



TaraVault™ | Source Code Hosting for Spira™

User Manual

Inflectra Corporation

Date: October 5th, 2015



Contents

1. Introduction.....	1
2. Activating TaraVault.....	2
3. Provisioning Projects & Users	3
4. Using Subversion	8
5. Using Git.....	19
Legal Notices	30

1. Introduction

TaraVault™ is the secure source code and file hosting service from Inflectra that allows you to host your source code and other assets in our secure cloud, integrated with our Spira™ application lifecycle management system.

This guide outlines how to activate and use TaraVault™ with a cloud-based instance of SpiraPlan® or SpiraTeam®. The guide covers the general activation process in addition to specific sections dedicated to the two supported SCM technologies – *Git* or *Subversion*.

This guide assumes that the reader is familiar with both SpiraPlan/SpiraTeam and the appropriate SCM platform (Git and/or Subversion). For information regarding how to use SpiraPlan/Team, please refer to the *SpiraPlan User Manual* or the *SpiraTeam User Manual*.

2. Activating TaraVault

To get started, log into your existing SpiraPlan or SpiraTeam instance (hereafter referred to as Spira) with a system administrator account and go to the main Administration menu:

The screenshot shows the Spira Administration interface. On the left is a sidebar menu with categories: Projects, Users, Custom Properties, Incidents, Requirements, Planning, Notifications, Documents, System, and Integration. The 'Integration' category is expanded, and 'TaraVault' is highlighted with a red box. Other options under Integration include Data Synchronization, Test Automation, and Add-Ons & Downloads. The main content area is titled 'View / Edit Projects' and displays a table of projects. The table has columns for Project Name, Project Group, Creation Date, Active status, ID, and Operations. The 'Sample Application Two' project is highlighted in yellow.

Project Name ▲▼	Project Group ▲▼	Creation Date ▲▼	Active ▲▼	ID ▲▼	Operations
Legacy Application	External Projects	1-Dec-2005	No	PR000004	> Select Edit Copy Delete
Library Information System	Internal Projects	1-Dec-2005	Yes	PR000001	> Select Edit Copy Delete
Sample Application One	Internal Projects	1-Dec-2005	Yes	PR000002	> Select Edit Copy Delete
Sample Application Two	External Projects	1-Dec-2005	Yes	PR000003	> Select Edit Copy Delete

You should see the 'TaraVault' menu entry under the "Integration" heading. If you don't see this option then you might either be self-hosted or running SpiraTest. If you are running SpiraTest, please contact Inflectra customer services to get your account upgraded to SpiraTeam.

Click on this TaraVault link to bring up the TaraVault administration home page:

The screenshot shows the 'TaraVault Configuration' page for 'Sample Application Two'. It includes a link to 'Change Project'. The page text states: 'TaraVault is the hosted source code repository and software configuration management (SCM) system provided by Inflectra. Projects in SpiraTeam can be enabled in TaraVault for storage of project source code files and revisions. TaraVault is provisioned based on a specific number of active users across all projects. You can change which users are active for source code management in the View / Edit Users screen.' A yellow message box says: 'Your TaraVault instance is not active. Once activated you will need to create a TaraVault project for each SpiraTeam project that you wish to have a version control repository for. You may activate TaraVault by clicking the button below.' Below the message is an 'Activate TaraVault' button.

This screen lets you know that your instance of Spira does not yet have an active instance of TaraVault. This is normal and you should now click on the 'Activate TaraVault' button to activate TaraVault. Once this is done, the screen will display:

The screenshot shows the 'TaraVault Configuration' page with settings for 'Sample Application Two'. It includes a link to 'Change Project'. The page text is the same as the previous screenshot. Below the text are two sections: 'TaraVault Global Settings' and 'TaraVault Project Settings'. The 'Global Settings' section shows: 'Available TaraVault Users: 5', 'Active TaraVault Users: 1 (View List)', and 'TaraVault Account Name: inflectratest3 (117326)'. The 'Project Settings' section shows: 'TaraVault Project Active: No', 'TaraVault Project Name: [text input]', 'TaraVault Project Type: GIT (dropdown)', and 'Client Connection: [text input]'. At the bottom of the 'Project Settings' section are 'Activate' and 'Clear Cache' buttons.

This provides you information on the name and ID of your TaraVault account as well as an indication as to how many source code users your subscription allows. If you need to purchase additional users, just contact Inflectra customer services who will be happy to help you out.

Now that your instance is active, you will need to provision individual projects and also activate the Spira users that will be allowed to commit code or files into the TaraVault repositories. Note: *All users in Spira will be able to view the files in TaraVault without needing a separate TaraVault license.*

3. Provisioning Projects & Users

Once you have activated TaraVault for your instance of Spira, you can now begin to provision specific projects and assign users.

3.1. Provisioning Projects

To provision a new project with TaraVault, go to the main Spira Administration page and select the appropriate project and then go to the TaraVault administration page. For example, in the screenshot below we have selected the 'Library Information System' sample project:

TaraVault Configuration | Library Information System ([Change Project](#))

TaraVault is the hosted source code repository and software configuration management (SCM) system provided by Inflectra. Projects in SpiraTeam can be enabled in TaraVault for storage of project source code files and revisions. TaraVault is provisioned for each project. You can change which users are active for source code management in the [View / Edit Users](#) screen.

TaraVault Global Settings

Available TaraVault Users: 5
Active TaraVault Users: 1 ([View List](#))
TaraVault Account Name: inflectratest3 (117326)

TaraVault Project Settings

TaraVault Project Active: **No**
TaraVault Project Name:
TaraVault Project Type:
Client Connection:

[▶ Activate](#) [▶ Clear Cache](#)

To provision this project with TaraVault, you need to choose the following:

- **Project Name** – this is the name of the project in TaraVault. This name is incorporated into the connection URL so it should be alphanumeric only.
- **Project Type** – this is the type of SCM repository you wish to use. Currently the choices are between Subversion and Git. We discuss the differences between Subversion (also known as SVN) and Git in sections 4 and 5 respectively.
Note: once you choose the repository type for a project it cannot be changed without deleting the entire repository, so make sure you understand the differences between the two technologies beforehand.

For example we will choose 'libraryinformationsystem' as the project name and 'SVN' (Subversion) as the project type. Once you are happy with your choice, click 'Activate' to enable this project for TaraVault:

TaraVault Configuration | Library Information System [\(Change Project\)](#)

TaraVault is the hosted source code repository and software configuration management (SCM) system provided by Inflectra.

Projects in SpiraTeam can be enabled in TaraVault for storage of project source code files and revisions. TaraVault is provided

You can change which users are active for source code management in the [View / Edit Users](#) screen.

TaraVault Global Settings

Available TaraVault Users: 5

Active TaraVault Users: 1 ([View List](#))

TaraVault Account Name: inflectrtest3 (117326)

TaraVault Project Settings

Source Control project created!

TaraVault Project Active: Yes

TaraVault Project Name:

TaraVault Project Type: Subversion

Client Connection: <https://svn6.taravault.net/inflectrtest3/libraryinformationsystem/svn/>

Users assigned to this project:

Full Name ▲▼	User Name ▲▼	TaraVault Login	Operations
System Administrator	administrator	administrator	Edit Users

[Deactivate](#)

[Clear Cache](#)

The system will then populate the 'Client Connection' section with the URL that you need to use to connect to this TaraVault project from your Subversion or Git client (this is described in more detail in sections 4 and 5 respectively). In addition, it will display a list of users that are enabled for this project.

3.2. Managing Users

By default, the built-in system administrator account will be automatically enabled for TaraVault use and will be added as a member of all TaraVault projects. To enable other users to commit code/files to a TaraVault repository, you need to go to the Administration > Users > View/Edit Users menu item:

View / Edit Users ([Import Users From an LDAP Server](#))

The following approved users exist in the system. To edit an existing user, please click on the 'Edit' link next to his/her name.
To create a new user either click on the [Add] button or choose the option to Import from an LDAP server.
To see any pending new user requests, click on the 'Pending Requests' item in the left hand navigation.

First Name ▲▼	MI ▲▼	Last Name ▲▼	User Name ▲▼	Admin ▲▼	Email ? ▲▼	Department ▲▼	User # ▲▼	Active ▲▼	Operations
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	-- Any -- ▼	-- Any -- ▼	<input type="text"/>	US <input type="text"/>	-- Any -- ▼	
Amy	E	Cribbins	amycribbins	No	No	Software Engineering	US000013	No	> Edit
Bernard	P	Tyler	bernardtyler	No	No	Software Engineering	US000011	No	> Edit
Donna	W	Harkness	donnaharkness	No	No	Software Engineering	US000006	No	> Edit
Fred		Bloggs	fredbloggs	No	No	QA	US000002	Yes	> Edit
Henry	J	Cooper	henrycooper	No	No	Software Engineering	US000012	No	> Edit
Jack		Van Stanten	jackvanstanten	No	No	Software Engineering	US000007	No	> Edit
Joe	P	Smith	joesmith	No	No	QA	US000003	Yes	> Edit
Martha		Noble	marthanoble	No	No	Software Engineering	US000010	No	> Edit
Ricky		Pond	rickypond	No	No	Software Engineering	US000005	No	> Edit
Roger	Q	Ramjet	rogerramjet	No	No	Software Engineering	US000004	No	> Edit
Rory		Jones	roryjones	No	No	Software Engineering	US000009	No	> Edit
Rose	T	Smith	roesmith	No	No	Software Engineering	US000008	No	> Edit
System		Administrator	administrator	Yes	Yes		US000001	Yes	> Edit

[Filter](#) [Add](#) [Clear Filter](#)

This will display the user details page. On this page, if you have activated TaraVault there will be a special TaraVault tab that is used to enable a specific Spira user for TaraVault:

Project Membership
Passwords & Security
Data Mapping
TaraVault Membership

Enable user for TaraVault?
☐

Now check this box and the following screen appears:

Project Membership
Passwords & Security
Data Mapping
TaraVault Membership

Enable user for TaraVault?
☒

TaraVault User Login:
fredbloggs@taravault.net ()

TaraVault User Password*:

Enter in the desired password for TaraVault and click the main 'Update' button and this user will now be added to TaraVault. *Note that you need to have sufficient TaraVault licenses for all of the users you wish to activate.*

Finally, you can now add this user to specific TaraVault projects by clicking on the user's administration page again:

Project Membership
Passwords & Security
Data Mapping
TaraVault Membership

Enable user for TaraVault?
☒

TaraVault User Login:
fredbloggs@taravault.net (234759)

TaraVault User Password*:

Projects Enabled for this User:

Project ID	Project Name	Operations
This user is active, but not assigned to any projects. Add Projects		
		Add Projects

Now click on the 'Add Projects' link and you can now choose which TaraVault projects to add the user to:

Add User Project Membership | Fred Bloggs (fredbloggs)

[<< Back To User Details](#)

The user is **not** a member of the following TaraVault projects.
Select the checkbox to add the user to those projects, and then click the 'Add' button.

ID	Project Name	Project Active	Add User to Project?
[PR-1]	Library Information System	Yes	<input checked="" type="checkbox"/>

Add
Cancel

You should now click the 'Add User to Project' checkbox and then click the 'Add' button:

Project Membership
Passwords & Security
Data Mapping
TaraVault Membership

Enable user for TaraVault? ☒

TaraVault User Login: fredbloggs@taravault.net (234759)

TaraVault User Password*:

Projects Enabled for this User:

Project ID	Project Name	Operations
[PR: 1]	Library Information System	Remove
		Add Projects

The user will now be listed for that specific TaraVault project.

If you now go back to the main Administration > TaraVault page:

TaraVault Global Settings

Available TaraVault Users: 5

Active TaraVault Users: 2 ([View List](#))

TaraVault Account Name: inflectratest3 (117326)

TaraVault Project Settings

TaraVault Project Active **Yes**

TaraVault Project Name:

TaraVault Project Type: Subversion

Client Connection: <https://svn6.taravault.net/inflectratest3/libraryinformationsystem/svn/>

Users assigned to this project:

Full Name ▲▼	User Name ▲▼	TaraVault Login	Operations
System Administrator	administrator	administrator	Edit Users
Fred Bloggs	fredbloggs	fredbloggs	Edit Users

Deactivate
Clear Cache

You will see the two users listed under the current project. If you click on the 'Edit Users' hyperlink it will let you make changes to the user's TaraVault settings. For example you may wish to deactivate a user to free up a TaraVault license for another user.

Individual users can see their own TaraVault profile from the main Spira profile page. They need to click on the 'My Profile' link under their user's avatar on the main Spira navigation page:

Passwords & Security	Email Preferences	LDAP Settings	TaraVault Connection	Regional Settings	Actions
User Login: administrator@taravault.net					
Project Connection: https://svn6.taravault.net/inflectratest3/libraryinformationsystem/svn/					
User Password: (Click to show password.)					

This page displays the current user's TaraVault login as well as the connection string they should use to connect to the current project (the format will depend on whether the user is using Git or Subversion).

They can click on the 'Click to show password' option to reveal their password. This is necessary since they will need to know the password to use when connecting to Subversion / Git using their desired SCM client (e.g. TortoiseSVN, TortoiseGit, etc.).

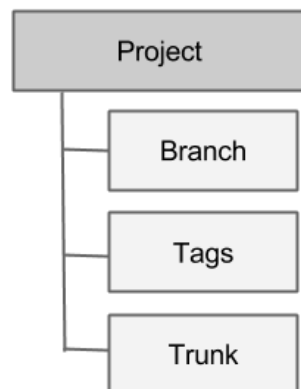
4. Using Subversion

Subversion is a version control system (VCS) – also known as a revision control system (RCS). That is, Subversion manages files and directories, and the changes made to them, over time. This allows you to recover older versions of your data or examine the history of how your data changed.

Some version control systems are also software configuration management (SCM) systems. These systems are specifically tailored to manage trees of source code and have many features that are specific to software development—such as natively understanding programming languages, or supplying tools for building software. Subversion, however, is not one of these systems. It is a general system that can be used to manage any collection of files, though it is often used for SCM, handling source code files

4.1. Repository Layout

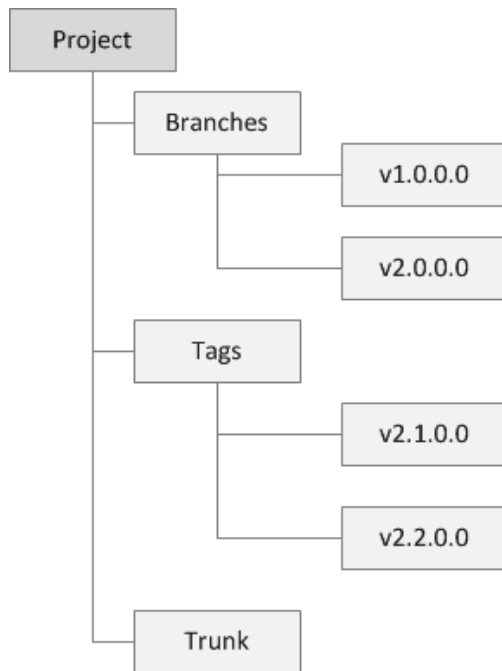
Before we get started with using Subversion, we need to discuss the standard folder layout. For TaraVault to display its branches, folders, files and revisions correctly in Spira you need to follow this layout for all your Subversion projects:



These three concepts are explained below:

- **Trunk** is the main folder containing all the data. This is the main line of current development for the project.
- A **Branch** contains copy of the trunk files and directories. These are used to denote older or non-primary versions of the Trunk. You may still commit changes into these branches. For example you may be still fixing bugs on an older version whilst primary development is occurring on the latest version.
- **Tags** can also be referred as milestone. This is a check-point to indicate that your project has reached a certain point. You can use this to mark various releases. Unlike a branch, you cannot commit changes into a tag.

An example setup for such a project could look like:



The same folders and files are typically stored inside each of the Branches, Tags and the Trunk. We shall illustrate this more closely when we get started in the next section.

4.2. Getting Started with Subversion

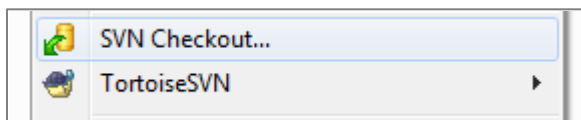
This section assumes that you have already provisioned at least one Subversion project in TaraVault following the steps in section 2 and 3. So you should now have a TaraVault user/password and a Subversion project with a connection URL:

Passwords & Security	Email Preferences	LDAP Settings	TaraVault Connection	Regional Settings	Actions
User Login: administrator@taravault.net					
Project Connection: https://svn6.taravault.net/inflectratest3/libraryinformationsystem/svn/					
User Password: (Click to show password.)					

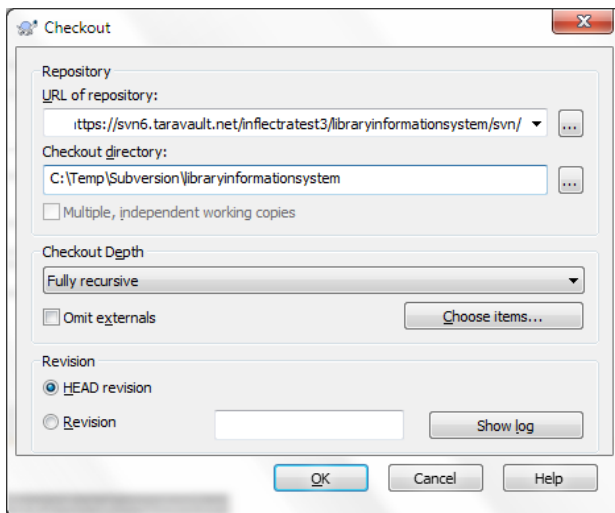
The next step is to actually connect to this repository using a Subversion client and commit some source code. We recommend using a GUI tool such as TortoiseSVN but you can use any standard Subversion client with TaraVault (command-line or GUI-based) just as well. In our examples we shall be using TortoiseSVN.

The first thing we need to do is perform an initial 'check-out' of our repository into a new working folder (that will initially be empty).

Assuming that you have already installed TortoiseSVN, you would now create a folder to hold all of your Subversion projects (in our example we shall use C:\Temp\Subversion) and right-click and choose TortoiseSVN > Check Out:



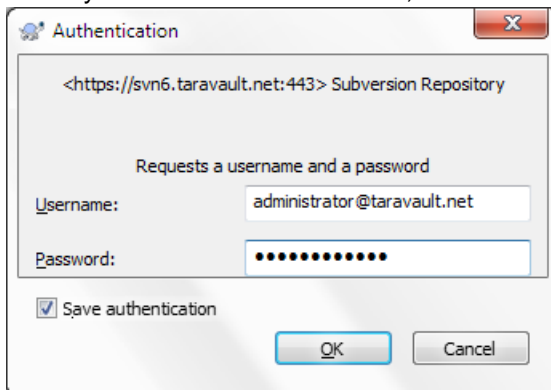
The following dialog box will appear:



You need to enter the following:

- **URL of repository** – needs to be the TaraVault connection string listed under 'My Profile' for the current project.
- **Checkout Directory** – needs to be the local name of the folder for this project. Typically it is best to make it the same as the name of the project in TaraVault (e.g. `C:\Temp\Subversion\libraryinformationsystem` in this example)

When you click on the 'OK' button, the following authentication dialog box will appear:

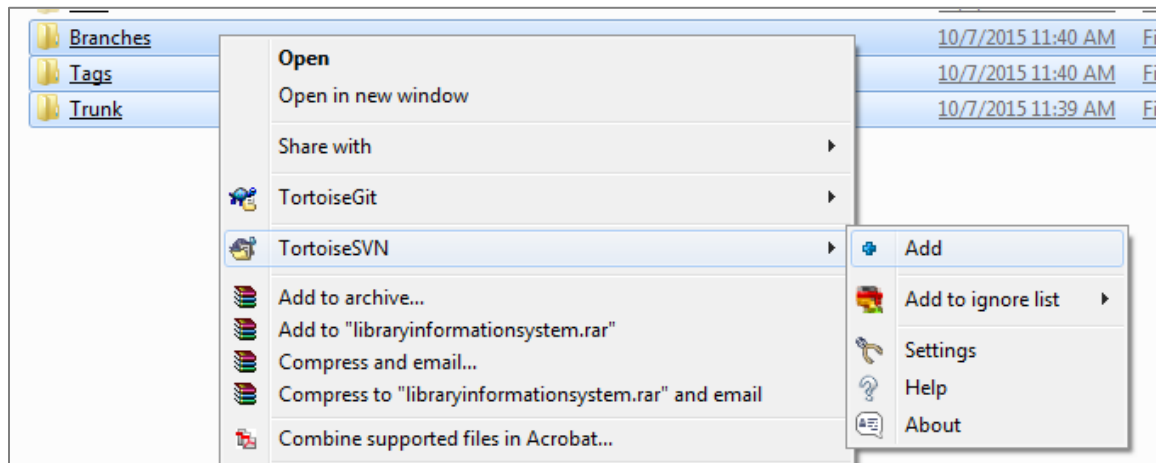


Enter your TaraVault username/password, choose 'Save authentication' and then click 'OK'. You will now get a folder `C:\Temp\Subversion\libraryinformationsystem` that is completely empty apart from a special `_svn` (or `.svn`) folder that is used by TortoiseSVN internally.

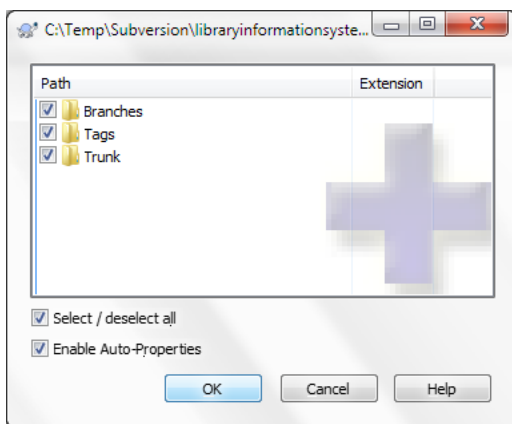
Now that you have your Subversion working folder downloaded, you should add the following three folders right now:

- Trunk
- Branches
- Tags

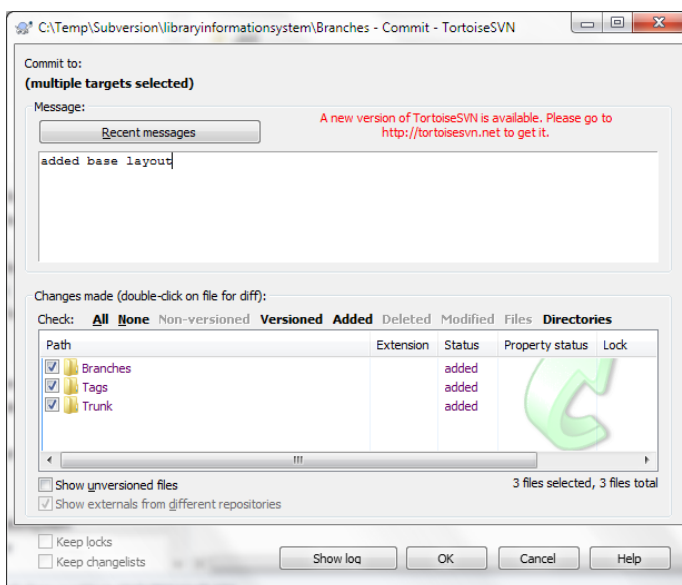
Once you have added them to Windows Explorer, select them all and choose TortoiseSVN > Add:



This will display the following:



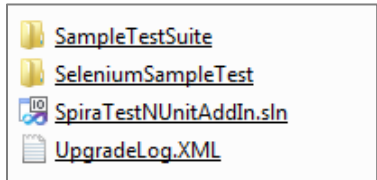
Once they are added, then choose TortoiseSVN > Commit to commit these folders to the TaraVault repository. That will display the following dialog box:



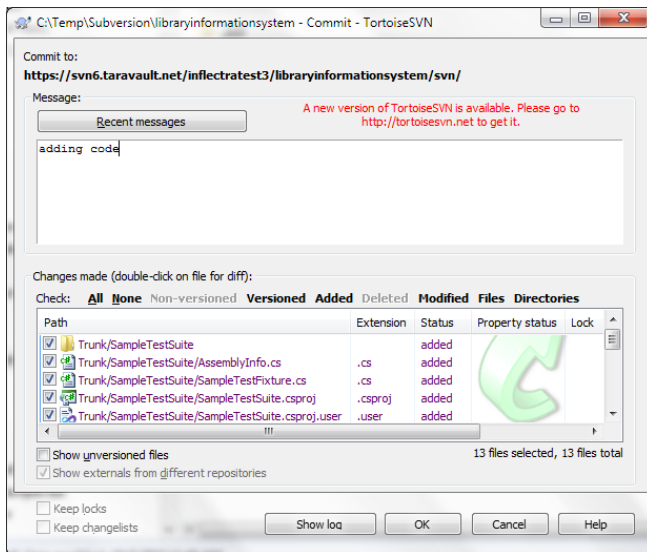
Typically you should add a message to describe what you did. Then click 'OK' and the three layout folders will now be added to your TaraVault subversion repository.

4.3. Adding Files to the Trunk

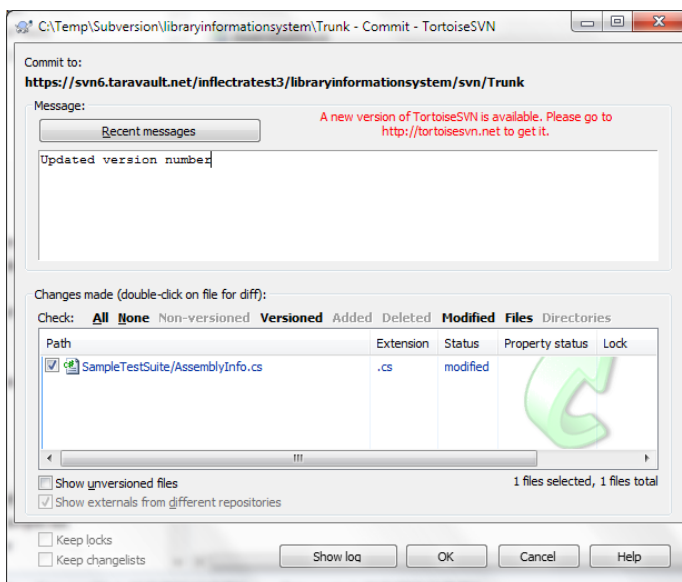
Now that you have your Subversion repository layout setup, we shall simulate working on a real project. You can now copy some code and folders into the Trunk top-level folder for your project. In this example we shall add some sample Inflectra code:



Select all the files and folders and choose TortoiseSVN > Add, then after adding the files and folders, choose TortoiseSVN > Commit to add these files to the repository.



Now, open up one of the files (we shall modify the SampleTestSuite\AssemblyInfo.cs file in our example) and make a change to it. Then right-click on Trunk and choose TortoiseSVN > Commit to commit the change. Make sure you add a comment:

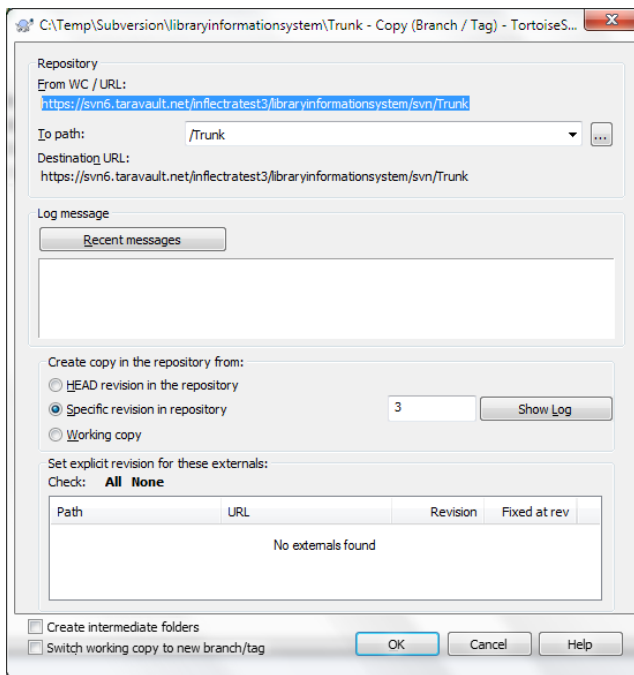


Click OK and the change (known as a revision) will now be committed into TaraVault.

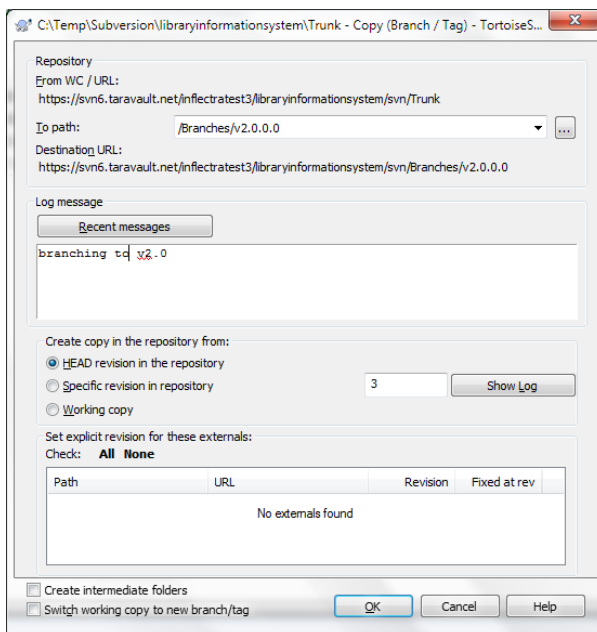
4.4. Branching and Tagging

Now that we have the primary development line in our Trunk, we can branch and tag a specific version of the code before we make other changes. For example we might be releasing a version and then making changes specific only to the next version.

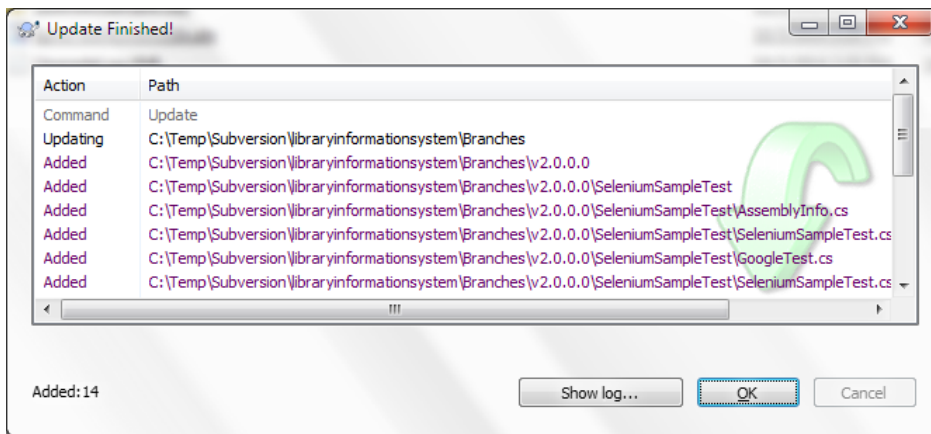
We shall create both a branch and a tag from the current Trunk. Firstly, to create a Branch, right-click on Trunk and choose TortoiseSVN > Branch/Tag:



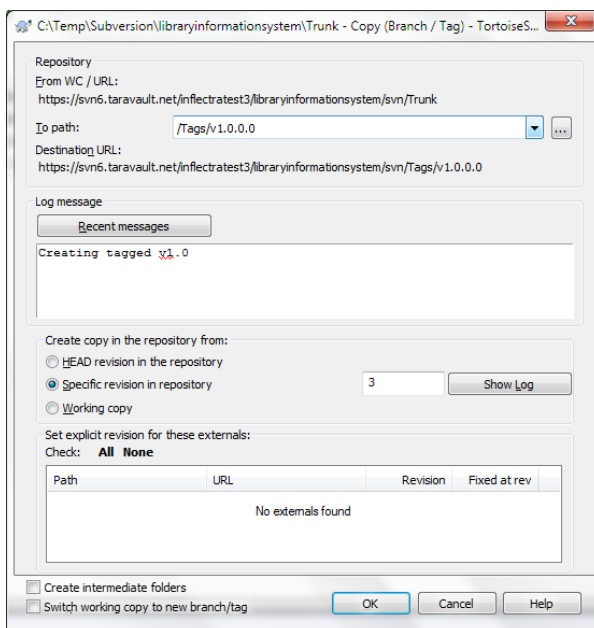
Change the 'To path' from /Trunk to /Branches/v2.0.0.0. You can either branch the latest revision (called the HEAD revision) or a specific revision:



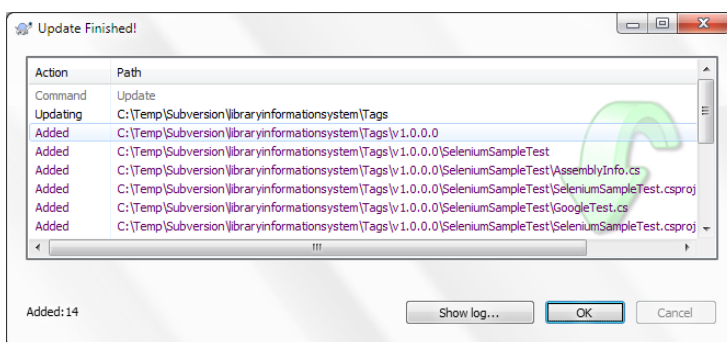
We also recommend adding a 'Log message' to describe the purpose of making the branch. Once you are happy with your choice, click 'OK' to confirm the branch. Once that is done, a copy of the Trunk will now be available under the Branches/v2.0.0.0 folder. To see this, right-click on Branches and choose TortoiseSVN > Update:



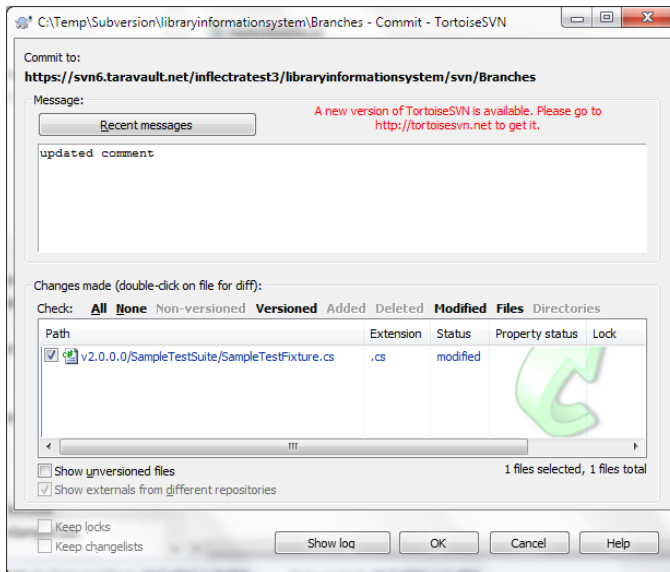
Similarly, to make a Tag, right-click on Trunk and choose TortoiseSVN > Branch/Tag, and change the 'To path' from /Trunk to /Tags/v1.0.0.0:



Once that has been completed, right-click on the Tags folder and choose TortoiseSVN > Update:



The main difference between a Branch and a Tag is that you can continue to make changes on a Branch, whereas a Tag is a fixed snapshot of the Trunk and cannot be modified. To illustrate this, let's simulate making a bug fix on the v2.0 branch we made. To do that, change one of the files in the /Branches/v1.0.0.0 folder structure and then right-click TortoiseSVN > Commit:

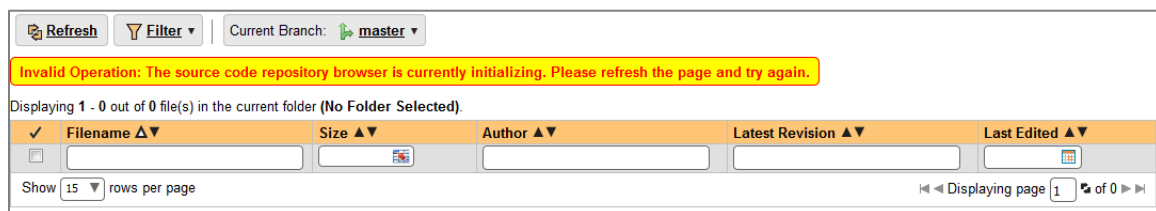


Click 'OK' and we are now ready to view the repository within Spira.

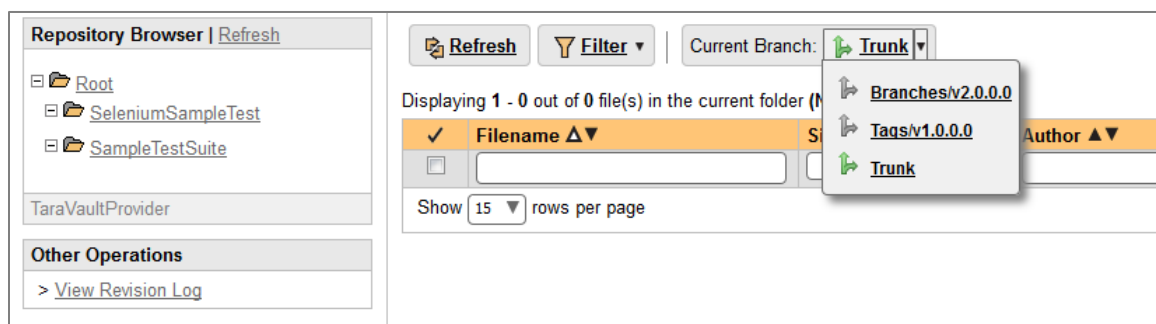
4.5. Viewing within Spira

In addition to being able to browse the source code repository in Spira, which is itself very useful, the real strength comes from linking artifacts in Spira - including Incidents, Requirements, and Tasks - to revisions checked into the software repository, indicating what was fixed or implemented in a specific revision.

Firstly, you can view the source code tree by selecting the Tracking > Source Code link under the main Spira menu. This will display the following:

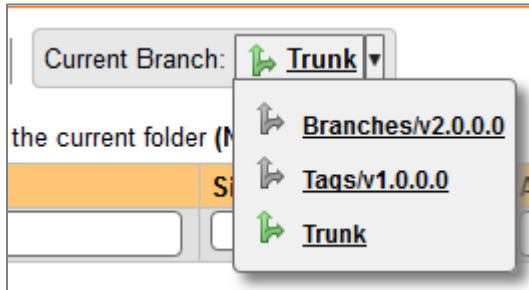


This is normal, it means simply that the source code viewer is initializing for initial use. Whenever you access a TaraVault project for the first time you will see this. After a few minutes it will be ready for use and you can refresh the page to display:



The folder tree of the repository is on the left, and files in the current selected directory will be listed in the right table. Note that this view will always show the current (HEAD) revision of the repository.

Also note that the page will display the folders and files for the currently selected branch (in the example above "Trunk"), you can change the current branch at any time by selecting it from the dropdown menu:



The file grid will display the filename, the current revision number of the file, the author of the last commit, and the date of the last commit. You can filter and sort on any of the columns, as well:

Displaying 1 - 5 out of 5 file(s) in the current folder SampleTestSuite.

✓	Filename ▲▼	Size ▲▼	Author ▲▼	Latest Revision ▲▼	Last Edited ▲▼
<input type="checkbox"/>	AssemblyInfo.cs	2426 KB	administrator@taravault.net	3	7-Oct-2015
<input type="checkbox"/>	SampleTestFixture.cs	1732 KB	administrator@taravault.net	2	7-Oct-2015
<input type="checkbox"/>	SampleTestSuite.csproj	6011 KB	administrator@taravault.net	2	7-Oct-2015
<input type="checkbox"/>	SampleTestSuite.csproj.user	452 KB	administrator@taravault.net	2	7-Oct-2015
<input type="checkbox"/>	SampleTestSuite.nunit	189 KB	administrator@taravault.net	2	7-Oct-2015

Show 15 rows per page

Displaying page 1 of 1

To view the file details, click on a file hyperlink in the grid. The file details page displays the details on the selected file. By default, it will be the HEAD revision, unless you clicked to view the file details from a specific revision. By clicking on the file name, you can download the specified revision of the file to your local machine. This does not do an Subversion checkout; you are merely downloading the file to your local machine.

[<< Back to File List](#)

- SampleTestSuite
 - AssemblyInfo.cs
 - SampleTestFixture.cs**
 - SampleTestSuite.csproj
 - SampleTestSuite.csproj.user
 - SampleTestSuite.nunit

Current Branch: [Trunk](#)

Source Code File: SampleTestFixture.cs

File Path: [/SampleTestSuite/SampleTestFixture.cs](#)

File Type (Size): C# Source (1732 KB)

Author: administrator@taravault.net

Last Edited: 10/7/2015 12:43:21 PM

Latest Revision: [2](#)

[Preview #](#)
[Revisions #](#)
[Associations](#)

```

1 using System;
2 using NUnit.Framework;
3 using Inflectra.SpiraTest.AddOns.SpiraTestNUnitAddIn.SpiraTestFramework;
4
5 namespace Inflectra.SpiraTest.AddOns.SpiraTestNUnitAddIn.SampleTestSuite
6 {
7     /// <summary>
8     /// Sample test fixture that tests the NUnit SpiraTest integration
9     /// </summary>
10     [
11         TestFixture,
12         SpiraTestConfiguration("http://localhost/SpiraTest", "fredbloggs", "fredbloggs", 1, 1, 2, SpiraTestConfigure
13     ]
14     public class SampleTestFixture
15     {

```

Underneath the file details are tabs that show a preview of the file (with syntax highlighting), a list of all the revisions that this file belongs in, or was committed to, who performed the commit, and the log message for the commit, and a tab that shows any artifact associations. Throughout SpiraTeam, revisions are indicated by the icon:

Revisions						
Refresh Apply Filter Clear Filter						
✓	Revision ▲▼	Author ▲▼	Summary ▲▼	Commit Date ▲▼	Content ▲▼	Properties ▲▼
<input type="checkbox"/>	2	administrator@taravault.net	adding code	7-Oct-2015	No	No
Show 15 rows per page Displaying page 1 of 1						

By clicking on a revision in SpiraTeam, you will be taken to the revision details page.

<< Back to Revision List

SampleTestFixture.cs

2

Current Branch: Trunk

Source Code Revision: 2

Notes: adding code

Edited By: administrator@taravault.net

Build:








Last Edited: 10/7/2015 12:43:21 PM

Content Δ: No

Properties Δ: No

Files Associations

> Refresh Apply Filter Clear Filter

✓	Filename ▲▼	Size ▲▼	Author ▲▼	Latest Revision ▲▼	Action ▲▼	Last Edited ▲▼
<input type="checkbox"/>						
<input type="checkbox"/>	 AssemblyInfo.cs	2426 KB	administrator@taravault.net	2	Added	7-Oct-2015
<input type="checkbox"/>	 AssemblyInfo.cs	2426 KB	administrator@taravault.net	3	Added	7-Oct-2015
<input type="checkbox"/>	 GoogleTest.cs	1541 KB	administrator@taravault.net	2	Added	7-Oct-2015
<input type="checkbox"/>	 SampleTestFixture.cs	1732 KB	administrator@taravault.net	2	Added	7-Oct-2015
<input type="checkbox"/>	 SampleTestSuite.csproj	6011 KB	administrator@taravault.net	2	Added	7-Oct-2015

The revision details screen shows the log for the Commit, the commit date and author. At the bottom of the page are two tabs, Files and Associations. The Files tab lists all files that were a part of this commit, with their full path and the action that was performed on them for this commit. Possible values are Added, Modified, or Deleted.

The Associations tab shows any artifact (Incident, Task, Requirement, Test Case, Test Set, etc.) that the log message references. See the next section for information on how to link a revision with a Subversion Commit:

<

4.6. Linking Artifacts

Linking an artifact is quite simple. To maintain the readability of Subversion's commit messages, we adopted a bracket token. The token is in the format of:

[<artifact identifier>: <artifact id>]

The first half, the Artifact Identifier, is a two-letter code that is used throughout SpiraTeam, and is visible on almost every page in the application. For example, a requirement's identifier is "RQ". Incidents are "IN", and test cases are "TC". The artifact ID is the number of the artifact. So by creating a commit message that reads:

Due to requirement #12 [RQ:12], the code for `.toString` in class `XMLparser` was modified. This also fixed Incident #1034 [IN:1034].

SpiraTeam will automatically detect tokens and will include links to them under the Associations tab for a revision detail.

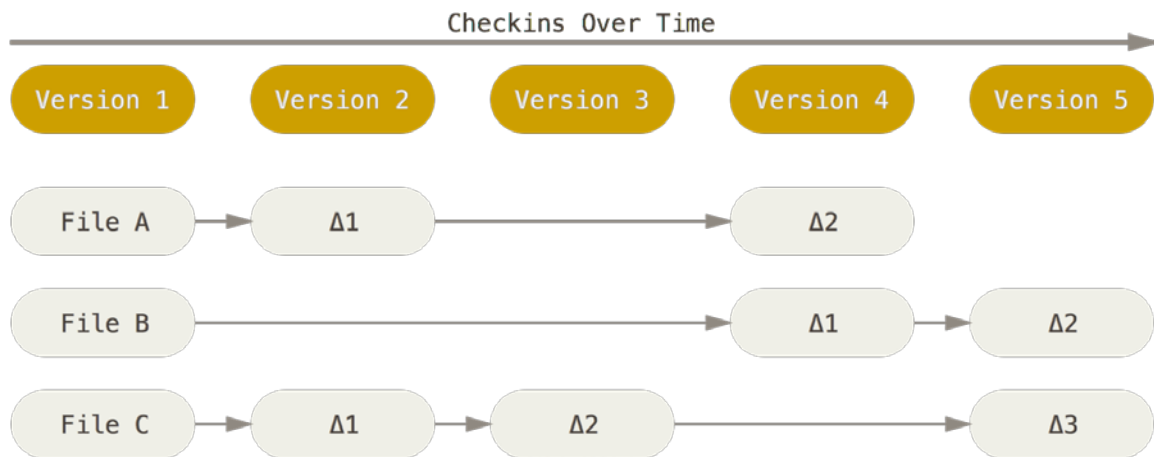
If you forget to add the association during the commit, you can use the 'Add Association' option within SpiraTeam to add the association after the fact. This is described in more detail in the *SpiraTeam User Manual*.

5. Using Git

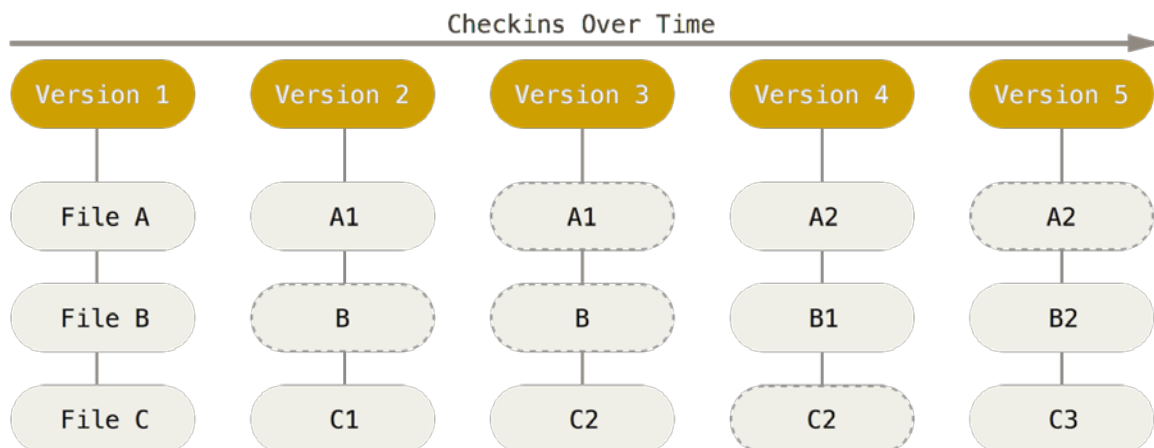
Git is a Distributed Version Control System (DVCS) system that keeps track of software revisions and allows many developers to work on a given project without necessarily being connected to a common network since it doesn't rely on a central repository, but instead distributes copies of the entire source code repository to each user's workstation.

5.1. Git Basics

The major difference between Git and most other VCS (e.g. Subversion described in the previous section) is the way Git thinks about its data. Conceptually, most other systems store information as a list of file-based changes. These systems (CVS, Subversion, Perforce, Bazaar, and so on) think of the information they keep as a set of files and the changes made to each file over time.



Git doesn't think of or store its data this way. Instead, Git thinks of its data more like a set of snapshots of a miniature filesystem. Every time you commit, or save the state of your project in Git, it basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot. To be efficient, if files have not changed, Git doesn't store the file again, just a link to the previous identical file it has already stored. Git thinks about its data more like a stream of snapshots:



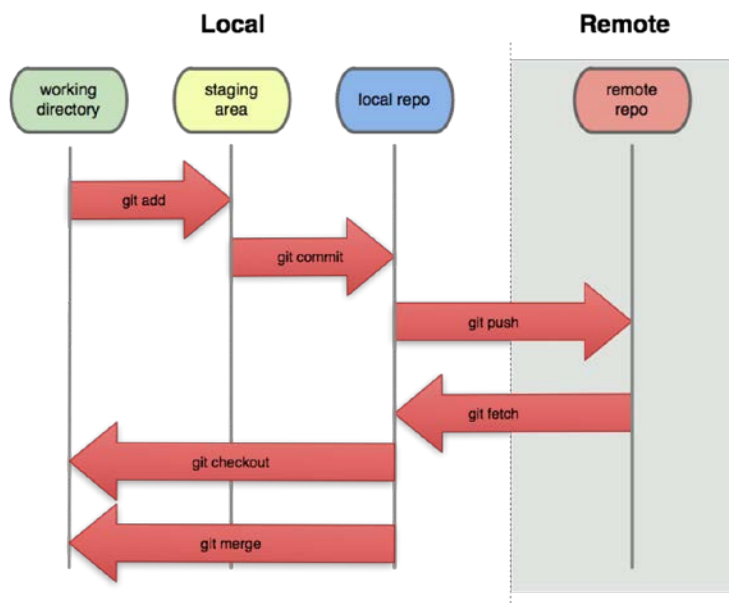
This is an important distinction between Git and nearly all other VCSs. It makes Git reconsider almost every aspect of version control that most other systems copied from the previous generation. This makes Git more like a mini filesystem with some incredibly powerful tools built on top of it, rather than simply a VCS.

Another major difference between Git and Subversion is that Git has built-in support for 'branches' instead of relying on a specific folder structure. In Git, Branches are used to develop features isolated from each other. The master branch is the "default" branch when you create a repository. You can then use other branches for development and merge them back to the master branch upon completion.

5.2. Working with Remotes

To be able to work on any Git project, you need to know how to manage your remote repositories. Remote repositories are the versions of your project that are hosted on TaraVault itself. Collaborating with others involves managing these remote repositories and pushing and pulling data to and from them when you need to share work. Managing remote repositories includes knowing how to add remote repositories, remove remotes that are no longer valid, manage various remote branches and define them as being tracked or not, and more.

This is in stark contrast to Subversion (described in section 4) where every commit and update is being performed directly on the central TaraVault remote repository. So when you make changes to your local repository, you will need to explicitly synchronize with TaraVault for other users to see them, and for them to appear in Spira:



5.3. Getting Started with Git

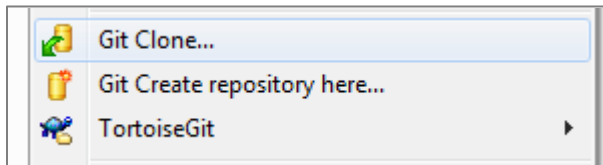
This section assumes that you have already provisioned at least one Git project in TaraVault following the steps in section 2 and 3. So you should now have a TaraVault user/password and a Git project with a connection URL:

Passwords & Security	Email Preferences	LDAP Settings	TaraVault Connection	Regional Settings	Actions
<p>User Login: administrator@taravault.net</p> <p>Project Connection: https://git6.taravault.net/inflectrat3st3/sampleapplicationone/git/</p> <p>User Password: 56310dc45e0</p>					

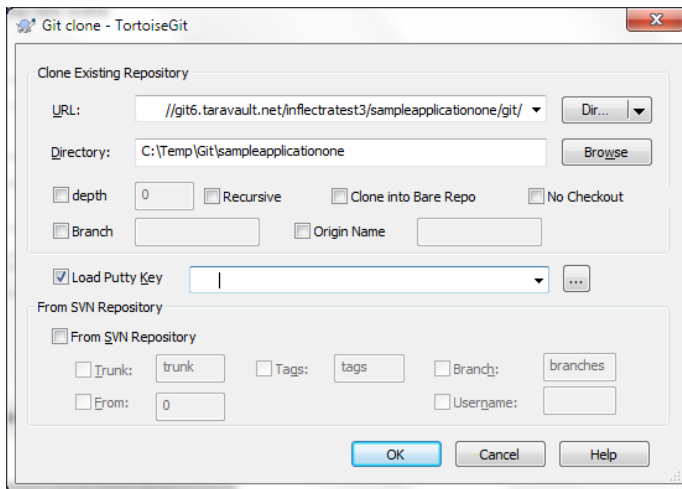
The next step is to actually connect to this repository using a Git client and commit some source code. We recommend using a GUI tool such as TortoiseGit but you can use any standard Git client with TaraVault (command-line or GUI-based) just as well. In our examples we shall be using TortoiseGit.

The first thing we need to do is perform an initial 'clone' of our remote Git repository into a local repository. This will also do an implicit checkout from the local repository into the working directory.

Assuming that you have already installed TortoiseGit, you would now create a folder to hold all of your Git projects (in our example we shall use `C:\Temp\Git`) and right-click and choose "Git Clone":



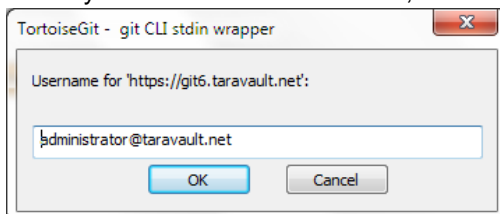
The following dialog box will appear:



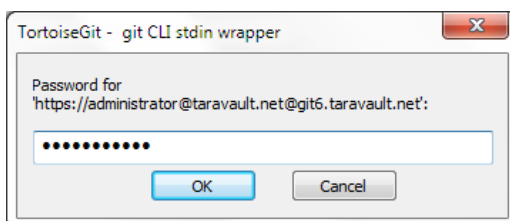
You need to enter the following:

- **URL**– needs to be the TaraVault Git connection string listed under 'My Profile' for the current project.
- **Directory** – needs to be the local name of the folder for the local repository. Typically it is best to make it the same as the name of the project in TaraVault (e.g. `C:\Temp\Git\libraryinformationsystem` in this example)

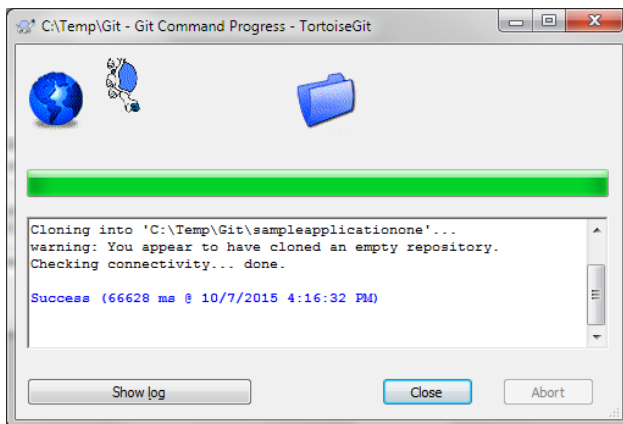
When you click on the 'OK' button, the following authentication dialog box will appear:



Enter your TaraVault Git username, and then click 'OK'. Then the following will appear:



Enter your TaraVault password and then click 'OK' again. The success dialog will appear:



You will now get a folder `C:\Temp\Git\sampleapplicationone` that is completely empty apart from a special `.git` folder that is used by Git internally.

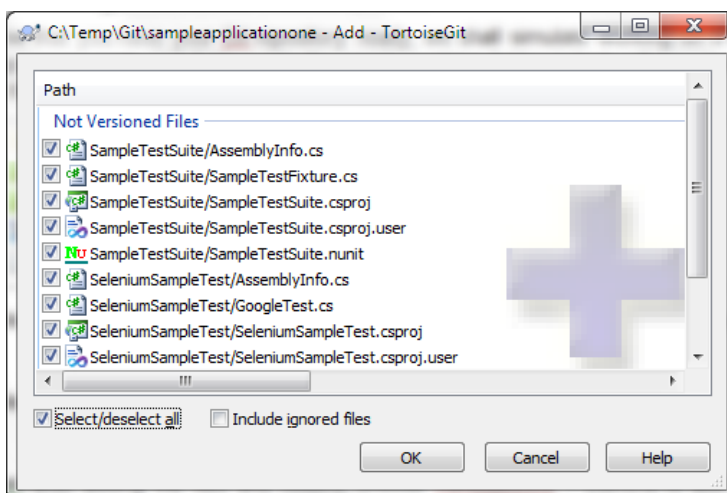
Now that you have your Git local repository and working folder available, you are ready to start using Git.

5.4. Adding Files to the Master Branch

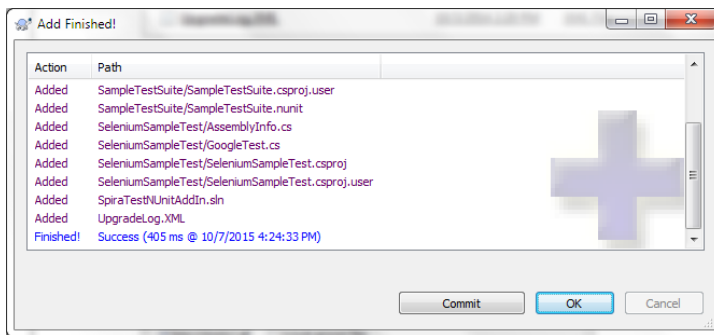
Now that you have your Git repository ready, we shall simulate working on a real project. You can now copy some code and folders into the empty working folder (which will be set to use the 'master' branch). In this example we shall add some sample Inflectra code:

<code>.git</code>	10/7/2015 4:16 PM	File folder	
<code>SampleTestSuite</code>	10/7/2015 4:20 PM	File folder	
<code>SeleniumSampleTest</code>	10/7/2015 4:20 PM	File folder	
<code>SpiraTestNUnitAddIn.sln</code>	10/3/2014 2:29 PM	Microsoft Visual S...	3 KB
<code>UpgradeLog.XML</code>	10/3/2014 2:29 PM	XML File	5 KB

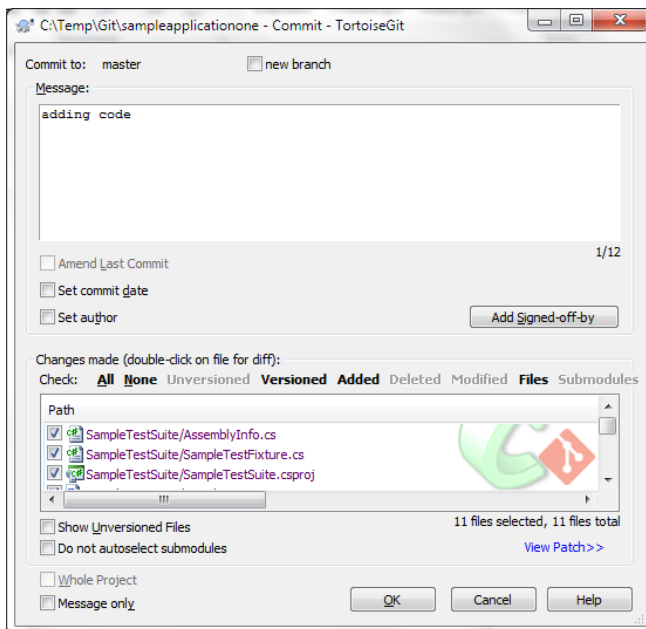
Select all the files and folders and choose TortoiseGit > Add. That will display the following dialog box:



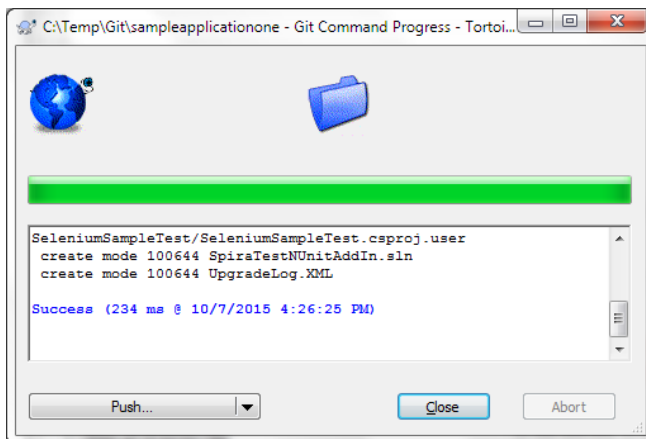
Select all the files and click OK.



Click on [OK] to commit the files to the local repository. Note that this does not send them to the remote TaraVault repository at this stage since this is a distributed version control system and all commits are done locally.

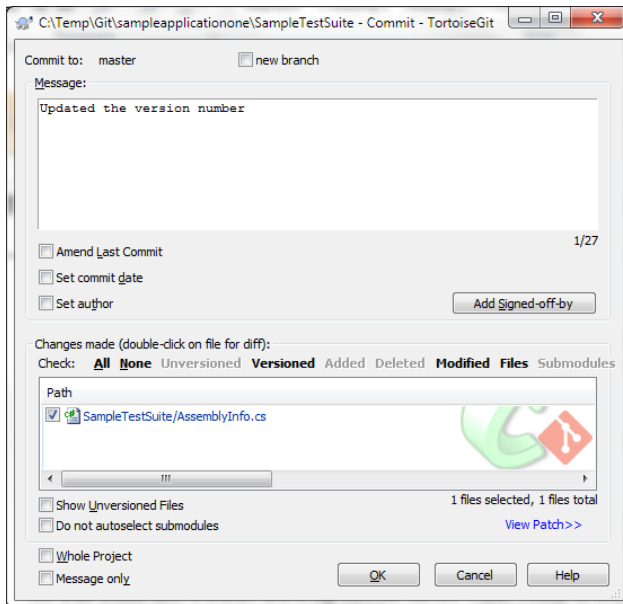


Enter in the appropriate commit message and then click [OK] to complete the local commit. The following screen will be displayed:



At this stage, we are not ready to 'Push' the repository to TaraVault so just click 'Close'.

Now, open up one of the files (we shall modify the SampleTestSuite\AssemblyInfo.cs file in our example) and make a change to it. Then right-click on the top-level 'sampleapplicationone' folder and choose Git Commit -> "master" to commit the change. Make sure you add a comment:

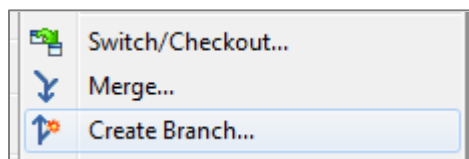


Click OK and the change (known as a revision) will now be committed into the 'master' branch of the local repository.

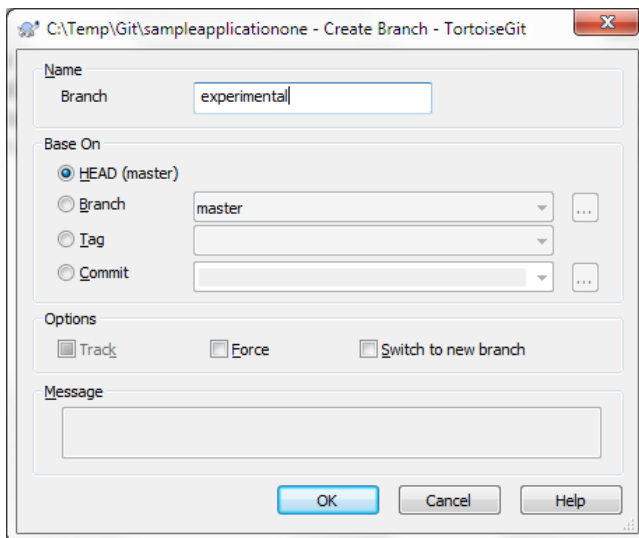
5.5. Working with Branches

Now that we have the primary development line in our master branch, we can work on a separate version of the code in a different branch, and then at a later date merge back in the changes to the 'master' branch. For example we might be working on an experimental new feature and we only want to merge it into the 'master' when it is internally stable and all the unit tests pass.

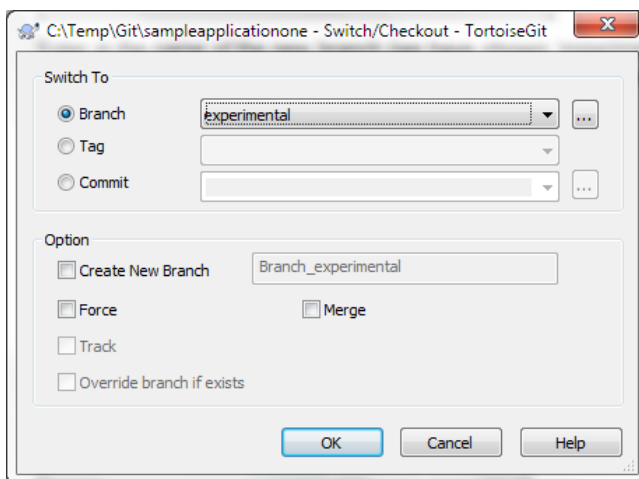
We shall create a new named branch in our local repository using TortoiseGit. To do that right-click on the top-level folder (sampleapplicationone) and then click Tortoise Git > Create Branch:



This will display the following dialog box:

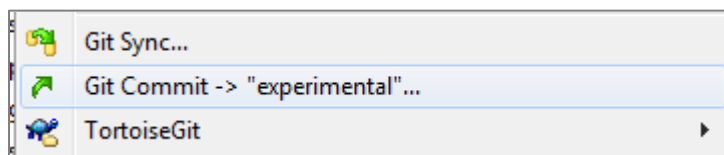


Enter in the name of the new branch (we have chosen 'experimental' in the example shown) and click [OK] to create the new branch. Since we want to start working in this new branch, we should now using TortoiseGit > Switch/Checkout to switch to this new branch:

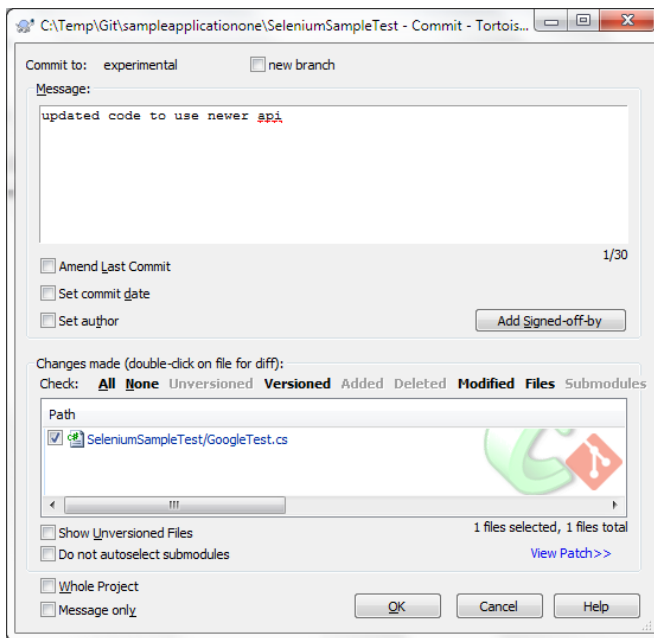


Click [OK] to confirm the switch. Now all your changes will be made on this branch.

Now, let's simulate making a code change on the 'experimental' branch we made. To do that, change one of the files in the folder structure and then right-click Git Commit -> "experimental":



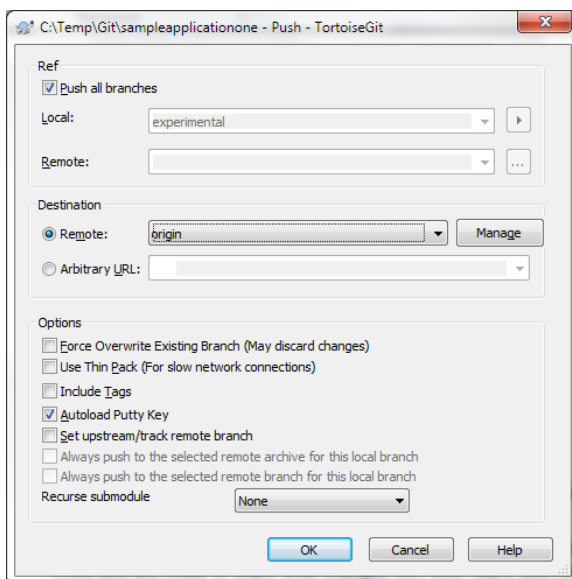
That will display the commit dialog box:



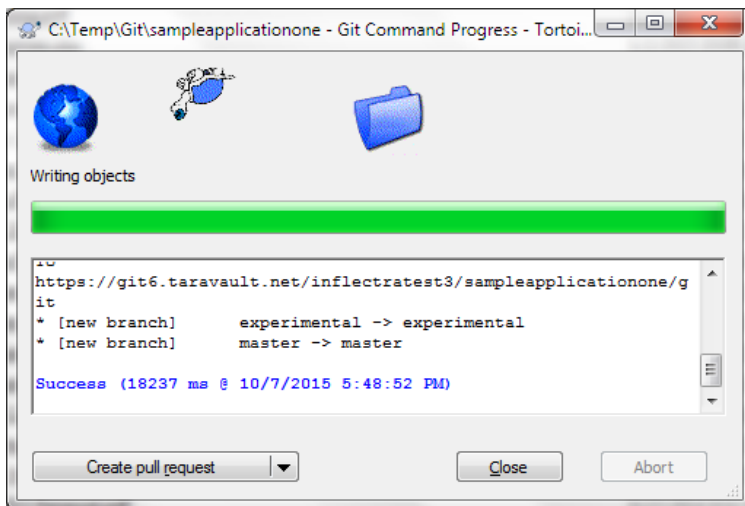
Enter in a comment in the 'Message' text box that describes the purpose of the change. Then click [OK]. On the success dialog box that appears, click 'Close'.

Now unlike other VCS such as Subversion, we have made all of these changes in the local Git repository. Once you are ready to share your changes with your team, you need to 'Push' the local branches to the remote TaraVault repository.

To do this, right-click on the top-level folder (sampleapplicationone) and choose TortoiseGit > Push. The following dialog box will be displayed:



In this case we want to push all branches (master and experimental) to TaraVault, so select the checkbox 'Push all branches'. Then click 'OK' to push the changes:

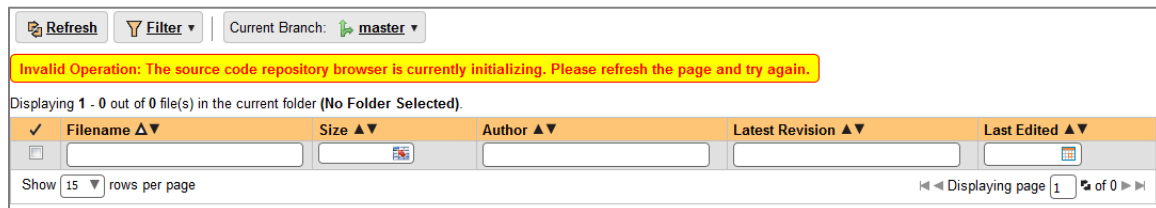


Once they have been pushed, we are now ready to view the repository within Spira.

5.6. Viewing within Spira

In addition to being able to browse the source code repository in Spira, which is itself very useful, the real strength comes from linking artifacts in Spira - including Incidents, Requirements, and Tasks - to revisions checked into the software repository, indicating what was fixed or implemented in a specific revision.

Firstly, you can view the source code tree by selecting the Tracking > Source Code link under the main Spira menu. This will display the following:

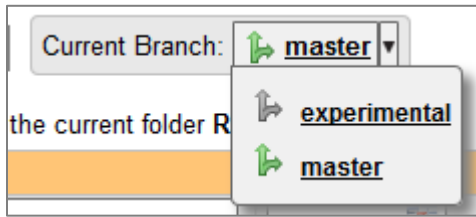


This is normal, it means simply that the source code viewer is initializing for initial use. Whenever you access a TaraVault project for the first time you will see this. After a few minutes it will be ready for use and you can refresh the page to display:



The folder tree of the repository is on the left, and files in the current selected directory will be listed in the right table. The file view will display the filename, the current revision number of the file and the date of the last commit. You can filter and sort on any of the columns, as well.

The page will display the folders and files for the currently selected branch (in the example above "master"), you can change the current branch at any time by selecting it from the dropdown menu:



To view the file details, click on a file hyperlink the main source code file grid. The file details page displays the details on the selected revision. By default, it will be the latest revision in the current branch, unless you clicked to view the file details from a revision. By clicking on the file name, you can download the specified revision of the file to your local machine. This does not do a Git clone or pull; you are merely downloading the file to your local machine.

[<< Back to File List](#)

- SampleTestSuite
 - AssemblyInfo.cs**
 - SampleTestFixture.cs
 - SampleTestSuite.csproj
 - SampleTestSuite.csproj.user
 - SampleTestSuite.nunit

Current Branch: **master**

Source Code File: AssemblyInfo.cs

File Path: [/SampleTestSuite/AssemblyInfo.cs](#)

File Type (Size): C# Source (2426 KB)

Author: Adam Sandman

Last Edited: 10/7/2015 4:29:25 PM

Latest Revision: [834ef79](#)

Preview

Revisions

Associations

```

1 using System.Reflection;
2 using System.Runtime.CompilerServices;
3
4 //
5 // General Information about an assembly is controlled through the following
6 // set of attributes. Change these attribute values to modify the information
7 // associated with an assembly.
8 //
9 [assembly: AssemblyTitle("")]
10 [assembly: AssemblyDescription("")]
11 [assembly: AssemblyConfiguration("")]
12 [assembly: AssemblyCompany("")]
13 [assembly: AssemblyProduct("")]
14 [assembly: AssemblyCopyright("")]
15 [assembly: AssemblyTrademark("")]
16 [assembly: AssemblyCulture("")]

```

Underneath the file details are tabs that show a preview of the file (with syntax highlighting), a list of all the revisions that this file belongs in, or was committed to, who performed the commit, and the log message for the commit, and a tab that shows any artifact associations. Throughout SpiraTeam, revisions are indicated by the icon:

Source Code File: AssemblyInfo.cs

File Path: [/SampleTestSuite/AssemblyInfo.cs](#)

File Type (Size): C# Source (2426 KB)

Author: Adam Sandman

Last Edited: 10/7/2015 4:29:25 PM

Latest Revision: [834ef79](#)

Preview

Revisions

Associations

[Refresh](#) | [Apply Filter](#) | [Clear Filter](#)

✓	Revision ▲▼	Author ▲▼	Summary ▲▼	Commit Date ▲▼	Content ▲▼	Properties ▲▼
<input type="checkbox"/>					-- Any --	-- Any --
<input type="checkbox"/>	834ef79	Adam Sandman	Updated the version number	7-Oct-2015	No	No
<input type="checkbox"/>	e97f2e1	Adam Sandman	adding code	7-Oct-2015	No	No

Show rows per page
 ◀◀ Displaying page 1 of 1 ▶▶

By clicking on a revision in SpiraTeam, you will be taken to the revision details page.

[<< Back to Revision List](#)

- AssemblyInfo.cs
- 834ef79
- e972e1

Current Branch: master

Source Code Revision: 834ef79

Notes: Updated the version number

Edited By: Adam Sandman

Build:

Last Edited: 10/7/2015 4:29:25 PM

Content Δ: No

Properties Δ: No

Files Associations

Refresh Apply Filter Clear Filter

Filename Δ	Size ▲▼	Author ▲▼	Latest Revision ▲▼	Action ▲▼	Last Edited ▲▼
AssemblyInfo.cs	2426 KB	Adam Sandman	834ef79	Modified	7-Oct-2015

Show 15 rows per page

Displaying page 1 of 1

The revision details screen shows the log for the commit, the commit date and author. At the bottom of the page are two tabs, Files and Associations. The Files tab lists all files that were a part of this commit, with their full path, size, latest revision and date of edit.

The Associations tab shows any artifact (Incident, Requirement, Test Case, Test Set) that the log message references. See section 8.2.4 for information on how to link a revision with a Git commit:

Source Code Revision: 5bfb75478c

Notes: Fixes [IN:002943]

Edited By: Adam Sandman

Build: Rapise Master #208

Last Edited: 10/1/2014 12:07:38 PM

Content Δ: Yes

Properties Δ: Yes

Files Associations

Date	Artifact Name	Creator	Comment	Artifact Type	ID	Operations
1-Oct-2014	RapiseLauncher doesn't execute on our internal ser...	Adam Sandman	Fixes [IN:002943]	Incident	IN002943	

Add Association

5.7. Linking Artifacts

Linking an artifact is quite simple. To maintain the readability of Git commit messages, we adopted a square bracket token. The token is in the format of:

[<artifact identifier>: <artifact id>]

The first half, the Artifact Identifier, is a two-letter code that is used throughout SpiraTeam, and is visible on almost every page in the application. For example, a requirement's identifier is "RQ". Incidents are "IN", and test cases are "TC". The artifact ID is the number of the artifact. So by creating a commit message that reads:

Due to requirement #12 [RQ:12], the code for .toString in class XMLparser was modified. This also fixed Incident #1034 [IN:1034].

SpiraTeam will automatically detect tokens and will include links to them under the Associations tab for a revision detail.

If you forget to add the association during the commit, you can use the 'Add Association' option within SpiraTeam to add the association after the fact.

Legal Notices

This publication is provided as is without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

This publication could include technical inaccuracies or typographical errors. Changes are periodically added to the information contained herein; these changes will be incorporated in new editions of the publication. Inflectra Corporation may make improvements and/or changes in the product(s) and/or program(s) and/or service(s) described in this publication at any time.

The sections in this guide that discuss internet web security are provided as suggestions and guidelines. Internet security is constantly evolving field, and our suggestions are no substitute for an up-to-date understanding of the vulnerabilities inherent in deploying internet or web applications, and Inflectra cannot be held liable for any losses due to breaches of security, compromise of data or other cyber-attacks that may result from following our recommendations.

Spira™, TaraVault™, SpiraPlan®, SpiraTeam® and Inflectra® are either trademarks or registered trademarks of Inflectra Corporation in the United States of America and other countries. Microsoft®, Windows®, Visual Studio, Explorer® and Visual SourceSafe® are registered trademarks of Microsoft Corporation. Subversion® is a registered trademark of Collabnet, Inc. All other trademarks and product names are property of their respective holders.

Please send comments and questions to:

Technical Publications

Inflectra Corporation

8121 Georgia Ave, Suite 504

Silver Spring, MD 20910-4957

U.S.A.

support@inflectra.com