

Subversion Repository Layout

Table of contents

1 Proposal Draft Information.....	2
2 Overview.....	2
3 Process for Development and Review (of this document).....	2
4 Summary.....	3
5 Structure of Subversion Repositories.....	4
6 Managing Read-Permissions.....	7
6.1 Governing membership in UNIX groups.....	7
7 Managing Write-permissions.....	8
8 Managing Passwords in the Repositories.....	8
9 Automated Deployment of Applications.....	9
10 Public Repositories.....	9
11 Repositories to support Research and Teaching.....	9

1. Proposal Draft Information

DRAFT - PROPOSAL

Last modified: 3 May 2007

Adapted from the [CVS Repository at Berkeley](#) proposal draft of April 2005.

2. Overview

This document is a proposal for configuration and use of a Subversion instance as a repository of code and other files for the UC Berkeley campus. It is being developed by IST-ASAG and will be brought to a broader audience in IST and beyond for negotiation and implementation of an SVN instance that will be useful to the campus as a whole.

The structure and permissions scheme described in this document are intended to balance the following goals:

- Conformity to the [UC Berkeley Campus IT Security Policy](#)
- Sharing code among campus developers to facilitate IT staff development, best practices, and code re-use
- Adherence to intellectual property, licensure, and security-related requirements regarding source-code access and distribution
- Appropriate preservation of revenue-stream potential for campus IT units funded on a recharge model

3. Process for Development and Review (of this document)

This document is being developed and reviewed in the following stages. Review occurred in sequential order through the IST-AS Director, and is ongoing (though in a less-rigidly ordered process - that is, this document, budget documents, and a proposal laid out in the IST Transition Team format are circulating through a number of channels as of the date of this draft). Stages of review already completed are listed in *emphasized* text.

- *IST-ASAG staff*
- *IST-AS Web Applications Group (the initial users of this repository instance)*
- *IST-IS Unix Team (administrators of the repository host)*
- *IST Architecture Groups (IST-ASAG, IST-ISAG, IST-DSAG)*
- *IST-AS Director*
- *IST Directors (via budget process and/or formal proposal submitted through the Transition Team)*
- *Micronet*
- *ITAC*
- Additional IS&T units interested in using Subversion
- Additional campus units interested in using Subversion

4. Summary

This substance of this document is divided into several sections:

- [Directory Structure of Repositories](#)
- [Managing Read-Permissions](#)
- [Managing Write-permissions](#)
- [Managing Passwords in the Repositories](#)
- [Automated Deployment and Migration](#)
- [Public Repository](#)

The repository structure is designed to facilitate code reuse among UC Berkeley staff, without compromising appropriate organizational security.

Read permissions on the repository files are effected through controlled membership in [UNIX groups](#) on the server hosting Subversion. In general, read permissions are intended to be granted broadly on multi-project nodes of the repositories' tree. The initial service offering is intended for staff IT developers on the campus. Access to the repositories (via svn+ssh) will therefore be limited to staff and appropriate affiliates who have CalNet IDs; where possible, these IDs will be used as the account name on the host server. CalNet authentication will be implemented for host login in the near future.

Write (a.k.a. "commit" or "update") permissions are effected at any level of a repository's directory tree via management of a configuration file.

Code committed to the repository is expected to utilize appropriate procedures to either separate passwords from the codebase or very tightly restrict read-access to the codebase; the former is strongly preferred, and a set of model processes for effecting this goal are under development.

Automated deployment of applications is very strongly preferred to manual deployment, in

order to achieve reliable replication of deployment steps and to archive for the organization deployment intricacies that may otherwise exist only in the memories of its staff.

5. Structure of Subversion Repositories

In outline, the repositories will be implemented as follows:

- Projects (source code, documentation, released binaries, etc.) will reside in a projects directory that is subdivided into organization-specific directories:
 - the org-specific directories' read permissions will be regulated via UNIX groups with relatively broad (but controlled) membership (administrative sections of the tree will be governed by groups with narrower membership; while,
 - more finely grained restriction on read permission will be regulated via UNIX groups with more restricted membership, as described [below](#).
- Organization-files that are not source code, per se, may reside in organization-specific, non-project repositories should these be desired. Such files could include HTML source for an organization's website(s), administrative data for which version control and centralized backups are desirable, etc.
- Whether or not the organizational hierarchy would be best represented by a tree that echoes the [CalNetAD](#) forest's [OU Names](#) is a topic under consideration.
- Third-party software will reside in repositories created in a `third-party` directory, to which access will be broad (except as required by license or security restrictions).
- Data used for accessing secure resources (such as passwords) will be protected by storing it in a repository to which access is highly restricted.

The repository structure is a directory tree on the repository host's filesystem. The following representation of the proposed tree may clarify the preceding narrative description. Note that, in the truncated representation, the existence of a directory owned by `repoadm` and a `/hooks` directory implies a separate repository; each repository has separately configurable write (commit) permissions, as described in the [Write Permissions section](#), below. A more detailed inventory of directory permissions and group ownership in a sample repository is described in the [local Subversion Admin Tips page](#) on this site (see section on [Creating a new repository](#)).

Note that:

- Repository-level directories are indicated where the path ends with a backslash (\); whereas,
- projects within the indicated repositories are indicated by lines that end with elipsis (...).
- As explained in the [Subversion manual section on FSFS repository data stores](#), projects in a multi-project repository do not actually exist in separate directories.
- Directory-level permissions that secure the repository tree from access by users on the

host who are not Subversion users are not represented in this truncated example.

```
        drwxrwsr-x [...] userid  repoadm   [...]
/berkeley/projects/commons/
        drwxr-s--- [...] userid  svnusers  [...]
/berkeley/projects/commons/hooks
        drwxrws--- [...] userid  svnusers  [...]
/berkeley/projects/commons/java/sharable-project100/...
        drwxrws--- [...] userid  svnusers  [...]
/berkeley/projects/commons/java/sharable-project101/...
        drwxrws--- [...] userid  svnusers  [...]
/berkeley/projects/commons/dotnet/sharable-project200/...
        drwxrws--- [...] userid  svnusers  [...]
/berkeley/projects/commons/dotnet/sharable-project201/...
        drwxrws--- [...] userid  svnusers  [...]
/berkeley/projects/commons/ruby/sharable-project300/...
        drwxrws--- [...] userid  svnusers  [...]
/berkeley/projects/commons/ruby/sharable-project301/...
                                {insert locally-developed projects
                                appropriate to campuswide sharing
here}

        drwxrwsr-x [...] userid  repoadm   [...] /berkeley/projects/ist/
        drwxr-s--- [...] userid  ist       [...]
/berkeley/projects/ist/hooks/
        drwxrws--- [...] userid  ist       [...]
/berkeley/projects/ist/autodeployproject/...
        drwxrws--- [...] userid  ist       [...]
/berkeley/projects/ist/streak/...
        drwxrws--- [...] userid  ist       [...]
/berkeley/projects/ist/maven-repo-j5/...

        drwxrwsr-x [...] userid  repoadm   [...] /berkeley/ist/as/
        drwxr-s--- [...] userid  istas     [...] /berkeley/ist/as/hooks/
        drwxrws--- [...] userid  istas     [...]
/berkeley/ist/as/website/...
        drwxrws--- [...] userid  istas     [...]
/berkeley/ist/as/otheradminstuff/...

        drwxrwsr-x [...] userid  repoadm   [...]
/berkeley/projects/ist/as/arch/
        drwxr-s--- [...] userid  asag      [...]
/berkeley/projects/ist/as/arch/hooks/
        drwxrws--- [...] userid  asag      [...]
/berkeley/projects/ist/as/arch/project1/...
        drwxrws--- [...] userid  asag      [...]
/berkeley/projects/ist/as/arch/project2/...

        drwxrwsr-x [...] userid  repoadm   [...]
/berkeley/projects/ist/as/webapps/
        drwxr-s--- [...] userid  aswa     [...]
/berkeley/projects/ist/as/webapps/hooks/
```

```

        drwxrws--- [...] userid  aswa      [...]
/berkeley/projects/ist/as/webapps/project1/...
        drwxrws--- [...] userid  aswa      [...]
/berkeley/projects/ist/as/webapps/project2/...
        drwxrws--- [...] userid  aswa      [...]
/berkeley/projects/ist/as/webapps/project3/...
        drwxrws--- [...] userid  aswa      [...]
/berkeley/projects/ist/as/webapps/project4/...

        drwxrwsr-x [...] userid  repoadm   [...]
/berkeley/projects/ist/xyz/
        drwxr-s--- [...] userid  xyz       [...]
/berkeley/projects/ist/xyz/hooks/
        drwxrws--- [...] userid  xyz       [...]
/berkeley/projects/ist/xyz/project1/...
        drwxrws--- [...] userid  xyz       [...]
/berkeley/projects/ist/xyz/project2/...
                                {insert repositories for other IST,
such as group xyz, here}

        drwxrwsr-x [...] userid  repoadm   [...] /berkeley/projects/abc/
        drwxr-s--- [...] userid  abc       [...]
/berkeley/projects/abc/hooks/
        drwxrws--- [...] userid  abc       [...]
/berkeley/projects/abc/project1/...
        drwxrws--- [...] userid  abc       [...]
/berkeley/projects/abc/project2/...
                                {insert repositories for other Berkeley
organizations, such as group abc, here}

        drwxrwsr-x [...] userid  repoadm   [...]
/berkeley/projects/ist/prodcontrol/
        drwxr-s--- [...] userid  pctrl    [...]
/berkeley/projects/ist/prodcontrol/hooks/
        drwxrws--- [...] userid  pctrl    [...]
/berkeley/projects/ist/prodcontrol/pc-project1
        drwxrws--- [...] userid  pctrl    [...]
/berkeley/projects/ist/prodcontrol/pc-project2
        drwxrws--- [...] userid  pctrl    [...]
/berkeley/projects/ist/prodcontrol/pc-project3
        drwxrwsr-x [...] userid  repoadm   [...]
/berkeley/projects/ist/prodcontrol/streak-secure/
        drwxr-s--- [...] userid  pctrl    [...]
/berkeley/projects/ist/prodcontrol/streak-secure/hooks/
        drwxrws--- [...] userid  pctrl    [...]
/berkeley/projects/ist/prodcontrol/streak-secure/db-access/...
        drwxrws--- [...] userid  pctrl    [...]
/berkeley/projects/ist/prodcontrol/streak-secure/ldap-access/...

        drwxrwsr-x [...] userid  repoadm   [...] /third-party/jboss/
        drwxr-s--- [...] userid  svntp    [...]
/third-party/jboss/hooks/
        drwxrws--- [...] userid  svntp    [...]
/third-party/jboss/jboss-4.0.4/...

```

```
drwxrwsr-x [...] userid repoadm [...] /third-party/apache/  
drwxr-s--- [...] userid svntp [...]  
/third-party/apache/hooks/  
drwxrws--- [...] userid svntp [...]  
/third-party/apache/apache-forrest-0.7/...  
drwxrwsr-x [...] userid repoadm [...]  
/third-party/apache-jakarta/  
drwxr-s--- [...] userid svntp [...]  
/third-party/apache-jakarta/hooks/  
drwxrws--- [...] userid svntp [...]  
/third-party/apache-jakarta/jmeter-2.1.2/...  
                                {insert other third-party software  
repositories here}
```

6. Managing Read-Permissions

Read permissions are effected via ownership of repository-level directory trees by UNIX groups, and appropriately controlled membership in those groups.

Unless there is a business reason that necessitates more finely grained control, read-access to projects will be granted to a broad spectrum of users (e.g., to IST developers in the berkeley/projects/ist repository); and an even broader spectrum of developers in the third-party/ repositories. More finely grained control of read access to project repositories will be predicated on business reasons, such as any one or more of the following:

- restrictions on intellectual property contained in the repository's files;
- licensure issues that restrict access to or distribution of the repository's files;
- passwords in source code where it is impossible or it is deemed too resource-intensive an effort to implement a password model such as that described [below](#)
- other security-related issues that are impossible or that are deemed too resource-intensive to resolve except by restriction of read-access; or,
- revenue-stream potential in the repository, such that access to all repository users would materially compromise the value of that potential.

6.1. Governing membership in UNIX groups

Membership in UNIX groups, which governs read access to repositories, will require a coordinated and efficient mechanism for submission and approval; details must be negotiated with IST-IS as this proposal is implemented. Development and implementation of processes and mechanisms to effect appropriately approved assignment of UNIX group membership is a pre-condition for organizing the repository and broadening access to it. The following rules are expected to be implemented:

- Creation of new repositories, designation of the UNIX groups associated with those

repositories (and, implicitly, which projects belong in a given repository), as well as assignment of repository-admin-group membership must be approved by at least one of a small number of appropriate staff, e.g., a designated individual or her/his backup for each of the major IST units (AS, CS, DS, IS) using repositories in the `/berkeley/projects/ist` tree shown in the [example](#), above.

- Repository admins identified by the process described in the prior bullet-point must approve assignment of repository-specific UNIX group membership to users with logins on the host. For example:
 - any one of a small number of designated individual in each major IST unit must approve assignment of "svnist" group membership for projects in the svnist-owned repository or repositories;
 - an individual or individuals designated for a major IST unit must approve assignment of group membership specific to that unit (e.g., "svnas" or "snvis") for projects appropriate to share among staff in an organizational unit such as IST-AS or IST-IS;

7. Managing Write-permissions

Commit permissions are configurable via hook scripts and corresponding configuration files (cf. [Implementing Repository Hooks](#) in the SVN manual for more info). Commit permissions may be set on an entire repository and/or on any part of a directory tree, down to the level of individual files; these permissions are effected by editing the hook script configuration file, which is versioned in the repository itself; this is a task performed by repository administrators. Local instructions for maintaining pre-commit configuration files are available on this site's [Subversion Admin Tips](#) page [cf. section on [Modifying commit \(write\) permissions on projects](#)].

8. Managing Passwords in the Repositories

Without articulating a full definition of the precise model to be used, it is proposed that some method(s) will be employed to gather restricted password data (such as database passwords, LDAP bind passwords, etc.) in a highly restricted repository, where they will be stored in encrypted form. Access to this repository would be restricted to system administrators (who run deployments to QA and PROD servers), and a small set of IST staff who will be responsible for adding to and maintaining files in this repository (i.e., those who may appropriately have knowledge of QA and PROD passwords, such as production control staff).

The method(s) to be used will allow developers to supply passwords known to them - but NOT committed to repositories to which they have access - for testing purposes (e.g., in order to run an application from a development machine).

9. Automated Deployment of Applications

Automated deployments such as those of Java-platform applications developed by the former IST-SIS unit utilize scripts organized in the `hydrant` and `servant` projects currently archived in a CVS repository. These are [Apache-Ant](#) driven processes, and are instances of the type that are strongly preferred to unscripted (manual) compilation and deployment. Java platform projects are currently being migrated to builds using [Maven](#).

In general, the goal is to migrate/deploy only tagged, version-controlled artifacts using scripts that are themselves tagged and version-controlled.

The [Streek site](#) contains [Hydrant \(automated deployment\) documentation](#), and [Servant \(automated JBoss server generation\) documentation](#).

10. Public Repositories

Hosting of publicly accessible repositories is an identified need within IST, and may be or become a more widely desired service. Though outsourcing this need is a possibility (eg., [SourceForge](#)), it may be of interest or advantage to UC Berkeley to share open-sourced code from a Berkeley URL.

Read-only access may be made available via Apache `httpd` and the `mod_dav` module and the `mod_dav_svn` plugin to that module. Configuration is documented in [httpd section](#) of the [Server Configuration chapter in the Subversion Manual](#). Authentication and authorization - prerequisites to selective commit privileges off the UC Berkeley campus - is also possible and documented in the same section of the Subversion manual.

As of this document's last update, neither read-only nor authenticated/authorized access via Apache have been implemented on IST's Subversion server. Mention is made of these possibilities in this proposal in order to register interest that has been expressed in these services.

11. Repositories to support Research and Teaching

Interest has been expressed in a centrally hosted and managed Subversion repository service for research groups on campus and collaborative research that crosses campus boundaries; and for classes (e.g., courses offered in EECS). Discussion with potential clients of such services is ongoing, and lessons learned as IST develops and offers code repository service to staff IT developers are expected to inform the development of offerings oriented to the research and teaching missions of the campus.