

User guide

Typical Workflow

From the developer point of view, the workflow associated with a subversion repository is as follows:
(For a full set of commands type `svn help`)

- For each repository that you want to work with, do the following once only:
 - in your \$OPENEHR project area, create a directory of the name of the repository, or the name you want to know it by. The names of the repositories are given on the project pages ("project" button from [home page](#)¹).
 - get an initial copy of the repository of interest, using **svn checkout** or the GUI equivalent. For example, with TortoiseSVN, right-click on the newly created directory and choose SVN-checkout. You need to enter the repository URL, which is also given on the project page. NOTE: you can choose as deep a directory as you like within the repository. Typically you will choose the TRUNK directory or deeper, unless you are doing branch development work. For example, if the repository top URL is http://www.openehr.org/svn/ref_kernel_java, you should browse the repository using this URL (it is the "raw" view of a subversion repository on the web) until you find the point you are interested in. Then copy the appropriate URL to the checkout dialog appropriately.
 - Once you have set the URL, hit Ok and watch the files come down...
 - Note that the work directory now remembers what repository URL is it associated with. If you want a workspace with a different part of the same repository, you need to repeat the above operations.
- Making changes (you must be a member of the relevant team to have commit privileges):
 - create a new file - just create the file as usual
 - change a file - just make changes directly - the file will already be writable
 - renaming, moving, deleting files: **DO NOT RENAME or MOVE FILES DIRECTLY IN THE EXPLORER OR ON THE FILE SYSTEM!**
 - rename: you **MUST** use **svn move** src dest, or the GUI equivalent.
 - move a file to another location: you **MUST** use **svn move** or the GUI equivalent
 - move a directory to another location: you **MUST** use **svn move** or the GUI equivalent
 - delete a file or directory: you **MUST** use **svn delete** or the GUI equivalent
- Checking in changes to the repository
 - do an **svn update** if there is any possibility of the repository having been changed by others since your last checkout. - otherwise the commit operation will not work.
 - use **svn commit** --message "some log message" URL to commit the changes in your workspace to the repository at URL. Usually it is easier to use the GUI tools for this.

Only developers who are defined as users on an *openEHR* Subversion repository are able to commit changes. See the appropriate project page to find out about becoming a committer..

Making Subversion ignore files

Usually you want version control systems to ignore temporary, backup and generated files, created by
1. <http://www.openehr.org/>

compilers and other tools. The way to do this in Subversion is with properties. Every file or directory you want to ignore has to have the subversion property `svn:ignore` set. The proper way to do this is to set up "auto-properties" in the subversion client side config file. An [example of such a file](#)³ includes lines which are used by a Subversion client to automatically set properties at the time of file creation or committal.

You can verify that the correct properties are set on a file or directory by doing:

- command line: **svn proplist**, and **svn propset** some-prop-name
- Windows Explorer: right-click and investigate properties (usually last option on menu)

If you have forgotten to do this before committing files, you can directly change the properties on files using **svn propset**. The easiest way to do this is to use the find command (unix and cygwin) as in the following example:

```
$ find -name '*.o' -exec svn propset svn:ignore {} \;
```

If you are not comfortable with unix commands like this, ask the project leader to perform the changes on the repository, and you can simply update your copy, and the changes will be made.

File MIME Types

The MIME type of a file determines how it will be versioned (binary or textual) and also how it will be displayed by the Apache server.

Aborting Changes

TBD

Making changes to FrameMaker documents

This section only for confident Frame users!

The *openEHR* specification documents are currently in FrameMaker. Here's how to modify a controlled Frame document.

```
$ cd to appropriate directory in your workspace
```

Files are now editable in Windows explorer, double click on xxx.book. In Frame: do
Shift+File > open all file

Now you have all the files opened and writable.

Updating the Revision History in FrameMaker files

Proceed as follows:

1. Bring up front.fm; hit Esc,v,t (this makes editing characters visible).
2. In the revision history table:
 1. select the top data row with the mouse;

3. <http://www.openehr.org/downloads/developer/config>

2. do a ctrl-v to paste it
 3. Frame will ask whether to paste above, below or replace; **CHOOSE BELOW!**
 4. when the paste has been done, change the text appropriately in the top data row, **BE CAREFUL** not to lose the date and version special markers (little T characters in first and last column)
3. Make your changes to the content
 - Do a regenerate of table of contents & references
 - Save all files (use Shft+File)
 4. regenerate the appropriate PDF file, if there is one - look under /publishing in the appropriate subdirectory in the repository.