Replacing the CDB with Subversion and Ant

C. Loomis
Laboratoire de l'Accélérator Linéaire (LAL), Orsay, France

April 25, 2005

Abstract

This document explains the benefits of using subversion and ant as an alternative to the standard Quattor Configuration Database (CDB). It lists the software necessary to do this, how to configure that software, and how to manage a site's configuration with this system.

Contents

1 Overview

The standard Quattor configuration database (CDB) is based on CVS with a custom front-end handling the interactions between CVS and the user. As an alternative (referred to below as SCDB), subversion can be used as the version control system with an ant build script defining standard configuration tasks.

This alternative has some advantages over the standard CDB implementation:

- Benefits directly from the advantages of subversion over CVS. (E.g. truly atomic commits, logically consistent branching and tagging, management of directories, integration with Apache web server, etc.)
- Exposes fully the code management features of subversion, making merging, conflict management, and tagging all evident for those accostumed to code management systems.
- Allows all changes to be verified and tested locally before committing and deploying those changes.
- Allows templates to be grouped into separate directories to facilitate management of those templates and to make it easier to identify changes in externally-supplied templates (e.g. Quattor templates, LCG templates, etc.).
- Allows site-specific changes to be isolated from standard templates to allow those changes to be easily identified.
- Allows multiple clusters to share the bulk of the templates but still make cluster-specific changes to some of the templates.
- Integrates well with existing, graphical code development systems, notably eclipse.
- Allows an alternative XML format to be used which integrates well with XML databases and the XPath/XQuery standards.

However, there are some disadvantages as well:

- It is not supported by the Quattor developers (but is supported).
- Requires subversion server and client software which is not yet a standard part of many operating system distributions.
- Requires java 1.5 which is not available on all platforms.

LAL has a desire to unify the management system for it's grid and non-grid resources. Because of this we need the capability to manage several "clusters" simulataneously. Because of this need and a pre-existing subversion infrastructure at the laboratory, LAL developed and uses this alternative configuration database.

2 External Software Needed

The SCDB alternative requires various external software packages. The needed packages are:

subversion A subversion server is needed to hold and control the configuration information. A subversion client is needed on the machine(s) from which the configuration will be managed. The source as well as links to binary packages are available from the subversion web site[1].

apache The subversion server runs within an Apache2 server. The code is available from the Apache Foundation[2] but is also included in most recent operating system distributions. Subversion requires an Apache2-series release; older version 1 releases will *not* work.

java The ant framework is written in java and consequently the quattor ant tasks are as well. A java virtual machine can be obtained from Sun[3] for Windows, Linux, and Solaris systems. For other operating systems, you need to contact your vendor. The quattor tasks *require* java 1.5.

panc To allow the local building and testing of machine profiles, the Quattor pan compiler (**panc**) is required. This is written in standard C++ with minimal dependencies. However, it has only been tested on unix-like systems.

ant The various configuration management tasks are controlled via an ant build script. Ant version 1.6.2 or later is required. It can be obtained from the Apache Foundation[4].

eclipse Eclipse is optional, but provides a very good front-end for graphical management of a site's configuration. See the eclipse web site[5] to download the version for your platform.

Ant does not need to be installed separately as it is included in the example distribution you will check out to seed your subversion module.

3 How SCDB Works

The configuration information for a site is managed via a standard subversion module which has a particular structure and a custom post-commit hook script. The module acts as a database for the site's configuration and additionally keeps a complete history of the changes.

Once setup, the configuration is checked-out from the server, modified locally, tested locally, and then committed back to the server. The checkout and commits are done with standard subversion commands. The modifications are done with your favorite editor. The compilation and testing of the configuration are done via pre-defined ant tasks.

[1] http://subversion.tigris.org/
[2] http://www.apache.org/
[3] http://java.sun.com/
[4] http://ant.apache.org
[5] http://eclipse.org/

Committed changes are visible to other administrators, but are *not* automatically deployed to the clients. This is to allow large changes to be tested and committed incrementally. Frequent commits avoid large and difficult conflict resolution.

To deploy changes to clients, the current `trunk` must be copied (tagged) to the `deploy` branch of the module. There is an ant task to perform this tag and to ensure that the tag has the correct name. The tag's name is a formatted version of the current date and time. This serves as record of all configurations deployed on a site. Separate named tags can be made directly with the subversion commands, if desired.

Tags with the proper date/time format to the `deploy` subdirectory trigger the subversion post-commit script to deploy the new configuration to the client machines.

The post-commit hook script does the following:

1. Creates or updates a workspace on the subversion server with the revision justed tagged.
2. Does a complete build of the configuration profiles.
3. For successful builds, it copies the generated profiles to the a web location accessible to the clients and notifies all of the clients that new profiles are available.
4. Sends an email to a predetermined list logging the action and telling whether the build was successful or not.

The script currently does this in the foreground, so the client which did the **ant deploy** command waits until the above procedure is finished.

Once the clients are notified, they pull the new machine profile and affect any necessary changes on the client. This is done entirely with the standard Quattor software.

4 Setup

4.1 Server Configuration

Subversion provides an excellent book[6] which describes both the use and administration of a subversion server. Please refer to the subversion book for instructions on setting up a server. Once the server is running, create a module using **svnadmin** to hold your site's configuration. Both the server and module are completely standard.

Download the prototype module from the QWG server. With a web browser go to the URL:

`http://svn.lal.in2p3.fr/WebSVN/QWG/SCDB/?rev=0&sc=0`

and click on the "tarball" link opposite to the "SCDB" to create and download a tarball of the directory. Unpack the tarball into a temporary area.

Check out your newly created module.

[6] http://svnbook.red-bean.com/

```
svn co <your SVN module URL> scdb
```

Copy the unpacked contents of the tarball (below the root of the tarball) into your checked out working copy of the repository. From the top-level of your working copy do the following:

```
svn add *
svn commit -m "Initial structure"
```

This provides you with the build file (`build.xml`), some of the required external software, and the particular structure used by the build script. You must also set the ignore property on the trunk directory:

```
cat > ignore.txt <<EOF
build
deploy
EOF
svn propset svn:ignore trunk -F ignore.txt
rm ignore.txt
svn commit -m "Ignore generated files"
```

This will ignore generated files. If this is not done, the deployment will usually fail because the build script checks for local modifications before deploying changes.

In the `external/scripts` subdirectory there is an example subversion post-commit hook script. You should modify this script to correspond to the values for your site. The important parameters are the notification email list and the directory on the server to use as a cache.

You should ensure that the additional, external software is installed on your client: a subversion client and the pan compiler.

You can modify any of the parameters of the build script by putting a `build.properties` file at the top-level of the cache on the server. For example, to modify the location of the pan compiler on the server and the deployment directory for the machine profiles, put the following into the `build.properties` file:

```
pan.compiler=/usr/local/pan/panc
deploy.xml=/var/www/html/profiles
```

Another property you might want to modify is "pan.xml.format" which by default is set to "xmldb". This format is appropriate for using the machine profiles in an XML database, but you might want the default panc format for some reason. To change the format set this value to "pan".

4.2 Client Configuration

The client configuration consists mainly of installing the necessary external software. The packages necessary are a subversion client, the pan compiler, and java. Once these are available, check out your module. The files you copied into your module contain an example machine profile and some example templates. This should allow you to test the targets described in the next section.

NOTE: This has only been tested on a linux client. In principle, any client can be used where the external dependencies are satisfied. If you have success with another client, please send an email with the details.

5 Targets

Except for making and committing changes to the templates, other common tasks are encoded as ant targets. The primary targets are:

all This is the default target and rebuilds the machine profiles if any changes have been made.

all.force This will rebuild the machine profiles from scratch. This just invokes "clean" followed by "all".

check.syntax Check the syntax of all of the pan templates. The profiles will not be recreated if there are any templates with syntax errors.

clean Remove the generated files in the "build" subdirectory.

clean.all This removes all generated files including those in the deployment area.

compile.profiles Compile the machine profiles. This is the target invoked by "all".

deploy This tags the current version in trunk by copying that version to the "deploy" subdirectory. This triggers the server to rebuild the machine profiles on the server and to notify the clients of the change. This target will *not* succeed if there are any uncommitted local modifications.

update.rep.templates Parses the repository templates for the servers' URLs, contacts the server, and updates the list of available packages. This modifies the local templates only; changes must be committed manually.

There are a few internal or rarely-used targets which are normally hidden. Type:

```
ant -verbose -projecthelp
```

to get a full list of all defined targets.

6 Structure of the Module

The subversion module has the following subdirectories:

cfg Contains all of the configuration templates.

cfg/shared Templates intended to be shared by all machines.

cfg/clusters Templates specific to individual clusters.

external Contains all of the external software (mainly ant tasks) required by the build script. There is also the required version of ant installed here.

src Contains prototype hook scripts for the server.

When running the build scripts, the following directories are created:

build Various intermediate files created in the build process. Also contains the local copy final generated templates.

deploy A deployment directory created when the internal **deploy.and.notify** task is run. Normally this is only run on the server, but can be run locally for tests. Note: the clients are notified by this task, but unless there are actually new deployed profiles, it will have no effect.

7 Typical Tasks

This section describes some of the typical tasks and the how to accomplish them with SCDB.

7.1 Change a template for all machines

The templates under the `cfg/shared` directory are shared by all of the machines in all of the clusters. If you want a change to affect all of the machines, make the change to the templates in this area. Steps are:

1. Edit and save the template to change.
2. Run **ant** to rebuild the local files.
3. Debug and verify the changes to the local machine profiles.
4. Commit changes with **svn commit**.
5. Deploy changes to machines with **ant deploy**.

The deploy task requires that the local build is successful; this is to minimize the chance of bad changes being deployed to running machines. The **deploy** target only needs to be called when you want to deploy changes; you can accumulate changes in the repository until you are ready to deploy them all.

7.2 Change a template for only one cluster

The templates within the `cluster` subdirectory contain the templates specific to a particular cluster. This includes the top-level object profiles, but also any templates specific for that cluster or any templates modified for that cluster.

To modify the template for only one cluster:

1. Copy the template to modify from the `shared` area to the `clusters/NAME` area where NAME is the name of your cluster. You do not need to keep the structure under the `shared` area, but it is good practice to do so.
2. Run **ant** to rebuild the local files.
3. Debug and verify the changes to the local machine profiles.
4. Commit changes with **svn commit**.
5. Deploy changes to machines with **ant deploy**.

Except for the location of the template this is the same procedure as for global modifications.

This override mechanism is an excellent way to isolate changes and is useful even if your site has only one cluster. It can be used, for example, to keep

local modifications to the LCG templates separate from the standard templates distributed with a release. This makes it easier to identify and reapply changes when a new release appears.

7.3 Add a machine to a cluster

To add a machine to a cluster simply put the machine profile in the appropriate "clusters/NAME/profiles" directory where "NAME" is the name of your cluster. Rebuild the local profiles, then commit and deploy the change as appropriate.

7.4 Add a cluster

To add a new cluster, just create a new subdirectory below the "clusters" directory. Add a "profiles" subdirectory and populate it with the machine profiles. Copy and modify any templates specific to this cluster. Rebuild the local profiles, then commit and deploy the changes as appropriate.

7.5 Update a configuration component

Configuration components are often updated to extend the functionality or to correct bugs. To deploy a new component to the clients do the following:

1. Build the package containing the new configuration component.
2. Copy this package into your software repository.
3. Make the package locally and copy the component templates into the `shared/components` subdirectory (assuming this is a global change).
4. Remake the repository templates with the **ant update.rep.templates** command.
5. Rebuild and verify the changes locally.
6. Commit and deploy the changes.

Updating non-configuration component packages is similar.

7.6 Reverting to Previous Versions

To revert to previous tagged versions or to undo changes, simply use the standard subversion commands for merging changes into the `trunk` copy. After the changes, just build, commit, and deploy as usual.

8 Using Eclipse

To be completed.

9 Feedback and Support

Using subversion and ant as alternative for the Quattor configuration database (CDB) has many benefits and is being effectively used to manage LAL's computing intrastructure.

This alternative, however, is not officially supported by the Quattor developers nor by the Quattor Working Group. It is supported by the developer (charles.loomis@cern.ch) who will respond to questions, bug reports, and requests for enhancement. Your constructive feedback is welcome.