ASN2CSV SVN policy

Version 0.5

Revision History

Date	Version	Description	Author
2008-11-11	0.1	Initial Draft	ŽKn
2008-11-17	0.5	Rules are defined	ŽKn

Doc. No.:

1. Introduction

1.1 Purpose of this document

The purpose of this document is to define set of rules that all team members must abide when using SVN.

1.2 Document organization

The document is organized as follows:

- Section 1, *Introduction*, describes contents of this guide, used documentation during developing process etc.
- Section 2, *Rules*, defines set of rules for using SVN

1.3 Intended Audience

The intended audience is:
- Team members

1.4 Scope

This document contains information about how SVN should be used, and in particular, what is the procedure for creating and modifiying new files and folders. This document should be used in conjunction with Coding style guidelines.

1.5 Definitions and acronyms

1.5.1 Definitions

Keyword	Definitions
Team members	People who the are part of the development process

1.5.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
SVN	Subversion

1.6 References

2. Rules

2.1 Think Twice before Committing

Committing something to SVN has serious consequences. All other developers will get your changes once they are in SVN, and if they break something, they will break it for everybody. All commits will be publicly available in the SVN repository forever.

On the other hand SVN allows one to revert changes, so it's possible to recover from mistakes. This is relatively easy for commits to single files but it can also be a significant amount of work for bigger changes. The baseline is: Be aware of the consequences of your commits. Take time to think about them before committing.

2.2 Never commit code that doesn't compile

Compile the code and correct all errors before committing. Make sure that newly added files are committed. If they are missing your local compile will work fine but everybody else won't be able to compile.

You certainly should make sure that the code compiles with your local setup. You should also consider what consequences your commit will have for compiling with the source directory being different from the build directory.

2.3 Test your changes before committing

Start the application affected by your change and make sure that the changed code behaves as desired.

2.4 Double check what you commit

Do a svn update and a svn diff before committing. Take messages from SVN about conflicts, unknown files, etc. seriously. svn diff will tell you exactly what you will be committing. Check if that's really what you intended to commit.

2.5 Always add descriptive log messages

Log messages should be understandable to someone who sees only the log. They shouldn't depend on information outside the context of the commit. Try to put the log messages only to those files which are really affected by the change described in the log message.

In particular put all important information which can't be seen from the diff in the log message.

2.6 Respect other developer's code

Respect the policies of application and library maintainers, and consult with them before making large changes.

Source control systems are not a substitute for developer communication.

2.7 Announce changes in advance

When you plan to make changes which affect a lot of different code in SVN, announce them in advance. For instance, changes in libraries might break other code even if they look trivial, e.g., because an application must also compile with older versions of the library for some reasons. By announcing the changes in advance, developers are prepared, and can express concerns before something gets broken.

2.8 Take responsibility for your commits

If your commit breaks something or has side effects on other code, take the responsibility to fix or help fix the problems.

2.9 Don't commit code you don't understand

Avoid things like "I don't know why it crashes, but when I do this, it does not crash anymore." or "I'm not completely sure if that's right, but at least it works for me.".

If you don't find a solution to a problem, discuss it with other developers.

2.10 Don't commit if other developers disagree

If there are disagreements over code changes, these should be resolved by discussing them, not by forcing code on others by simply committing the changes to SVN.

2.11 Don't add generated files to the repository

Files generated at build time shouldn't be checked into the repository because this is redundant information and might cause conflicts. Only real source files should be in SVN. An exception to that are files generated by tools that would be an unusual requirement for building KDE.

2.12 Make "atomic" commits

SVN has the ability to commit more than one file at a time. Therefore, please commit all related changes in multiple files, even if they span over multiple directories at the same time in the same commit. This way, you ensure that SVN stays in a compileable state before and after the commit.

2.13 Don't mix formatting changes with code changes

Changing formatting like indenting or white spaces blows up the diff, so that it is hard to find code changes if they are mixed with re-indenting commits or similar things when looking at the logs and diffs later. Committing formatting changes separately solves this problem.