HOWTO: Subversion for Windows with Apache server - a beginner's guide [version 0.4]

Index

1. Introduction

The Subversion for Windows HOWTO describes from a beginner's perspective, how to install the Subversion server on a Windows system, and get it running. This how to describes how to use the Apache 2.0 server as the network server component of the Subversion.

This guide is from beginner to beginners. The author of this HOWTO doesn't have any experience with any other Version Control system, so when we decided to start using such a system, everything was mostly new.

The guide will describe topics like the installation, basic configuration and setting up backup. It will provide working examples of configuration, ready for you to modify and use. Hopefully the HOWTO will grow, as the author gets more familiar with the system, more topics will get added, and providing the decision I have made are correct, not much will get changed.

Before I begin, I must mention the Subversion book which was really helpful, but still sometimes there just isn't time to read the book, and you're looking for a quick guide that

will help you get the system up and running ASAP. There is always time to read the book later [yeah, right].

Hopefully, this will turn out to be such a guide. It contains examples that you can download, modify and start using immediately. It should help you set up a Subversion server, including repository backup, in under two hours. It does not contain instructions or examples for the Subversion client. But, if you're using Eclipse, the Subclipse client will not take more than additional five minutes of your time.

2. Installation

This chapter describes the process of installation, beginning with the OS installation, continuing with Subversion server installation and ending with Apache 2.0 server installation. I have installed the Subversion and Apache 2.0 server as the local Administrator. All later configuration was done under a different user name, but still as a member of the local Administrators group.

2.1. Setting up the OS

Since this isn't a Windows HOWTO, this chapter will be rather short, and answer a few simple questions that I had before I began. The two OS related questions that I was asking myself before starting with the OS installation were:

- Does the Subversion server work with Windows XP SP2 and all the patches?
- Does the server work on DHCP enabled interface? [some apps don't, you know]

And the answers are obviously YES, and YES. So, to put it simply, install the Windows XP and all the latest service packs and patches - don't forget to turn off the firewall [or in this security crazed world, don't forget to open the port the server is using in it]. It depends on your network situation, really.

For IP address I have selected automatic interface configuration [DHCP], the host was highly originally named subversion, and the system is part of a Windows domain. For a server, this is a strange choice, but it was good enough for the testing phase of the project. With the testing phase finished, I have assigned a static address to the interface - I suggest you do the same from the beginning.

2.2. Installing Subversion

The Subversion Windows installation package can be downloaded from the Subversion server, at the following URL:

http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=91

I have downloaded the distribution packaged in a standard Windows installation program. For the Subversion version 1.1.3, grab the file named svn-1.1.3-setup.exe .

The installation itself is completely straight-forward. You read [and accept] a license agreement, select the destination directory, press Next a couple of times, wait for the file copying to stop and press finish. What you get in the end, is a link to the documentation on your desktop [the icon image is broken], and the Subversion is installed in the directory you have specified.

The default directory is C:\Program Files\Subversion, and here is a list of directories the installation creates:

C:\Program Files\Subversion\bin	Contains all the binaries like svn.exe, svnadmin.exe and svnlook.exe .
C:\Program Files\Subversion\doc	Contains the Subversion book in the Windows help file format svn-doc.chm
C:\Program Files\Subversion\helpers	
C:\Program Files\Subversion\httpd	Contains the Apache 2.0 plug-in modules mod_authz_svn.so and mod_dav_svn.so .
C:\Program Files\Subversion\iconv	
C:\Program Files\Subversion\share	

The C:\Program Files\Subversion\bin is added to the path.

And that's about all there is to the installation.

2.3. Installing Apache 2.0

The Apache 2.0 server for Windows installation package can be downloaded from the Apache.org server, at the following URL:

http://httpd.apache.org/download.cgi

I have downloaded the distribution packaged in a standard Windows installation program - the Win32 Binary (MSI Installer) .

The installation itself is completely straight-forward. You read [and accept] a license agreement, enter your domain name, server name, the administrator's e-mail address, and the port the server will be listening on. The default values are already selected for you. For dedicated a subversion server, I suggest you leave it running on port 80. You can always change it later.

Then you can select typical, or custom install. Selecting typical install lets you choose the destination directory, and after that it starts copying files. When it finishes, you're done.

At this point it is probably recommendable to restart the system, before moving on to configuration.

3. Configuration

This chapter describes how to configure the Subversion system, and how configure the Apache server to make it available over the network.

3.1. Configuring Subversion

The Subversion stores the content into so called repositories. You need at least one repository to store all your data into, or may have multiple repositories, one for each project. This HOWTO will assume multiple repositories are used. We will call these projects project1 and project2.

So, let's create a directory for all our projects, and then a subdirectory for each of the projects, e.g.:

```
C:\Repositories\project1
C:\Repositories\project2
```

These are just directories to hold our repositories, now we must create the repositories themselves, using the svnadmin utility:

```
svnadmin create C:\Repositories\project1
svnadmin create C:\Repositories\project2
```

As the Subversion book warns, make sure to create all repositories on your local disks [FiberChannel is treated as a local disk]. Failing to do so, may result in repository corruption.

Basically this is pretty much it, when it comes to creating the repositories. As the Subversion book tells us, each repository is stored in a Berkeley DB database, which can be configured in many different ways, but the default configuration works, and for a beginner, I found no reason to change anything.

To make them available to your development teams, the Apache server needs to be configured.

3.2. Configuring Apache 2.0 server

As the Apache server will only be a front end for the Subversion system, I suggest that all the Subversion specific files are stored in a separate directory, which is at hand, and not hidden away in some Apache directory. In the best spirit of *nix systems, let us name that directory etc.

C:\etc

But, before we start with the Subversion specific configuration, let us make the necessary steps and configuration changes, to link the Apache server with Subversion.

Please note, that the Apache server requires, that you write all the directories using forward slash as the separator, e.g. C:/Program Files/Apache Group/Apache2 .

Step 1
Copy the files mod_authz_svn.so and mod_dav_svn.so from C:\Program Files\Subversion\httpd into C:\Program Files\Apache Group\Apache2\modules .

Step 2
Modify the C:\Program Files\Apache Group\Apache2\conf\httpd.conf file:

Add the modules to the Apache server

```
    LoadModule  dav_module             modules/mod_dav.so
    LoadModule  dav_svn_module         modules/mod_dav_svn.so
    LoadModule  authz_svn_module       modules/mod_authz_svn.so
```

Add the Access lines to the <Directory> sections, to protect your system.

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Order Allow,Deny
    Allow from 10.0.1
</Directory>

<Directory "C:/Program Files/Apache Group/Apache2/htdocs">

#
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI
MultiViews
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important.  Please see
# http://httpd.apache.org/docs-2.0/mod/core.html#options
# for more information.
#
    Options Indexes FollowSymLinks
```

```
#
# AllowOverride controls what directives may be placed in .htaccess
files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
    AllowOverride None

#
# Controls who can get stuff from this server.
#
    Order allow,deny
    Allow from 10.0.1

</Directory>
```

This allows access from all computers in the address range 10.0.1.1 - 10.0.1.254 .

At the end of the file, include a Subversion configuration file. We will create this file in one of the next steps.

```
    Include c:/etc/subversion.conf
```

We place the subversion.conf file in the before mentioned etc directory.

Step 3
Our decision was that anonymous access to repositories will not be allowed. Also, we would like that only the developers working on a specific project, can modify the contents of that project's repository. All other developer will have read only permissions to all projects. After all, we want our developers to share their code, don't we.

So first, we create a password file for authentication. All the developers that will use our Subversion server must chose a user name and a password. Unfortunately, the simplest way to do this is locally, so you must ask all the developers to come to the system and enter chose a password.

In the best tradition of Subversion book, let us name our developers Harry and Sally. Since we have two projects, we'll have a somewhat bigger development department, adding Ross and Rachel to our list of employees.

```
C:\etc>C:\Program Files\Apache Group\Apache2\bin\htpasswd -cm
C:\etc\svn-auth-file harry
New password: *****
Re-type new password: *****
Adding password for user harry

C:\etc>C:\Program Files\Apache Group\Apache2\bin\htpasswd -m
C:\etc\svn-auth-file sally
New password: *******
Re-type new password: *******
```

```
Adding password for user sally

C:\etc>C:\Program Files\Apache Group\Apache2\bin\htpasswd -m
C:\etc\svn-auth-file ross
New password: *****
Re-type new password: *****
Adding password for user ross

C:\etc>C:\Program Files\Apache Group\Apache2\bin\htpasswd -m
C:\etc\svn-auth-file rachel
New password: *****
Re-type new password: *****
Adding password for user rachel
```

When using the command for the first time, add the -c option. This creates the file named C:\etc\svn-auth-file . The -m option instructs the htpasswd utility to use MD5 algorithm to encrypt the passwords.

Step 4
Now that we can authenticate our users, we must configure the access rights to our repositories. To do this, we create another file in our etc directory.

C:\etc\svn-acl

```
#
# specify groups here
#
[groups]
team1 = ross, rachel

#
# team1 group has a read/write access to project1 repository
# all subdirectories
# all others have read access only
#
[project1:/]
@team1 = rw
* = r

#
# project2 repository, only harry and sally have read-write access to
project2
#
[project2:/]
harry = rw
sally = rw
* = r

#
# ross is helping with the time zone part of the project2
#
[project2:/timezone]
harry = rw
sally = rw
```

```
ross = rw
* = r
```

The groups section can be used to define groups of users. For repository project1, only users from the group team1 have read/write access. All other users have read only access.

It is possible to define access for the entire repository, or for specific directory within repository.

Step 5
In the end it is time to link the Apache server with the Subversion. This is done using the C:\etc\subversion.conf file:

```
<Location /project1>
  DAV svn
  SVNPath C:/Repositories/project1

  AuthType Basic
  AuthName "Subversion Project1 repository"
  AuthUserFile c:/etc/svn-auth-file

  Require valid-user

  AuthzSVNAccessFile c:/etc/svn-acl
</Location>

<Location /project2>
  DAV svn
  SVNPath C:/Repositories/project2

  AuthType Basic
  AuthName "Subversion Project2 repository"
  AuthUserFile c:/etc/svn-auth-file

  Require valid-user

  AuthzSVNAccessFile c:/etc/svn-acl
</Location>
```

The developers can access the C:\Repositories\project1 repository at the http://subversion/project1 URL. The access is only available to a valid user, and a basic HTTP authentication is used. The Apache server can read the valid user names and passwords from the C:\etc\svn-auth-file file. The c:\etc\svn-acl file defines the access rights to the repository.

Don't forget to restart the Apache server for the configuration changes to take effect.

Conclusion
So, we have created the Subversion repositories, and configured an Apache server to get them over the network. User names and passwords have to be used to access the

repositories, and different levels of access are given to different users. So, we now have a working Version control system - it is time move on to the next chapter.

4. Backup

Having a central Version control system without a backup may very soon prove to be an exercise in futility. But, what to backup?

After reading the Subversion book I have decided, that since I don't know anything about the Berkeley DB administration, and that if our server crashes, we might move to a new version of Subversion, it seems wisest to backup all the data in the Subversion dump file format. This will [hopefully] allow any future version of Subversion to digest the data, and in the event of system recovery, we will not have to battle with a set of unknown database tools.

Hopefully, getting to know the system will not make me regret this decision. So, all we are left to do is create a system utility that will run every once in a while, and dump all the changes made to a repository in a new file. Moving that data to some sort of a permanent storage [tape or CD-ROM] is not a subject of this HOWTO.

4.1. What to back up?

Your repositories, obviously, but how? Well, the dumps will get written into directory named c:\backup\dumps, so this is the directory to put on tape. Besides that, I also back up the c:\etc directory. I keep the latest version of the Apache httpd.conf file in it. This makes a total of 2 directories. Everything else can be downloaded from the Internet, if the worst happens.

You may turn on the compression on the backup folder, to preserve space.

4.2. Creating a repository dump

Since writing programs is what I do, I have decided to use VB Script for the backup procedure. The script basically has a subroutine [CreateDump] that gets the last known revision number for a given repository, compares it to the current revision, and if necessary, dumps the most recent changes into a file.

If you're wondering why not Perl, I wanted to learn something new, and didn't want to add yet another bloatware to the system. VB Script is part of the OS installation.

To break this down further, the subroutine has five parameters:

1. A name of the log file
2. A name of the file containing the last know revision
3. A command string for getting the youngest revision of the defined repository

4. The repository that is to be dumped
5. A fragment of the dump file name

In the example, for the Project1 the subroutine opens a log file, and compares the last know revision number [stored in the file c:\etc\proj1-last] to the youngest revision number for that repository. For example, if the last known revision number for repository C:\Repositories\project1 is 4712 and the youngest revision is 4738, the subroutine executes the following command:

C:\Progra~1\Subversion\bin\svnadmin.exe dump C:\Repositories\project1 --revision 4712:4738 --incremental

The dump is saved into a file c:\backup\dumps\proj1-4712-4738.dmp .

If the file c:\etc\proj1-last does not exist, the last known revision number is assumed to be 0 and the option --incremental is omitted from the dump command.

The example below creates backup files for two repositories. It uses two separate log files, but the script can easily be modified to only use one. But, it must use different lastFileName for each repository. It can be scheduled to run at your convenience. I run it once a day.

The script can be run with the following command:

C:\windows\system32\cscript.exe c:\etc\backup.vbs

The file backup.vbs is available for download.

```
Const ForReading = 1
Const ForWriting = 2
Const ForAppending = 8

Const folderName = "C:\backup\dumps\"
Const repositoryProj1 = "C:\Repositories\project1"
Const repositoryProj2 = "C:\Repositories\project2"

getYoungestProj1 = "C:\Progra~1\Subversion\bin\svnlook.exe youngest " +
repositoryProj1
getYoungestProj2 = "C:\Progra~1\Subversion\bin\svnlook.exe youngest " +
repositoryProj2

Set objFSO = CreateObject( "Scripting.FileSystemObject" )
Set WshShell = CreateObject( "WScript.Shell" )

Call CreateDump( "C:\backup\proj1.log", "C:\etc\proj1-last",
getYoungestProj1, repositoryProj1, "proj1" )
Call CreateDump( "C:\backup\proj2.log", "C:\etc\proj2-last",
getYoungestProj2, repositoryProj2, "proj2" )

WScript.Quit( 0 )
```

```
'**********************************************************************
**********
'*
'* End of script body
'*
'**********************************************************************
**********

Sub CreateDump( logFileName, lastFileName, getYoungestCmd, repository,
dumpName )

  ' Open the log file
  Set objLogFile = objFSO.OpenTextFile( logFileName, ForAppending,
True )
  objLogFile.WriteLine Now & " - - Script started - -"

  ' Default last revision is 0
  lastRev = 0

  ' Does the file exist?
  If ( objFSO.FileExists( lastFileName ) ) Then
    Set objFile = objFSO.GetFile( lastFileName )
    ' Does it contain anything?
    If ( objFile.Size > 0 ) Then
      Set objTextFile = objFSO.OpenTextFile( lastFileName, ForReading )
      ' Get the last revison and increase it by 1
      lastRev = objTextFile.Readline
      lastRev = lastRev + 1
    End If
  End If

  ' Execute the getYoungestCmd and read its output
  Set objExec = WshShell.Exec( getYoungestCmd )

  Do While ( objExec.Status <> 1 )
       WScript.Sleep 100
  Loop

  youngest = objExec.StdOut.Readline

  ' Is the youngest revision above the last one?
  If ( CLng( lastRev ) > CLng( youngest ) ) Then
    objLogFile.WriteLine Now & "   Exiting: lastRev (" & lastRev & ") >
youngest (" & youngest & ")"
    objLogFile.WriteLine Now & "     Script done"
    objLogFile.Close
    Exit Sub
  End If

  ' Compose the file name
  dumpFileName = folderName & dumpName & "-" & lastRev & "-" & youngest
& ".dmp"

  ' Add incremental, if not starting a new dump
  incremental = ""
  If ( lastRev > 0 ) Then
```

```
    incremental = " --incremental"
  End If

  ' Compose the dump command for the current repository
  dumpCommand = "C:\Progra~1\Subversion\bin\svnadmin.exe dump " &
repository & _
                " --revision " & lastRev & ":" & youngest & incremental

  ' Open the destination file and execute the dump command
  Set objDumpFile = objFSO.OpenTextFile( dumpFileName, ForWriting,
True )
  Set objExecDump = WshShell.Exec( dumpCommand )

  ' Read the dump output and write it to the file
  Do While True
       If Not objExecDump.StdOut.AtEndOfStream Then
            input = objExecDump.StdOut.Read( 1 )
            objDumpFile.Write input
       Else
         Exit Do
       End If
  Loop
  objDumpFile.Close

  ' Write the latest revision into the file
  Set objTextFile = objFSO.OpenTextFile( lastFileName, ForWriting,
True )
  objTextFile.Write youngest
  objTextFile.Close

  ' Close the log file and exit
  objLogFile.WriteLine Now & "     Script done"
  objLogFile.Close

End Sub
```

5. Upgrade

Subversion 1.2.0 was just released, and obviously I was pretty eager to upgrade. I ran into some minor issues, but there were no serious show-stoppers, and I'm going to skip them in this HOWTO, so everything should run smoothly.

At the same time I decided to upgrade the Apache server as well. The entire process is described in the next two sections.

5.1. Upgrading Apache 2.0 server

Well, Apache upgrade is a bit specific. It doesn't want to upgrade, so you need to uninstall the currently installed version first. There is nothing much to it, just go to the Windows Control Panel \ Add or Remove Programs and select Remove. But before you do that, remember to store your latest httpd.conf file in a safe location. C:\etc should do the trick.

After the un-installation you are left with a couple of folders, namely:

C:\Program Files\Apache Group\Apache2\conf
C:\Program Files\Apache Group\Apache2\logs
C:\Program Files\Apache Group\Apache2\modules

Since Apache installation doesn't like the fact that files and folder it is trying to create already exists, I suggest you rename the C:\Program Files\Apache Group\Apache2 to something original, like C:\Program Files\Apache Group\Apache2-old. That way you get to keep all your Apache log files. If you don't need them, you are welcome to delete the Apache2 folder all together.

Now you are ready to install the new Apache 2.0 server as described in section 2.3. Installing Apache 2.0. After that, all you need to do is put the old httpd.conf in the C:\Program Files\Apache Group\Apache2\conf folder, copy the mod_authz_svn.so and mod_dav_svn.so into the C:\Program Files\Apache Group\Apache2\modules folder, and you're done. You have just successfully apgraded Apache 2.0 server.

5.2. Upgrading Subversion

Now comes the wee bit tricker part of the upgrade, that is the Subversion upgrade. Why tricky? Well, there were some surprises with the version 1.2.0, but at least our backup file choice turned out to be right.

Before you begin you need to decide whether you will keep your repositories as they are, or reload them from the dump files.

I have decided to reload them from my backups, because of the speedup in repository operations. It now takes much less time to commit, update or simply get file contents at specific revision. But loading from the dump files takes some time, so if your backup has become too large, you may want keep the current repositories. If you want to keep your current database, this is what you need to do before you upgrade [taken from the Win32 release notes]:

```
***** IMPORTANT *** Upgrading from 1.1.x to 1.2.x *** IMPORTANT *****

In this release, we've upgraded BerkeleyDB from version 4.2.52 to
4.3.27.  If you are currently using Subversion 1.1.x as a server on
Windows with BerkeleyDB, use the following steps to upgrade your
repositories:

  -- Make sure nobody is using the repository (stop Apache/svnserve,
     restrict access via file://).
  -- For each repository, using the old (1.1.x) binaries:
     -- Run "svnadmin recover <repos-dir>";
     -- Create a backup of the repository;
     -- Run "svnadmin list-unused-dblogs <repos-dir>"
        and remove the listed files;
     -- Delete all the "<repos-dir>\db\__db.00?" files.
```

```
  -- Upgrade Subversion.

Once again, this is only necessary for repositories based on BDB.
You do NOT have to dump/reload your repositories.

********************************************************************
```

You may also want to change your repositories from BDB to FSFS, or vice versa. In that case you will need to reload your repository from the dump files anyway. So, let's start with the upgrade process.

Step 1
First you need to make sure that nobody can access your repositories while you're doing the upgrade, so stop the Apache server.

Step 2
Make sure that your last backups contain the latest revision stored in your repositories. You can check your backup names against each repository head revision number by hand

```
svnlook youngest C:\Repositories\project2
```

or you can simply run the backup script:

```
C:\windows\system32\cscript.exe c:\etc\backup.vbs
```

This will bring your backups up to date. You may want to transfer them to the tape, burn your CD, or whatever you store the backups on.

Step 3 is only needed, if you want to reload your repositories from the dump files. If not, you may skip it.

Step 3
If you have enough space on your disk, rename your C:\Repositories folder to something else, so you will have a binary backup of your current repositories.

Step 4
Now you can run the Subversion installation. The process is described in section 2.2. Installing Subversion, with a few minor differences.

The installation process detects that you are running an Apache 2.0 server, and notifies you, that it will stop its services, and restart them after the installation. Also, it offers to copy the mod_authz_svn.so and mod_dav_svn.so to the C:\Program Files\Apache Group\Apache2\modules folder for you, but there is an error in the current installation of Subversion, which prevents this from happenning.

After the installation is complete, check the Apache services, and stop them if they are running.

Steps 5, 6 and 7 only apply, if you have decided to reload your repositories from the dump files. If you have decided to keep your repositories unchanged, you may skip directly to step 8.

Step 5
Recreate the directories first:

```
    C:\Repositories\project1
    C:\Repositories\project2
```

Step 6
And the repositories themselves; the default storage for the repository has changed from BerkeleyDB to FSFS, so you need to specify --fs-type bdb explicitly if you want to create a BerkeleyDB repository.

```
    svnadmin create --fs-type bdb C:\Repositories\project1
    svnadmin create --fs-type bdb C:\Repositories\project2
```

Step 7
Now you need to reload the repositories from the backup files, using the load command.

```
    svnadmin load C:\Repositories\project1 < C:\backup\dumps\proj1-0-
53.dmp
    svnadmin load C:\Repositories\project1 < C:\backup\dumps\proj1-54-
64.dmp
    [ etc ]

    svnadmin load C:\Repositories\project2 < C:\backup\dumps\proj2-0-
32.dmp
    svnadmin load C:\Repositories\project2 < C:\backup\dumps\proj2-33-
109.dmp
    [ etc ]
```

Step 8
One last thing you need to do, is manually copy the mod_authz_svn.so and mod_dav_svn.so files to the C:\Program Files\Apache Group\Apache2\modules directory. The location of the files has changed with the version 1.2.0 and the msi script doesn't seem to be aware of that either. You can find both files in the C:\Program Files\Subversion\bin directory.

Step 9
There is just one final thing left to do, and that is start the Apache server.

Subversion is now upgraded to version 1.2.0, and if that is what you have decided, your repositories have been reloaded from revision 0 up with all your data, resulting in a faster repository operations.

Conclusion

This has been my experience with the Subversion. If you have any comments, suggestions, have better ideas, or plainly want to correct some colossal stupidity this HOWTO contains, you can contact me at mike5@eunet.si.

Please note, that this is my public e-mail address [think TONS of SPAM, and more to come after this, I guess :( ], so please start your Subject with "Subversion HOWTO" or something, to make it stand out from all the Viagra offerings, and unknown hot women trying to get a date with me :).

History of changes

20th March 2005 - Version 0.1 [Miha Vitorovic]
15th April 2005 - Version 0.2 - Changed < to < and > to > [Miha Vitorovic]
23rd April 2005 - Version 0.3 - XHTML 1.0 compliant, capitalized all the drive letters in the script [Miha Vitorovic]
17th May 2005 - Version 0.3 - Moved to new location, many thanks to Sebastian Schally. Renamed the linked files to match the document names.
24th May 2005 - Version 0.4 - Description of the upgrade process added. Sections 5, 5.1 and 5.2. [Miha Vitorovic]

=