Eric 4
as
Python 2.x
Integrated Development Environment

Version Control
Technical Report
Dear Reader,

So now, after the main “Eric 4 as Python 2.x Integrated Development Environment” and the subsequent “Eric 4 – Web Browser” [see], we enrich the collection of Technical Reports with this “Eric 4 – Version Control”, a booklet dealing with the collaborative development feature integrated into this unique IDE.

Having to do with a collection, for all subjects of general scope, introductory Foreword included, you’re kindly invited to refer to the named “Eric 4” central volume, but anyhow we deem worth recalling here too the basic principles characterizing this whole work. In summary:

• All what you’ll read here is the result of a test & documentation project carried on in fair collaboration with the Eric producer but, nevertheless, in total independence from him.

• Besides positive technical documentation and instructions, also our possible doubts, perplexities and difficulties have been here fairly reported.

• Technical dialogue is strongly encouraged, to the obvious benefit of all involved parties, and on which we'll be glad to rely for the future editions of this work. Thank you.

See you.

The Author

- = -
Essentials

Essential reference data, so to precisely identify the product we are talking about, its hosting environment and running action.

S–PM 120700

Version Reference

<table>
<thead>
<tr>
<th>Nr</th>
<th>Item</th>
<th>Version</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Windows XP Pro</td>
<td>5.1 (SP 3)</td>
<td>Host Operating System</td>
</tr>
<tr>
<td></td>
<td>Windows Vista Ultimate</td>
<td>6.0 (6002:SP 2)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>\Python27</td>
<td>2.7.1</td>
<td>Host Directory</td>
</tr>
<tr>
<td>3</td>
<td>Eric4</td>
<td>4.5.4 (r4349)</td>
<td>IDE Application</td>
</tr>
<tr>
<td>4</td>
<td>PySvn</td>
<td>svn173-1.7.6-1457</td>
<td>VC System API</td>
</tr>
</tbody>
</table>

Subject  “Version Control System” (VCS), short form: “VC System” (VC)

Title  “Eric4 Version Control”, short form: “E4-VC”

Prerequisite  Same as for the Eric4 host main product [see: “Eric4 – Technical Report”].

Execution  Main dedicated Eric4 menu section: Project > Version Control, plus some other context and related menus as listed at chapter: 4. Eric4 Version Control – Menu Commands, Menu Sections [see].

True Multi-Platform

Eric is a true multi-platform product, as ubiquitous as Python. That is, in principle, equally usable and compatible with all such platforms as: Windows, Macintosh and Linux, where it operates in the same way, but for some clearly unavoidable differences.

In this Report, however, and for many good reasons, we have chosen to make precise reference only to just one of these platforms: Windows [see table: Version Reference], leaving to each user the freedom of choosing any other compatible environment he wants, along with the the consequent burden of taking care of the necessary and, hopefully, easily manageable specific differences.
1. Introduction to the Eric4 VC System

**Viewpoint**

Be prepared: VC System in general, and with Eric (4) in particular, is a rather rich, innovative, useful, powerful, confused, unstable, parochial, incoherent, childish, evolving and increasingly relevant new environment, which is here to stay and which you cannot afford to ignore any longer.

And, if just arrived there from the outside world, you dared to ask clarifying questions, be prepared to be looked scornfully at by the local residents, wondering at your lack of understanding of their sectarian dialects, peculiar accents, obscure jargon, mysterious sign-posts and sign-boards.

Well aware of all that, having hardly survived ourselves, we opened this book with a hopefully clarifying and coherent set of concepts and definitions, followed by operative instructions based on facts, and interspersed with unresolved doubts and unanswered questions. Sort of a first exploration guide, designed to possibly be of some help to next adventurous newcomers.

Be prepared. And let us know, and learn, from your experience too.

The Eric IDE program has clearly been designed to fully honor the “Integrated” attribute contained in its definition, in the sense that it is really meant to grant all tools a s/w development activity will require. Projects' Version Control included.

“Version Control System” (VCS), or Revision Control System (RCS), or Source Code Management (SCM), is how are currently—and rather inexpressively—called the extraordinary s/w tools enabling collaborative development of projects, executed in network. Typically projects of technical documentation, mapping, or s/w development, that is projects particularly suited to be stored and managed as computer files, and carried on by operators working independently, in different geographical locations and with different timings.

As a comprehensive IDE, Eric can integrate and become compatible with a selected Version Control System, to be used directly within an Eric session. “Can integrate” means that the availability of such VC System is not mandatory, means that it doesn't constitute an Eric prerequisite as, for instance, the Python 2.x programming language and the PyQt4 GUI library are [see prerequisites, at the main “Eric 4 Technical Report”]; but that such VC System can be easily added to and used with Eric, would you want to.

Purpose of this E4-VC booklet is to describe:

- Which is the VC System that can be integrated into Eric, and how;
- Which are the Eric commands provided to manage and use it.

For clearness' sake, be aware of what is NOT a purpose of this booklet: to treat the VC System as

---

1 See our related “Viewpoint – In conclusion” as expressed at the end of this Report.
considered in general, as a subject in itself\(^2\). That is also to say that, understandably enough, VC System will be here treated only in relation with this Eric (4) hosting IDE.

**Which VC System: Subversion, via PySvn**

Of the various VC Systems currently available, Eric4 is declared compatible with Subversion, as described in its dedicated web site:  [http://subversion.apache.org][1]. More precisely, the original project Subversion revealed vital enough to inspire the production of a whole set of other distinct and valid incarnations and offsprings; such as: Application Programming Interfaces (APIs) library, command console, and GUI application packages.

\(<!>\) Of all these incarnations in this Report we'll consider exclusively the PySvn API library as integrated into Eric4, and as detailed in Appendix, section “Subversion Packages: Download & Setup” [see].

**Subversion vs. E4-VC**

For clarity's sake it's worth here recalling that, in this Report, with “Subversion” we intend such VC System considered in its complete capability, whereas with “E4-VC” we intend the same VC System as available and usable within Eric4.

**Remark**

In addition to the fact that this Report is dedicated exclusively to E4-VC operating with the PySvn API library, you may be interested in knowing that Eric4 is currently declared compatible also with these other VC Systems:

- **svn** Subversion command line client\(^3\), via such package as slik-subversion [see section: VC System Console]
- **CVS** “Concurrent Versions System”, a Subversion precursor; via a specific Eric4 add-on [see the Eric official web site]
- **Mercurial** Recently announced as back-ported from Eric5, via a specific Eric4 add-on [see the Eric official web site]

Possibilities here simply cited, but not treated in any practical way.

\(^2\) Just in case, here some specific reference books:
- **Version Control with Subversion (1.6)**, by Collins-Sussman, Fitzpatrick, Pilato; ed. SoHoBooks, 2009
- **Version Control with Git**, by Jon Loeliger, ed. O'Reilly, 2009
- **Pragmatic Guide to Git**, by Travis Swicegood, ed. Pragmatic Bookshelf, 2010

\(^3\) By the way, hence it's why the E4-VC user is offered such VC System Selection drop-down list box, as shown at the Common Dialog Boxes section [see].
Key-Concepts and Keywords

Section summarizing the conceptual building-blocks required to describe and work with E4-VC, here treated only to this specific purpose, that is without any pretension of completeness or universality.

Centralized vs. Distributed Repository

One of the main distinctions amongst different Version Control Systems lays on the so called “Repository”, that is where the projects under development are stored. And the distinction is whether there is such thing as a Central Repository, or not. By the way: Subversion does have a Central Repository.

Of course it's anyhow obvious that, if you are to generate any real project, at some point of the story such thing as a unique & final “Main Repository”, or “Reference Repository”, or whatever you may call it, must exist. But, while the work is in progress, you may have the choice between Version Control Systems relying onto a unique Central Repository, or not.

In the first case, each workstation will operate on a local repository that, from time to time, is to be synchronized with a Central Repository, by each designer, operating in a client–server scheme.

In the second case, each workstation will operate autonomously on its own VC Distributed Repository, starting from an original repository reproduced or, as usually said, “cloned” locally. A Repository that, from time to time, will be synchronized somewhere into a unique version, as an “administrative” action, care of the top project management.

Client vs. Server Functions

As said, this E4-VC Report is centered on Subversion, which is a Centralized Repository VC System. More precisely Eric4 integrates exclusively the VC Client function of Subversion, whereas the VC Server function is assumed to be managed independently from Eric, operating directly with a suited Subversion package. A fact that has also a practical consequence for this E4-VC Report.

In fact, having the purpose of testing the Client side of Subversion, as integrated into Eric, we'll necessarily make use of the very Server function of Subversion for building the appropriated environmental conditions of use for the Eric IDE. Therefore we'll have to describe here also the Server side of Subversion, to some conveniently limited extent [see next chapter: 2. Basics of Subversion Management].

---

4 In VC System terms, it's simply matter of permission level assigned to an operator.
5 Whereas, for instance, Mercurial VC System, natively integrated into Eric 5, is a Distributed Repository VC System.
**VC Keywords**

**Repository**

It's the unique centralized container of an Eric Project when versioned with such a Version Control System as Subversion. In operative terms, it is a directory accessible in Client-Server fashion in a network of developers.

When created and managed as Server it is identified by its path, according to the rules of the hosting file system. In this Report, we'll prevalently use a path according to the standard Windows “drive” Local File System (LFS); such as:

```
C:\Users\User111\svnRepository
```

From the E4-VC point of view, a Repository is an existing resource, available to Eric but created and maintained externally from it. In practice, for this Report, we'll make use of a Repository created and maintained at system level, making use of the Subversion `svnadmin.exe` program [see section: Creation of a Local `svn`-Repository].

When used as Client, a Repository is identified by a special `svn`-URL [cf. hereafter]. In practice, for this Report, we'll prevalently use such URL schema as: “file://” or, sometimes: “https://”

Other possible `svn`-URL types will be not here considered, simply because out of our reach.

**Eric Project**

A specific and well known Eric concept, with no precise correspondence within Subversion as such\(^6\), but certainly valid and useful with its E4-VC implementation. In this Report we'll refer to E4-VC projects as “versioned”, to tell the difference with the other possibly “unversioned” ones [cf. hereafter].

**Working Copy**

Is the local incarnation of a versioned project on which a developer works autonomously on, independently from the other possibly concurrent developers, and relying upon the E4-VC mechanism that will grant the harmonious integration of all contributions into a final common versioned Eric Project.

From the Eric point of view a Working Project is a usual Eric Project enriched by the versioning benefit. Of course there is a way to tell if what you are dealing with is a versioned project, or not; in fact, two:

- The directory hosting a versioned Eric Project will host also a specific “`.svn`” Windows hidden directory\(^7\).
- Opening a versioned project, the command menu Project > Version Control will switch from its “reduced” version, as in section 2.1/3 Initial Menu: Project > Version Control [see], to the “full-fledged” version, as in section 2.2/3 Main Menu: Project > Version Control [see].

**Master Copy**

Is referred to the Repository contents, what the Working Copy is derived from.

---

\(^6\) For instance, there is no specific reference to such “Project” concept within any `svn.exe` command.

\(^7\) That you may find convenient to show, via Folder Option: “Show hidden files and folders” [see].
1. Introduction to the Eric4 VC System

Versioned

Is how are usually—and, to some extent, rather inexpressively\(^8\)—defined the projects developed under a Version Control System, as here considered.

Inversely, the projects not enjoying this condition are defined as “unversioned”.

URL

Stands for “Universal Resource Locator”, the address used to identify items available in Internet. The Subversion VC client, for its own specific needs, must adopt always, even when dealing with a Repository located locally, a special \texttt{svn}-URL syntax of this kind: \texttt{file://localhost/svnRep/}

Where “\texttt{file://localhost}” refers to the \textit{schema protocol} in use\(^9\). That is, in this case, a usual Local File System.

Other notable schema protocols are:
\begin{itemize}
  \item \texttt{file:///} as short form of \texttt{file://localhost/C:/}, for current working drive C;
  \item \texttt{file://localhost/X:/}, \texttt{file:///X:/}, \texttt{file:///X//}, for generic local drive unit X;
  \item and: \texttt{http://}, \texttt{https://}, \texttt{svn://}, \texttt{svn+ssh://} [see section: \textit{Which Data Protocol}].
\end{itemize}

Path

Is how is called the address used to identify items contained in a computer file system\(^10\). It can assume different aspects in different systems and environments. In Windows most notable syntax is the so called:
\begin{itemize}
  \item “drive” Local File System (LFS), e.g.:
    \texttt{C:\Dir\SubDir\File.ext}
  \item and “share” Universal Naming Convention (UNC), e.g.:
    \texttt{\\Computer\Share\Dir\File.ext}
\end{itemize}

Server / Client

Usual distinct tools and roles in a network, here assumed as fairly known to the reader. In this specific Subversion Version Control System game, E4-VC will play exclusively the role of a Client.

\footnotesize
\begin{itemize}
  \item \(^8\) See our “\textit{Viewpoint – In conclusion}” as expressed at the end of this \textit{Report}.
  \item \(^9\) Note here the “/” forward backslash separator, instead of the “\" backslash separator typical of the standard Windows paths.
  \item \(^10\) Far from exhausting such a vast theme, here we’ll merely recall some elements useful within the scope of this \textit{Report}.
\end{itemize}
Versioning Layout:

**Repository, Project, Trunk, Branches, Tags; and Working Copies**

A VC System in general, and Subversion in particular, is a centralized system enabling the sharing and collaborative development of meaningful collection of data\(^{11}\), such as an Eric software project. An Eric Project developed this way is referred to as versioned.

**Repository and Working Copies**

At the top of the versioning process there is the availability and use of a common data container, called Repository. At the other end of the versioning process there are the so called Working Copies, locally available for development. Working Copies are obtained from the central Repository via such E4-VC commands as: New from Repository... and Update from Repository [see]. Edited Working Copies goes back to enrich the Repository data via such E4-VC commands as: Commit Changes to Repository and Add to Repository... [see].

Then E4-VC offers also a comprehensive set of tools aimed at monitoring and supporting the versioning process, and at preventing and solving conflicts of data possibly generated by different contributors in different times.

**Revision No.**

At Repository level each new contribution to the common data is tracked by means of an integer Revision Number, used to precisely refer to each single step of the development. Each Revision Number refers to the entire set of data within the Repository, is managed exclusively at Repository level and can be seen and used as a reference by E4-VC commands.

**Trunk and Tags**

A Project under development, for its very nature, is a dynamic entity, evolving with time. To obtain and release a Product out of a Project it is necessary to “freeze” it—that is: to make a read-only snapshot—at a given Project's version and revision level. Such snapshot copy, in Subversion terms, is called a Tag, and is generated at Repository level via E4-VC command Tag in Repository... [see]. This way we get a stable Tag-Product, whereas the Project, as such, can keep evolving with time. In usual Subversion terms, to tell the difference from a Tag, the main development line of a Project is called Trunk, necessarily unique for each given Project.

Trunk and Tags are part of the standard Repository layout of a Subversion versioned Project. Such a structure is not mandatory, it's simply convenient and used normally. Then there is a third element in this standard layout: the so called Branches.

\(^{11}\) In principle, any kind of computer-storable data can be versioned. With this Report, though, we're not dealing with Subversion in general, but as it is integrated into Eric, as E4-VC, which represents a specialized use of this technology.
Branches

At a certain point of the development of a complex Project it may become convenient for one or more contributors to focus their activity on a well delimited aspect of the Project, with all surrounding aspects remaining stable. In such a case it may come handy to derive a so called Branch out from the main Trunk, at a given Revision level. Versioned development will proceed on Working Copies relative to such a Branch, instead of the entire Trunk. At least until it becomes convenient to merge this sort of sub-Project into the main stream, that is into the Trunk. Branching technique is standard Subversion and E4-VC procedure.

Standard Repository Layout

In conclusion you may have a versioned Project structured into: one Trunk (that is: main line of development), some Branches (that is: sub-Projects) and some Tags (that is: released Products); and subsequent Working Copies.

Custom Repository Layout

Such Repository Layout, as defined in the former section, might be considered “Standard”, but it's not at all compulsory. It's perfectly possible, if not even wise, to operate with nothing more than a simple custom Repository, and then with the consequently derived Working Copies.

To that purpose just un-check the “Create standard repository layout” check-box, at the Project > New... command action, and then you'll be happily using a compact, much simpler layout as you'll find out looking at the “Importing project into Subversion repository” form, as it would appear in this case:

![Subversion Importing Project Dialog](image)

and then comparing it with the other one, as shown hereafter at the 1/3 Menu Section: Project > New..., where the “Standard” layout is accepted [see].
Remark

Dealing with different svn-Repository layouts—that is: Standard and not-Standard—brings about two mildly annoying consequences you'd better be aware of:

- You'll have to manually control the “standard repository layout” check-box at each subsequent svn-Working Copy creation, as the E4-VC default condition may not correspond to the current svn-Repository layout.

- As in other similar circumstances, once taken the initial decision in favor of Standard or not-Standard svn-Repository layout, E4-VC will not allow you to change your mind about this matter later on.

svn-Repository and Eric Projects

Now just a hint about a possible multi-Project Repository. It is true that, in principle, it's possible to create more than one Eric Project into the same Repository, but this Report will not encourage such possibility, and for some good reasons.

One of these reasons is that the svn-Revision No. is anyhow unique within a Repository, not within each Eric Project, and that also in case of a multi-Project Repository. Indeed, as a matter of fact, the very concept of Project, as intended by Eric, is unknown to, and ignored by, Subversion.
2. Basics of Subversion Management

E4-VC is designed to offer solely the client functions of a VC System, assuming that all the server and administrative functions are to be executed independently, at another level. But, of course, to thoroughly test all VC client features, we need also a test Repository available to “play” with. That is, we need some autonomous capacity of basic VC server management. As typically, and foremost, the capacity of creating, using and deleting a svn-Repository at will.

Indeed, this chapter is dedicated to some specific aspects of the management of the Subversion VC System environment where E4-VC will operate, which is external to it and whose knowledge is required or, simply, useful for its comprehension and usage.

VC System Console: Slik-Subversion

Having anyhow to start from somewhere, after an initial exploration of the Subversion URLs suggested by the official Eric web site, we decided to begin with what appeared to be the VC System console alternative less obscure and most explicitly compatible with our environmental conditions. That is the Slik-Subversion, as here detailed in Appendix, section “Subversion Packages: Download & Setup” [see].

This package, so described: “... a standalone command line Subversion package for Windows [...] contains all command line tools (svn, svnadmin, svnsync, svnserve, svnmucc, etc.)”, sounded good enough to download and try it, hoping for the best. No reason, so far, to regret that choice12.

Documentation of Reference

Documentation of reference here adopted for this Slik-Subversion VC System:

  Here on, for short, simply: “svn-Official Guide”

- Same book, for Subversion 1.7 (Compiled from r4230), in svn-book.pdf file edition, 2011, resulting available as part of the Slik-Subversion package setup [see].

Remark

As the scope of this Report is defined by the area of interaction between Eric and the VC System in use, for all aspects concerning the versioning technology in general and beyond such declared scope, the reference here adopted is that of the above listed books.

12 Of course we'll be more than happy to be seriously informed about other interesting alternatives...
Why the Slik-Subversion Package

So far, as said, we have no reason for regretting such choice. Therefore, here on, in this Report, we'll assume this Slik-Subversion package as the VC System console of reference, when not otherwise explicitly specified. Reason for this choice are:

- It is actually available on the Internet, easy to get and install.
- It is a complete package, offering both Client, Server, and administrative functions.
- It is fully compatible\(^{13}\) with the Windows system platform here in use.
- It is well documented.
- It enjoys a supportive community of users.
- It works.

For all practical reference to this package see Appendix, section: Subversion Packages: Download & Setup.

\(^{13}\) Actually it revealed compatible even with E4-VC, which is able to detect its presence and use it. Anyhow we do not consider this possibility as a valid alternative to the PySvn package here assumed for that role [see also section: Not an Alternative, in Appendix].

Creation of a Local svn-Repository

The creation of a local svn-Repository is rather straightforward. You just need to open a Command Prompt shell, accept the default directory and enter the command:

```
> svnadmin create svnRep
```

Where `svnRep` is the name here adopted for the svn-Repository to create.

Remark

As in the above case, all tools of this Subversion package are supposed to be used within a Command Prompt shell, which, to some extent, is a world apart where, for instance, such usual keyboard shortcuts as the `CTRL+C` and `CTRL+V`, to Copy/Paste text, aren't available.

\(^{14}\)
To this purpose it may be useful knowing that, under the icon of title bar, there is such a command menu:

with an **Edit** command offering some useful text handling functions, compatible with the surrounding Windows environment too.

And then, this the consequent result of the **svnadmin** creation. A standard directory structured as a brand new **svn-Repository**, ready to be used as such and also managed as any other standard Windows directory; i.e, for instance, deleted.
**svn-Command Test and Comparison**

A “standalone command line Subversion package”, as the Slik-Subversion is defined, comes handy also to manually test and investigate some of the commands automatically assembled by E4-VC, or to enter other specific Subversion commands, other than those available with it. Of course that would imply also a specific knowledge of the subject, beyond what is required to master the very E4-VC and as considered in this Report.

Here just an example of such a direct Slik-Subversion “svn list” command:

```shell
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\User111>svn list file:///C:/Users/User111/svnRep
branches/
tags/
trunk/

C:\Users\User111>svn list --verbose file://localhost/C:/Users/User111/svnRep
  2 User111  Apr 11 18:41 ./
  1 User111  Apr 11 18:33 branches/
  1 User111  Apr 11 18:33 tags/
  2 User111  Apr 11 18:41 trunk/

C:\Users\User111>svn list --verbose file://SPM101/_Share101/svnRepository
  0  ?  Apr 11 19:43 ./

- = -
```
3. Variations on a Theme

Central theme of this Report is the Version Control System possibly integrated into the Eric IDE, and here too, as with music, this is a theme that can have many, many variations. The primordial variation is that, being it not a precondition, Eric may have no VC System internally available at all.

Another possible basic variation is that, even with a VC System available, Eric can well be be used with a not-versioned project. Of course. All other possible variations will be hereafter described in this chapter, and then referred to—that is to say, not redundantly repeated—throughout this whole Report.

With a warning, though, which is a standard rule for the entire collection of the Eric Reports. That is that all operative and detailed descriptions are actually based on real experience of use. And also that when, as in this case, so many are the conceivable variations about such a theme, of all possible configurations only the actually tested ones will be thoroughly described, with all the remaining others left to the free and adventurous initiative of the reader.

Eric4 Referred Environment, and Initial Condition

What, on the contrary, cannot hereafter be subject to variation is the referred Eric4 environment, as it is precisely specified with table “Version Reference”, in the initial section “Essentials” [see].

The standard basic Eric4 setup includes a Subversion plug-in capable of detecting whether an actual Subversion package is there available, or not. When a VC System is actually available, at each new Eric Project definition the user will see such a dialog box:

![Project Properties dialog box](image)

Where the “Version Controlled Project” check-box will permit to decide whether to create a versioned new project, or not.

And when no VC System is available:

- No “Version Controlled Project” check-box results available at a new Eric Project inception.
3. Variations on a Theme

- Menu command Project > Version Control results “dimmed” and disabled [cf. section: 2.2/3]

- Menu command Settings > Preferences..., Version Control Systems > Subversion results anyhow available and active [see].

Then the other possibilities are:

- The VC System is actually available to Eric, but with no versioned project currently opened.
  
  - In that case you'll have the menu command Project > Version Control enabled and usable in its “reduced” version, as shown here above [see];
  
  - With the “Add to Repository...” command enabled only when an Eric unversioned project is currently open, otherwise dimmed-disabled [see].

- The VC System is actually available to Eric, and with a versioned project currently opened.
  
  - In that case you'll have the menu command Project > Version Control enabled in its “full fledged” version, as shown at: 2.2/3 Main Menu: Project > Version Control, the main section of this Report [see].
Which Data Protocol

As it can be seen on this drop-down list of the “Repository Infos” dialog box [see e.g. command: Project > New...]:

![Repository Infos dialog box]

the interchange of data with the E4-VC System can be done through these different protocols:

- **file://** With local disks
- **http://** With Apache server, via WebDAV
- **https://** Same as above, but with SSL encryption
- **svn://** With “svnserver”
- **svn+ssh://** Same as above, but through SSH tunnel

In this Report we’ll make use and describe the first three most commonly used protocols, with the related svn-Repository identified this way [see also chapter: 2. Basics of Subversion Management]:

- Protocol **file://**, for local disc and Local File System (LFS) “drive” path; e.g.: C:\Dir\SubDir\File.ext,
- Protocol **file://**, for local area network and Universal Naming Convention (UNC) “share” path; e.g.: \Computer\Share\Dir\File.ext,
- Protocol **https://**, for Internet and a svn-hosting provider such as: Springloops; e.g.: https://e4vc.slsapp.com/source/svnwebrep

The other last two protocols here listed, **svn://** and **svn+ssh://**, are known to be much less used and to be fit for low traffic svn-Repositories.
**svn-Repository in Local Disk**

Example of creation of a svn-Repository into a local disc:

```
> svnadmin create svnRep
```

Then identified with its usual local path:\(^{14}\): C:\User111\svnRep

for hosting a new versioned Eric Project.

---

\(^{14}\) Note that this is a file path, not an URL, as the label on the form mistakenly says. The resulting URL or, more precisely, the **svn**-URL to identify the **svn**-Repository is a suited composition of the protocol header with this file path, this way: `file:///C:/Users/User111/svnRep` or, equivalently, this other way: `file://C:/Users/User111/svnRep`
svn-Repository in Local Area Network

Example of creation of a svn-Repository in local area network:

> `svnadmin create \SPM101\_Share101\svnRepository`

Then identified with its UNC path: `\\SPM101\_Share101\svnRepository`

for hosting a new versioned Eric Project.
svn-Repository in Internet

Of course, if there is an environment where a Version Control System can result particularly useful that's the Internet. Of course. This Report, being concerned with E4-VC, will look at such a subject necessarily from the Client's point of view. In other words, the question to be here to be answered is:

Can E4-VC operate with a svn-Repository possibly available in Internet?

To offer a reliable answer to such a question we have looked for a suitable svn-Web Hosting provider, so to explore hands-on such possibility. It ended up a rather pleasant experience, characterized as here follows.

- Springloops (URL: [http://www.springloops.com](http://www.springloops.com)) as the test svn-Web Host of choice, which revealed efficient and supportive.

- An Eric versioned project easily created into such a web svn-Repository:

```plaintext
Path (relative to project): ...
Repository Information

Repository root URL: https://e4vc.slsapp.com/source/svnwebrepo/trunk
Repository UUID: 9342f9c7-3501-0010-bf53-5f1201c3a4d8
Last changed author: studio-pm@hotmail.com
Last Changed Date: 2012-04-17 15:46:41
Last changed revision: 1
Node kind: directory
Schedule: normal
```

With full and smooth on-line usability, after having accepted:
3. Variations on a Theme

Version Control

- And with such E4-VC options possibly set to remember the user's credentials [see: Version Control > Command Options...]:

- This here shown for illustration purpose only, as it would be a preferable, whenever possible, entering these credentials when so requested by the very svn server, in its own way, and possibly letting it to record them directly, instead of E4-VC.

For completeness' sake, then this same web svn-Repository has been satisfactorily explored also directly via a standard Subversion console command “list”; that is, out of E4-VC [see section: svn-Command Test and Comparison], this way:

```
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.
C:\Users\User111>svn list --verbose https://e4vc.slsapp.com/source/svnwebrep
Authentication realm: <https://e4vc.slsapp.com:443> Springloops Repository
```

15 Here we'd welcome description of your possibly different experiences.
Username: studio-pm@hotmail.com
Password for 'studio-pm@hotmail.com': ********

| 1 studio-p              Apr 17 15:46   | ./
| 1 studio-p              Apr 17 15:46   | branches/
| 1 studio-p              Apr 17 15:46   | tags/
| 1 studio-p              Apr 17 15:46   | trunk/

Viewpoint

<?> With this Springloops hosting service there is a problem, though. If you've got to go beyond such a basic test, you'll find no operative documentation available, so that you'll have no other choice but to proceed by try-and-error, or renounce. Not so an infrequent condition, alas.

Access Permissions to the svn-Repository

This of the “Access Permissions” to the svn-Repository in use is a subject that you, as a svn-Client, can only be aware of, and be informed about, not being supposed of enjoying svn-Administrator status. Of course that's a fundamental aspect of the versioning activity but, being E4-VC exclusively a svn-Client tool, all possible permission rules are under control of the Administration of the specific VC System in use, operating at the svn-Server side of the hill. And, therefore, as such, out of the scope of this Report.

Typical E4-VC Test Structure

Reading this Report it may come useful to know which is the typical structure of the resources here used for testing the E4-VC commands and features.

- ePrj - Generic Eric Project, whether versioned or not
- ...
  - \unvPrj - Local container of a un-versioned Project
  - \vPrj - Local container of a versioned Project
  - \svnWrkCpy - Local container of a Subversion Working Copy
  - \svnRep - Local Subversion Repository, typically located into the current working directory of the command shell
  - \webPrj - Local container of a Project versioned utilizing a web svn-Hosting Service
4. Eric4 Version Control – Menu Commands

Reference chapter for each and all E4-VC menu commands available for user interaction.

Menu Sections

The “Eric4 Version Control” functionality (short form: “E4-VC”) is not located into a unique Eric menu command as, for instance, the Web Browser functionality, which is located into the menu command Help > Helpviewer..., and which is even callable as a distinct “eric4_webbrowser” program. But it is, in some way, distributed among these distinct menu sections:

1/3 Menu Section: Project > New...
   Where a new versioned Eric Project can be created;

2/3 Menu Section: Project > Version Control
   Where the Eric Project versioning process is managed; subdivided into three distinct subsections:
   2.1/3 Initial Menu: Project > Version Control
      When no Eric Project versioning is yet under way;
   2.2/3 Main Menu: Project > Version Control
      When an Eric Project versioning is under way;
   2.3/3 Project-Viewer Context Menu: Version Control
      As above, as right-click context menu on the Eric Project-Viewer form pane;

3/3 Menu Section: Settings > Preferences..., Version Control Systems
   Where the Eric Project versioning preferences can be set up and tuned.

These menu sections will assume a different aspect, and functionality, if and when the Eric compatible Version Control System is actually installed and in use. The detailed and thorough presentation of all such functionality is the core purpose of this Report.
E4-VC as a Macro-Command Generator

E4-VC as a Macro-Command Generator

Note that most of the E4-VC menu commands are a sort of short-hand for the actual svn commands that will be executed, and whose full syntax, and resulting response, will be then anyhow explicitly shown on a subsequent E4-VC dialog box.

For instance, this single E4-VC command: Version Control > New from Repository...will originate such kind of svn command–response sequence [see]:

checkout file:///C:/Users/User111/svnRep/trunk C:\_Activity101\Eric\Eric4\_Lab\svnWrkCpy

Add C:\_Activity101\Eric\Eric4\_Lab\svnWrkCpy/__init__.py
Add C:\_Activity101\Eric\Eric4\_Lab\svnWrkCpy/vPrj.e4p
Add C:\_Activity101\Eric\Eric4\_Lab\svnWrkCpy/_eric4project
Update C:\_Activity101\Eric\Eric4\_Lab\svnWrkCpy
Common Dialog Boxes

Throughout the whole set of E4-VC commands the same parameter selection and dialog boxes can appear over and over, may times. So that it is convenient to collect and describe some of them hereafter in this one dedicated section.

VC System Selection
Where to select the desired VC System to be used, among the possibly available.

svn-Repository Location
It's composed by two distinct controls, whose values are to be then automatically composed to form the special svn-URL identifying a svn-Repository. Note that, at this svn-Client side, svn-Repositories are always identified by a special svn-URL\(^\text{16}\) [see section: Keywords and Key-Concepts].

The first control, labeled Protocol, is a drop-down list where to select the connecting svn-protocol [see also section: Which Data Protocol]:

---

16 Whereas at the svn-Server side they are seen as directories, identified with the standard paths of the file system in use.
The second control, labeled URL, is a field where to enter the related Repository locator:

With this “file://” Protocol case, two considerations hold:

- The “URL” label is rather inappropriate, even misleading. Indeed what you are expected to enter here is a standard directory path, not a URL. A URL or, more precisely, a special kind of svn-URL [see section: VC Keywords], is what then E4-VC is supposed to automatically compose, conveniently assembling these two field values as entered by the operator.

- This “subversionroot” directory name is a most unfortunate hard-coded default name, stubbornly appearing here in spite of any other different name you might be repeatedly using in your real case17.

With such other “http://” (or “https://”) Protocol case, what is expected is an actual web-Repository URL, but the heading Protocol code:

Optional Editing of svn --Options

With many E4-VC commands you have the possibility to edit the default svn “--Options”.

17 This (mild) unfriendly behavior has been justified by the Eric producer saying that the entering of such a value is an infrequent event. We do not agree.
Here answering “Yes”, you'll enter the same condition as you had called the command: Version Control > Command Options... [see].
1/3] **Menu Section: Project > New...**

Designed to create a brand new Eric Project, possibly “versioned” since the very beginning. That is: “Imported” into an existing svn-Repository and then also “Checked-out” as a svn-Working Copy, right away, as soon as created\(^\text{18}\), and into the very same original directory.

**Command sequence** [see section: E4-VC Common Dialog Boxes] – Steps:

- VC System Selection
- svn-Repository Location

Then these other svn “Repository Infos” will follow:

![Repository Infos Dialog Box]

Where:

- **Log Message** Attached message, mandatory with svn
- **Create standard repository layout** Check-box to consent the creation of the recommended svn-Repository standard layout structure, articulated into: Trunk, Tags, Branches [see section: Versioning Layout: ].

And here is how it appears the birth certificate of such a brand new Eric versioned project:

---

\(^{18}\) A usual Eric Project, born “unversioned”, can be “versioned” too, at any moment, via command: Version Control > Add to Repository... [see], but with a relevant difference. The original project remains unversioned, as the associated reduced version of the menu command Version Control reveals [see], with the corresponding full-fledged versioned copy which then can be then obtained as a standard svn-Working Copy.
Where you see the initial `svn` “import” action \[cf.: Version Control > Add to Repository...\], with the related message and the suggested standard Trunk, Tags and Branches structure. Then followed by the corresponding initial `svn` “checkout” action \[cf.: Version Control > New from Repository...\], so to complete the versioning process of this new Eric Project.

As a conclusion, here you'll have a new versioned Eric Project stored into the “.../svnRep/trunk” of a `svn`-Repository, with Revision No. 1, and available to be locally edited into the original directory “.../Lab\vPrj” as a `svn`-Working Copy.

Besides, as a consequence of the fact that this new Eric Project is a versioned one, you'll see the Project > Version Control menu expanded from its “reduced” version, as in the section Eric4 Referred Environment, and Initial Condition \[see\], to its full extension, as in the section 2.2/3/ Main Menu: Project > Version Control \[see\].
Remark

With the above procedure just one Eric Project can be versioned into a given svn-Repository, a possibly second one would lead to an error condition of the kind: “File already exists”. A possible technique for creating a multi-Project Repository has been considered—but not encouraged—at the section svn-Repository and Eric Projects [see: Versioning Layout].
2/3] **Menu Section: Project > Version Control**

Main menu section where the Eric Project versioning process is managed; organized into three distinct subsections:

2.1/3] **Initial Menu:** Project > Version Control
   When no Eric Project versioning is yet under way;

2.2/3] **Main Menu:** Project > Version Control
   When an Eric Project versioning is under way;

2.3/3] **Project-Viewer Context Menu:** Version Control
   As above, as right-click context menu on the Eric Project-Viewer form pane;
2.1/3] Initial Menu: **Project > Version Control**

This is the Version Control menu section available when no versioned project is currently active. It plays a role equivalent to that of the ordinary Project > Open... command, that is: to bring into operation an existing project, in this case a versioned project. Provided, of course, you can specify the related svn-Repository.

![Version Control Menu](image)

**Command List**

New from Repository...
Export from Repository...
Add to Repository...

---

Version Control > **New from Repository...**

Designed to Checkout—that is: download and store—the contents of an existing svn-Repository into a given directory, obtaining this way a svn-Working Copy, available to work on locally.

Command sequence [see section: E4-VC Common Dialog Boxes]:

- VC System Selection
- svn-Repository Location

Then this is the dialog box section to complete the call:
4. Eric4 Version Control – Menu Commands

Version Control

Where:

Tag
Optional selection of a particular Branch or Tag possibly defined on the svn-Repository [cf. commands: Version Control > Tag in Repository...]

Project Directory
Local directory where to store the svn-Working Copy

Repository has standard layout
Check-box to tell if the svn-Repository is organized into standard Trunk, Tags and Branches [see section: Versioning Layout:], or not. If you have no information about this fact, just un-check the box so to accept the contents, however organized.

Final command sequence [see section: E4-VC Common Dialog Boxes]:

- Optional Editing of svn --Options

At conclusion, such a dialog box will show you the log of the executed checkout:
Version Control > **Export from Repository...**

Designed to export—that is: download and store—the sheer contents of an existing `svn`-Repository into a given target directory, without retaining any `svn` versioning information. In other words, with this command the target directory will not be structured as a `svn`-Working Copy, and will receive the sheer `svn`-Repository data contents, unversioned.

Besides, this command sequence is identical to that of Version Control > New from Repository... [see].

Version Control > **Add to Repository...**

Designed to Import—that is: upload and store—an unversioned Eric Project, currently active, into an existing `svn`-Repository. This way versioning it. A command “dimmed” disabled when no Eric Project is active.

It's very similar to Project > New..., with which it shares an equal command sequence [see]. But, whereas this last may permit to versionize a new Eric Project when just born, its purpose is to versionize an Eric Project already existing, but not yet versioned. With a relevant difference, though:

<!-- ! --> Here the original Eric Project remains unversioned—as the associated reduced version of the menu command Version Control, which remains unchanged, will reveal [see]. Then a corresponding full-fledged versioned copy of the Project can be possibly obtained afterwords, as a standard `svn`-Working Copy via the usual Version Control > New from Repository... command [see].

**Warning**

<!-- ? --> The cited difference implies that this same command will keep remaining enabled and, therefore, usable more than once with the same Project, and even possibly towards the same `svn`-Repository\(^\text{19}\); exactly as with any other not-yet-versioned Project. That is to say that it is up to you to know if the target `svn`-Repository you're pointing at is suited to receive such an addition, or not.

**Remark**

<!-- ! --> This command has two other remarkably useful versions applicable to single files instead of, as here, to a full Eric Project. One as a command within the Project-Viewer Context Menu [see section: 2.3/3 Project-Viewer Context Menu: ], and one with the “Add” button inside the Subversion Status dialog box [see command: Show Status].

\(^{19}\) Anyhow, trying to over-Add the same Project into the same `svn`-Repository will be simply rejected, with an error of the kind: *File already exists: filesystem 'c:\users\user111\svnrepdummy\db', transaction '1-1', path '/trunk'*
2.2/3] **Main Menu: Project > Version Control**

Throughout this Report, and particularly in this section, only the technical descriptions specifically concerning E4-VC will be given, whereas those more specifically concerning the Version Control System in use are kept to a minimum, as given for known and granted and, anyhow, off the scope of this Report.

---

**Remark**

Comparing this menu with its sibling described at the corresponding next section no. 2.3/3, you'll see that while these commands here prevalently deal with the Eric Project considered as a whole, those there can be aimed at selected files or directories within the same Project.

---

20 A good **svn-Client commands reference** is offered by the **svn-Official Guide**: *Version Control with Subversion – For Subversion 1.6*, by Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato, ed. SoHoBooks, 2009
Command List

Subversion
Update from Repository
Commit Changes to Repository...
New from Repository... Export from Repository...
Tag in Repository...
List Tags... List Branches...
List Repository Contents...
Show Log Show Limited Log Show Log Browser
Show Status
Show Change Lists
Show Repository Info

Show Difference Show Difference (Extended) Show Difference (URLs)
Revert Changes
Merge Changes...
Conflicts Resolved
Switch...
Relocate...
Set Property...
List Properties...
Delete Property...
Cleanup
Execute Command...
Repository Browser...
Command Options...
Configure...

- -
Version Control > **Subversion**

Command designed to show this kind of plate information about the currently active E4-VC Project.

![Repository Information](image)

**Remark**

- Particularly relevant is the first item in this list, where you see which is the VC System in use with the current Eric Project. That is here: PySvn, as selected at project inception [see section: VC System Selection].

- Similar information, displayed on a form with this same “Repository Information” title bar, can be inspected via command: Version Control > Show Repository Info [see].

Version Control > **Update from Repository**

Designed to update project data from the svn-Repository into the currently open svn-Working Copy. Then, at command completion, such a dialog box:
could possibly remind you that the current Eric Project, having been updated, requires to be “re-read” by Eric, so to refresh and synchronize it for possible correct execution of the source text search and debug operations [about this same re–synchronization subject see also the main Eric Report, at the sections: Edit > Search, Menu Command: Debug and Menu Command: Project].

Conflicts

This one is the typical E4-VC action which can possibly detect a status of conflict between the svn-Repository and the current svn-Working Copy contents, as it can be inspected via Show Status command [see]. Conflicts arisen between contributions possibly entered independently by different operators and aimed at the same piece of data.

Solution of conflicts requires human intervention, and sound team work. Role of a VC System in this game can only be that of offering valid tools to support the process. Such as the E4-VC commands: Revert Changes, Merge Changes..., and Conflicts Resolved [see]. Comprehensive analysis and strategies for conflict solving with versioned projects is beyond the scope of this Report21.

Remark

Here in case of a “working copy locked” error, a Version Control > Cleanup command action can come handy [see].

21 About this subject you may see the svn-Official Guide, section Resolve Conflicts, chapter Basic Work Cycle.

Anyway our preferred strategy is this twofold one:

▫ Rely on sound team work;
▫ Rely on simple & sensible editing actions, beware of too sophisticated tools & strategies.
**Viewpoint**

<!> Not all conflicts possibly here detected are worth to be resolved. Some are pseudo-conflicts that, as with the command Commit Changes to Repository... [see], should be ignored as they are related to files of sheer local relevance, not to be bothered globally.

To that purpose the “global-ignores” option of the “Config” file is provided, where files that are to be kept out of the versioning mechanism—e.g. the directory’s: \_eric4project—can be conveniently listed [see section: Off Versioning Files, after Version Control > Show Status command].

---

**Version Control > Commit Changes to Repository...**

Designed to Commit—that is: upload and store—the project's file data changes, from your currently active svn-Working Copy into the original svn-Repository. This the control dialog box for such action:

![Subversion Commit Dialog](image)

Where:

Commit Message Text message to be associated to this Commit action; mandatory

Changelists To possibly apply this Commit action exclusively to the files belonging to the given Changelists [see command: Add to Changelist, at section: 2.3/3] Project-Viewer Context Menu: ]; optional
Keep Changelists

Check-box corresponding to the “--keep-changelists” svn-option, so to possibly have this same Changelists not erased at command completion.

File data changes are here to be intended in a very restrictive sense, in fact the possible addition of new data files cannot be handled with this command and requires the use of the specifically dedicated command Add to Repository... [see].

Whereas general changes existing between a svn-Working Copy and the referred svn-Repository, intended in all possible forms, addition of new files and directories included, can be inspected and managed with the command Show Status [see], where both “Commit” and “Add” action can be manually performed on selected list items. Along with other rather useful actions via button or context commands [see].

Remark

This command is the inverse of Update from Repository, and also here data conflicts can be detected, as described with section Conflicts of that command [see].

Version Control > New from Repository...

Version Control > Export from Repository...

Preliminary action of these two commands, as here available in this menu, is the closure of the versioned project currently active, so then to operate exactly as described in the former section 2.1/3 Initial Menu: [see], to which you're invited to refer.

Viewpoint

We didn't feel appropriate to duplicate here too the same description offered in the cited section 2.1/3 Initial Menu: because, here too, the fact that there is a versioned project currently active is totally irrelevant and ignored, not even used for the initial setup of default values such as the referred svn-Repository.
Version Control > **Tag in Repository...**

Designed to show this “Subversion Tag” dialog box:

![Subversion Tag dialog box](image)

where svn-Repository Tags and Branches instances of the project's Trunk can be managed, that is: created and deleted [see also section: Versioning Layout: ]. What in this form is labeled as “Regular Tag” and “Branch Tag”, in our svn-Official Guide [see section: Documentation of Reference] is simply called “Tag” and “Branch”, which is how we'll try to refer to them here on.

In case of a creation action, you'll then get this kind of dialog box, informing you exactly how this action executed.\(^{22}\)

![Subversion Tag dialog box with action log](image)

Where, in particular, you'll see that a Tag or a Branch is created, respectively, into the “/tags” or “/branches” virtual sub-directory of a svn-Repository necessarily hosting a project with a standard layout [see check box: Create standard repository layout of command: Project > New...].

\(^{22}\) That is, executing a standard svn “copy” command.
Subversion Tags are to be intended as a frozen “snapshots” of a given project, that is a stable product obtained from a Project at a given time, and at a given revision and version level. Therefore a Tag, as such, is not intended to be neither edited nor committed; to that purpose Branches are to be used instead. But anyway, if you want to edit and then Commit a Tag, you can. And then, at that very moment, it will simply stop being a proper Tag, and simply became a Branch. That's all.

Whereas the Subversion Branches are Product's instances still meant to be developed but, in some way, independently from the main Trunk; typically for specific purposes. Both Tags and Branches can be conveniently selected on the “Tag” field of the Version Control > New from Repository... command [see], so to derive externally the desired svn-Working Copy to work locally on.

Tags can also be directly changed into Branches, and vice versa, via command Version Control > Switch... [see].

**Remark**

The option buttons on this “Subversion Tag” dialog box:

![Subversion Tag Dialog Box](image)

result conveniently dimmed-disabled in case of a not-Standard svn-Repository layout, that is a custom Repository, not structured into Trunk-Tags-Branches [see section: Versioning Layout: ]. In this case this command mutates into a plain svn “copy”, anyhow less conveniently usable than the more appropriated Version Control > Copy /Move commands [see].
Version Control > **List Tags...**

**Version Control > List Branches...**

Commands available in case of an Eric Project versioned with a **svn**-Standard Layout, otherwise replaced by command: **List Repository Contents...** in case of a **svn**-Custom Layout [see].

They are designed to show the list of **svn**-Tags / Branches defined for the current Eric Project [see also section: Versioning Layout: and command: Tag in Repository...].

![Subversion Tag List](image)

Version Control > **List Repository Contents...**

Command available in case of an Eric Project versioned with a **svn**-Custom Layout, otherwise replaced by commands: **List Tags... / Branches...** in case of a **svn**-Standard Layout [see].

It is designed to list the contents of any **svn**-Repository, whether associated with the current Eric Project or not, and with a Standard Layout or not; provided you know and enter its URL [see also section: Versioning Layout: and command: Tag in Repository...]. Current **svn**-Repository's URL is offered as default.

![Subversion List](image)
Viewpoint

The expression “containing the tags or branches” in this caption has a historical justification, when this command was the only one capable of showing the root layout of a Repository. Now the Version Control > Repository Browser... command [see] is a much better way to perform such inspection.

Remark

Anyhow this command can equally display the contents of any Standard or Custom Layouts, or even of inner sections of a svn-Repository, provided you know the URL. Whereas the other corresponding List Tags / Branches... commands [see], available in case of a svn-Standard Layout, are designed to show exclusively the contents of the svn-Repository associate with the very Eric Project currently active.

Version Control > Show Log

Version Control > Show Limited Log

Version Control > Show Log Browser

Designed to show / or show limited in length\textsuperscript{23} / or show and browse, the current svn activity history log, on such a “Subversion Log” display form:

\begin{center}
\includegraphics[width=\textwidth]{subversion_log.png}
\end{center}

\textsuperscript{23} With the upper limit to the number of last entries to be shown that can be pre-set as Log parameter on the Settings > Preferences..., Version Control Systems > Subversion [see].
Where a “diff to 1” link is there to show the same “Subversion Diff” form as with command Version Control > Show Difference [see].

Or on such other “Subversion Log” display form, where to browse a possibly lengthy and crowded log:

![Subversion Log](image)

---

**Version Control > Show Status**

Command designed to perform a crucial task: that of inspecting and managing the synchronization between the *svn*-Working Copy you are working on and the corresponding contents of the central *svn*-Repository.

In fact, it is through this command that the operator can be informed about the actual updating status of each and all parts of the Project he is working on, compared with the central *svn*-Repository. So then to possibly take the appropriate updating actions.\(^\text{24}\)

\(^\text{24}\) If you naively believed [as we originally did] that the Commit Changes to Repository... command
That said with particular reference to the items declared of Status \textit{unversioned}. That is, files actually belonging to the Eric Project, but not yet added to the \texttt{svn}-Repository. Selecting any such an “\textit{unversioned}” item, you’ll then see the \texttt{Add} button conveniently enabled, and ready to be used.

\textbf{Remark}

Note that setting the “\texttt{--show-updates}” option for the \texttt{svn}-Status command \cite{Version Control > Command Options...} the status of the remote repository will be checked as well, and that will let you know if an update of the \texttt{svn}-Working Copy is necessary.

---

\textbf{Status LED Indicators}

Besides the “Subversion Status” dialog box, as hereafter described, E4-VC status information is also quickly monitored by these Status LED Indicators:

\textit{Status Monitor LED}

Main Eric4 window, bottom right. Color coded as for \texttt{Shift-F1} \texttt{Help Hint} \cite{see}.

\textit{VCS Status LED}

Project-Viewer tab bar, top-left. Color coded as for \texttt{Configure Version Control Systems} \cite{see}.

---

\cite{see} is all you needed to the cited purpose, forget it: it is not so. What you need is to inspect your Project with this command, and then take the desired action for each selected item possibly listed on this “Subversion Status” dialog box. Similar actions can also be performed using specific \texttt{Project-Viewer} context commands \cite{section: 2.3/3 Project-Viewer Context Menu: Version Control}.
Here an example of such “Subversion Status” form, where button:

Commit is for files locally modified [cf. command: Commit Changes to Repository...]
Add is for files present locally, but not in the svn-Repository [cf. command: Add to Repository..., particularly within the Project-Viewer context menu]
Differences is to inspect the differences between local and remote files [cf. command: Show Difference]
Revert is to undo last un-Commited changes, as with command Revert Changes [see]
Restore is to restore from the svn-Repository items possibly found in “missing” Status, that is: locally not available

Whereas a possibly empty Subversion Status form, with all buttons disabled, is to tell you that the svn-Repository and svn-Working Copy contents are perfectly synchronized.
“Show Status” Form Used to “Add” Sub-directories

<!> Operating on a Show Status form you have the possibility to easily add new sub-directories to a svn-Repository, with all their contents. To do that, what you need is just to create a sub-directory inside a versioned Eric Project, then open this Show Status form, where you'll see it detected and declared “unversioned”. Then, selecting it, the Add button will let you get it versioned, along with all the rest of the project.

Functions such as this “Add” can become here available also on a pop-up context menu, right-clicking a Status line. That reveals particularly useful for Add-ing a single unversioned file inside an already versioned sub-directory.

Off Versioning Files

It is here worth recalling that not all files of a versioned Eric Project are to be added and committed into a central svn-Repository. Some files, such as the automatically created “_eric4project”, are of exclusive local or temporary relevance, and therefore should be ignored by the global versioning process.

In case you'll see some of them still listed in this “Subversion Status” box, you'd better check and edit the “global-ignores” option of the “Config” file, so to have them conveniently ignored [see: PySvn, Setup Completion; in Appendix].
Version Control > **Show Change Lists**

Designed to show a form displaying all the Changelists currently defined.

![Subversion Change Lists](image)

Role and management of the Commit Changelists are described with the related commands: Add to Changelist, Remove from Changelist, and Commit Changes to Repository... [see].

Note that also the Show Status form offers a Changelist column, along with a related handy context menu [see command: Version Control > Show Status].

Version Control > **Show Repository Info**

Designed to show a whole set of useful “Repository Information” on this dialog box:
Even more detailed information about this same subject can be inspected with command: Repository Browser...[see].

**Remark**
- Particularly relevant is the first items in this list, where you see which is the exact location of the `svn`-Repository and `svn`-Master Copy in use.
- Similar information, displayed on a form with this same “Repository Information” title bar, can be inspected via command: Version Control > Subversion [see].

Version Control > **Show Difference**

Version Control > **Show Difference (Extended)**

Version Control > **Show Difference (URLs)**

Designed to show in the:
- “**Diff**” file mode, or “Extended” mode, or “URLs” mode [see next sections] the differences possibly intervened between the original `svn`-Revision and the current `svn`-Working Copy.
“Diff” File Mode

Here the differences between the original and the modified projects, as listed on the two heading lines (marked respectively “---” and “+++”), will be shown on such “Subversion Diff” form:

In case of no change, a “There is no difference” message will be displayed instead.

The changes are here encoded in “unified diff format”\(^{25}\) (or “unidiff”) XML text, conveniently available to be copied into the clipboard and also saved via the Save button [see] as a “*.diff” Patch File. That is, a file somewhat compatible with the Unix standard patch program\(^{26}\), and fully compatible with the svn patch subcommand [see: svn-Official Guide].

\(^{25}\) Actually it’s a special svn-flavor, whose detailed description is off scope of this Report.
\(^{26}\) Ported also to Windows, care of GnuWin32 and UnxUtils.
“Extended” Mode

Extended version of the same Show Difference base command [see], allowing the selection of which the svn-Revisions to compare, on such a control form:

Where Revisions can be specified through their identifying Number, Date, or through their standard keywords, as hereafter listed:

- **HEAD**: Latest—i.e.: youngest—in svn-Repository
- **WORKING**: svn-Working Copy’s
- **BASE**: Original revision number of an item in the svn-Working Copy
- **COMMITTED**: Most recent in which an item changed, with respect to the BASE revision (included)
- **PREV**: Immediately before the last in which an item changed
“URLs” Mode

Distinct version of the same Show Difference base command [see], allowing the selection of which the revisions to compare within the same svn-Repository, by choosing on this control form which the svn Trunk, or Tag, or Branch to compare:

The “Summary only” check-box is to possibly obtain just a brief summary list.

Version Control > Revert Changes

Designed to undo all local edit actions performed on the current svn-Working Copy after last Commit action, typically so to then resolve “manually” any possible conflicting changes.

Here then, if intervened any actual change\(^\text{27}\), the user may be automatically required to “re-read”—i.e.: refresh—the project [cf. command: Update from Repository], so to update the actual resulting condition of the Eric Project.

Version Control > Merge Changes...

Designed to get and merge the differences between two svn-Data Sources into a single svn-Working Copy. Typically used to utilize material developed in distinct Branches of a Trunk back into a Branch, or Trunk.

This the related control form:

\(^{27}\) Detected on the “.e4p” Eric file.
Where (according to the \texttt{Shift-F1} \texttt{Help Hint}[\textit{see}]):

1./2. \texttt{URL/Revision}

\texttt{svn}-Repository URL or Revision number to be merged into the target \texttt{svn}-Working Copy

\texttt{Target}

Target \texttt{svn}-Working Copy for the merge operation, a field to be left empty to tell the current one. Needed only for Revision Number merging sources, instead of URL.

\texttt{Enforce merge}

Check-box to force the merge operation

See here an execution example:

\texttt{Subversion Merge}

\texttt{Subversion}

\texttt{Conflicts Resolved}

A command merely aimed at removing the “conflicted” state on \texttt{svn}-Working Copy files or directories, possibly detected at an update action and assumed as already conveniently fixed [\textit{see command: Update from Repository}].
In other words, a command is designed to permit a subsequent Commit action after the user have manually resolved a conflict condition.

This command, in particular, will remove the conflict record files: *.mine, *.rn1, *.rn2, and switch the Status of a file from “conflict” to “modified” [see reported Subversion Status example figures].

28 The real action here performed [cf. section: E4-VC as a Macro-Command Generator] is actually a call to a PySvn API, only conventionally named after the very “resolved” svn-command, which, in the svn Official Guide, is declared as “deprecated in favor of svn resolve --accept”[see].
4. Eric4 Version Control – Menu Commands

Version Control > **Switch...**

Designed to show a dialog box where to possibly convert Tags into Branches, and vice versa [see command: Version Control > Tag in Repository...].

Version Control > **Relocate...**

In case the referred svn-Repository has been assigned to a different location\(^\text{29}\), or a new layout, this

\(^{29}\) Typically: svn-Repository moved by the system administrator to another server.
command is aimed at updating the current svn-Working Copy so to follow the new reference.

With this control dialog box, where:

New repository URL
New svn-URL, initially defaulted to the “old” value. Note that no search button can be here offered, as such info must necessarily come from the svn-Administrator.

Relocate inside repository
Check-box where to possibly tell that it is the svn-Repository layout that has been changed, not its URL.

Version Control > Set Property...
Designed to show a dialog box where svn Property\textsuperscript{30} metadata items can be defined [\textit{cf. commands}: List Properties..., Delete Property...].

Where:
Name
Usual syntax of a XML \texttt{Name}, but the restriction: “svn:” prefix, reserved for svn

\textsuperscript{30} Rather powerful svn feature, see Properties section at svn-Official Guide.
4. Eric4 Version Control – Menu Commands

Version Control

standard properties

<table>
<thead>
<tr>
<th>Value</th>
<th>Possible value to associate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply recursively</td>
<td>Through the entire repository tree structure</td>
</tr>
</tbody>
</table>

Version Control > List Properties...

Designed to show the list of the currently defined custom svn-Properties [cf. commands: Set Property..., Delete Property...].

When no svn-Property results defined, a “None” caption will appear under the Name heading.

Version Control > Delete Property...

Designed to show a dialog box where svn Property items can be deleted [cf. commands: Set Property..., List Properties...].
Version Control > **Cleanup**

Designed to execute a standard Subversion “cleanup” action onto the current svn-Working Copy. Typically useful for removing possible stale locks, denounced by a “working copy locked” error due to unfinished operations, preventing the execution of a Version Control > Update from Repository command action [see].

Version Control > **Execute Command...**

Designed to show this dialog box, where a Subversion svn Client command can be directly entered as onto a standard svn command shell, but for the “svn” prefix, here not required.

It may come handy to enter special Subversion commands, possibly not available as standard E4-VC commands [see also: Version Control > Command Options..., Remark and Viewpoint included].

Version Control > **Repository Browser...**

Designed to show this form, aimed at browsing the svn-Repository specs, structure and contents.
The initial default URL points to the versioned Project currently active, but then any other valid svn-Repository URL can be here entered. Similar information can be also inspected with the command: Show Repository Info [see].

Version Control > Command Options...

Designed to show such an impressive “VCS Command Options” dialog box:
Where to possibly preset, that is: enter and store to the benefit of subsequent E4-VC commands, custom values for standard svn “--Options”, collected into these categories [see also the associated Shift-F1 Help Hint):

Global

Global svn options valid in general, such as:

--username myUserName --password myPassword

>Status

Specific option for this svn command, typically: --show-updates
so to display info about which files in your svn-Working Copy are out of date.

Remark

To make an effective use of this customization feature you must, of course, be acquainted with the very Subversion subject; which that leads to the following two rather relevant notes:

• It's off the scope of this Report to treat extensively this matter. Just Global and Status notable examples are here offered [see].

• E4-VC, in essence, is a macro-builder of “bare” Subversion commands, assembled ready available to the benefit of Eric users. Sort of high-level language aimed at unleashing the power of a VC System to the Eric users, preventing them from the burden of learning in detail this other low-level control command language.

     Of course sometimes, in special cases, it can be anyhow useful, or even necessary, to be able to go to a lower svn-level, as here considered—or as with the command Version Control > Execute Command... [see].—. Provided such a possibility were not intended as a justification for any possible E4-VC weakness, or inefficiency; given the fact that, this way, you have anyhow the freedom of entering directly the exact svn command you want.

Viewpoint

<??> The former point could also be expressed this other way:

• A valid measure of the usefulness of this whole E4-VC tool, as opposed to the direct use of a VC System, independently from Eric, could be given by how much you are forced to apply directly to a standard VC System to get results that you cannot get through the integrated E4-VC31.


---

31 Rather serious matter, about which we'd love to know the frank opinion of the Eric users community...
Version Control > **Configure...**

Designed to show the Eric Preferences control form [cf. menu command: Settings > Preferences...], conveniently positioned onto the Subversion sub-section of the Version Control Systems section. Subject described somewhat more in general in the next section 3/3/ Menu Section: Settings > Preferences..., Version Control Systems [see].
2.3/3] Project-Viewer Context Menu: Version Control

This context menu, available within the Eric Project-Viewer form pane, is a variant of the same one treated in the preceding section 2.2/3] [see], and is available via a right-click on precise items—that is: files, directories—or on the empty zone in that area.

Different commands only will be here treated. Descriptions already offered at section 2.2/3] Main Menu: Project > Version Control [see] will be not duplicated.

Remark
<\!> Comparing this menu with its sibling described at the corresponding previous section no. 2.2/3] [see] you'll see that, while those commands there prevalently deal with an Eric Project considered as a whole, these here can be aimed at distinct files, or directories, selected within the same Project.

Command List

Remove from Repository (and Disk)  Move
Copy  Move
Add to Changelist  Remove from Changelist
Show Annotated File
Lock  Unlock
Steal Lock  Break Lock
Select All Local File Entries  Select All Versioned File Entries
Select All Local Directory Entries  Select All Versioned Directory Entries

S-PM 120700
Version Control > **Remove from Repository (and Disk)**

Designed to remove selected items both from the current `svn`-Working Copy and the related `svn`-Repository.

![](remove-from-repository.png)

It's an action that requires confirmation.

Version Control > **Copy**

Version Control > **Move**

Designed to copy or move selected files on the current `svn`-Working Copy of a project.

![](subversion-copy.png)

Where:

- **Source**: Selected item to be Copied / Moved
- **Target**: Destination path, relative to current `Source` or absolute to a known `svn`-Working Copy
- **Enforce operation**: Check-box to set the `svn` “--force” condition. Enabled for “Move” command only [see].


Remark

Such copy and move operations can well be done also the usual way, without a recourse to these specific svn-commands. Reason for doing the svn-way is to have it versioned, that is recorded in all details: source, destination, author and date. And examinable at any future time, as any other versioned actions.

Version Control > Add to Changelist

Version Control > Remove from Changelist

Designed to possibly delimit the scope of a Commit action to some selected files only, instead of the entire Eric Project. With the target “Changelist” here created, if not already existent.

These same Add and Remove commands come available also on the context menu of the Show Status command form, where there is a handy Changelist column too [see].

The so defined set of “Changelists” can be inspected with the commands Show Change Lists and Show Status [see], and are intended to be used with the command Commit Changes to Repository... [see].

Remark

<?> svn-Changelist mechanism still suffers from some mild limitations that, once known, can be easily tolerated:

- Adding twice a given file or Removing a file never added to any Changelist are quietly accepted as no-operations.
- Any given file can be added to one Changelist only. So that adding a file to a Changelist will first imply the canceling of any other possible former addition.
- Remove action doesn't require the name of the target Changelist, and can be repeated even if dummy.
• Each defined Changelist will vanish after having been used—i.e.: Committed—just once, unless a “--keep-changelists” svn-option is set32.

Version Control > **Show Annotated File**33

Designed to show all svn-relevant data about a selected source file. That is: Revision No., Author and the contents; line by line.

Example:

![Subversion Blame](image)

Version Control > **Lock**

Version Control > **Unlock**

Even though Subversion uses a so called “Copy-Modify-Merge” operating model, and not a “Lock-Modify-Unlock” one, also with Subversion there are situations in which locking a file is the right thing to do34. And these two commands, along with subsequent Steal / Break Lock [see], are the ones designed to do the job.

---

32 For further reference about this subject see: svn-Official Guide, chapter 3. Advanced Topics, section Changelists.

33 If curious to know where such “Annotated” and “Blame” terms come from, you might have a look at the “svn blame (praise, annotate, ann)” command, on the svn-Official Guide.

34 For further reference about this subject see: svn-Official Guide, chapter 1. Fundamental Concepts.
A job which here implies also such a “Subversion Lock” dialog box:

Where to enter a “lock comment”, mandatory.

Version Control > **Steal Lock**
Version Control > **Break Lock**

Subversion-style file locking feature is anyhow not so strict, though. It works more as a warning flag than a real binding condition. In fact, besides the two previous “plain” Lock / Unlock commands [see], there are also these other special two, which do exactly the same job just, simply, with an added `svn --force` option.

Commands that are available to any user, besides the original “owner” of any given lock. So that, this way, anyone can take possession of—that is: “Steal”—or “Break” somebody else's lock.

Version Control > **Select All Local File Entries**
Version Control > **Select All Versioned File Entries**
Version Control > **Select All Local Directory Entries**
Version Control > **Select All Versioned Directory Entries**

Set of Project-Viewer context menu commands available also with no item selected on that form, and designed precisely to select all items matching the named condition. That is:

Local / Versioned

To select all items listed on the current svn-Working Copy and not-versioned /

---

35 For further reference about this subject see: **svn-Official Guide**, chapter 3. Advanced Topics, section Locking.
versioned into the related svn-Repository.

Items here selected as “Local” are the same declared “unversioned” by the Show Status command, on the Status column of the Subversion Status form [see].

Possible void result, with message: “There were no matching entries found.”
Menu Section: **Settings > Preferences...**

Command aimed at managing the Eric Preferences referring to the “Version Control Systems” section. It represents an alternative way to reach the same controls shown by the command **Version Control > Configure...** [see], with the difference that this command here is always available, and not only with a versioned project opened.

**Remark**

- Controls of this section are assumed as self-explanatory, and clear enough not to require any further description.

- Standard Subversion “Config” and “Servers” configuration files can be here conveniently inspected and edited. In particular with respect to the “global-ignores” Config parameter, as suggested at section: **Setup Completion, PySvn;** in Appendix [see]. Detailed technical description for these two files can be found at: [svn-Official Guide, chapter 7. Customizing Your Subversion Experience](#).
Appendix

Topics here treated should be considered as a specialized addition to the corresponding ones treated in the central volume of this collection, the “Eric4 – Technical Report” [see], which remain valid in general and are not going to be duplicated here.

---

Typographical Conventions

Prevalent typographical conventions here adopted to increase readability.

Capitalized

Usual font, but with the initial capitalized, to specify a term to be intended in a specific technical meaning; such as: Working Copy, Tag, Branch.

Courier New

Standard technical elements and references, such as:

- **URL**: http://sourceforge.net/projects/
- **File path**: C:\Program Files\...
- **Menu command**: Settings > Preferences...

*Italics*

Segment within a Courier New string, referring to a replaceable/variable item, such as a generic *filename.ext* into a standard path: C:\Dir\Subdir\fileneme.ext

---

Subversion Packages: Download & Setup

Detailed reference information about the Subversion packages used in this Report; that is: PySvn and Slik-Subversion.

**PySvn**

“PySvn” is the Subversion API library used to integrate this VC System into Eric4 [cf. section: Which VC System: ].

**Essentials**

<table>
<thead>
<tr>
<th><strong>Web Site</strong></th>
<th><a href="http://pysvn.tigris.org/">http://pysvn.tigris.org/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item</strong></td>
<td>py27-pysvn-svn173-1.7.6-1457.exe (or latest)</td>
</tr>
<tr>
<td><strong>Setup</strong></td>
<td>As Administrator, into C:\Python27\Lib\site-packages\pysvn; consequent Start Menu: “PySVN for Python 2.7”, pointing at the items:</td>
</tr>
</tbody>
</table>

36 Besides **PySvn** there you’ll also find such a “WorkBench”, presumably a GUI tool made up using PySvn. Discouraging first impression.


Setup Completion

The E4-VC environment generated by the setup action as here described is rich enough to deserve—or even require—some subsequent setup completion and adjustment to be performed via the dedicated menu command Version Control > Configure... [see]. Particularly relevant is here the preventive definition of which are the files of an Eric versioned Project that should be ignored by the versioning process. In general terms such files are those of exclusive local or temporary scope, and whose storage into a central svn-Repository would be less than useful, a actual damage. To that purpose Subversion offers the “global-ignores” option within the standard “Config” file\(^\text{37}\), here available to be inspected and edited via the “Edit config file” on the “Configure Subversion Interface” [see].

What you see here is the initial configuration of such “global-ignores” section, as automatically generated by E4-VC. It's wise to give it a look, and then possibly tune it to your specific needs.

Uninstall

On Windows Control Panel, pointing at program: “Python 27 PySVN 1.7.6-1457”.

---

\(^{37}\) See also the Documentation of Reference section.
Remark

As an Application Programming Interfaces (APIs) library, PySvn is certainly well suited to be integrated into a program such as the Eric IDE, but, obviously, cannot be used interactively, “by hand”. And this implies that for all our svn management needs, external to Eric, such as the creation of local svn-Repositories and the test and comparison of specific svn-Client actions, PySvn cannot replace an ordinary Subversion command package such as SlikSvn [see next section].

Slik-Subversion

“Slik-Subversion” is the VC System console here used for all required Subversion functions to be executed independently from Eric. [cf. section: Which VC System: ].

Essentials

Web Site http://subversion.apache.org/packages.html
Item Slik-Subversion-1.7.2-win32.msi (or latest), 32 bit edition, for Windows 2000/XP/2003/Vista/7
Setup As Administrator, into C:\Program Files\SlikSvn\, “Complete” (not “Typical”); consequent Start Menu: “Subversion”\(^\text{38}\), pointing at the items:
  - Release Notes
  - Subversion Changes
  - Version Control with Subversion, (svn-book.pdf)
Use Within the ...\SlikSvn\ setup directory you'll find the programs:
  - svn.exe Client
  - svnadmin.exe Administrator/Server
Which, as all other programs of this package, are executable within the DOS command shell, with the usual default initial condition as shown by the prompt:
  - C:Users\User111>

Uninstall On Windows Control Panel, pointing at program: “Slik Subversion 1.7.2 (x86)”.

\(^{38}\) No operative calls there available, info only.
Not an Alternative

After having possibly installed a standard Subversion command console package, such as the Slik-Subversion [see chapter: 2. Basics of Subversion Management], Eric will realize its presence, and offer it too as a possible versioning alternative to PySvn. This way:

After having initially used and compared both alternatives, we've decided in favor of a single one, the PySvn's, which revealed more reliable, and which also permitted us to keep this Report more uniform and manageable. Anyhow, if you do not share our choice and intend to install and utilize a different Subversion package, we presume this Report will anyhow retain most of its validity and usefulness.

---

39 And we'd welcome news about your experience.
**Viewpoint – In conclusion**

_In conclusion, there is a specific aspect of the subject treated in this Report that we feel as fascinating: the fact that it’s about a tool designed to permit different people to collaborate to a common task. People possibly belonging to different regions of the world, to different time zones and habits, made technically capable of harmoniously contributing to a common project. Fascinating._

Of course this doesn’t mean that a VC System couldn’t be utilized in a single-user environment, for its specific storaging and versioning functions, rather than for its collaborative capabilities. It’s simply that we value the collaborative utilization as particularly innovative and powerful.

Well, being ourselves involved in a couple of collaborative projects based upon different VC Systems, we matured two convictions about this subject, one formal and one practical. Convictions hat we’d like to share hereafter.

The formal conviction has to do with names and definitions.

_This same tool, besides supporting the collaborative development of certain projects, is also capable of recording the versioning history of the same projects. Such collateral feature, as relevant as it might be, seems to us almost negligible, by comparison. Therefore we find rather inappropriate, if not even misleading, to refer to such a tool as a “Version Control System” instead of, say, a “Collaborative Production System”._

The practical conviction has to do with technological barriers.

_We are convinced that we are already in need of overcoming the barriers currently existing between different and incompatible VC Systems. Delighted to know whether someone of you is already working on that. Are you?_

– P.M.
Table of Contents

Foreword..............................................................................................................................3
Essentials....................................................................................................................................4
   True Multi-Platform...........................................................................................................4
1. Introduction to the Eric4 VC System..................................................................................5
   Which VC System: Subversion, via PySvn...........................................................................6
   Key-Concepts and Keywords...............................................................................................7
   Versioning Layout:
      Repository, Project, Trunk, Branches, Tags; and Working Copies..............................10
2. Basics of Subversion Management..................................................................................13
3. Variations on a Theme.......................................................................................................17
   Eric4 Referred Environment, and Initial Condition............................................................17
   Which Data Protocol..........................................................................................................19
   Access Permissions to the svn-Repository..........................................................................24
   Typical E4-VC Test Structure............................................................................................24
4. Eric4 Version Control – Menu Commands......................................................................25
   Menu Sections.....................................................................................................................25
   Common Dialog Boxes......................................................................................................27
   1/3 Menu Section: Project > New... .................................................................................30
   2/3 Menu Section: Project > Version Control..................................................................33
   2.1/3 Initial Menu: Project > Version Control..................................................................34
      Version Control > New from Repository......................................................................34
      Version Control > Export from Repository...................................................................36
      Version Control > Add to Repository...........................................................................36
   2.2/3 Main Menu: Project > Version Control....................................................................37
      Version Control > Subversion.......................................................................................39
      Version Control > Update from Repository..................................................................39
      Version Control > Commit Changes to Repository.......................................................41
      Version Control > New from Repository......................................................................42
      Version Control > Export from Repository...................................................................42
      Version Control > Tag in Repository............................................................................43
      Version Control > List Tags..........................................................................................45
      Version Control > List Branches..................................................................................45
      Version Control > List Repository Contents..................................................................45
      Version Control > Show Log........................................................................................46
      Version Control > Show Limited Log...........................................................................46
      Version Control > Show Log Browser..........................................................................46
      Version Control > Show Status....................................................................................47
      “Show Status” Form Used to “Add” Sub-directories.......................................................50
      Off Versioning Files......................................................................................................50
      Version Control > Show Change Lists..........................................................................51
      Version Control > Show Repository Info.......................................................................51
Version Control > Show Difference............................................................................................................52
Version Control > Show Difference (Extended)....................................................................................52
Version Control > Show Difference (URLs)...............................................................................................52
Version Control > Revert Changes.............................................................................................................55
Version Control > Merge Changes.............................................................................................................55
Version Control > Conflicts Resolved........................................................................................................56
Version Control > Switch...........................................................................................................................58
Version Control > Relocate..........................................................................................................................58
Version Control > Set Property..................................................................................................................59
Version Control > List Properties............................................................................................................60
Version Control > Delete Property............................................................................................................60
Version Control > Cleanup..........................................................................................................................61
Version Control > Execute Command.......................................................................................................61
Version Control > Repository Browser....................................................................................................61
Version Control > Command Options.......................................................................................................62
Version Control > Configure......................................................................................................................64
Version Control > Remove from Repository (and Disk).............................................................................66
Version Control > Copy...............................................................................................................................66
Version Control > Move...............................................................................................................................66
Version Control > Add to Changelist............................................................................................................67
Version Control > Remove from Changelist...............................................................................................67
Version Control > Show Annotated File....................................................................................................68
Version Control > Lock.................................................................................................................................68
Version Control > Unlock............................................................................................................................68
Version Control > Steal Lock.........................................................................................................................69
Version Control > Break Lock......................................................................................................................69
Version Control > Select All Local File Entries.......................................................................................69
Version Control > Select All Versioned File Entries.................................................................................69
Version Control > Select All Local Directory Entries..............................................................................69
Version Control > Select All Versioned Directory Entries.....................................................................69

2.3/3 Project-Viewer Context Menu: Version Control..............................................................................65
Version Control > Remove from Repository (and Disk).............................................................................66
Version Control > Copy...............................................................................................................................66
Version Control > Move...............................................................................................................................66
Version Control > Add to Changelist............................................................................................................67
Version Control > Remove from Changelist...............................................................................................67
Version Control > Show Annotated File....................................................................................................68
Version Control > Lock.................................................................................................................................68
Version Control > Unlock............................................................................................................................68
Version Control > Steal Lock.........................................................................................................................69
Version Control > Break Lock......................................................................................................................69
Version Control > Select All Local File Entries.......................................................................................69
Version Control > Select All Versioned File Entries.................................................................................69
Version Control > Select All Local Directory Entries..............................................................................69
Version Control > Select All Versioned Directory Entries.....................................................................69

3/3 Menu Section: Settings > Preferences...
Version Control Systems..........................................................................................................................71

Appendix...........................................................................................................................................................72
Typographical Conventions..........................................................................................................................72
Subversion Packages: Download & Setup......................................................................................................72
PySvn...............................................................................................................................................................72
Slik-Subversion...............................................................................................................................................74
Not an Alternative...........................................................................................................................................75
E4-VC Command Index..................................................................................................................................79
E4-VC Command Index

All E4-VC commands treated in this Report are here conveniently listed in alphabetic order, so to offer another useful navigating tool besides the standard Table of Contents [see].

<table>
<thead>
<tr>
<th>Command</th>
<th>Section</th>
<th>Pg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project &gt; New...</td>
<td></td>
<td>30</td>
</tr>
</tbody>
</table>

Initial

<table>
<thead>
<tr>
<th>Command</th>
<th>Section</th>
<th>Pg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project &gt; Version Control &gt; Add to Repository...</td>
<td></td>
<td>36</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Export from Repository...</td>
<td></td>
<td>36</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; New from Repository...</td>
<td></td>
<td>34</td>
</tr>
</tbody>
</table>

Main

<table>
<thead>
<tr>
<th>Command</th>
<th>Section</th>
<th>Pg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project &gt; Version Control &gt; Cleanup</td>
<td></td>
<td>61</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Command Options...</td>
<td></td>
<td>62</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Commit Changes to Repository...</td>
<td></td>
<td>41</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Configure...</td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Conflicts Resolved</td>
<td></td>
<td>56</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Delete Property...</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Execute Command...</td>
<td></td>
<td>61</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Export from Repository...</td>
<td></td>
<td>42</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; List Branches...</td>
<td></td>
<td>45</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; List Properties...</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; List Repository Contents...</td>
<td></td>
<td>45</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; List Tags...</td>
<td></td>
<td>45</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Merge Tags...</td>
<td></td>
<td>55</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; New from Repository...</td>
<td></td>
<td>42</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Relocate...</td>
<td></td>
<td>58</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Repository Browser...</td>
<td></td>
<td>61</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Revert Changes</td>
<td></td>
<td>55</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Set Property...</td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Show Change Lists</td>
<td></td>
<td>51</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Show Difference</td>
<td></td>
<td>52</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Show Difference (Extended)</td>
<td></td>
<td>52</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Show Difference (URLs)</td>
<td></td>
<td>52</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Show Limited Log</td>
<td></td>
<td>46</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Show Log</td>
<td></td>
<td>46</td>
</tr>
<tr>
<td>Project &gt; Version Control &gt; Show Log Browser</td>
<td></td>
<td>46</td>
</tr>
</tbody>
</table>
Project > Version Control > Show Repository Info ................................................................. 51
Project > Version Control > Show Status .............................................................................. 47
Project > Version Control > Subversion ................................................................................ 39
Project > Version Control > Switch ..................................................................................... 58
Project > Version Control > Tag in Repository ...................................................................... 43
Project > Version Control > Update from Repository ............................................................ 39

Version Control > Add to Changelist .................................................................................... 67
Version Control > Break Lock .................................................................................................. 69
Version Control > Copy ........................................................................................................... 66
Version Control > Lock ............................................................................................................ 68
Version Control > Move .......................................................................................................... 66
Version Control > Remove from Changelist .......................................................................... 67
Version Control > Remove from Repository (and Disk) .......................................................... 66
Version Control > Select All Local Directory Entries ............................................................ 69
Version Control > Select All Local File Entries .................................................................... 69
Version Control > Select All Versioned Directory Entries ..................................................... 69
Version Control > Select All Versioned File Entries .............................................................. 69
Version Control > Show Annotated File .................................................................................. 68
Version Control > Steal Lock ................................................................................................... 69
Version Control > Unlock ......................................................................................................... 68

Settings >
Settings > Preferences ........................................................................................................... 71