the toolbar to call up the QVCS Manager **Main** Window.
 - 3 Select the object for the comparison operation in the list box of the **Main** window.
You can alternately select object in the **Query** window.
 - 4 In the **Tools** menu, select **Object Versions** to open the **Versions** window.
 - 5 Right-click the latest version of the object in the **Versions** window and select **Compare > With previous version** from the list to call up the **Diff Tool** window.
You can alternately select **With another version**, **With monitoring version**, or **With checked out version**, as applicable.
-
-  **Attention**
 ERDB will not be updated with the changes done from the monitoring side till you do an Upload Operation.
-
- 6 Compare data for different versions in the list boxes using menu selections and scroll bar, as required. See the previous “QVCS Difference Tool (Diff Tool) Window” on page 52 section for more information about the Diff Tool.
 - 7 In the **File** menu, click **Exit** to close the Diff Tool.
 - 8 This completes the procedure.

5.20 Loading QVCS Objects to a Controller

You can load objects that are checked in to QVCS with a qualification state that permits loading to a controller. The qualification state of a CBT or UDT has no impact on the ability to load an object. You cannot load objects that are checked out or are in a qualification state that sets its Load to Controller Allowed attribute to No to a controller.

You can load objects that are not in the QVCS to the Controller due to the relaxed load function. See the “Relaxed Loading Support for QVCS” on page 129 section for more information about the relaxed load function.

5.20.1 Typical version mismatch load scenario

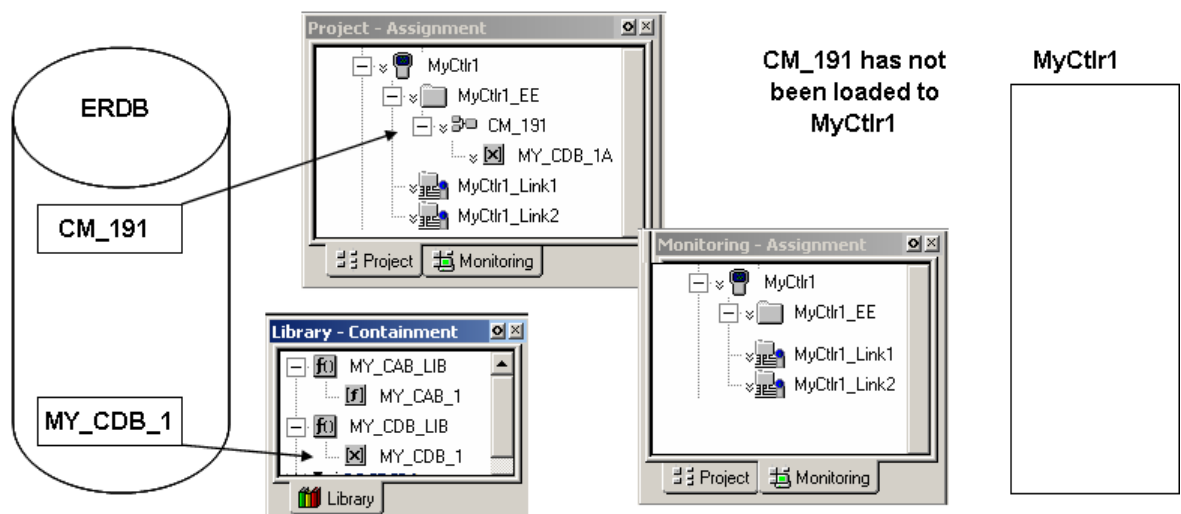
The **Project** tab always shows the current working version of an object. For example, assume that a CM named CM_001 has three versions numbered v1, v2, and v3; the v3 version is displayed in the Project tab since it has the most recent edits. All three versions are checked in to QVCS and are in a qualification state that permits loading to a controller. The user loads CM_001 v3 to a controller.

At a later time, the user reverts to CM_001 v1. This means CM_001 v1 replaces CM_001 v3 as the version now on display in the **Project** tab even though CM_001 v3 is still in the **Monitoring** tab and the controller. A delta flag symbol appears next to the CM_001 v1 icon in the **Project** tab signaling that the CM_001 version in **Monitoring** does not match the one in **Project**. Users are responsible for making sure that the correct version of an object is loaded in the controller.

5.20.2 Graphical example of CBT load scenario

The following illustrations provide a graphical representation of what occurs when a container that contains a CBT is loaded to a controller. In this example, the container is named CM_191 and it contains an instance of a CBT named MY_CDB_1 in the Library. The controller is named MyCtrl1.

Example of state before CM_191 is loaded to MyCtrl1



Example of state after CM_191 is loaded to MyCtrl1

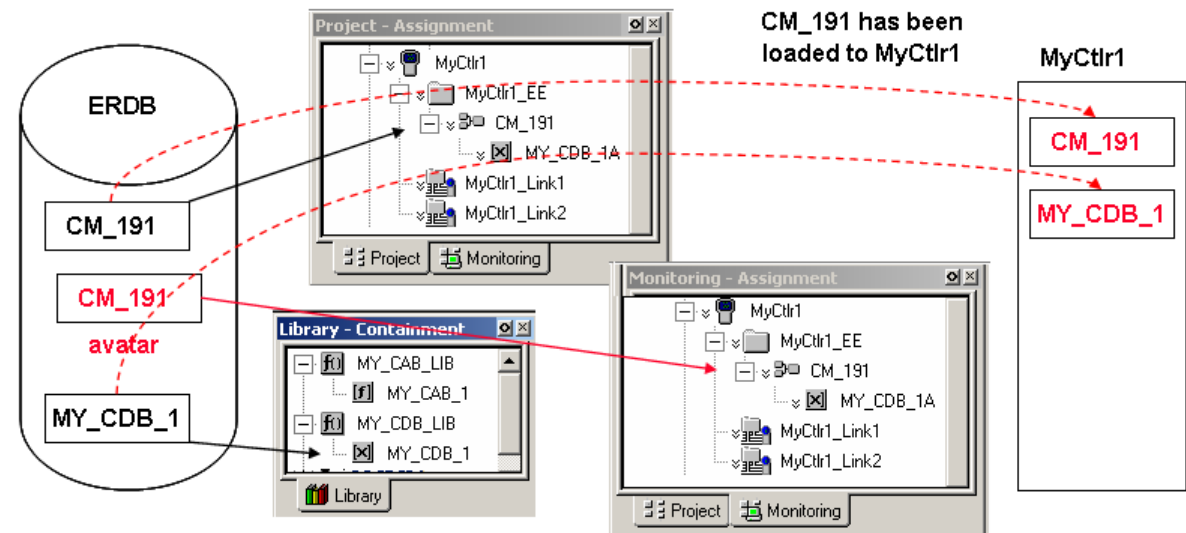


Figure 22:

Example of state after making change to MY_CDB_1 after CM_191 is loaded

For CAB, the program that is in the controller container instance can be different from the CAB that is in the controller since the program source is a parameter.

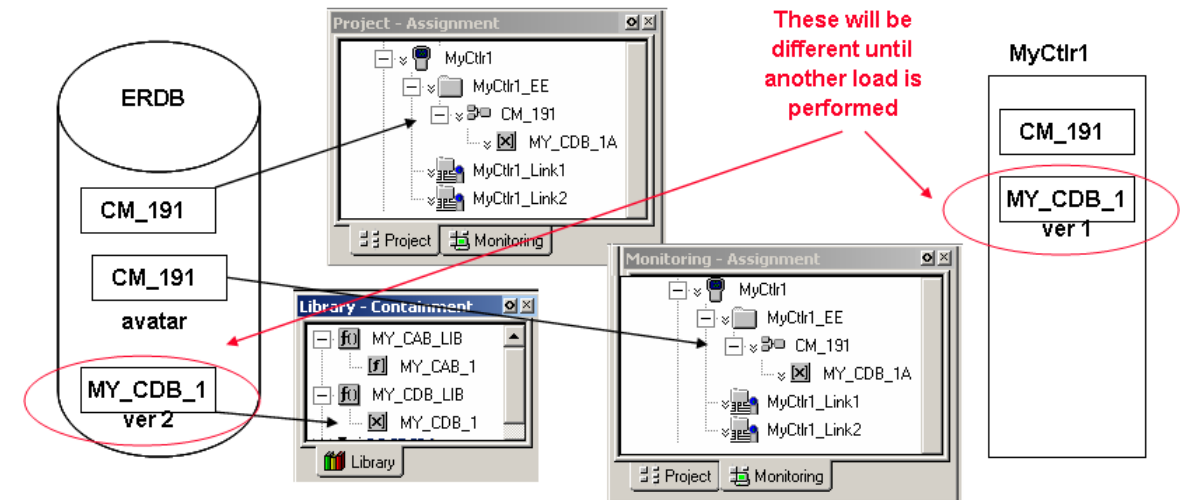


Figure 23:

If a parameter is added to the CBT after its instance is loaded, the Monitoring object has access to the parameter but it has not been loaded to the controller so an error will result if the user attempts to access the new parameter on the controller.

5.20.3 Example CBT load scenarios

The following table outlines various load scenarios based on the “Graphical Example of Layered Recipe Check In and Check Out Scenario” on page 81 provided in another section of this document. Since load is only supported for instances, load for CBT type only or CBT plus Instance does not apply (N/A).

Table 11: Layered Recipe Scenarios for CBT Load

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform Load)	CBT Instance Only (Perform Load)	CBT + Instance(Perform Load)
1	2.0 (Checked In)	3.0 (Checked In)	N/A	Allowed if qualification state of instance allows load	N/A
2	2.0 (Checked In)	3.0 (Checked Out)	N/A	Cannot do a load since instance is checked out	N/A
3	2.0 (Checked Out)	3.0 (Checked In)	N/A	Cannot exist since an Instance cannot be checked in if CBT is checked out	N/A
4	2.0 (Checked Out)	3.0 (Checked Out)	N/A	Cannot do a load since instance is checked out	N/A
5	2.0 (Checked In)	Instance 3.0 is loaded to Controller and is Checked In.	N/A	Allowed if qualification state of instance allows load	N/A
6	2.0 (Checked Out)	Instance 3.0 is loaded to Controller and is checked In	N/A	Cannot exist since an Instance cannot be checked in if CBT is checked out	N/A
7	2.0 (Checked In)	Instance 3.0 Checked Out. (After it is loaded to the controller)	N/A	Cannot do a load since instance is checked out	N/A
8	2.0 (Checked Out)	3.0 Checked Out (After it is loaded to controller)	N/A	Cannot do a load since instance is checked out	N/A
9	CBT is Deleted from Library	Deleted from Project	N/A	N/A	N/A
10	2.0 (Checked In)	Deleted from Project	N/A	N/A	N/A
11	2.0 (Checked Out)	Deleted from Project	N/A	N/A	N/A

5.21 Using upload or update to project commands

Keep the following considerations in mind when using the Upload or Update to Project commands with an object containing CBT instance. Refer to the Control Building User's Guide for more information about the Upload and Update to Project commands.

5.21.1 Considerations for upload command

- The QVCS state of the object in the **Project** tree does not have any effect on uploading to the Monitoring object
- The QVCS will guarantee that the version of CBT in the library is equal to or greater than the CBT being uploaded. There is no impact from having a different version of a CBT in the library than what exists in the Controller.

5.21.2 Considerations for update to project command

- If object is not in QVCS, update to project is allowed.
- If object is in QVCS, it must be checked out since update to project is equivalent to an edit.
- Migration rules are used when updating to project, which is unchanged from current processing and occurs regardless of QVCS.

5.21.3 Example CBT Update to Project scenarios

The following table outlines various update to project scenarios based on the “Graphical Example of Layered Recipe Check In and Check Out Scenario” on page 81 provided in another section of this document. Since update to project is only supported for instances, load for CBT type only or CBT plus Instance does not apply (N/A). No scenario table is required for upload, since it can be performed to the Monitoring object regardless of the QVCS state.

Table 12: Layered Recipe Scenarios for CBT Load

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform Update to Project)	CBT Instance Only (Perform Update to Project)	CBT + Instance (Perform Update to Project)
1	2.0 (Checked In)	3.0 (Checked In)	N/A	Cannot do an Update to Project since instance is checked in	N/A
2	2.0 (Checked In)	3.0 (Checked Out)	N/A	Allowed	N/A
3	2.0 (Checked Out)	3.0 (Checked In)	N/A	Cannot exist since an Instance cannot be checked in if CBT is checked out	N/A
4	2.0 (Checked Out)	3.0 (Checked Out)	N/A	Allowed	N/A
5	2.0 (Checked In)	Instance 3.0 is loaded to Controller and is Checked In.	N/A	Cannot do an Update to Project since instance is checked in	N/A
6	2.0 (Checked Out)	Instance 3.0 is loaded to Controller and is checked In	N/A	Cannot exist since an Instance cannot be checked in if CBT is checked out	N/A

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform Update to Project)	CBT Instance Only (Perform Update to Project)	CBT + Instance (Perform Update to Project)
7	2.0 (Checked In)	Instance 3.0 Checked Out. (After it is loaded to the controller)	N/A	Allowed	N/A
8	2.0 (Checked Out)	3.0 Checked Out (After it is loaded to controller)	N/A	Allowed	N/A
9	CBT is Deleted from Library	Deleted from Project	N/A	N/A	N/A
10	2.0 (Checked In)	Deleted from Project	N/A	N/A	N/A
11	2.0 (Checked Out)	Deleted from Project	N/A	N/A	N/A

5.22 Importing Objects

The QVCS must enforce validation of the import since the function results in changes to block configurations. If the object being imported already exists in the ERDB and is under control of the QVCS, the QVCS performs an *Automatic Check In* of the object. This ensures that the object configuration that is contained in the import file is recorded in QVCS without being altered. The following section summarizes the rules that are used when importing an object.

5.22.1 Import Rules for QVCS

The following table outlines the rules for importing objects based on the QVCS status or existence of the object being imported.

Import version number status	Existing object checked in	Existing object checked out	Existing object not in QVCS	Object does not exist
<i>Version no. less than or greater than tree object version no.</i>	Import OK, replaces existing. Automatically checked in and version number adjusted as needed.	Import not allowed.	Import OK, replaces existing. Version number parameter is set from import file	N/A
<i>Version no. equal to tree object version no.</i>				
<i>No object in tree</i>	N/A. If object is checked in, it must also be in tree	N/A. If object is checked in, it must also be in tree	N/A	Import OK. Version number is set from import

As shown in the table, you can only import an object if the object in the destination database is:

- Checked in, or
- Is not under version control, or
- Does not exist at all.

In addition, the resultant imported object will be automatically checked in if the existing object is under control of the QVCS. If the object does not exist in the QVCS, the version number that is assigned is based on the version number in the import file. For an object that is automatically checked in, the rules for determining the exact version number that will be assigned is outlined in the next section Version number rules.

5.22.2 Version number rules

The following table outlines the rules for determining the exact version number that will be assigned for an object.

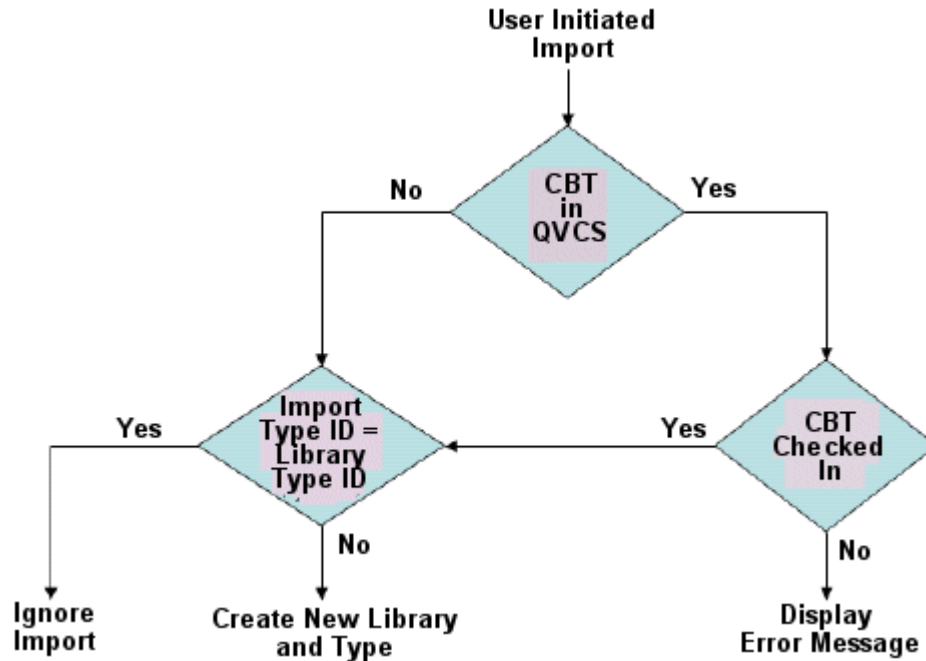
Import Version Number	Resultant Version Number on check in
<i>Version number less than highest version number</i>	Version number is set to major increment of highest version number
<i>Version number equal to highest version number</i>	Version number is set to minor increment of highest version number
<i>Version number greater than tree object version number</i>	Version number is set to the imported version number
<i>Object not under version control or does not exist</i>	

5.22.3 General import considerations

- When the object being imported is a strategy, or a user defined template, that has a derivation parent of a user defined template, if the derivation parent does not exist in the **Library** tab, then the import will fail, and a message will be displayed. Note that when doing multiple imports as a group, the derivation parent may be one of the imported objects.
- The Substitute Name List entries that are included with the import file, as part of the import process, will be processed to determine if any updates need to be applied to the current Substitute Name List. If there is a conflict in the definition of a Substitute Name List entry, then an error message is generated, and it is the responsibility of the user to take the appropriate corrective action.

5.22.4 CBT and UDT import considerations

- It is not possible to import a UDT and its instances simultaneously. The reason is, for a UDT to be imported successfully, its instance has to be in checked out state. Import of an instance fails if it is checked out. The workaround is to import the UDTs and its instances separately.
- When a CBT is not under control of QVCS, an import of a CBT creates a new library for the CBT, if it is different from an existing CBT; and creates a new CBT in that library with the same name as the import name. If no difference is detected, the import of CBT is ignored. The determination of a difference is based on the “block type ID” for the CBT which is updated any time a change has been made to the CBT, regardless if it is a “major” or “minor” change. The new library will have the standard suffix appended. For example, if the library name was MY_LIB, then the new library will be named MY_LIB_1. The new library is created whether there are any instances of the CBT or not. If there are already any instances of the CBT (being imported) existing in the ERDB, the user must perform a manual “Change Parent” to the new CBT. If the user is importing a CBT and Control Modules that contain instances of the CBT at the same time, those instances will be updated to reflect the change in the library name automatically.
- When the CBT is under control of QVCS, it must be checked in to perform an import just like all other objects. If CBT is checked out, the import is not allowed and an error message is generated. If the CBT is checked in, the import is allowed to proceed. Unlike other objects, however, *Automatic Check In* is **not** supported for CBT in the R400 release. This is because it is not possible for QVCS to determine if the CBT being imported has a *major* or *minor* change, and to insure that version consistency rules are not violated, the new library and CBT will be created. This is true if the Block Type ID's (the Block Type ID of the CBT in ERDB, and the CBT being imported) do not match a new Type is created. Although the import creates a new type, the requirement for the CBT to have a status of checked in will be enforced to maintain consistency with import requirements for QVCS. For example, if an object is in QVCS, then it must be checked in for an import to be allowed. The following illustration summarizes the CBT import behavior



- If CBT is under control of QVCS, instance must be checked in.
 - If version of CBT used by instance is in the Library, the import will succeed. An automatic check in occurs, making the imported version the new highest version.
 - If version of CBT used by instance is not in the Library, the import will fail.
 - If the instance and CBT are imported at the same time, and the instance is based on the CBT being imported, and the CBT version is different than the CBT in the library; the import of the instance will automatically check in the instance since the instance is referring to the original CBT, which is checked in.
- If CBT is not under control of QVCS, the following rules apply.
 - If the CBT upon which the instance is based exists in the library (based on matching of the block type ID), then the import for the instance occurs normally.
 - If the CBT exists in QVCS (but has been deleted from the library), the import fails unless the CBT is also selected for import. If subsequently a revert of the deleted CBT is attempted, it will fail due to duplicate name.

5.22.5 Example CBT import scenarios

The following table outlines various CBT import scenarios based on the “Graphical Example of Layered Recipe Check In and Check Out Scenario” on page 81 provided in another section of this document. Since update to project is only supported for instances, load for CBT type only or CBT plus Instance does not apply (N/A). No scenario table is required for upload, since it can be performed to the Monitoring object regardless of the QVCS state.

Table 13: Layered Recipe Scenarios for CBT Import

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform Import)	CBT Instance Only (Perform Import)	CBT + Instance (Perform Import)
1	2.0 (Checked In)	3.0 (Checked In)	Version 2.0 - Ignored. If not Version. 2.0, new CBT will be created.	Version 3.0 is allowed. Instance will be automatically checked in Other versions are not allowed because the matching CBT does not exist in library.	Version 3.0 of Instance + version 2.0 of CBT are allowed. Other consistent sets are not allowed. (Reason is Auto Check In for CBT is N/A for R310.
2	2.0 (Checked In)	3.0 (Checked Out)	Version 2.0 - Ignored. If not Version 2.0 new CBT will be created.	Not allowed to perform import since instance is checked out. Error message will be displayed	Not allowed to perform import since instance is checked out. Error message will be displayed
3	2.0 (Checked Out)	3.0 (Checked In)	Cannot exist since an Instance cannot be checked in if CBT is checked out	Cannot exist since an Instance cannot be checked in if CBT is checked out	Cannot exist since an Instance cannot be checked in if CBT is checked out
4	2.0 (Checked Out)	3.0 (Checked Out)	Not allowed to perform import since CBT is checked out. Error message will be displayed.	Not allowed to perform import since instance is checked out. Error message will be displayed.	Not allowed to perform import since CBT and instance are checked out. Error message will be displayed.
5	2.0 (Checked In)	Instance 3.0 is loaded to Controller and is Checked In.	Version 2.0 - Ignored. If not Version. 2.0, new CBT will be created.	Version 3.0 is allowed. Instance will be automatically checked in Other versions are not allowed because the matching CBT does not exist in library.	Version 3.0 of Instance + version 2.0 of CBT are allowed. Other consistent sets are not allowed. (Reason is Auto Check In for CBT is N/A for R310.
6	2.0 (Checked Out)	Instance 3.0 is loaded to Controller and is checked In	Cannot exist since an Instance cannot be checked in if CBT is checked out	Cannot exist since an Instance cannot be checked in if CBT is checked out	Cannot exist since an Instance cannot be checked in if CBT is checked out
7	2.0 (Checked In)	Instance 3.0 Checked Out. (After it is loaded to the controller)	Version 2.0 - Ignored. If not Version. 2.0, new CBT will be created.	N/A (Any version)	N/A (Any version)

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform Import)	CBT Instance Only (Perform Import)	CBT + Instance (Perform Import)
8	2.0 (Checked Out)	3.0 Checked Out (After it is loaded to controller)	Not allowed to perform import since CBT is checked out. Error message will be displayed.	Not allowed to perform import since instance is checked out. Error message will be displayed.	Not allowed to perform import since CBT and instance are checked out. Error message will be displayed.
9	CBT is Deleted from Library	Deleted from Project	Any version of the CBT is allowed. It will create a new CBT. The new CBT will not have any QVCS history.	Not allowed (any version) as the corresponding CBT does not exist in the library. Error message will be displayed	As far as rule is concerned, any consistent sets can be imported. New CBT and instance created. Inconsistent sets are not allowed as the corresponding CBT does not exist in the library.
10	2.0 (Checked In)	Deleted from Project	Version 2.0 - Ignored. If not version 2.0, New CBT will be created.	Version 3.0 is allowed. Instance will be automatically checked in Other versions are not allowed because the matching CBT (matching Block Type ID) does not exist in library.	Version 3.0 of instance + version 2.0 of CBT are allowed. Other consistent sets are not allowed. (Reason is Auto Check In for CBT is N/A for R310)
11	2.0 (Checked Out)	Deleted from Project	Not allowed to perform import since CBT is checked out. Error message will be displayed	Not Allowed for any version. Version 3.0 is not allowed because as per import rules, on importing version 3.0 it will be automatically checked in. But this is not allowed as CBT is checked out. Other versions are not allowed because the matching CBT does not exist in library	Not allowed to perform import since CBT is checked out and instance violates the rules of auto check in of the instance. since CBT is checked out. Error message will be displayed

5.23 Exporting Objects

Since the export function is a *read only* operation, QVCS places no restriction on exporting a CBT or a CBT instance regardless of its QVCS state. You can export an object, if it is not in QVCS, if it is checked in, or if it is checked out. In addition, the qualification state of the object does not have any effect on the ability to export. These are the same rules for all objects in the system.

6 Relaxed Loading Support for QVCS

Related topics

“Relaxed Load Function” on page 130

6.1 Relaxed Load Function

If you are using a licensed Qualification and Version Control System (QVCS) application in your Experion system running software version R101 or higher, the relaxed load function lets you load an object that has not yet been checked in to QVCS to a controller.



Attention

Once an object is “checked in” to QVCS, it must pass the QVCS loading restrictions before it can be loaded to a controller.

6.1.1 Typical relaxed load functional scenario

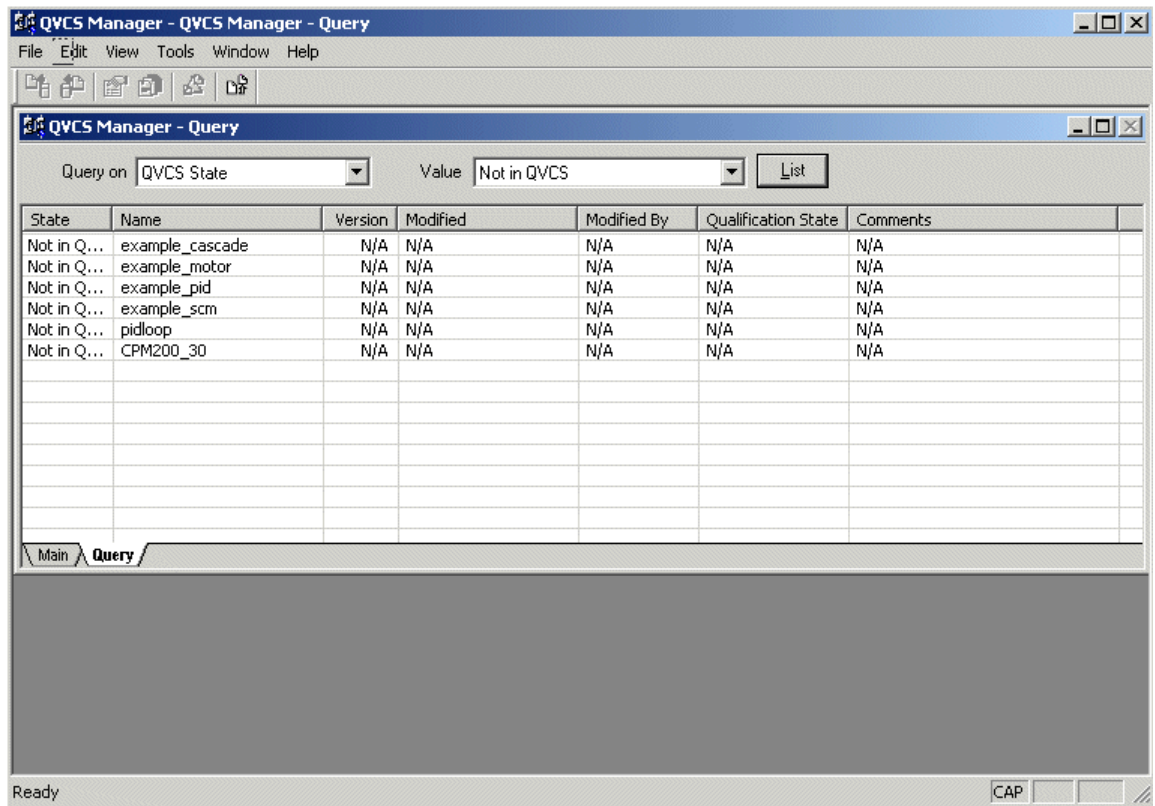
The following table outlines the stages in a typical development and testing scenario that uses the relaxed load function. This scenario assumes that you are familiar with the Control Builder application, have a QVCS license, and are running Experion software version R101 or higher.

Stage	Description
1	Create the controller/execution environment.
2	Create a control strategy
3	Assign the control strategy to the controller/execution environment.
4	Load the controller/execution environment and the control strategy to the controller.
	This is considered a relaxed load, since neither the controller/execution environment nor the control strategy is checked in to QVCS.
5	Test the control strategy to determine if it is executing correctly.
6	If control strategy executes correctly, go to Step 7. If control strategy does not execute correctly, make changes as required and repeat Step 5.
7	Check in the control strategy and controller/execution environment to QVCS.
8	Make any qualification state transitions as defined by the site's Standard Operating Procedure (SOP).
	The user is responsible for making sure that the process control system that is loaded has been qualified and that it contains no No-QVCS objects. The Query dialog of the QVCS Manager provides a Not in QVCS query and a Loaded, not in QVCS query, so users can quickly determine the status of their system.
9	Load the controller/execution environment and control strategy with the QVCS version in the desired qualification state to the controller.
10	If problems exist, use QVCS Manager to check out the problem object, make the necessary change in Control Builder, and check it into QVCS. With object in required qualification state, load it to the controller.

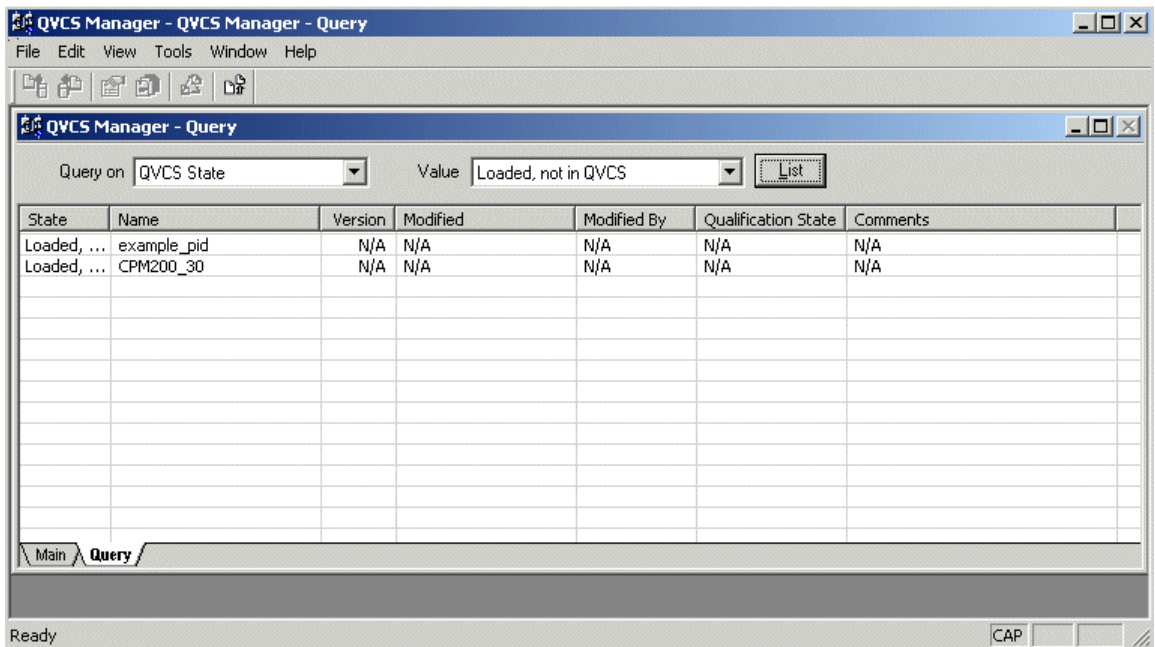
6.1.2 Initiating queries for No-QVCS objects

The following table outlines the typical steps for checking the state of the control system for any No-QVCS objects. The illustrations used are for example purposes only.

- 1 Click **Tools** -> **QVCS Manager** to launch the Manager.
- 2 Click the Query tab.
- 3 Click the down-arrow button to the right of **Query on** field and select **QVCS State** from the shortcut menu.
- 4 Click the down-arrow button to the right of **Value** field and select **Not in QVCS** from the shortcut menu.
- 5 Click the **List** button and view the results of the query in the dialog.



- 6 Click the down-arrow button to the right of **Value** field and select **Loaded, not in QVCS** from the shortcut menu.
- 7 Click the **List** button and view the results in the dialog.

**Tip**

You can use common Windows-based shortcut keystroke functions such as Shift + Click and Ctrl + Click to select all or only selected objects in the list to be checked in at one time.

- 8 This completes the steps for using QVCS queries to find No-QVCS objects in your control system.

7 QVCS Revert Operations

This section provides procedural references for doing Revert operations such as applying, removing, and maintaining labels as well as reverting to a version or label.

Related topics

- “About QVCS Revert function” on page 134
- “Maintaining QVCS Revert Labels” on page 135
- “Applying or Removing QVCS Revert Labels” on page 138
- “Reverting to a Version or a Label” on page 141
- “Example revert operation scenarios” on page 146

7.1 About QVCS Revert function

The revert function lets you retrieve any individual version of an object or a set of objects that have been grouped together with a user defined label. You may combine any objects that are under version control.

- You can only revert objects that are checked in at the time of the reversion. If any of the specified objects are checked out, none of the objects will be reverted. If a revert operation fails, the user is responsible for taking the appropriate corrective action such as checking in all objects and re-doing the revert operation.
- When an object that has been reverted, it becomes the *active* or *working* version of the object and it must be checked out to perform any edits.
- Since the Controller/CEE assignments and IOM associations are not part of the QVCS, the current assignments and associations are retained when older versions are reverted, and the qualification state is set to the qualification state of the version that is reverted.
- You can use the revert function to restore an object that is in QVCS after it has been deleted from the **Project** or **Library** tab in Control Builder. See the previous “Restoring Deleted Object” on page 97 section for more information.
- While QVCS performs some validation prior to the revert operation starting, it is not possible to check for all possible error conditions. If an error is encountered during the actual revert process, the revert operation is halted and the configuration that existed before the revert process began is restored.

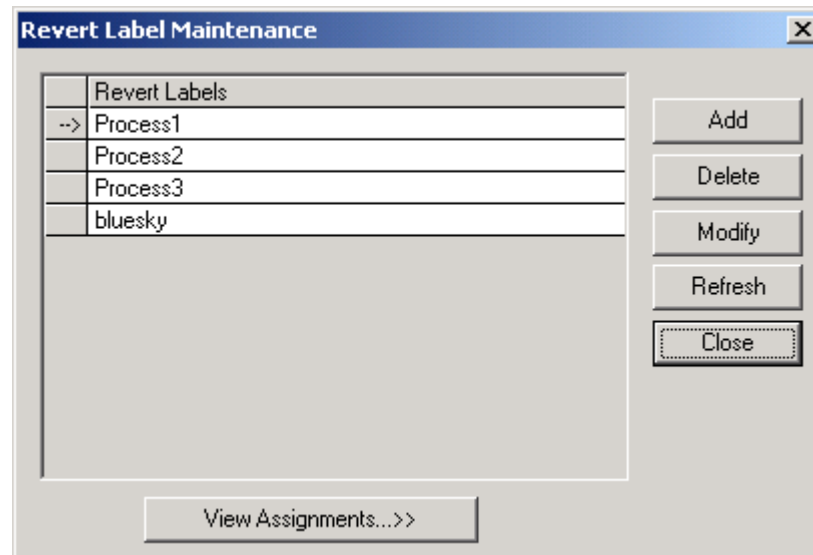
7.2 Maintaining QVCS Revert Labels

You can add, delete, modify, and view revert label assignments at any time.

- A revert label can be any combination of letters, numbers, punctuation, or special characters up to 98 characters in length without spaces. Labels are case insensitive so using upper or lower case makes no difference in terms of name uniqueness. A valid name must be unique and at least one character in length with no spaces.
- There are no restrictions on deleting a label that was used for a revert operation that still has active objects on view.
- You can call up the Revert Label Maintenance dialog through the context menu and dedicated buttons on the Apply/Remove Revert Label dialogs as well as the QVCS Manager Tools menu.

7.2.1 Calling up the Revert Label Maintenance dialog box

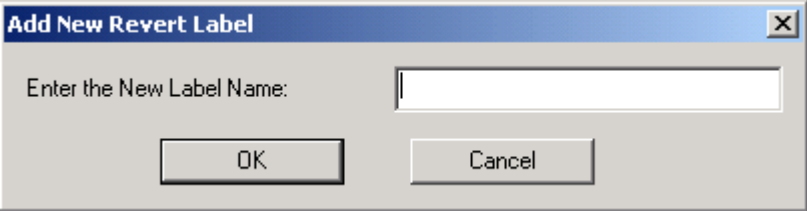
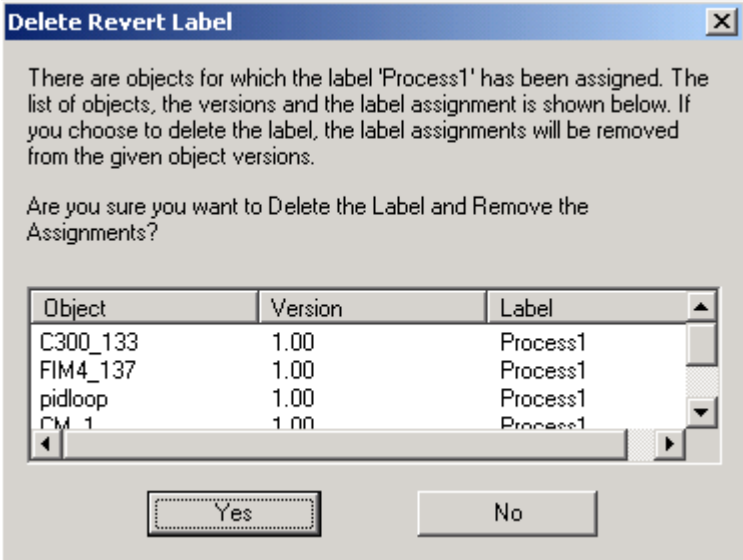
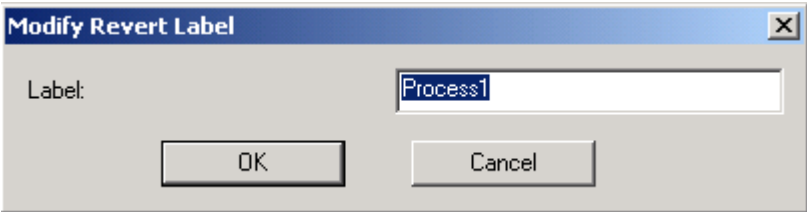
- 1 In the Control Builder **Tools** menu, click **QVCS Manager** to call up the application.
- 2 In QVCS Manager **Tools** menu, Click **Revert Labels > Label Maintenance** to call up the **Revert Label Maintenance** dialog.

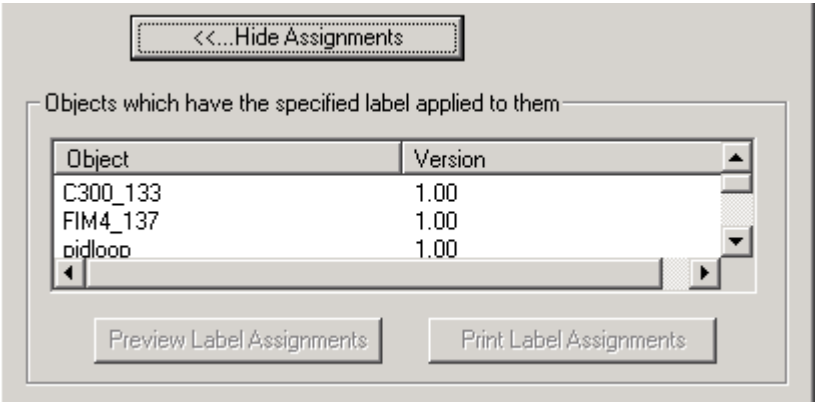


- 3 Go to the next section *To add, delete, modify or view assignments for revert labels* to continue; or the next Step to quit.
- 4 Click the **Close** button to close the dialog and exit the function.

7.2.2 Adding, deleting, modifying or viewing assignments for revert labels

The procedural actions in the following table assume that you have completed the previous procedure up to Step 3: The illustrations used in the following table are for example purposes only.

If you want to . . .	Then . . .
Add a revert label	<ul style="list-style-type: none"> Click the Add button to open the Add New Revert Label dialog.  <ul style="list-style-type: none"> Key in the desired name in the Enter the New Label Name box. Click the OK button to save the new label or the Cancel button to close the dialog without saving the new label. If you clicked OK, check that the new label is added to the Revert Labels list box.
Delete a revert label	<ul style="list-style-type: none"> Select the label you want to delete in the Revert Labels list box. Click the Delete button to open the Delete Revert Label dialog.  <ul style="list-style-type: none"> Click the Yes button to confirm the action or the No button to cancel the delete action. If you clicked Yes, wait for the action to complete and then check that the label is removed from the Revert Labels list box.
Modify a revert label	<ul style="list-style-type: none"> Select the label you want to modify in the Revert Labels list box. Click the Modify button to open the Modify Revert Label dialog.  <ul style="list-style-type: none"> Key in the desired modification to the revert label in the Label box. Click the OK button to save the changes or the Cancel button to close the dialog without saving the changes. If you clicked OK, check that the modified label in the Revert Labels list box is correct.

If you want to . . .	Then . . .
View assignments for given revert label	<ul style="list-style-type: none"> • Select the label whose assignments you want view in the Revert Labels dialog. • Click the View Assignments button to show the object assignment portion of the Revert Label Maintenance dialog.  <ul style="list-style-type: none"> • Use the horizontal and vertical scroll bars to scroll data in the Objects list box. • Click the Hide Assignments button to close the Objects portion of the dialog or just select another label in the Revert Labels list box.
Refresh the contents of the Revert Labels list box	<ul style="list-style-type: none"> • Click the Refresh button.

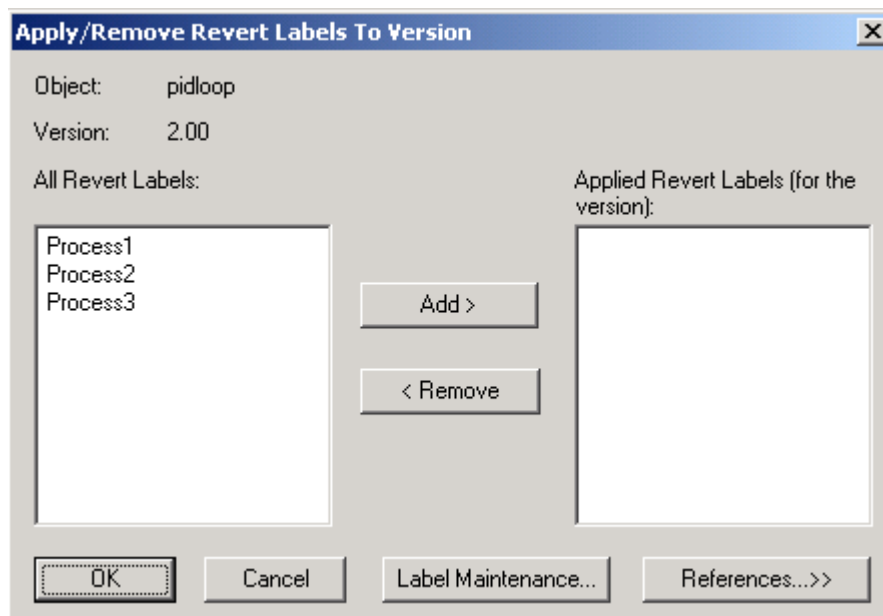
7.3 Applying or Removing QVCS Revert Labels

You can apply or remove revert labels for objects checked in to QVCS.

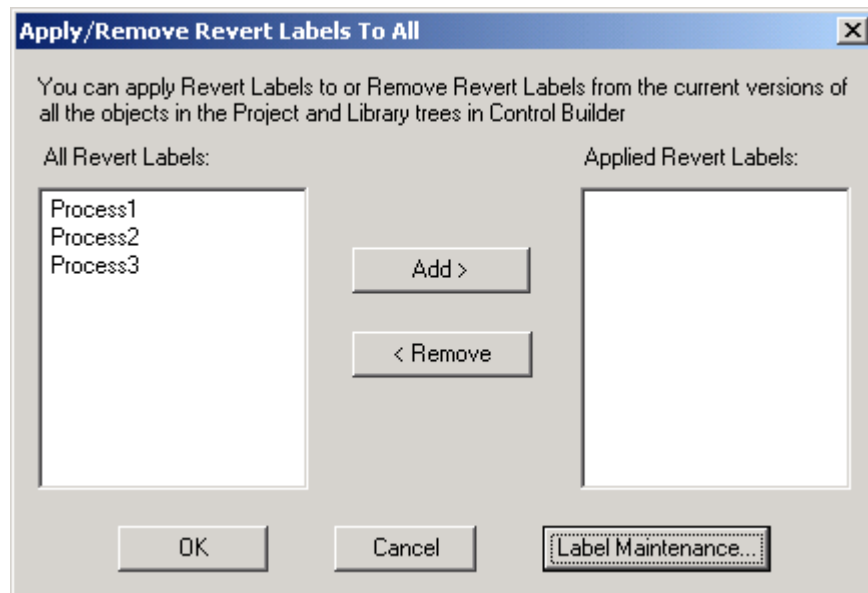
- You have created/added revert labels in QVCS.
- To apply a revert label, you need a revert label and the version of the object to which the label is to be applied.
- For control strategies, you cannot apply the same revert label to more than one version of the same object.
- For user defined templates (UDT), you can apply the same revert label to multiple versions of a template. This is allowed so that a group of strategies can have different derivation parents.
- You can also apply revert labels through the **Check In** dialog and **Properties** dialog.

7.3.1 Calling up the Apply/Remove type dialog box

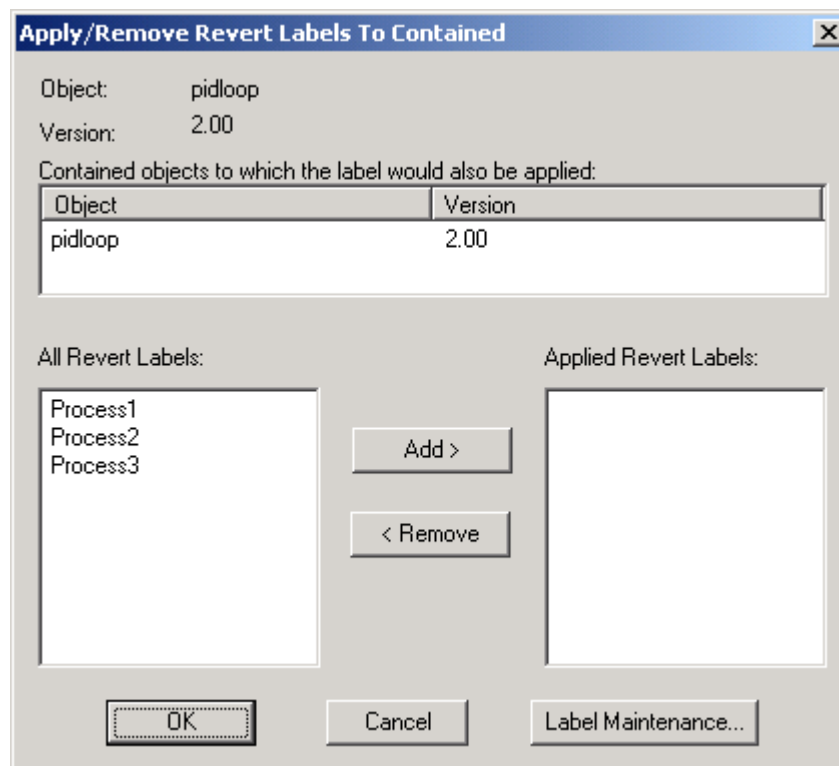
- 1 Select checked in objects in the **Project** or **Library** tab of Control Builder.
- 2 In the Control Builder **Tools** menu, click **QVCS Manager** to call up the QVCS Manager Main window. You can alternately right-click selected object and select **QVCS Manager** from the list or click the QVCS button in the toolbar to call up the QVCS Manager.
- 3 Select desired checked in object in the Main window.
- 4 In the QVCS Manager **Tools** menu, select one of the following depending on the Apply/Remove function you want to implement:
Revert Labels>Apply/Remove/To selected version to open the **Apply/Remove Revert Labels To Version** dialog and work with the selected object only,



Revert Labels>Apply/Remove/To all objects to open the **Apply/Remove Revert Labels To All** dialog and work with all applicable objects, or



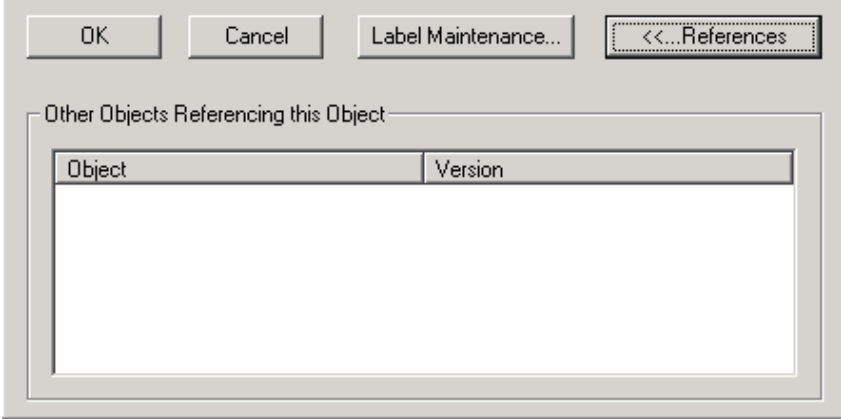
Revert Labels>Apply/Remove/To all contained objects to open **Apply/Remove Revert Labels To Contained** dialog and work with contained objects for selected object.



- 5 Go to the next section *To apply or remove revert labels* to continue; or the next Step to quit.
- 6 Click the **Cancel** button to close the dialog and exit the function.

7.3.2 Applying or removing revert labels

The procedural actions in the following table assume that you have completed the previous procedure up to Step 5: The illustrations used in the following table are for example purposes only.

If you want to . . .	Then . . .
Apply a revert label	<ul style="list-style-type: none"> • Select the desired revert label in the All Revert Labels list box. • Click the Add button to move the selected label to the Applied Revert Labels list box. • Click the OK button to apply the selected revert label or the Cancel button to close the dialog without applying the label.
Remove a revert label	<ul style="list-style-type: none"> • Select the label you want to remove in the Applied Revert Labels list box. • Click the Remove button to move the selected label to the All Revert Labels list box. • Click the OK button to remove the selected revert label or the Cancel button to close the dialog without removing the label.
Add a new revert label to the All Revert Labels list box	Click the Label Maintenance button call up the Revert Label Maintenance dialog. See the previous “Maintaining QVCS Revert Labels” on page 135 section for more information.
Check for other objects referencing the selected object version	<p>Click the References button to open the Object referencing portion of the Apply/Remove Revert Labels To Version dialog as show below.</p> 

7.4 Reverting to a Version or a Label

You can revert any individual version of an object or a set of objects that have been grouped together with a user defined label.

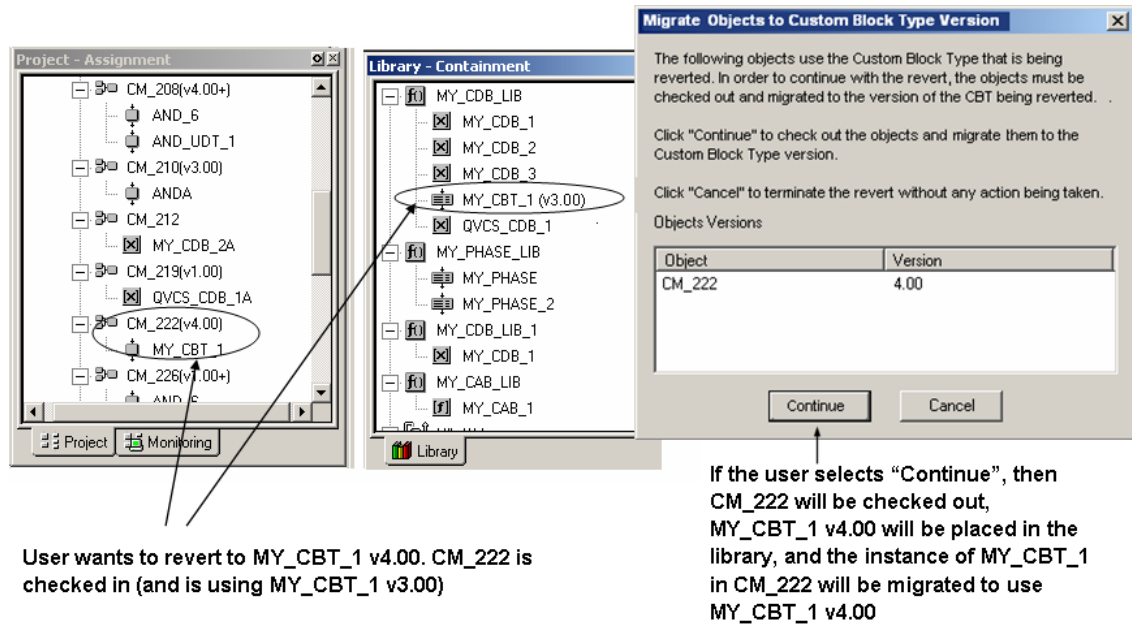
- See the previous “About QVCS Revert function” on page 134 for additional information.
- If you want to revert to an earlier version of an object, such as a CM instance, and that instance is dependent on a version of a UDT that is not currently in the Library tab, the revert operation will fail unless the correct version of the template is also included in the revert contents.
- If no instances of CBT exist, you can perform the revert operation to any version of CBT.
- If there are instances of CBT in the ERDB, observe the following rules for the revert operation.
 - It is only possible to revert to a higher version of the CBT.
 - Instances in the ERDB must be migrated to the CBT higher version. In this case, the instances must be checked out by the revert operation to continue, which is also true for UDTs. All instances must be checked in prior to the revert of the CBT, so they can be checked out by the revert operation and to insure that the current user has correct privilege for editing the instances.
- When reverting objects with instances of CBTs and UDTs, the criteria for both CBT and UDT must be met. If both criteria cannot be satisfied, then the revert cannot occur.
- If a *version consistent* set of CBTs and instances of CBTs are specified for revert as a group through revert to label action, then there is no restriction on going forward or backward with versions unless there are instances that are loaded to controllers.
 - When instances of CBTs are loaded to controllers, then version rules are enforced.
 - An error message instructs users to unload the loaded object(s) before re-initiating the revert operation.
- You must use the Revert by Label function for a CBT plus Instance revert operation. You can only select a single object when using the Revert by Version function.

7.4.1 Graphical example of CBT revert considerations

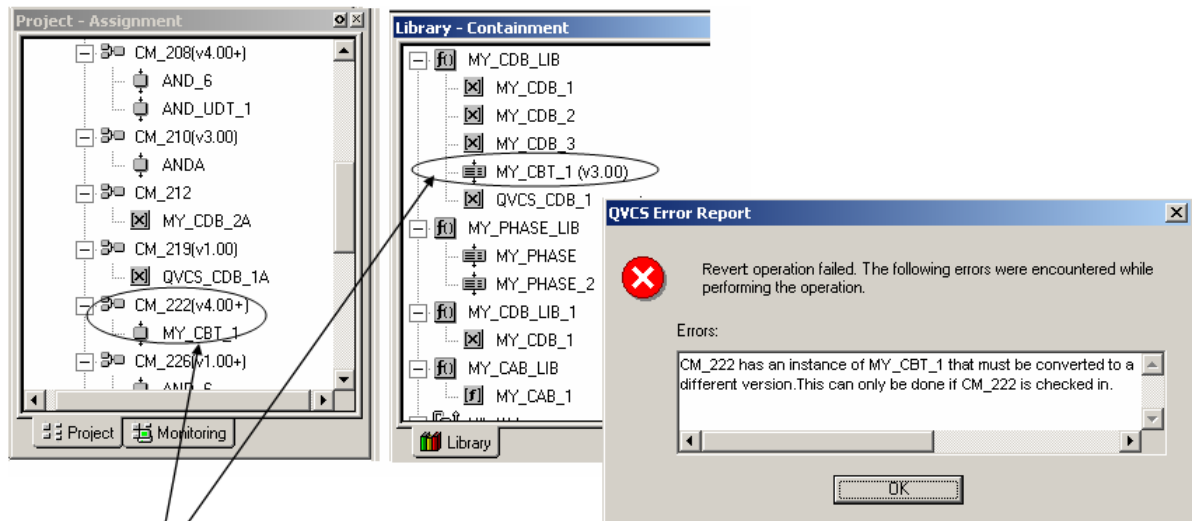
For this example, assume that there is a Control Module CM_222 that contains an instance of the MY_CBT_1 block with the version dependencies shown in the following table.

If CM_222 Version Is . . .	Then, It Uses MY_CBT_1 Version . . .
1.0	1.0
2.0 (check out, edit, and check in of the instance only)	1.0
3.0	2.0
4.0	3.0
5.0	4.0

As shown in the following illustration, the user wants to revert to MY_CBT_1 version 4.0 and the object containing the instance of the CBT is checked in prior to the revert request.



As shown in the following illustration, the user wants to revert to MY_CBT_1 version 2.00 and the object containing the instance of the CBT is checked out prior to the revert request. But, this is **not** allowed.

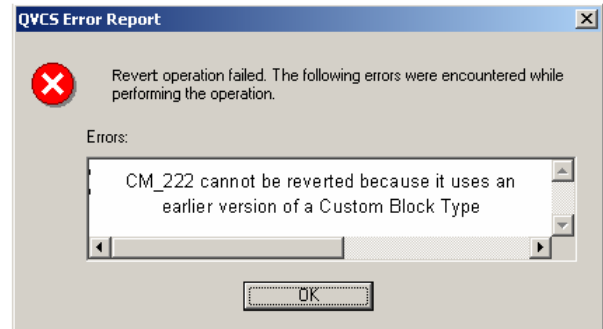


7.4.2 Graphical example of CBT instance revert considerations

- If user wants to revert to an instance of CBT version that exists in the Library tree, the standard revert occurs. As shown in the illustration below, a user can revert version 2.0 of CM_222 in the Library tree to version 1.0 without any extra actions.
- If CBT does not exist in the Library tree, revert cannot occur. User needs to perform revert of CBT to get it back into the Library tree.
- If CBT version does not exist in the Library tree, revert cannot occur if an earlier version of the CBT is required. As noted in the illustration below, if version 4.0 of CM_222 is in the tree and the user wants to revert to version 3.0, revert will not be allowed.

CM_222 Version	Uses MY_CBT_1 Version	Revert of CM_222 allowed?
1.0	1.0	No
2.0	1.0	No
3.0	2.0	No
4.0	3.0	Yes
5.0	4.0	Yes

Error message results if attempt to revert to any of these versions of CM_222



7.4.3 Graphical example of CBT version consistent set revert considerations

For the example shown below, assume that version 4.0 of CM_222 and version 3.0 of MY_CBT_1 are in the ERDB and that CM_222 is loaded to the controller. User selects both CM_222 and MY_CBT_1 from the same row for revert operation.

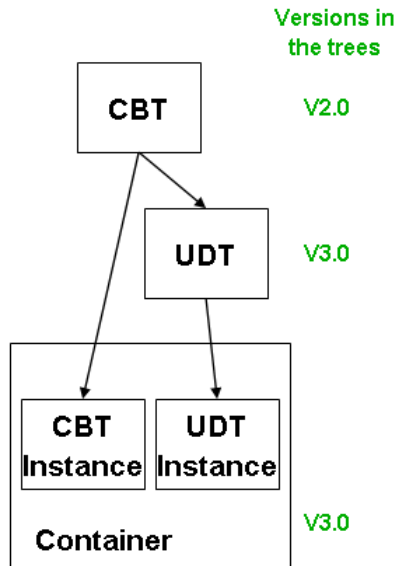
CM_222 Version	Uses MY_CBT_1 Version	Revert of CM_222 and MY_CBT_1 allowed?
1.0	1.0	No
2.0	1.0	No
3.0	2.0	No
4.0	3.0	Yes
5.0	4.0	Yes

Even though each row constitutes a version consistent set because loaded CM_222 uses version 2.0 of CBT, cannot restore earlier CBT version

Each row constitutes a version consistent set, so revert is allowed for any row because no version constraint is violated for loaded instance

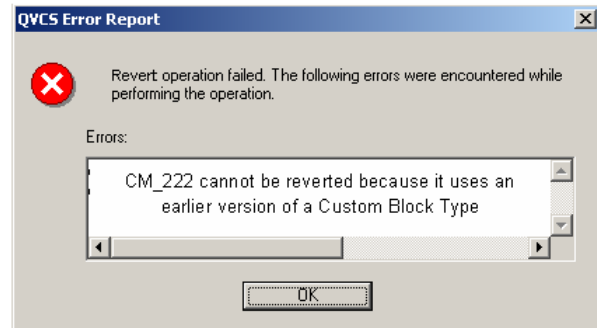
7.4.4 Graphical example of CBT with UDT revert considerations

For the example show below, with given version dependencies on the UDT example for check out (see “CBT and UDT check out considerations” on page 66 for reference) and the current versions active in the trees, user wants to revert to version 2.0 of the container (CM_222). Revert cannot occur.



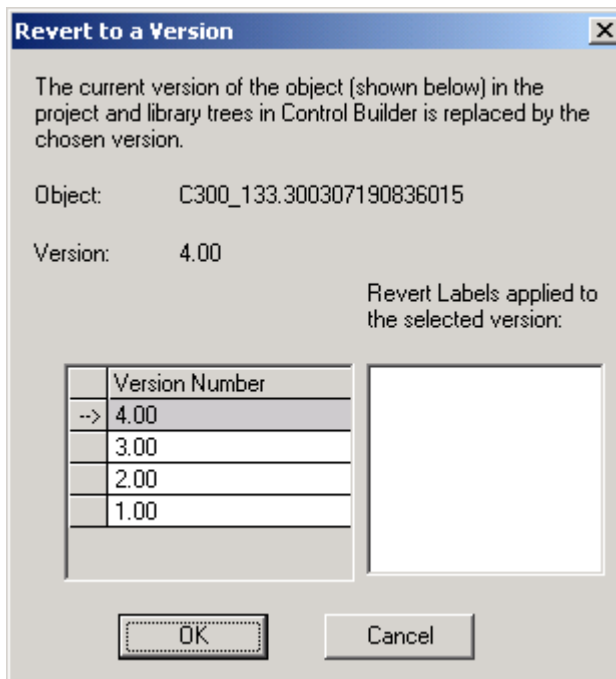
Dependencies of v2.0 of the container are v2.0 of the UDT and v1.0 of the CBT.

This revert request cannot be completed since the correct version of the CBT does not exist in the library. Thus, even though the UDT version discrepancy could be resolved, the CBT discrepancy cannot be resolved



7.4.5 Reverting to a version

- 1 Select the object in Control Builder tree that you want to revert to another version.
- 2 In the **Tools** menu, click **QVCS Manager** to call up the QVCS Manager Main window.
- 3 With the object you want to revert selected in either the Main or Query window, click **Tools > Revert Labels > Revert > To a version** to call up the **Revert to a Version** dialog.
Alternately, you can right-click the object and select **Revert Labels > Revert > To a version** to call up the dialog.

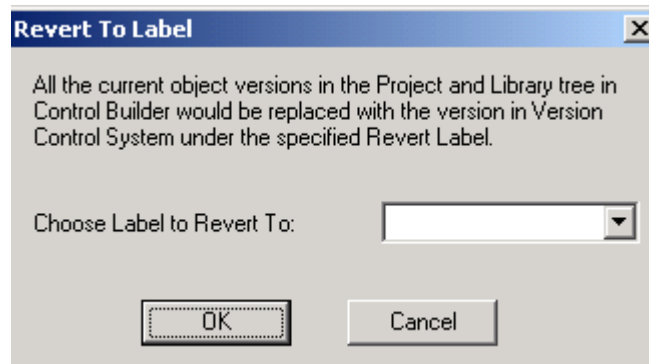


- 4 In the **Version Number** list box, click the version you want to revert to.

- 5 Click the **OK** button to initiate the revert operation. The actual reversion process does not start until you click the **OK** button. This may take a few minutes to complete.
 - Wait for the operation to complete.
 - Check that selected version of the object is restored to the **Project** or **Library** tab, as applicable.
- 6 This completes the procedure.

7.4.6 Reverting to a label

- 1 In the **Tools** menu, click **QVCS Manager** to call up the QVCS Manager Main or Query window.
- 2 With no object selected in either the Main or Query window, click **Tools > Revert Labels > Revert > To a label** to call up the **Revert to a Label** dialog.



- 3 Click the down arrow button in the **Choose Label to Revert To** box and select the desired label from the list.
- 4 Click the **OK** button to initiate the revert operation. The actual reversion process does not start until you click the **OK** button. This may take a few minutes to complete.
 - Wait for the **operation to complete**.
 - View the Properties form for selected objects to confirm current revert label assignment.
- 5 This completes the procedure.

7.5 Example revert operation scenarios

The following table outlines various revert operation scenarios based on the “Graphical Example of Layered Recipe Check In and Check Out Scenario” on page 81 provided in another section of this document. Refer to the Control Builder Error Codes Reference document for more information about listed Error Codes.

Table 14: Layered Recipe Scenarios for Check In

Scenario Number	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform Revert)	CBT Instance Only (Perform Revert)	CBT + Instance (Perform Revert by Label)
1	2.0 (Checked In)	3.0 (Checked In)	Can revert version 2.0 or higher. While reverting to a higher version, all the instances will get to be checked out by revert operation. Reverting to a lower version than 2.0 is not allowed. (See Error Message CBTVERCANNOTBEL ESS in following table.)	Only version 3.0 is allowed (which is already in the tree) since other versions require different versions of the CBT. Other versions not allowed. (See Error Code 14716)	Any versions consistent sets can be reverted using the Revert by Label option.
2	2.0 (Checked In)	3.0 (Checked Out)	Not Allowed. (See Error Message CBTINSTANCECHECKIN in following table.)	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14626)
3	2.0 (Checked Out)	3.0 (Checked In)	Not Possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out
4	2.0 (Checked Out)	3.0 (Checked Out)	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14626)
5	2.0 (Checked In)	Instance 3.0 is loaded to Controller and is Checked In.	Can revert to version 2.0 or higher. While reverting to a higher version, all the instances have to be checked out. Reverting to a lower version than 2.0 is not allowed. (See Error Message CBTVERCANNOTBEL ESS in following table.)	Only version 3.0 is allowed (which is already in the tree) since other versions require different versions of the CBT. Other versions not allowed. (See Error Code 14716)	To support the loaded instance, the versions of the CBT should be version 2.0 or higher. Any version consistent set of the instance and CBT where the CBT is at least version 2.0 can be reverted. An attempt to revert an earlier version of the CBT is not allowed. (See Error Message CBTVERCANNOTBELESS in following table.)
6	2.0 (Checked Out)	Instance 3.0 is loaded to Controller and is checked In	Not possible since instance cannot be checked in if CBT is checked out	Not Possible since instance cannot be checked in if CBT is checked out	Not possible since instance cannot be checked in if CBT is checked out

ScenarioNumber	CBT (In Library Tree)	CBT Instance (In Tree)	CBT Only (Perform Revert)	CBT Instance Only (Perform Revert)	CBT + Instance(Perform Revert by Label)
7	2.0 (Checked In)	Instance 3.0 Checked Out. (After it is loaded to the controller)	Not Allowed. (See Error Message CBTINSTANCECHECKIN in following table.)	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Message CBTINSTANCECHECKIN in following table.)
8	2.0 (Checked Out)	3.0 Checked Out (After it is loaded to controller)	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14626)
9	CBT is Deleted from Library	Deleted from Project	Can perform revert of any version of CBT.	Cannot perform revert. (See Error Code 14716. Also, see Error Code 14660)	Version consistent sets can be reverted together using the Revert by Label option.
10.	2.0 (Checked In)	Deleted from Project	Can perform revert of any version of CBT.	Only version 3.0 is allowed since other versions require different versions of the CBT. Other versions not allowed. (See Error Code 14716)	Version consistent sets can be reverted together using the Revert by Label option.
11.	2.0 (Checked Out)	Deleted from Project	Not Allowed. (See Error Code 14626)	Not Allowed. (See Error Code 14710)	Not Allowed. (See Error Code 14626)

7.5.1 Error Message related to revert operations

The following table describes some error messages that can get generated during revert operations and are not associated with any error code.

Error Message	Description	Cause	Workaround
VCS_CBTVERCANNOTBELESS	Cannot revert Version <number> <LibraryName:CBTName> because <Object Name> requires at least version <number>.	Revert of non-current version of CBT less than current version when any instances exist.	Either convert the instances of CBT to another type or delete the instances.
VCS_CBTINSTANCECHECKIN	The <Object Name> uses Custom Block Type <LibraryName: CBT Name> that must be migrated to a higher version. Revert must perform the check out	Object containing instance of CBT is checked out prior to revert request	Check In the CBT instance before trying to revert the CBT because the revert must be able to check out the instance (same as for UDT)

8 QVCS Administration Operations

This section provides procedural references for doing administration operations such as configuring qualification states and configuring and changing Electronic signature requirements.

Related topics

- “About QVCS Qualification State” on page 150
- “Configuring Qualification States” on page 151
- “Qualification States and Import Considerations” on page 158
- “Qualification State and Template Defining Parameters Consideration” on page 159
- “QVCS Electronic Signature Purpose” on page 160
- “Configuring QVCS Electronic Signature” on page 161
- “Viewing QVCS Logs” on page 162

8.1 About QVCS Qualification State

In the QVCS system, the qualification state refers to the stages a control strategy may pass through as part of the customer's qualification process. For example, a customer may define *Implemented*, *Testing*, *Released*, and *Disabled* as states that are important for tracking the control strategy qualification process. The control strategy would go through these different phases as it is developed and refined. It may be necessary for the control strategy to go through certain phases for it to be termed *qualified*. For example, it may be necessary that for the control strategy to be *Implemented*, that *Testing* be performed on it and that it be *Released*.

8.2 Configuring Qualification States

The Qualification State configuration dialog lets you define the configuration states and to manage the qualification life cycle transition. You need an Administrative privilege to access this dialog box. You can use the dialog to add, modify and delete the configuration states as well as specify qualification life cycle transition states and define electronic signature requirements for the state transitions.

- You must have administrative access privileges to use administrative functions.
- There must be at least one qualification state defined and you can define up to 12 states.
- There is no implied hierarchy of states, but states will be listed in the order in which they are defined.
- When you first check an object into QVCS, its initial state is set to the fallback state of the first state defined in the Qualification State Configuration dialog.
- You can only change the state of a version after it has been checked in, since you cannot change its state at the time of a check in.
- You should carefully consider the states that are desired and you should configure them before placing any objects under version control. While it is possible to add, delete, and modify states after objects have been placed in the QVCS, you cannot delete a state if it is used by any version.
- Renamed states will be automatically propagated to all versions that reference that state.
- Changes to other qualification state criteria such as fallback state, allowable transitions, and so on, will take effect the next time the criteria are used.
 - For example, An object in the Project tab has been loaded to the controller, which means it is in a state that is configured with the Load to Control Allowed option set to true or yes. The Load to Controller Allowed option is then changed to false or no. The change is logged, but no other action occurs. At some later time, the user requests the object to be loaded. The load does not occur because the state is no longer configured as yes/okay to load.
- An object is placed in the Fallback state, when it is checked in to the VCS. The fallback state may be different for each of the qualification states.
- The restricted signing state is used only if signing is required to make a state transition. If the restricted state is not defined as **None**, then, when a state change to the actual state is made, the signature to authorize the change must be different from the signature that authorized the most recent change to the restricted state.

8.2.1 Qualification life cycle configuration tips

- Configure users before deploying and using the QVCS system.
- Define and configure the qualification lifecycle before developing objects, avoid making changes to the lifecycle as much as possible (define behavior on deleting adding, and so on of qualification states).
- If the QVCS system is used in a non-regulated industry and the main goal is to have version control of objects then define only one qualification state, for example the 'default' state and allow this state to be loaded to the controller. This will basically eliminate the qualification lifecycle.
- A double signature for a certain qualification state transition as part of the lifecycle definition can be accomplished by defining two subsequent qualification states. Each state change to be performed by a different user/group with electronic signature enabled. The restricted signing state of the second qualification state (which corresponds with the second signature) should be set to the qualification state of the first qualification state signature (which corresponds with the first signature).

8.2.2 Calling up the Qualification State Configuration dialog box

- 1 In the Control Builder **Tools** menu, click **QVCS Manager** to call up the application.
- 2 In QVCS Manager **Tools** menu, Click **Tools > Administration > Configure Qualification States** to call up the **Qualification State Configuration** dialog.

Qualification State Configuration

Qualification Life Cycle States

	State	Fallback State	Restricted Signing	Load To Controller Allowed
-->	Default state	Default state	Default state	Yes

Add

Delete

Modify

Qualification Life Cycle Transitions

-- To States -->

	Default state
From States -->	Default state
	Not Configurable

Configure State Transition Requirements

Close

- 3


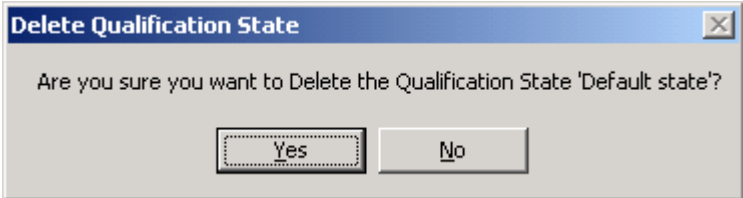
Go to the next section *To add, delete, or modify qualification states* to continue; or the next Step to configure state transitions.
- 4


Go to the following section *To define state transition requirements* to continue or the next Step to quit.
- 5

Click the **Close** button to close the dialog and exit the function.

8.2.3 Adding, deleting, or modifying qualification states

The procedural actions in the following table assume that you have completed the previous procedure up to Step 3: The illustrations used in the following table are for example purposes only.

If you want to . . .	Then . . .
Add a qualification state	<ul style="list-style-type: none"> Click the Add button to open the Add New Qualification State dialog.  <ul style="list-style-type: none"> Key in the desired name for the new state in the Qualification State box. Click the down-arrow button in the Fallback State box and select the desired existing qualification state from the list. Click the down-arrow button in the Restricted Signing box and select the desired existing qualification state or None from the list. Check the Load To Controller allowed check box, if it is okay (Yes) to load an object in that state to a controller. If it is not okay (No), leave the check box blank. Click the OK button and check that the new state is added to the Qualification Life Cycle States grid box.
Delete a qualification state	<ul style="list-style-type: none"> Select the state you want to delete in the Qualification Life Cycle States grid box. Click the Delete button to open the Delete Qualification State dialog.  <ul style="list-style-type: none"> Click the Yes button to confirm the action or the No button to cancel the delete action. If you clicked Yes, wait for the action to complete and then check that the state is removed from the Qualification Life Cycle States grid box.

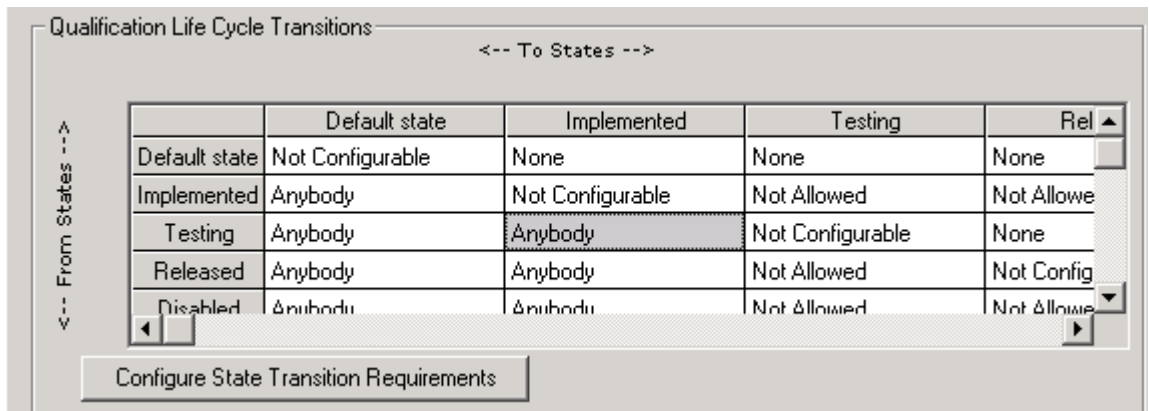
If you want to . . .	Then . . .
Modify a qualification state	<ul style="list-style-type: none"> Select the state you want to modify in the Qualification Life Cycle States grid box.. Click the Modify button to open the Modify Qualification State dialog.  <ul style="list-style-type: none"> Key in the desired modification to the name of the state in the Qualification State box. And/Or, Click the down-arrow button in the Fallback State box and select another existing qualification state from the list. And/Or, Click the down-arrow button in the Restricted Signing box and select another existing qualification state or None from the list. And/Or, Check the Load To Controller allowed check box, if it is okay (Yes) to load an object in that state to a controller. If it is not okay (No), uncheck the check box. Click the OK button and check that the modification(s) for the state appear in the Qualification Life Cycle States grid box.

8.2.4 Defining state transition requirements

The procedural actions in the following table assume that you have completed the previous “Calling up the Qualification State Configuration dialog box” on page 151 procedure up to Step 4: The illustrations used in the following table are for example purposes only.

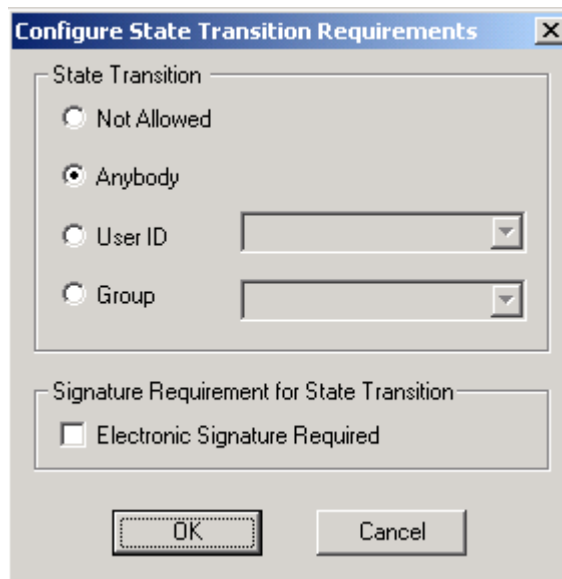
To define state transition requirements

- 1 Select the cell in the **Qualification Life Cycle Transitions** grid box that you want to configure.



	Default state	Implemented	Testing	Rel
Default state	Not Configurable	None	None	None
Implemented	Anybody	Not Configurable	Not Allowed	Not Allowed
Testing	Anybody	Anybody	Not Configurable	None
Released	Anybody	Anybody	Not Allowed	Not Configurable
Disabled	Anybody	Anybody	Not Allowed	Not Allowed

- 2 Click the **Configure State Transition Requirements** button to call up the **Configure State Transition Requirements** dialog.



- 3 Click the desired selection in the State Transition box.

If you select . . .

Then, requirement is . . .

Not Allowed

Transition between states is not allowed.

Anybody

Any user can change the transition state.

User ID

Only the user selected from the drop-down list in the box can change the transition state.

Group

Only a user belonging to the group selected from the drop-down list in the box can change the transition state.

- 4 Check the **Electronic Signature Required** check box if you want to prompt the user for Electronic Signature before the qualification state transition occurs. This check box is not available when the State Transition selection is **Not Allowed**.
- 5 Click the **OK** button to save the changes.
- 6 Confirm that selected cell data has changed accordingly.
- 7 Repeat Steps 1 to 6 for other cells in the grid, as required.
- 8 This completes the procedure.

8.2.5 Graphic Example of Qualification States Configuration and State Chart

The following state chart provides a basic description of the states and transitions based on the qualification state configuration shown in the following figure. You can only make state changes to objects under version control and checked in. Any subsequent check out and check in operation will cause an object's qualification state to revert to the fallback state defined for its state when the checkout was performed.

Qualification State Configuration

Qualification Life Cycle States

	State	Fallback State	Restricted Signing	Load To Controller Allowed
--->	Default state	Implemented	None	No
	Implemented	Implemented	None	No
	Testing	Implemented	None	Yes
	Released	Implemented	Testing	Yes
	Disabled	Disabled	None	No

Add

Delete

Modify

Qualification Life Cycle Transitions

<-- To States -->

	Implemented	Testing	Released	Dis
Implemented	Not Configurable	Group1	Not Allowed	None
Testing	Anybody	Not Configurable	Group2	Anybody
Released	Anybody	Not Allowed	Not Configurable	User1
Disabled	Anybody	Not Allowed	Not Allowed	Not Config

Configure State Transition Requirements

Close

Figure 24: Sample Qualification State Configuration Referenced in Following State Chart Diagram

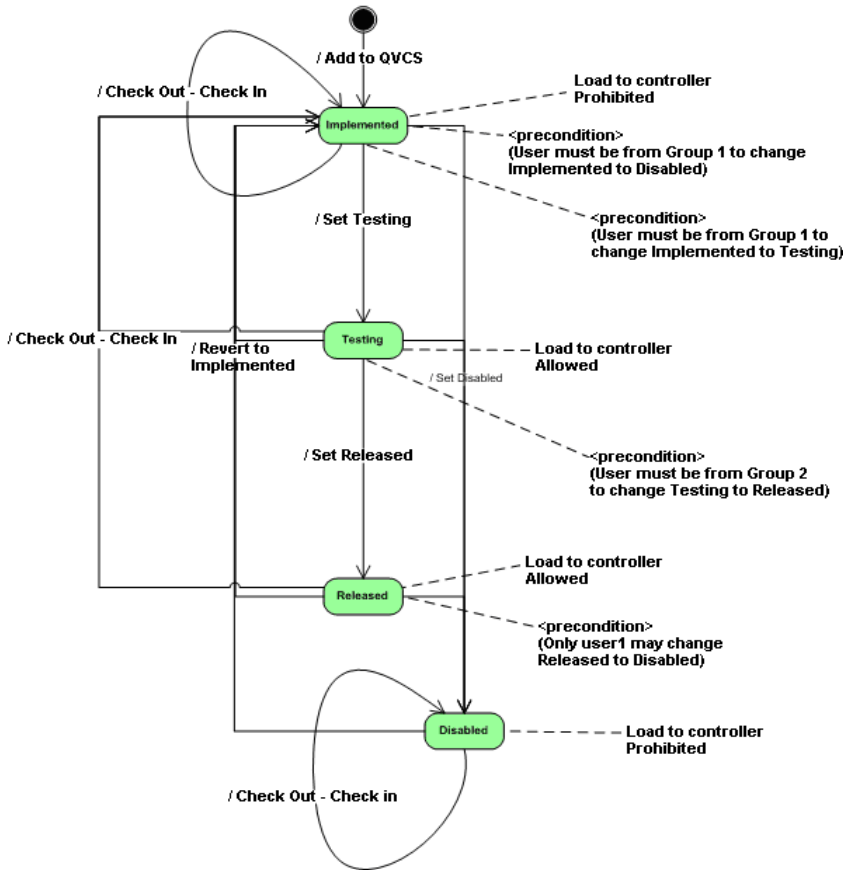


Figure 25: Example Qualification States State Chart

8.3 Qualification States and Import Considerations

The qualification state resulting from an import depends on whether the object is a strategy or a user defined template. For a strategy, the qualification state is set to the fallback state of the state that is specified in the import file. For a user defined template, the qualification state remains unchanged. See the “Example CBT load scenarios” on page 118 Importing Objects section for more information.

When importing control strategies to other Experion installations, you must make provisions to handle the situation where the qualification state referenced in the import file does not exist in the destination database. The matching is based strictly on name, so *Released* matches with *released* but not with *release*. When a match does not exist, the qualification state is set to the first state defined in the destination database.

8.4 Qualification State and Template Defining Parameters Consideration

The QUALSTATE parameter that contains the qualification state for the current version of an object cannot be defined as a template defining parameter.

8.5 QVCS Electronic Signature Purpose

Depending upon the qualification state transition configuration requirement, the QVCS may require verification of the action through an electronic signature. The QVCS supports a single signature type that requires a user identification (ID) and password. The user ID or group of the user that must sign is included on the electronic signature dialog.

8.6 Configuring QVCS Electronic Signature

The Configure Electronic Signature dialog lets you define the time out period allowed for the Electronic Signature validation as well as specify the Electronic Signature requirement for Revert operations.

You must have administrative access privileges to use administrative functions

The user ID defaults to the current user, and can be changed to a different ID.

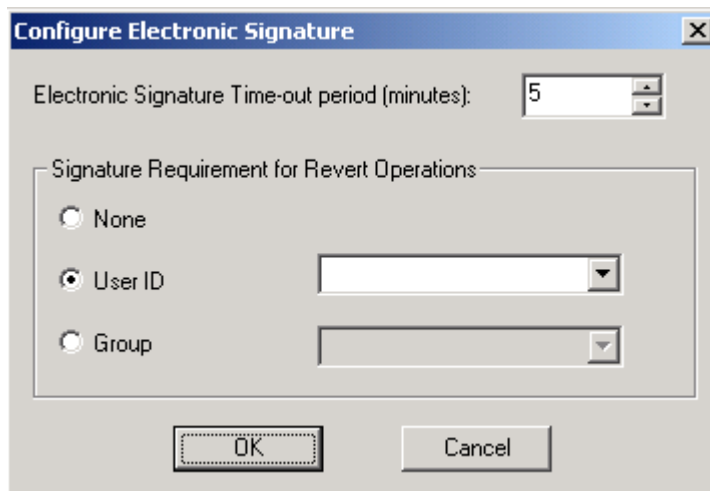
Entry of the password is always required, and both the ID and password are masked.

The time out period determines how many minutes are allowed before recording the attempt as a failure. This value may be changed at any time and will take effect the next time an electronic signature is required. A zero (0) value results in no time out.

The revert action can be defined as requiring authorization before it proceeds. The user ID or group name of the required signer is entered, if an electronic signature is required.

To configure electronic signature

- 1 In the Control Builder **Tools** menu, click **QVCS Manager** to call up the application.
- 2 In QVCS Manager **Tools** menu, Click **Tools > Administration > Configure Electronic Signature** to call up the **Configure Electronic Signature** dialog.



- 3 Click the up or down-arrow button in the **Electronic Signature Time-out period (minutes)** box to increase or decrease the value. The default value is 5 minutes and the range is 0 to 1440 minutes.
- 4 Do you want to require an electronic signature for revert operations?
 - If the answer is NO, click **None** to select it.
 - If the answer is YES, select one of the following:
 - User ID:** Click down-arrow button in box and select desired user ID from the list.
 - Group:** Click down arrow button in box and select desired group from the list.
- 5 Click the **OK** button to save the entries and close the dialog.
- 6 This completes the procedure.

8.7 Viewing QVCS Logs

The QVCS logs provide a means for administrators to track attempted security violations.

You must have administrative access privileges to use administrative functions.

To view QVCS logs

- 1 In the Control Builder **Tools** menu, click **QVCS Manager** to call up the application.
- 2 In QVCS Manager **Tools** menu, Click **Tools > Administration > View QVCS Logs** to call up the **QVCS Admin Log** window.
- 3 Select the desired viewing criteria. See the “QVCS Manager QVCS Admin Logs Window” on page 43 section for more information.
- 4 Repeat Step 3 to view other logs as desired.
- 5 Close the window when finished.
- 6 This completes the procedure.

9 QVCS Migration and Interoperability

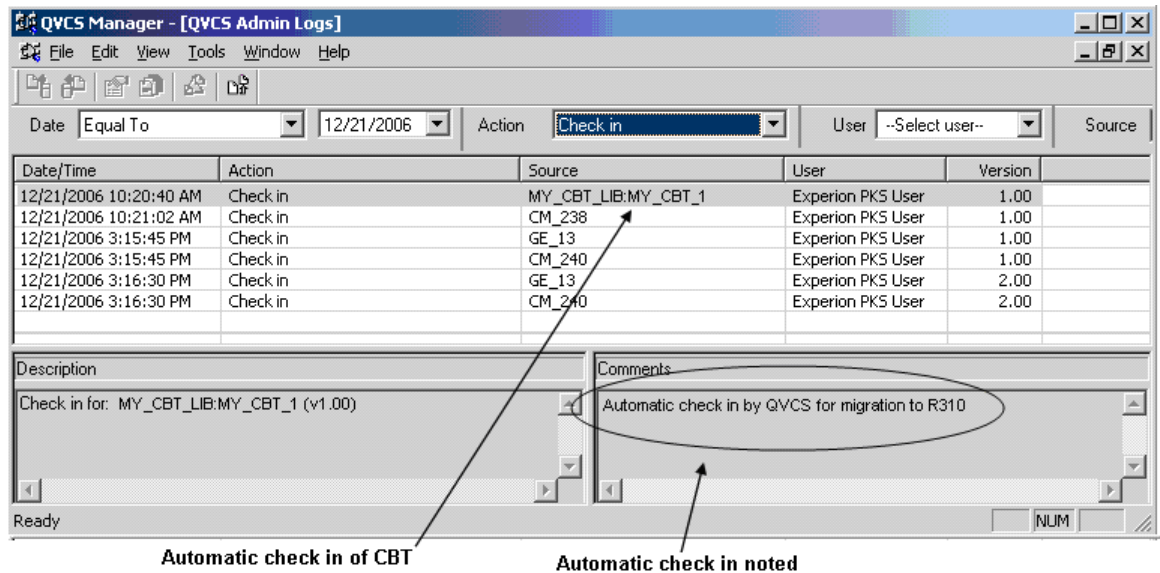
Related topics

“Migration Considerations” on page 164

“Interoperability Notes” on page 165

9.1 Migration Considerations

- The QVCS does not support the versioning of CBTs in releases prior to R400. The QVCS in prior releases does keep track of instances of CBTs that exist in objects that are placed in QVCS. This information will be used by QVCS during a migration from R310.x to R311.1 to automatically check in those CBTs that are used by QVCS objects. The CBTs that are automatically checked in during migration will be given a version number of 1.0, and a log entry will be generated with the comment that the check in was performed during migration. An example of how this information is shown in the QVCS log is shown in the following Figure. The migration of QVCS database from R301 to R311 or from R310 to R311, only happens during Experion database migration.



- Each CBT that is checked in during migration will take approximately 30 KB to store the definition files in the QVCS database. This means that if there were 100 different CBTs that were referenced by objects in QVCS, the additional disk space used will only be about 3 MB, well within server sizing requirements. The worst case would be where larger CBTs were defined, say 1400 parameters in each (which when the files are compressed for storage in QVCS means about 150KB for each version), and where the maximum number of controllers (20 plus up to 7 ACE) and the maximum number of types per controller are defined. This calculation is then:

$$150 \text{ KB} * [(200 \text{ Types/controller} * 20 \text{ controllers/ERDB}) + (400 \text{ types/ACE} * 7 \text{ ACE/ERDB})] = 1,020,000 \text{ KB} = 1 \text{ GB}$$

- Note that CBT's that are not referenced by objects in QVCS will not be checked in as part of migration to preserve the working environment. These CBT's may be checked in after the migration, subject to the standard rules imposed by QVCS for object check in.
- In addition to the automatic check in of referenced CBTs, QVCS will also add the "USER" library name to all UDTs that are in QVCS. This will not have any impact on using the UDT; it is simply to bring the information in the QVCS database up to the new requirements for the R400 release for displaying the library name with any UDT or CBT.
- Note that the INC.XML files that are part of the "base" definition of all types are not included in the definition files saved in QVCS. The INC.XML files may change with a new release, and since they are not themselves versioned, the CBTs will be automatically migrated to the new release when they are placed in the library through a QVCS check out or revert, or simply by existing in the library when the migration is performed.

9.2 Interoperability Notes

QVCS is directly invoked from Control Builder, and since release interoperability for CB is not required for R311, QVCS interoperability will not be supported for R311. This means that:

- You cannot connect to a pre-R311 QVCS database from an R311 QVCS Manager
- You cannot connect to a R311 QVCS database from a pre-R311 QVCS Manager

In terms of support for objects that were checked into QVCS in earlier releases, the mechanism that QVCS uses to restore earlier versions to the ERDB uses the migration rules that are defined for each release. Thus, an object version that was created in a pre-R311 release will automatically be migrated when it is checked out or reverted.

10 QVCS Maintenance

This section provides links to QVCS functions supported by the Database Administration Utility (DBADMIN) that can be used to help maintain the QVCS database.

Related topics

“Using DBADMIN for QVCS database maintenance” on page 168

10.1 Using DBADMIN for QVCS database maintenance

Use the following links to access information about using the DBADMIN utility to initiate the given QVCS database related operation.

- [Initiating QVCS database synchronization and clearing locks](#)
- [Making a Backup QVCS database](#)
- [Restoring a backup QVCS database](#)

11 QVCS Troubleshooting

Related topics

“Isolating Problems” on page 170

“Initial Checks” on page 171

“Getting Further Assistance” on page 173

11.1 Isolating Problems

This section offers some general data about initial checks that may help you isolate the cause of a problem.

11.2 Initial Checks

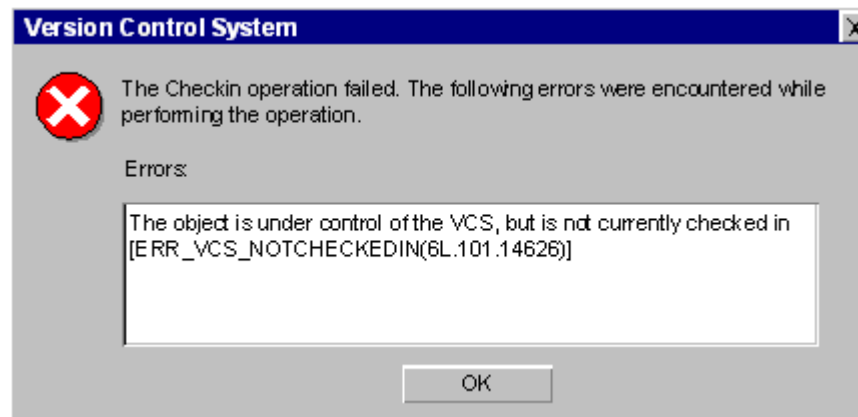
Related topics

- “Checking Control Builder error code reference” on page 171
- “Viewing flash log” on page 171
- “Viewing release information log” on page 171
- “Viewing trace log” on page 172
- “Checking version and revision log” on page 172
- “Checking server point build log” on page 172
- “Checking server point build error log” on page 172
- “Checking error log” on page 172

11.2.1 Checking Control Builder error code reference

An indication of a problem may be in the form of an error dialog that includes an error message and possibly an error code in Control Builder.

The syntax for a typical Control Builder error message is shown in the following illustration



In this syntax, the error code is the last five digits in the message or **14626**.

Refer to the *Control Builder Error Codes Reference* book for applicable error code information.

11.2.2 Viewing flash log

The Flash.txt log provides a list of firmware updates that have been initiated.

To view the log, navigate to this file location on the server: C:\Program Files\Honeywell\Experion PKS\Engineering Tools\system\bin\Flash.txt.

11.2.3 Viewing release information log

The ReleaseInfo.txt log provides a list of Experion software releases that have been installed on the computer.

To view the log, navigate to this file location on the server: C:\Program Files\Honeywell\Experion PKS\Engineering Tools\system\bin\ReleaseInfo.txt.

11.2.4 Viewing trace log

The TraceLogRs.txt log provides a list of definitions for strings associated with *breadcrumbs* data for given hardware components. The *breadcrumbs* provide a way to trace operations leading up to an event.

To view the log, navigate to this file location on the server: C:\Program Files\Honeywell\Experion PKS\Engineering Tools\system\bin\TraceLogRs.txt.

11.2.5 Checking version and revision log

The ver_rev.txt log provides a list of components by model number with software version/revision along with compatible Experion software release(s).

To check the log, navigate to this file location on the server: C:\Program Files\Honeywell\Experion PKS\Engineering Tools\system\bin\ver_rev.txt.

11.2.6 Checking server point build log

The SvrPtBld_servername.txt log provides list of process (CB) points built in the server database.

To check the log, navigate to this default file location on the server: C:\Documents and Settings\currentlogonprofile\Local Settings\Temp\SvrPtBld_servername.txt.

11.2.7 Checking server point build error log

The svrptblderr_servername.txt log provides list of any errors associated with process (CB) points built in the server database.

To check the log, navigate to this file location on the server: C:\Documents and Settings\currentlogonprofile\Local Settings\Temp\svrptblderr_servername.txt.

11.2.8 Checking error log

The Errlog_n.txt log provides a running list of Control Builder detected errors in chronological order. The n represents any number that is assigned to the most recent log.

To check the log, navigate to this file location on the server: C:\ProgramData\Honeywell\Experion PKS\ErrLog_n.txt.

Prior to R400, the Errlog_n.txt file was stored in the following location on the server: C:\Documents and Settings\All Users\Application Data\Honeywell\Experion PKS\Errlog_n.txt.

11.3 Getting Further Assistance

Related topics

“Other troubleshooting sources” on page 173

“Guidelines for requesting support” on page 174

11.3.1 Other troubleshooting sources

The following table lists other documents and sections that contain troubleshooting information for other Experion subsystems.

All of these documents are available from PDF collection. Some documents are also supplied as part of Station Help. For documents that can be accessed directly from this page, click on the link, otherwise look for the document within PDF collection. Substitute the latest Experion software release number for the XXX designation.

Document/Section	Comments
Experion RXXX > Reference	There is a separate interface reference for each type of controller other than the Process Controller; for example, the <i>ASEA Interface Reference</i> . Most of these references contain an interface-specific troubleshooting section.
Experion RXXX > Reference > TPS Integration Guide > Troubleshooting	Troubleshooting an integrated system that uses Experion “TPS Integration” option.
Experion RXXX > Reference > Control Builder Error Codes Reference	Describes error codes generated from within Control Builder.
Experion RXXX > Troubleshooting and Maintenance > Control Hardware Troubleshooting and Maintenance Guide	The main repository for troubleshooting, maintenance and repair of Process Controllers.
Experion RXXX > Configuration > DeviceNet Implementation Guide > Troubleshooting DeviceNet Status Failures	Describes error codes generated from DeviceNet Interface Board.
Experion RXXX > Installation and Upgrades > Fault Tolerant Ethernet Bridge Implementation Guide > Service > Troubleshooting	Troubleshooting FTE bridges.
Experion RXXX > Installation and Upgrades > Fault Tolerant Ethernet Installation and Service Guide > Troubleshooting FTE Nodes	Troubleshooting FTE nodes.
Experion RXXX > Reference > Honeywell TDC 3000 Data Hiway Interface Reference > TDC error codes and Troubleshooting	Troubleshooting TDC 3000 Hiway problems.
Experion RXXX > Configuration > Qualification and Version Control System User Guide > QVCS Troubleshooting	Troubleshooting QVCS.
Experion RXXX > Operations > SafeView User's Guide > Appendix D - SafeView Error Messages	Describes the meaning of SafeView configuration errors.
Experion RXXX > Reference > Server Scripting Reference > Server scripting error messages	Describes the meaning of error messages in the server log specific to server scripting.
Experion RXXX > Reference > System Management Configuration Guide > Troubleshooting System Management	Describes the meaning of System Management Configuration errors.

Document/Section	Comments
Experion RXXX > Reference > System Management Configuration Guide > Troubleshooting SES	Describes the meaning of SES Configuration errors.
Experion RXXX > Reference > System Management Configuration Guide > Troubleshooting SPS	Describes the meaning of SPS Configuration errors.
Experion RXXX > Planning and Design > Planning, Installation, and Service for WS360	Troubleshooting workstation nodes used in Experion and TPN.

11.3.2 Guidelines for requesting support

If you cannot resolve a problem using this guide, you can request support from your Honeywell Solution Support Center.

When requesting support, please supply as many relevant details as possible, including:

- **Short summary of the problem**
- **Product Name and release.**
- **Recent changes**, such as upgrades/service packs, to Experion software, Windows or other applications.
- **Subsystem and its version/build**, if the problem relates to a particular subsystem, such as Station or Quick Builder. If the problem relates to **Display Builder**, please specify whether it is HMIWeb Display Builder (for HMIWeb displays) or Display Builder (for DSP displays).
- **Operating system, variant and service pack**, for example “Windows 2000 Server, SP5”.
- **Instructions on how to reproduce the problem.** If the problem is reproducible, please supply step-by-step instructions - the more detailed the steps the better.
- **Diagnostic package which contains any relevant logs.**

12 Notices

Trademarks

Experion®, PlantScape®, SafeBrowse®, TotalPlant®, and TDC 3000® are registered trademarks of Honeywell International, Inc.

OneWireless™ is a trademark of Honeywell International, Inc.

Other trademarks

Microsoft and SQL Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Trademarks that appear in this document are used only to the benefit of the trademark owner, with no intention of trademark infringement.

Third-party licenses

This product may contain or be derived from materials, including software, of third parties. The third party materials may be subject to licenses, notices, restrictions and obligations imposed by the licensor. The licenses, notices, restrictions and obligations, if any, may be found in the materials accompanying the product, in the documents or files accompanying such third party materials, in a file named third_party_licenses on the media containing the product, or at <http://www.honeywell.com/ps/thirdpartylicenses>.

12.1 Documentation feedback

You can find the most up-to-date documents on the Honeywell Process Solutions support website at:

<http://www.honeywellprocess.com/support>

If you have comments about Honeywell Process Solutions documentation, send your feedback to:

hpsdocs@honeywell.com

Use this email address to provide feedback, or to report errors and omissions in the documentation. For immediate help with a technical problem, contact your local Honeywell Process Solutions Customer Contact Center (CCC) or Honeywell Technical Assistance Center (TAC) listed in the “Support and other contacts” section of this document.

12.2 How to report a security vulnerability

For the purpose of submission, a security vulnerability is defined as a software defect or weakness that can be exploited to reduce the operational or security capabilities of the software.

Honeywell investigates all reports of security vulnerabilities affecting Honeywell products and services.

To report a potential security vulnerability against any Honeywell product, please follow the instructions at:

<https://honeywell.com/pages/vulnerabilityreporting.aspx>

Submit the requested information to Honeywell using one of the following methods:

- Send an email to security@honeywell.com.
- or
- Contact your local Honeywell Process Solutions Customer Contact Center (CCC) or Honeywell Technical Assistance Center (TAC) listed in the “Support and other contacts” section of this document.

12.3 Support

For support, contact your local Honeywell Process Solutions Customer Contact Center (CCC). To find your local CCC visit the website, <https://www.honeywellprocess.com/en-US/contact-us/customer-support-contacts/Pages/default.aspx>.

12.4 Training classes

Honeywell holds technical training classes on Experion PKS. These classes are taught by experts in the field of process control systems. For more information about these classes, contact your Honeywell representative, or see <http://www.automationcollege.com>.

