

API description for Netopeer repository library

Ondřej Zloský

17.11.2003

1 Motivation

Our motivation for writing a repository library was to provide higher-level functions to using them in frontends and backends and thus simplify the access to the repository. Programers of Netopeer's utilities want to put in their codes something like *conf_checkout(url)* and everything else could be hidden. They only need to get the requested configuration. Whether the repository is under Subversion, CVS or other version control system is not important.

Our API design tries to offer this functionality. The current version is written for Subversion because Subversion has some features that are good for our project. First, Subversion has it's own API that provides communication with repository through direct call of Subversion's functions. It is much better than writing some wrapper around Subversion, CVS or other clients. For example, we have Subversion's internal error structures, we can change visual output from functions and it is for sure that we can't forget something in wrapping. Second, Subversion provides the opportunity to use other programs instead of the usual text 'diff' so we can use 'xmldiff' that is better for our XML configurations.

The repository API design is still in progress and some functions could be incomplete or missing in this document.

2 Using repository API

2.1 API description

For using repository API you need to install Subversion including devel packages (or install Subversion from source). See the installation instructions on how to get and install Subversion and other tools properly.

The repository library is now written in the C language and every public functions in it has prefix 'conf_' for better orientation and in order to avoid naming

conflicts. Every function is well documented in the source code. The following functions are defined:

conf_list (char *repos_url);: Same as *'svn list url'*.

Return a repository listing with information about stored files

conf_checkout (const char *repos_url, char *target_path, char *rev_wanted);:

Same as *'svn checkout URL [path] [-r revision]'*.

Returns a repository directory and store it in target_path parameter

conf_copy (const char *source_path, const char *dest_path, const char *message);:

Same as *'svn copy URL URL -m "message"'*.

Copy a codument in the repository from source URL to destination URL where URLs are directories in repository

conf_diff (const char *repos_url, char *rev_wanted, const char *type);:

Almost same as *'svn diff URL -r revision:revision'*.

Return a diff between two revisions of the same file. The third argument 'type' is the type of diff wanted. It could be 'xml' for xmldiff or anything else for normal diff.

conf_import (const char *source_path, const char *repos_url, const char *message);:

Same as *'svn import path URL -m "message"'*.

Import new configuration from source_path to repository URL.

conf_commit (const char *items, const char *message);: Same as *'svn commit path -m "message"'*.

Commit changes made on working copy.

conf_status (const char *source_path);: Same as *'svn status path'*.

Show status of the files in the working copy.

Every function takes care about memory managment and administration of user context. It means that programmers of front&backends need not code these functions themselves.

2.2 Requirements

All you need is to include header file *repos.h* using

```
#include "parser.h"
```

and have the file *repos.c* in the working directory. This is an interim setup, the final solution will use a shared library. For now, however, it has to be built directly into clients.

2.3 Compilation

Compilation is system dependent respectively in that it depends on installed libraries and parameters with which Subversion was compiled. Check the latest Makefile to see the actual versions of libraries necessary for a successful compilation

2.4 Other planned functions

- Syntactic control. Our installation of Subversion implements syntactic validation of configurations. It is done by a *pre-commit* hook installed in */path/to/repos_db/hooks*. Semantic checker is a bash script which uses a program written by Petr Novak.
- Access control through WebDAV. Controlled access should be applicable to every configuration separately.
- Unified configuration list independent of the version control system (Every control version system has its own numbering and Subversion's numbering scheme is not satisfactory for our purposes.)

3 Subversion repository installation

Subversion repository can be installed either from source using the usual procedure or as pre-built binary packages. The latter approach is slightly easier. When installing from source, you should have all the libraries and header files that are needed for a correct operation of the Netopeer's repository library. When installing from binary packages, you have to install devel packages as well.

Subversion¹ can be installed on various operating systems and pre-built binary packages are available from the download page² of Subversion.

3.1 Installation from source code

3.1.1 Client

Client-only installation of Subversion is simpler and depends on just a few software packages. With this installation you can use actions like *checkout*, *commit*, *list* and others.

¹<http://subversion.tigris.org>

²http://subversion.tigris.org/getting_subversion.html

In order to compile and use the Subversion client, you need to have the following:

1. Apache Portable Runtime 0.9.4³ (included in Subversion distribution)
2. autoconf 2.50 or newer
3. libtool 1.4 or newer
4. bison or yacc
5. Neon library 0.24.2⁴ (included in Subversion distribution)

After downloading the latest source you can compile and install it using the standard procedure: *./configure*

make

make install

For advanced functions you may want to use few special parameters for *./configure*:

- *-enable-maintainer-mode* - turns on debugging
- *-with-ssl* - turns on ssl communication with neon - you need to have OpenSSL installed on your system
- *-with-zlib* - turns on support for sending compressed data using the zlib library
- other parameters can be found by *./configure -help*

The installation can be tested by running some svn commands, e.g., *svn co http://svn.collab.net/repos/svn/trunk svn*. If everything works, after checkout you will have files from the Subversion's tree in the svn directory.

³<http://apr.apache.org/>

⁴<http://www.webdav.org/neon/>

3.1.2 Server

In addition to all third-party tools needed for the client installation, the Subversion's server requires:

1. Berkeley DB 4.0.14 or newer⁵
2. Apache Web Server 2.0.47 or newer⁶

If these components are present in your system then Subversion will be compiled as a server. You can do it by the same procedure again:

```
./configure  
make  
make install
```

If you have Berkeley DB in a non-standard location, you can add the parameter `-with-berkeley-db="/path/to/berkeleydb/"` and a non-standard location of Apache can be specified by the parameter `-with-apxs="/path/to/apache/apxs"`.

After installation, you can test the functionality of the server by:

```
svnadmin create /path/to/repos/  
svn co file:///path/to/repos
```

and you should see `Checked out revision 0.` without any errors.

3.1.3 Note

There are many other configuration options and details enabling or disabling special features, libraries, etc. This is just a mini guide on how to install Subversion if there are no specific troubles and/or requirements. If you encounter problems or miss some functions, please consult the Subversion INSTALL file or documentation.

3.2 Installation from pre-built binary packages

You can download binary packages from <http://summersoft.fay.ar.us/pub/linux>⁷. All other packages that are needed for Subversion can be downloaded from usual distribution-specific places. For example, on a RPM-based Linux distribution, you can install Subversion by running the following command as root:

```
rpm -ivh subversion*.i386.rpm
```

After installation you can test the functionality by running the same tests as before.

⁵<http://www.sleepycat.com/download/patchlogs.shtml>

⁶<http://httpd.apache.org/download.cgi>

⁷<http://summersoft.fay.ar.us/pub/linux/redhat/RPMS/i386/subversion/>

4 XMLdiff installation

XMLdiff is used by the Netopeer configuration repository is XMLdiff for identifying changes in two configurations. Because our configuration data are in the XML format, it is better than the normal text diff. Subversion can use various diff commands and our library supports this function too.

4.1 Requirements

XMLdiff is written in Python. You need to have Python⁸ version 2.0 or higher. For extended functions of XMLdiff the PyXML module is also needed.

4.2 Installation

Download XMLdiff⁹, untar it and install by running:

python setup.py install or

python setup.py help to see advanced parameters.

Or you can download rpm from the internet and install it (as root) simply by:

rpm -ivh xmldiff.rpm*

⁸<http://www.python.org/>

⁹<http://www.logilab.org/xmldiff/>