TCAT Source Code repository Customer Instructions

Contents

1. Introduction

This document describes the use of the TCAT source code repository. Customers may use this repository to stay current with source code made available by TCAT for firmware and Host software development using the DICE family of chips.

The repository holds TCAT Firmware and Host software under development, along with other information and utilities used internally by TCAT and (parts of it) externally by outside customers.

The latest version of this document is kept in:

https://dev.tctechnologies.tc/tcat/tags/release/public/latest/docs/TCAT_repos_how_to.pdf

2. Background

SCM method

The server uses Subversion. There are lots of Subversion clients available for any platform. Examples here will show the *command line svn client* since it is common across all platforms. For Windows users, we include a bonus section at the end showing how to use Tortoise SVN with the TCAT repository.

Some links are provided here for convenience:

Subversion Books

You can download the svn-book here:
http://svnbook.red-bean.com/

It's a good idea to be familiar with the concepts from Chapter 2 of this book, and perhaps Chapter 1 starting with "Subversion in Action"

A list of others can be found here:
http://subversion.tigris.org/links.html#books

Subversion Clients
A fairly exhaustive list of clients can be found here:
http://subversion.tigris.org/links.html#clients

For those who will be developing firmware on Windows, we recommend using the command-line version of Subversion. This is included in the TCAT Firmware Development Environment. If you use A GUI client, such as Tortoise SVN, it will not properly handle directory and file permissions in the Cygwin environment and you will get file access errors. **Also, do not mix use of the command-line and GUI clients.**

Logins

Once we know who will receive logins at your organization, we will create username/password combinations and relay the info. In general each company will have one login that everyone in the company should use. If the company has commit access to some of the repository (see below) then separate logins will be given to each individual who will be committing changes.

Tags

For our purposes, a "Tag" is simply a URL which points to a single directory on the server containing copies of relevant parts of the repository.

Each developer only needs to Checkout a single tag, which contains all of the source code and other files that correspond to the license agreement(s) that the developer's organization has in place with TCAT. We'll send the tag when we send your login information.

This tag is a secure http URL, such as:

https://dev.tctechnologies.tc/tcat/path/to/customers/tag/

The current release is always in your tag inside the "latest" directory. Aside from your tag, the other place in the repository that you can see is

https://dev.tctechnologies.tc/tcat/tags/release/public/

You'll start at these URL's, i.e. you can't browse down to them from a higher level web address such as https://dev.tctechnologies.tc/tcat/ . Note that these are secure-http URL's requiring "http**s**" in the address.

Browsing

You can browse files within your repository tag with a web browser like a normal URL to see what is contained there, and you can create shortcuts for URL's within, map them to Web Folders, local disk drive letters, etc. This may be convenient if you want to see the latest version of a PDF document without using a Subversion client to Checkout or Update first.

Initial Checkout and Updates

Using a Subversion client, Developers who wish to work with the sources should initially *Checkout* their "latest" tag into an empty directory, and then just *Update* from then on. You can look at the differences before updating as well to see what will change. See below for detailed steps.

As updates are committed by TCAT to the files in the repository, *the names of the tags will not change over time*. Your tag will always contain the latest release available from TCAT.

Previous releases are archived

When we make a new release and update your "latest" tag, a number of previous releases will still be available for your convenience. Before we update your tag, we copy it to a URL in your tag with the release number. For example, if your tag URL is

https://dev.tctechnologies.tc/tcat/tags/release/basic/latest

Then one of the previous releases will look something like:

https://dev.tctechnologies.tc/tcat/tags/release/basic/3.2.1/

3. Your local sources

Every organization has different development practices, so we don't have a lot of *requirements* for how to use the repository. However, here are a few things that may be helpful to know if you decide to use the repository for keeping up to date with TCAT.

Tags are read-only

Any changes you make to your local checked-out working copy cannot be committed back to the repository by customers.

Working copies

The source code is structured so that your product-specific code is usually separate from the core sources, and so your changed files shouldn't normally conflict or need merging when you Update from the repository.

Note that if you do develop your product out of your local checked-out working copies, Subversion, like its predecessor *cvs*, will generally try to merge our changes with yours and allow you to manually merge where there are conflicts. You will have to do this each time there are updates, however. Of course every developer has their own tried-and-true methods of doing this.

Common Code Change requests

If you have any changes you wish to make to the *common sources*, then please feel free to send them to TCAT. You can send a patch or the complete changed files.

TCAT co-development with Customer sources

There is one exception to the read-only tags rule mentioned above. If your company is co-developing a product-specific part of your sources with TCAT, then your tag will have some parts of its directory structure that *can* be committed back to the repository.

In all cases, *your custom sources* are kept in a separate part of the repository that is only viewable by TCAT and authorized developers at your company.

In most cases, these are files such as a firmware Project directory, host driver customization files, or host utility program of some kind.

It will not be obvious by looking at your files which ones can be committed back to the repository, however you can easily find out using the **svn proplist** command, or of course by asking TCAT. See below for how to find out which of your sources can be committed back to the repository.

Note that these external sources do not show up in a repository browser such as *Tortoise SVN*, or in **svn info –R** commands. This is normal, and is because the sources aren't actually copies in the tag's URL, but are copied to your local working copy by the Checkout command when it sees the *svn:externals* property in your tag.

4. Daily Use Guide

How-to:

1) Set up my initial Tag checkout

It's important to point out that you should make a new empty directory to keep your local files. The directory should not already have a ".svn" directory in it. In the Checkout example below, note the trailing "dot" This will copy the contents of your tag into the local directory specified. If you leave off the dot, it will create a folder in your directory and copy your tag into the folder, which may not be what you wanted.

When your tag is checked-out into this directory, it is now your working copy. The directory can be located anywhere, and projects (firmware and host) will always build as long as they are kept together inside of this working copy directory.

In the examples the user's home directory is used, but Windows users can replace **~/dev** with something like **c:/dev** if that is preferred, and we remind you that the Cygwin bash equivalent of this is **/cygdrive/c/dev**

The first time:

```
~ $ mkdir /dev
~ $ cd dev
~/dev $ svn checkout https://dev.tctechnologies.tc/tcat/path/to/my/tag .
```

Enter your login which you have obtained from TCAT. When finished, your new working copy is ~/dev

2) Keep up to date

Once you have checked out your working copy, you simply Update from then on to get the latest.

```
~ $ cd dev
~/dev $ svn status
~/dev $ svn update
```

3) Merge

You will want to merge whenever you want to synchronize changes in the repository with files that you are also making changes in. You can use your favorite diff/merge tools or use the ones included with the svn client.

When TCAT has a major point release, we may change the name of the tag. In that case we will let you know the best way to handle the new tag with the least amount of manual merging (if any).

4) Commit changes to co-developed sources

Go to the directory containing the sources you want to commit, and enter:

```
~/dev $ svn commit –m "my commit message"
```

If you want to diff first, you can use **svn diff** here or your preferred tool.

5) Determine which parts of my code are read-write in the repository

Code that you can commit changes to are, read-write for you and TCAT in your own separate location in the repository. These are linked in to your tag via "unrevisioned svn:externals properties," which gives you access to the active live sources in the repository.

To find out which of your sources are "live" go to your **working copy** directory and list the properties, e.g.:

~ $ **cd /dev**
/dev $ **svn proplist -v**

The files or directories containing files which can be committed will appear as *svn:externals*

For example, if an external applies to a directory, you'll see a result like:

```
svn:externals /firmware/project/productABC
https://dev.tctechnologies.tc/tcat/path/to/my/branch/firmware/project/productABC
```

and all of the files in your local **/firmware/project/productABC** directory are read-write in the repository.

If an svn:externals property specifies a file, then only the file specified is read-write.

6) Clear the authentication cache, use the repository as different user

On a Mac or Linux, the authentication cache is kept in the local user's home directory

```
~/.subversion/auth/svn.ssl.server
```

There will be a separate file in this directory for each repository you have accessed. Simply remove the file that corresponds to https://dev.tctechnologies.tc:443

You can find out which repository each file refers to, by looking at the last lines of the files in this directory.

[Note that some Mac clients use the keychain instead to store authentication data. In that case, follow the instructions included with the client to managed cached authentication data.]

On Windows, the cache is kept here:

"C:\Documents and Settings\<username>\Application Data\Subversion\auth\svn.ssl.server"

where "C:\Documents and Settings\<username>\Application Data" is the directory stored in %APPDATA% environment variable for your current Windows login.

With no cache files, the first svn command you use will ask again if you want to accept the server's SSL certificate and then will ask again for a login.

Using Tortoise SVN

Due to popular demand, this section is a short description for using the TCAT repository with Tortoise SVN, a Windows subversion client and shell extension.

For those who will be developing firmware on Windows, we recommend using the command-line version of Subversion. This is included in the TCAT Firmware Development Environment. If you use A GUI client, such as Tortoise SVN, it will not properly handle directory and file permissions in the Cygwin environment and you will get file access errors. **Also, do not mix use of the command-line and GUI clients.**

As a Widows shell extension Tortoise SVN adds menu items, for most svn client commands, to the Context Menu in Windows explorer. It also adds a graphical repository browser, as well as a number of utilities such as a visual diff and merge. Similar tools are available for other platforms.

You can find the installer for Tortoise SVN at http://tortoisesvn.tigris.org

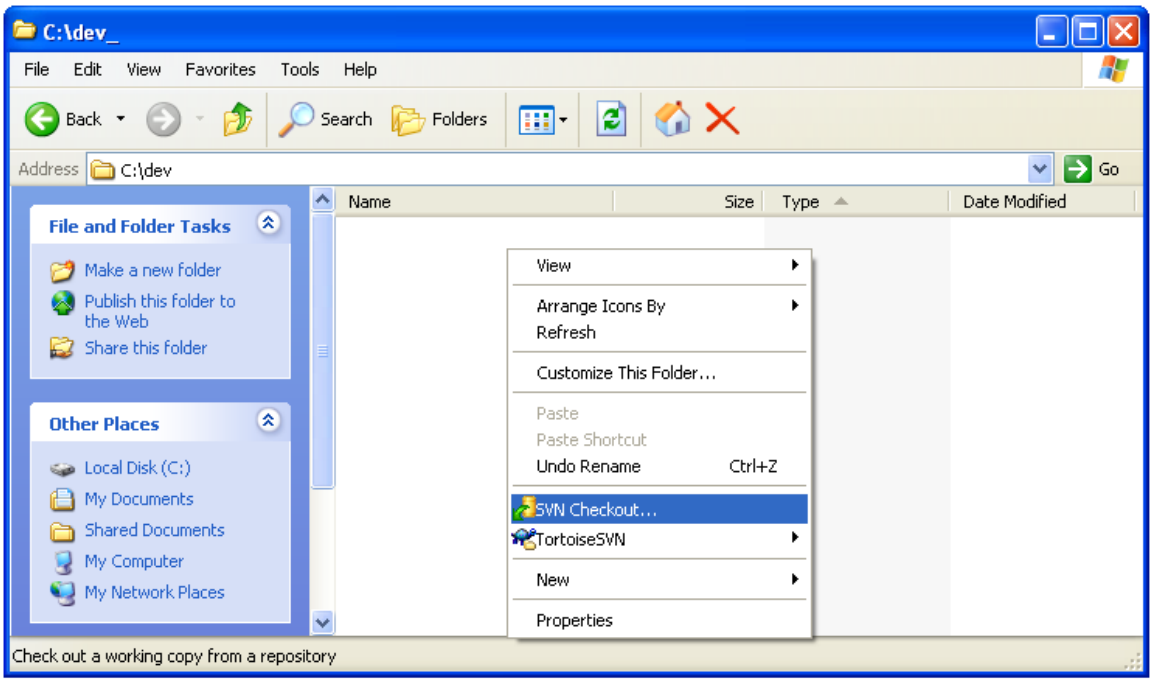
Tortoise adds different menu items to the windows context menu, depending on where and what you are right-clicking on. You can set preferences for this in the Tortoise Settings dialog, also of course available in the context menu. Tortoise Help is available this way also, and it's a good idea to read the Daily Use Guide, section 5 of the help file. The repository browser is also brought up from the context menu, and gives an explorer-like tree view of a repository.

Here are the How-to's from above, accomplished with Tortoise SVN:

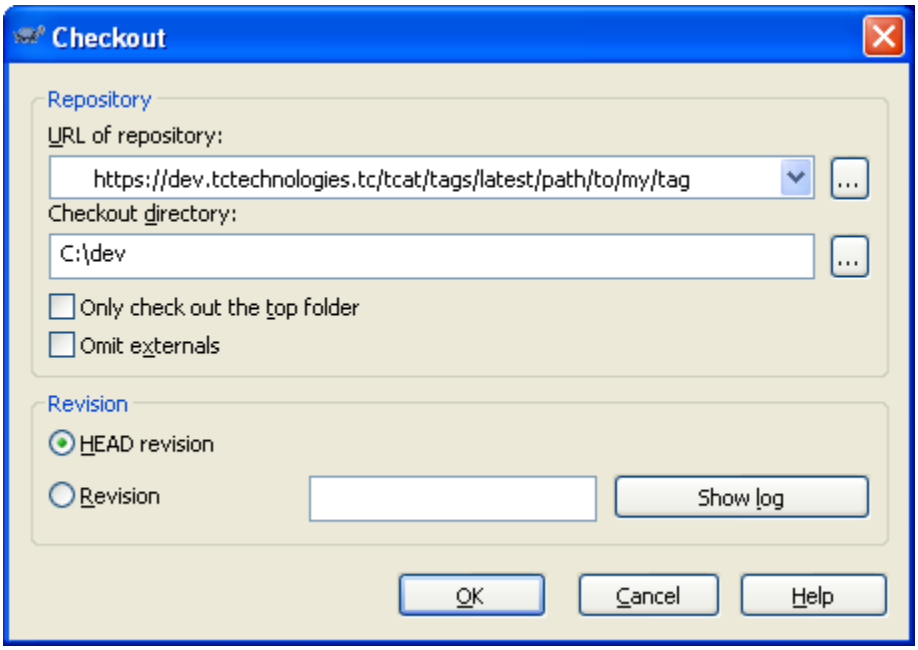
1) Set up my initial Tag checkout

This will set up the first-time checkout of a tag. In Windows Explorer, browse to an empty directory, such as **`c:/dev`**

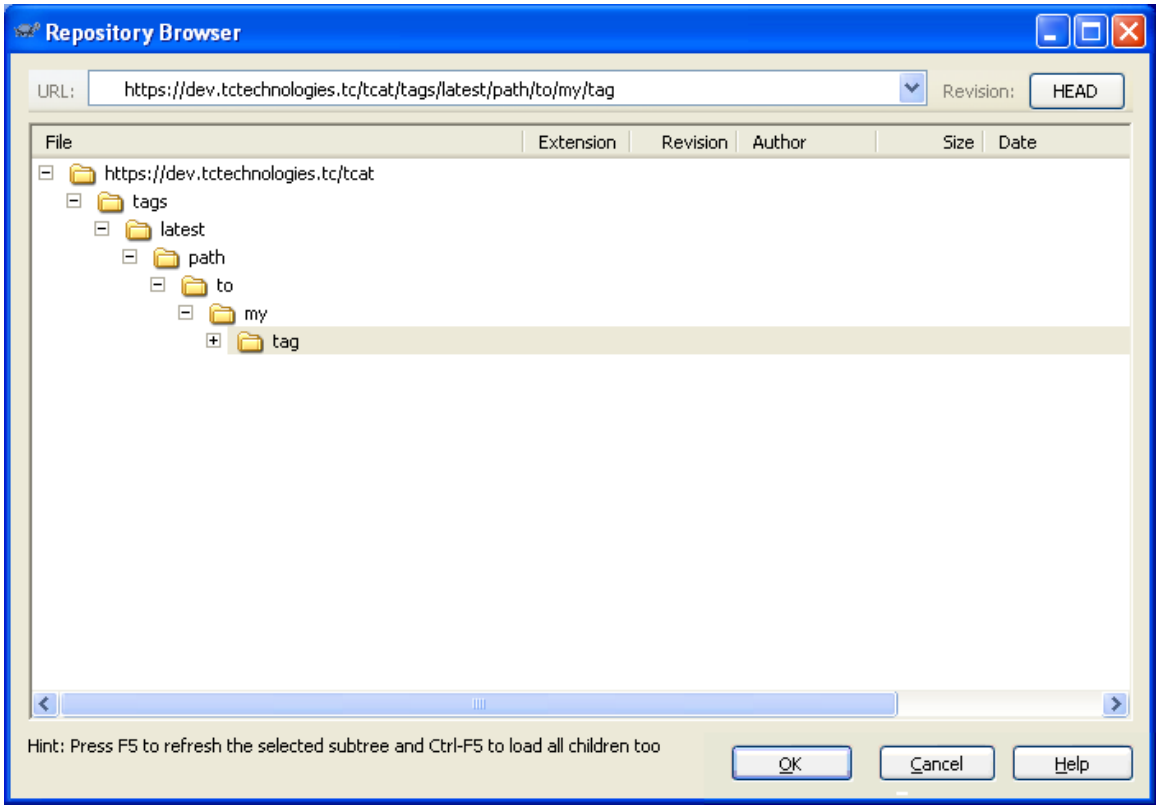
Right-click in the directory and choose "SVN Checkout …" from the popup menu.



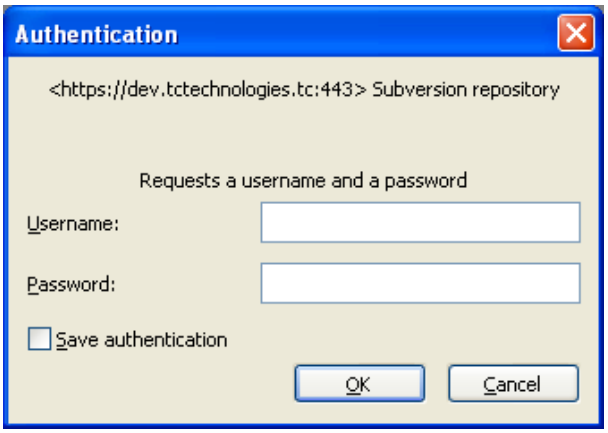
The Checkout dialog appears, and you can type in the URL of your tag, or select one from the drop-down history list.



From the Checkout dialog (or also from the context menu) you can bring up the Repository Browser to see the subdirectories *within your tag* to see what is there.

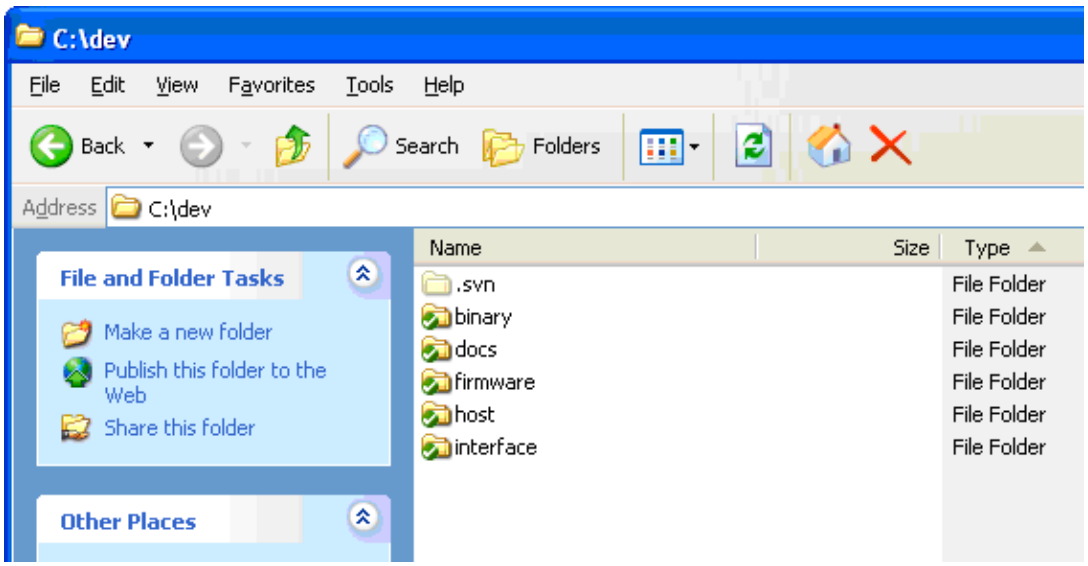

Repository Browser
URL:
https://dev.tctechnologies.tc/tcat/tags/latest/path/to/my/tag
Revision:
HEAD
File
Extension
Revision
Author
Size
Date
https://dev.tctechnologies.tc/tcat
tags
latest
path
to
my
tag
Hint: Press F5 to refresh the selected subtree and Ctrl-F5 to load all children too
OK
Cancel
Help

Then enter your login which was obtained from TCAT.



If you check the "Save authentication" checkbox, you will not have to enter this for future requests to the repository. This can be cleared, in case you want to remove this authentication data or use the repository with a different username. See below for info.

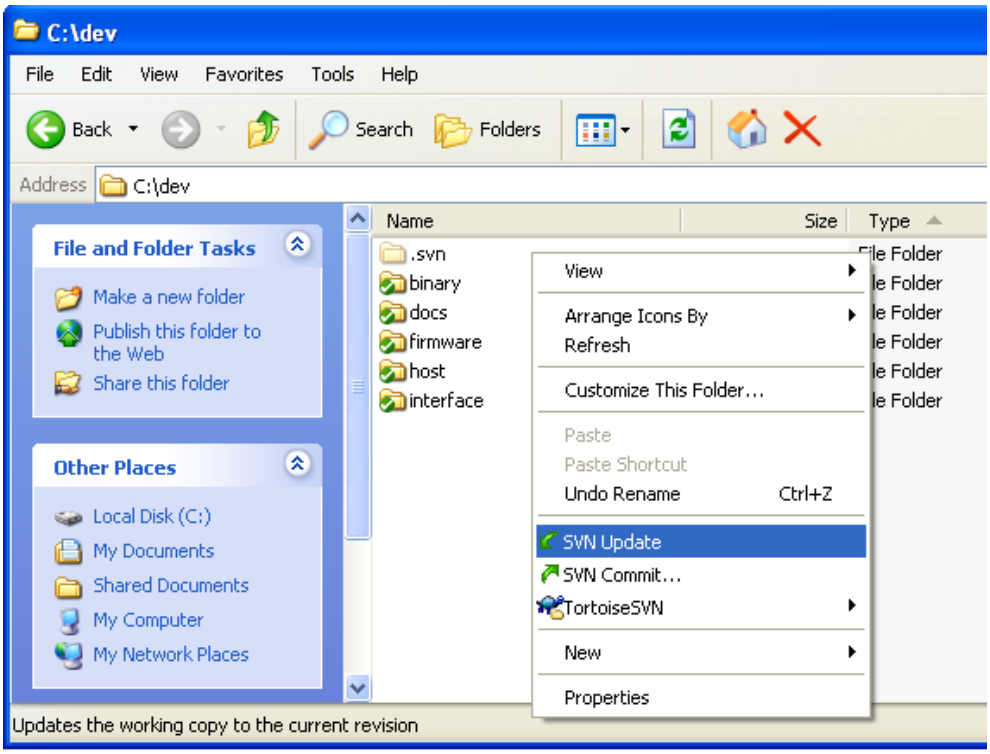
When you have authenticated, Tortoise then opens a message window and shows you the progress of the Checkout. When it's finished, your new working copy is **c:\dev**



Notice that the explorer icons now have icon overlays, or "badges," on them. These indicate the Subversion status of the files. The Green checkmark badges indicate a normal status. In the case of directories, that means that all of the files inside have normal Subversion status. Other status badges indicate whether files have been modified, are conflicted with the repository version, are scheduled for Add or Delete, etc. The Tortoise help file has all the details.

2) Keep up to date

Once you have checked out your working copy, you simply Update from then on to get the latest.



Note that if you have changed files under source control, Subversion will update those files as well, in which case it will try to merge them with your changes. If there are conflicts, the update log will tell you and you can decide how to resolve the conflict. See section 5 of the help file for details about resolving conflicts.

3) Merge

As mentioned above, you will want to merge whenever you want to synchronize changes in the repository with files that you are also making changes in. When your changes happen to the same place in a file that someone else has changed, you will have a conflict which needs to be manually resolved using the Tortoise Conflict Editor, which is just a merge tool. You can use your favorite diff/merge tools or use the ones included with Tortoise.

When TCAT has a major point release, we may change the name of the tag. In that case we will let you know the best way to handle the new tag with the least amount of manual merging (if any).

4) Commit changes to co-developed sources

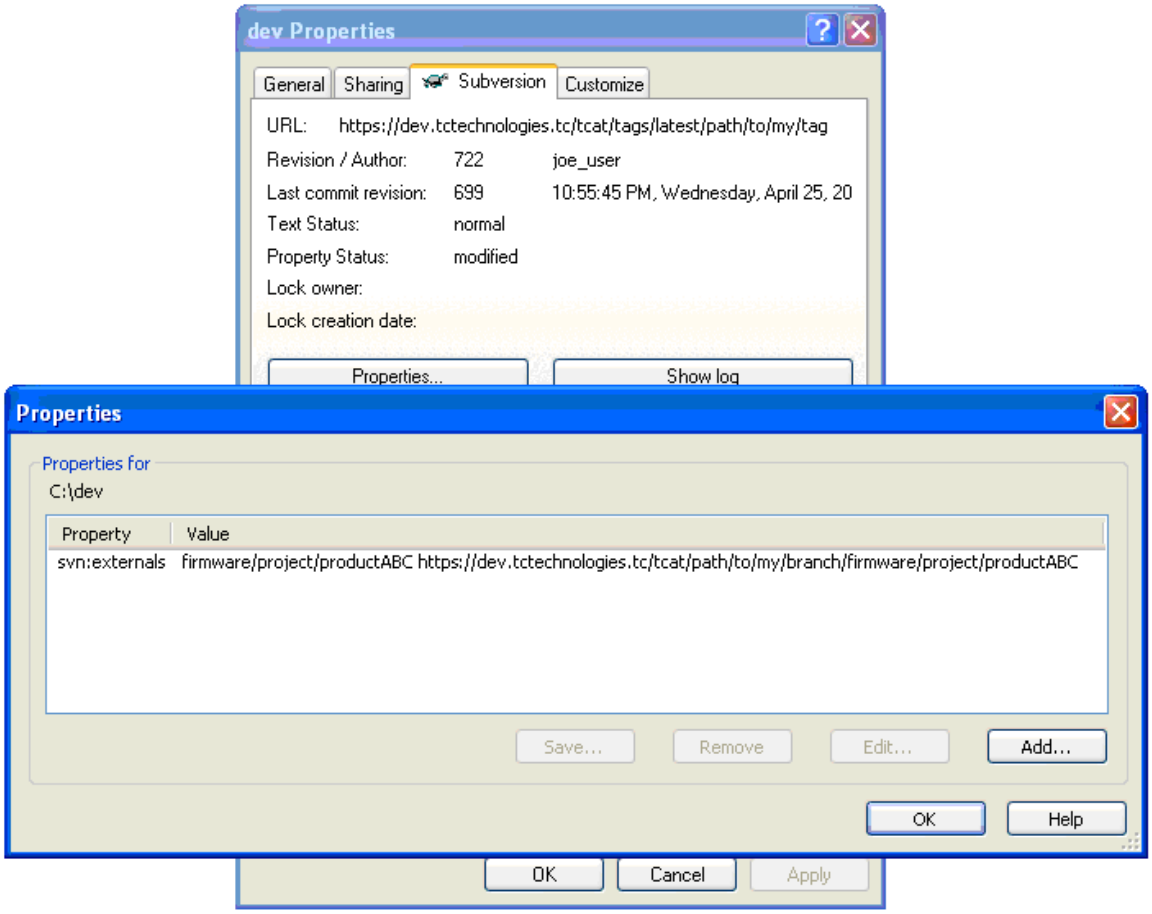
As mentioned above, right-click in the directory you want to commit, or on the particular file you want to commit. Then choose "SVN Commit …" in the context menu. A status log will appear and show you the result.

If you want to diff first, you can use the **diff** command from the context menu or your preferred tool.

5) Determine which parts of my code are read-write in the repository

Directories with code that you can commit changes to are read-write for you and TCAT. These are kept in your own separate location in the repository. These are linked in to your tag via "unrevisioned svn:externals properties" of your tag, which give you access to the active live sources in the repository.

To find out which of your sources are "live" go to your **working copy** and list the properties. In this case, right-click on the folder icon for **c:\dev** and choose Properties. You'll see the usual explorer property dialog for the folder, with an added Subversion tab. In the Subversion tab, click the properties button to bring up a window that lists all of the Subversion properties for the c:\dev directory.



The directories containing files which can be committed will appear as *svn:externals*. Here, we see two parts, the **firmware/project/productABC** part refers to a destination directory within the local working copy directory, and the https://dev.tctechnologies.tc/tcat/path/to/my/branch/firmware/project/productABC part describes a directory in the repository.

This means that when I Checkout my tag, the subversion client shall also go get that external directory out of the repository and put it in /firmware/project/productABC. Further, since the svn:externals property doesn't apply to a specific revision, I can commit my changes to these back to the repository and other developers will see the changes whenever they Update.

Note that this is a property of the tag I used to check out the files. Therefore, these properties will only show up when I get properties on the working-copy checkout directory **c:\dev**.

Note also that the Subversion tab in the Properties dialog shows from where the working directory was check out of the repository.

6) Clear the authentication cache, use the repository as different user

To do this, bring up the Settings dialog, using the context menu.

Select the "Saved Data" icon in the left pane. Then click the "Clear" button that shows up in the right pane.

This clears the local cached authentication files for the user who is currently logged in to Windows. With no cache files, the first svn command you use will ask again if you want to accept the server's SSL certificate and then will ask again for a login.