Qlik Deployment Framework

Integrated Development Environment (IDE) with Qlik

March, 2017

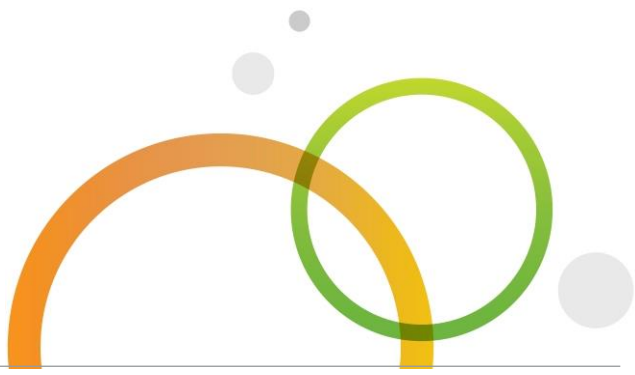


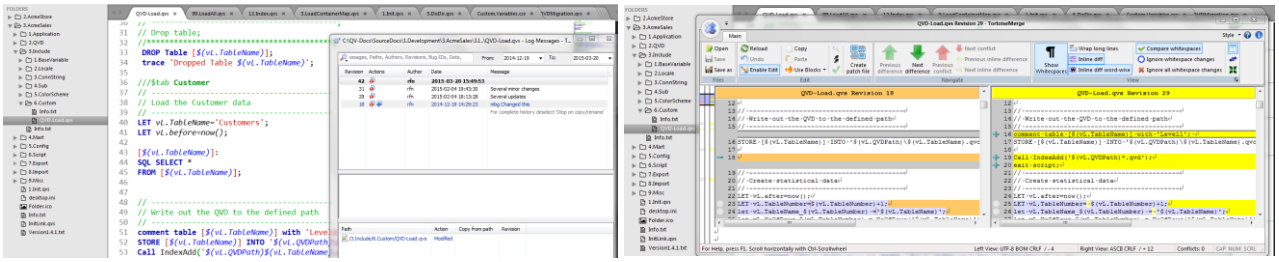
Table of Contents

Version 1

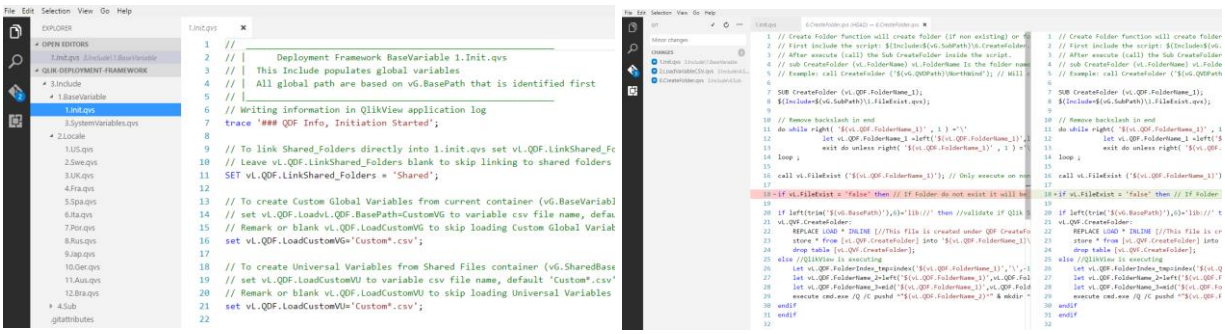
Qlik IDE Development environment

Overview

An integrated development environment (IDE) is a programming environment that has been packaged as an application program. The IDE may be a standalone application or may be included as part of one or more existing and compatible applications. The suggested environment in this document blends several components together into a complete development experience including version control and multi-development support.



Sublime text editor and Subversion IDE environment



Visual Studio Core and GitHub IDE environment

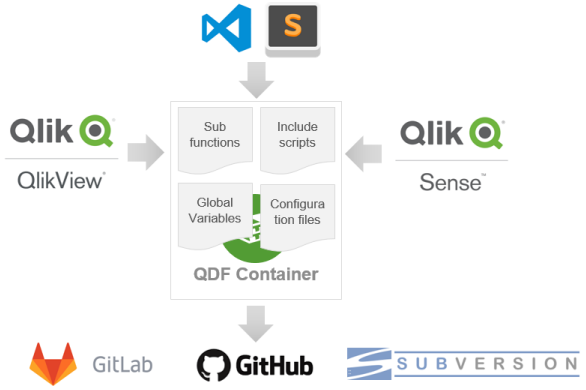
This document describes how to prepare and setup IDE for Qlik Sense and/or QlikView. The IDE environment and versioning are based on Qlik Include scripts (qvs) and data (csv) files to store variables, these are reusable across multiple applications, platforms and environments. The developer can create reusable sub functions and script snippets with Qlik Include scripts and customize *global variables* reused across applications and between Qlik products, these variables can be used to store common formulas, expressions, color scheme, descriptive text and more. *Qlik Deployment Framework* (QDF) reusable code structure (containers) is used as the interface between Qlik and the IDE.

Qlik IDE development is done using an external script editor, *Sublime Text* or *Visual Studio Core* are recommended as these editors have integration with Qlik.

- **Sublime use *InQlik-Tools* module** for syntax high lighting is more suitable when using Subversion as there are modules available for this.
- **Visual Studio Core use *Qlik* module** for syntax high lighting is more suitable when using GitHub as its integrated into the IDE.

QlikView Deployment Framework (QDF)

To create a structured development environment, we strongly recommend using QlikView Deployment Framework (QDF) available for free under Qlik Community. In short QDF is based on a *resource container architecture* where each container holds an isolated code base related only to the purpose of that container. No hard references exist outside a container so *Qlik script* files can move across containers without relations breaking. QDF containers also stores variables reusable by applications inside each container, this is called global variables. There is also a common *Shared Folders* container that stores code and global variables reusable by everyone. All containers have the same structure and there are placeholders to store developed code, each place have a global variable connected to it. These variables should be referenced in the *Qlik script*. Read more on QDF at Qlik Community.



QlikView Deployment Framework resource containers are the glue in the IDE environment

Container Include folder structure containing Qlik scripts

3.Include		*Folder where Qlik Scripts are resided segmented into sub folders storing different types of code.*
$(vG.BaseVariablePath)	*1.BaseVariable*	*Contains framework initiation script (1.Init.qvs), also contains custom variables in Custom.Variables.csv*
$(vG.LocalePath)	*2.Locale*	*Locale for different regions, used for easy migration between regions*
$(vG.ConnStringPath)	*3.ConnString*	*Stores connection strings to data sources (QlikView only)*
$(vG.SubPath)	*4.Sub*	*Store for sub functions, this is a way to reuse code between applications*
$(vG.ColorSchemePath)	*5.ColorScheme*	*Place holder for Standard Color Schemes (QlikView only)*
$(vG.CustomPath)	*6.Custom*	*Store for custom include scripts*

To use a Qlik Script stored under *3.Include\6.Custom* use the Include statement in Qlik:

```
$(Include=$(vG.CustomPath)\MyCode.qvs);
```

To use a personal function stored under *3.Include\4.Sub*

```
$(Include=$(vG.SubPath)\MyFunction.qvs);
```

To use a pre-populated locale file:

```
$(Include=$(vG.LocalePath)\1.US.qvs);
```

Version control and Containers

As Containers are Isolated structures with no hard references to the outside they are ideal to use as repositories in version control systems. The example systems in this document are Subversion (SVN) and GitHub, As the development using QDF is text based most version control systems should be possible to use, for example TFS can be integrated into Sublime or VSC and utilized in a similar way as described in the article.

Visual Studio Code with GitHub

Visual Studio Code (VSC) is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. VSC have built in support for GitHub which makes it perfect for development using GitHub as the version control system.

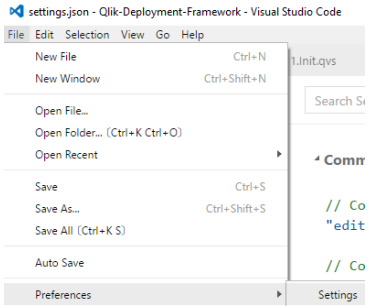
Setup Visual Studio Code

First we need to download VSC from the official web site the site is also rich documentation how VSC works. Run the Installation package and open the editor.

File Encoding tweak

After install you should change the file encoding to `windows1252` especially if you are in a non-English speaking country as special charters might be incorrect.

1. First open the VSC settings page as seen below



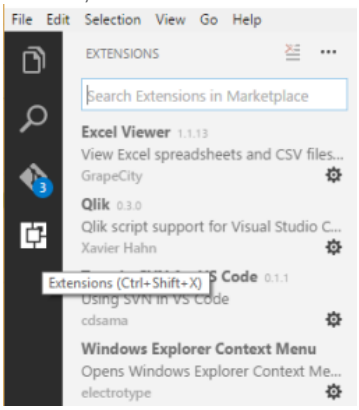
2. In your user settings -inside the brackets- add: `"files.encoding": "windows1252"`

```
{
    "files.encoding": "windows1252"
}
```

3. Save and restart editor

VSC Extensions

After VSC setup the appropriate modules need to be added, just click on the extensions button (square) to the left, here we can search and install community driven extensions.



- **Qlik** is an extension that enables syntax highlighting for Qlik scripts
- **Excel Viewer** is an extension that previews csv and xls files as tables, this one is nice to have when adding custom variables into csv files

- **Windows Explorer Context Menu** if using *Subversion* and *TortoiseSVN* this extension lets you use the explorer context direct in VSC.

GitHub Account

Before doing anything, a GitHub account owning the project need to be created. You need a private repository to lock for outside view. VSC has integrated support for GitHub and will automatically identify GitHub repository when using the official *GitHub Desktop*

- Create account in github.com. You need a private repository to lock for outside view
- Download and install *GitHub Desktop* from github.com.
- Add your account first time use.

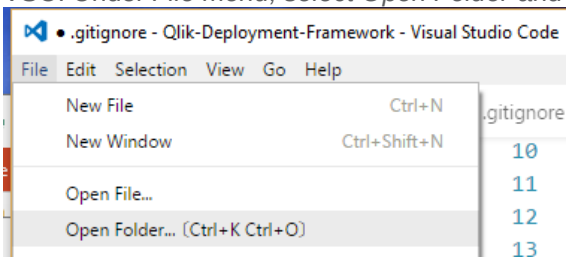
.gitignore

Gitignore is a settings file where you add things that should not be versioned. In our case QVD, QVF, Log, .index (QDF internal) and QVW files. Important is that this file is updated before adding a container into GitHub.in newer QDF version the ignore file have already been added.

```
11  # Windows Installer files
12  *.cab
13  *.msi
14  *.msm
15  *.msp
16  *.qvd
17  *.qvf
18  *.qvw
19  *.log
20  *.index
```

Add resource container into VSC

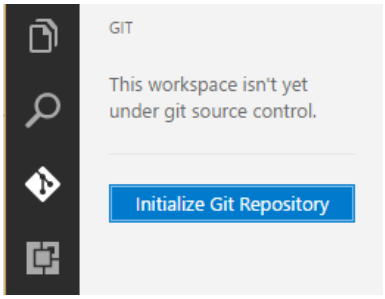
Now when the first resource container been merge into a GitHub repository we can add the container into VSC. Under *File menu, select Open Folder* and point to the Resource container.

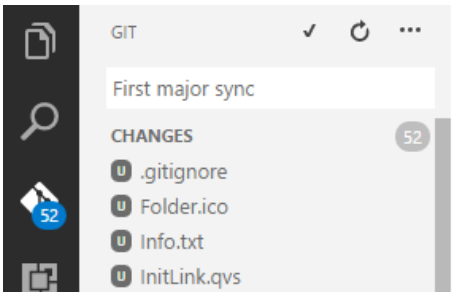


If you have several containers easy switch between them using *File menu, Open Recent*

Add container to GitHub

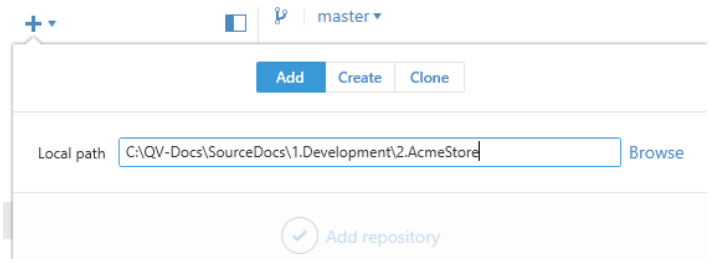
After the resource container been added to VSC and GitHub Client been installed (account registered) and gitignore is avalible in the container it's time to create a repository. Just select the version control symbol in the left meny and press *Initiate Git Repository* and type information message in the text box. Last press the Commit All symbol (Ctrl/Enter).



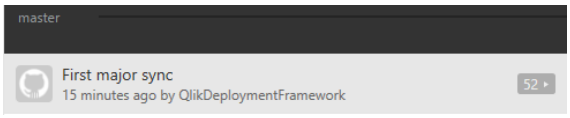


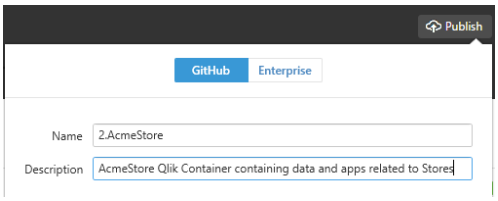
Sync to remote GitHub repository

Last we need to sync the container repository to the remote *GitHub* repository. For this we need to open *GitHub Desktop* and add a repository using the plus sign pointing to our container location.



Press Add repository, as the first sync allready been done we should get the message First major sync and the amounts of minutes ago and the user. Press Publish in the top right corner to publish the repository to the remote *GitHub.*





Sublime Text 3

Sublime Text 3 is a very competent and modular editor. Sublime have modules for Qlik script, TFS, SVN and many other systems. You can say that Sublime Text is perfect for Qlik Script development. In this article we explain how to use Sublime as Qlik script (qvs) editor, launch point for qvw files and used for version control. Sublime is try and buy so if you start developing professional using sublime please pay the $70. Installation instructions of Sublime and surrounding components are found under Environmental Installation section.

Qlik script syntax highlighting

To get Qlik script syntax highlighting and other cool Qlik development functionality into Sublime one of our partners have created the InQlik-Tools Sublime module. Read more under Environmental Installation section.

Sublime Installation and configuration

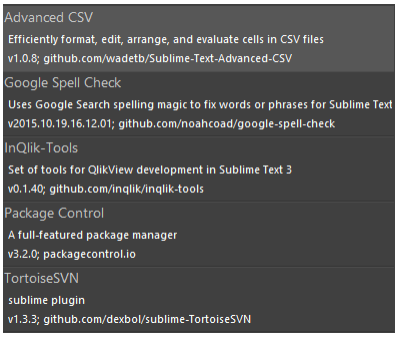
Below is a step by step guide how to install Sublime IDE environment.

Sublime Install

Download and install Sublime Text 3. Open sublime and add the modules presented below. Sublime Text need to be installed and setup for every developer

Sublime module Installation

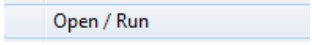
Install the Sublime modules described below



- **Package Control**
  This will enable the search and installation module for Sublime this is recommend for easy install. Always install Package Control first and install the other modules with help of Package Control, read: How to install Package Control

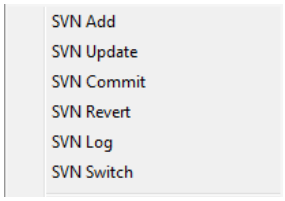
- **InQlik-Tools**
  This is an open source module that enables syntax highlighting for Qlik scripts and logs and additional enhancements. Search and add InQlik-Tools from Package Controls Install menu.

- **SideBarEnhancements**
  Adds functionality to the sidebar needed to open QlikView applications inside Sublime. Search and add SideBarEnhancements from Package Controls Install menu. Use this module to start files (qvw) directly from the sidebar, right click on the file and select Open/Run.



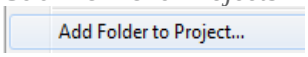
- **Advanced CSV**
  A nice module to get better formatting in csv files. Used to get a better interface when editing *$(vG.BaseVariablePath)*.variables.csv* variable (expression) files like Custom.Variables.csv

- **TortoiseSVN** If using Sublime TortoiseSVN module adds version control functions into the *Sublime SideBar*. Search and add TortoiseSVN from Package Controls Install menu. Using this module you can commit Qlik Scripts into SVN directly from the SideBar. Remember that *TortoiseSVN* client needs to be installed before this functionality starts working.

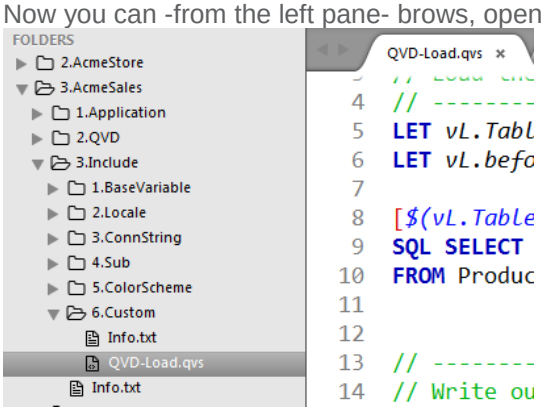


Add QlikView Deployment Framework project to the SideBar

For good project overview add your development containers to the Sublime sidebar. This is done under Sublime menu *Projects -> Add Folder to Project…*



Now you can -from the left pane- brows, open and version control content inside the development containers.

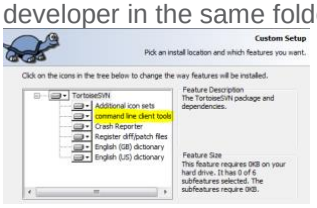


Subversion (SVN)

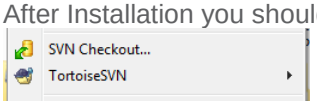
Subversion is an open source version control system. Subversion exists to be universally recognized and adopted as an open-source, centralized version control system characterized by its reliability as a safe-haven for valuable data. The below instructions is using the Windows based Visual SVN bundling.

TortoiseSVN client

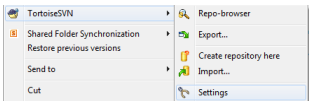
TortoiseSVN is the Subversion client that works as the underlying Windows component. *TortoiseSVN* must be installed on every client. Download *TortoiseSVN* and run the Install program, check in *command line client tools* as these are used by QlikView Developer. *TortoiseSVN* must be installed and setup for every developer in the same folder structure, recommendation is to use default location.



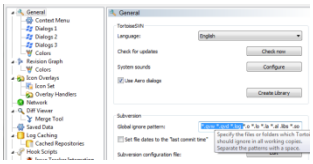
After Installation you should have several new *Window Icons* when right clicking on Folders.



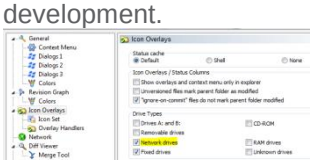
First we should create an ignore rule for QlikView qvw and qvd files as they can get really big, Version control of qvw will be handled inside QlikView instead. Right click on any folder in explorer to get the *TortoiseSVN* menu after select *TortoiseSVN/Settings*



In the first General tab add **.txt *.qvw *.qvd *.log* to Subversion Global ignore pattern and press apply.



Select *Icon Overlays* menu and check in the box Network Drives as we are using a common share for development.



Press the apply button and exit the menu.

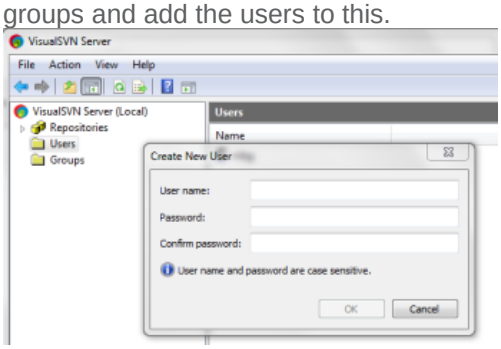
Visual SVN Server Overview

Visual SVN Server comes in two versions one that is free to use commercially. And an extended version that has Windows SSO integration and support, this version costs $950. There are several other SVN distributions as well, chose depending on customer needs. SVN is installed on a central server this could be the same as your Qlik development environment. Recommendation is to create a DNS C-Name for the version control repository instead of the server name, so that it can easily be moved.

VisualSVNServer Setup

Download and Install Visual SVN Server. During install use the port 8443 so it won't conflict with Qlik.

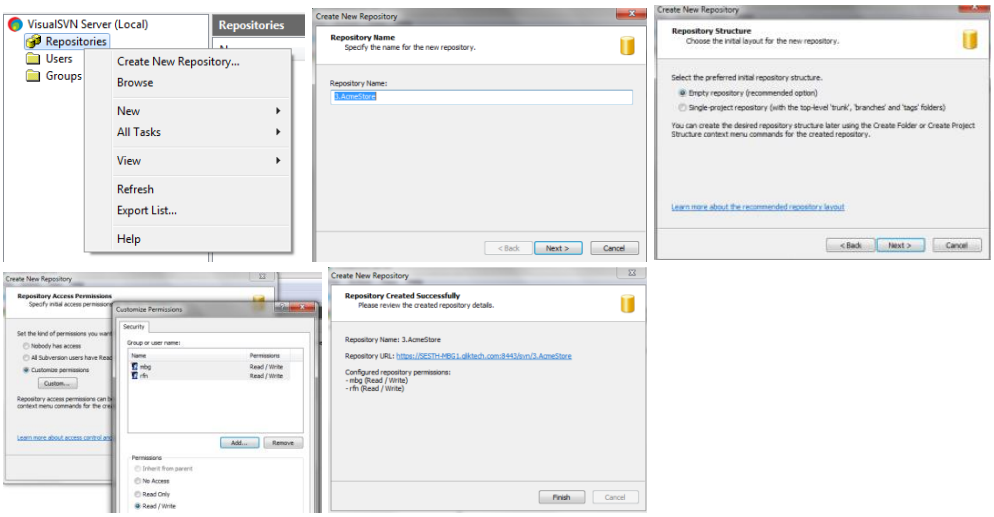
Start the *VisualSVN Server Manager* and create as many users (developers) needed, create development groups and add the users to this.



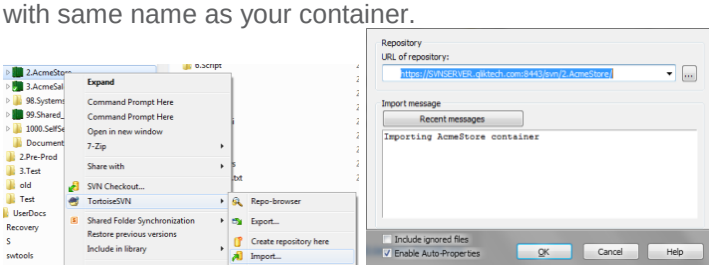
Remember to back up the central SVN database (D:\Repositories or similar path)

Creating SVN repository's

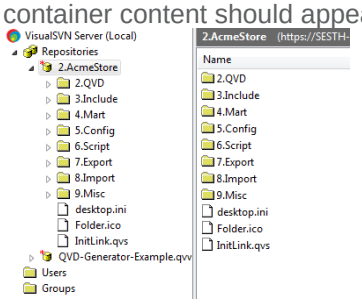
For each development container we need to add a SVN repository, one repository for each container. First we'll create the *2.AcmeStore* container repository using the *VisualSVN Server Manager*, after we'll add authorized users or groups to the repository. Remember to write down the URL as this will be used several times also document this in your development guide.



The repository is used to version control scripts and configuration files in *2.AcmeStore* container. We will add content into the repository by selecting *2.AcmeStore* container right click and select *TortoiseSVN Import* function. An Import box should appear, type the repository URL that was given earlier, the URL should end with same name as your container.



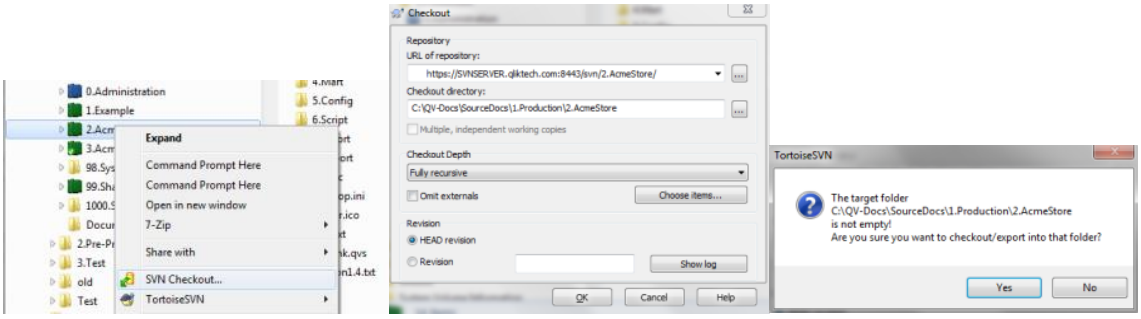
Go into the *VisualSVN Server Manager* and the *2.AcmeStore* repository and press F5 to refresh, the container content should appear. From here you can remove folder that does not need versioning.



Check out SVN containers

Now we should check out *2.AcmeStore* content so that we can start developing, do this from the same location as the files you imported earlier, this is different than traditional development where every developer have its own repository and check in the code from there. In Qlik development we want to run the applications using the code just developed, this would be difficult if code and app are stored different locations.

In an explorer window select *2.AcmeStore*, right click and select *SVN Check Out.* Check that you have the correct repository URL and checkout directory, press OK. An information box will state that target folder in not empty, select yes. Our container will be overwritten by the repository content (that is exactly the same). Now we should have green check-box icons under the *AcmeStore* container.



Now we are ready for development, commit and diff Qlik scripts directly from Subversion or from explorer using the *TortoiseSVN* menus.

Note: If for some reason the check-out fails of you want to remove repository content after check-out remove the *.svn* folder stored directly under the container, apply your modifications and redo the check-out.

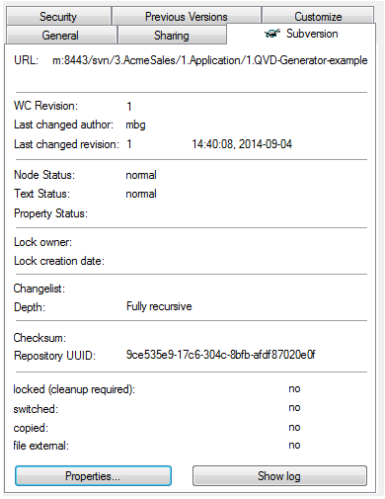
Keyword Substitution with TortoiseSVN

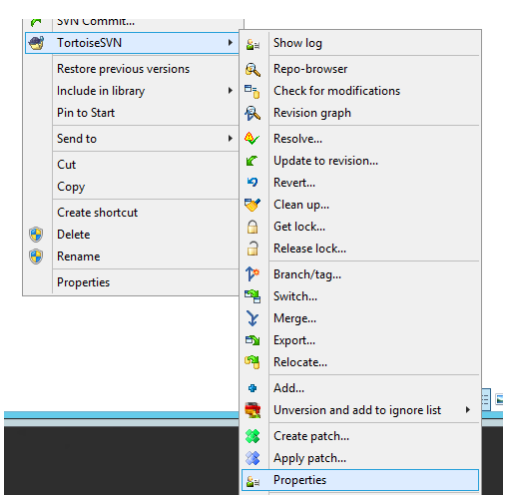
A cool feature of Subversion is that you can get it to update comments in your script automatically after commit to Subversion. Below is an example of this feature in Sublime.

```
11  // Subversion information
12  /*=============================================================================================
13  Repository path:    $HeadURL: https://sesth-mbg1.qliktech.com:8443/svn/3.AcmeSales/3.Include/1.BaseVariable/4.Custom.qvs $
14  Last committed:     $Revision: 12 $
15  Last changed by:    $Author: rfn $
16  Last changed date:  $Date: 2014-12-18 09:26:43 +0100 (to, 18 dec 2014) $
17  ID:                 $Id: 4.Custom.qvs 12 2014-12-18 08:26:43Z rfn $
18  =============================================================================================*/
```

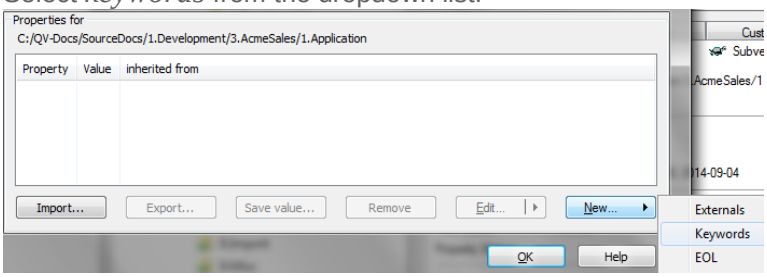
Follow these steps to activate keyword substitution on each SVN repository:

1. Right click on the root folder of your working copy using Windows Explorer.
2. Select *properties* (Note: this is not the TortoiseSVN properties option!)
3. Select the *subversion* tab. Or use TortoiseSVN Properties menu.

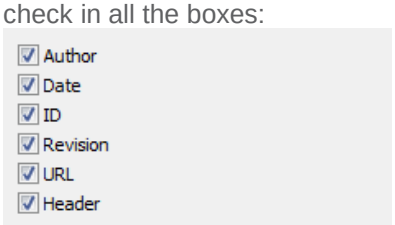




4. Click the *properties* button.
5. In the new *properties* dialog, select the 'New' button.
6. Select *keywords* from the dropdown list.



7. check in all the boxes:



8. If you want to apply keyword substitution to all sub files and folders, tick the box titled 'Apply property recursively'
9. Click *OK*

10. In the qvs scripts add the lines:

```
/*============================================================================================

Repository path:    $HeadURL$
Last committed:     $Revision$
Last changed by:    $Author$
Last changed date:  $Date$
ID:             $Id$
============================================================================================*/
```

Additional notes

- Programs described in the document Sublime, Visual Studio Code, Github, Subversion and all modules are not Qlik products and thereby not supported by Qlik
- Version control is a way for developers to revoke changes it is not a replacement for traditional backups.
- Use a common share for development, do not check-out QlikView applications and code to a local drive as done in traditional development.
- Do not use version control on physical qvw and qvd files as they can grow big. The objects in a qvw file are versioned using QlikView's built in system (out of scope in this article).
- You can also use TFS instead of Subversion by replacing the Sublime modules.