QlikView

QlikView 11 Source Control Walkthrough

A QlikView Technology White Paper

Originally published: August, 2011

Updated August, 2012

www.qlikview.com

QlikView

Table of Contents

Background

Starting in version 11, QlikView has the capability to integrate with source control systems. In the initial release a Microsoft TFS connector was introduced and a further connector for Subversion was added in SR1. The feature is available when using QlikView Desktop.

A menu option in the **File** menu allows QlikView developers to connect a QlikView document to a source control system. This feature works with the XML project files that QlikView Desktop can generate and offers basic integration into third party source control systems.

QlikView 11 Source Control integration now ships with native support for Microsoft Team Foundation Server and Subversion for source control.

Source Control Basics

You can connect QlikView Desktop to a source control system.

Once connected to a supported source control system, QlikView developers can add projects to source control. The following occurs during the Add process:

- The QlikView document is saved.
- The project folder is created.
- The project files are exported into the project folder.
- The project files are added to source control.
- The project settings file is created.

The project settings file is stored in the local project folder and contains the settings necessary to access the source control information for the project. The project settings file is not included in the files managed by the source control system. The existence of the project settings file notifies QlikView that a given document is managed by a source control system. This check is performed every time a document is opened in QlikView and in QV11 SR1 onwards, a settings file appears in the project folder with information about the Source Control Provider, i.e. SourceControlSettings.ini. The only exception relates to backward compatibility issues. If a project folder ‘abc-pjr’ does not contain ‘SourceControlSettings.ini ‘but contains a file called ‘abc-prj.scc’ it means it was created using the V11 Initial Release, not SR1 onwards.

If the document is attached to source control, the status bar contains an indication of the document's status.

QlikView integration with source control only affects document layout. No actual data loaded into QlikView is placed into nor fetched from, source control. The operation, **Get Project from Source Control** loads a document that contains everything except data. A Reload must be executed to populate the document with data.

After a QlikView document is connected to source control saving the document automatically checks out the files that have changes. QlikView does not provide any other way of checking out the project files.

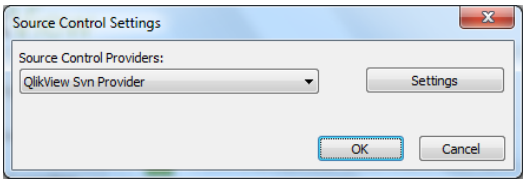
If Source Control system is not available while performing document Save, QlikView tries to work "offline", i.e. removes read-only flags from the modified files and saves the latest version. On the next document save, when Source Control system is available, QlikView performs check-out for all locally modified files.

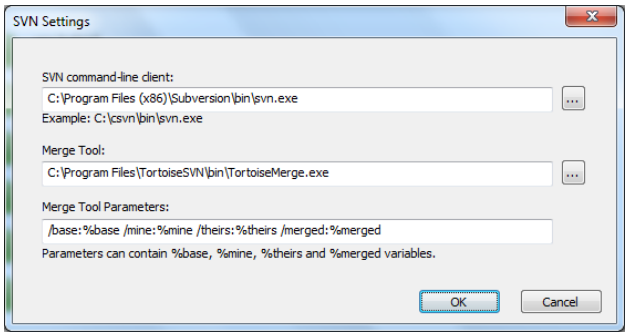
On document save, project files are not checked in. To check in document updates to source control, the **Check In Pending Changes** menu item must be used. A single QlikView check-in operation can generate several change sets when using Microsoft TFS and only one when using Subversion. As a result, rollback between randomly selected change sets in Microsoft TFS does not guarantee that the document will be in a stable state. If you have written your own custom provider, you need guidance from you source control administrator.

Preparations and Upgrade Considerations (V11 to V11 SR1)

In the initial release the source control system needed an implementation of the MSSCCI API to use this feature, but this has been replaced, meaning 'MsscciWrapper32.exe' is no longer a part of the installation. Two new files, 'QvMsscciProvider.exe' and 'QvSvnProvider.exe' are installed in the same folder as 'QV.exe'.

Note: Before embarking on the following process, it is assumed that you have a working knowledge of TFS or Subversion and general best practices associated with source control systems. It is also assumed that your system administrator has reviewed the text found in the QlikView manual and this White Paper, and following installation of QlikView 11 SR1 or the planned SR2, has set up integration with your TFS or SVN system ensuring that necessary integration settings have been applied. For instance, QlikView client requires settings input at **File/Source Control Settings/Settings** and the following screens show the settings used while testing with Subversion and Tortoise Merge tool.





Now you want to add your existing production QVW files to TFS or SVN and use as your source code manager. Source control adds a ‘prj’ folder for each document that is checked in. The folder name is the same as the QVW and will contains the contents of the QVW exploded into individual XML files.

For example, ‘ABC.qvw’ will have a folder of ‘ABC-prj’ and will be located in the same directory path as the QVW itself.

If your existing documents already have ‘prj’ folders, it is likely that you have used some other source control system. Migration of project files from earlier QlikView versions to QlikView 11 needs more care if you already use TFS or another source system to track changes. See the release documents for QlikView 11 onwards and speak to your QlikView consultant if you are unsure of how to proceed.

Project File Architecture

For reference, the project file architecture has been changed between QlikView 10 and QlikView 11.

- The QlikView.txt file is replaced by QlikViewProject.xml.
- The QlikViewProject.xml file contains all information about inter-object relationships (objects on a sheet, objects in a container, etc). Previously, these relationships were diverged between TopLayout.xml, SHxx.xml, CTxx.xml and AllProperties.xml.
- DocProperties.xml is updated – user-specific information, such as reload and save information is removed.
- Objects IDs are generated in order, i.e. the first list box is identified as LB01, the next as LB02, etc. If multiple users add objects to the same project, objects that are not related to each other may get the same Object ID. After adding the project to source control, each new object is assigned a random suffix to its Object ID, for example LB03_473861035. This ensures that all objects get unique identifiers.

In addition to this the QV 11 SR1 release brought the following change:

- New settings file in the project folder. All projects that are added to Source Control from QV11 SR1 will have a settings file with information about the Source Control Provider, i.e. SourceControlSettings.ini. QlikView needs to know which Provider to use when opening the project.

Changes to Graphical User Interface

- For reference, the project file architecture has been changed between QlikView 11 and QlikView 11 SR1.
- Settings dialog different. The dropdown list with all providers and a button to send a message to the provider to open the selected provider settings dialog.

- Add Project to Source Control/Get Project from Source Control. Use the Source Control Provider selected in the settings. ‘QvSvnProvider’ is the default.
- Language changes in QlikView will not be reflected in the Source Control Providers. Messages from the Msscci provider are shown in English only.

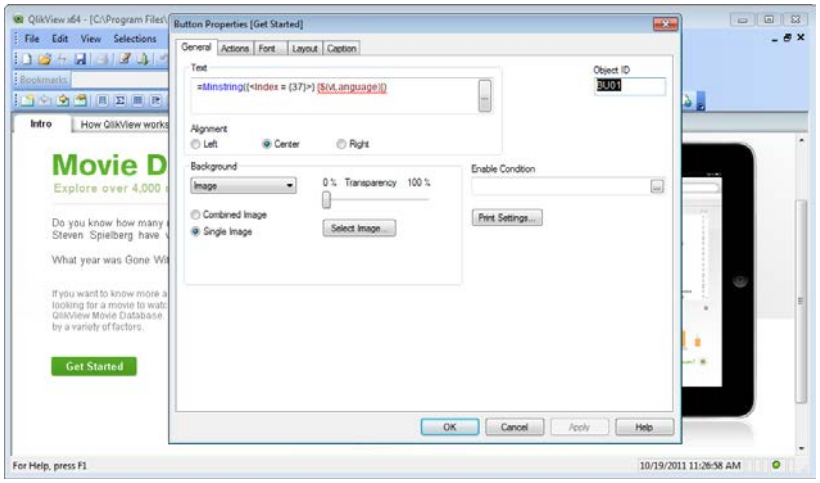
Known Issues

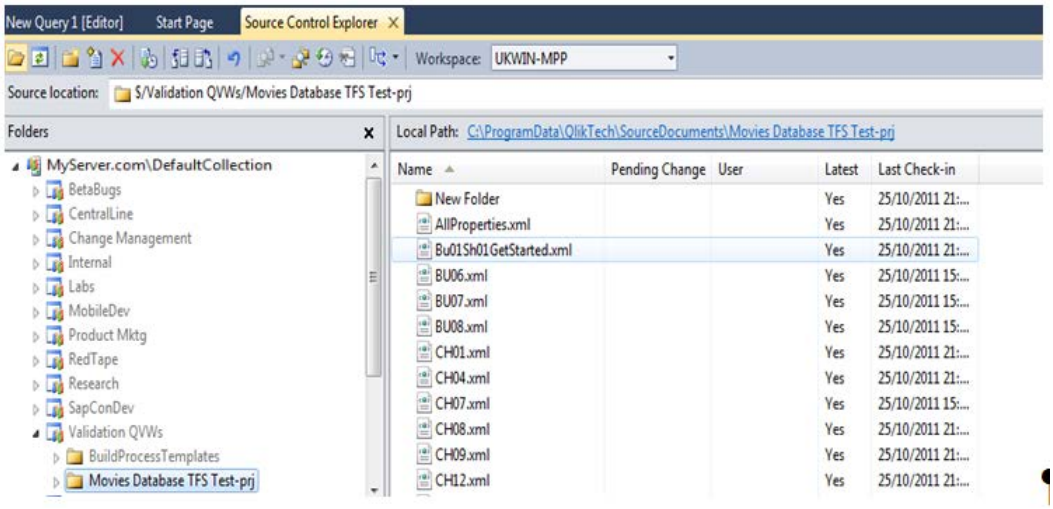
With QlikView 11 SR1 we support SVN 1.6.17. If you use SVN 1.7.x, problems may occur, e.g. with the link to the file to compare; there are two paths in the link, e.g. C:/xxxxxxxx and C:/xxxxxxx. **QlikView 11 SR2 will support SVN 1.7.**

Something to Consider

The checked-in files use the Object ID as the identifier (Name). Prior to the initial check-in of your documents, these can be edited in the properties of each object to make identification of objects and their location easier when using source control. This also makes it easier when viewing the objects in the QlikView document, by selecting **Settings/Document Properties/Sheets**.

The following is an example of what may be a more suitable ID – Bu01Sh01GetStarted (Button01/Sheet01/GetStarted).





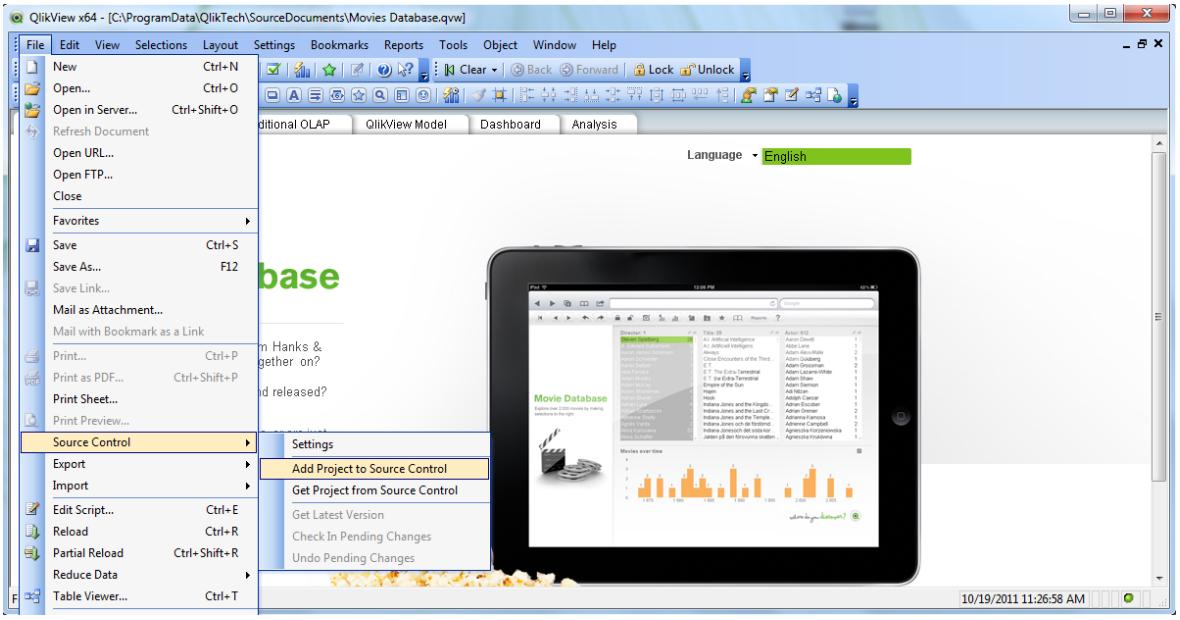
Should you wish to introduce naming conventions post initial check-in, it is likely you will cause inconsistencies in the ‘prj’ folder. Ensure that object naming conventions are considered before you start the source control process.

Source Control Using Microsoft TFS

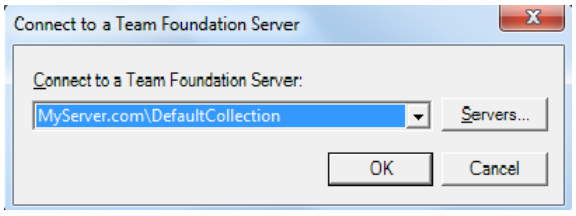
Adding QlikView Files to Source Control Using Microsoft TFS

In this example, an application called *Movies Database* is used.

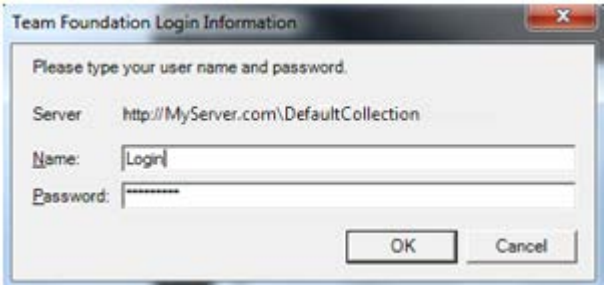
Start by opening the first document that you wish to add to source control, and select **File/Source Control/Add Project to Source Control**.



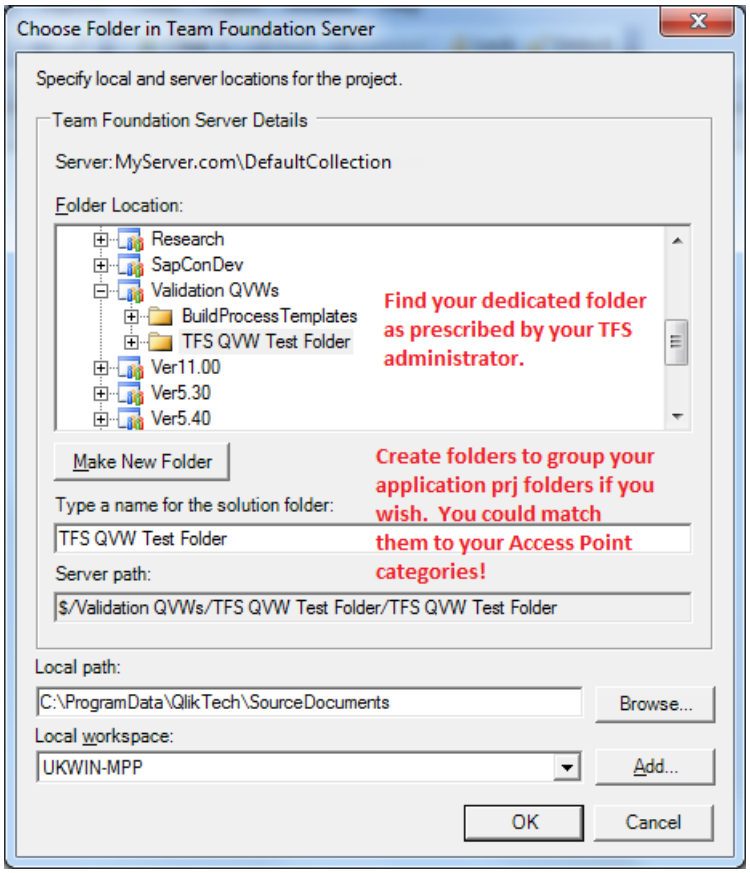
You are asked for the TFS server login details. Your TFS system administrator would have advised you of these. If you have multiple TFS servers and/or 'Collections', you need to ensure that you are using the correct one.



Typically your logon details will mimic your network login.

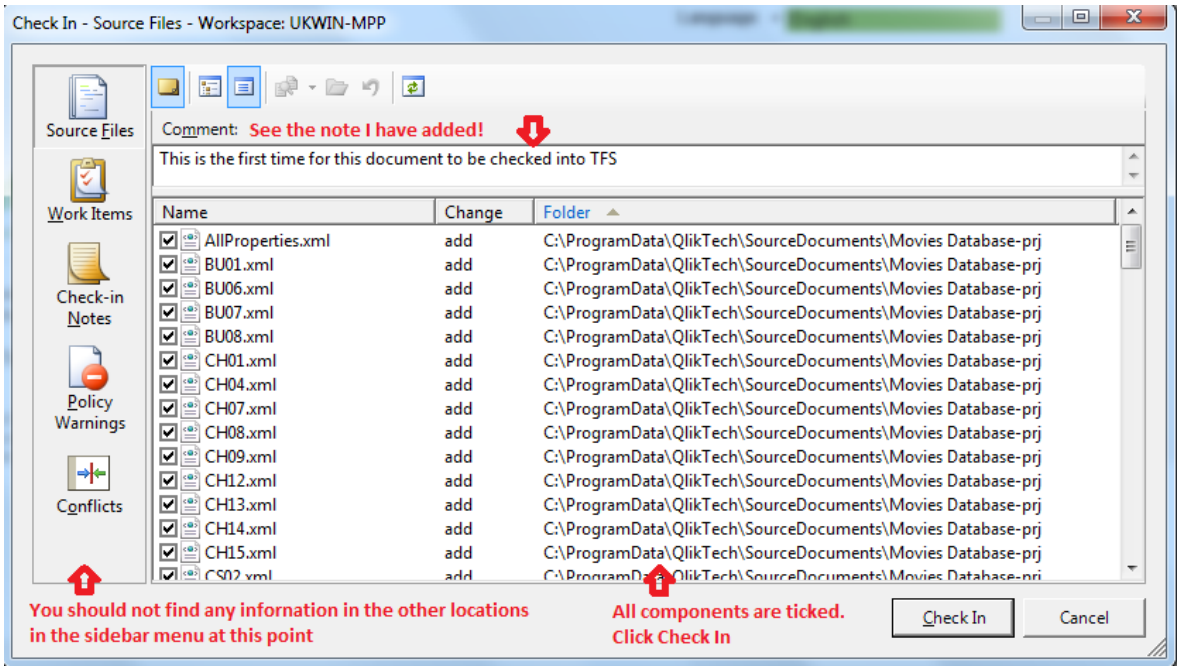


You need to browse to the folder to store your project name in. You can add folders to group your application project files. You may wish to match these to the categories on your Access Point for easy association.



After selecting the correct location, click **OK** and you will be transferred to the **Check In – Source Files – Workspace** screen.

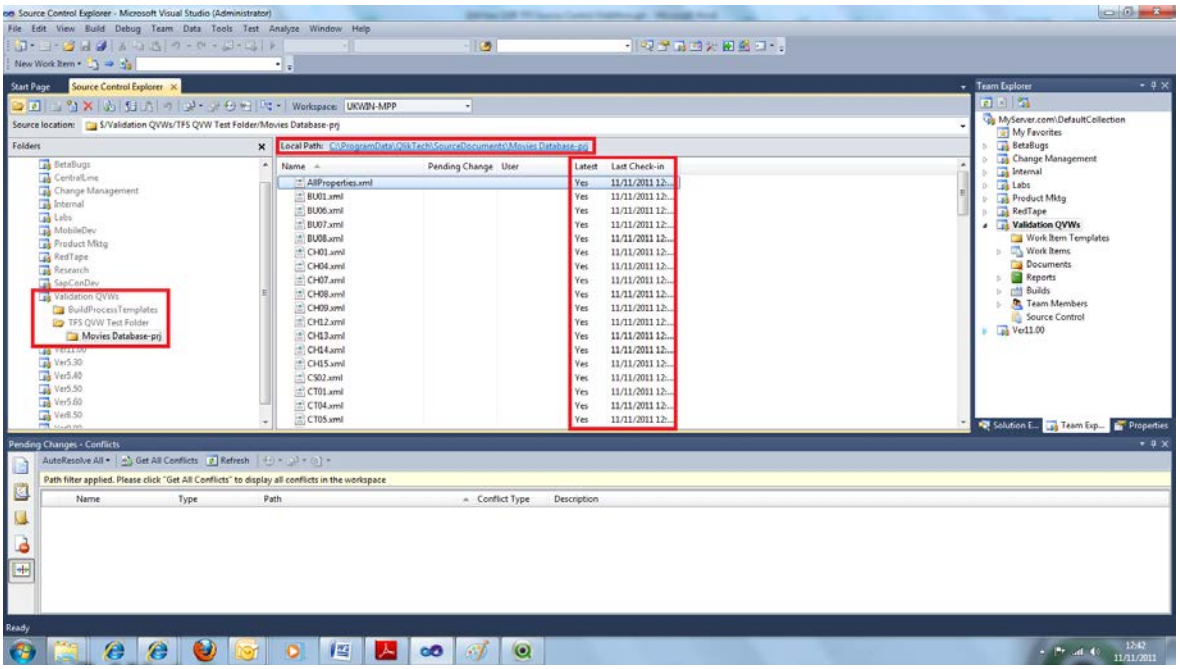
You notice that the local path of this example is *SourceDocuments* which is where tasks open/reload/distribute production documents from. In your environment this may be a *DevelopmentSourceDocuments* folder and the final step to implement to production may be a simple action of overwriting the version in the *SourceDocuments* folder when you are ready.



After typing a descriptive note, click **Check In**.

You see a progress bar which closes on completion, and you are left with the open QlikView document which you can close.

Using Microsoft Visual Studio, you can browse to the TFS project folder you just created by adding your application to source control.



On the left, you see the folder created by the TFS administrator for this example, and the resulting folder created when the *Movies Database TFS Test* QlikView application was added. At the top you can see the local path from which the developer is working. The main pane shows the document subjects, showing them all as *Latest* and with their *Last Check-in* date. On the right you see the Team Explorer which can be used to add work items and documents to the projects.

You can close this for now.

Making Changes to QlikView Files Added to Source Control Using Microsoft TFS (Single Developer)

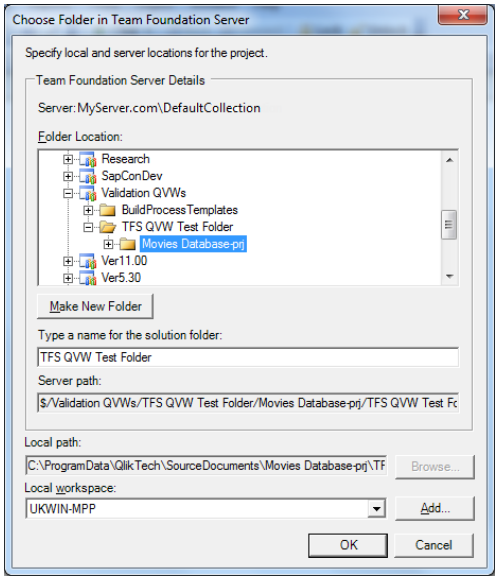
The following section describes the process when there is only one developer accessing the source control at any one time.

As a developer, you may need to edit a document you previously obtained from source control. It may even be a document you added to source control yourself. To edit a document, you simply open QlikView and either select your local copy of the document from the latest documents list, or browse to it in Windows Explorer and open it. It is important that when the document opens, you get the latest version to ensure you make your changes on current checked-in version. You can do this by selecting **File/Source Control/Get Latest Version**.

This is good practice in both single- and multi developer environments.

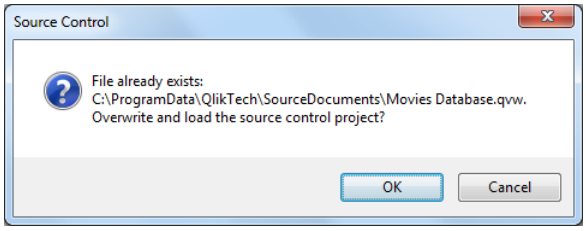
To edit a document that has been added to Source Control for the first time, you need to obtain a copy of the source file from the server. You then apply changes to your local copy, until you are ready to check in.

In QlikView, this can be done by selecting **File/Source Control/Get Project from Source Control**. You may have to select the correct server again and supply login details, but you will be able to navigate to the TFS folder that holds the projects for each QlikView application that you have added to source control. Select the project and click **OK**.



The example project is selected

In case a local copy already exists, you are notified by the following dialog. If you have no changes in your local file that you need to keep, click **OK**. You are asked to accept creation of the folder, if it does not already exist.

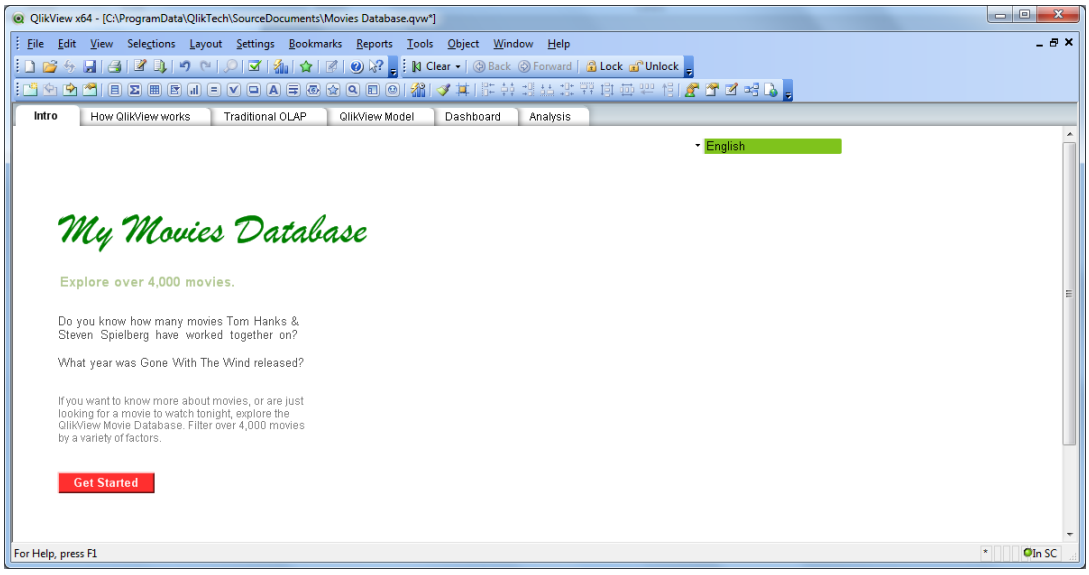


Opening the QlikView document this way means you open it with no data, which is how it is stored. To work with data, reload the script. Now you can continue working on the latest version and make any necessary changes.

Test Source Control Check-In Using Microsoft TFS

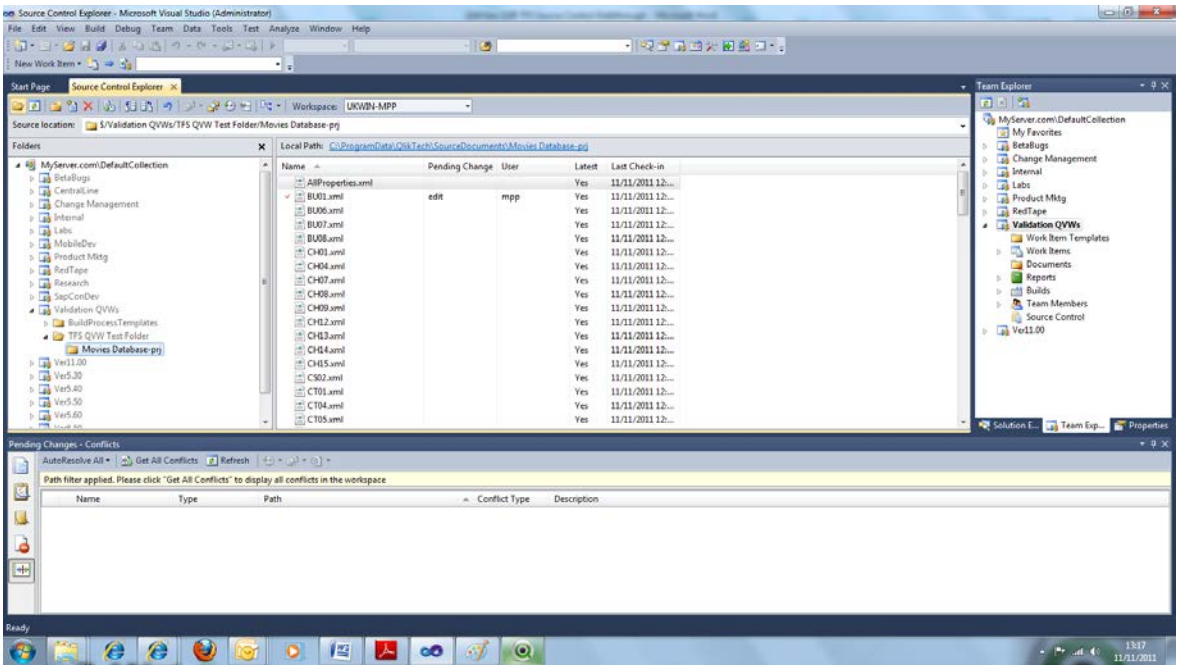
Perform the following steps to test checking in an updated document:

1. Change an object - in this example turning the *Get Started* button to light red in color and removing the background image on sheet 1.
2. Delete an object – in this example removing the label *Language* next to the language selection field on sheet 1.
3. Create an object – in this example adding a text box with a new title on sheet 1.



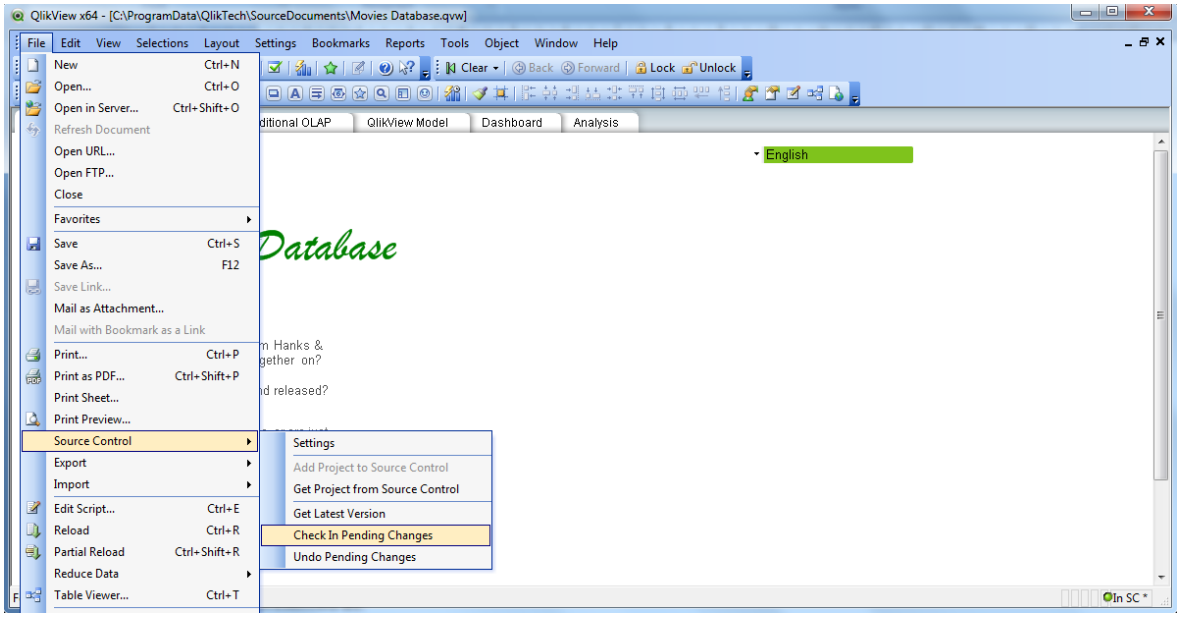
The local result of the example

Note: The very act of saving places the changes ready for check-in on the TFS folder, as you can see in the following view in Microsoft Visual Studio.



When you are happy with your changes, you want to check the changes into source control, ready for the next reload and distribution.

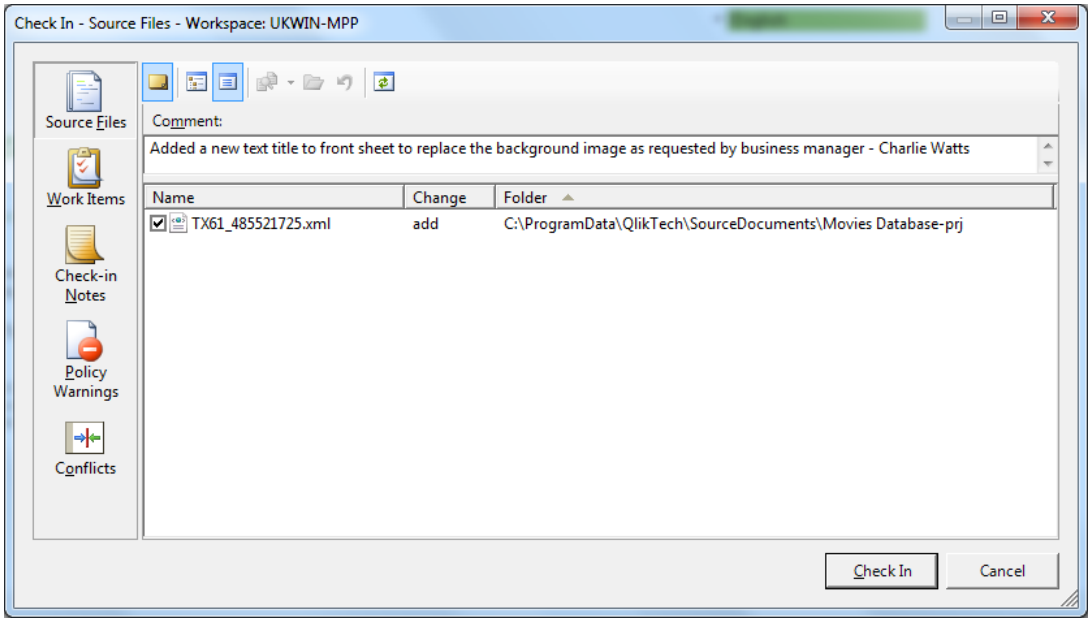
Select **File/Source Control/Check In Pending Changes**.



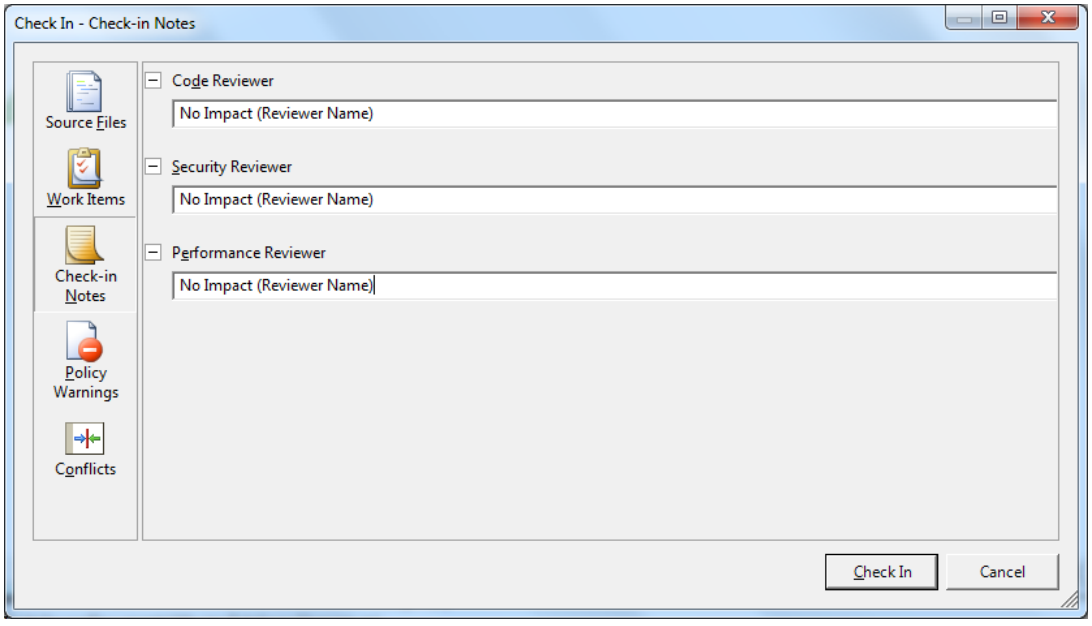
Each set of change types is then presented in sequence for notes and approval. The sequence is presented as:

1. *add* changes
2. *delete* changes
3. *edit* changes

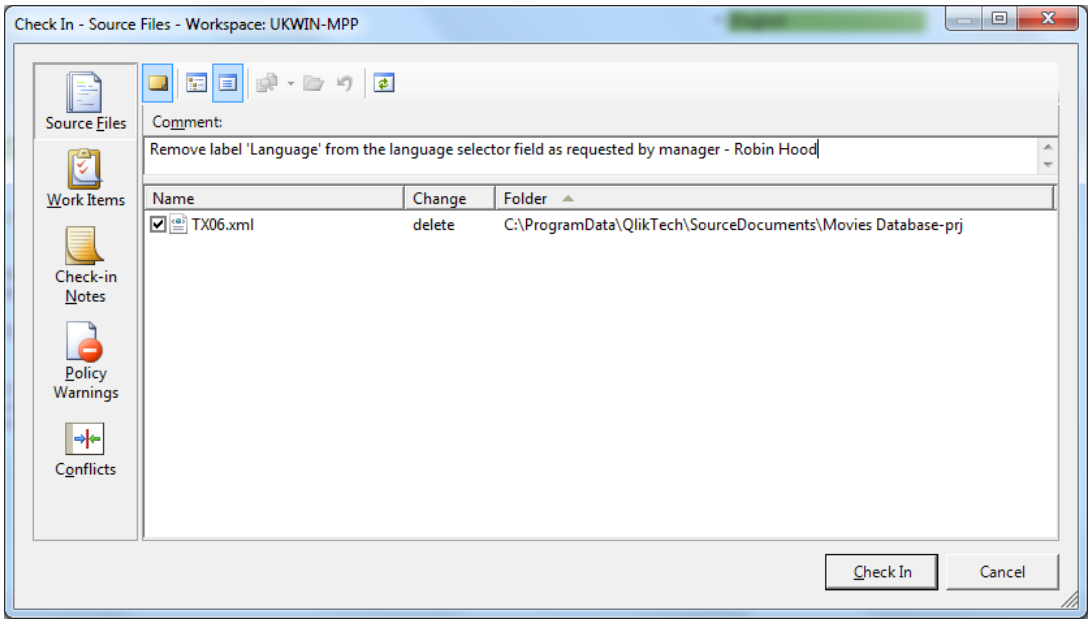
Type a check-in comment for the *add* check-in.



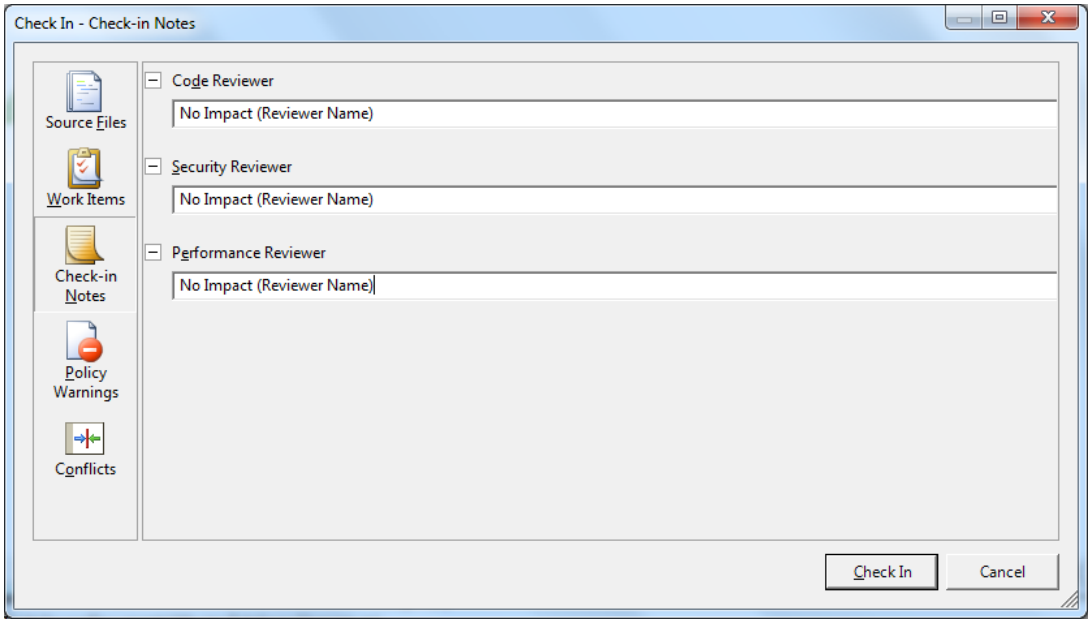
Before clicking **Check In**, you may wish to highlight impact and identify the controller signing off by name by clicking **Check-in Notes**.



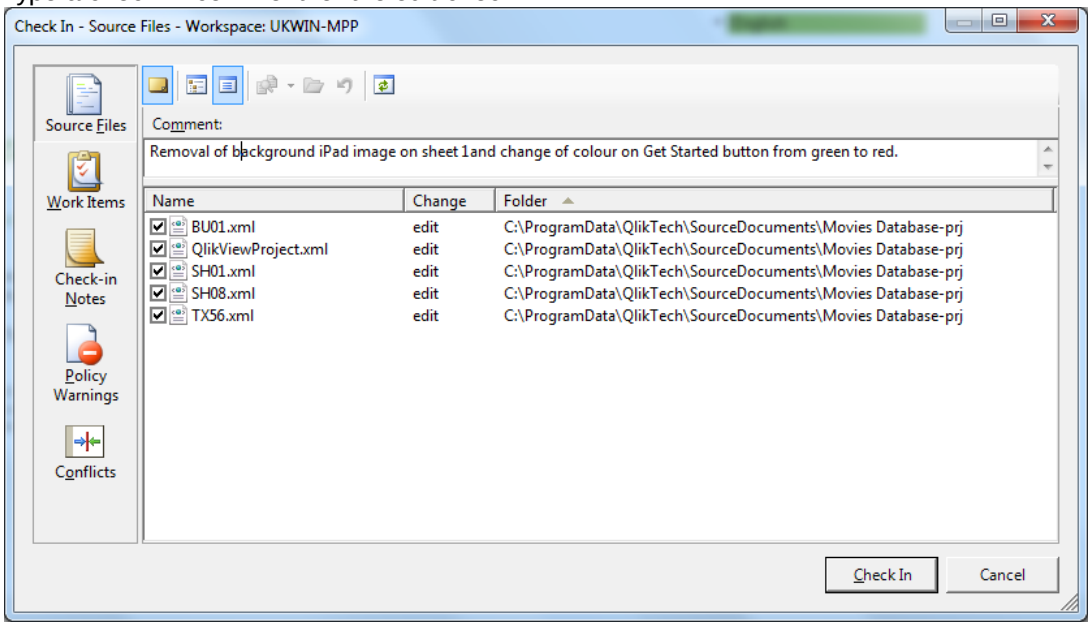
Type a check-in comment for the *delete* check-in.



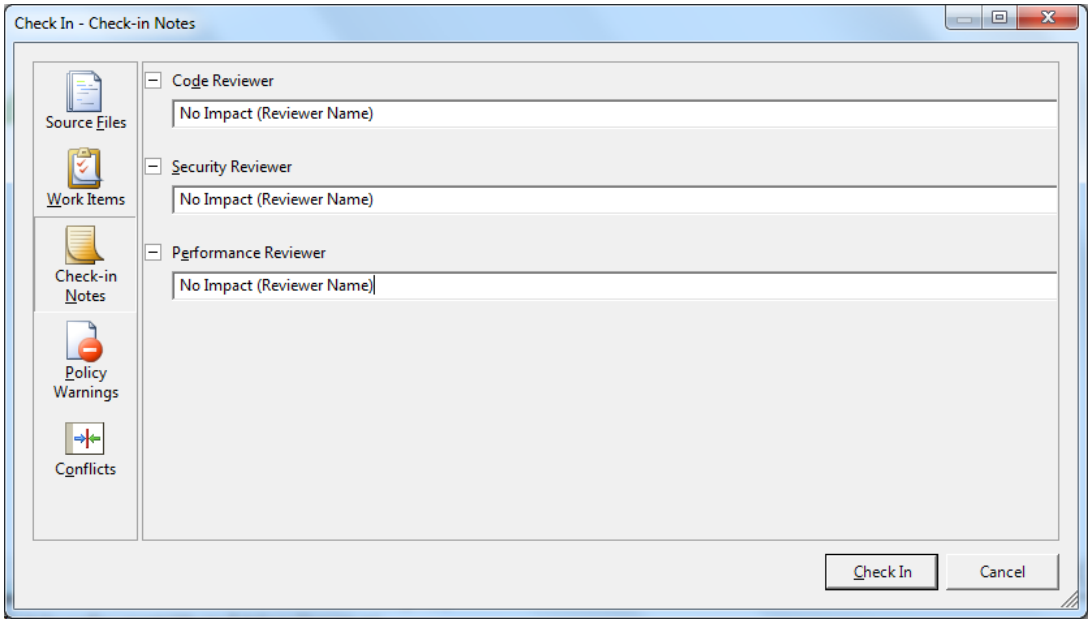
Before clicking **Check In**, you may wish to highlight impact and identify the controller signing off by name by clicking **Check-in Notes**.



Type a check-in comment for the *edit* check-in.

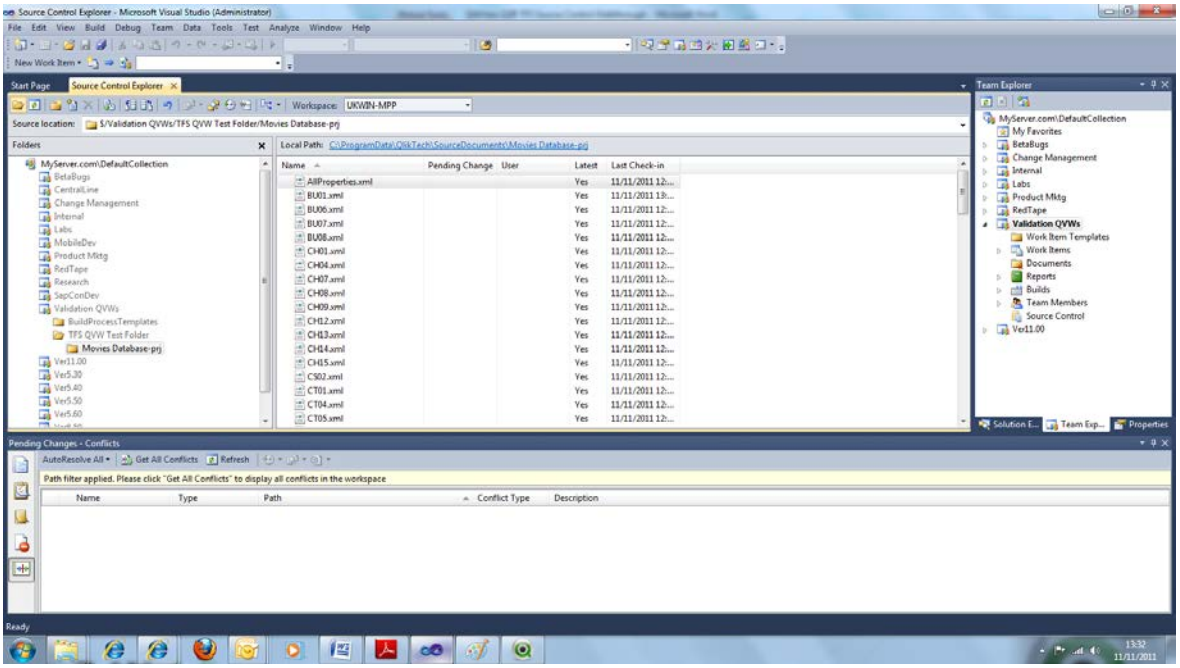


Before clicking **Check In**, you may wish to highlight impact and identify the controller signing off by name by clicking **Check-in Notes**.



The changes have been successfully checked in and the QlikView document can now be closed.

The changes will now be reflected in Microsoft Visual Studio as follows.



You may find it useful to work on add/edit/delete over separate check-in jobs for more precise rollback options, i.e. make desired additions and check in, then make desired edits and check in, etc.

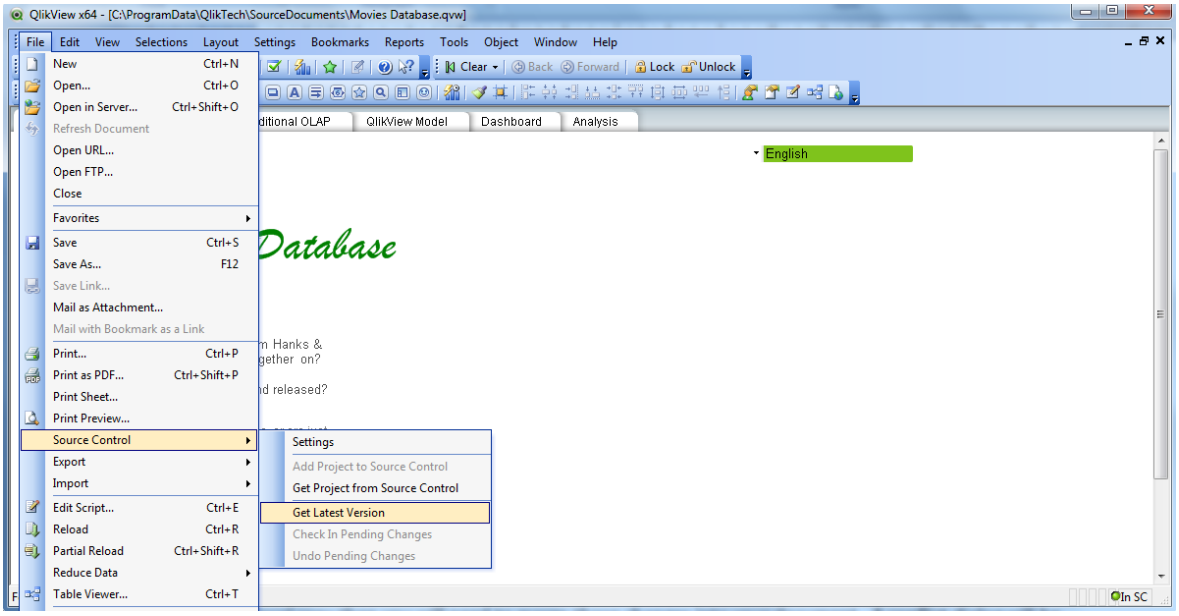
If you check TFS at this point, there should be no pending changes and the folder viewer should have the latest object files dated with their most recent timestamp.

You have completed an example of adding and editing a QlikView document that is added to TFS source control!

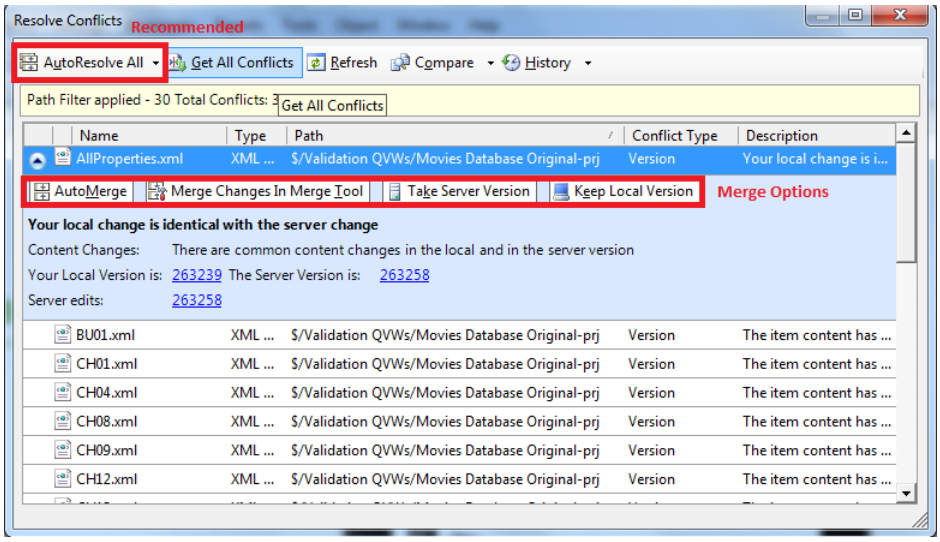
Making Changes to QlikView Files Added to Source Control Using Microsoft TFS (Multiple Developer)

The following section describes the differences to the process when there is more than one developer accessing the source control at any one time. Again, it is assumed that you have a working knowledge of TFS and general best practices associated with source control systems.

In a multi developer process there is an extra step. If you are working on a copy and are about to check in you changes, it is essential that you update your copy before doing so, by selecting **File/Source Control/Get Latest Version** in QlikView.



If someone else has checked in changes to the document after you started making changes to your local copy, you need to merge those changes into your document. A conflict dialog is displayed:

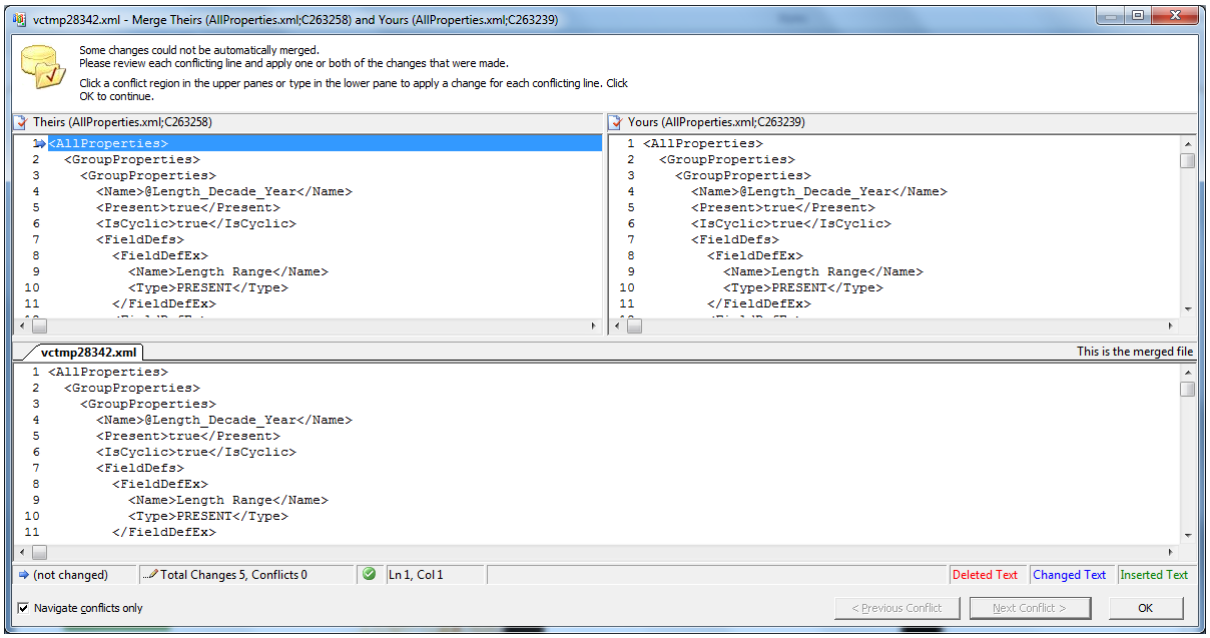


Selecting **AutoResolve All** conflict types resolves most issues. For the remaining items you need to select one of the highlighted merge options.

If the item is an object that you have not changed in your copy, you can choose to keep the server object, but when possible, you may want select **Merge Changes In Merge Tool**. This allows you to compare the XML code and choose which version you want to keep, or force a merge.

Select **Navigate conflicts only** to make the process quicker. When you reach a conflict, you can decide which copy to take.

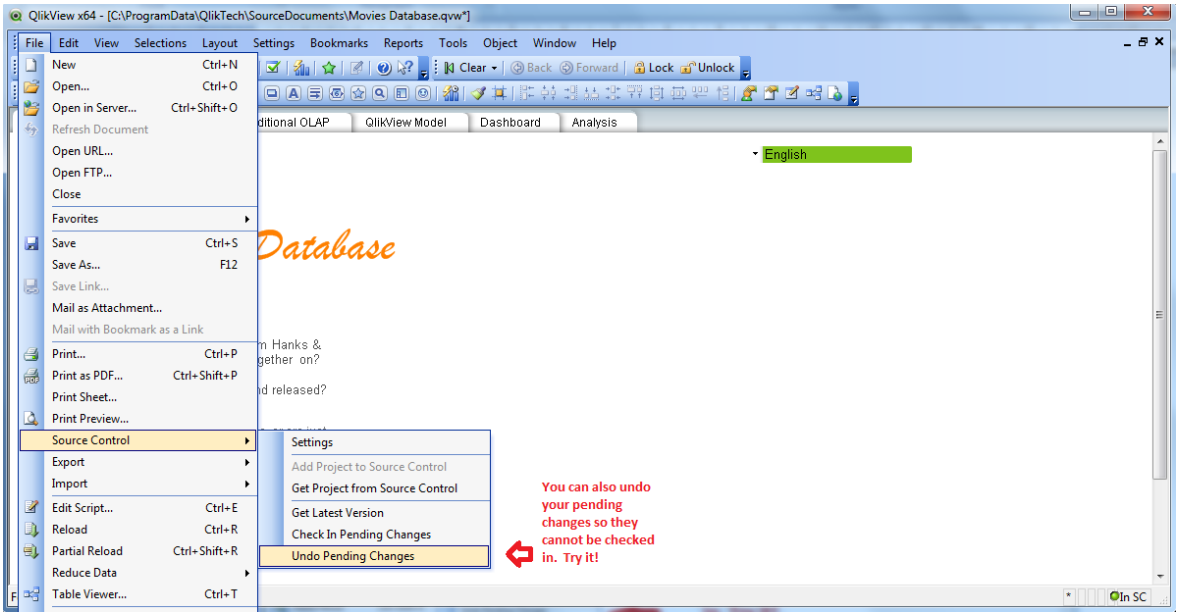
In the following example, for the last conflict in this file, the server version is selected (**Theirs**). As it is the last conflict, I can click **OK**, and all my conflicts in this file will be resolved as I requested.



A conflict may be that an object has been added to the server file, and you have created an object and you need to merge both in the project file. In this case at the point of conflict, you would highlight both the **Theirs** and **Yours** and then navigate to the next conflict or select **OK** if you have reached the end of conflicts in that particular file. A conflict in the binary file should be discussed with the system owner as it is likely that important security information has changed, and selecting the correct version needs appropriate care. When there are no conflicts left, the merge tool can be closed and your open QVW will be refreshed containing all the changes you elected to merge.

You can now check in your changes free from conflict. Naturally, checking in should be coordinated and the process previously described should only be performed by one person at a time. At this point you may decide not to check in your changes. If so, select **File/Source Control/Undo Pending Changes**.

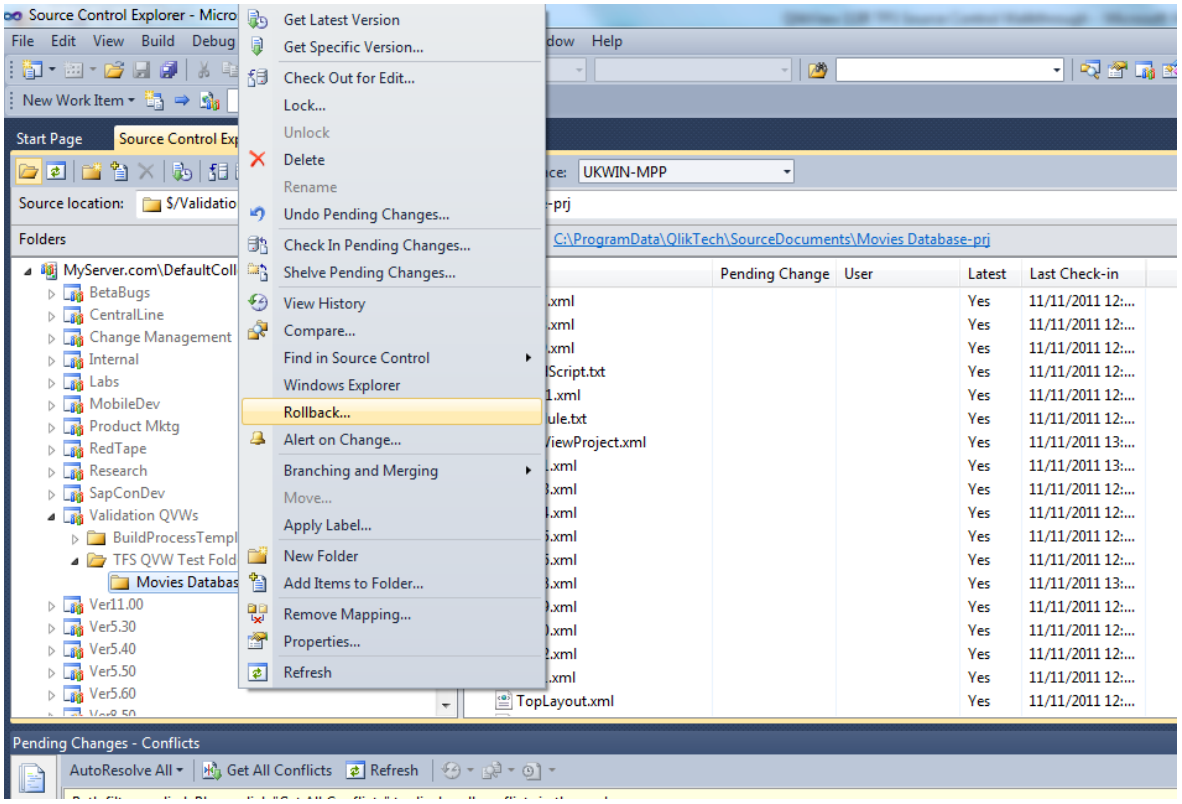
Example: Undo Pending Changes



In the example application, the title text was changed to orange, and the document was saved. By selecting **File/Source Control/Undo Pending Changes**, this change was disregarded. Checking Microsoft Visual Studio throughout the process enables you to see how the changes from the save become items to check in, but are removed when selecting this option.

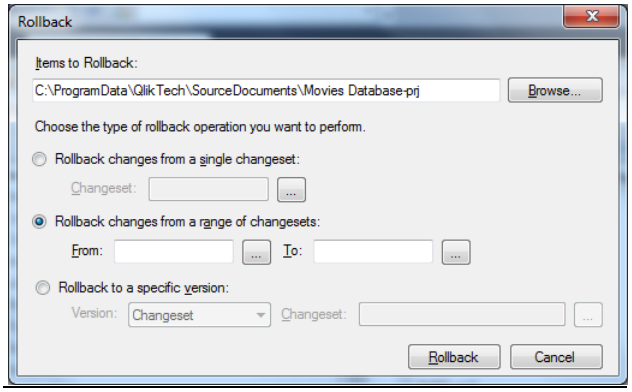
Retrieving Previous Versions (Using Microsoft Visual Studio)

From inside QlikView, you can only select to get the version from the server, or to get the latest version from the server/local files. TFS holds multiple versions and not just your latest currently pending check-in version. It is possible to perform a rollback, but remember that this should be coordinated in good source control fashion.



You need to open TFS; typically by opening Microsoft Visual Studio. To use the Rollback function, TFS Power Tools has to be installed. If these tools are not installed, speak to your TFS administrator. After having installed TFS Power Tools, you can navigate to the correct folder and right-click the folder to use **Rollback** as shown in the preceding picture.

After having selected **Rollback**, you are presented with the following dialog:

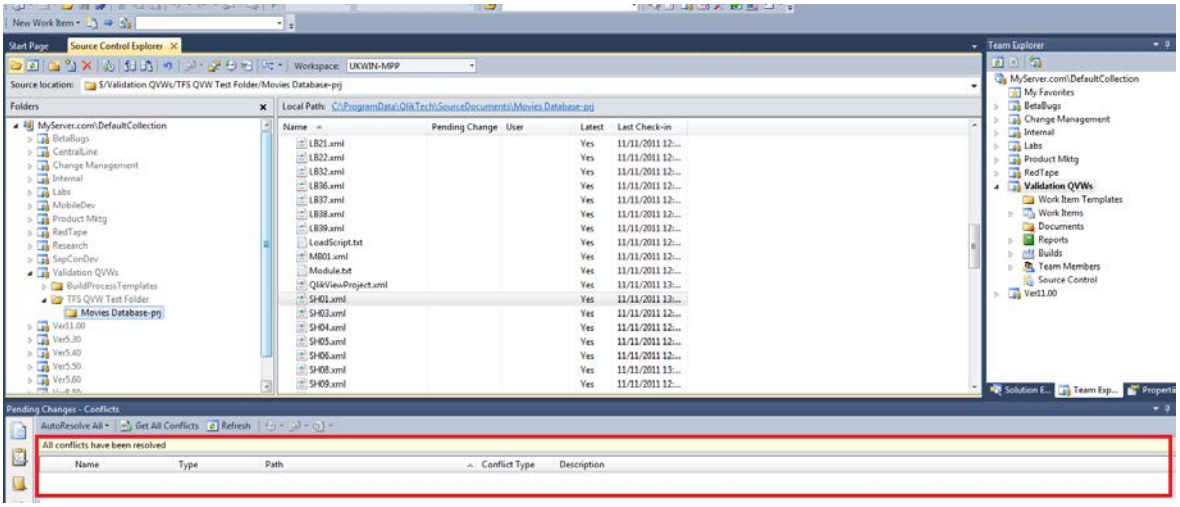


As explained earlier, *add*, *edit* and *delete* actions receive different change set numbers in the same check-in, which means that you need to rollback all the changes from a single check-in of a QVW to ensure that the structural files remain clean. That is why your preference may be to split the three different types of changes into different work packages each with their own check-ins. Good notation is very important to identify check-ins with multiple change types, so these can be rolled back together.

If *add*, *edit*, and *delete* actions were contained in the check-in you wish to include in a rollback, they are revealed in the search with consecutive change set numbers, but the notation is what really enables you to identify them. For example, “Splash Sheet Update Nov 2011 Text Box I love Films ADDED” and “Splash Sheet Update Nov 2011 Language Label DELETED” would generate two change set numbers for “Splash Screen Update Nov 2011”.

Selecting **Rollback changes from a range of change sets** allows the associated range to be selected and those changes will be reversed. If you check in after a change set type (e.g. *add*), you could select **Rollback changes from a single change set** and select that change set only.

Any conflicts need to be merged and checked in within Microsoft Visual Studio.



A developer who opens a local copy needs to select **File/Source Control/Get Latest** to get the rolled back version.

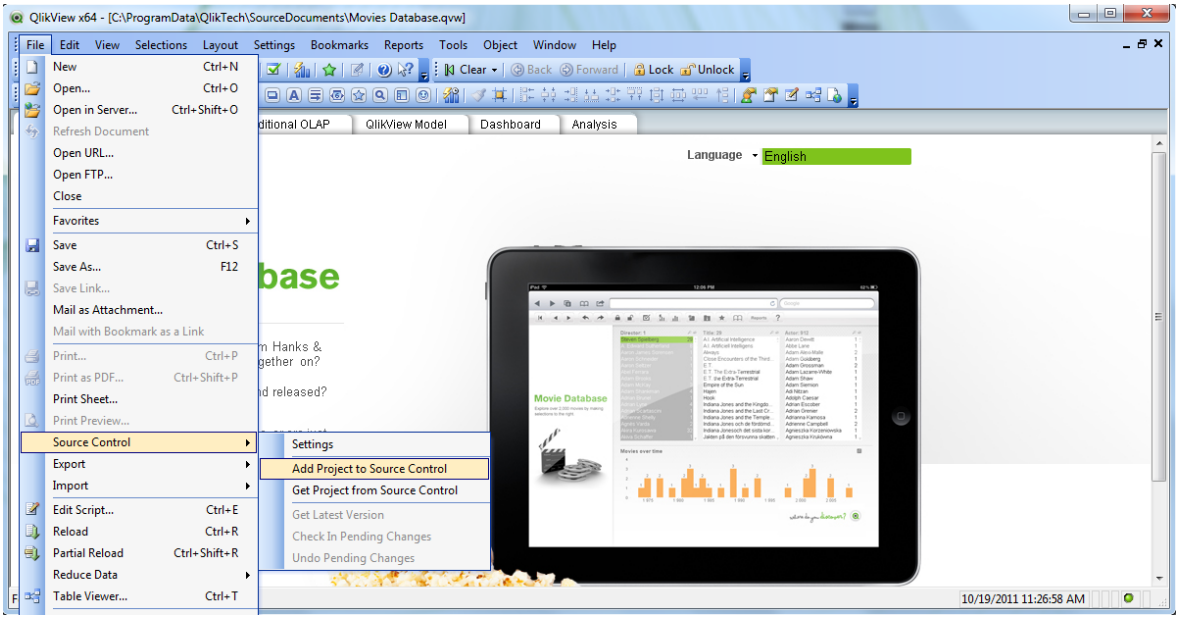
Besides *Rollback from* (out of) a change set you can also *Rollback to* a change set. The highest change in a change set associated with a single check-in should be selected if that is to happen.

Source Control Using Subversion

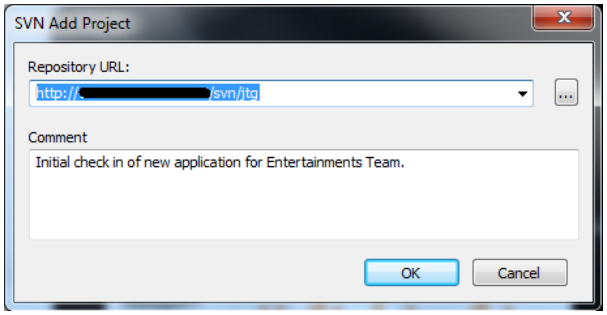
Adding QlikView Files to Source Control Using Subversion

In this example, an application called *Movies Database SVN* is used.

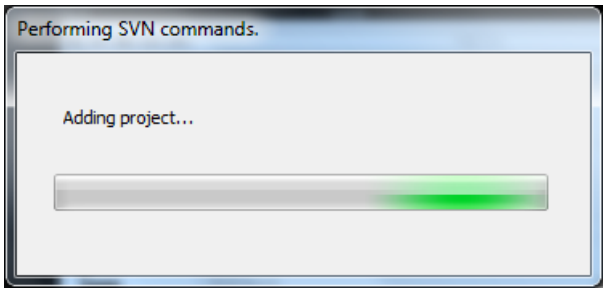
Start by opening the first document that you wish to add to source control, and select **File/Source Control/Add Project to Source Control**.



You are asked to select a Subversion repository URL. Your Subversion system administrator would have advised you of these. If you have multiple Subversion servers and folders you need to ensure that you are using the correct URL. Make a check-in comment like in the following dialog.



You may see the following screen while the project is added to the pre-designated Subversion server and folder and your designated local directory.

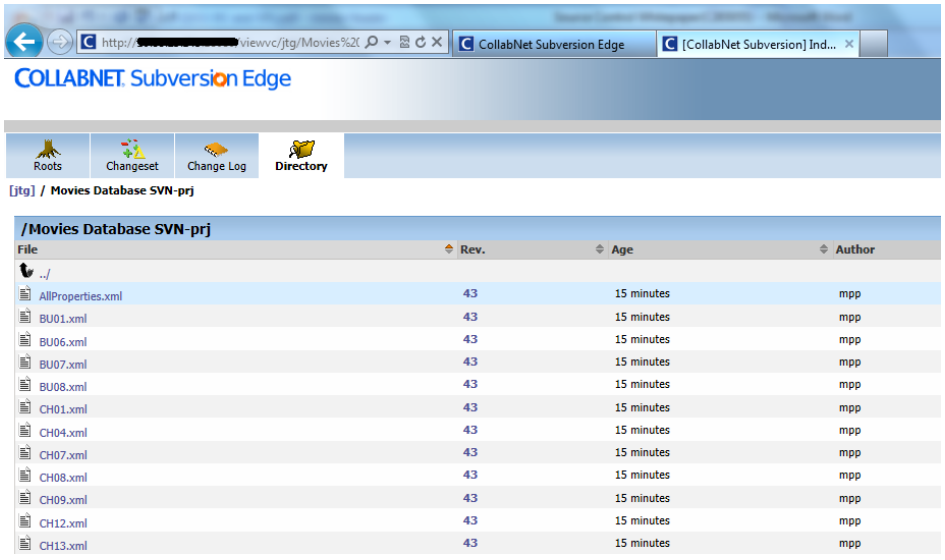


If you are challenged for login credentials it is usually aligned with your network login details, please speak to your Subversion administrator for further details.

QV11 SR1 does not save any username and password for Subversion. Instead it is the specified Subversion client that saves it (encrypted) in C:\Users\xxx\AppData\Roaming\Subversion. This path is shared among all Subversion clients, including Tortoise.

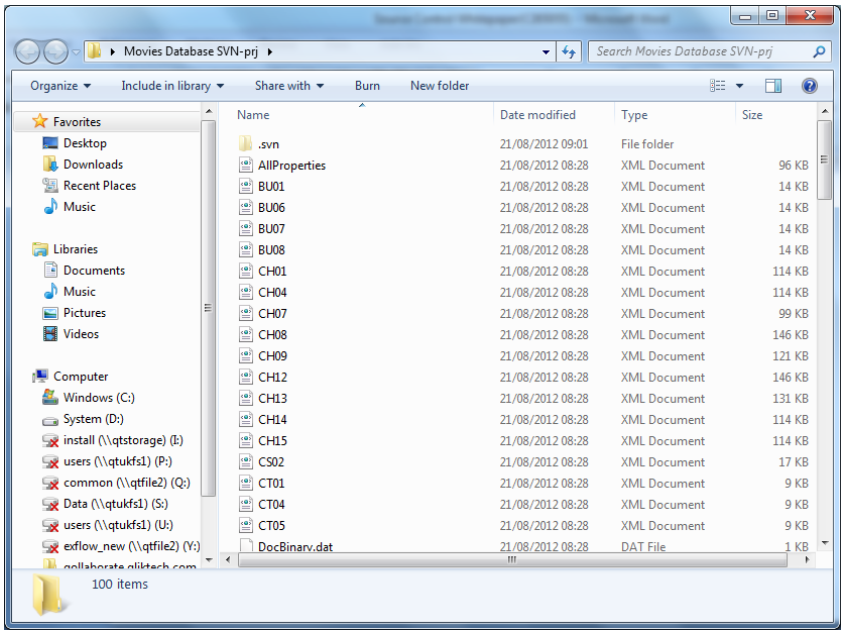
After selecting the URL, click **OK** to complete the check-in. You are returned to the QlikView document you were working on.

If you have direct access to the Subversion server, you can browse to the folder you just added your application to. You will see all the objects, author and any check-in comments made.



You can close this for now.

The local 'PRJ' folder is stored in the same directory (next to) the QVW you have checked in. During the setup of your local Subversion client you should have chosen a place to store and edit your applications for ease of administration and governance. Navigate there to see the divided project files.



You can close this for now.

Making Changes to QlikView Files Added to Source Control Using Subversion (Single Developer)

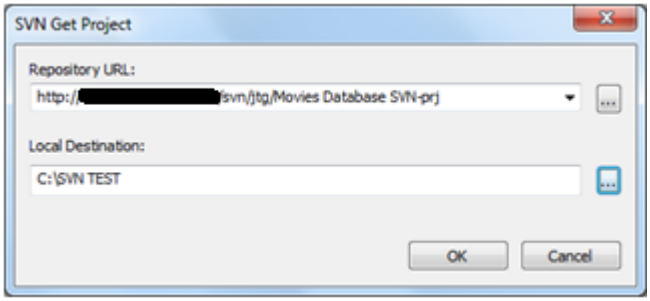
The following section describes the process when there is only one developer accessing the source control at any one time.

As a developer, you may need to edit a document you previously obtained from source control. It may even be a document you added to source control yourself. To do this, you would simply open QlikView and either select your local copy of the document from the latest documents list, or browse to it in Windows Explorer and open it. When the document opens, it is important that you get the latest version to ensure you will make your changes on current checked-in version. You can do this by selecting **File/Source Control/Get Latest Version**. If you do not have the latest version of the code on disk, QlikView refuses to check in the changes with a warning that you must perform a get latest first. This is consistent with the default behavior of Subversion.

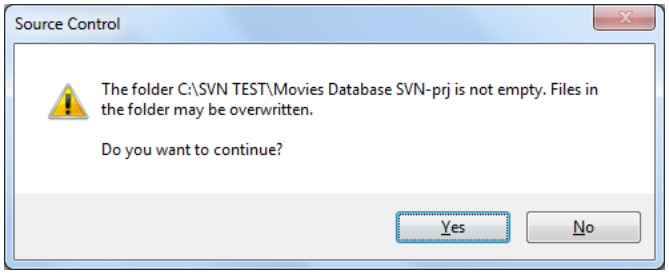
This is good practice in both single and multi developer environments.

To edit a document that has been added to Source Control for the first time, you need to obtain a copy of the source file from the server. You then apply changes to your local copy, until you are ready to check in.

In QlikView, this can be done by selecting **File/Source Control/Get Project from Source Control**. Select/type the correct URL for the server and project as well as your local folder and click **OK**.



Note: If you already have this application in your local directory you are prompted to accept an overwrite which destroys any changes not checked in. If you wish to keep these you should cancel and open the document from your local folder to complete your edits before checking them in. If you wish to start a fresh edit on the current version click **Yes** to continue.



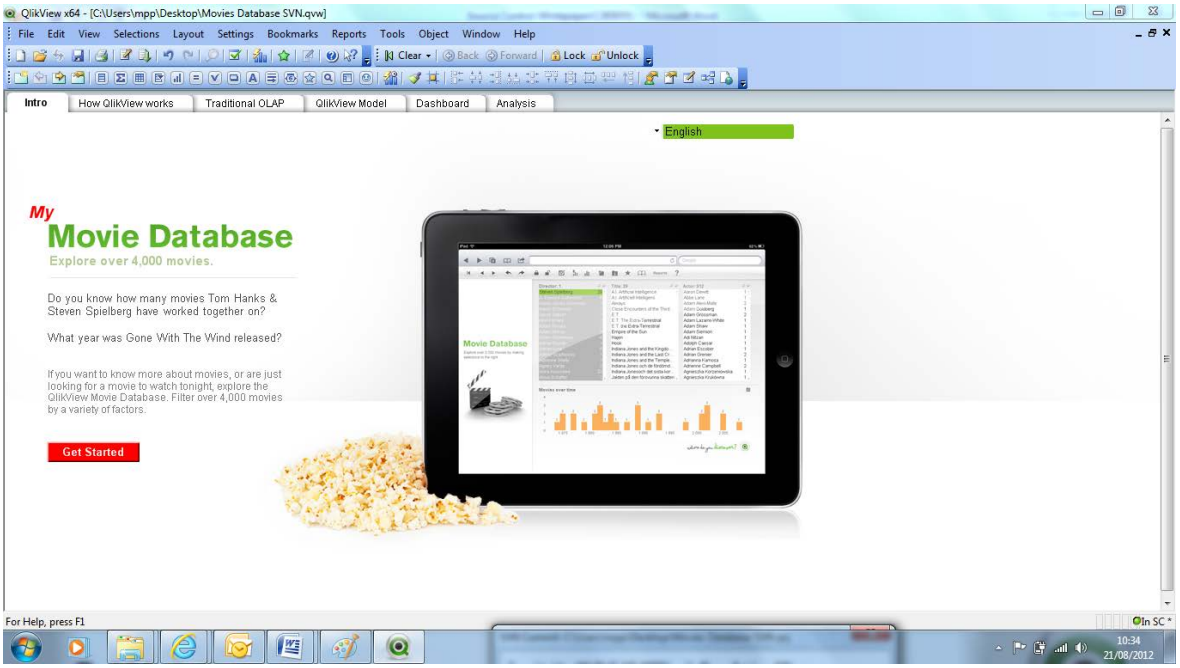
Opening the QlikView document this way means you will open it with no data, which is how it is stored.

To work with data, reload the script. Now you can continue working on the latest version and make any necessary changes.

Test Source Control Check-In Using Subversion

Perform the following steps to test checking in an updated document:

1. Change an object - in this example turning the *Get Started* button to red in color and removing the background image on sheet 1.
2. Delete an object – in this example removing the label *Language* next to the language selection field on sheet 1.
3. Create an object – in this example adding a text box with a new title ‘My’ on sheet 1.

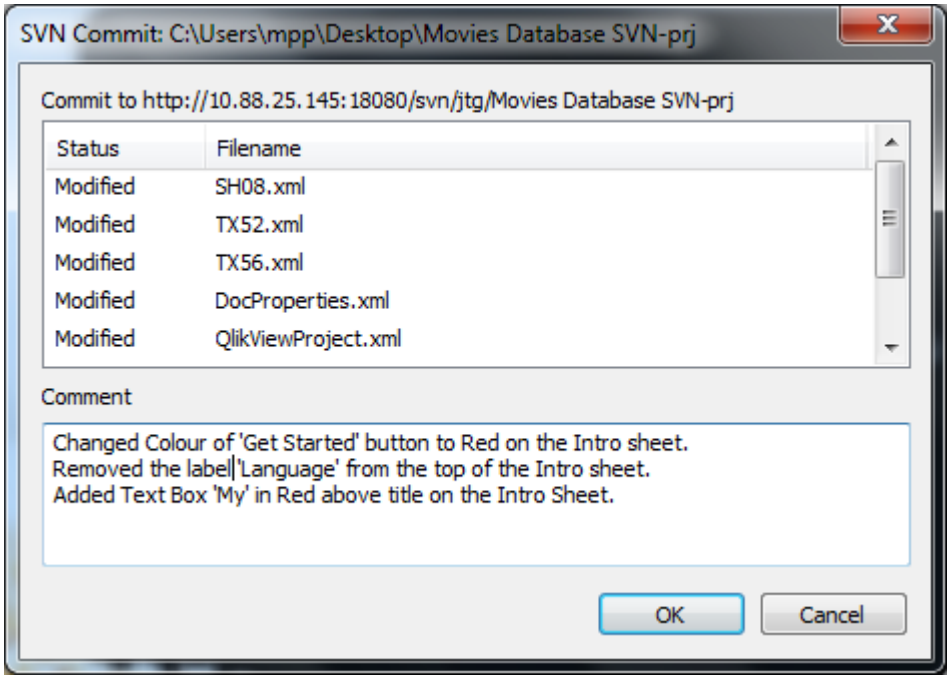


The local result of the example

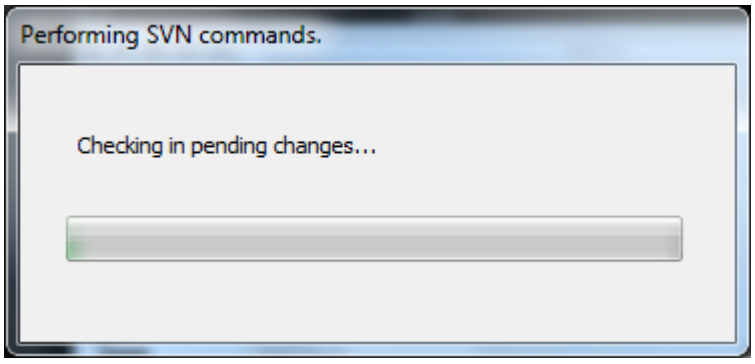
When you are happy with your changes, you want to check the changes into source control, ready for the next reload and distribution.

Select **File/Source Control/Check In Pending Changes**.

Unlike TFS, all change types are handled in this one check-in dialog and comments so be sure to add sufficient detail as in the following picture.



When you are happy with your comments, click **OK**, and you will see the following screen.



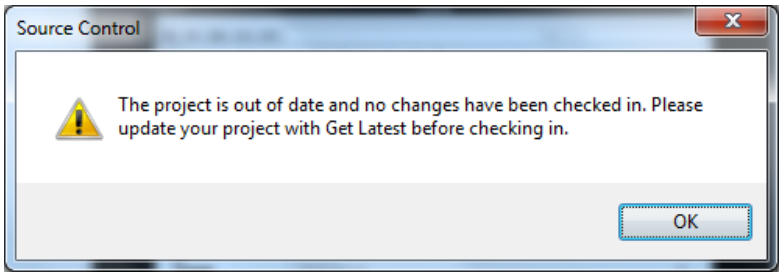
When the check-in is complete this dialog disappears and you are presented with your edited local QVW that has been merged to the server copy. You have completed an example of adding and editing a QlikView document using Subversion source control!

Making Changes to QlikView Files Added to Source Control Using Subversion (Multiple Developer)

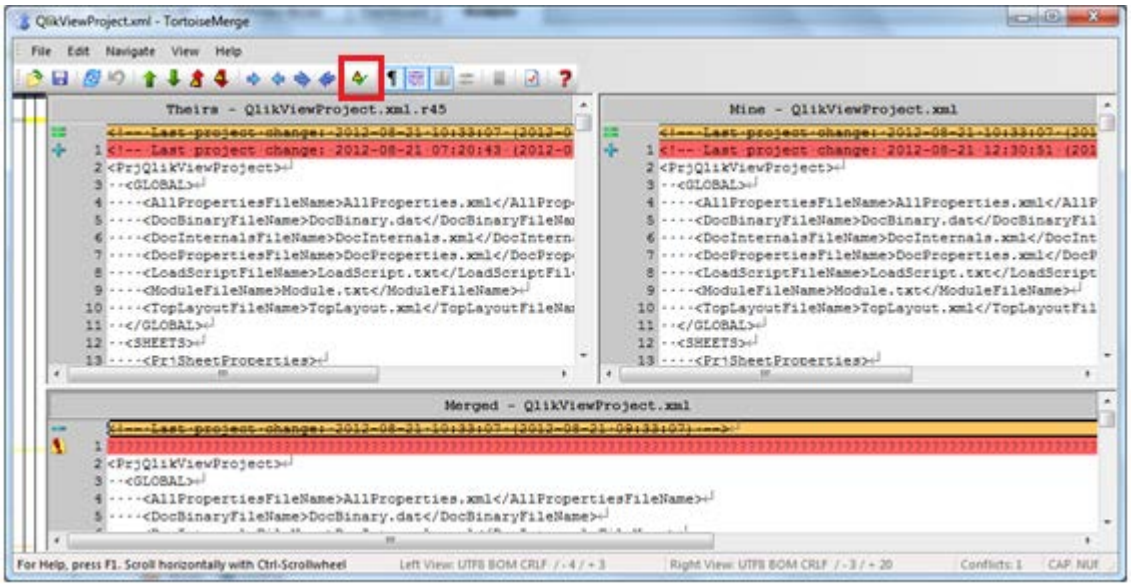
The following section describes the differences to the process when there is more than one developer accessing the source control at any one time. Again, it is assumed that you have a working knowledge of TFS and general best practices associated with source control systems.

In a multi developer process it is essential that you update your copy with changes from other developers before your changes are checked in. The default behavior of Subversion checks for this and forces this to take place.

If someone else has checked in changes to the document after you started making changes to your local copy, you need to merge those changes into your document. The following dialog is displayed.

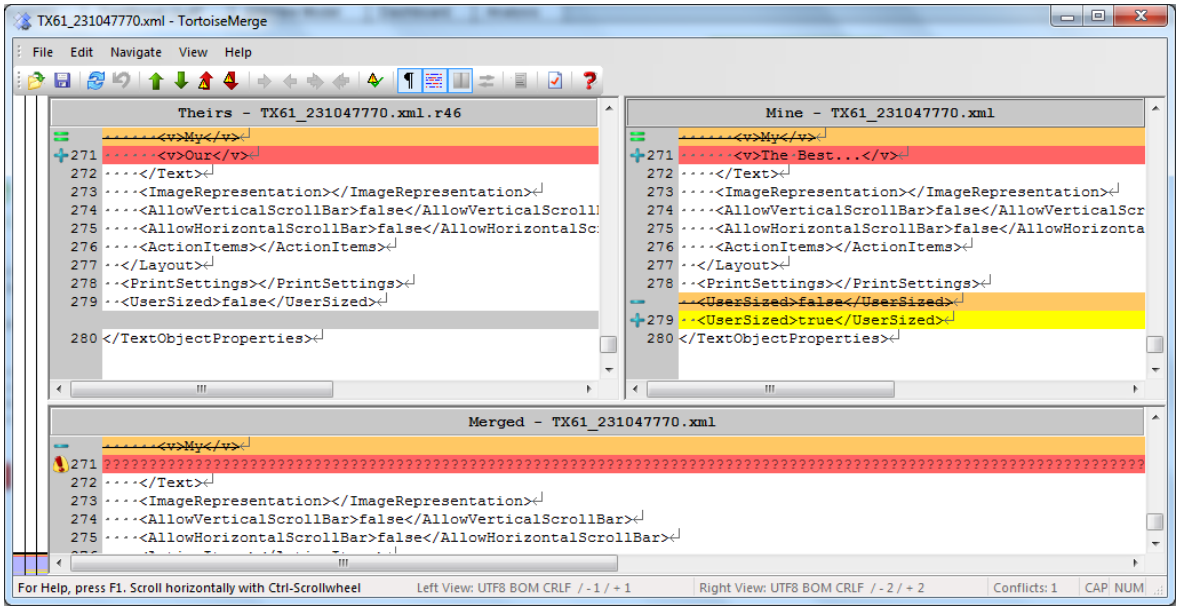


By selecting **File/Source Control/Get Latest Version** you start the conflict/merge resolution process and your merge tool should be presented to you. The merge tool available is dependent on your chosen installation so guidance from your Subversion administrator may be necessary. In the following examples, Tortoise is used.

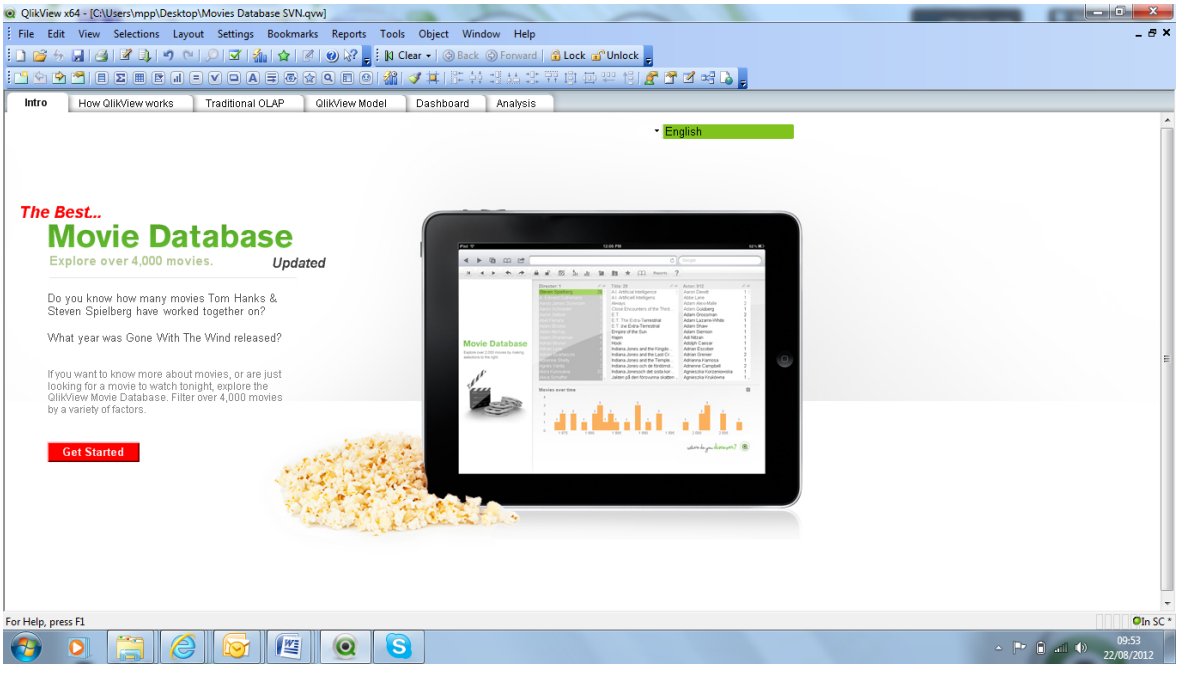


The screen presents you with the server copy which is called ‘Theirs, your copy (Mine) and the merged copy. Using the menu bar to skip through the changes you need to select which to keep. If it is clear the changes in theirs are different to yours, you can accept the merged copy and finish this by clicking the menu button highlighted in the red square on the preceding screen scrape. Close the merge tool and save if you have not already done so. You can then check in your changes as previously described.

If, however, both you and another developer have changed the same object, a real conflict occurs. In the following example, at xml code line 271, both myself, and my colleague changed the same text box. He changed ‘My’ to ‘Our’ and I had been advised to change it to ‘The Best…’. Before completing the merge we agreed that my text was the correct text to be added and in this case I right-clicked the highlighted text in the ‘Mine’ screen and selected to take my changes only.



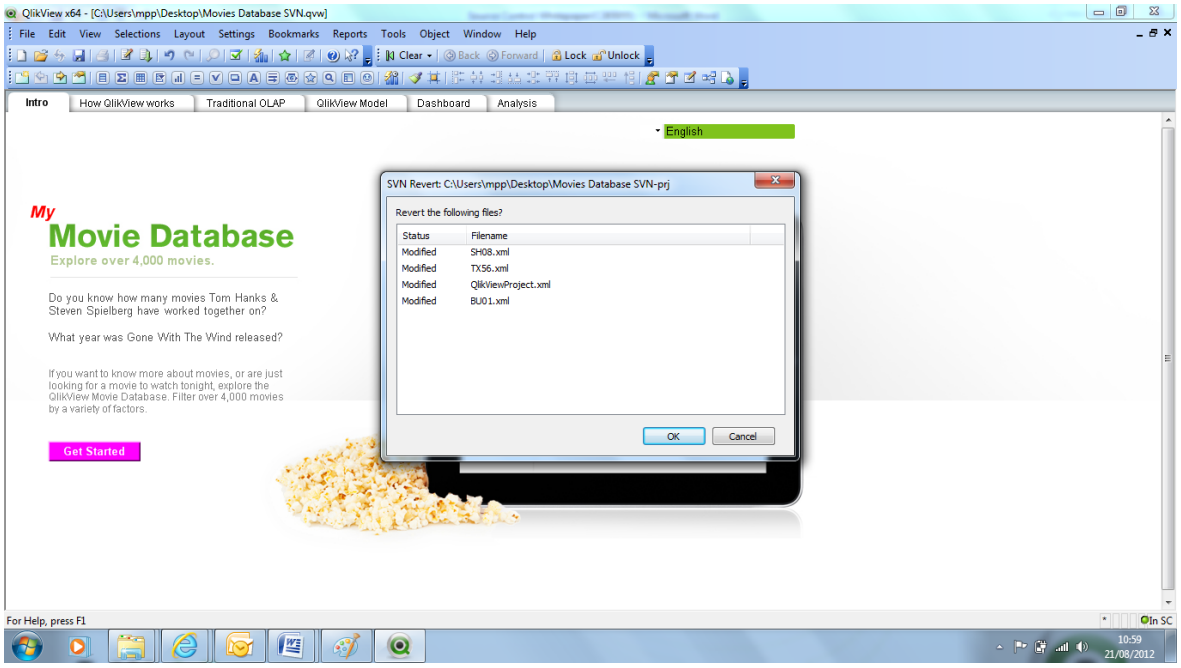
After completing the rest of the merge the latest version presented correctly to me as follows.



To complete the process I then checked in my changes.

If you experience a message stating ‘SVN Binary Conflict you need to discuss with the system owner as it is likely that important security information (e.g. a hidden script) has changed, and selecting the correct version needs appropriate care. When there are no conflicts left, the merge tool can be closed and your open QVW is refreshed containing all the changes you elected to merge. Naturally, checking in is to be coordinated and the preceding process should only be performed by one person at a time. If you decide to discard your changes and not check them in, select **File/Source Control/Undo Pending Changes**.

Undo Pending Changes Example



Prior to check-in, you can remove all changes (all or nothing) and start again. In the example application, the ‘Get Started’ button on my local copy was changed to pink, and the document was saved. By selecting **File/Source Control/Undo Pending Changes**, this change was disregarded. A dialog box is revealed to show what will be undone and after clicking **OK** the reversal takes place.

Manually Editing Source Control Files

You may want to make changes to the object XML code in TFS, but at this point in time we recommend that you do not do this. In this release you need to make your changes via the QlikView application.

QlikView