



User Guide

Installation and Configuration

About	Description
Revision	3.1
History	22/06/2022; Updated from 13/10/2020
Authors	Sean Kirwan; Jerry Clark; Peter Eszlari; Adrien Hayder

Overview

This documentation refers to release 4.6.5 r29721 and later.

Repository is an application that interfaces with a subversion repository to store and retrieve Swift projects and changes or updates to projects.

Repository is built around Subversion, a widely used centralised version control system. A Subversion repository is based on a versioning file system where only the changes to the base import (i.e. Swift project) are stored. Each change creates a new version along with a user description of the changes made and a list of the files changed.

Accordingly, it becomes possible to retrieve any version of the repository all the way back to the original import, offering users a flexible and safe means to deploy or rollback critical updates, at any point in time, whilst keeping the size on disc at a minimum.

Pre-Install Considerations

PC Functionality (Role) Types

Before running the Repository installer it is vital to decide what functionality is required on each PC - what role it plays in the Repository network. There are four possible PC roles each described below:

1. SVN Server
2. SVN Client
3. SVN User
4. SVN Distribution

Please note that it is entirely possible for a single machine to fulfil 2 or more of these roles and on smaller installations this is not uncommon. In larger installations it is more likely to have PCs dedicated to a particular role.

SVN Server

This PC runs the Subversion Server software and stores the Subversion repository. There is normally only a single SVN Server in a Repository deployment. Typically it is an existing PC file server with redundant PSUs, RAID disks and possibly a backup strategy in place but it can actually be just about any PC that has sufficient disk space for the repository. Initially the repository will be very small but as projects are modified and additional projects added it can grow quite large. Please ensure that at least 500GB will be available for the Subversion repository and consider what plans should be in place to ensure data security in the event of a hardware failure.

SVN Client

This is a PC that just needs to have its Swift projects updated remotely and requires no other Repository functionality. Typically these are dedicated TX machines - i.e. a Grass Valley ICE or a Render head.

SVN User

This is a PC that needs to update the Subversion repository. Typically these will be Edit workstations used to modify and create Swift projects.

SVN Distribution

This is a PC that may require SVN User functionality but also needs to be able to distribute Swift projects to the SVN Client PCs. As Swift project distribution is updating PCs that may be "live" this PC is normally a Sys Ops or Supervisors machine.

Repository User Roles

Administrator

Responsible for administering the server, adding users and machines, doing backups etc... Typically will be using a machine of type SVN Server & SVN Distribution

Designer

Responsible for creating/modifying Swift projects and graphics. Typically will be using a PC of type SVN User

System Operator

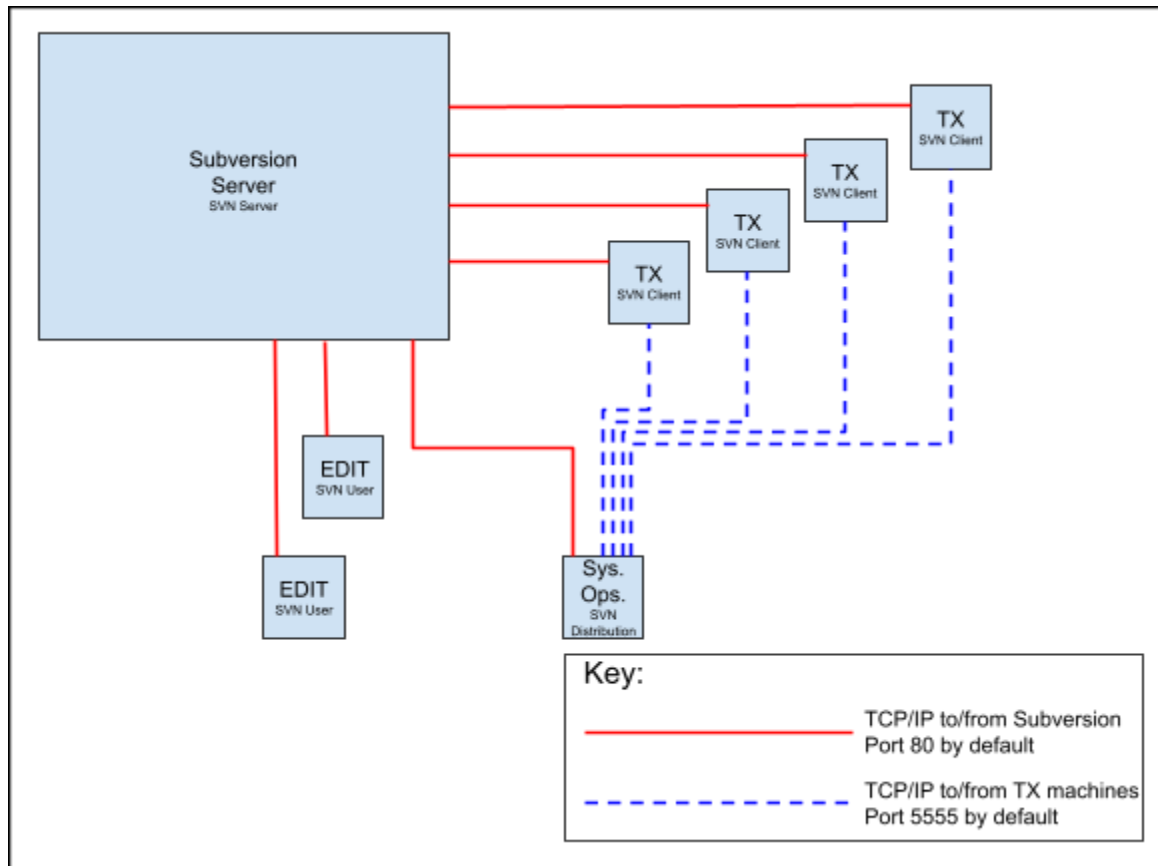
Responsible for updating the TX machines with the correct project version from the repository. Typically will be using a PC of type SVN Distribution

Network Considerations

The PCs in a Repository deployment need to communicate with other PCs in the deployment. The communication requirements are:

1. PCs designated as SVN Client and/or SVN User and/or SVN Distribution need to be able to initiate communication with the SVN Server. The communication is done over port 80 by default but can be configured to a different port number.
2. The PCs designated as SVN Distribution need to be able to initiate communication with the PCs designated as SVN Client. The communication is done over port 5555.
3. It is quite common in a broadcast environment for PCs to exist on 2 or more networks and it may be necessary to configure "tunnels" through various firewalls to allow the required communication to take place.

The diagram on the next page illustrates the communication required between PCs in a Repository deployment.



Network connectivity problems may include:

- The TX PCs, those PCs directly responsible for producing the video/audio output, are usually on their own highly secure “broadcast” network fire-walled off from the corporate network. These are the SVN Client PCs. These PCs need to be able to receive commands from the SVN Distribution PCs and pull down Swift projects from the SVN Server PC.
- The Edit workstation PCs are likely to be on the corporate network with no access to the broadcast network. These are the SVN User PCs. These PCs need to push/pull projects to/from the SVN Server PC.
- The “Sys Ops” PCs could be on either network but will probably have access to both the broadcast and corporate network. These are the SVN Distribution PCs. These PCs need to be able to query the SVN Server PC and send commands to the SVN Client PCs.
- The Subversion server PC is likely to be on the corporate network with no access to the broadcast network. This is the SVN Server PC. This PC needs to be able to serve/receive Swift projects to/from the SVN User PCs and serve Projects to the SVN Distribution PCs and the SVN Client PCs.

Installation

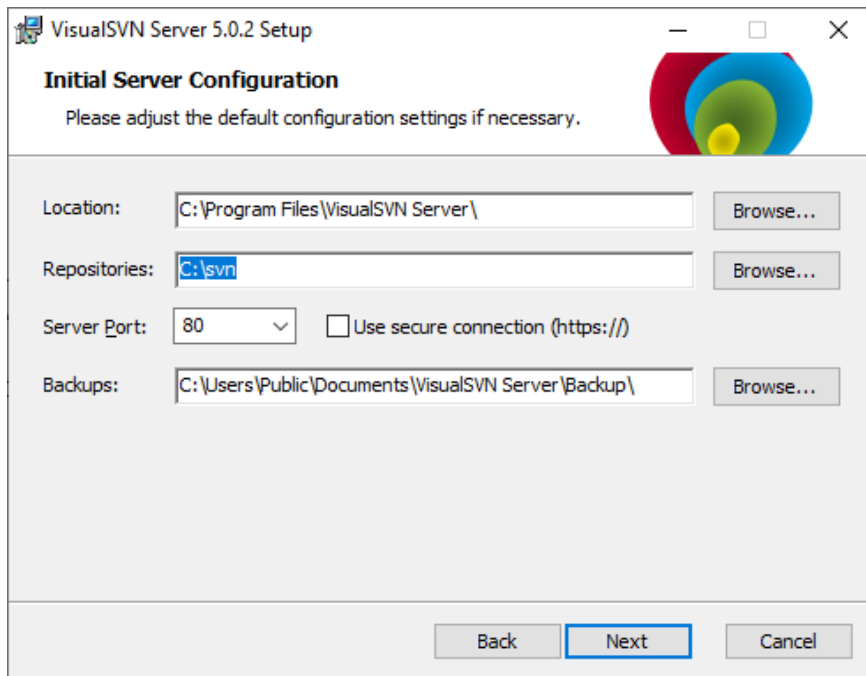
Required Installers

Ensure you have the appropriate installers available:

1. VisualSVN-Server-5.0.2-x64
 - i. This can be downloaded from <https://www.visualsvn.com/downloads/>
2. TortoiseSVN-1.7.9.23248-x64-svn-1.7.6.msi (optional - not required)
 - i. This can be downloaded from <https://tortoisesvn.net/downloads.html>
3. The latest version of Repository
 - i. Currently TogRepository-4.6.5_r29721-windows-installer.exe
4. The latest version of DataClient
 - i. Currently DataClient-5.4.3_r29736-windows-installer.exe

SVN Server Installation

- Double-click the VisualSVN-Server-5.0.2-x64
- Once you get to this screen, make sure the settings are as below:
 - "Repositories" set to "C:\svn"
 - "Use secure connection" is unchecked



SVN User & Distribution Installation

IMPORTANT: This install requires the SVN server to be available. Please ensure that the PC that acts as the SVN server has had SVN installed and configured prior to running this install.

- Double-click the *TogRepository-r[latest]-windows-installer.exe*

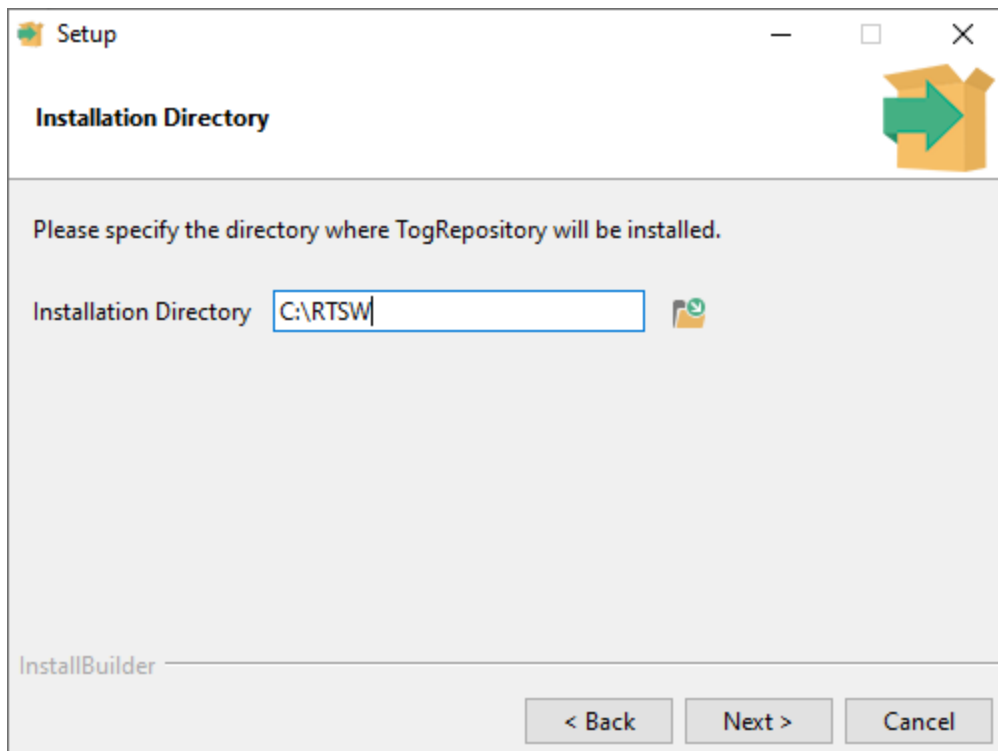
NOTE: on running the installer the first prompt is to specify the installation directory. The default location is the RTSW folder located on the users Desktop which is normally: *C:\Users\<username>\Desktop\RTSW*.

The RTSW folder will be created if it does not exist.

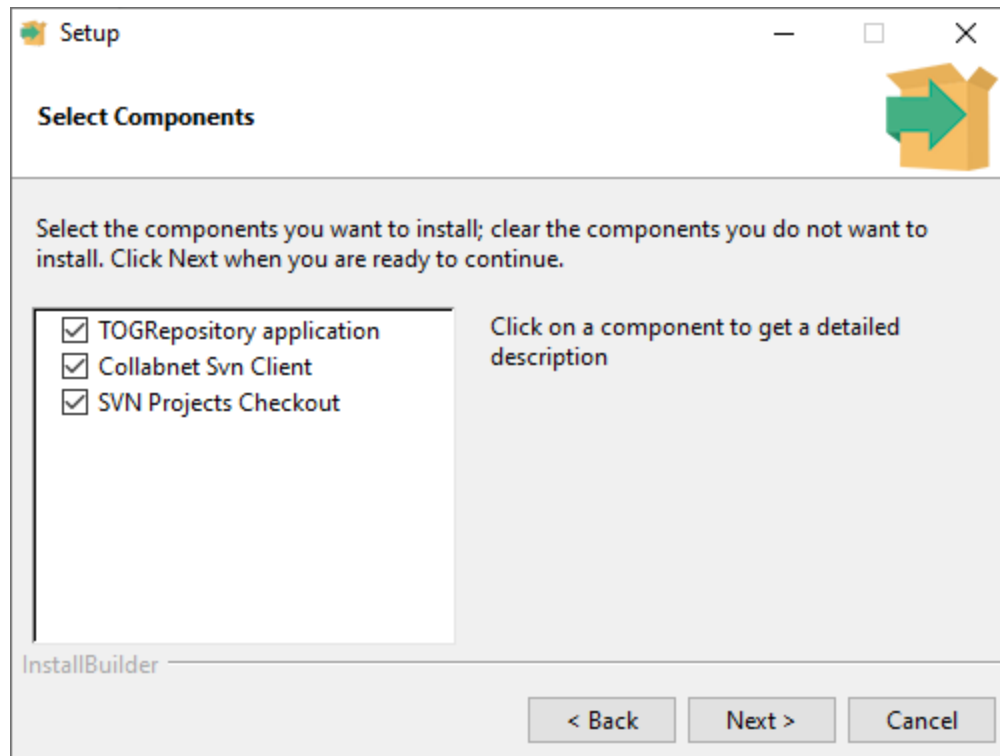
The installation directory can be changed, as shown in the example below. RTSW can be named differently, however, it is recommended to leave it unchanged to ease future maintenance.

NOTE that for each project in the SVN server that Repository will distribute will have to be a copy in this location. Make sure that there will be enough space here for future project checkouts.

Accept the Licence Agreement, select an installation directory, then click Next.



Select the options required for this installation, then click Next. Please note that DataClient is now a separate installer.



The options selected above will install the following:

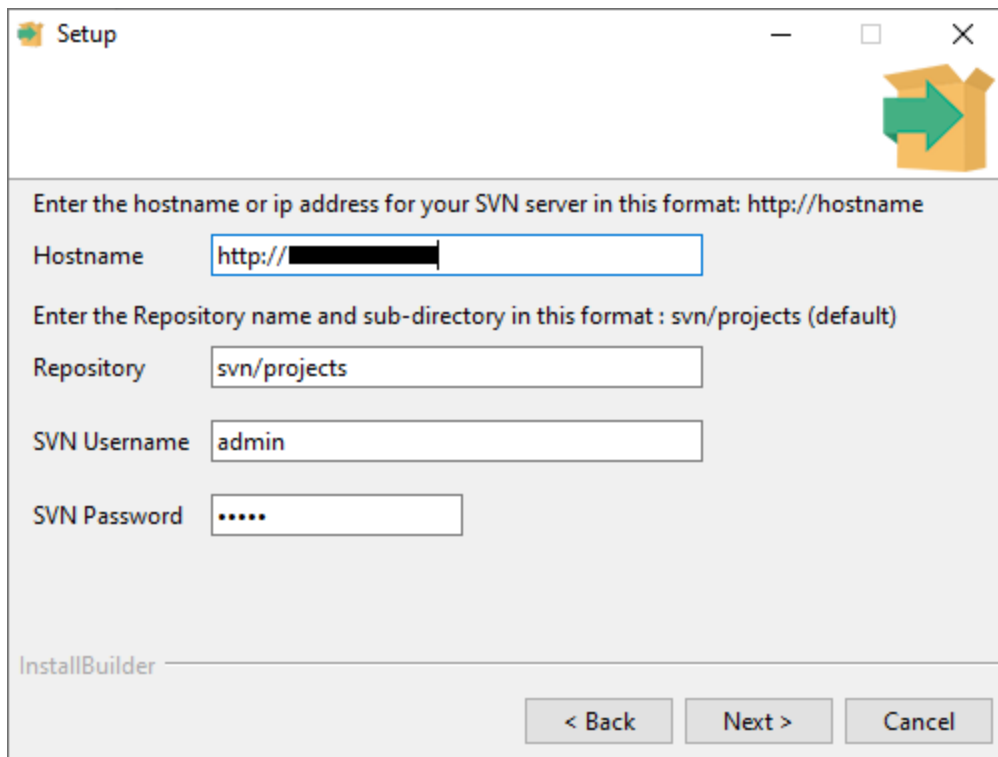
Repository

- A rootfolder, default is RTSW, it will be merged if it already exists.
- \bin\TOGRepository with executables
- \conf with a TOGRepository.conf file (see section 6.3)
 - and a TOGRepository_sample.conf file as a backup of the conf file structure
- \icons : for the app shortcut icon
- \logs : a tr.log file will be updated here in runtime. This can be deleted from time to time to save space
- \TOGRepository : it's a workspace for the application to update some files dynamically.
- Uninstallers : holds the uninstaller executable that updates at each new component installation

Collabnet SVN Client:

Installs the Collabnet SVN client, currently CollabNetSubversion-client-1.11.0-1-x64.exe. This will provide command line SVN features.

Fill in the fields, click Next



Setup

Enter the hostname or ip address for your SVN server in this format: http://hostname

Hostname

Enter the Repository name and sub-directory in this format : svn/projects (default)

Repository

SVN Username

SVN Password

InstallBuilder

< Back Next > Cancel

Hostname

The host name or IP address of your SVN server. Should be obtained from the steps in chapter 4.1.2

Repository

The default is svn/projects. The svn part must not be changed.

projects is the name that was given to the repository in the server. Changing this will not change the name of the projects folder in the install directory, only the address of the svn target.

SVN Username

Default is admin

SVN Password

Default is admin

NOTE: This password and a set of users can be configured in the VisualSVN Server setup. The user and password details in the installer only refer to the projects folder checkout at the installation stage. Repository will use the details that can be found and edited in the `.conf` file.

TortoiseSVN Installation (optional)

Subversion is normally driven via the command line. TortoiseSVN is an application that allows Subversion to be driven via a user-friendly GUI. It is not actually necessary as Repository provides everything necessary but some Designers/Operators may want to use some more advanced features of Subversion.

When installing TortoiseSVN it is important to **ensure that the command line tools are NOT installed** as they are supplied by Repository.

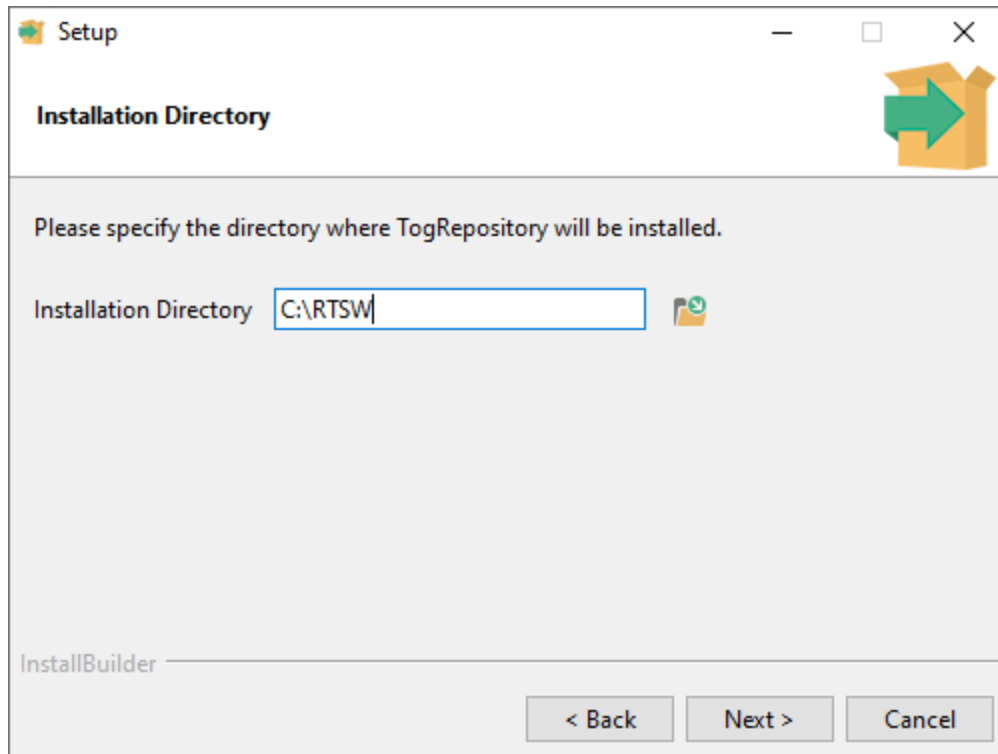
SVN Client Installation

IMPORTANT: This install requires the SVN server to be available. Please ensure that the PC that acts as the SVN server has had SVN installed and configured prior to running this install.

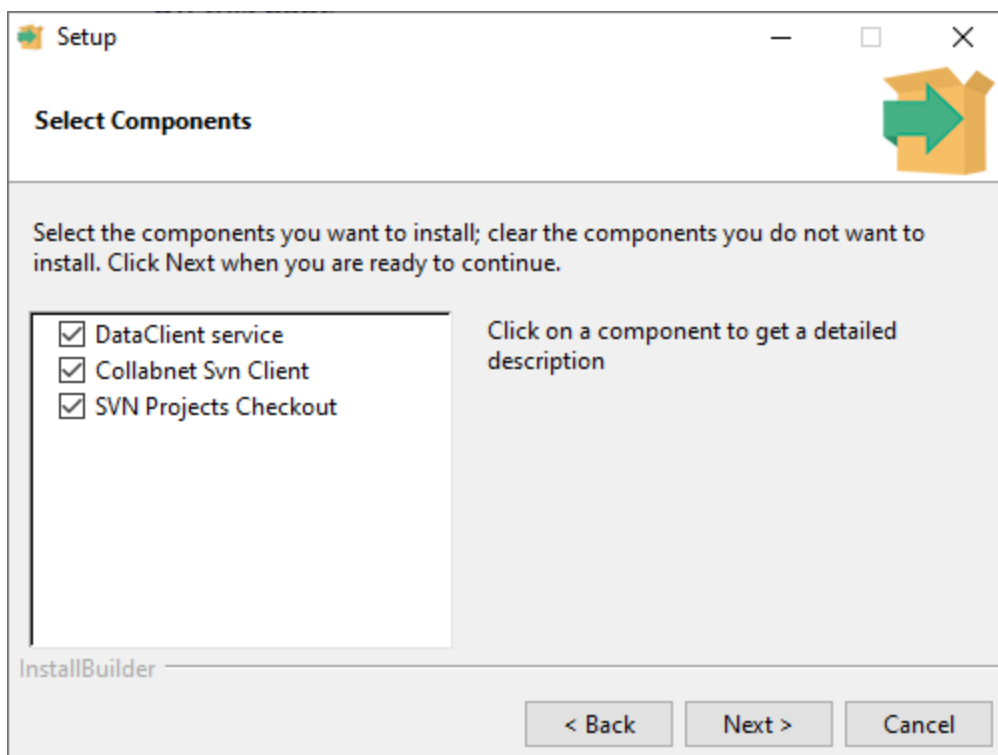
- Double-click the *DataClient-r[latest]-windows-installer.exe*

Select an installation directory

This is where the DataClient service application and the local copy (checkout) of the projects' repository will be located.



Select the options required, then click next.



NOTE: In some Windows systems DataClient and Collabnet SVN Client cannot install clean in a sequence and an error message is returned. In this case, uninstall with the uninstaller and run the installer separately for these two features.

In some other cases, the Microsoft Visual C++ Redistributable might not install properly, if this happens, uninstall Microsoft Visual C++ 2015-2019 Redistributable as well as DataClient, and then install DataClient again.

These options will install the following:

DataClient service

- A rootfolder, default is RTSW, it will be merged if it exists.
- \bin\TOGDataClient with executables and 3 batch scripts that can be used to control the Windows service that has been installed and running. Right click on the file and select Run As Administrator from the menu.
 - togDataClient_DELETE.bat, togDataClient_STOP.bat, togDataClient_START.bat
- Another script, togDataClientService.bat was used by the installer to install the service.
- Uninstallers : holds the uninstaller executable that updates at each new component installation

Collabnet SVN Client

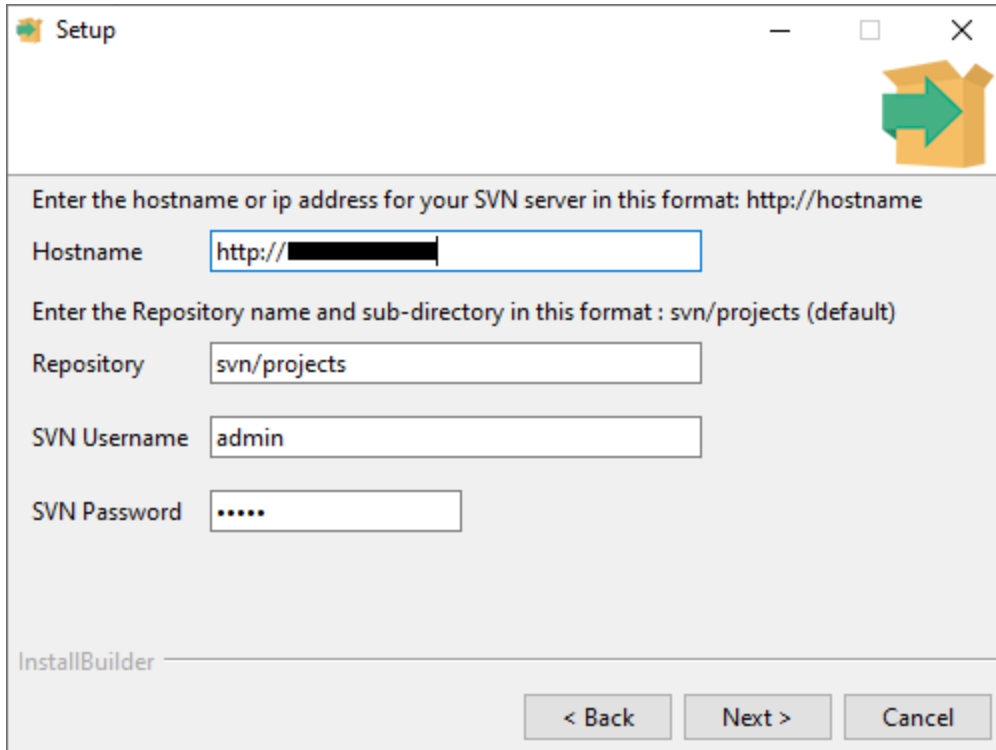
Installs the Collabnet SVN client , currently CollabNetSubversion-client-1.11.0-1-x64.exe. This will provide command line SVN features.

SVN Project Checkout

Creates a /projects folder where an empty checkout of the server repository will be initialised. The repository address will have to be specified on the next page.

The following page is only needed if **SVN Project Checkout** is selected, for which the Collabnet Svn Client is needed.

If so, fill in the fields and click Next.



Setup

Enter the hostname or ip address for your SVN server in this format: http://hostname

Hostname

Enter the Repository name and sub-directory in this format : svn/projects (default)

Repository

SVN Username

SVN Password

InstallBuilder

< Back Next > Cancel

Hostname

The host name or IP address of your SVN server. Should be obtained from the steps in chapter 4.1.2

Repository

The default is *svn/projects*. The *svn* part must not be changed.

projects is the name that was given to the repository in the server. Changing this will not change the name of the projects folder in the install directory, only the address of the svn target.

SVN Username

Default is *admin*

SVN Password

Default is *admin*

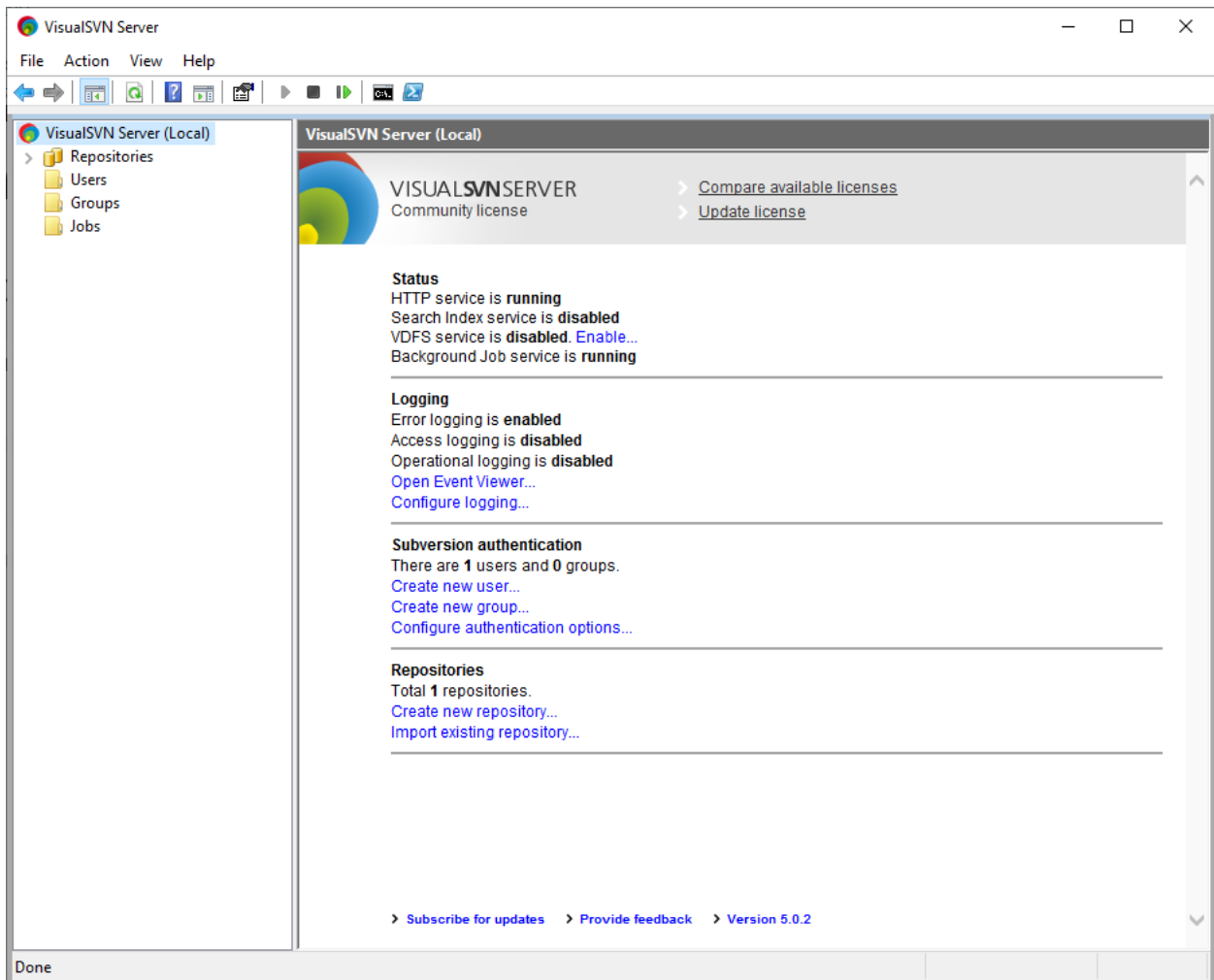
NOTE: This password and a set of users can be configured in the Collabnet SVN Server setup. The user and password details in the installer only refer to the projects folder checkout at the installation stage. Repository will use the details that can be found and edited in the .conf file.

Subversion Server Administration

Setting up a windows Subversion server

The subversion server would be installed on a central machine with fast network links and disks but otherwise no special hardware. See: <https://www.visualsvn.com/server/>

Subversion runs as a service which is managed using a Windows Computer Management window called VisualSVN Server Manager.



Creating a user account

First a user account must be created. Right click on Users and click Create User. It is

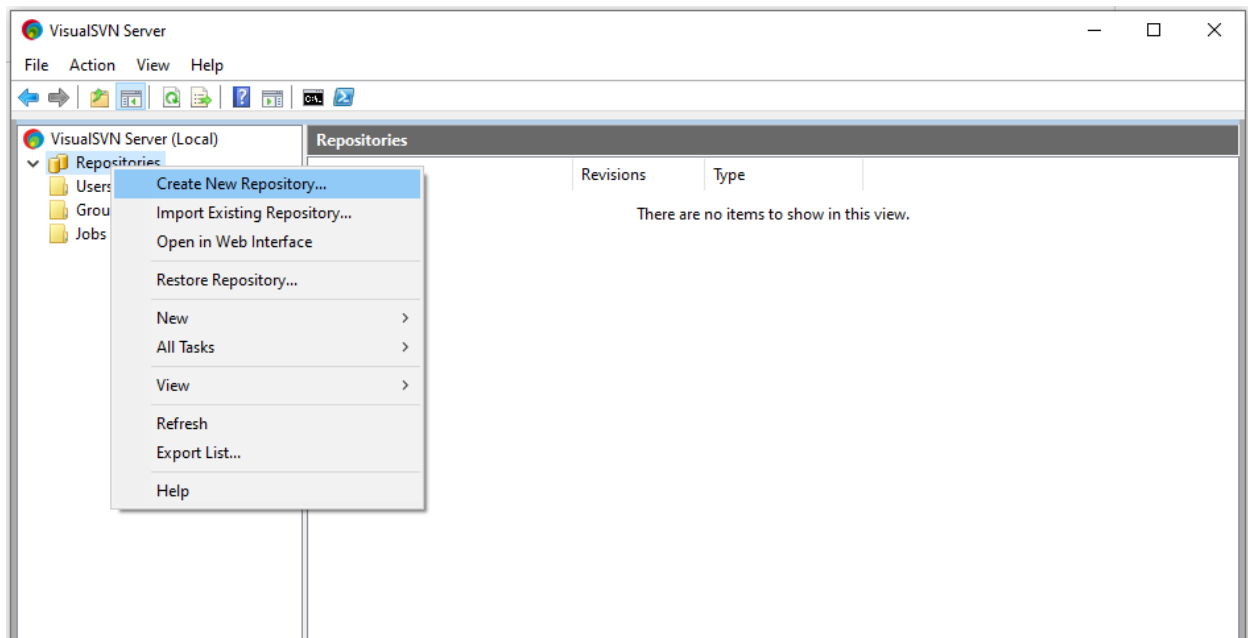
recommended to use the following details:

Username: **admin**

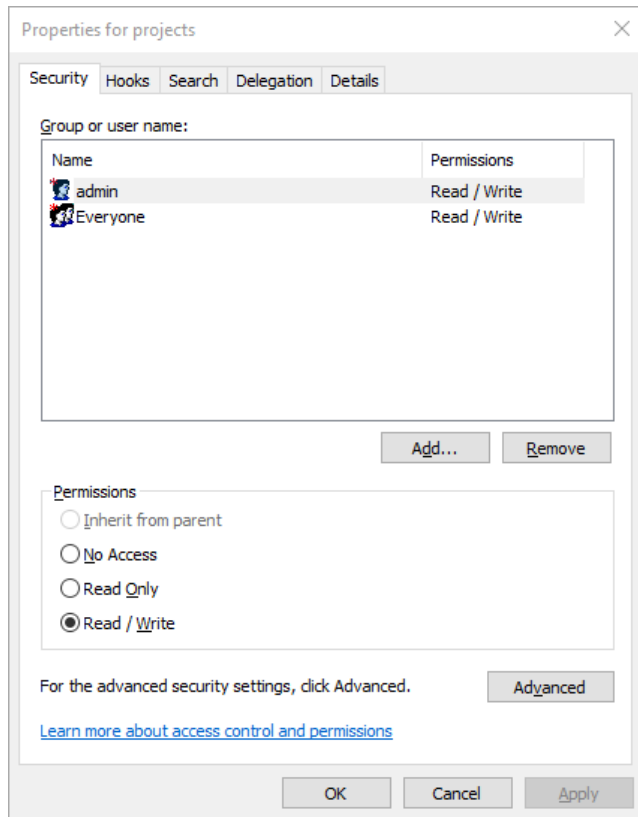
Password: **admin**

Creating a Repository

Repositories can be created, deleted and dumped. Only one repository is recognised by Repository and that must be called “projects”.



- Right-click on the Repositories menu button
- Click Create New Repository
- Select the appropriate type of repository (FSFS for most cases, VDFS if using VisualSVN replication - see section 4.4 for detail)
- Name your repository **projects**. **Only use this name, all lower case.**
- Select Empty repository
- Finish creating repository
- Right click the projects repository and select Properties
- Ensure that the user created earlier has read/write access



Administration

The administrator can create and delete users as well as setting access permissions. These can be quite detailed - giving users read or write access to the whole or parts of a project. This is generally only used by advanced users.

For example, a junior designer might have write access to the **GMScripts** folder of a project but not the **GMData folder** - meaning the designer could make up new graphics but not change the look of existing graphics.

Scheduling backups

VisualSVN can automatically create backups for the repository, however this requires a separate licence. More information can be found [here](#).

Repository Users

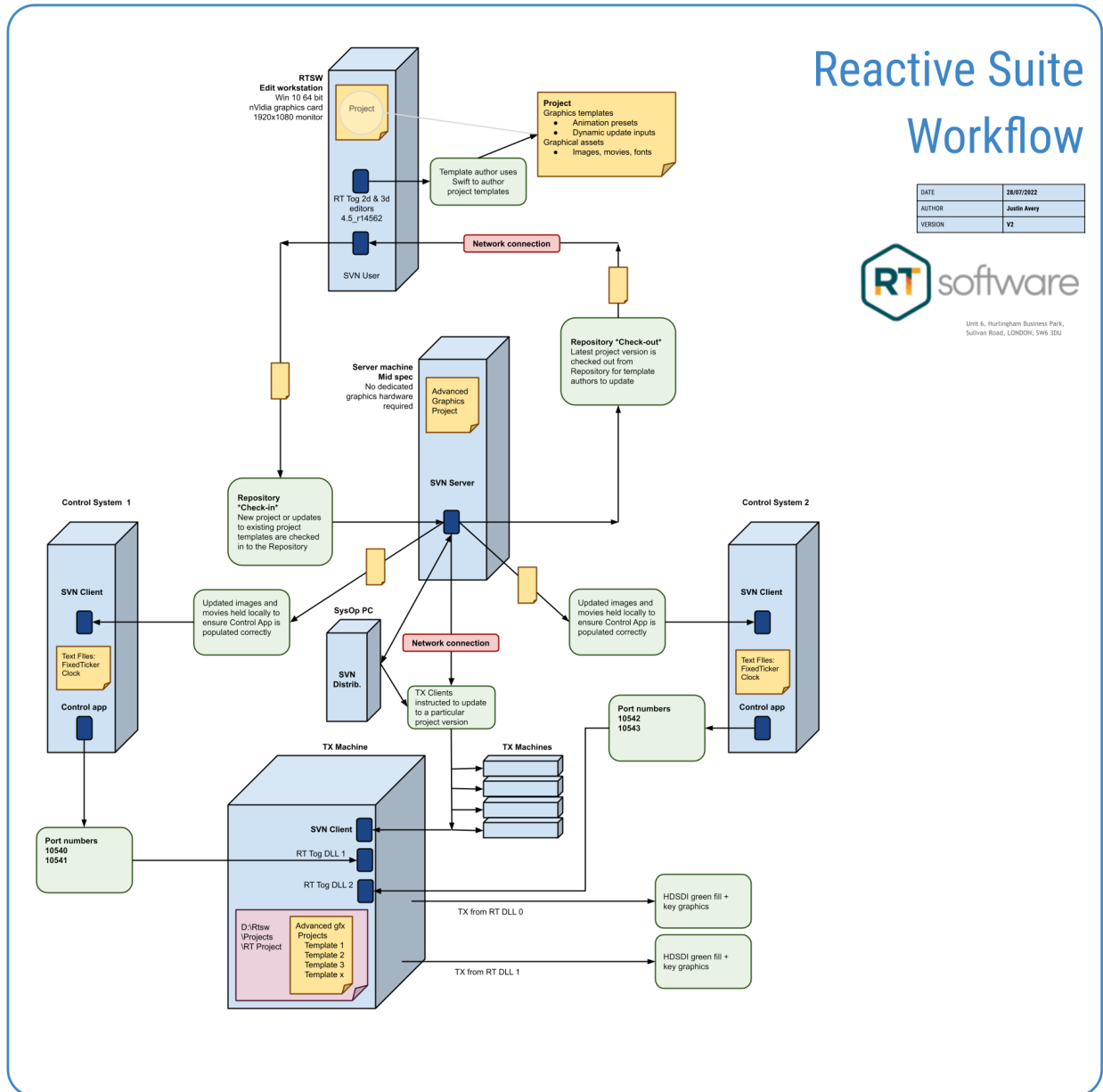
There are three users involved:

- **Administrator** - the administrator uses the VisualSVN MMC snap-in to create and administer the central repository (add new users and set their access rights, etc.),
- **Designer** - the designer creates advanced graphics projects using Swift and stores these in the central repository using Repository.
- **Operator** - the operator uses Repository to distribute graphics projects from the central repository to renderers when necessary.

Repository workflow example

1. **Designer** checks-out subversion project from the repository
2. **Designer** modifies project
3. **Designer** checks-in the modified project into the repository
4. **Operator** distributes the modified project to the relevant TX PCs
5. **Operator** synchronises the TX PCs

Repository Workflow principles



Setting up Repository Replication

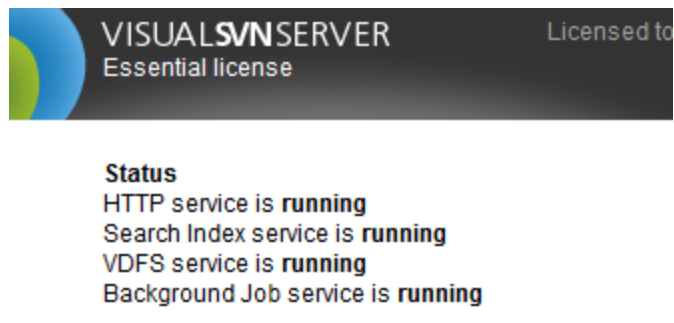
VisualSVN can provide repository replication through its VisualSVN Distributed File System (VDFS). This section will give a brief overview of setting Replication up for Repository, for more information please refer to documentation on the [VisualSVN website](#).

Overview & Prerequisites

VDFS works on a “Master/Slave” model, where one “master” is always available to write to and read from, and one “slave” is writable to so long as there is an active connection to the “master” server, and are readable regardless of the connection to the master.

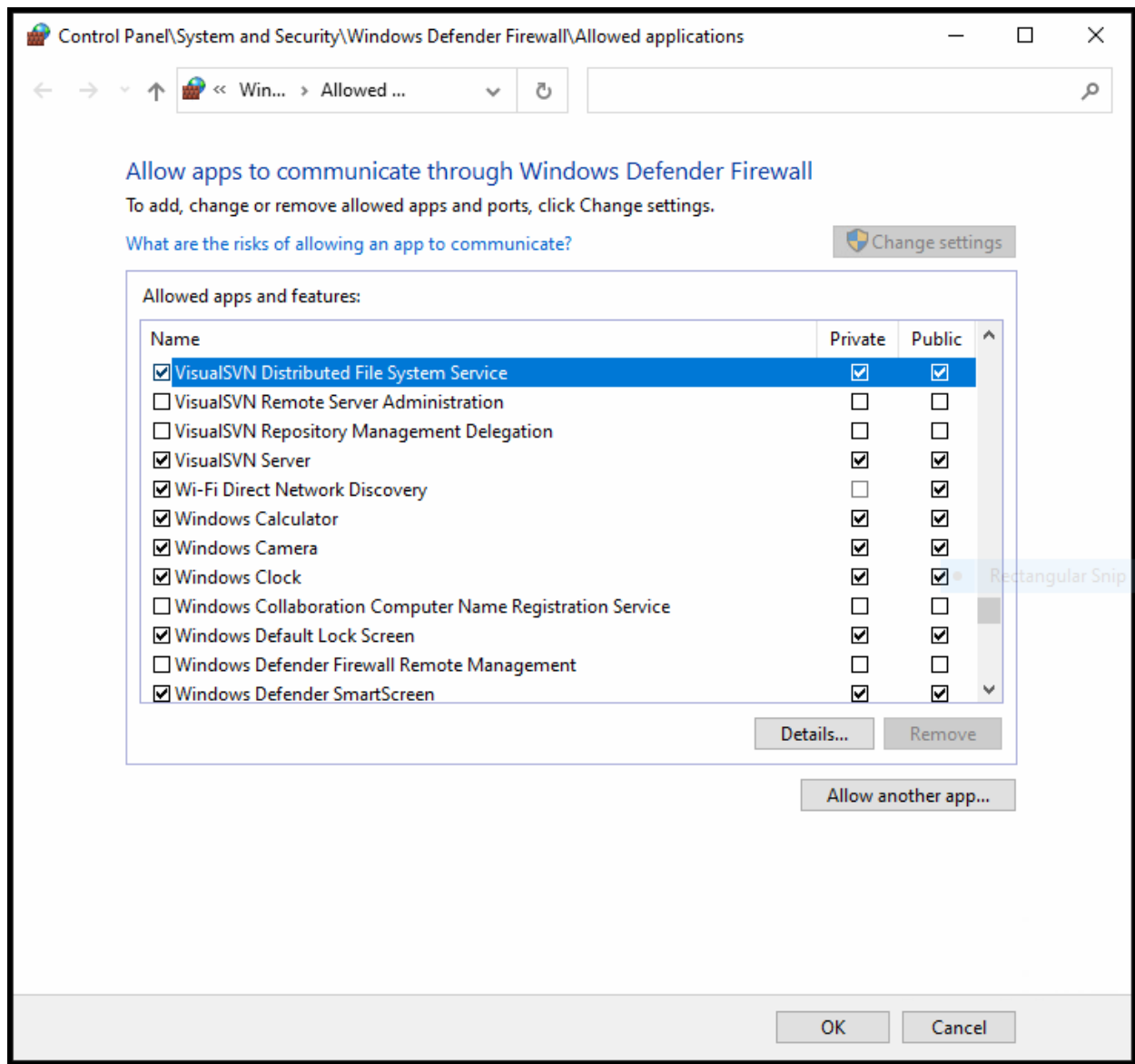
Before setting up replication, ensure the following:

- Every instance of VisualSVN is version 3.7 or higher
- Every instance of VisualSVN has an Essential, Enterprise, or Evaluation licence
- Every instance of VisualSVN has VDFS enabled
 - From the VisualSVN Server Manager snap-in for MMC, click enable VDFS service from the first screen.



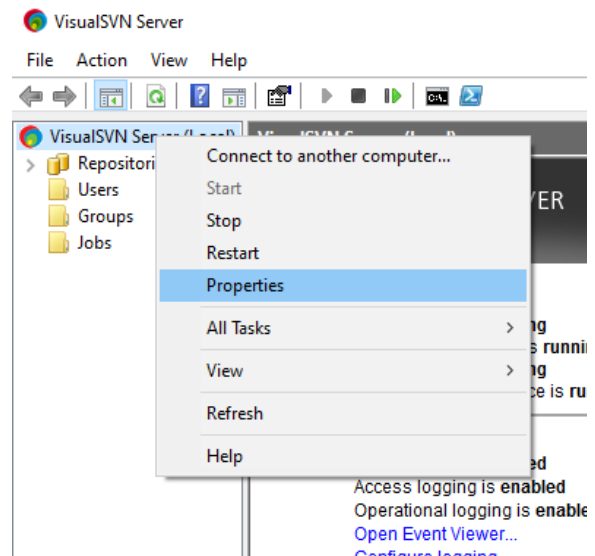
Enable VDFS Inbound Firewall Rule

On the “Master” server, open the Control Panel, go to Windows Firewall, and click “Allow a program or feature through Windows Firewall”. Click “Change Settings”, then scroll down to VisualSVN Distributed File System Service and check the appropriate boxes.

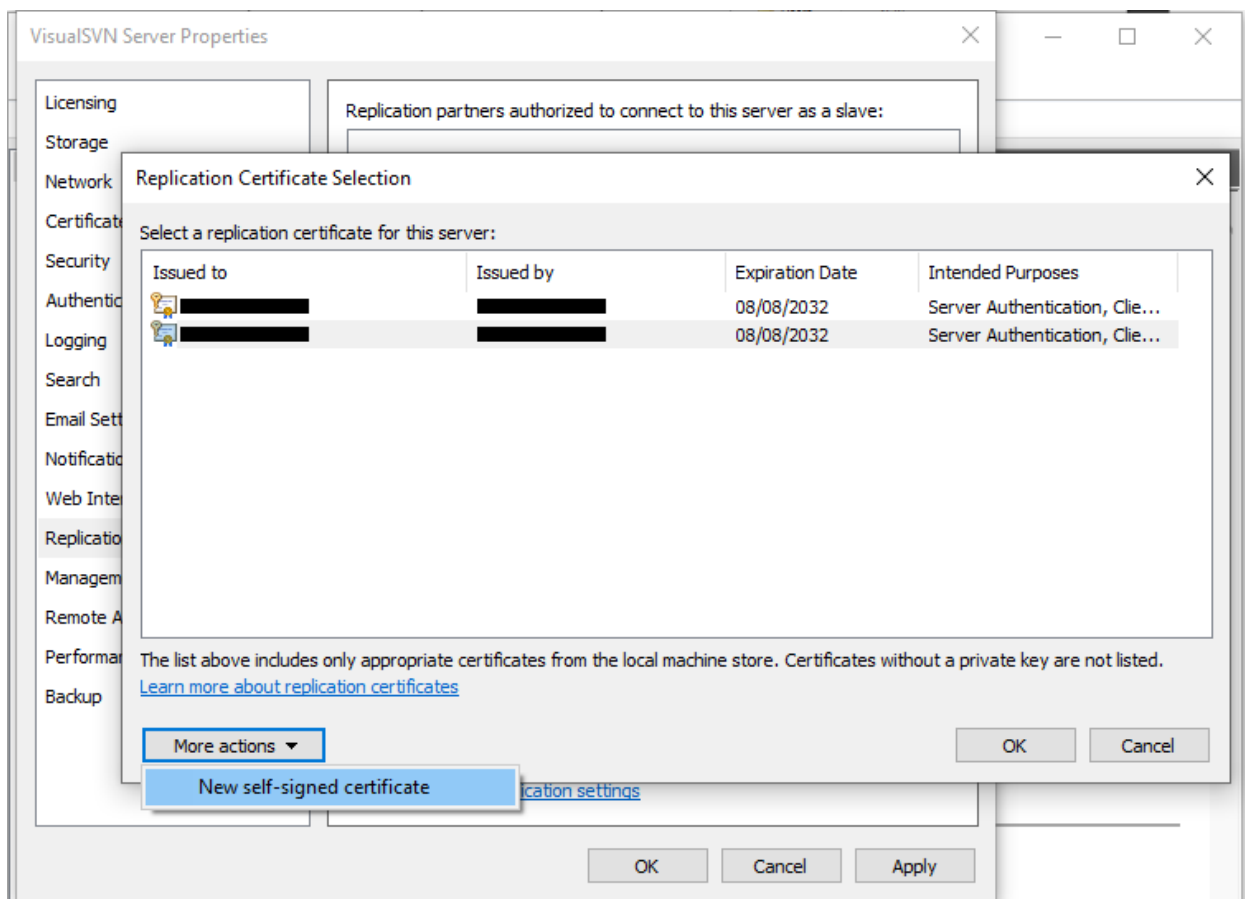


Enable certificate based replication

- Right click on the VisualSVN Server at the top of the tree on the left of the MMC snap-in and click Properties.
- In the Properties window, click on the Replication tab and enable certificate-based authentication for replication. Click on "Select Certificate", then click on "More Actions", and select "New self-signed certificate".
- Enter the hostname of the server, hit next, fill in any additional information needed for the certificate, and then hit "create".
- Select the newly created certificate and click "OK", then click "Apply".



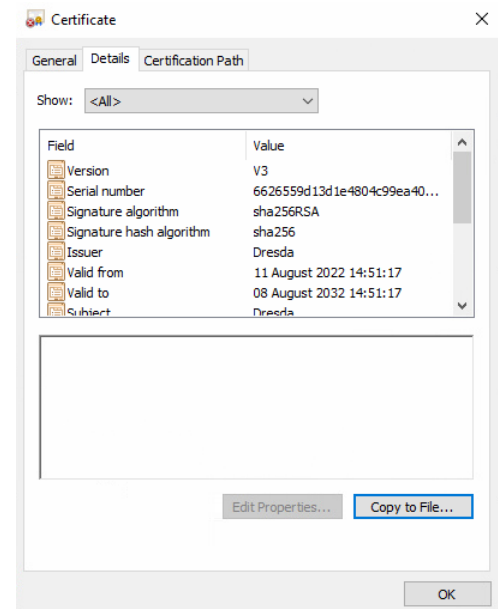
Do this on the "Master" and "Slave" machine



Make certificates trusted on “Master” server

To make the “slave” machine trusted on the “master” server, the self-signed certificate from the “slave” needs to be imported to the “master” server’s trusted certificates. To do this:

- Open the VisualSVN Manager snap-in on the “slave” machine and go the Replication tab of the Server properties
- Click “View Certificate”, go to the details tab, and click “Copy To File”
- In the dialogue window that opens, first hit “Next”, then select Base-64 encoded X.509 (.CER), then hit next again
- Enter in the certificate’s name, we recommend you use the machine’s hostname for simplicity, and export the certificate to file.



← Certificate Export Wizard

Export File Format

Certificates can be exported in a variety of file formats.

Select the format you want to use:

- ☐ DER encoded binary X.509 (.CER)
- ☒ Base-64 encoded X.509 (.CER)
- ☐ Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)
 - ☐ Include all certificates in the certification path if possible
- ☐ Personal Information Exchange - PKCS #12 (.PFX)
 - ☐ Include all certificates in the certification path if possible
 - ☐ Delete the private key if the export is successful
 - ☐ Export all extended properties
 - ☐ Enable certificate privacy
- ☐ Microsoft Serialised Certificate Store (.SST)

Next

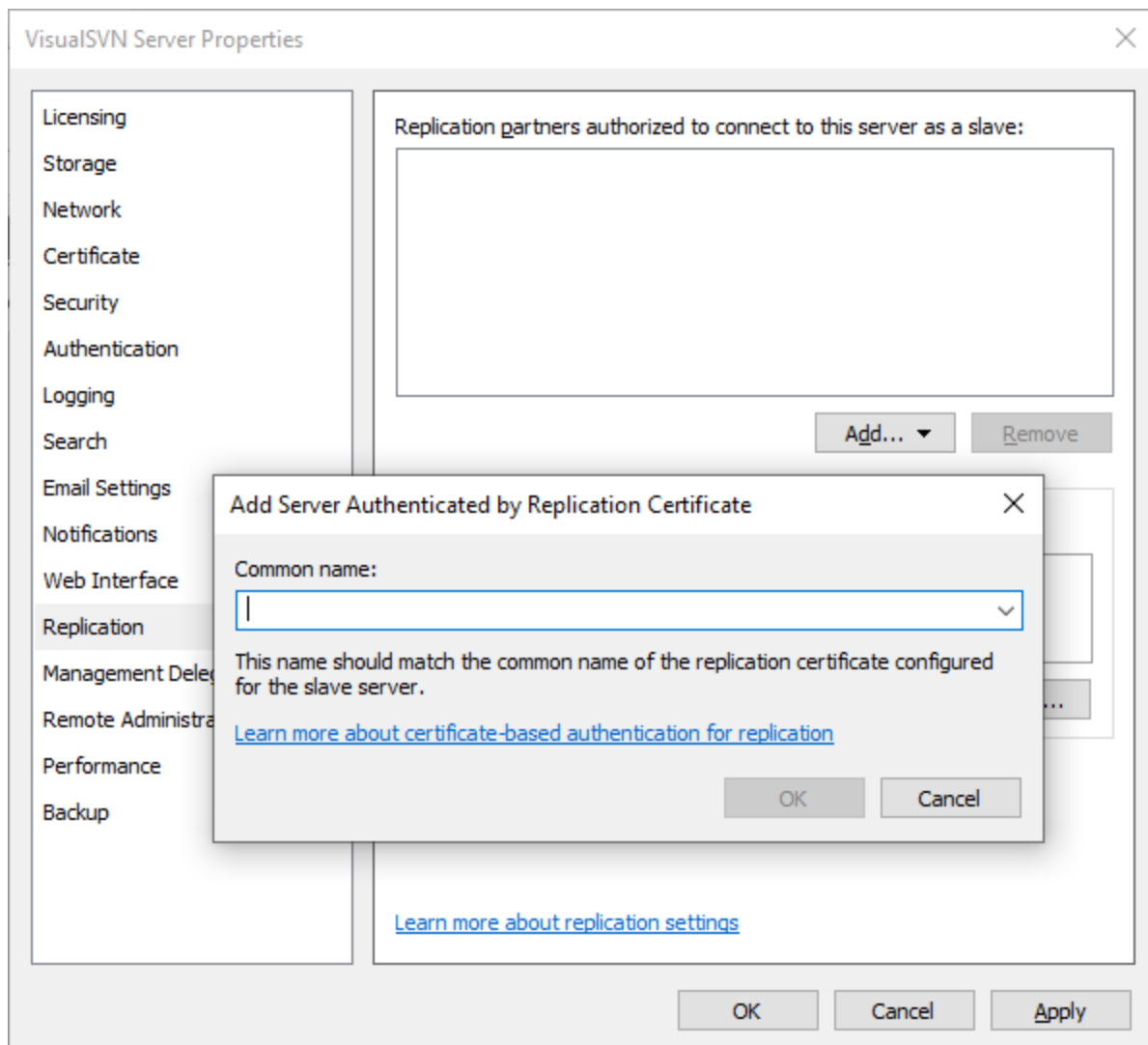
Cancel

- Copy the file to the “master”, then open Certificates snap-in (certlm.msc)
- Right-click on “Trusted People”, and under “All Tasks” click Import
- Click Next, then click the browse button and navigate to where the certificate from the “slave” machine was saved. Hit next, then hit finish.

Do the same, but copying the “master” certificate to the “slave”, to enable mutual authentication.

Authorise “slave” server to connect to “master”

From the replication tab in the “master” server’s properties window, click the Add menu-button and then click “Add server authenticated by Replication Certificate”. Enter the name given to the certificate from the previous step.



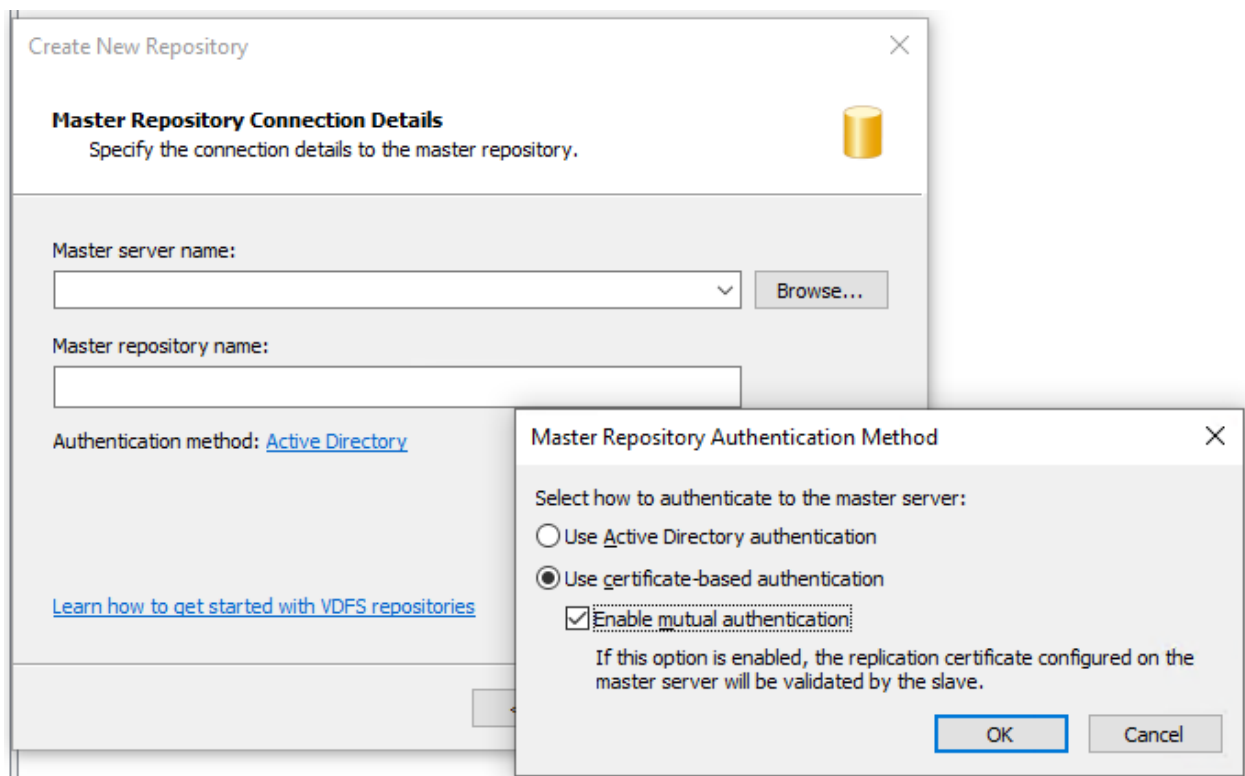
Create the “Master” and “Slave” repositories

Refer to 4.1.2 for the specific settings required for RTSW Repository.

On the “Master” server:

- Select Distributed VDFS repository, and then select Master
 - If there is an existing “Projects” repository, select “Import data from existing repository”
 - Alternatively, stop VisualSVN services, right-click on the repository, click All Tasks, and select Convert to VDFS Format.
- Remember to name the repository “Projects”.
- At Repository Replication Permissions, click Add and select “Add server authenticated by Replication Certificate”. Input the certificate name from the last two steps.

Repeat the above steps on the “slave” server, selecting Slave, inputting the “master” hostname, and of course name the repository “Projects”. Make sure the Authentication Method is set to Certificate-based authentication. Make sure mutual authentication is enabled for disaster recovery.

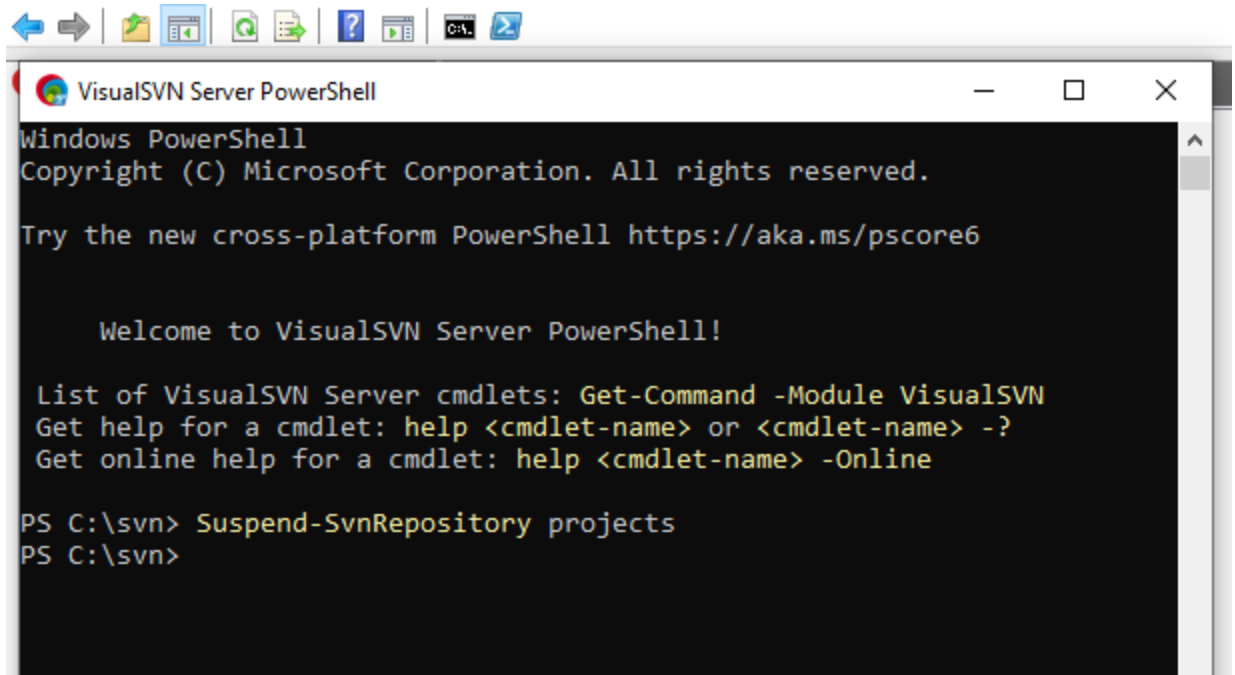


Failover & Disaster recovery

If the master server becomes unreachable, the “slave” server will become Read-only. To allow writing to this machine again, it needs to be upgraded to a Master.

Upgrade “slave” machine to Master

- Make sure the firewall settings are correct, see 4.4.2
- Ensure their certificates are trusted on the machine that is to be the temporary “master”, and make sure it is set as an authorised replication partner. See 4.4.4 & 4.4.5
- Suspend repository to prevent it from losing sync. This can be done from the VisualSVN Powershell window with the command *Suspend-SvnRepository projects*



```
VisualSVN Server PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

Welcome to VisualSVN Server PowerShell!

List of VisualSVN Server cmdlets: Get-Command -Module VisualSVN
Get help for a cmdlet: help <cmdlet-name> or <cmdlet-name> -?
Get online help for a cmdlet: help <cmdlet-name> -Online

PS C:\svn> Suspend-SvnRepository projects
PS C:\svn>
```

- On the temporary “master”, execute the following command: *Switch-SvnRepository -Name projects -Role Master*
- Replication needs to be re-enabled on the temp “master”, run *Set-SvnRepository projects -ReplicationEnabled \$true*, alternatively go into Projects repository properties and enable it from the Replication tab.
- Resume the server with the command *Resume-SvnRepository -Name Projects*

Replicate from Temporary Master to New Master

Once the original master has either been put back online or replaced, it needs the up-to-date data from the temporary master.

- Add new master to the temp's allowed replication partners (see section 4.4)
- Set up the new master as a "slave" machine
- Allow the new master to replicate the repository

Upgrading the New Master

Once the repository has been replicated onto the new server, it is ready to be designated as "Master". Follow the same steps as in 4.5.1, but with the New Master in place of the temporary.

- Demote the temporary "master" back to a "slave" with the command
Switch-SvnRepository projects -Role Slave -MasterServer [master server hostname]
 - Replace [master server hostname] with the "master" hostname.

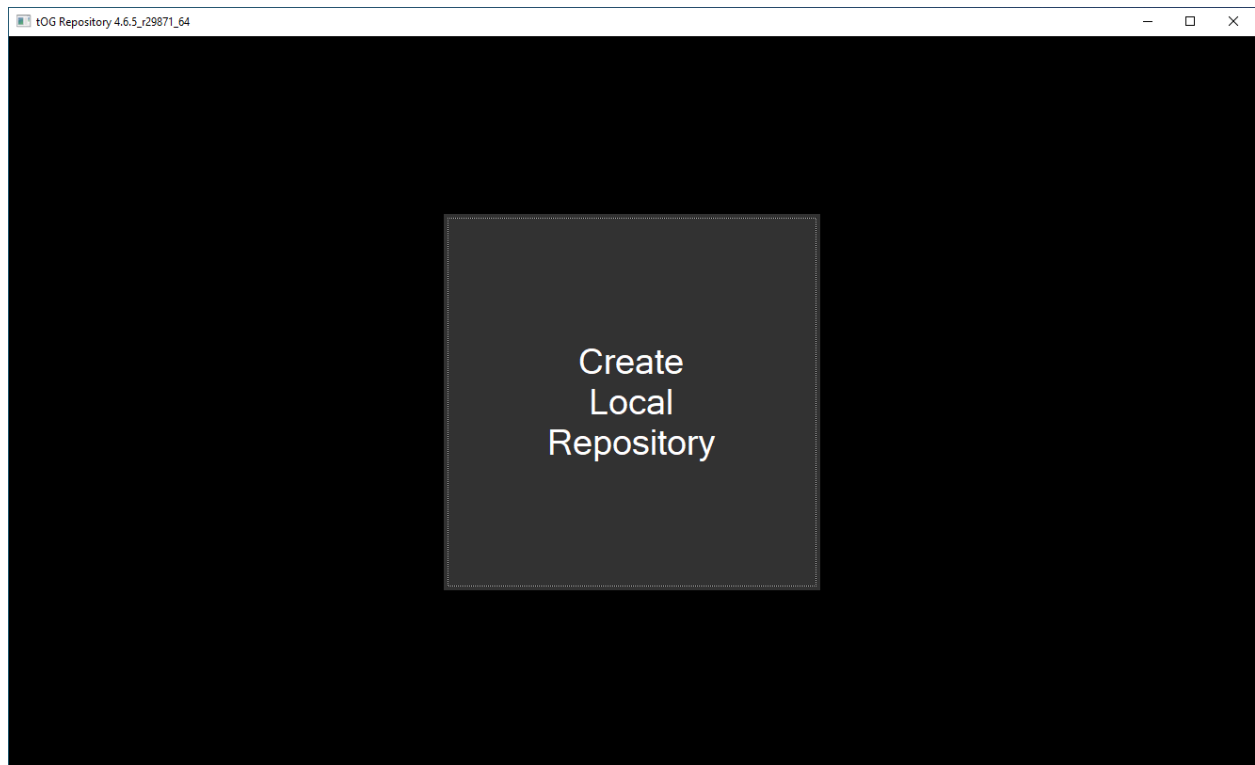
Using Repository

The Repository application can only be run via a Windows shortcut that is created by the installer and placed in the installation directory chosen in Chapter 3.3. It can be dragged to a more convenient location if desired.

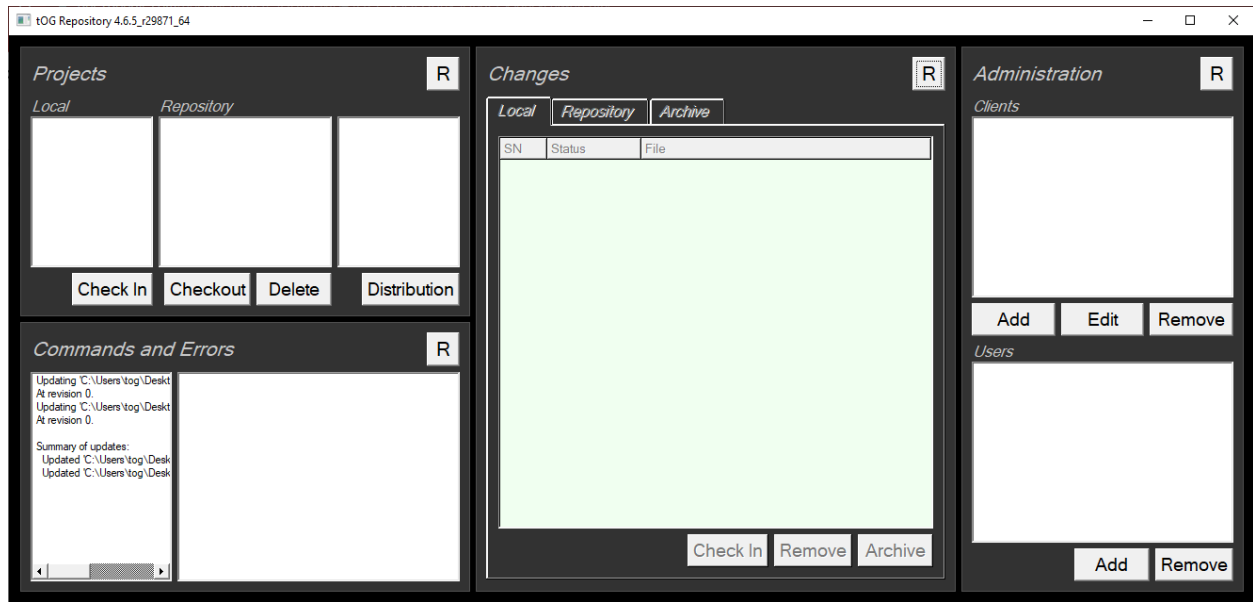
- To create a shortcut manually read section 6.3.
- Double-click on the Repository icon to start.

Creating a Local Repository

When the Repository is first run up, if the local repository does not exist the user is prompted to create the local repository. This will initialise an empty checkout of the server repository that was set during installation. This is only if SVN Projects Checkout was not selected in the installer. See below:



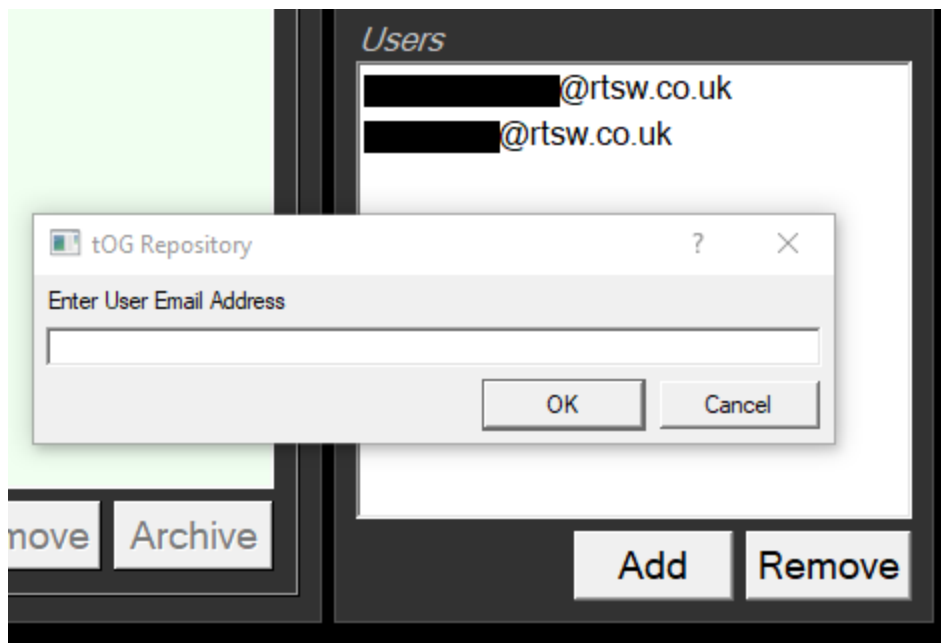
Click the “Create Local Repository” button to create this directory. This in effect retrieves the projects folder from the central repository and creates a folder corresponding to it on the local disk in the installation directory, e.g: .../RTSW/projects. It is initially empty.



Registering Users

Every Repository user should be added as a User. The list of users is also stored on the SVN Server so it is available to all instances of Repository.

- To add a user click on the “Add” button under the User list

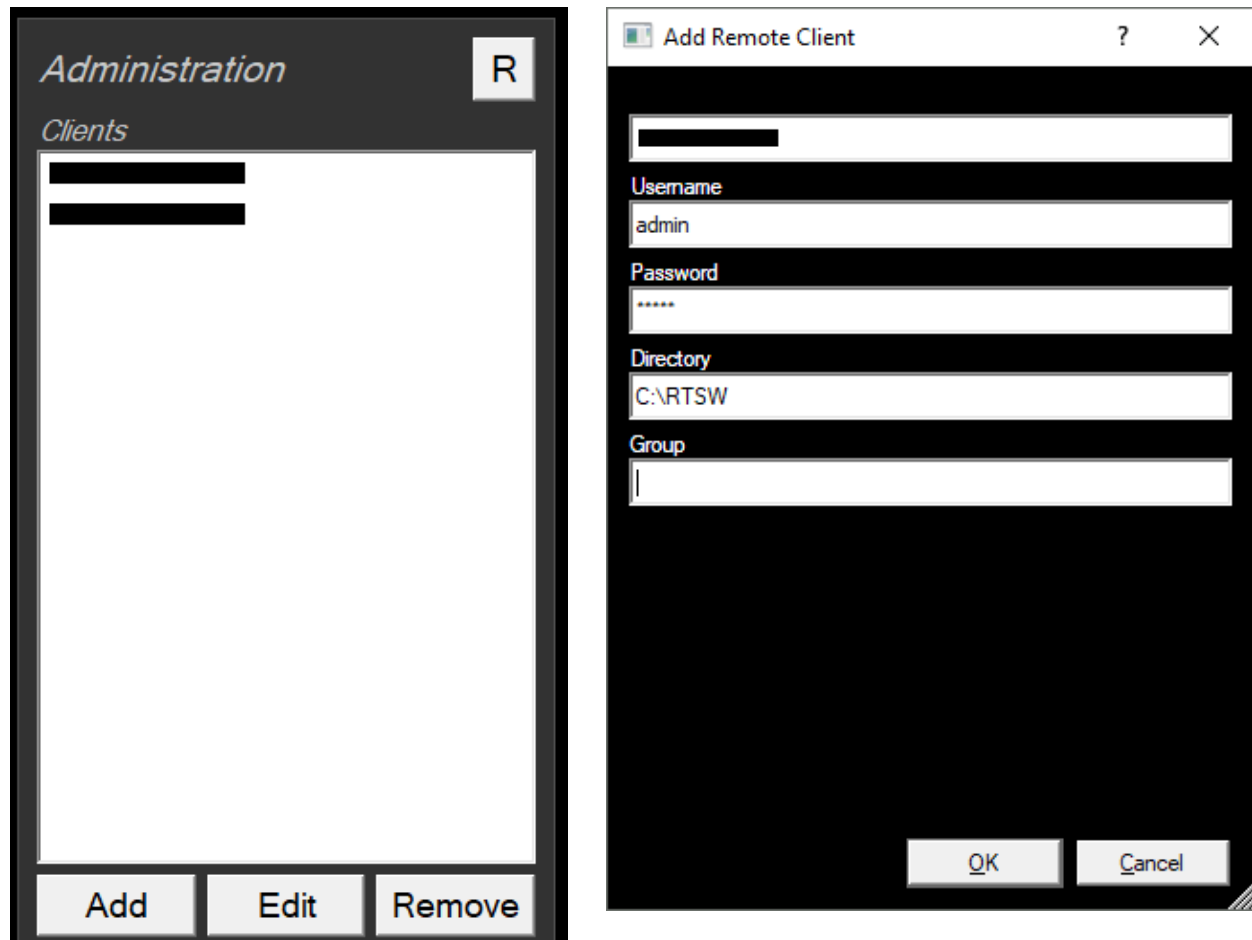


- Enter the users’ email address and click “OK”

Registering Clients

Each SVN Client (the TX machines) should be entered as a Remote Client. The list of remote clients is also stored on the SVN Server so it is available to all instances of Repository.

To add a remote client click on the “Add” button under the Remote Client list



Enter the remote client details and click “OK”:

- IP Address: the IP address of the remote client (must be unique)
- Username: unused
- Password: unused
- Directory: the Installation Directory that was specified when Repository was installed on this remote client (see Chapter 3.4)
- Group: A group name for this remote client - used to filter client names when distributing a project - defaults to “main” (must NOT contain a space)

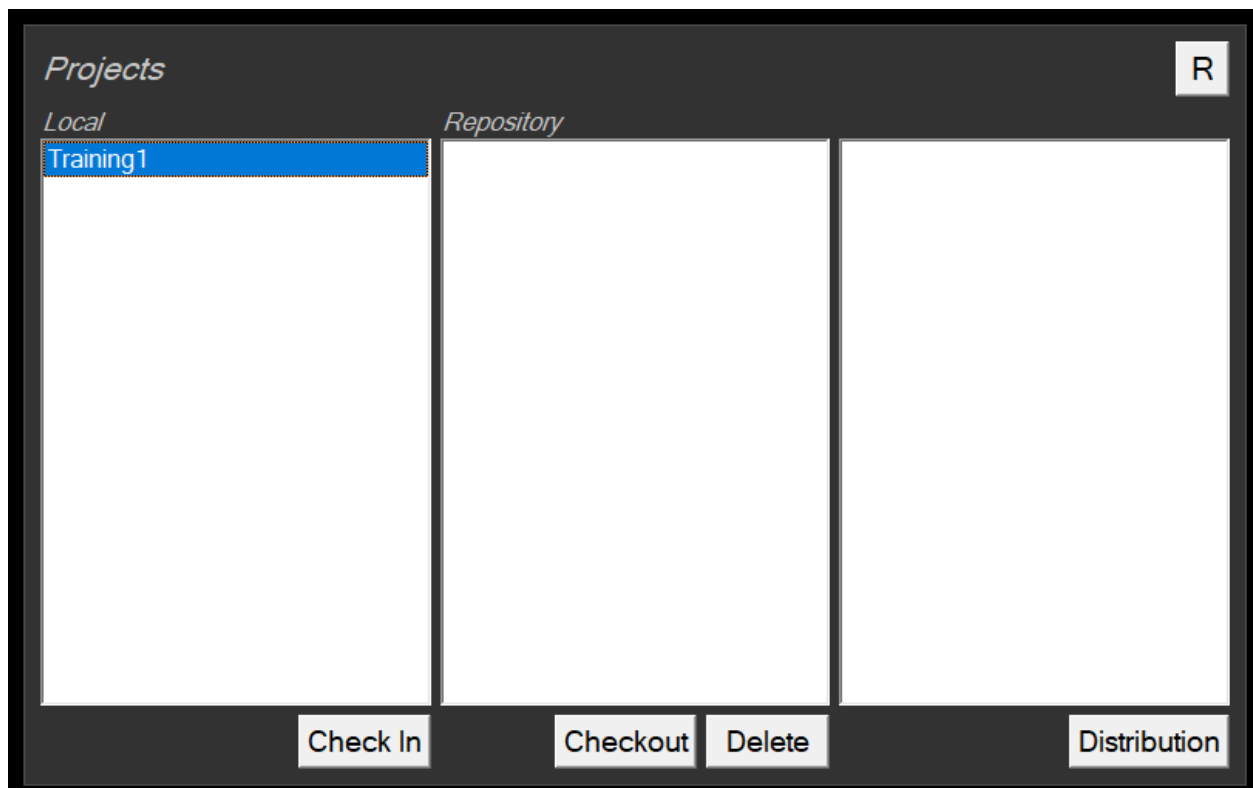
Checking-In Projects

Initial Check In

This is required when Repository has just been installed and the SVN Server contained no project, and if a new project has been created and it is not yet on the SVN Server

A Designer will create a project with graphics templates using RT editors. To Check-In the project:

- Copy the project folder to the local repository *projects* folder
- Click the Projects button
- Click the Local Tab
- Click "R" to refresh the list



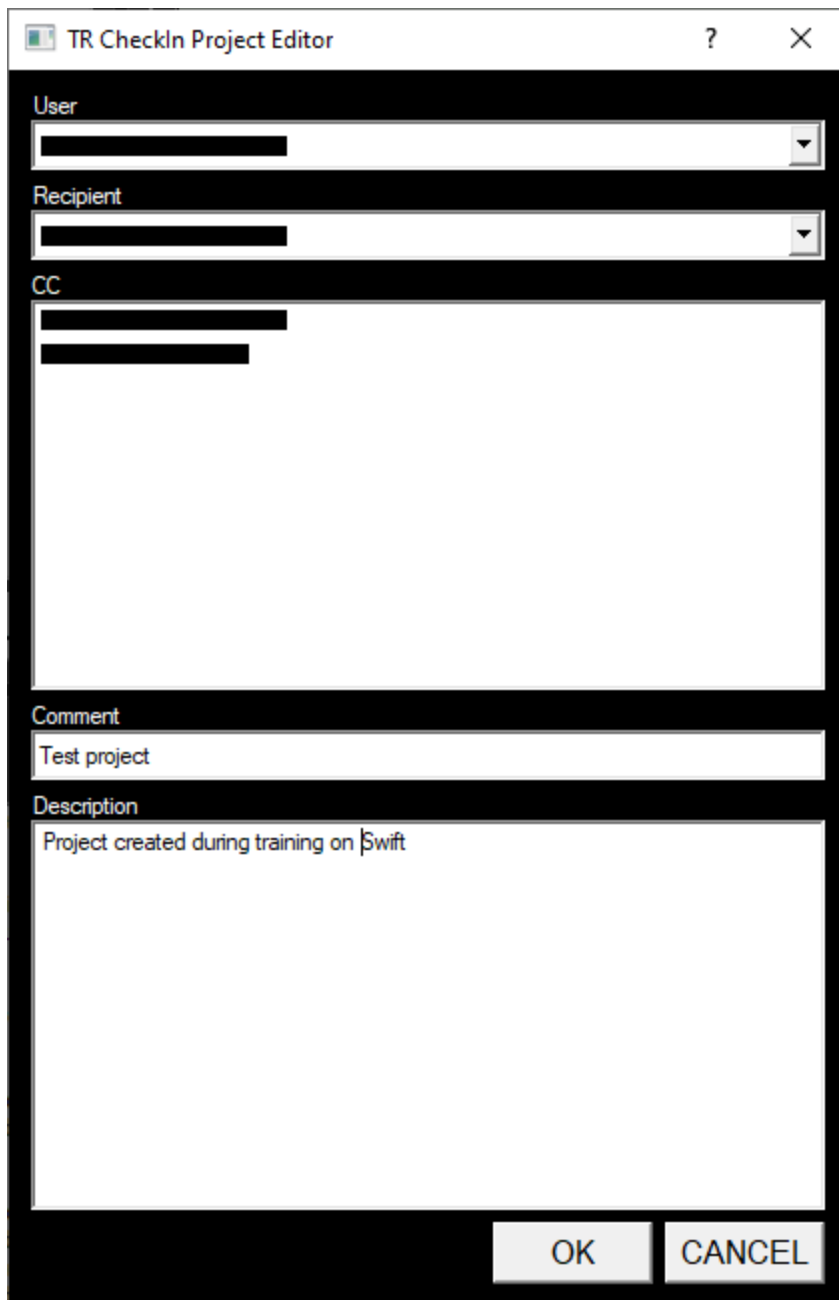
This list shows projects that are in the *local repository* folder but not yet in the *central repository*.

- Select it in the list
- Click on the Check In button

After accepting the check-in the following menu/form is displayed.

Fill in the fields and click on "OK"

- Recipient: the person to be emailed on completion of this check in (not yet implemented)
- CC: person(s) copied into the above email (not yet implemented)
- Comment: A brief comment about the project (mandatory)
- Description: A description of the project (optional)



The image shows a screenshot of a software dialog box titled "TR CheckIn Project Editor". The dialog box has a standard Windows-style title bar with a question mark icon and a close button (X). The main area of the dialog is divided into several sections, each with a label and a corresponding input field:

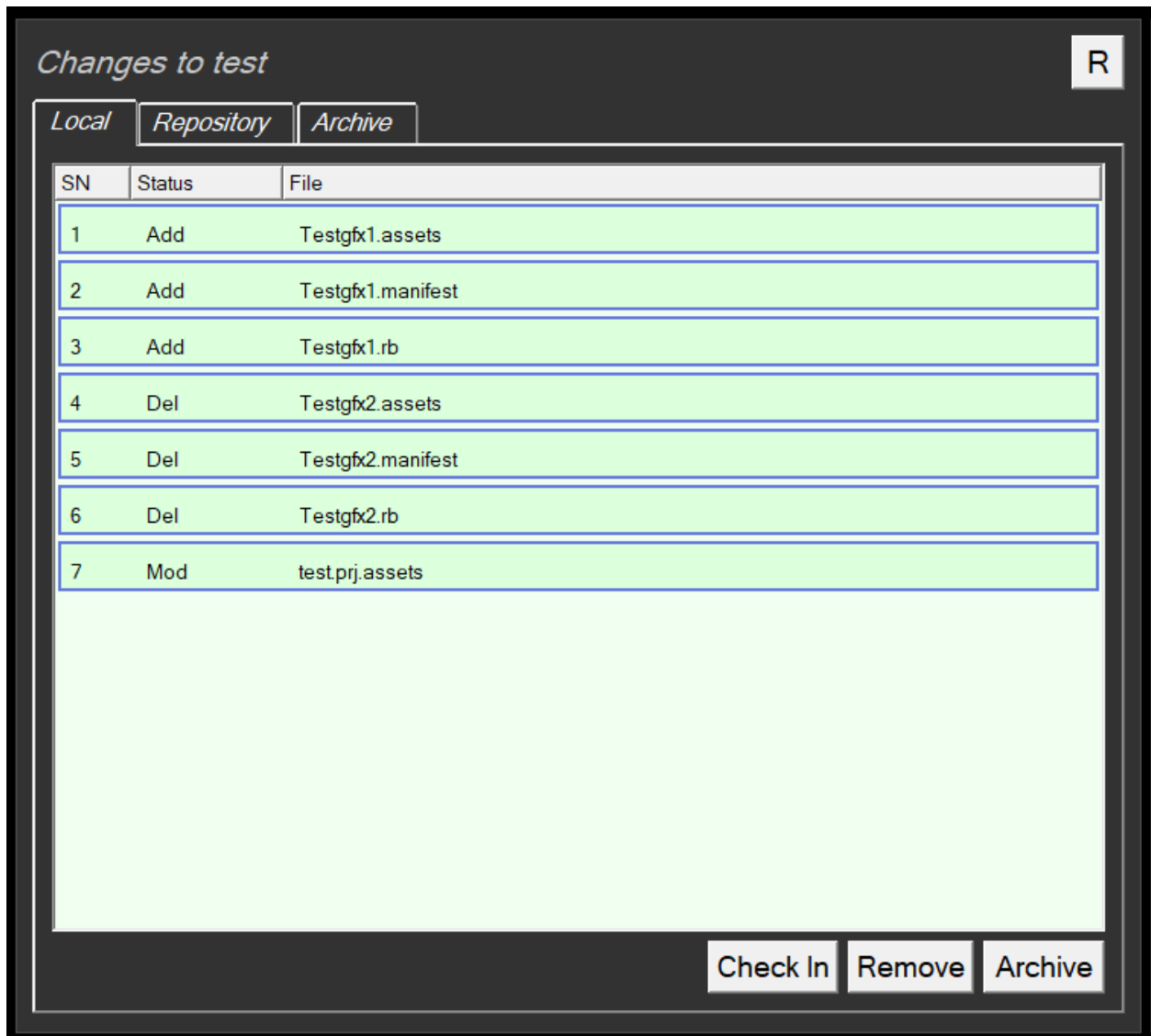
- User:** A dropdown menu with a black bar obscuring the selected value.
- Recipient:** A dropdown menu with a black bar obscuring the selected value.
- CC:** A text area containing two lines of text, both obscured by black bars.
- Comment:** A text field containing the text "Test project".
- Description:** A text area containing the text "Project created during training on Swift".

At the bottom right of the dialog box, there are two buttons: "OK" and "CANCEL".

Check In of Modified Projects

If a local project has been modified it should then be checked into the SVN Server

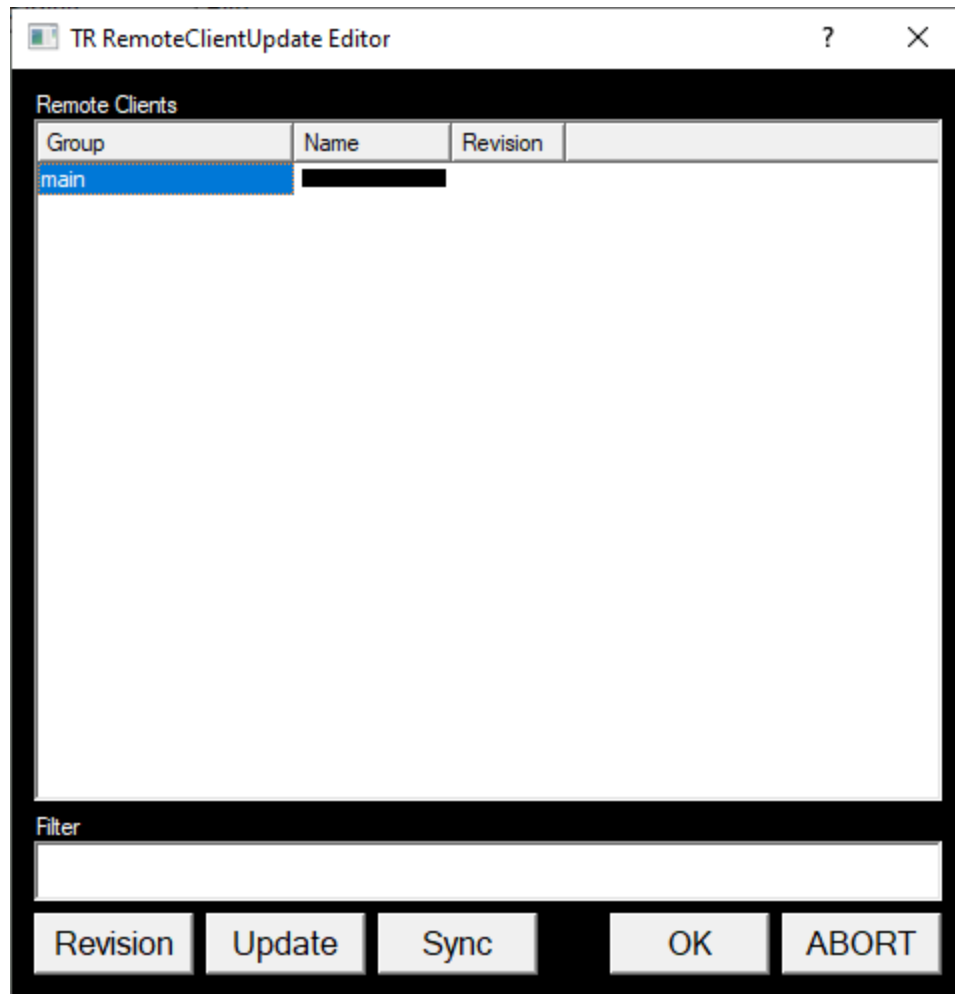
- Click the Changes button
- Click the Local Tab
- Click on the project in the project list



The screenshot shows a dialog box titled "Changes to test" with a tabbed interface. The "Local" tab is selected, displaying a table of changes. The table has three columns: SN, Status, and File. There are seven rows of changes, each highlighted in green. At the bottom right of the dialog are three buttons: "Check In", "Remove", and "Archive".

SN	Status	File
1	Add	Testgfx1.assets
2	Add	Testgfx1.manifest
3	Add	Testgfx1.rb
4	Del	Testgfx2.assets
5	Del	Testgfx2.manifest
6	Del	Testgfx2.rb
7	Mod	test.prj.assets

- The local changes are now shown in the green area of the menu.
- To Check-In these changes click on the "Check In" button.
- After accepting the check in the same form is presented as above in 5.3.1
- Fill in the form and click on OK.
- Following a successful check-in the following menu will be displayed:



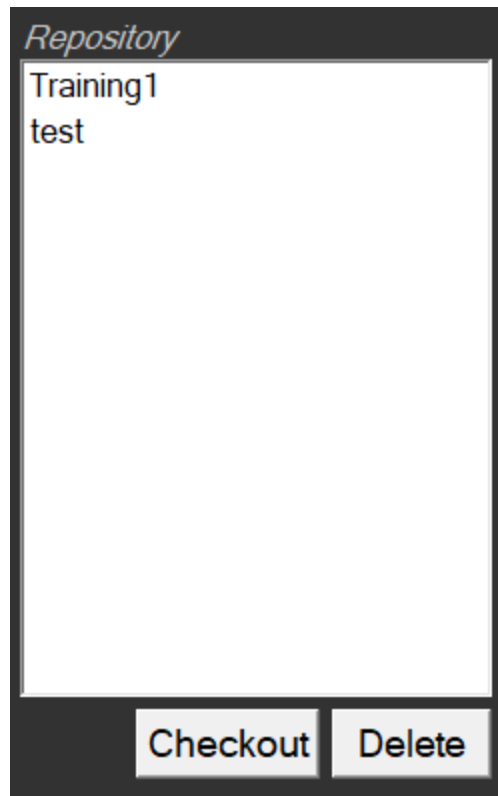
This menu gives the option of distributing the project to the TX machines that will be detailed in section 5.6.

Clicking on "OK" will leave this menu and NOT distribute the project.

Checking-Out Projects

Viewing projects in the central repository

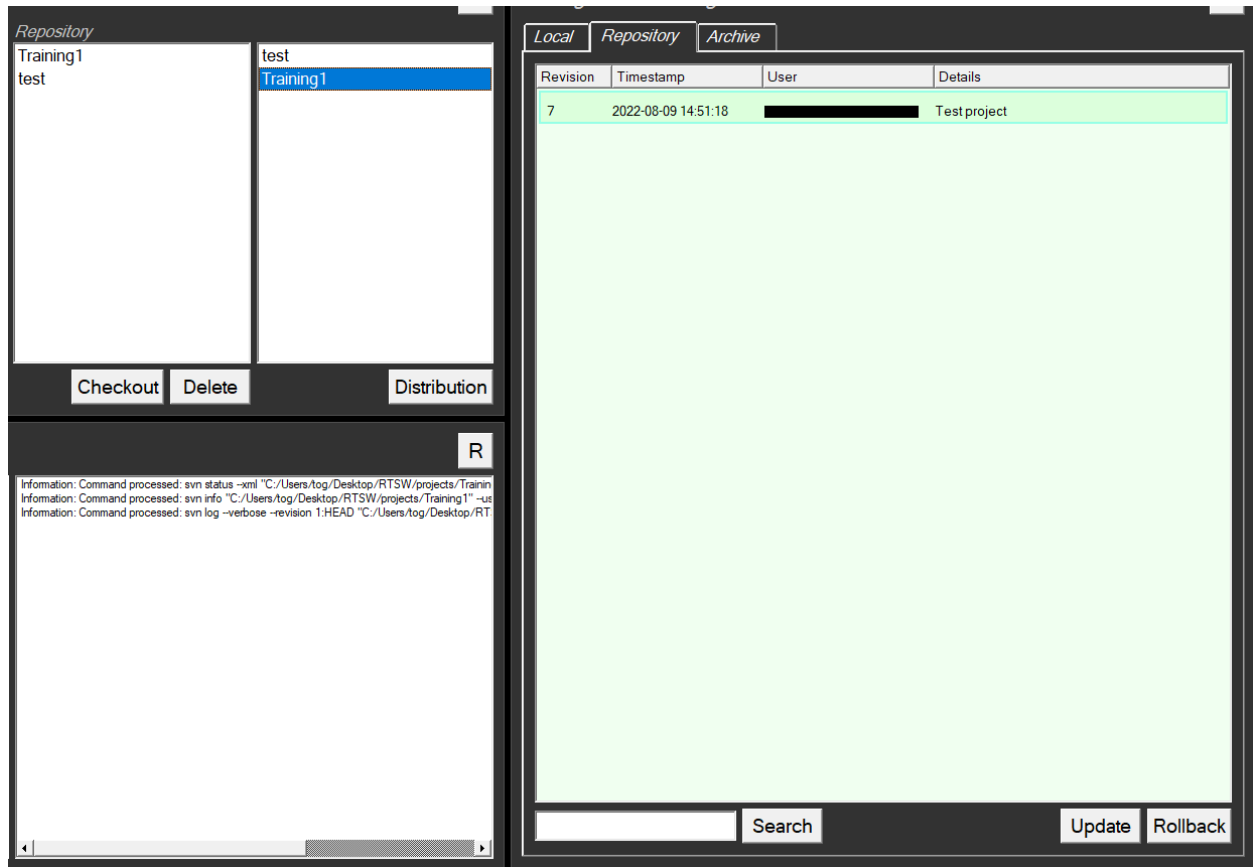
Users can see any projects in the central repository. See below:



Viewing projects already checked out

- Launch Repository
- Click on CHANGES

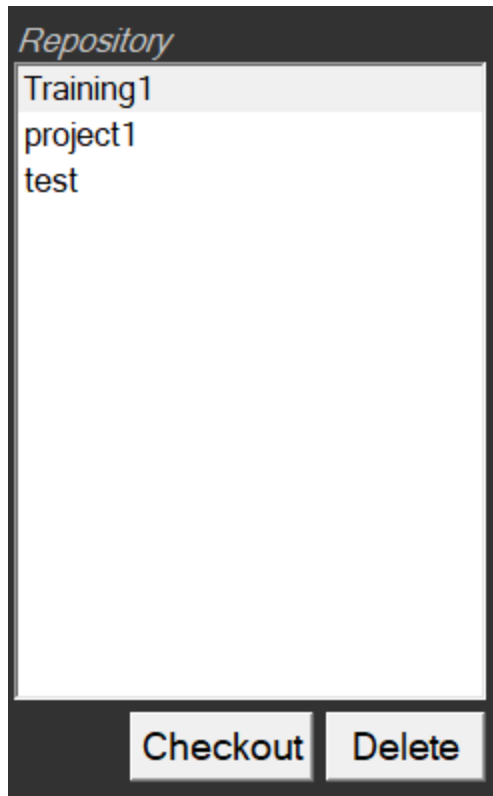
Projects already checked out are shown in a list:



Checkout a project not already checked out

If a project is shown as in section 5.5.1 but not shown as in section 5.5.2 then it needs to be checked out to the local repository.

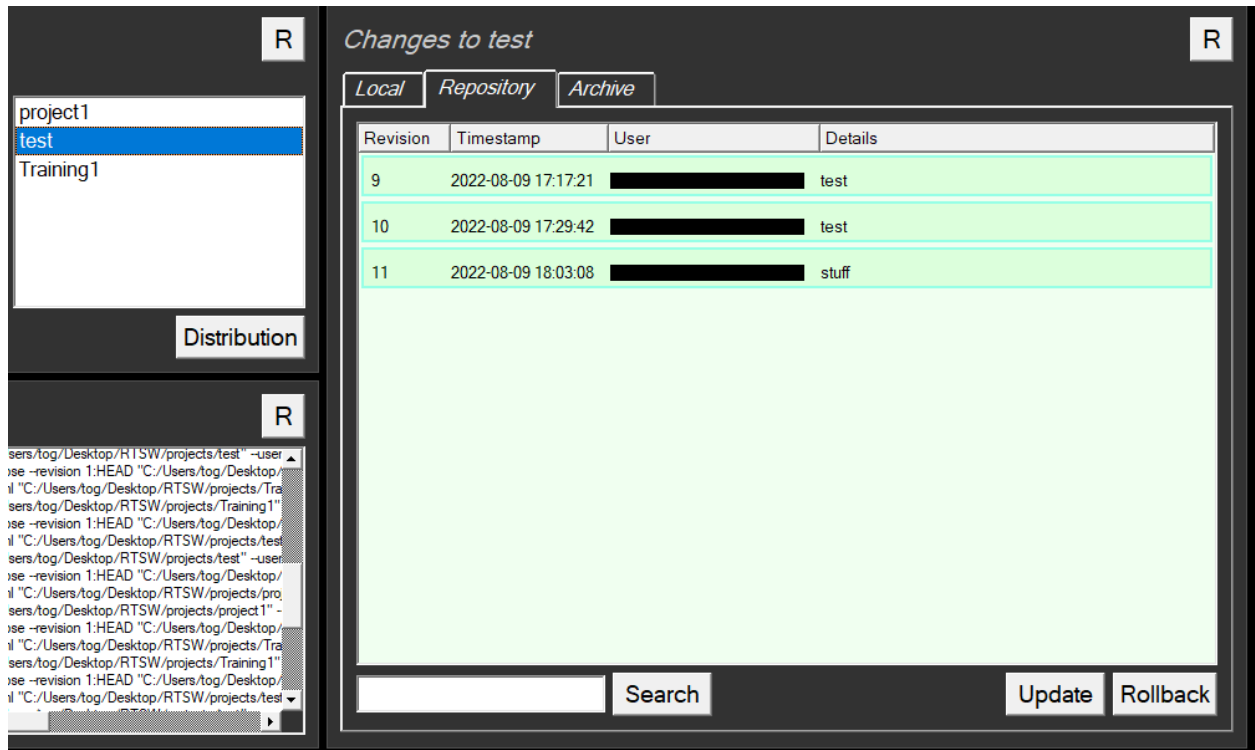
Select the project from the list and click on Checkout:



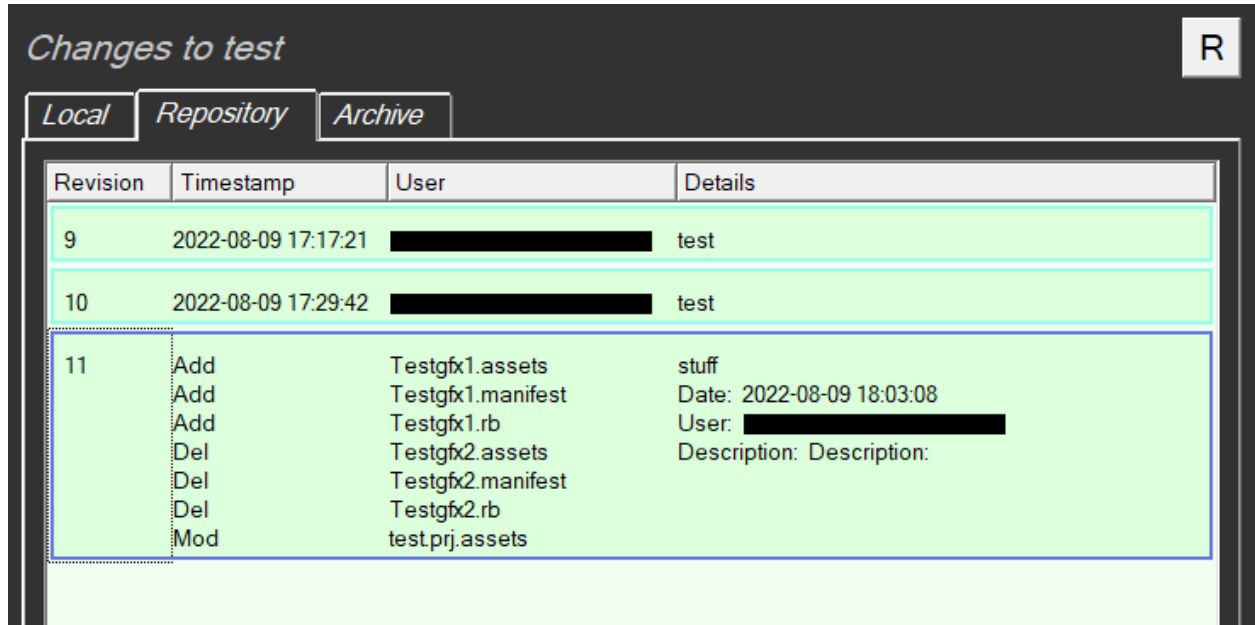
Update a project already checked out

To see the evolution of the project, and the changes made to it:

- Select the project
- Click the Repository tab.
- The project selected on the left will then display all changes on the right as a checked out project. Any changes that have occurred to the project, but are not held locally, will appear in cyan, see below:



The details of each change can be seen by clicking on a particular revision:



To update the local project to a particular revision from the repository select the desired revision and click on Update.

Pushing changes to remote clients

This is the process by which the remote clients - usually the TX machines - are updated to a given revision from the central repository. This operation is normally done by an Operator as it can have a disruptive effect on the TX machines.

The operation is done in 2 phases:

1. Update the remote machines - the new/modified files are downloaded onto the remote PCs
2. Sync the remote machines - the instance of tOG running on the remote machine is told that new files are available and it should re-read its assets files

Update the remote PCs

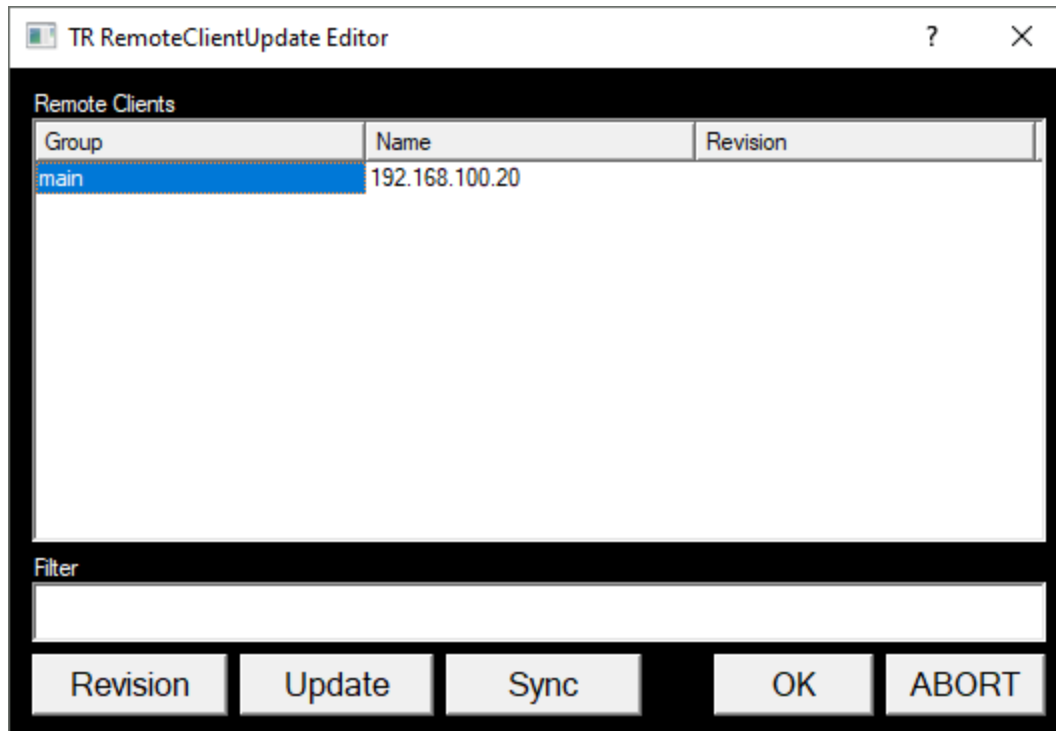
To be able to update a remote machine the Operator must have the project checked out to their local machine. See section 5.5 for details.

To push a change to 1 or more remote clients:

- Click the Repository tab
- Click on the relevant project from the list of the left
- Click on the revision to be pushed to the remote clients
- Click on the Update button

On the popup menu select the clients that the new changes will be uploaded to. In this example I only choose one machine to update.

- Putting an entry in the Filter box will filter the clients using their group name - the first field in the list of clients.
- Clicking on Revision will show the current revision in the form rXX::YY where XX is the current remote revision and YY is the current local revision.



Clicking on Update will push the selected project/revision to the selected clients.

The way this works is that every client is told to update itself from the central repository so strictly speaking the changes are not “pushed” to the client - the client is just instructed to update itself.

Synchronise the remote PCs

This is the same process as above but click on Sync instead of Update. The instance of Swift running on each remote client is notified that new assets are available and to re-read its assets.

This should be done at an appropriate time - i.e. when the render PCs are not using the graphic(s) to be updated.

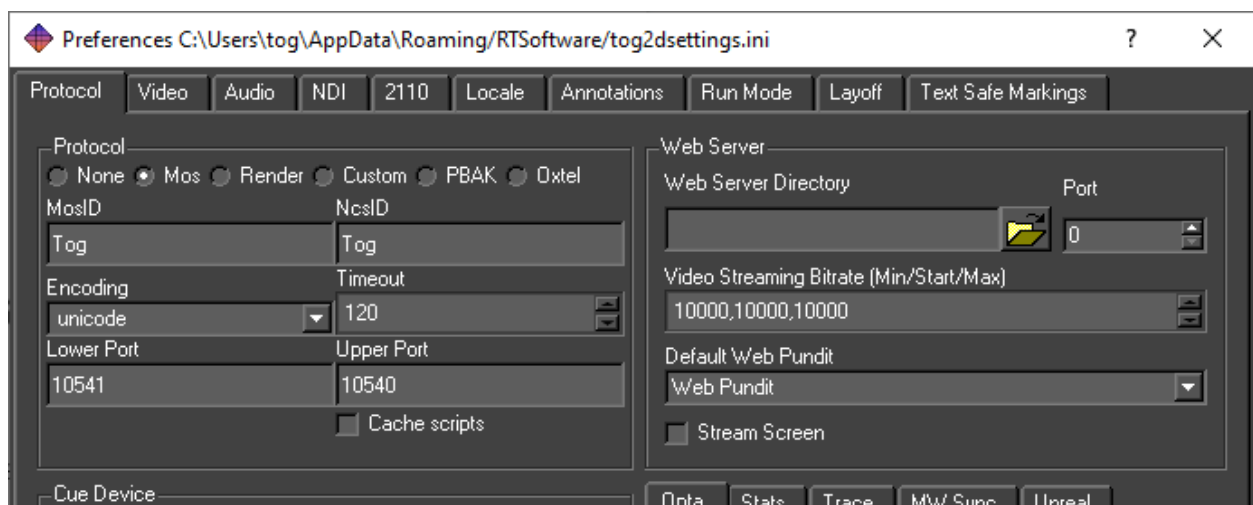
Configuring the Remote PCs to Synchronise

The synchronisation command is sent via MOS and instances of Swift running on the remote PCs must be configured to accept MOS commands.

Set the protocol to MOS. The parameters should be set to:

- MosID: Tog
- NcsID: Tog
- Encoding: unicode
- Timeout: 120.0
- Lower Port: 10540
- Upper Port: 10541

This is normally done from the Preferences Menu:



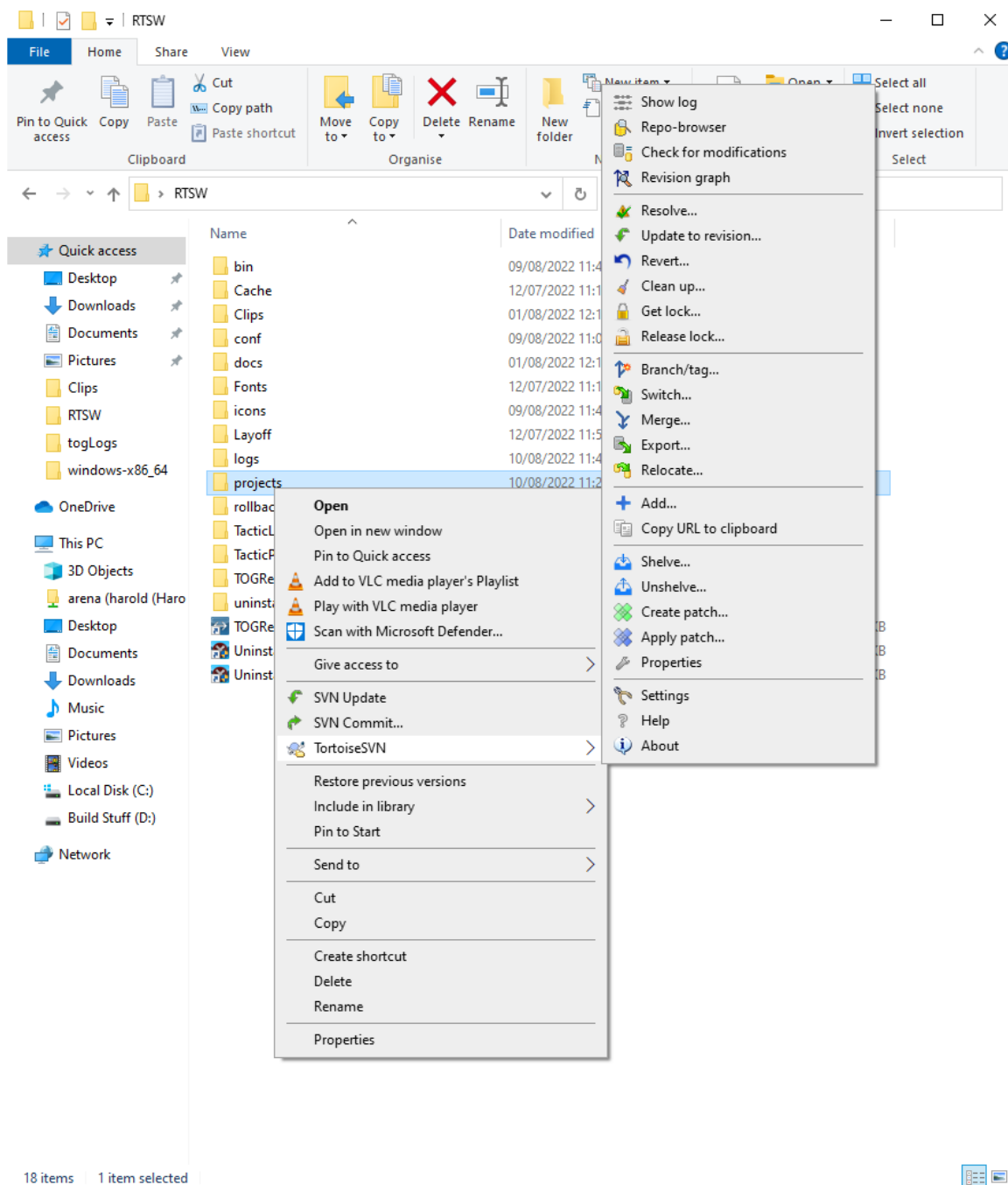
Using Subversion commands directly

This requires TortoiseSVN to have been installed and should only be used by experienced SVN users.

You may at this point use subversion commands directly.

- Right click on any folder
- A popup menu with the subversion tools will be listed.

These can be used to directly do all that Repository does and more besides. However, using Repository wraps them up in a safe and usable manner for the non-expert.



Conflicts

It is possible for files stored locally and files stored centrally to conflict. This will not happen in the simple model outlined above where one user writes to the repository and one (or more) reads from the repository. However, it can happen when more than one user writes to the repository from the *same version*.

- Conflicting changes will be coloured red in the Repository tab.

If a template author makes a change to a graphic without first checking out changes relevant to that graphic, subsequent attempts to commit changes may result in conflicts.

Avoiding conflicts

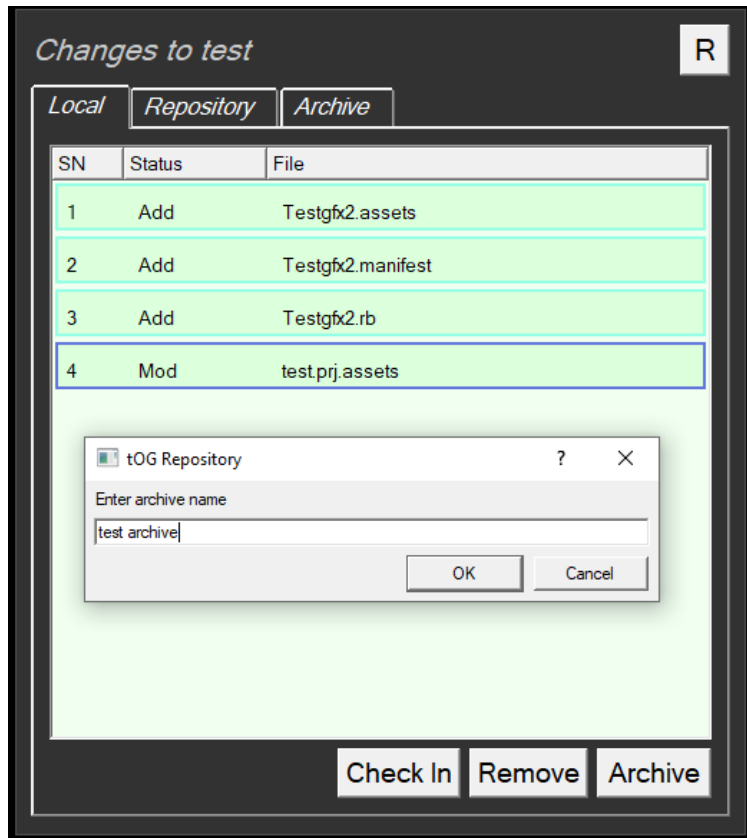
It is possible to resolve these changes by archiving and unarchiving local changes but it is best to avoid them completely:

- Coordinating changes to projects so no two authors are changing the same files at the same time.

Alternatively - there are tools available within SVN to interactively resolve conflicts.

Archiving Changes

Usually it is possible to directly access the machine hosting the central repository over the internet. For those times when it is not possible the user can **Archive** local changes to disk as a package.



Using Archive

- Click on the change required
- At the prompt, name your Archive

The resulting archive is saved into a folder on disk (which can be zipped) and emailed to a remote user.

The remote user can then copy the zip file to the archive folder and unzip it. This will then appear as an archive in the ARCHIVES tab.

Using UnArchive

To unarchive the file:

- Click the Archive tab
- Select the archive
- Click the Unarchive button

These changes will then be listed under the LOCAL changes tab - these will be changes to the actual project on disk. Archiving or removing changes is a very convenient way to clean up a project.

Searching for Changes

It is possible to search for changes made in projects that are stored in the central repository.

- Click the Repository tab
- Type the search term to locate the item(s)
- Click Search. See below:

The screenshot shows a software interface titled "Changes to Training1" with a red "R" icon in the top right corner. Below the title bar are three tabs: "Local", "Repository", and "Archive". The "Repository" tab is currently selected. Inside the tab area is a table with the following data:

Revision	Timestamp	User	Details
7	2022-08-09 14:51:18	[REDACTED]	Test project

Below the table is a large empty rectangular area. At the bottom of the window, there is a search bar containing the text "test", followed by three buttons: "Search", "Update", and "Rollback".

The text is matched against each change and if it forms part of any field in the change (user, comment, date etc.)

- If the textbox is empty, and Search is clicked, all changes are listed.

Update and Rollback Changes

If a user regrets a change, or if a user needs to work with a previous graphic template or style, they can update their local repository folder from a previous version.

Using Update to revert

- Click the Changes button
- Click the Repository tab
- Select the desired version to revert to by selecting the corresponding change
- Click the Update button.

This will change the local project so it matches that version - adding, deleting and modifying files as appropriate. This will result in all the changes which come after this revision changing colour to cyan. This is a temporary solution.

Note: *To completely remove changes from a repository is not possible.*

However, there is an option to **reverse** all the changes up to a previous revision and then commit these changes as a single change, Rollback.

Using Rollback to revert

- Click the Changes button
- Click the Repository tab
- Select the desired version to revert to by selecting the corresponding change
- Click the Rollback button

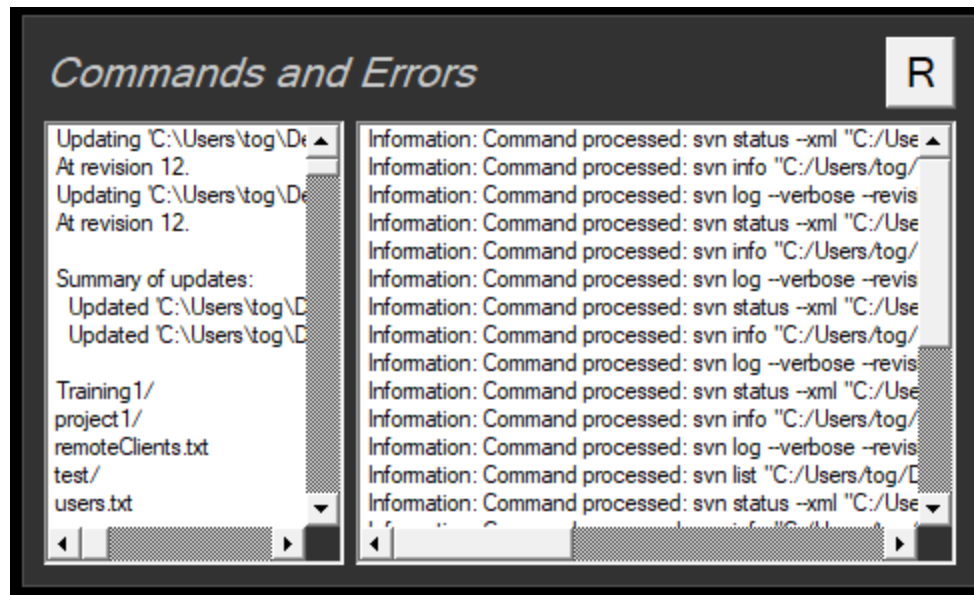
This action appears to **remove** these changes from the Repository and Update the Local version. The user will then be prompted to select remote clients to be updated.

Note: *This does not remove the changes from the repository but overrides them.*

These changes are not listed on the Repository tab Change List as they are hidden from the user.

Logs and Troubleshooting

The results of successful and unsuccessful commands are shown under Commands and Errors. The full log can be found in C:\users*username*\Desktop\RTSW\logs\



The two most common problems encountered are:

1. Failure to connect to the subversion server - This will present itself as a fast failure on all commands and an error dialog informing the user to check the error log. The subversion server is very stable and this kind of failure is either due to a network problem or to the server being in a Stop state. Ping the server machine - if the ping fails there is a network problem. If the ping succeeds, then it will be necessary to logon to the server administration web page to check its status.
2. Failure for a command to complete - This will present itself as a never-ending wait cursor. Some commands can take a while, especially checking in a project. But if the delay is unreasonable, it will almost certainly be due to permission failure for subversion or for update commands. The updates for remote servers are done using the plink.exe application. This requires that at least one attempt has been made on the command line with each of the remote clients. This establishes the remote client as an authenticated client.

DataClient

DataClient logs are found at:

C:/tOGLogs/tdc.log

Best to make sure that the directory exists as this is where the installer copies its log file too.

Installation log

The installer has a log that details its process. It is named accordingly:

bitrock_installer_10080.log

If the "C:/togLogs" folder exists it will be copied there. Otherwise it can be also found in:

C:\Users\<username>\AppData\Local\Temp

NOTE that in some Windows systems DataClient and Collabnet SVN Client cannot install clean in a sequence and an error message is returned. This will also be reflected in the installation log as:

"The service did not respond to the start or control request in a timely fashion."

In this case uninstall with the uninstaller and run the installer separately for these two features.

Configuration file

The Repository application can be configured with a configuration file in a simple text form. The file is placed in the root directory under /conf and named Repository.conf.

Some of the details are filled in by the installer but can be edited later.

Example:

localDirectory=C:\Users\<username>\Desktop\RTSW : installation directory

host=<http://192.168.100.52/svn/projects> : server repository address

commandProtocol=webService : client communication protocol. Only **webService** is supported

servicePort=5555 : TOGDataClient com port **5555**, cannot be changed and has to be available on the client machine.

emailServer= optional and used for email notifications. E.g: smtp.gmail.com

emailPort= optional and used for email notifications.e.g: 587

emailUsername= optional and used for email notifications.e.g: sean.kirwan@rtsw.co.uk

emailPassword= optional and used for email notifications. email password

username=admin : SVN Server user name. Default is *admin*. Has to be updated manually if a different user needed

password=admin : SVN Server password. Default is *admin*. Has to be updated manually if needed

pcUsername=tog : unused

pcPassword=tog : unused

emailAdminSender= : optional and used for email notifications. sender email address

emailAdminRecipient= : optional and used for email notifications. recipient email address

emailAdminCCList= : optional and used for email notifications. cc email address

Repository launch shortcut

The application can only be run via a Windows shortcut that is created by the installer and placed in the installation directory.

or can be created manually:

- 1) In <installdir>\bin\TOGRepository right click on TOGRepository_r24966.exe.
- 2) Select *Create shortcut*
- 3) Right click on the shortcut and select *Properties*
- 4) Edit the *Target* line:

```
<installdir>\bin\TOGRepository\TogRepository_r24966.exe --conf  
"<installdir>\conf\TOGRepository.conf"
```

Example:

```
C:\Users\tog\Desktop\RTSW\bin\TOGRepository\TogRepository_r24966.exe --conf  
"C:\Users\tog\Desktop\RTSW\conf\TOGRepository.conf"
```