European Organisation for Astronomical Research in the Southern Hemisphere

Programme: E-ELT

Project/WP: E-ELT Telescope Control

ELT SVN Usage Guidelines

Document Number: ESO-283837

Document Version: 1

Document Type: Procedure (PRO)

Released On: 2017-07-11

Document Classification: ESO Internal [Confidential for Non-ESO Staff]

Owner:	Guirao, Carlos
Validated by PA/QA:	Verzichelli, Gianluca
Validated by WPM:	Kornweibel, Nick
Approved by PM:	Kornweibel, Nick
	Name

Authors

Name	Affiliation
C. Guirao	ESO

Change Record from previous Version

Affected Section(s)	Changes / Reason / Remarks
All	First release
All	Change E-ELT with ELT. Chapter-8

Contents

1. Introduction

1.1 Scope

SVN or Apache Subversion was declared the standard Version Control tool for the ESO ELT project (see RD4), therefore it applies to both ESO internal and development (at consortia or for industry contracts)

This document specifies several aspects of the SVN implementation for the ELT CS Development Environment like its directory structure; users; access control internally and externally; release management.

The document intends to be the SVN usage guidelines to be followed by all SVN users developing Control System software for the ELT project.

This text is a work in progress- highly subject to change.

1.2 Definitions and Conventions

1.2.1 Abbreviations and Acronyms

The following abbreviations and acronyms are used in this document:

TBC	To Be Confirmed
TBD	To Be Defined
PM	Project Manager
LCS	Local Control System
ADC	Atmospheric Dispersion Compensator
AO	Adaptive Optics
SVN	Apache Subversion (Software Versioning System)
DoE	Directorate of Engineering
CSE	Control Software & Engineering Department in DoE
CSE-ISI	CSE-Infrastructure SW & Integration Group
E-ELT	European Extremely Large Telescope
WP	Work Package
AIV	Assembly, Integration & Verification
TBC	To Be Confirmed
TBD	To Be Defined

2. Related Documents

2.1 Reference Documents

The following documents, of the exact version shown herein, are listed as background references only. They are not to be construed as a binding complement to the present document.

RD1 E-ELT Control System Software Development Process;
ESO-265043 Revision 1

RD2 E-ELT Product Breakdown Structure
ESO-23248 Revision 2

RD3 ESO-PM Framework
ESO-216010 Revision 1

RD4 E-ELT Control System Standards
ESO-193358 Revision 5 Chapter 8.2 Version control

2.2 Other Documents:

- Version Control with Subversion. Version 1.7. O'Reilly Media
- TortoiseSVN. A Subversion client for Windows. Version 1.9. Stefan Küng, Lübbe Onken, and Simon Large.

3. SVN server.

The SVN server is hosted at ESO Garching and maintained by ESO-IT department. It is accessible by external and internal software development teams under URL:

http://svnhq8.hq.eso.org/p8/

Access to the ELT SVN server is password protected and permitted only to registered users.

The server is maintained in a data center environment with a backup policy in place, however ESO makes no guarantees on its availability or reliability. Suppliers should therefore treat this service as a means of delivering interim and final software products to ESO, and not rely on this service as a part of backup/recovery or disaster recovery strategy.

The contents of the repository are NOT checked by a virus checker on upload or download, and this responsibility lies with the user.

ESO internet connectivity is likewise monitored and maintained however there are no guarantees made on the availability (access) to the SVN server.

4. Top-level directory structure

The ELT SVN repository is organized in three main top directories:

/branches

/tags

/trunk

The */trunk* top-level directory is where the main software development occurs.

The */tags* top-level directory contains the official releases of the programs maintained and developed in the ELT repository. Each individual module has a subdirectory here following the guidelines of chapter “Semantic Versioning”. Inside it, you will find the releases numbers. For instance the module */tags/…./FIFOInterface/3.4.0/* represents version 3.4.0 of module FIFOInterface.

The */branches* top-level directory contains the branch versions of modules after a major release.

5. Project layout

Under each top-level directory *(/branches, /tags and /trunk*) a fixed and identical directory structure is located. This depicts the ELT breakdown structure for software products as defined in AD2. The major subsystems of the ELT consist of:

AIV-PROTOTYPING-TESTING

CONTROL-SYSTEMS

DOME

INSTRUMENTS

SCIENCE-DATA

TELESCOPE

Each major subsystem is in turn divided in other subsystems until it ends up in individual work packages (WPs). The SVN project layout depicts the project layout as a directory tree (See Appendix A for a complete structure).

6. Roles and Responsibilities

6.1 Project Manager

Defines the SVN Project Layout, usage policy, appoints Software Work Package Managers and SVN administrators.

6.2 SW Work Package Manager

They decide the software modules (first directory level) under each Work Package. They are responsible to communicate to the SVN administrators the list of software developers allowed to access the Work Package level and their permissions (read-only or read-write)

6.3 SVN Administrators

Appointed by the Project Manager. They implement the project directory structure, register developers appointed by SW Work Package Managers, implement user access lists and set software hooks to enforce rules and general usage policy.

6.4 Software Developer

Appointed by SW Work Package Managers, they can commit code, changes, creates tags and branches. Software developers must be identified by their real names, email address and the software work package they are going to work for. They receive from SVN administrators a login account, with encrypted password of their choice.

7. SVN news and requests

A web document containing the most relevant information and latest updates regarding the ELT SVN usage guidelines is available at (TBC).

Together with each new registration users will receive from SVN administrators a copy of the latest usage guidelines or a link to the web page.

Any relevant information related to the general maintenance of the SVN, or changes that may interfere their activities will be communicated to affected users via their registered email addresses.

All requests to administrators regarding the ELT SVN repository will be done via the EELTMGR ticketing system. The ticketing system can be accessed via web (ESO internal use) or from outside ESO with an email to eeltmgr@eso.org and the text “SVN” as part of the Subject.

Typical requests from Project manager:

- Appoint/change Work Package managers (and deputies)
- Authorizes access to other directories outside the Work Package directory structure.

Typical requests from SW Work Package managers:

- Registration/removal of new/old accounts.
- Changes in the access list, e.g. authorizes read-write access to a developer.

Typical requests from Developers:

- Changes in the registration (user account, password)

In general:

- Comments, improvements and feedback

8. Groups and Access Lists

At the time of writing the SVN repository implements two different groups of users accessing the work package area:

- Users with read-write access
- Users with read-only access

It is the SW Work Package Manager who classifies software developers in these two lists. It is the SVN administrator who implements them on request from SW Work Package Manager.

9. Rules and Hooks

Tags can only be created but not removed.

List of file extensions considered software products thus ignored and not archived by SVN:

- .o
- .so

TBC

10. Semantic versioning

Whenever a software module is tagged, the versioning number MUST follow the schema: MAJOR.MINOR.PATH or X.Y.Z, where:

- MAJOR number (X) is also known as the main version number.
- MINOR number (Y) Also known as feature number. Increase this number if the change contains new features with or without bug fixes.
- PATCH number (Z): Also known as hotfix number. Increase this number if the change only contains bug fixes.

1. A normal version number MUST take the form X.Y.Z where X, Y, and Z are non-negative integers, and MUST NOT contain leading zeroes. X is the major version, Y is the minor version, and Z is the patch version. Each element MUST increase numerically. For instance: 1.9.0 -> 1.10.0 -> 1.11.0.
2. Once a versioned package has been released, the contents of that version MUST NOT be modified. Any modifications MUST be released as a new version.
3. Major version zero (0.y.z) is for initial development. Anything may change at any time. The public API should not be considered stable
4. Version 1.0.0 defines the first public API. The way in which the version number is incremented after this release is dependent on this public API and how it changes.

5. Patch version Z (x.y.Z | x > 0) MUST be incremented if only backwards compatible bug fixes are introduced. A bug fix is defined as an internal change that fixes incorrect behaviour.
6. Minor version Y (x.Y.z | x > 0) MUST be incremented if new, backwards compatible functionality is introduced to the public API. It MUST be incremented if any public API functionality is marked as deprecated. It MAY be incremented if substantial new functionality or improvements are introduced within the private code. It MAY include patch level changes. Patch version MUST be reset to 0 when minor version is incremented.
7. A pre-release version MAY be denoted by appending a hyphen and a series of dot separated identifiers immediately following the patch version. Identifiers MUST comprise only ASCII alphanumeric and hyphen [0-9A-Za-z-]. Identifiers MUST NOT be empty. Numeric identifiers MUST NOT include leading zeroes. Pre-release versions have a lower precedence than the associated normal version. A pre-release version indicates that the version is unstable and might not satisfy the intended compatibility requirements as denoted by its associated normal version. Examples: 1.0.0-alpha, 1.0.0-alpha.1, 1.0.0-0.3.7, 1.0.0-x.7.z.92.
8. Build metadata MAY be denoted by appending a plus sign and a series of dot separated identifiers immediately following the patch or pre-release version. Identifiers MUST comprise only ASCII alphanumeric and hyphen [0-9A-Za-z-]. Identifiers MUST NOT be empty. Build metadata SHOULD be ignored when determining version precedence. Thus two versions that differ only in the build metadata, have the same precedence. Examples: 1.0.0-alpha+001, 1.0.0+20130313144700, 1.0.0-beta+exp.sha.5114f85.
9. Precedence refers to how versions are compared to each other when ordered. Precedence MUST be calculated by separating the version into major, minor, patch and pre-release identifiers in that order (Build metadata does not figure into precedence). Precedence is determined by the first difference when comparing each of these identifiers from left to right as follows: Major, minor, and patch versions are always compared numerically. Example: 1.0.0 < 2.0.0 < 2.1.0 < 2.1.1. When major, minor, and patch are equal, a pre-release version has lower precedence than a normal version. Example: 1.0.0-alpha < 1.0.0. Precedence for two pre-release versions with the same major, minor, and patch version MUST be determined by comparing each dot separated identifier from left to right until a difference is found as follows: identifiers consisting of only digits are compared numerically and identifiers with letters or hyphens are compared lexically in ASCII sort order. Numeric identifiers always have lower precedence than non-numeric identifiers. A larger set of pre-release fields has a higher precedence than a smaller set, if all of the preceding identifiers are equal. Example: 1.0.0-alpha < 1.0.0-alpha.1 < 1.0.0-alpha.beta < 1.0.0-beta < 1.0.0-beta.2 < 1.0.0-beta.11 < 1.0.0-rc.1 < 1.0.0.

11. Good practices. Guidelines

1. **Think Twice before Committing**. Committing something may have serious consequences. All other developers will get your changes once they are in the SVN repository, and if they break something, they will break it for everybody. All commits will be publicly available in the SVN repository forever.

2. **Never commit code that doesn't compile.** Compile the code and correct all errors before committing. Make sure that newly added files are committed. If they are missing your local compile will work fine but everybody else won't be able to compile.
3. **Commit logical changesets.** When you commit a change to the repository, make sure your change reflects a single purpose: the fixing of a specific bug, the addition of a new feature, or some particular task. Your commit will create a new revision number which can forever be used as a "name" for the change. You can mention this revision number in the bug tracking system, or use it as an argument to svn merge should you want to undo the change or port it to another branch.
4. **Test your changes before committing.** Start the application affected by your change and make sure that the changed code behaves as desired.
5. **Always add descriptive log messages.** Log messages should be understandable to someone who sees only the log. They shouldn't depend on information outside the context of the commit. Try to put the log messages only to those files which are really affected by the change described in the log message. The official language for ESO ELT Software Development is **English**.
6. **Use bug tracking system numbers.** If you fix bugs reported on the bug tracking system, add the bug number to the log message. In order to keep the bug tracking system in sync with the SVN repository, you should reference the bug report in your commits, and close the fixed bugs in the bug tracking system. This doesn't mean that you don't need an understandable log message. It should be clear from the log message what has been changed without looking at the bug report.
7. **Don't add generated files to the repository.** Files generated at build time shouldn't be checked into the repository because this is redundant information and might cause conflicts. Only real source files should be in the SVN repository.
8. **Make use of SVN merging and revert.** SVN merging is something very powerful and gives you the possibility to keep history logs and relationships in modifications between branches, trunk and tags. SVN revert is also less error prone than manually removing code and committing the old one as new (See chapter *Basic Work Cycle* from book **Version control with Subversion** where merging is explained extensively).
9. **Don't add contractual or confidential files to the repository.** Sensible documents like contracts or any other confidential files do not belong to the software repository. No access list or restrictions will be implemented to protect directories or files falling in this category.
10. **Do not commit third party software protected by copy-rights or distribution policies.**

Appendix A. ELT SVN project structure

At the time of writing the current ELT SVN directory structure follows Work Package (WP) breakdown:

```
AIV-PROTOTYPING-TESTING

|-- AIV-ToolsAndEquipment
```

```
|-- Development
|    |-- CS-DevEnvModel
|    |-- CommonSoftware
|    |-- Templates
|-- PrototypesAndTesting
|    |-- ControlModel
|    |-- M1-TestBench
|    |-- M4-ProtoAndBreadboards
```

CONTROL-SYSTEMS

```
|-- CentralControlSystem
|    |-- CommMiddleware
|    |    |-- ApplicationFrameworks
|    |-- CoreIntegrationInfrastr
|    |-- HLCoordinationAndControl
|    |-- LocalSupervisors
|    |    |-- ADC
|    |    |-- CalibrationUnits
|    |    |-- Dome
|    |    |-- LaserGuideStarUnit
|    |    |-- M1-Unit
|    |    |-- M2-Unit
|    |    |-- M3-Unit
|    |    |-- M4-Unit
|    |    |-- M5-Unit
|    |    |-- MainStructure
```

```
|   |   |-- PrefocalStation
|-- InstrumentControlSystem
|-- NetworkInfrastructure
|-- TimeReferenceSystem
```

DOME

```
|-- DomeLCS
|   |-- LocalCommInfrastructure
|   |-- LocalControlUnit
|   |-- LocalSafetyUnit
INSTRUMENTS
|-- EltCamera
|-- EltIntegratedField
|-- EltMidir
|-- EltTestCamera
|-- Instrument-4
|-- LaserTomographAO
|-- MultiConjugateAO
```

SCIENCE-DATA

```
|-- DataFlow
|-- ScienceArchive
```

TELESCOPE

```
|-- ADC
|   |-- LCS
|-- CalibrationUnits
|   |-- LCS
```

```
|-- LaserGuideStarUnits
|   |-- LCS
|-- M1-Unit
|   |-- LCS
|-- M2-Unit
|   |-- LCS
|-- M3-Unit
|   |-- LCS
|-- M4-Unit
|   |-- LCS
|-- M5-Unit
|   |-- LCS
|-- MainStructure
|   |-- LCS
|-- PrefocalStations
|   |-- LCS
```

Appendix B: Examples

TBC