An Overview of Subversion for ESP-r Central Users
Version of January 4, 2008

ESP-r's source code was placed under a version control system in April 2006 to facilitate its management and development. This version control system is called Subversion. Subversion maintains a record of changes to the source code and provides a means for dealing with concurrent changes to source files. It also allows developers a convenient method of contributing to the ESP-r code base. This document describes how to use Subversion to checkout a working copy, to add and remove files, and to commit files back to the ESP-r code base. It is also useful to read the other documents in the *archive* folder which complement this document.

Revision history

This document is under versioning control, and suggestions and contributions are strongly encouraged. The troff-formatted source file for the latest version can be obtained at the following url:

https://esp-r.net/espr/esp-r/branches/development_branch/src/archive/subversion.trf

To generate an A4 postscript document from this file via the groff suite of tools (available for many operating systems) issue the following command:

cat subversion.trf | eqn | tbl | groff -mms -dpaper=a4 -P-pa4 > subversion.ps

1. Introduction

What is a Subversion Repository?

A Subversion repository is both a storage area for project source code and a tracking system for source code changes. It keeps track of the history of changes to every file and directory contained within it. The ESP-r Central Subversion repository ensures that all developers and users have the most up-to-date version of the ESP-r source code. Subversion also allows for developers to commit their bug-fixes and/or enhancements to the repository.

For a complete description please refer to Chapter 2 of Versioning control with Subversion, by Collins-Sussman, Fitzpatrick and Pilato:

http://svnbook.red-bean.com/en/1.1/ch02.html

Obtaining a Subversion Client

Command line clients for various operating systems can be downloaded from the subversion website:

http://subversion.tigris.org/project_packages.html

There are also a number of graphical user interfaces that can be used to access a Subversion repository. Due to variations in their use, they are not documented here. Most of these tools also provide a command line mode, and this documentation is still applicable.

2. Anonymous access for Non-Developers

If you do not plan to modify ESP-r source code, you may download a working copy using anonymous access. This is convenient for students and professionals who simply wish to download and compile the latest version of ESP-r or make minor modifications such as increasing the maximum geometric complexity by adjusting parameters in the source code header files.

The following command will download ESP-r's source code from the ESP-r Central repository to the current working directory of your local computer:

```
svn checkout  https://esp-r.net/espr/esp-r/trunk/
```

With this working directory, you will be able to compile ESP-r on your own computer. Though you will also be able to alter the source code, you, will be unable to make changes to the ESP-r Central repository. The *trunk* is the official release that is updated a few times per year. If you want the latest version you would use an alternative checkout command:

```
svn checkout  https://esp-r.net/espr/esp-r/development_branch/
```

If you're contemplating modifying ESP-r, are strongly encouraged to obtain a developer's account. The GNU public license includes a provision that changes you make to ESP-r should be shared with the community and such contributions are made via subversion commands described in this document and the other documents in the *archive'* folder.

3. ESP-r Development with Subversion.

If you wish to make changes to the ESP-r Central repository, you must become familiar with the concepts of working on ''branches'', merging your changes back into your branch in the repository and documenting your work so that others in the community can take advantage of the changes that you contribute. These concepts are not trivial, and it is very important that you become comfortable with them.

For a complete description of branches and merging, please refer to chapter 4 of Versioning control with Subversion, by Collins-Sussman, Fitzpatrick and Pilato:

http://svnbook.red-bean.com/en/1.1/ch04.html

ESP-r development occurs on separate ''sub-branches'' that are assigned to individual contributors or teams of contributors. Contributors modify and commit their code on to these sub-branches, where they can be inspected by others. After completing a rigourous testing process, the ESP-r archivist merges contributions from a sub-branch on to the main development branch, where they can be accessed by all developers.

A complete description of the ESP-r Subversion branch structure is available in the document ''Structure of the ESP-r source code archive'' which also includes further suggestions for how development work is managed within the ESP-r development community.

Obtaining a Developer's Account and Sub-branch

In order to work on a sub-branch, you must first have an ESP-r developer's account and a sub-branch name assigned to you. To obtain an account and sub-branch, contact Alex Ferguson (aferguso@nrcan.gc.ca).

Checking out a Sub-branch

You must perform a repository ''checkout'' to obtain a working copy of your sub-branch. A ''checkout'' will download the module into your current working directory, where you can compile, alter the source code, and ''commit'' your changes back to your developer-specific sub-branch for others to view. To perform a checkout, use the following command:

```
svn checkout https://esp-r.net/espr/esp-r/branches/<sub-branch_name>
```

Provide your developer's account name and password when prompted.

Note that changes that you make in your working directory are NOT recorded in your branch of the repository until you issue a relevant subversion command.

4. Common Subversion Commands

There are many commands available in Subversion; the following are the most common commands you will use. For a more extensive list, please refer to Chapter 3 of the book Versioning control with Subversion:

http://svnbook.red-bean.com/en/1.1/ch03s05.html

None of these commands will affect other developer-specific sub-branches, they only affect the sub-branch you have been assigned to work with.

ADD and DELETE files

Use these commands to schedule adding/removing files or directories to/from the sub-branch you are working on. Additions and deletions will only take effect in the repository once you perform a ''commit'' command.

```
svn add <path_to_file_to_be_added>
svn delete <path_to_file_to_be_deleted>
```

Check the STATUS of your workspace.

This command will list all the files you have changed relative to the sub-branch you are working on, which is handy to use before a ''commit'' or an ''update''.

```
svn status <directory_or_filename>
```

If the status list is long you may wish to re-direct the output to a file named current_status.txt use the following command:

```
svn status <directory_or_filename> >current_status.txt
```

If you see a file marked with a '?' in the status list this signals that it is not known within the repository. If you want the file to be known then issue the following sequence of commands:

```
svn add <path_to_file_or_folder_to_be_added>
svn commit <path_to_file_or_folder_to_be_added>
```

If the file or files to be added are the only pending tasks then you could issue a more general command:

```
svn add <path_to_file_or_folder_to_be_added>
svn commit
```

Remember that files within the working directory which are not part of the repository risk being lost. Some files should not be included in the repository - for example, object files and executables created during the compile process are not part of the repository. Typically only the ASCII version of databases are included in the repository (binary versions are created during the Install process).

UPDATE files/directories of your workspace.

While you'll initially work on your own personal branch, you may also be involved in collaborative projects requiring several developers to share the same branch. In these projects, you will need to periodically update your working copy with changes other developers have committed to the project branch. The ''update'' command will update your local copy with any changes that other developers have committed to the project branch since your last checkout/update. *But be careful!* It automatically merges code into your files, so inspect all updated files and ensure your code still works correctly. Some developers create a local archive or backup copies of files which are work-in-progress prior to issuing ''update'' commands.

```
svn update  <directory_or_file_to_be_updated>
```

A *conflict* may occur if changes in the repository affect the same files you've modified in your local copy. Conflicts are discussed in detail below.

COMMIT your changes to the repository.

This command will commit all file changes, as well as Adds and Removes, to the branch you are working on. See the repository document for further advise on how to plan your commits. Only valid ESP-r developer account holders can use this command to update their branch of the repository or joint branches which they may be working on.

```
svn commit <directory_or_filename>
```

A text editor will be opened after you issue the above command. You can nominate which editor to use by setting the SVN_EDITOR environment variable. Enter a detailed message that elaborates the reasons for your coding change/addition, the intent of your code, and the testing that you have conducted. You should indicate in detail what impact this change has upon ESP-r functionality, and in particular, the impact it has upon calculation results. This message will be permanently recorded in ESP-r Central's repository log and will act as a reference for other developers and for yourself in the future. Use proper sentence structure and grammar to effectively communicate this critical information to your colleagues.

Within 24 hours of committing your changes, you will receive an automatically-generated test report comparing the new version you've submitted with the previous version on your sub-branch. This test report will tell you if your new version compiles correctly in various configurations, and will also highlight any questionable syntax and potentially erroneous code introduced by your commit. Note that this report is based on differences between your current commit and the previous state of your branch. To review the full syntax report you will have to run the syntax checking software yourself.

5. Conflicts

Conflicts arise when changes received from another developer, during an update or merge, overlap with local changes that you have in your working copy. You must resolve these conflicts before committing your changes to the repository. Subversion will flag files in conflict with a ''C'' after an update or merge:

```
svn update
```

```
---------------------- (OUTPUT) ---------------------------------------
U  Install               <-- U indicates the file Install updated
C  esrubps/bps.F         <-- C indicates conflicts exist in esrubps/bps.F
Updated to revision 3.  <-- Notification of update to revision number
```

Subversion will not allow you to commit any files until the conflict is manually resolved. A full discussion on resolving conflicts can be found in Chapter 3 of Versioning control with Subversion, by Collins-Sussman, Fitzpatrick and Pilato:

http://svnbook.red-bean.com/en/1.1/ch03s05.html#svn-ch-3-sect-5.4

6. Merging changes from the development branch

ESP-r Central's development branch will be updated periodically as developers submit new features and bug fixes for inclusion in ESP-r. As the maintainer of your sub-branch, it's your responsibility to merge these changes into your sub-branch. Your contributions will not be accepted into the development branch until you've synchronized your branch with the development branch, proving your code is compatible with the current source from the development branch.

It is very important to use a clean working copy when merging changes from development branch. The archivist cannot correctly incorporate your contributions onto development branch if you commit them along with merged changes from development branch. *If you do not use a clean working copy when merging changes from development branch, the archivist may ask you to revert the changes on your sub-branch, and perform the merge again using a clean working copy!*

The following example illustrates the steps involved in updating your sub-branch with changes in the development branch for the first time:

1. Determine the revision number at which your branch was created:

```
svn log --verbose --stop-on-copy \
    https://esp-r.net/espr/esp-r/branches/<sub-branch name>

---------------------- (OUTPUT) ---------------------------------------
r12 | author | 2006-04-25 10:31:38 -0400 (Tue, 25 Apr 2006) | 3 lines

This update make some more changes....


------------------------------------------------------------------------
r11 | author | 2006-04-25 10:26:58 -0400 (Tue, 25 Apr 2006) | 3 lines

This update makes some changes ...


------------------------------------------------------------------------
r10 | ibeausol | 2006-04-25 08:53:27 -0400 (Tue, 25 Apr 2006) | 3 lines

This update makes a copy of the development branch for use by
Author.

------------------------------------------------------------------------
```

   Note revision r10. This is the revision number at which your sub-branch was created.

2. Check out a clean copy of your work:

```
svn  checkout https://esp-r.net/espr/esp-r/branches/<sub-branch name>
```

3. Determine the most recent revision (i.e. the highest number) on the development_branch.

```
svn info https://esp-r.net/espr/esp-r/branches/development_branch
---------------------- (OUTPUT) ---------------------------------------
Path: development_branch
```

```
URL: https://esp-r.net/espr/esp-r/branches/development_branch
Repository Root: https://esp-r.net/espr/esp-r
Repository UUID: 7d53e970-de11-0410-8a54-3d01b9da36cf
Revision: 385
Node Kind: directory
Last Changed Author: ibeausol
Last Changed Rev: 355
Last Changed Date: 2006-07-28 08:03:06 -0400 (Fri, 28 Jul 2006)
```

Note the current revision number (385).

4. Merge the changes that have occurred on development_branch from r10 (when your sub-branch was created) to r385.

```
cd <your_working_copy_from_step_2>
svn merge -r 10:385 https://esp-r.net/espr/esp-r/branches/development_branch
---------------------- (OUTPUT) ---------------------------------------
U integer.c
U button.c
U Makefile
```

6. Check to see if there are any conflicts and check the changes that have been merged. In the case that there are conflicts between the local changes and those on the development branch subversion will create a *left* and *right* version of the source file and it will embed in the source files markings <<<<<<< or >>>>>>> indicating where the conflict is located. Manually edit the source file and if you are happy with the result issue a "svn resolved" command. If you want to take the development branch version execute an "svn revert". To check the status of your work issue the command:

```
svn status
---------------------- (OUTPUT) ---------------------------------------
M integer.c
M button.c
M Makefile
```

7. Commit the merged changes into your sub-branch and provide an appropriate log message. You will rely on this message the next time you merge in changes from development_branch; rather than merging from the creation point of your sub-branch (r10 in the example above) you will merge from the last point at which your synchronized (r385 in this example).

```
svn commit -m "Merged development_branch changes r10:385 into <sub-branch name>."
---------------------- (OUTPUT) ---------------------------------------
Sending integer.c
Sending button.c
Sending Makefile
Transmitting file data ...
Committed revision 386.
```

For more information on Subversion Merging, read Chapter 4: Branching and Merging of Versioning control with Subversion, by Collins-Sussman, Fitzpatrick and Pilato:

http://svnbook.red-bean.com/en/1.1/svn-book.html#svn-ch-4